Guest Commentary:
Isaac Asimov
And The Three Laws

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

Interview With
Kit Spencer:
Commodore's
New Marketing VP

COMPUTE!

$2.50
November, 1981
Issue 18
Vol. 3, No.
63379

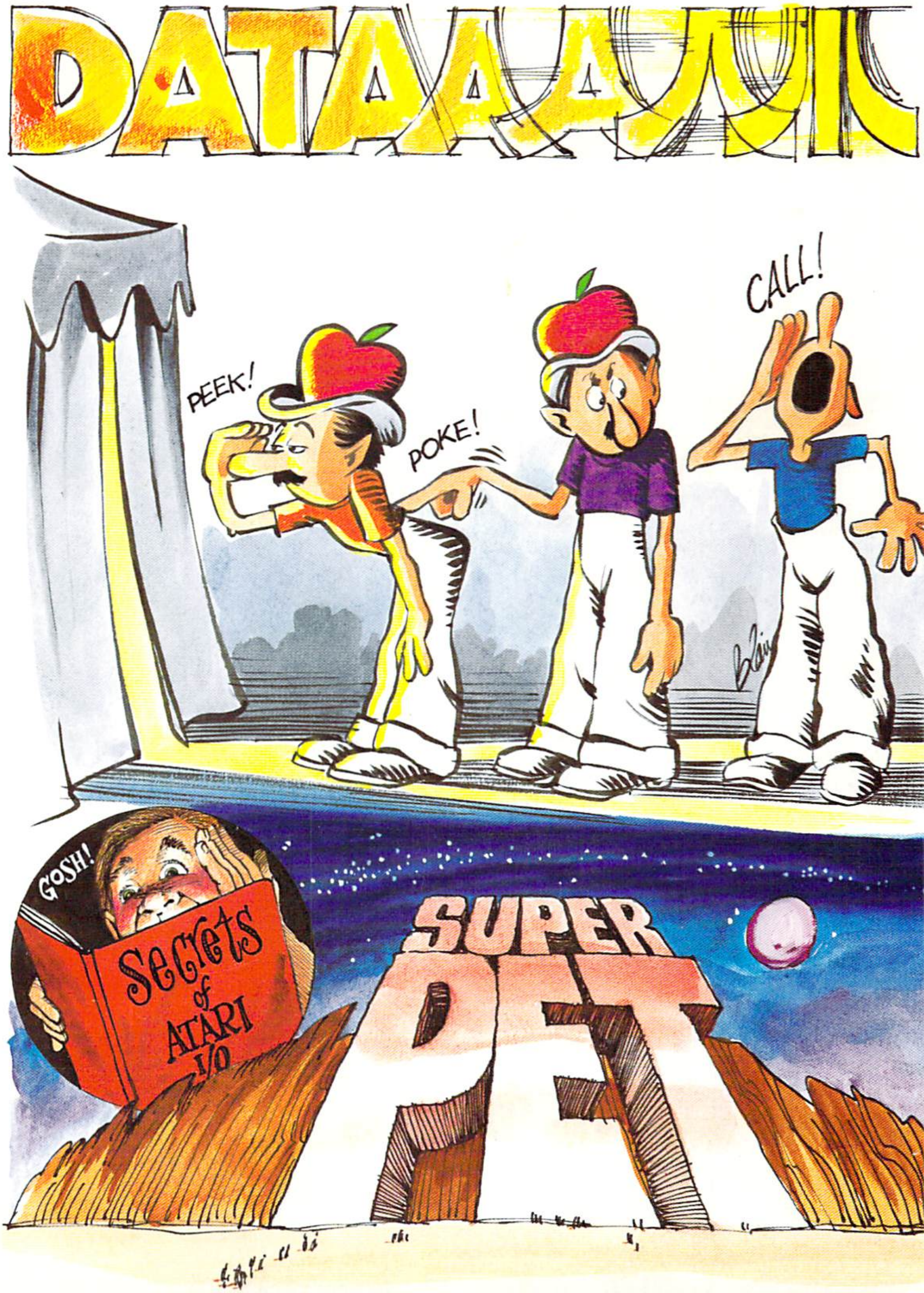
The Journal For Progressive Computing™

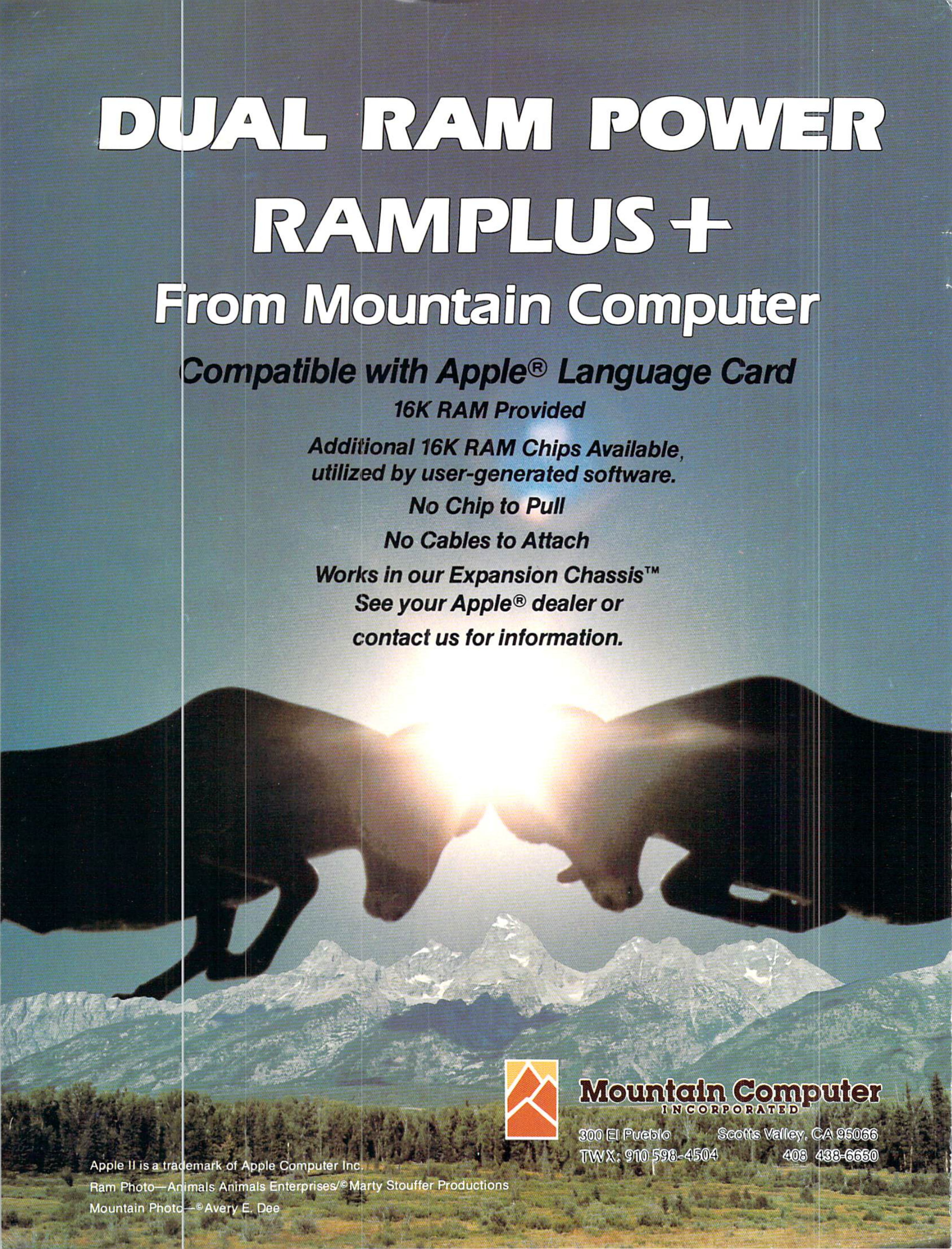
An Atari Data Management Program

An Apple Primer:
Peeks, Pokes, And Calls

Commodore's Super PET Revealed

Insight: Atari
Exclusive I/O Secrets
First of Two Parts




DUAL RAM POWER
RAMPLUS+
From Mountain Computer
Compatible with Apple® Language Card
16K RAM Provided
Additional 16K RAM Chips Available, utilized by user-generated software.
No Chip to Pull
No Cables to Attach
Works in our Expansion Chassis™
See your Apple® dealer or contact us for information.
Mountain Computer
INCORPORATED
300 El Pueblo
Scotts Valley, CA 95066
TWX: 910 598-4504
408 438-6650
Apple II is a trademark of Apple Computer Inc.
Ram Photo—Animals Animals Enterprises/©Marty Stouffer Productions
Mountain Photo—©Avery E. Dee

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

TM WordPro is a Registered Trademark of Professional Software, Inc. WordPro was written by Steve Punter.

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

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

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

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

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

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

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

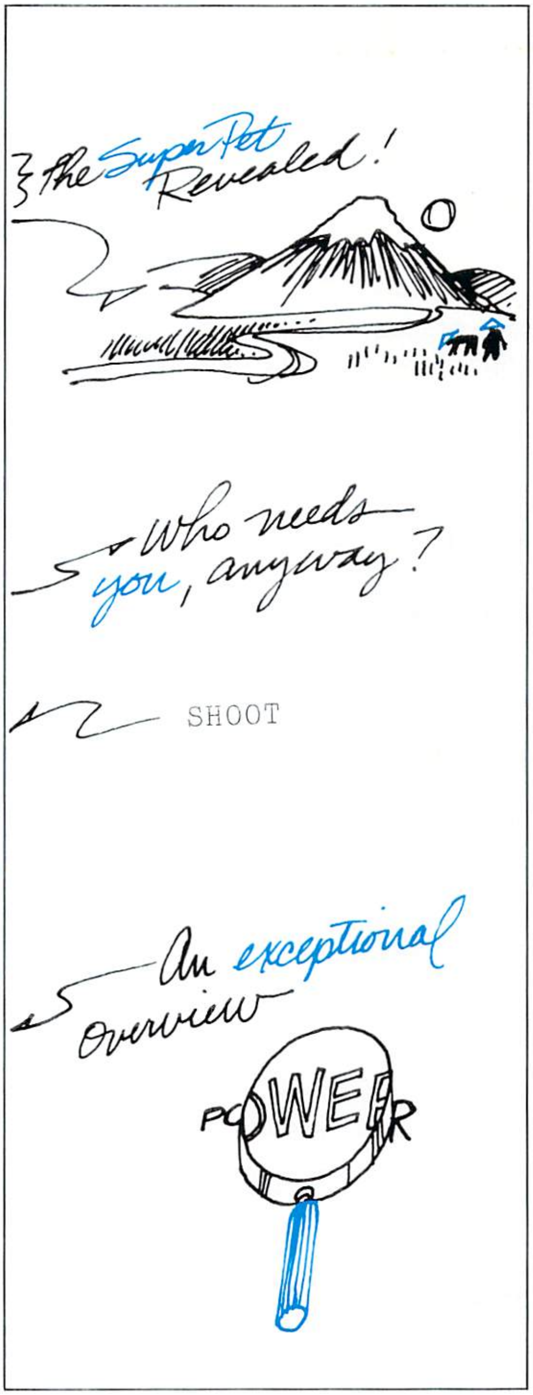
Complete information? It's at your dealer's now. Or, we'll send it to you and include a dealer list. Write us. Call us.

MICROSOFT
CONSUMER PRODUCTS

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

Table of Contents

November, 1981, Vol. 3, No.



COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published 12 times each year by Small System Services, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 625 Fulton Street, Greensboro, NC 27403.

Domestic Subscriptions: 12 issues, $20.00. Send subscription orders or change of address (P.O. Form 3579) to Circulation Dept., **COMPUTE!** Magazine, 515 Abbott Drive, Broomall, PA 19008. Controlled circulation postage paid at Greensboro, NC 27403. Application to mail at controlled circulation rates pending at Hickory, NC 28601. ISSN 0194-357X.

The Editor's notes...

Robert C. Lock, Publisher/Editor

Atari Educational Sales Revisited

Last issue we mentioned Atari's aggressive pricing moves at the educational "state contract" level. In this context, we mentioned that Atari, Inc. had obtained the new state contract for Minnesota. A recent newsletter from the Minnesota Educational Consortium clarified the current state of the contract. Atari has been added as a vendor as we mentioned last time. Apple, Inc. is still part of the contract as well.

In This Issue

Our extensive articles on Commodore's new "Super Pet" (written by its developers), are fascinating reading. Regardless of your interest, I recommend them as an insight into a new generation of computing.

Bill Wilkinsons' column, *Insight: Atari*, begins a two-part exploration of input/output functions. The information presented is excellent...and unique.

Recreational Computing Magazine Joins COMPUTE! Family

I'm quite pleased to announce that we have merged *Recreational Computing* magazine into our family of publications. Former RC subscribers will now be receiving **COMPUTE!** or *Home And Educational COMPUTING!*. This merger gives additional breadth and depth to our magazines, and an added sense of pride, since RC was the oldest of the personal computing magazines. As always, we welcome the growth!

Home and Educational COMPUTING! Expands

We've made the decision to broaden the base of *Home And Educational COMPUTING!*, in part because of the acquisition of *Recreational Computing*. Beginning with the January/February Issue, *Home and Educational COMPUTING!* will expand editorial coverage to include most of the personal and educational computers selling for $500 or less. We'll provide the same excellent resource and applications information we've established **COMPUTE!** with, to owners and users of the Commodore *VIC-20*, the Radio Shack *Color Computer*, the Texas Instruments *99/4A*, The Atari *400*, and others as well.

Present plans include ongoing columns like *Friends Of The Turtle* by David Thornburg, Ramon Zamora's *Rainbow Machine* column, telecommunications, educational applications and uses, and a great deal more.

Northeast Computer Show

We recently attended the Northeast Computer Show, and were impressed by the growth of the show. Coincidentally, that's the show where **COMPUTE!** was first introduced two years ago. That was the show's first run...we (the collective exhibitors) filled one hall. This year the show filled two downstairs halls, and most of one upstairs hall.

The atmosphere was festive, with Commodore giving away (by drawing) a VIC-20 every day, talking cars, numerous "robots," and an Atari booth that covered an entire stage at one end of a downstairs hall. An interesting change reflecting our mutual growth is that we're seeing less and less of the national firms at these shows, and more and more regional distributors and marketing organizations. Atari and Commodore do continue to bring in corporate level support. Apple, on the other hand, was in evidence through local dealers.

Atari User-Group Drive

I had the pleasure of spending some time with Earl Rice, Atari's new User Group Support Manager. He was showing off an excellent video teaching-tape produced by him and Chris Crawford of Atari. The tape is the first of a planned series which will eventually be made available to user groups and others on a loaner basis. The sketches we saw were not only quite humorous, but excellent and informative. We'll try to keep you posted on availability.

By the way, if you're involved with an Atari User Group, or interested in starting one up, contact Earl at Atari, Inc. He's working hard to set up two-way communications and support. His address is:

Earl Rice, Manager
User Group Support Program
Computer Division
Atari, Inc.
P.O. Box 427
Sunnyvale, CA 94086

"Teaching" Software

Several vendors were displaying well-conceived, well-structured software designed for youngsters. It's really a pleasure to see vendors utilizing the features of the computers they support.

For example, "Sammy The Sea Serpent" from Program Design, Inc.: with a joystick, an eight year old, and good graphics, first glance seemed to indicate another good "game" of simple quality. By using the voice-over capability of the Atari, however, this program is transformed into a talking/learning story.

We watched an absolute novice eight year old work through a set of exercises of ever-increasing complexity, cleverly couched in a storybook setting. We were all entranced by the narrative (and rather pleased when Sammy escaped to the sea!).

John Victor of PDI is one of the few vendors treating the voice capability of the Atari as an extra dimension, and the merits of that treatment were obvious in the resulting software. ©

THE GRAPHIC DIFFERENCE

BETWEEN ATARI® COMPUTERS AND ALL OTHERS.

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

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

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

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

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

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

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

*ATARI 800™ computer only.

ATARI®

Computers for people.™

A Warner Communications Company

Subscription Information
(12 Issue Year):
COMPUTE! Circulation Dept.
P.O. Box 5406
Greensboro, NC 27403 USA
U.S. $20.00
Canada $25.00 (U.S. funds)
Europe: Surface Subscription, $25.00 (U.S. funds)

TOLL FREE Subscription Order Line
800-345-8112
In PA 800-662-2444

Canadian Retail Dealers should contact:
Micron Distributing
409 Queen Street West
Toronto, Ontario M5V 2A5
416-361-0690

Advertising Sales

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

Phoebe Thompson and Associates
101 Church Street
Suite 13
Los Gatos, CA 95030
408-354-5553
PHOEBE THOMPSON

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

New England, New York State:
The Gittleman Company
Statler Office Building
Suite 582
20 Providence Street
Boston, MA 02110
617-451-0822
JOAN DONAHUE

The Gittleman Company
Summit Office Centre
7622 Summit Avenue
Fort Washington, PA 19034
215-646-5700
DOUG JOHNSON

New York City Metro Area, Mid-Atlantic and Southeastern States:
Local Numbers:
New York 212-567-6717
Atlanta 404-523-1252

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

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

COMPUTE! Home Office Contacts
Advertising Sales Manager, Harry Blair
Advertising/Production Coordinator, Alice S. Wolfe

Address all advertising materials to:
625 Fulton Street
Greensboro, NC 27403 USA

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

Telephone: 919-275-9809

Small System Services, Inc. publishes:

COMPUTE! The Journal For Progressive Computing

Home and Educational COMPUTING!

COMPUTE! Books

Robert C. Lock, President
W. Jerry Day, Corporate Comptroller
Kathleen Martinek, Assistant To The President
Ellen Day, Accounting Manager
Carol Holmquist Lock, Research Assistant

Corporate offices are located at 625 Fulton Street, Greensboro, NC 27403 USA

COMPUTE! Subscription Rates

US	$20 (one yr.)
	$36 (two yrs.)
	$54 (three yrs.)
Foreign Surface Mail	$25
Air	
Europe, US Possessions	$38
Middle East, Central America and North Africa	$48
South America, South Africa, Far East and Australia	$88

Coming In The December Issue of COMPUTE!:

Adding High Speed Vertical Positioning To Atari Player Missile Graphics

File Recovery On The Commodore 8050 And 2040 Disk Drives

Animating Applesoft BASIC Graphics

Speeding Up Your Atari Repeat Key

A Commodore Super PET Update

Computers And Society

David D. Thornburg
Innovision
Los Altos, CA

Interfaces And Languages For The Mass-Market Micro

Last month we explored some ideas about what it will take to get personal computers into the true mass market. Clearly, extreme simplicity of operation will be a major factor for the acceptance of this product by its millions of potential users. Properly designed software can allow the computer to be used for many things without requiring that the user be a proficient touch typist.

Any true "mass-market" computer which, for example, lets you connect to a remote data will *not* require the extensive time consuming log-on procedures in use today. It is easy to see why this must be so. Consider the effort it takes for a typical data-base user to connect to an information utility. First, a telephone number must be dialed (seven keystrokes, minimum). Second, a log-on procedure must be carried out (typically 24 keystrokes). Third, the data base access commands must be entered (anywhere from 10 to 50 characters or more). To read one's mail on Source Telecomputing requires 41 keystrokes, minimum.

While you or I might be willing to go to all this effort in exchange for the tremendous power these data bases provide, I find it hard to believe that such lengthy procedures will be acceptable to the non-technical user who presently gets access to the evening news by simply pressing a button on the TV remote control while seated comfortably in the living room armchair.

The home terminal enviroment of the future will most likely offer menu-driven access to data bases (mail, stocks, news, sports, theater tickets, etc.). A simple joystick or light pen will be used to move the cursor to the desired selection. Once the choice has been made, the *computer* will then carry out the lengthy procedure of dialing the remote host and logging on to the system. The use of non-volatile memory (such as battery powered RAM, or bubble memory) will allow the user to establish the log-on procedure once. After that, this data will remain available in the system until altered by the user.

If a full keyboard isn't required for menu selection tasks, it certainly will be needed by users who want to generate electronic messages for others, or who want to generate their own programs. It is my guess that the home computer of the future will have a full keyboard as an option. Users who are primarily information *receivers* will be able to use simple pointing devices and menu-driven software. People who are also information *providers* will want the flexibility inherent in a full alphanumeric keyboard.

If we believe this scenario, we can then speculate on the shape that such a device might take. The mass-market computer may very likely resemble today's programmable video games more than it resembles today's computers. Imagine a video game with a disk drive and a telephone link and you might not be far from the mark. Many of the popular video games contain complete microcomputers inside them (the Atari VCS uses the 6502, and the Mattel Intellevision uses a 16-bit processor and support chips from General Instruments). The 8-bits of parallel interface needed to support two joysticks can also support a keyboard quite nicely. I have even heard of someone who is selling a plug-in cartridge for the Atari video game which lets you write your own assembly language programs for it. That's something to think about, isn't it!

Our scenario of the true mass-market computer is not complete, of course. The "video game" model presented above carries with it the idea that none of the software will be user generated. Imagine, the next time you walk in a record store, what that store will look like with racks of software. Instead of Country, Jazz, and Classical labels, you might see Business, Games, and Home Management labels instead. There will probably even be "cut-out" bins for software that has peaked in popularity!

While many of the future users of computers will be more than content to purchase their software off the shelf, there will be quite a few users who will want to generate their own programs. Just as the home computer of the future may be different from the machines we use today, the languages used by these people most certainly will be.

Most readers of **COMPUTE!** probably know how to write programs in one of several dialects of BASIC. I would guess that, after BASIC, the popular languages would be Assembler, Pascal, and Forth. Given these choices, the casual home user would gladly embrace BASIC.

And yet BASIC is far from being a user friendly language. For example, the fact that program branching commands go to line numbers rather than labels can make it hard to trace program flow in this language. Nonetheless, BASIC has proven quite useful, and has allowed many millions of people to create programs which might not otherwise have been created.

The problem with introducing BASIC into the true home market is that it is perhaps too

"mathematical," and therefore too intimidating for someone who wants to gently ease into writing his or her own programs. There are other languages which are more user friendly — notably PILOT and LOGO. It is not mere coincidence that Atari's and Texas Instruments' push into the mass market coincided with their respective introductions of Atari PILOT and TI LOGO.

One of the most appealing aspects of these languages (from the viewpoint of the casual user) is the ease with which small procedures can be created and tested, and then used as building blocks in larger procedures and programs. It is as if the user were able to create extensions to the language, thus personalizing it.

I have written a real-time game in Atari PILOT which is (for me) quite large — over 20 thousand bytes. The main program is less than 30 statements long, and most of the procedures used by this program and by other procedures are 20 to 40 statements long. By building the program out of a great many small modules, many of which use each other from time to time, I was able to create a very complex program and debug it quite rapidly. When a problem was uncovered in an output routine, I was able to fix it and test it without tampering with any other part of the program. One result of this high level of modularity is that PILOT is kept quite busy keeping track of the pointers which show where the modules should return when they are completed. Some portions of the program involve up to six nested modules at any given time. The fact that this program was easy to read after it was written is its greatest asset to me — especially since the program's speed of execution was not unduly compromised by these nested procedure calls.

Even though I write in BASIC almost every week, I would never have tackled this project in that language. I have heard that people who write in LOGO feel the same way.

Are user friendly languages enough to entice the mass market into programming? They will probably capture the interest of many people, but my guess is that the typical home user wants something even simpler.

Recall that it was Visicalc which was responsible for the tremendous acceptance of personal computers by business users. Visicalc sits on the fine line between a language and a program. The user creates a "mask" which contains all the personalized information associated with the spread sheet. Once this is created, the data is then entered much as it would be in a fully "canned" program. The two tasks of the user (creating the mask, and entering the data) are separable. Since the user both *creates* and *uses* the spread sheet, he or she plays the role of both programmer and user, without having to learn about data types, loop structures, recursion, and the like.

Perhaps there will be a new class of languages for the home market which are generalized "task translators." By responding to displayed prompts, the user conveys information to the computer which it can then use in creating a program which is tailored exactly to the user's needs. I have experimented with a few such programs and find their promise to be quite exciting. There is, of course, a great deal of difference between a spread sheet program like Visicalc and a Generalized program writer which could handle anything from video games to personal finance, but I would not be surprised to see our concept of computer languages undergo a radical change in the next five years.

The power contained in the smallest personal computer today can be put to tremendous benefit to all who would use this technology. The key to the acceptance of this technology in the consumer marketplace will be the software tools which allow the user to unleash and mold this power to fill personal needs.

Once this happens, we will have entered a new era. ©

WORDS! WORDS! WERDS?

Oh NO!, another misspelled word. Did you catch it?

WORDCHECK DID

WordCheck can find those minor errors in even the longest WordPro text file. This program was not designed to eliminate proofreading, but to be used as a utility program. The most commonly misspelled words are the ones we use all the time.

WordCheck is capable of identifying 7 to 10,000 words and will support multiple dictionaries for specialized applications such as; medical, legal, or scientific. A standard dictionary is included that can be modified at any time by the user, or duplicated to create additional dictionaries.

WordCheck lays no claims to **"FLASH AND SIZZLE"**

Just a major claim on **"WORDS"**!

Call now for your claim on **"WORDCHECK"**!

CREATE-A-BASE WITH SUPER SCAN/EDIT

CREATE-A-BASE, the ideal data management system, has added a touch of class with its new **SUPER SCAN/EDIT**. No other program gives the user such ease of operation and **Create-A-Base** still has all the other features for which you asked: interactive with WordPro, mathematic functions, sort 650 records in 19 seconds, specialized reports, merging, transferring, and duplicating files with a few easy commands.

The **Super Scan/Edit** puts the operator in control. The Scan can locate an eleven character match anywhere in a record scanning 10, 24 field records a second. Cut the fields to 8 and it will scan 21.8 records a second. Speed is of the essence, with the located record on the screen you have full Editing functions. Never again will you have to rewrite an entire line, just cursor over, make the change, add, delete, or rewrite the record with the touch of a key.

CREATE-A-BASE is here with **SUPER SCAN/EDIT**, don't miss it!

AVAILABLE

at your local COMMODORE dealer or distributed exclusively in CANADA by
B.P.I. Micro Systems, Ltd.
80 Barbados Blvd. #14
Scarborough, Ontario M1J1K9
Special Dealer Introductory Package Available

Micro Computer Industries Ltd.

1520 E. Mulberry, Suite 170 Fort Collins, CO 80524

1-303-221-1955

NEECO

WHY BUY FROM THE BEST?

Service... Support... Software...

commodore

16K B (16K RAM-40 Column) - Lim. Qty	$ 995
32K B (32K RAM-40 Clm.) - Lim. Qty	$1295
4016 (16K RAM 4.0 Basic-40 Clm.)	$ 995
4032 (32K RAM 4.0 Basic-40 Clm.)	$1295
8032 (32K RAM 4.0 Basic-80 Clm.)	$1495
8050 Dual Disk (1 Meg Storage)	$1795
4040 Dual Disk (343K Storage)	$1295
8010 IEEE Modem	$ 280
C2N Cassette Drive	$ 75
CBM - IEEE Interface Cable	$ 40
IEEE - IEEE Interface Cable	$ 50
VIC 20 Home/Personal Computer	$ 295

ALTOS

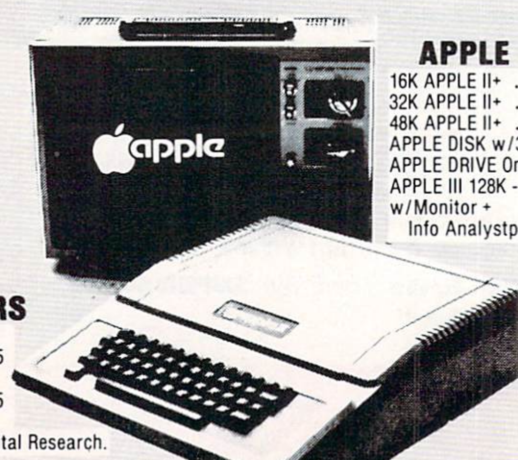
ACS 8000-2 64K 1M	$ 4500
ACS 8000-15 64K 1M	$ 5990
ACS 8000-6 208K 14.5M	$10490
ACS 8000-7 208K 29.0M	$11690
ACS 8000-10 208K 10M	$ 8500
ACS 8000-10/MTU	$10990

NEC SPINWRITER PRINTERS

5530 (Parallel)	$3055
5510 (Serial)	$3055
5520 (KSR-Serial)	$3415
Tractor Option	$ 225

EPSON PRINTERS

MX-80 PRINTER	$ 645
MX-80 FT	$ 745
MX-100	$ 945
MX-70	$ 459
INTERFACE CARDS	
8141 (RS-232)	$ 75
8150 (2K Buffered RS-232)	$ 150
8161 (IEEE 488)	$ 55
8131 (Apple Card)	$ 85
8230 (Apple Card)	$ 25
8220 (TRS-80 Cable)	$ 35



APPLE

16K APPLE II+	$1330
32K APPLE II+	$1430
48K APPLE II+	$1530
APPLE DISK w/3.3 DOS	$ 650
APPLE DRIVE Only	$ 490
APPLE III 128K - In Stock! w/Monitor + Info Analystpak	$4740

DIABLO 630 PRINTER

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

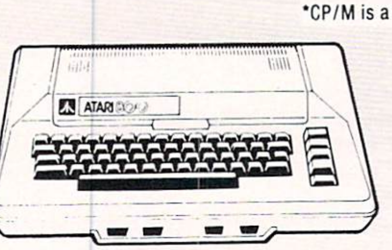
AMDEK MONITORS

Video 100 12" B+W	$ 179
Video 300 12" Green	$ 249
Color I 13" Low Res	$ 449
Color II 13" High Res	$ 999

INTERTEC COMPUTERS

64K Superbrain (360 Disk Storage), CP/M™	$3495
64K QD Superbrain (700K Disk Storage), CP/M™	$3995

*CP/M is a registered trademark of Digital Research.



ATARI COMPUTERS

Atari 400 (16K RAM)	$ 399
Atari 800 (32K RAM) - good thru 8/31	$1080
Atari 410 RECORDER	$ 89.95
Atari 810 DISK DRIVE	$ 599.95

NEECO carries all available ATARI Software and Peripherals.

PROFESSIONAL SOFTWARE

WordPro 1 8K	$ 29.95
WordPro 3 (40 Clm.) 16K	$ 199.95
WordPro 3+	$ 295
WordPro 4 (80 Clm.) 32K	$ 375
WordPro 4+	$ 450

JUST A SAMPLE OF THE MANY PRODUCTS WE CARRY. CALL US FOR OUR NEW 60-PAGE CATALOG.
WE WILL MATCH ANY ADVERTISED PRICE ON PRODUCTS LISTED UNDER SIMILAR "IN STOCK" CONDITIONS.

NEECO

679 HIGHLAND AVE.
NEEDHAM, MA 02194

(617) 449-1760
Telex: 951021

MON-FRI 9:00 - 5:00

MasterCharge and VISA Accepted

Ask The Readers

Robert Lock, Richard Mansfield
And Readers

Please address questions or answers to: Ask The Readers, **COMPUTE!** Magazine, P.O. Box 5406, Greensboro, NC 27403. Special thanks this month to Joseph Wrobel who sent in several extensive answers.

Answers

*"In reply to Rita Norton (**COMPUTE!** #15): filmstrips in Computer Science are available from Educational Activities, Inc., Freeport, NY 11520."*
Arnold Friedman

"In your August issue, you published a letter in the "ASK THE READERS" column about Edward Sweeney's problems with interfacing the Vortrax TYPE 'N TALK speech synthesizer with his Atari 800. The best way to do this is with the Atari 850 interface (list price is about $219.95). Since the TYPE 'N TALK uses an RS-232 interface to communicate with the computer, an RS-232 interface (such as the Atari 850) should be used to hook it up to any personal computer.

I have seen (and heard) the TYPE 'N TALK at a computer store, and I think that it is fantastic*! It may not sound as human as the TI line of synthesizers (Speak 'N Spell, etc.), but it* does *have an* unlimited vocabulary*! (The TI ones don't).*

For those of you who have been to the arcade recently, you may have noticed the video games called "Gorf" and "Wizard Of Wor." The Vortrax TYPE 'N TALK sounds almost exactly like the voice synthesizers used in these games (but with much more clarity and understandability). For fun (after you get the connections up and the software running) try making it say "bite the dust space cuddet" (exactly as spelled). Programming instructions are included with the Atari 850 in a large manual. You program using such common things a XIO XX, #X,X,X,"X:", OPEN #X,X,X,"X:", etc. (Note: The TYPE 'N TALK may also require an interface cable from Atari. Contact Atari for details on the cables.) The last thing that the TYPE 'N TALK needs is an 8-ohm speaker (available at Radio Shack, etc.).

One last thing (now changing the subject). In the same issue as Mr. Sweeney's letter, there was another one by Jerry Stern, asking about using the keypads for data entry. The answer to his question is "Yes!". For a program, Dr. Stern should get an Atari Basic Reference Manual *and look at appendix H, page H-14. I hope I was of help!" Greg Marquez*

"I'm writing with regards to the questions raised in your [Clyde Spender] **COMPUTE!** *#14 article entitled "Atari Graphics: 16 Colors!" concerning graphics modes 9 through 11. Actually, there are two questions I'll attempt to answer. The first asks how graphics modes 9 through 11 are supposed to operate. The second asks how they are currently implemented.*

The answer to the first question can be found on pages 172-174 of the Atari Personal Computer System Operating System User's Manual *(inside California call ATARI at 1-800-672-1430 for purchase info). These pages comprise the manual's Appendix H, entitled "Screen Mode Characteristics," which describes the characteristics of graphics modes 0 through 11. What it tells us about modes 9 through 11 is that they are all 80 pixel by 192 line modes, and all use four bits per pixel. Thus, they occupy the same amount of display memory as graphics mode 8, namely 192 line of 40 bytes per line. The difference between the three modes is how the four bits per pixel are interpreted.*

In mode 9 these bits are interpreted as luminance data; since the low-order luminance bit is ignored, this mode supports an eight-level gray scale display. The color of the display is determined by color register 4, the one which also sets the background color. Mode 11 is the inverse of mode 9; the pixel data select one of ATARI's 16 standard colors while the overall luminance is determined by color register 4. Graphics mode 10 is somewhat less straightforward. Here the different data values reference one of the 9 color registers (5 normal + 4 player missile). Because there are 16 possible data values and only 9 color registers, there is some duplication. The bottom line is that 9 different color/luminance combinations of your choosing are supported in graphics mode 10.

I believe the colors that you observe are false, due to the alaising caused by the high frequency transitions in the 320 dot per line display. If you're skeptical, run the accompanying program. If your ATARI behaves like mine, you should see no change in the graphics display except for the border. This border change is due to the fact that in graphics mode 11, and only in graphics mode 11, color register 4 is initialized to six rather than its normal initial value of zero.

Note that the program accesses the graphics modes using a BASIC command of the general form:

```
OPEN #6,X,G,"S:"
```

where G is the graphics mode and X is usually 8, i.e. OPEN for writing. This command can be used for all graphics modes (0 through 11). Add 32 to X to inhibit screen clear; add 16 to X for split screen display."
Joseph Wrobel

```
10 OPEN #6,8,10,"S:"
20 FOR I=0 TO 4
30 FOR J=0 TO 15
```

SOFT ROM

- 4096 BYTES OF SOFT ROM
- STORE MACHINE CODE SOFTWARE BEYOND THE BOUNDARIES OF BASIC
- WRITE PROTECT RAM WITH A FLIP OF A SWITCH
- STORE VARIABLES OR INDICES OUTSIDE OF BASIC
- SOLVE THE CONFLICTING ROM PROBLEM BY SOFT-LOADING THE APPROPRIATE ROM IMAGES

The SOFT ROM is compatible with any large keyboard PET/CBM or similar 2532 EPROM systems. It may be placed in any ROM socket to give the user room for machine code. If the SOFT ROM is placed in an occupied ROM socket, the user can transfer the PET/CBM ROM into the on-board ROM socket and select between ROM and RAM to manipulate the Commodore operating system.

Since the SOFT ROM places write protectable RAM into any of the computer's ROM sockets, it is ideally suited to use as a development tool to test ROM or EPROM based software systems before they are burned in.

Examples of software presently available for the SOFT ROM includes BASIC AID, UNIVERSAL WEDGE, SUPERMON, EXTRAMON, USER PORT PRINTER (Centronics parallel) and a buffered BACKGROUND PRINTER routine.

Installation is a simple plug-in into any available ROM socket.

$129.00

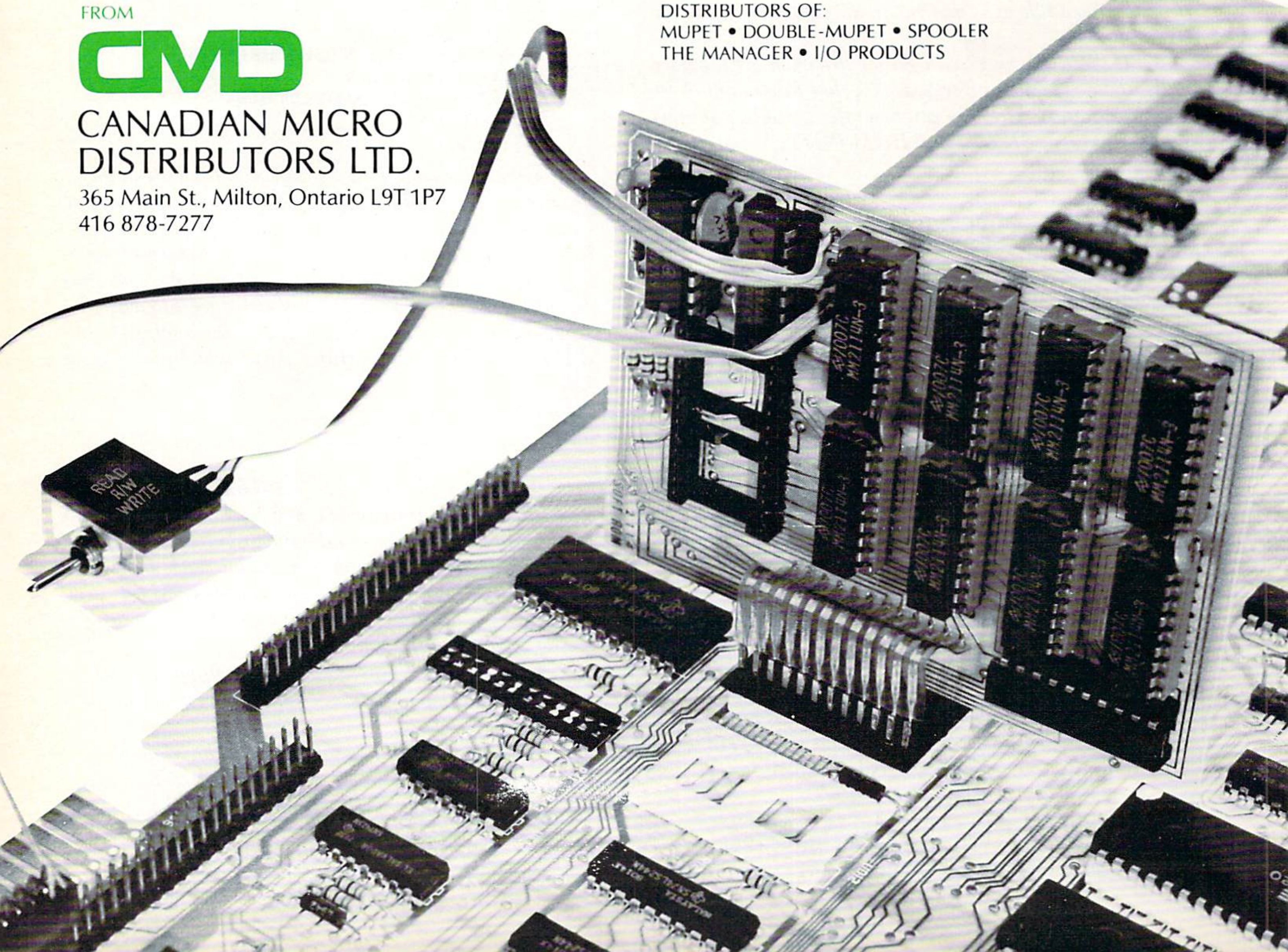
FROM

CMD

CANADIAN MICRO DISTRIBUTORS LTD.

365 Main St., Milton, Ontario L9T 1P7
416 878-7277

DISTRIBUTORS OF:
MUPET • DOUBLE-MUPET • SPOOLER
THE MANAGER • I/O PRODUCTS



```
 40 COLOR J
 50 PLOT 15*I+J,0:DRAWTO 15*I+J,191
 60 NEXT J
 70 NEXT I
 80 CLOSE #6
 90 FOR G=8 TO 11
100 OPEN #6,40,G,"S:"
110 FOR I=1 TO 250:NEXT I
120 CLOSE #6
130 NEXT G
140 GOTO 90
150 END
```

Questions

"What can we poor $395.00 Commodore 4010 Voice Synthesizer victims do to get any kind of programming help? Can you advise me of any users that can give me some kind of back-up in this matter?" L. W. Goesch

"I have written a program to cover my personal financial accounts which I run through monthly and add on the month's expenditures and income under various headings. The program then does various things with the data, such as forecasting for the whole and future years, highlighting items of over- and under-expenditure, etc.. The problem is that each month I have to list the DATA statements in order to amend them by adding on the current month's figures.

"Is there any way I can get the program to automatically update the date information so that I do not have to do this by hand? Any help you can offer would be appreciated...I have an 8K Original ROM PET...." Peter Shafe

You want the computer to imitate what you are now doing by hand: print a changed DATA statement on the screen and then press RETURN to place it into the program. This is a "self-modifying program." It can be done by telling the computer to print the new DATA statement on the screen and then POKEing carriage returns into the keyboard input buffer. Its starting address is 527 (your Original ROM PET), 623 (Upgrade and 4.0). The number of automatic RETURNs you need must be POKEd into address 525. For example, if you POKE 527,13: POKE 528,13 (putting two carriage returns into the buffer), you must then POKE 525,2. (For Upgrade and 4.0: POKE 158,2.) Finally, this line of POKEs must end with END.

This program would replace the DATA statement in line "L" with an updated set of DATA. (Notice that the new data is in variables Y and Z which, when printed on the screen, will be numbers. Also, the CLEAR SCREEN and HOME are used to correctly position the cursor so that the RETURNS will be made over the new DATA line):

```
100 PRINT"{CLEAR}{03 DOWN}"L"DATA"Y","Z"{
    DOWN}L="L+2":GOTO500{04 UP}":POK
    E525,2:POKE527,13:POKE528,:END
500 IFL>50THEN540:REM YOU PASS BACK THE V
    ALUE OF L TO THE PROGRAM
510 REM TO ALLOW THE PROGRAM TO KNOW WHIC
    H DATA LINE IT LAST CHANGED--
520 REM A DIRECT-MODE RETURN WILL NOT LEA
    VE THE VARIABLE VALUES INTACT.
530 GOTO 1000: REM THEN YOU CONTINUE TO C
    HANGE OTHER DATA UNTIL DONE.
540 PRINT"JOB ACCOMPLISHED":END
```

A similar technique is used to automatically delete lines in a program in **COMPUTE!** #12, pg. 116. This same result can be achieved on the Atari *as well: see* **COMPUTE!** #15, pg. 80. Also see last month's **COMPUTE!** page 22.

"First allow me to compliment you for creating what has been needed in a computer magazine for a long time. Your "Ask the Readers" column really puts you head and shoulders above the competition! Here is a problem that has had me frustrated for a year now:

I own an OSI C3 and the OS65D 3.0 operating system, with which I have been perfectly happy with except for one thing: I cannot successfully use sequential access data files. For example, let us say I have created the file, "DATA1". I would use the following program:

```
10 A$="DATA1"
20 DISK OPEN,6,A$
30 PRINT#6,A:PRINT#6,B:PRINT#6,C
40 DISK CLOSE,6,A$
50 PRINT "DATA NOW ON DISK"
60 GOTO 100
70 ..... (etc.)
```

Although I would have a line number 100 in the program, I would always get a US (undefined statement — no such line number) error in line 60. When I try to LIST the program, I just get garbage on the screen. I have followed the manual letter for letter and asked others about it, but all for naught; I still can't get it to work. I know that there must be some way — I have software in BASIC that does it successfully. Any suggestions? Please help!"
John Fry

"I wish to operate a Commodore 3016 computer and 3040 floppy disk drive from a marine 12v battery supply via wiring modifications and outboard circuitry if necessary. The result should allow conversion back to the usual AC power, but it need not be switchable between the two (although this would be an advantage). Can you supply any of the following?

1. *Details of required DC power supply voltages, current capacity, and regulation specifications for the two units.*

2. *A tested circuit by someone who has done it*

I realize that a 12v-to-240v inverter of 200 watts capacity is an alternate, but the direct approach is preferred." Frank Chambers ©

Guest Commentary:

The Three Laws

Isaac Asimov
New York, NY

Now that computerized robots are not only possible, but actual; now that they are rapidly invading industry, and becoming a factor that may produce extraordinary economic and social changes over the next generation; I can't help but think back forty years to the time when I invented the Three Laws of Robotics, in 1942.

These are:

1. A robot may not injure a human being, or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given it by human beings, except where such orders would conflict with the First Law.

3. A robot must protect its own existence, as long as such protection does not conflict with the First or Second Law.

Since these Laws are often quoted, quite seriously, in articles and books on robotics (a word I was, apparently, the first ever to use, back in 1942), I am sometimes tempted to wonder at the prescience of my 21-year-old self, and to suspect that perhaps the high opinion some people have of me may possibly be deserved.

But then rationality intervenes, and I know that this is nonsense. The Three Laws are obvious from the start, and everyone is aware of them subliminally. The Laws just never happened to be put into brief sentences until I managed to do the job.

The Laws apply, as a matter of course, to every tool that human beings use.

Consider a knife, for instance. The first law of knifedom is that it be used safely. No one would use a knife if it meant cutting one's fingers off in the process. Therefore, to begin with, a knife is equipped with a handle. To generalize, any cutting instrument must offer a way of being safely held while it is being used to cut.

The second law of knifedom is that it be used effectively. Therefore, a knife must be given a sharp edge (provided that is safe), for no one is interested in hacking away uselessly with a dull blade.

The third law of knifedom is that it maintain its integrity during cutting. Of what use would a knife be if it broke or dulled while cutting? A knife is therefore made of some tough material that holds an edge and that doesn't snap (provided such toughness doesn't interfere with either its safety or its effectiveness.)

You can apply this sort of reasoning, not only to material tools, but, also, without too much difficulty, to a social institution such as the Constitution of the United States.

The delegates to the Constitutional Convention of 1787 endeavored to work out a document that (first) would be safe to use, and would not subject Americans to a tyranny; and that (second) would be flexible enough to be responsive to the needs of the people, provided that did not compromise its

safety; and that (third) would be sufficiently durable to serve new times and new conditions, by means of amendments if necessary, provided that did not compromise either its safety or its effectiveness.

You can even apply this sort of reasoning to your own behavior: to your attitude toward your diet, or toward exercise, or toward your job. That behavior must insure first safety — then effectiveness — then durability.

Consequently, I have my answer ready whenever someone asks me if I think that my Three Laws of Robotics will actually be used to govern the behavior of robots, once they become versatile and flexible enough to be able to choose among different courses of behavior.

My answer is, "Yes, the Three Laws are the only way in which rational human beings can deal with robots — or with anything else."

— But when I say that, I always remember (sadly) that human beings are not always rational. ©

The Beginner's Page

Richard Mansfield
Assistant Editor

Searching Files

Here is a program to maintain a master index of all **COMPUTE!** articles. It will demonstrate a way that data can be *managed* by a computer to make entering and retrieving information fast, easy, and accurate. In specific, we will look at the problem of searching for data within a data file.

Central to most data management tasks is the job of searching through a list of records (a file) for a particular record or class of records. You might want to see all **COMPUTE!** articles on the topic of computer shows. The string-manipulating BASIC commands (LEFT$, RIGHT$, MID$, and LEN) are both fast and flexible when used as searching tools. This program will illustrate some of the major considerations when setting up a database management program:

Program 1. Microsoft Version

```
4 REM ******************************
5 REM *       INITIALIZATION       *
6 REM ******************************
10 T = 100
20 DIMA$(T)
30 FOR I = 1 TO T:READ A$(I)
40 IF A$(I)="END"THEN T=I-1:GOTO100
50 NEXT I
99 REM ******************************
100 REM*          MAIN LOOP         *
101 REM******************************
110 PRINT: PRINT "PLEASE CHOOSE:"
120 PRINT "1.AUTHOR
130 PRINT "2.SUBJECT
140 INPUT K$
160 ON VAL(K$)GOTO 200,300
199 REM******************************
200 REM*        SUBROUTINES         *
201 REM******************************
210 PRINT "TYPE AUTHOR'S LAST NAME
220 INPUT NAME$
230 FOR I = 1 TO T
240 L = LEN(A$(I))
250 FOR B = 1 TO L
260 IF MID$(A$(I),B,1)  =  "*"  THEN ~
   ~ GOTO 280
270 NEXT B:PRINT "MISSING *'S IN RECO~
   ~RD #"I:END
280 B = B+1:IF MID$(A$(I),B,LEN(NAME$~
   ~)) = NAME$ THEN PRINT A$(I)
290 NEXT I:GOTO 100
300 PRINT "PLEASE TYPE THE TARGET SUB~
   ~JECT"
310 INPUT SUBJECT$
320 FOR I = 1 TO T
330 L = LEN(SUBJECT$)
340 IF LEFT$(A$(I),L) = SUBJECT$ THEN~
   ~ PRINT A$(I):Q = 1
350 NEXT I:IF Q = 0 THEN PRINT "NO MA~
   ~TCHES FOUND"
360 Q = 0:GOTO 100
499 REM******************************
500 REM*            DATA            *
501 REM******************************
510 DATA SHOW--TRENTON COMPUTER FESTI~
   ~VAL*BUTTERFIELD*15
520 DATA PREVIEW--CBM FAT 40*BUTTERFI~
   ~ELD*15
20000 DATA END
READY.
```

Program 2. Atari Version

```
2 REM ATARI VERSION
3 REM
4 REM **********************
5 REM *   INITIALIZATION   *
6 REM **********************
10 T=100
20 DIM A$(T*80),T$(80),L(T),NAME$(40),SU
BJECT$(40)
30 FOR I=1 TO T:READ T$
40 IF T$="END" THEN T=I-1:GOTO 100
50 A$((I-1)*80+1,I*80)=T$
60 L(I)=LEN(T$)
70 NEXT I
100 REM **********************
101 REM *      MAIN LOOP     *
102 REM **********************
110 PRINT :PRINT "Please choose:"
120 PRINT "1. Author"
130 PRINT "2. Subject"
140 INPUT K
150 ON K GOTO 200,300
199 REM **********************
200 REM *    SUBROUTINES     *
201 REM **********************
210 PRINT "TYPE AUTHOR'S LAST NAME"
220 INPUT NAME$
230 FOR I=1 TO T
240 L=L(I):T$=A$((I-1)*80+1,(I-1)*80+L)
250 FOR B=1 TO L
```

ATTENTION COMPARISON SHOPPERS

HOW DOES A $299 BYTEWRITER-1 STACK UP AGAINST A $650 EPSON MX-80? YOU DECIDE!

The Only 80 Column Dot Matrix Printer Under $300.

Why do we dare to compare the Bytewriter-1 to the Epson MX-80, the industry leader? Because we feel strongly that dollar for dollar, the Bytewriter-1 is tough to beat for performance and quality.

Our extensive testing has proved that the Bytewriter-1 interfaces problem-free to the TRS-80, the Apple II and the Atari 400 and 800.

We are not going to tell you that the Bytewriter-1 is better than the MX-80, but by comparison, and for half the cost, you get more than a reliable printer — you get a great value.

Call or write for more information today.

Comparable features. Uncomparable price.

FEATURES	BYTEWRITER-1	EPSON MX-80*
Print speed	60 lines per minute	46 lines per minute
Paper feed	Friction feed original plus 3 copies	Pin feed original plus 2 copies
Ribbon	Black, cartridge $9.95	Black, cartridge $14.00
Life expectancy	Printhead — 100 million char. Drive Mech. — 10 million char. Ribbon — 5 million char.	50 — 100 million char. 5 million char. 3 million char.
Dimensions	3.8″ × 15″ × 9″	5.2″ × 14.7″ × 12″
Character set	96 ASCII	96 ASCII
Interface	Parallel	Parallel
Warranty	90 days	90 days
Printhead replacement	$29.95	$30
Cost	**$299**	**$650**

*Data source: Epson MX-80 Operation Manual

MICROTEK inc.
9514 Chesapeake Drive
San Diego, CA 92123
(714) 278-0633
Outside CA call
TOLL FREE (800) 854-1081
TWX. 910-335-1269

30 Day Money Back Guarantee

```
260 IF T$(B,B)="*" THEN GOTO 280
270 NEXT B:PRINT "MISSING *'S IN RECORD
#";I:END
280 B=B+1:IF T$(B,B+LEN(NAME$)-1)=NAME$
THEN PRINT T$
290 NEXT I:GOTO 100
300 PRINT "PLEASE TYPE THE TARGET SUBJEC
T"
310 INPUT SUBJECT$
320 FOR I=1 TO T
330 L=LEN(SUBJECT$)
340 T$=A$((I-1)*80+1,(I-1)*80+L(I))
345 IF T$(1,L)=SUBJECT$ THEN PRINT T$:Q=
1
350 NEXT I:IF Q=0 THEN PRINT "NO MATCHES
 FOUND."
360 Q=0:GOTO 100
499 REM ****************
500 REM *      DATA      *
501 REM ****************
510 DATA SHOW--TRENTON COMPUTER FESTIVAL
*BUTTERFIELD*15
520 DATA PREVIEW--CBM FAT 40*BUTTERFIELD
*15
20000 DATA END
```

How It Works

Let's see how this program searches through the file for records which match whatever is required. This kind of file (lines 510-520) is called a *variable-field file* because each of the three fields — the subject, the author, and the issue number — can be of any length. The DATA statements each contain one record. Within each record, the author field could be as short as "Cox" or as long as "Butterfield." The fields can *vary* in length. This saves memory space, but at the expense of speed. So, for large databases, where a search will go through hundreds or thousands of records, *fixed fields* are used because speed becomes an important factor. A name like "Cox" would be padded with blanks to take up the proper amount of fixed space required by its field. But we are setting up a smaller database and will take advantage of the memory efficiency of variable fields in this program.

Because the fields *are* variable, we have to let the computer know where one ends and another begins. Otherwise, how would it know that the author's name in line 510 wasn't Festival Butterfield? We separated the fields by using the "*". This means that we will have to look for "*" within each record when we are searching for the author fields. This is why the variable-field can be slow: each record must be handled individually, watching for *delimiters* (fences between fields). Let's take a look at what the program does:

Line Number

10 Here we tell the computer that there are a maximum of 100 records in our file. This is necessary because (in line 20) the computer must set aside memory space for each record.

20 Set up an *array* which is DIMensioned to 100 string-variable zones. Each string (record) will fill each zone, but Microsoft BASIC dynamically expands the zone sizes so the DIMensioning is merely the *number* of zones, not their size. (In Atari BASIC, the DIM statement sets up one giant string. The DIMensioning in Atari is the *size* of this huge string).

...for large databases, where a search will go through hundreds or thousands of records, *fixed fields* are used because speed becomes an important factor.

30 READ each DATA statement (place it into its proper, numbered zone in memory). READ up to the maximum (T) unless...

40 one of the DATA statements is the word "END" in which case you went *beyond* the last real record — you must subtract one from counter (I) so the program can now know the true total (T) number of records in this database.

50 Keep raising the counter (I) until you reach the limit (100) or get the word "END" (line 40), then...

110-160 A typical *main loop* structure with a menu of choices, an INPUT from the user, and then a *branching* which depends on the INPUT.

230 Establish a loop which will "pull out" each record, from record 1 to the total (T).

240 In each case, find out the length (L) of a record.

250 A second loop, *nested* within the other loop, which will count each *character* within a particular record. It will end, of course, when it reaches the total number of characters in a record (L).

260 When one of the characters is a "*", we know that we are positioned at the "author field" of the record. So we skip the "NEXT B" count up, in line 270.

270 If we have no more "NEXT B," then we went through an entire record without finding the "*" symbol and we print an error message on the screen and END the program so the bad DATA statement can be fixed.

280 First, since B now is the position of the

CRYSTALWARE

APPLE 48K-w/disk

THE FINEST IN FANTASY GAME SOFTWARE

ATARI 40K-w/disk

At Crystal we are doing our best to provide the finest state-of-the-art graphic adventure software in the world. Our list of credits include the first indoor-outdoor graphic adventure, the first multi-disk graphic adventure, and now for the Atari, the first graphic adventure in the world which includes screen scrolling and animation. The era of the text adventure and games which are simple combinations of static graphics and text is rapidly drawing to a close. We attempt to utilize the full potential of your computer. True, many of our games use up to 48K and we only deal in disk products, but there are a lot of users out there who have worked hard to upgrade their systems to the max and we think they deserve games that will give their computer system a run for its money.

★★★ ADVENTURE GAMES ★★★

1-THE HOUSE OF USHER- Haunted house type adventure game with scrolling in the Atari version. Wander the creepy hallways of the three story castle based on Edgar Allen Poe's short story of the same name. Written in graphics, of course, with animation and sound. We have introduced a new mystery for another $100 prize. $29.95/1 disk

2-FANTASYLAND 2041 A.D.- The largest disk based adventure game in the world (that we know of). Enter the Hall of Heroes and prepare yourself for the greatest fantasy-role-playing game you will see for years to come. To win you must survive Congoland, Arabia, King Arthur, Captain Nemo, Olympus (a sea voyage), and Dante's Inferno (Hell itself). In both the Atari and Apple versions it takes up more than 400,000 bytes of memory and uses more than 400 hires screens. The winner of the contest described in the manual with this game will receive $1000.00 and a bronze trophy. We have pushed the award date forward to February 1982 to allow more people to participate in the contest. $59.95/6 disks

3-GLAMIS CASTLE- Yes, Pat and I are on our way to Britain to stay in the dreaded Glamis Castle. If we survive our real life adventure, we'll be measuring it and will be able to provide you with a 3-D game based on this ancient haunted site where King Duncan met his end at the hands of Macbeth. Our good friend, Mark Benioff, after much research, said there's a mystery room that has never been found in this castle and a half beast, half-man creature that guards a treasure therein. Our stay will be covered by the British media and we hope to share our experience with you through the writing of this game. $49.95/2 disks

4-BENEATH THE PYRAMIDS- You are an archaeologist in 1932 and must find your way through the perilous chambers beneath the pyramids to discover a golden statue of the cat goddess Bast. This game is in hires graphics, includes sound, your little man actually moves through the corridors which you can see on the screen. The monsters are animated and very agressive. There is a new $100 prize for the first to solve the mystery; which is a toughie! $29.95/1 disk

★★★ SPACE GAMES ★★★

5-GALACTIC QUEST- An excellent combination of Star Trek and Space Trader. Battle the animated Vegan fighters as you warp from galaxy to galaxy. At the same time, you may land on and trade with hundreds of planets. Super hires graphics and lots of sound. This has been one of our most popular games. $29.95/1 disk

6-SANDS OF MARS- Take an exciting voyage to the planet Mars via the Starship Herman. This game compared to the rest, is second only to Fantasyland 2041 A.D. It includes scrolling on the Atari and hundreds of full screen graphics. You can move your little man through the terrain of Mars; if, of course, you survive the exciting journey to Mars, which occupies the whole first disk. There is a new mystery and another $100 prize just waiting for some clever adventurer out there. Good luck! $39.95/2 disks

★★★ WAR GAMES ★★★

7-WORLD WAR III- You Atari gamers will have to see this in the Atari version to believe it! If your tired of war games which take 15 minutes a move and have a manual the size of a telephone book; but still want a complex, real-time action war game-this is it! It is designed for two arm-chair generals which may manuever up to 128 seperate type of units at a time. The game displays a map of Iran & Iraq in the first scenario and later on you will find yourself moving nuclear submarines and battleships through two world wars. This is not a boring copy of a board based game but an original war game which takes a lot of skill and may take weeks to play. $29.95/1 disk

8-WATERLOO II- If you had been Napolean would you have done a few things differently? Well as you approach this final battle you are equipped with the same forces, face the same opposition, and survey the same terrain which he did. We have done a great deal of research to make this historically accurate as well as extremely complex. Even the angle of sight, fatigue of the individual soldier, and his psychological profile are included in the calculations. Oh by the way, your opposition is no slouch. You may find it more difficult to change the course of history than you think! $49.95/2 disks

★★★ ARCADIA ★★★

9-LASAR WARS- Hires-3d space war simulation. Protect the earth from alien invaders. $29.95

10-LITTLE CRYSTAL- The first of our line of education software, which will be completed by December. It includes a very fine version of Hangman, Mr. Music; which transforms the computer into a piano, Gunk-a hilarious shoot-em up game, and Storytime- an anthology of bedtime stories featuring Herman, the cat, Oscar, the Hamster, and of course, Little Crystal. $39.95

11-IMPERIAL WALKER- A fine game pack written by our Atari programmer, Michael (graphics) Potter. Includes the Walker animation which is superb, Gunfight, and Lasar Nim, a game of 'how many robots'. $29.95

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

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

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

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

Special Note! For those of you who live near beautiful scenic Silicon Valley, we'll be opening up a store called Country Computer at 7453 Monterey St. in Gilroy. Hopefully, we'll be carrying a full line of Apple and Atari products and invite you to drop in for our Grand Opening II on September 15.

VISA

master charge

Our order lines are open 24 hrs. a day 7 days a week.

(408) 778-2966 **CRYSTAL COMPUTER** **17429 Bluejay Dr., Morgan Hill, CA 95020**

"*" within the record, we add one to it so that B points to the first character of the author's name. Then we compare the portion of the string (record) which starts at position B and which is as long as the name which was INPUT in line 220. This will give us an exact comparison. It also has the benefit of allowing us to INPUT "BU" and search for all names starting with "BU" instead of writing out the entire name "BUTTERFIELD."

290 If we have finished checking and printing out the matches, we simply return to the main loop menu to see if other searches will be requested.

320 Again we loop through each record. This time we are looking at the first field: the subject field in a record. Therefore, we will not need to look for the "*". The subject field starts at the first position within the record.

330 Here we measure the length of the INPUT request. This allows us to have any level of specification. For example, if "DISK" is INPUT in line 310, all the records which refer to disks will be printed. If "DISK STORAGE" is INPUT, however, only records which match the entire SUBJECT$, "DISK STORAGE," will appear.

340 A simpler version of the compare in line 280. Q is set to equal 1 to show that a match was found so that line 350 will not print its message.

360 Q is reset to zero so it will function correctly as a *flag* of matches. We return to the main loop menu.

If you add more DATA lines, this program will hold and examine as many records as your RAM memory permits. If you have 16K RAM, you can put in more than twice as many DATA statements as would be possible in 8K RAM. (The program itself uses up some of the RAM.) ©

Basically Useful BASIC

Editor's Note: Peter is nine. His mother says that Peter designed the program himself, but she helped with the coding. — RTM

A Flower Sale Program

Peter Deal
Malvern, PA

I am a Cub Scout in Den 3, Willistown Pack 98. Every year we sell pansies. All my neighbors buy them, like almost everybody on my street. The next Saturday, my mom and I deliver the pansies. To know how many boxes are sold, and to know how much money everyone owes me, I use this program and my mom helped me write it on the PET. This year, pansies sold for $2.25 a box.

Helper's notes:

- PET people can type lines 100–390. Users of other computers should type in lines 270–590.
- This little program checks input for valid entries (between 1 and 50). The PET version does so by overwriting the input prompt, the other version by repeating that prompt. Incorrect entries can be further changed after the computer asks "7 boxes Y/N". Lines to 200–210 in the PET Version, lines 540–560 in the other version handle that.
- It is likely that lines 530–540 are system dependent, so they deserve an explanation. Line 530 clears the buffer of all key presses. Line 540 waits until "Y" or "N" is typed in. All other keys are rejected. Line 550 then sends the control back to the input prompt if the answer was "N". Otherwise the program proceeds to calculate.
- Jim Butterfield's routine is used in several places by specifying V, V1 and V2 parameters. This routine has been fully described in **COMPUTE!** #9 (pg. 30).

```
100 REM==============================
110 REM PANSY SALE - PET VERSION
120 REM==============================
130 PR=2.25:MN=Ø:BX=Ø:TC=Ø:TB=Ø:REM PRICE
    ,MONEY,BOXES,TOTAL MONEY,TOTAL B
    OXES
140 SP$="           ":SP$=SP$+SP$+SP$
150 PRINTSP$:PRINT"{UP}";:V$=""
160 INPUT"> 'END' OR HOW MANY BOXES   _{Ø3
     LEFT}";B$ : IFB$="END"GOTO25Ø
170 BX=VAL(B$):IF BX<=Ø OR BX>5Ø THENPRIN
    T"{UP}";:GOTO15Ø
180 V1=3:V2=Ø:V=BX:GOSUB3ØØ:PRINT "     {Ø
    4 LEFT}" V$ " BOXES - Y/N";
190 FORJ=1TO9:GETQ$:NEXT
200 GETQ$:IFQ$<>"Y"ANDQ$<>"N"GOTO2ØØ
210 IFQ$="N"THENPRINT:PRINT"{Ø2 UP}";:GOT
    O15Ø
220 MN=PR*BX : TB=TB+BX : TC =TC+MN
230 V1=4:V2=2:V=MN:GOSUB3ØØ:GOSUB37Ø: PRI
    NT"--------------"V$
240 GOTO16Ø
250 PRINT:PRINT:V1=7:V=TC:GOSUB3ØØ:GOSUB3
    7Ø:PRINT"I SOLD" TB "BOXES FOR"V
    $:END
260 REM==============================
270 GOTO48Ø
280 REM 'USING' ARRANGE IN COLUMNS, JIM B
    UTTERFIELD ROUTINE TO 35Ø
290 REM V IS VALUE; V1.V2 PRINTS
300 V4=INT(V*1Ø↑V2+.5):REM ROUNDED
310 V$=RIGHT$("            "+STR$(V4),V1+V2
    +1):IFV2<1GOTO34Ø
320 FORV5=V1+2TOV1+V2+1:IFASC(MID$(V$,V5)
    )<48THENNEXTV5
330 V6=V5-V1-1:V$=MID$(V$,V6,V1+1)+LEFT$(
    ".ØØØØØØØØØ",V6)+MID$(V$,V5)
340 IFASC(V$)>47THENV$=LEFT$("*********",
    V1+V2+2+(V2=Ø))
350 RETURN:---
360 REM FLOAT $
370 FORV7=1TOLEN(V$):IFMID$(V$,V7,1)=" "T
    HENNEXTV7
380 IFV7>1THENV$=LEFT$(V$,V7-1)+"$"+MID$(
    V$,V7)
390 RETURN:---
400 REM==============================
410 REM SIMPLIFIED VERSION FOR
420 REM OTHER MICROSOFT SYSTEMS
430 REM    U N T E S T E D  !
440 REM==============================
450 REM TYPE CODE FROM LINE 27Ø DOWN
460 REM TO INCLUDE BUTTERFIELD ROUTINE
470 REM==============================
480 PR=2.25:MN=Ø:BX=Ø:TC=Ø:TB=Ø:REM PRICE
    ,MONEY,BOXES,TOTAL MONEY,TOTAL B
    OXES
490 PRINT"> 'END' OR HOW MANY BOXES":INPU
    T B$: IF B$="END"GOTO59Ø
500 BX=VAL(B$):IF BX<=Ø OR BX>5Ø GOTO49Ø
510 V1=3:V2=Ø:V=BX:GOSUB3ØØ
520 PRINT V$ " BOXES - Y/N"
530 FORJ=1TO9:GETQ$:NEXT
540 GETQ$:IFQ$<>"Y"ANDQ$<>"N"GOTO54Ø
550 IFQ$="N"GOTO49Ø
560 MN=PR*BX : TB=TB+BX : TC=TC+MN
570 V1=4:V2=2:V=MN:GOSUB3ØØ:GOSUB37Ø: PRI
    NT"-----------------------"V$
580 GOTO49Ø
590 PRINT:PRINT:V1=7:V=TC:GOSUB3ØØ:GOSUB3
    7Ø:PRINT"I SOLD" TB "BOXES FOR"V
    $:END
```

©

Editor's Note: Program in APL, FORTRAN, Assembly, BASIC, or PASCAL. Have 96K user RAM available. Connect directly to mainframes. Modify variables during a program RUN. These are some of the reasons the new PET is called Super.

We'll be testing a SuperPET here at **COMPUTE!** *and next month we'll let you know the results, when the machine will be in the stores, and 8032 upgrade news. — RTM*

SuperPET's Super Software

Terry Wilkinson
Waterloo Computing Systems

As most of our readers know, the University of Waterloo has been involved for many years in the development of computer software for its own applications. In the 1960's, a number of specialized batch systems were created for teaching computing. As well, pioneering work was done in the development of interactive terminal systems to make the large batch oriented computers easier to use. Language processors such as WATFOR, WATFIV and WATBOL were augmented by such systems as WITS (an early interactive terminal system). Through it all, the main emphasis was on enhancing the learning environment at the University of Waterloo by developing software to meet our specific needs.

In the 1970's, these efforts continued, using the concepts of distributed processing and the emerging minicomputer technology. Powerful interactive systems on remote PDP-11's and IBM SERIES/1's allowed the preparation of jobs and the examination and printing of output to take place "offline" from the computer. Waterloo developed packages like WIDJET which allowed more students to use the computer at a lower cost. High-speed BISYNC lines provided communication between the minis and the mainframes allowing access to the language processors available there. In addition, packages such as WATFOR-11 and WATBOL-11 were created to run on stand alone mini systems.

The underlying theme through all these developments was the production of systems that could make allowances for small mistakes and give good error diagnostics, thereby making the program development process a little easier. These systems have always included reasonable enhancement beyond prevailing language definitions so users could become familiar with, and make use of, the latest programming technology as it was being developed in the industry.

It is, therefore, consistent with this longstanding tradition that the University of Waterloo would extend these concepts one step further in the 1980's. That step is in microcomputer technology. It addresses the use of "stand-alone" microcomputer systems as well as their use in distributed processing.

The First Step: Waterloo Microsystems

The first step in exploiting this new technology was to apply the lessons learned over the years concerning software development. A new family of language processors was created which would support APL, BASIC, FORTRAN and PASCAL. Also, a general purpose text EDITOR was developed to allow easy manipulation of program and data files. These packages were written in a system-independent, portable manner to provide a very high degree of flexibility in implementation. The success of this approach is evident in that completely compatible versions of these packages have been installed on IBM's VM/CMS system for large 370-like machines and the new Commodore SuperPET with the Motorola 6809 microprocessor chip. Work is also underway to install them on the DEC PDP-11 system with RSTS/E.

This means that, using this family of language processors and subject to memory constraints, a program written for one of these machines will run on any other of these machines *unchanged*. This provides great flexibility in software development. Applications can be created which run on a variety of computers without modification. Components of a system can be created using one type of computer and then used on another type of computer.

In addition, a 6809 Assembler Development System was created to provide a straightforward, yet powerful, facility to create programs in machine language for use on the SuperPET or other 6809-based systems.

The second step was to provide a simple-to-use interface between the microcomputer and the mainframe computer. Programs and files of data would need to be transferred from one machine to the other. And, data files on the mainframe should be easily accessible by the programs running on the microcomputer.

Since most large-to medium-size computers support ASCII-type terminals, the approach involved an RS232-C serial line from the micro to the mainframe. Also required was an interface program for the mainframe to service the data management requests from the micro in the appropriate way. This interface program is called HOSTCM, standing for "host communication module."

Because of this approach, programs on the microcomputer can access data files on a local disk

or on a remote host disk *with equal ease*.

The result of these developments is a very powerful collection of hardware and software which can be used in various configurations to service the needs of a wide range of users. For example, a stand-alone SuperPET can have local disk(s) and printer(s) and support all five languages and the editor without any connection to a remote computer. Alternatively, a SuperPET connected to a host computer might have no local devices and keep all its files on the host. Of course, a combination of the above configurations yields a powerful micro configuration with the additional ability to transfer information to and from the host machine.

A SuperPET With: APL, BASIC, FORTRAN, and PASCAL

It is clear that the IBM 370-like machines and the PDP-11's were quite capable of handling their part in a system such as the one described above. It was not as easy, however, to discover a microcomputer which could be used in this design. The best approach seemed to be to modify an existing micro and thereby give it the required facilities. Our previous involvement with the Commodore PET had given us considerable knowledge of its construction and we felt confident that the CBM 8032 could be modified to do the job. It had a MOS6502 micro-processor chip and 32K (kilobytes) of user RAM.

Three fundamental changes were required:

a) conversion to the Motorola 6809 microprocessor chip
b) addition of more RAM
c) addition of an RS-232 serial interface

It also seemed desirable to retain the previous 6502 processor and allow the machine to operate as a normal CBM 8032, if desired. This would preserve its ability to run already existing packages developed for the 6502.

The initial Waterloo designs were taken by BMB CompuScience Ltd. of Milton, Ontario and developed into a working prototype which could be mass produced. This firm used its considerable experience in hardware design to produce the final product which contains two microprocessors, an MC6809 and an MOS6502. An external switch was included to allow the user to select one mode or the other. In 6502 mode, the machine operates as a CBM 8032 using Commodore BASIC in ROM and has a 32K RAM. In 6809 mode, a different ROM is selected. At the same time, Waterloo Computing Systems Ltd. undertook the task of implementing system software to operate using the BMB hardware configuration in 6809 mode. The following list of software was implemented: the Waterloo microSystems Supervisor (resident in the 6809 ROM set); interactive interpreters for APL, BASIC, FORTRAN and PASCAL; and a development system for 6809 machine language programming. Subsequently, Commodore has begun manufacturing this hardware configuration, called the "SuperPET," under license from BMB and is including the entire collection of software with each machine sold.

An additional 64KB of RAM was installed to allow room for the Waterloo microSystems language processors. The user selects which language he wishes to use from a menu which appears on

The technique is called *bank-switching* and... 64K of RAM is logically divided into 16 *pages*, each containing 4K.

the screen when the unit is turned on. The processor for that language is "soft-loaded" into the additional RAM. This means that the user still has the entire 32K of original RAM available for his use regardless of which language he chooses.

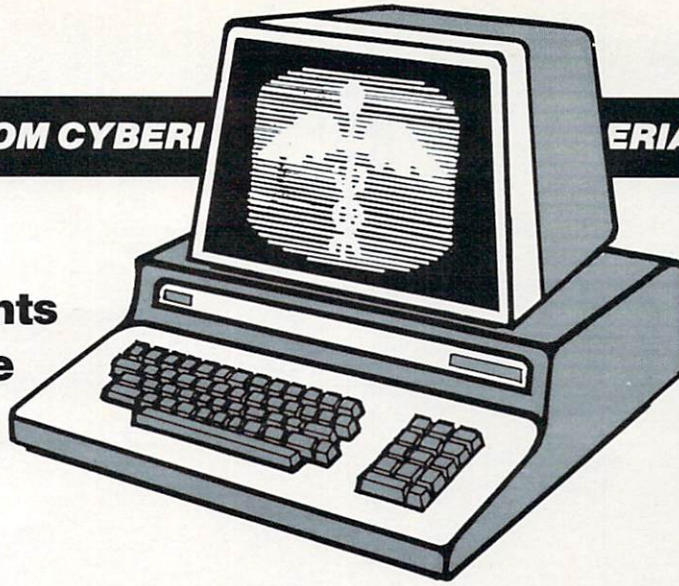
The usable space in the 16-bit address structure of the 6809 system was almost fully allocated and a special technique was required to allow addition of the 64K of RAM which the processors would need. The technique used is called *bank-switching* and, with it, the 64K of RAM is logically divided into 16 *pages*, each containing 4K. A 4K *window* in the address space can be positioned over any one of these 16 pages by simply setting a byte in the I/O area of memory. In the SuperPET, this window occupies addresses 9000-9FFF (hexadecimal) in the address space (see figure).

The Language Processors

The various high level languages are implemented in the Waterloo microSystem by means of interpretive language processors. This means the APL, BASIC, FORTRAN and PASCAL programs are stored internally in an encoded format. These encoded statements are then interpretively executed by a "run-time" supervisor unique to each language. Such an approach makes it possible to stop program execution, examine and modify program variables and data, and then resume execution from the place where it suspended. In APL and BASIC, it is even possible to interrupt and modify the program itself and then continue execution.

All the languages interface with the *host* computer, the serial line and the various disks and printers on the IEEE-488 bus using a common file system interface supplied in the Waterloo microSystem library. It provides 100 percent compatibility among data files across the various language processors. These file system functions, and many

NEW FROM CYBERIA ★ NEW FROM CYBERIA ★ NEW FROM CYBERIA



Mari

A comprehensive accounts receivable and insurance billing system for modern health care offices and clinics

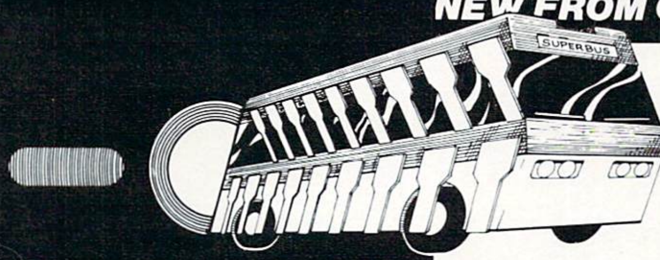
DESIGNED FOR COMMODORE 8000 SERIES COMPUTERS AND DISK DRIVES

- Help functions are always on-line
- Supports CPT, ICD and RVS medical coding
- Supports open item or balance forward accounting systems
- Interfaces to popular word processing programs
- Multiple terminals may be added with Superbus 4.0
- Multiple disk drives may be used—no limit on number of patients or accounts
- Includes a data base and forms generator to fill out any insurance form
- Includes a computer aided instruction program to train new users

Includes these standard reports:

- A/R aging
- A/R transactions
- General ledger
- Instant cash receipt
- Income analysis by physician
- Doctor referral report
- Patient/account cross reference
- Standard SuperBill insurance form

NEW FROM CYBERIA ★ NEW FROM CYBERIA ★ NEW FROM CYBERIA



NEW ★ NEW ★ NEW

SUPERBUS 4.0

Commodore Computer Networking System

- Up to 18 computers can be interconnected
- Multiple disk drives, printers and other devices may be added to the network
- Sophisticated security system prevents unauthorized use
- WordPro and Wordcraft programs are supported to allow multiple terminal word processing

*WordPro is a trademark of Professional Software, Inc.

CYBERIA INC. 2330 LINCOLN WAY, AMES, IA 50010 • 515-292-7634

ADVENTURE has gone GOLD!

AND WE WANT YOU TO CELEBRATE WITH US.*

The 12 Scott Adams' Adventures are presented in our **Limited Gold Edition.** Each tape and disk is individually numbered, and guaranteed until July 10th, 2001. Each package contains a certificate of authenticity, a registration card and an autographed, (rather interesting) photo of the author. The 12 Adventures normally retail for $239.40 individually on tape and $159.80 for 4 triple-pack disks.

The **Limited Gold Edition** is yours . . . **forever** . . . for $100.00 tape or disk, value for value. To Order: The **Limited Gold Edition** is available in very limited quantity, from interested Software Retailers. Ask your local dealer. If he does not have The **Limited Gold Edition** . . . Then call toll free 1-800-421-5770. In California 1-800-262-4242 (Local 213-670-9461) . . . And we will direct you to a dealer who does have the **Limited Gold Edition.** ***Supply Is Limited!***

The **Limited Gold Edition** is available in the following different configurations.

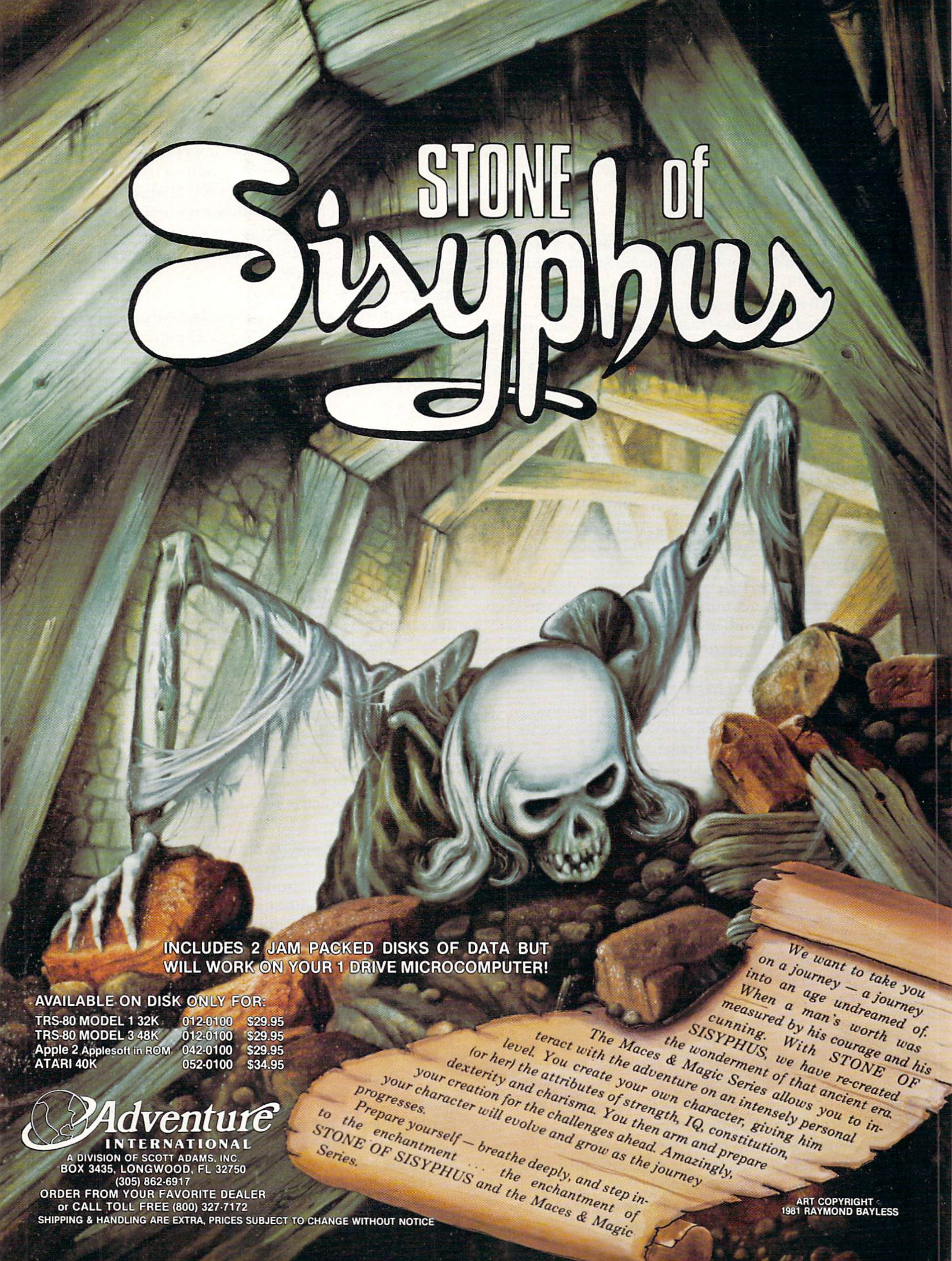
- Apple 2 Disk with TRS-80 Disk in same package
- Atari Tape
- Atari Disk
- TRS-80 Tape

$100.00 Each

*In keeping with the spirit of **Adventure**, $1,000.00 in Gold Coins is hidden within The **Limited Gold Edition.** Its waiting for **you.**

The **Limited Gold Edition** from Adventure International is distributed exclusively by: SOFSEL

8295 SO. LA CIENEGA BLVD. • INGLEWOOD, CA 90301


STONE of
Sisyphus
INCLUDES 2 JAM PACKED DISKS OF DATA BUT WILL WORK ON YOUR 1 DRIVE MICROCOMPUTER!
AVAILABLE ON DISK ONLY FOR:
TRS-80 MODEL 1 32K 012-0100 $29.95
TRS-80 MODEL 3 48K 012-0100 $29.95
Apple 2 Applesoft in ROM 042-0100 $29.95
ATARI 40K 052-0100 $34.95
We want to take you on a journey — a journey into an age undreamed of. When a man's worth was measured by his courage and his cunning. With STONE OF SISYPHUS, we have re-created the wonderment of that ancient era.
The Maces & Magic Series allows you to interact with the adventure on an intensely personal level. You create your own character, giving him (or her) the attributes of strength, IQ, constitution, dexterity and charisma. You then arm and prepare your creation for the challenges ahead. Amazingly, your character will evolve and grow as the journey progresses.
Prepare yourself — breathe deeply, and step into the enchantment . . . the enchantment of STONE OF SISYPHUS and the Maces & Magic Series.
Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.
BOX 3435, LONGWOOD, FL 32750
(305) 862-6917
ORDER FROM YOUR FAVORITE DEALER
or CALL TOLL FREE (800) 327-7172
SHIPPING & HANDLING ARE EXTRA, PRICES SUBJECT TO CHANGE WITHOUT NOTICE
ART COPYRIGHT © 1981 RAYMOND BAYLESS

others such as trigonometric functions, string manipulation routines and a floating point emulator, are available through documented interfaces to the machine language program as well using the 6809 Assemble Development System.

System Highlights

APL

The APL processor follows the well known IBM/ ACM 79 standard for the language. Some highlights of the system:

— All the standard primitives are implemented, including matrix-divide, dyadic-transpose, format, and execute.

— Functions such as quad-CR and quad-FX allow the dynamic creation and modification of functions.

— Direct access to memory and machine language programs is provided with quad-PEEK, quad-POKE, and quad-SYS (including generalized parameter-passing).

— A powerful, full-screen editor allows easy modification of functions. It also accepts indentation and comments to enhance program readability.

— The SuperPET screen supports all the APL characters (including overstrikes) and also provides a number of common graphics.

— An APL-sequential file feature allows storage and retrieval of complete APL data items including rank, shape, and type.

— A BARE-sequential file feature allows transmission of arbitrary strings of bytes in and out of the workspace.

— Relative files are also supported.

BASIC

The BASIC processor includes the ANS Minimal BASIC standard features as well as several noteworthy extensions:

— Variable, Array, Function and Procedure names can each be uniquely defined using up to 31 characters.

— Multi-line functions and procedures can be written and called with parameter passing. This feature can be used to implement recursive algorithms.

— The family of MAT (matrix manipulation) statements are implemented.

— A number of structured control statements provide a facility to enhance program style.

— Support a program text indentation and comments following statements further enhance program readability.

— A powerful generalized string/substring feature has been included.

— Built into the system is a broad set of functions to perform common operations such as SIN, COS, LEN, HEX$, ORD, and VALUE. There are about 35 such functions available.

— Error trapping allows interception of, and recovery from, most run-time errors.

— Commands such as RENUMBER, AUTOLINE, and MERGE allow easy manipulation of the BASIC program source code.

A full-screen editor makes changing existing statements simple.

FORTRAN

The FORTRAN processor implements a powerful subset/extension of the standard language. It includes many of the popular features of the well known WATFIV-S compiler as well as many features described in the FORTRAN-77 standard. This "Waterloo dialect" of FORTRAN includes:

Multi-line functions and procedures can be written and called with parameter passing.

— FORMAT statements

— Subroutines and functions

— Multi-dimensional arrays

— Extended character string manipulation

— Structured Program Control statements

— Sequential and Relative file support

— An interactive debugging facility

PASCAL

The PASCAL processor implements a version of the language which corresponds closely to the draft of the International Standard Organization (ISO) PASCAL committee, a refinement of the original language definition by Jenses and Wirth. Features include:

— Text file support

— Pointer variables

— Multi-dimensional arrays

— An interactive debugging facility including breakpoints, single step etc.

— Extensive data-typing capabilities

Assembly

The EDITOR is a processor which provides a powerful means of creating and maintaining general data and program files. It is a line-oriented, contextual editor with many features:

— GET and PUT commands retrieve entire

At long last. A complete line of Voice I/O peripherals for Commodore computers, starting at $119.

Let's face it. Voice I/O is a fascinating and efficient way to communicate with computers. And now, thanks to VOICETEK, Voice I/O peripherals are easily available, easy to use and very affordable.

If you own a Commodore computer, we give you a choice of three different peripherals that will enable your computer to understand your spoken commands. And two of these peripherals will talk back.

What's in a name.

We call each member of our family of Voice I/O peripherals COGNIVOX, from COGNItive and VOX. It helps you remember that when you talk, COGNIVOX thinks about what you say.

The top-of-the-line COGNIVOX model for the PET/CBM is VIO-1002. It offers natural sounding voice output and excellent performance as a speech recognizer. It costs only $249.

If the quality of voice output is not important for your application, then you can save $100 by ordering VIO-432. Priced at $149, VIO-432 is ideal for hobbyists or persons mainly interested in speech recogniton.

Finally, if you have an 8K PET, there is insufficient memory for voice response, so we offer a recognition-only COGNIVOX, model SR-100P. It costs $119, making it the lowest priced speech recognizer ever offered for sale. Yet its performance rivals that of units selling at much higher prices.

Which brings us to the next point we would like to make, namely, why we offer so much performance for so little money.

It's the technology.

Our Voice I/O peripherals are based on a technological breakthrough that made it possible to compress the required electronics onto a single integrated circuit chip. We are the only company so far that has achieved this remarkable feat. No wonder we offer such reasonably price voice peripherals.

In addition, COGNIVOX uses an exclusive non-linear, learning pattern matching algorithm to do speech recognition. Which means more reliable performance and ease of use.

What makes it talk.

COGNIVOX digitizes and stores in memory (using a data compression algorithm) the voice of the user. This gives three major advantages:

First, there are no restrictions to the words COGNIVOX can say. If you can say it (or sing it, or whistle it for that matter) your computer can do it too. Second, It is very easy to program your favorite words: just say them in the microphone.
Third, you have a choice of voices, male, female, child, accents, etc. this unprecendented flexibility offered by COGNIVOX is a must in the personal computer environment. Voice synthesizers and the "talking chips" do not offer this flexibility and therefore we feel they are not suitable for use with personal computers. In addition, voice output quality can be poor, especially for synthesizers. In that respect, VIO-1002 is clearly superior to anything else on the market and it is a must if voice quality is important (for example, business applications).

Some specifications

COGNIVOX can be trained to recognize words or short phrases drawn from a vocabulary of up to 32 entries chosen by the user.

Training COGNIVOX to your vocabulary is easy. All you have to do is repeat the words three times at the prompting of the computer.

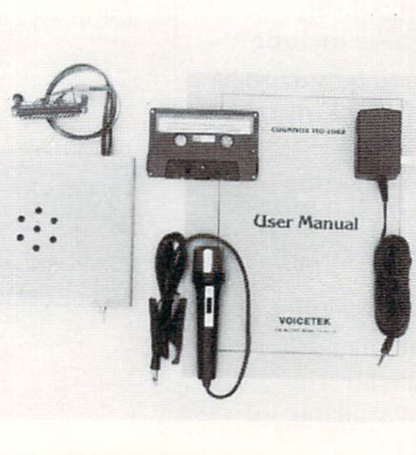
If you would like to have COGNIVOX respond to more than 32 words, you can have two or more vocabularies of 32 words and switch back and forth between them using a word.

The Voice output vocabulary can have up to 32 words phrases. Data rate is approximately 700 byte per word.

Ready to listen.

All COGNIVOX units are complete Voice I/O peripherals ready to plug in and use. They come assembled and tested and they include microphone, cassette with software and manuals. VIO units include built-in speaker and amplifier (yes, CB2 is also connected for music and sound effects).

They all plug into the user port and they receive their power from the cassette port except VIO-1002 which uses a wall transformer supplied with the unit.



Easy to use.

All you need to get COGNIVOX up and running is to plug it in and load one of the programs supplied. Load the demo program and start talking to your computer right away. Or load one of the games and discover the magic of voice control.

It is easy to write your own talking and listening programs too. A single statement in BASIC is all that you need to say a word or to recognize a word. Full instructions on how to do it are given in the manual.

Works with all versions.

COGNIVOX will work with all versions of the PET/CBM line. Old, new and newer ROMs. At least 16K of RAM is required (SR-100P will work with 8K of RAM).

If you have a disk system, you can use it to save vocabularies. Instructions are given in the manual.

Many uses.

With COGNIVOX your imagination is not the limit as the saying goes. It is the starting point. Cognivox is a super toy, an educational tool, an aid to handicapped, a data entry device while hands and eyes are busy, a foreign language translator, a sound effects generator, a telephone dialing device, an answering machine, a talking calculator. Using the IEEE 488 port you can control by voice instruments, plotters, test systems. And all these devices can talk back to you, telling you their readings, alarm conditions, even their name.

Order your COGNIVOX now.

To order by mail, give us the model number of the unit you wish to order, the make and model of your computer and your name and address. Enclose a check or money order and make sure to include $5 for shipping and handling. CA residents please add 6% tax. You may also order by phone and charge it to your Master Charge or VISA. Our phone number is (805) 685-1854 9AM to 5PM PST, Monday through Friday. Foreign orders are welcome, please add 10% for air mail shipping and handling. Payments must be in US funds. COGNIVOX is backed by an 120-day limited warranty against manufacturing defects.

VOICETEK

P.O. Box 388, Goleta, CA 93116

files of data into the machine for editing and then save them back on external devices.

— SEARCH and CHANGE facilities, including global change, provide for easy modification of text.

— A full-screen capability allows individual changes to be made by simply moving the cursor around the screen and entering the changes.

— A set of function keys provides extensive cursor movement and scrolling facilities.

The 6809 Development System is comprised of an Editor, and Assembler and a Linker. Code entered via the Editor can be assembled into relocatable modules by the Assembler. Then these modules are combined and relocated by the Linker to produce an "executable load module." This load module can be loaded into the machine, executed, and debugged using the Monitor built into the Waterloo microSystems Supervisor. Some features of the Assembler are:

- — Motorola 6809 Assembler language
- — Macro capability
- — Pseudo-opcodes for structured programming
- — Long label names allowed
- — Produces relocatable object code

Some features of the Linker are:

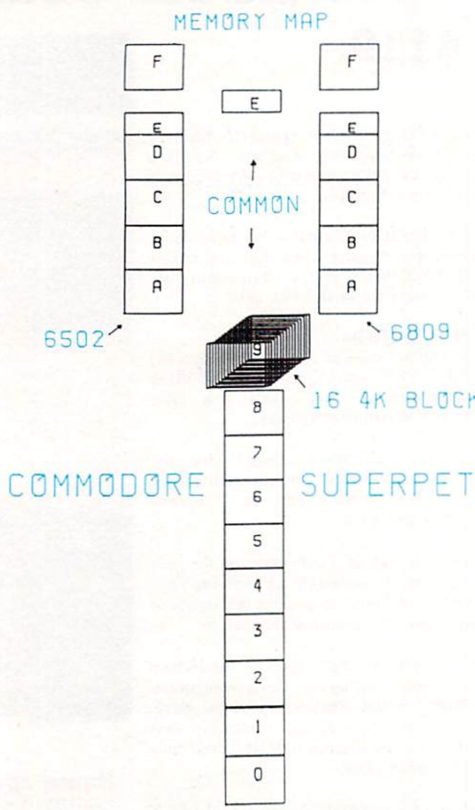
- — The ability to combine many relocatable object files into a single load module.
- — Relocates code to any arbitrary machine location
- — Supports the "bank-switched" RAM feature of the SuperPET making it almost transparent to the user.
- — When used with the supplied file "WATLIB.EXP," provides automatic symbolic linkage to the functions available in the Waterloo microSystems Library.

The Operating System

The Waterloo microSystems Supervisor resides in the ROM of the SuperPET. It has many of the features of a true operating system. These include:

- — The facility to load 6809 machine language programs into the SuperPET RAM including bank-switched memory.
- — Full-screen monitor facilities to examine and modify arbitrary locations in memory; bytes are displayed in both hexadecimal and character formats.
- — A built-in disassembler to facilitate examination of programs residing in the machine.
- — A built-in library of functions used by the high level language processors, but available to the machine language programmer.

Figure 1.



— A serial-line setup routine to set the baud rate, stopbit, and parity used to establish the host-protocol parameters for mainframe communications with the HOSTCM program.

— A "passthru" terminal mode to communicate directly with a remote computer via the serial RS232 line. ©

SuperPET: A Preview

Bill MacLean
BMB Compuscience
Milton, Ontario

The day is near when we will all start to see a lot of SuperPETs, so we thought it was time to describe the system and try to speculate on the significance of the design. In reality, the SuperPET is two different computers in the same cabinet. It is the old (today!) 8032 with which we are all familiar, with a few new wrinkles. It is also a brand new 6809 based computer system with outstanding potential in its own right. We will try to evaluate it in each of these two categories.

SuperPET — 6502

Looked at one way, the only difference between the SuperPet and the 8032 is the RAM capacity and the I/O system. There is 64K of additional RAM in the SuperPET. It is mapped as 16 4K blocks all residing in the block $9000-$9FFF. This may seem impossible, but it is done by write-only register at $EFFC (61436) which allows the programmer to select which of the 16 blocks can be read and written to in the $9000 block. For example, to select the 15th block, the register at $EFFC numbered 0-15.) In addition to this extra RAM, there is an intelligent USART (Universal Synchronous, Asynchronous Receiver Transceiver) or RS-232C port located at the base address $EEF0 (61424). This device allows the programmer the option of software selecting such serial communications parameters as Baud rate, word length, number of stop bits, parity, etc. The USART used is the 6551 system. It is a chip currently being manufactured by MOS Technology and is simulated in software on the VIC 20.

The first question everyone seems to ask about the RAM capacity is "will my Visicalc use the extra memory" or "will my BASIC programs now access the additional memory?" The answer is *no*. It would require substantial modification for any of the existing operating systems to use this additional memory. However, this is true of any conventional memory expansion scheme that takes one over the 65K boundary on an 8-bit processor. That was the bad news, now for the good. Many of your favorite programs will be modified to run on the SuperPET and will utilize the extended memory. The concept of *paged* memory is a very powerful one. Once the step has been taken to design systems using this concept, virtually unlimited RAM can be used, guaranteeing much easier expansion in the future as the price of RAM drops.

SuperPET — 6809

You may well ask what is a 6809, and why is there one in a Commodore product? This microprocessor is a logical extension of the 6800 and the 6502, just as the 6502 was a logical extension of the 6800. It is a pseudo 16-BIT processor. This means that all of the internal registers and the stack pointer(s) are 16 bits long. This allows comfortable addressing anywhere in the address bus range. There are some other major differences, particularly suitable to the creation of position-independent code. It is in the areas of compactness of run-time code, the use of the stack, and the position-independent code capabilities that the 6809 shines. These characteristics made it much more attractive to design the SuperPET software systems using the 6809.

Describing the potential of this system is a little difficult. Under the Watcom operating system, the machine is really several distinct computers. This operating system consists fundamentally of a 22K ROM set occupying the top half of the 65K address range just as the CBM system does. This area contains all system libraries, I/O routines, monitor (6809) etc. The languages (BASIC, FORTRAN, APL, PASCAL, and Assembler) all reside on a diskette in drive 1 and are called from a menu. The languages each load into the $9000 blocks and execute there. This is one of the first unusual things to notice about the system. No matter how large the interpreters are, they don't take more than 4K out of the memory map. This means that the lower 32K of 8032 memory is available to each of the languages for the program and variable storage. The facility for creating software that runs out of this paged memory is included in the assembler development system supplied with the SuperPET. This is one of the most significant contributions to the future software growth of this product.

The Manuals

On the subject of documentation, there are five manuals included in the system, one for each language. These manuals are tutorial examples. Of

course, since the languages are standards, many texts are available to help the beginner.

We mentioned libraries above. It has appeared that Microsoft and Commodore have gone out of their way in the past to hide their code entry points

This bridging of the tremendous gap between traditional computerists and micro users can only benefit both sides.

from legitimiate users. I am happy to say that this is not the case with the Watcom system. The entire operating system is jump table oriented and the table entries and setup conditions are described in the very complete manuals which accompany the system. I sat down and wrote a complete disk handler with error trapping and all in under 200 bytes using the available calls and documentation.

The greatest significance of the SuperPET is the perception people will have of it. To the micro user, it is a logical extension of the Commodore product line, but to the more traditional computer community, it is something entirely different. This is probably the first small computer at a reasonable cost that supplies the operating system and languages of larger systems and the communications interface to effectively use them. The impact of having the traditional computer community using this type of machine will be felt very rapidly in software availability. People used to programming in FORTRAN (or especially APL) will be able to put complex software systems, which have been used for years, on the micro. This bridging of the tremendous gap between traditional computerists and micro users can only benefit both sides. There are an awful lot of accumulated man hours of experience from which we should be able to profit. ©

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

Japanese Micros: A First Look

Andy Gamble
Columbia College, Vancouver, Canada

Surprise, surprise! The Japanese passion for things small and electronic has finally turned to the world of the computer. The expertise the Japanese brought to transistor radios, stereos, and — dare I say it? — cars, now moves into microcomputer technology. A familiar story?

Let me backtrack a little. I first visited Japan in spring 1980, and, although this was just a holiday, I couldn't resist investigating what they had to offer in the way of micros. (I think this particular disease has been termed "microholism.") The strange thing was that I couldn't find hide nor hair of the little beasts. It seems that personal computers were almost unknown in Japan then.

At last I found the Bit Inn in Tokyo: it wasn't that it was hidden, just low-profile. It seemed a typical computer shop by western standards: some PETs, Apples, TRS-80s. But what was that over there? Yes, a true Japanese micro, the Sharp MZ-80. And was that a machine made by NEC? More about these later.

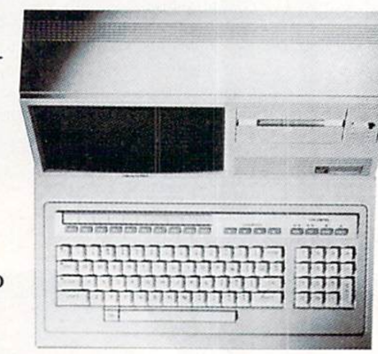
Try as I might, I couldn't find much further evidence of personal computers in Japan (I feel that I'm going to be contradicted on this) until my second visit a year later. By 1981, the whole story seemed to have changed. It wasn't that there were significantly more computer shops around, but most large department stores (and they get really large in Japan) now had their own micro department.

Here, then, is a brief rundown on some machines available today in Japan. The prices given are the result of a rough conversion of yen to dollars which, as we all know, is subject to change. It should also be noted that Japanese consumer products are often sold at a discount of up to 20 percent.

SHARP MZ-80B

Designers of personal computers should take a long, close look at the Sharp MZ-80B. Its modern silver-gray case contains a 10" monitor, a cassette drive with "logical" tape control, and a full-size keyboard. In a word, it looks really, ahem, sharp.

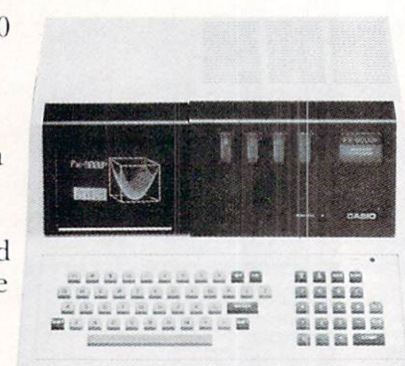
The Z-80A microprocessor runs at 4Mhz and can support up to 64K of RAM. High-resolution graphics capability enables the Japanese katakana symbols to be written. The cursor keys are exceptionally appealing. Ten user-defined function keys complement the keyboard layout. The machine is RS-232 and IEEE-488 compatible.



Apart from the usual BASIC, other languages and operating systems available include FORTRAN 80, COBOL 80, a BASIC compiler, FORTH and CP/M. The prices is around $1400. Sharp also manufactures the MZ80BP5 dot-matrix printer, an 80 cps machine selling for $700, the MZ-80CR card reader, capable of reading 150 cards per minute and the MZ-80BF dual mini-floppy disk drive with 572 Kbyte storage for $1500. A truly impressive system.

NEC PC-8000 Series

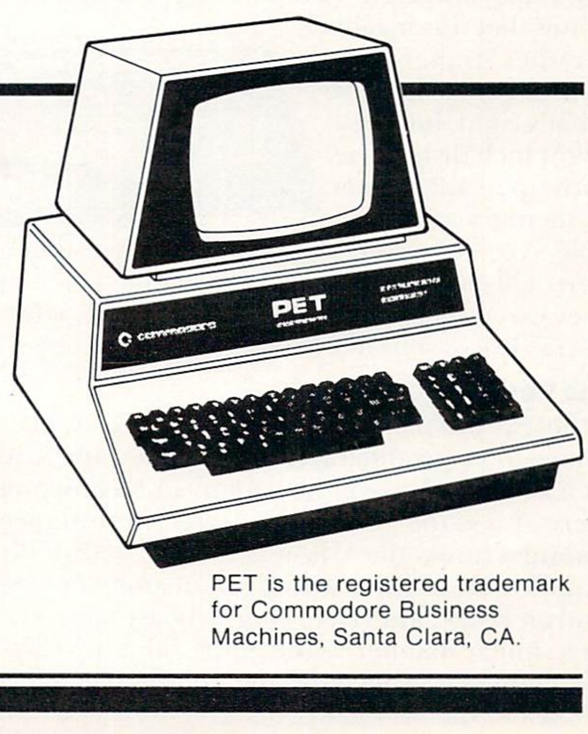
NEC is clearly aiming the $800 PC-8001 micro at the business market, but interest in the machine is very strong from the educational and scientific-professional fields. The PC-8001 supports high-resolution color — the monitor is $450 extra — and has perhaps the best color I have ever seen. The CPU is a PD780C-1, which apparently is similar to the Z80A, and runs at 4MHz. The Japanese seem to be big on function keys: the PC-8001 has five.



This series also includes the 8023 printer, a dot-matrix type at $900, and two dual disk drives of 286Kbytes at about $1300. Is the cheaper one an "expansion" disk drive? I couldn't find out.

CASIO FX-9000P

Casio is a well known name in North America for its range of calculators and watches, but in Japan also for computers and electronic musical instru-


PET

ments. The FX-9000P is another well designed machine with an integral monitor. The Z80A microprocessor runs at 2.75MHz and supports up to 32K RAM with high-resolution graphics. Most BASIC keywords are available from the keyboard with a single stroke of a key — a nice time saver. There are several statistical commands. BASIC itself, and expansion RAM (16K dynamic), is obtained by plug-in ROM-packs inserted into the front of the machine. The base price is $800, with the ROM-packs costing around $90 each.

Fujitsu Micro 8

Hold onto your hats. This machine is especially impressive. For around $1200 you get the following: Not one, but two microprocessors, both 6809s, which address up to 128K. Available RAM is only 32K, but that's because the resident BASIC is huge and color is supported with high-resolution graphics. Some idea of the power involved can be guessed from the graphic commands CIRCLE, CONNECT, SYMBOL and PAINT. UCSD Pascal and FLEX are also available.

For the Japanese, one of the delights this computer offers is the ability to reproduce the three different Japanese scripts: katakana (used for foreign words), hiragana, and kanji. When you realize that katakana and hiragana each contain about 50 characters, and a small, usable subset of the kanji (Chinese characters) would amount to several hundred characters, you cannot fail to be impressed. Normal English letters, numerals, and symbols are also there, of course.

Bubcom 80

Apart from the now usual Z80-based, 64K user RAM, 640x200 pixel high-resolution color display (optional) and eight function keys, I would have to admit that this machine is rather strange. It has 64K of bubble memory. That's right, bubbles. Eight inch disk drives with up to 1.2 Mbytes of memory are available. Are we looking at the future generation of personal computers here, with so much memory they can remember your shopping lists for 10 years? Prices start at $1800.

The Rest

Whereas you'd expect electronic concerns like Sharp to be producing computers, it came as a bit of a shock to discover how many hi-fi companies were also in the game. HITACHI, for instance, manufacturers the MB-6890, whose 6809 CPU can handle 32K of color with five function keys (again) and an RS232 interface. Plug-in boards are accepted in a similar manner to the Apple, all for $1500. The same series includes the MP-1040 printer at $900 and the MP-3540 dual disk drive at $1500.

TEAC produces the 4MHz Z80A-driven PS-85: the case for the monitor also contains the two disk drives. FORTRAN is available on this system.

SANYO makes several monochrome monitors, including the popular DDN-120C at $230.

As for "foreign machines," there's still a great interest shown in Commodore, Apple, and Radio Shack products. It's no secret that Commodore introduced the VIC into Japan first, where it's had time to create a strong following. Be on the lookout for some impressive VIC programs to surface from Japan. The VIC-20 is called VIC-1001 there, but the only difference that I can see is the yen symbol where the English pound sign is, and the katakana characters replacing one set of graphics characters on the keys. The VIC-1001 costs about $350 in its 5K form, more than holding its own against competitively priced Japanese models.

Exactly who buys these machines? No doubt the computer companies would like to say that scientists, professionals, and educators do: the advertising is certainly aimed at those people. And, true, it is not yet common for people to have computers for fun, or for their own erudition, as it is in North America. But then why is it that the software offered for sale includes so many games? The truth is that Japan is a nation besotted with video games (America is fast catching up), and this is reflected in the programs that are produced. Not just passable — *perfect* imitations of the popular video games are to be found for all the major machines. Most Japanese computers support the required high-resolution graphics and sound for this.

In The Coming Years

The Japanese have recently announced that their government and private sector will cooperate on a ten-year plan: the development of a "fifth generation" computer. The goal is the development of a machine which will perform with great power. It is hoped that the machine will have the ability to write its own programs, understand spoken human commands at a quite sophisticated level, and accomplish other tasks which are far beyond current technology. By "fifth generation," the Japanese seem to mean, essentially, artificial intelligence.

What impact will the present Japanese microcomputers have on world markets in the next few years? Will they manage to surge ahead in high-speed processor technology, in memory size and density, in artificial intelligence? Computerists around the world will be paying attention to the Japanese efforts in coming years.

LETTER QUALITY WORD PROCESSOR PRINTER/TYPEWRITER

FOR NEC, APPLE, TRS 80, COMMODORE, ATARI, H.P., OSBORNE 1

OLYMPIA ES100

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

REN TEC ES

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

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

RENAISSANCE TECHNOLOGY CORPORATION

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

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

DEALER INQUIRIES WELCOME

Perfectly Balanced

educational software

from

MICRO-ED

for

PET® and VIC®

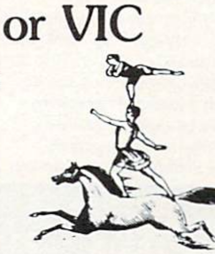
Send for our free catalog*

*please specify PET or VIC

MICRO-ED, Inc. • P.O. Box 24156

Minneapolis, MN 55424

or telephone us at (612) 926-2292



VIC and PET are the registered trademarks for Commodore Business Machines

KMMM Pascal for PET/CBM $85

A subset of standard Pascal with extensions.
- Machine language Pascal Source Editor with cursor oriented window mode
- Machine Language P-Code Compiler
- P-Code to machine language translator for optimized object code
- Run-time package
- Floating point capability
- User manual and sample programs

Requires 32K Please specify configuration.

EARL for PET (disk file based) $65

Editor, Assembler, Relocater, Linker

Generates relocatable object code using MOS Technology mnemonics. Disk file input (can edit files larger than memory). Links ,multiple object programs as one memory load. Listing output to screen or printer. Enhanced editor operates in both command mode and cursor oriented "window" mode.

RAM/ROM for PET/CBM

4K or 8K bytes of soft ROM with optional battery backup.

RAM/ROM is compatible with any large keyboard machine. Plugs into one of the ROM sockets above screen memory to give you switch selected write protectable RAM.

Use RAM/ROM as a software development tool to store data or machine code beyond the normal BASIC range. Use RAM/ROM TO LOAD A ROM image where you have possible conflicts with more than one ROM requiring the same socket. Possible applications include machine language sort (such as SUPERSORT), universal wedge, Extramon, etc

RAM/ROM - - 4K	$85
RAM/ROM - - 8K	120
Battery Backup Option	30

SUPERSORT by James Strasma $35

Supersort is an excellent general purpose machine language sort routine for PET/CBM computers. Sorts both one and two dimensioned arrays at lightning speed in either ascending or descending order. Other fields can be subsorted when a match is found, and fields need not be in any special order. Sort arrays may be specified by name, and fields are random length. Allows sorting by bit to provide 8 categories per byte. The routine works with all PET BASICs, adjusts to any memory size, and can co-exist with other programs in high memory.

SuperGraphics

by John Fluharty **$30**

SuperGraphics provides machine language extensions to Commodore BASIC to allow fast and easy plotting and manipulation of graphics on the PET/CBM video display, as well as SOUND commands.

Animations that previously were too slow or impossible without machine language subroutines now can be programmed directly in BASIC. Move blocks (or rocketships, etc.), or entire areas of the screen with a single, easy to use BASIC command. Scroll any portion of the screen up, down, left, or right. Turn on or off any of the 4000 (8000 on 8032) screen pixels with a single BASIC command. In high resolution mode, draw vertical, horizontal, and diagonal lines. Draw a box, fill a box, and move it around on the screen with easy to use BASIC commands.

The SOUND commands allow you to initiate a note or series of notes (or even several songs) from BASIC, and then play them in the background mode without interfering with your BASIC program. This allows your program to run at full speed with simultaneous graphics and music.

SuperGraphics commands include GRAPHIC, TEXT, RVS, SET, DRAW, FILL, PLOT, MOVE, PRINT, CSET, CMOVE, DISPLAY, PUT, SWAP, PAUSE, and SOUND.

Please specify machine type and ROM version, disk or tape.

FLEX-FILE

for PET/CBM Computers

Self Calculating
DATA BASE
REPORT WRITER
MAILING LIST

FLEX-FILE is a set of flexible, friendly programs to allow you to set up and maintain a data base. Print files with a versatile Report Writer or a Mail Label routine. Programmers will find it easy to add subroutines to their own programs to make use of Data Base files.

RANDOM ACCESS DATA BASE

Record size limit is 250 characters. The number of records per disk is limited only by the size of each record and the number of records per disk is limited only by the size of each record and the amount of free space on the disk. File maintenance lets you step forward or backward through a file, add, delete, or change a record, go to a numbered record, or find a record by specified field. The Find command locates any record when you enter all (or a portion of) the desired key. Field lengths may vary from record to record to allow maximum packing of information. Files may be sorted by any field, and any field may be specified as a key. Sequential files from other programs may be converted to Flex-File format, and Flex-File records may be converted to sequential (WordPro, PaperMate, other word processors may also use Flex-File data). Maximum record size, fields per record, and order of fields may be changed at any time.

MAILING LABELS

With typical record size of 127 characters, each disk can handle over 1000 records (about 2800 with 8050 drive). Labels may be printed any number wide, and may begin in any column position. There is no limit on the number or order of fields on a label, and two or three fields may be joined together on one line (like first name, last name, and title). A "type of customer" field allows selective printing.

REPORT WRITER

Print any field in any column. For numeric fields, use decimal point justification (and round to any accuracy). Define any column as a series of mathematical functions performed on other columns. These functions include arithmetic operations and various log and trig functions. Pass results of operations such as running total from row to row. At the end of the report, print total and/or average for any column. Complete record selection, including field within range, pattern match, and logical functions can be specified individually or in combination with other parameters.

FLEX-FILE by Michael Riley $60

Please specify equipment configuration when ordering.

Low Cost Disk Drive for PET/CBM

PEDISK II from cgrs Microtech is a new disk system ready to plug into your large keyboard PET/CBM.

PEDISK II offers speed, reliability, IBM compatibility. Complete system prices with DOS and cable:

5" 40 track, 1 drive, 143K	$525
5" 40 track, 1 drive, 286K	690
8" IBM 3740 format, 77 track, 250K	995

PROGRAM YOUR OWN EPROMS

Branding Iron for PET/CBM $79

EPROM Programmer with software for all ROM versions. Includes all hardware and software to program or copy 2716 and 2532 EPROMs.

CBM Software

Legal Time Accounting Package	445
Medical Accounting Package	
Complete CBM Business Software Package	
Can be tailored to meet most business requirements.	
Technician's Investment Analysis Package	500
Dow Jones Portfolio Management	129
Personal Tax Calculator	65
Tax Preparation System	445
Wordcraft 80 Wordprocessor Package	325
Pascal Development Package	235
Assembler Development Package	99
Intelligent Terminal Emulator	30
Softpac-1 (Competitive Software)	29

FORTH for PET

BY L. C. Cargile and Michael Riley **$50**

Features include:
- full FIG FORTH model.
- all FORTH 79 STANDARD extensions.
- structured 6502 Assembler with nested decision making macros.
- full screen editing (same as when programming in BASIC).
- auto repeat key.
- sample programs.
- standard size screens (16 lines by 64 characters).
- 150 screens per diskette on 4040, 480 screens on 8050.
- ability to read and write BASIC sequential files.
- introductory manual.
- reference manual.

Runs on any 16K or 32K PET/CBM (including 8032) with ROM 3 or 4, and CBM disk drive. Please specify configuration when ordering.

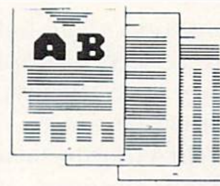
Available soon:

Metacompiler for FORTH $30

simple metacompiler for creating compacted object code which can be executed independently (without the FORTH system).

PaperMate
60 COMMAND WORD PROCESSOR

by Michael Riley



Paper-Mate is a full-featured word processor for CBM/PET. Paper-Mate incorporates 60 commands to give you full screen editing with graphics for all 16K or 32K machines (including 8032), all printers, and disk or tape drives.

For writing text, Paper-Mate has a definable keyboard so you can use either Business or Graphics machines. Shift lock on letters only, or use keyboard shift lock. All keys repeat.

Paper-Mate text editing includes floating cursor, scroll up or down, page forward or back, and repeating insert and delete keys. Text block handling includes transfer, delete, append, save, load, and insert.

All formatting commands are imbedded in text for complete control. Commands include margin control and release, column adjust, 9 tab settings, variable line spacing, justify text, center text, and auto print form letter (variable block). Files can be linked so that one command prints an entire manuscript. Auto page, page headers, page numbers, pause at end of page, and hyphenation pauses are included.

Unlike most word processors, CBM graphics as well as text can be used. Paper-Mate can send any ASCII code over any secondary address to any printer.

Paper-Mate functions with 16/32K CBM/PET machines, with any printer, and with either cassette or disk.

To order Paper-Mate, please specify configuration.

Paper-Mate on disk or tape 40.00

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

Hayden Software

Complex Mathematics	12.70
Engineering Mathematics	12.70
General Mathematics	12.70
MCAP:Circuit Analysis Program	21.00
Energy Miser	24.50

Jukebox Series for PET by L C Cargile
Excellent 4 part harmony music--write for list

Automated Simulations (EPYX) Fantasy Games

MICRO-REVERSI for PET by Michael Riley **10**
super machine language version of Othello

Tunnel Vision / Kat & Mouse by Michael Riley **10**
two excellent machine language maze programs

252 Bethlehem Pike
Colmar, PA 18915 **215-822-7727** **A B Computers**

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

Telecommunications. What Is It?

Michael E. Day
Chief Engineer, Edge Technology

Editor's Note: **COMPUTE!** *is pleased to announce Michael Day's new monthly column,* Telecommunications. *Mike is a recognized expert on this exciting aspect of computing. — RCL*

Telecommunications has a rather ominous ring to it, but, in reality, it simply means communication at a distance. Specifically, as the term is currently used, it means that some sort of equipment is required to allow the communication to occur.

Some time ago, before there was a personal computer or microcomputer, computers were very expensive. Because of this, only large companies could afford to buy and use them. Also, because of the very high cost, these buyers had to use the computer as much as possible in order to justify the cost of the computer.

Companies which were not able to utilize the computer all of the time found that they could *share* their computer with other companies who did not have a sufficient work load to justify purchasing a computer of their own.

At first, the information to be processed was hand carried to the computer site. This placed limits on the type and volume of work that could be shared. It became apparent that there was a need to communicate with the computer from a location away from the computer. It wasn't too bad when the computer was only a couple of rooms away, but from across town, existing equipment just did not work.

There was, however, a communications system in existence which had been built specifically to solve that very problem — the telephone system. Unfortunately it was designed for voice communications, not data communications.

Computers require binary (on/off) signals to communicate. The telephone system was designed to handle the continuously varying signals (analog) which make up voice signals. Because of this difference, the computer could not be directly connected to the phone lines.

The Solution

To help solve this, Bell Labs came up with a device that they called a MODEM. This allowed computers to be attached to the telephone network.

It was done by using a continuously varying signal (a carrier) that the telephone system could handle. Then, that signal is changed (MOdulated) by the computer's binary signals by a fixed amount.

At the other end of the phone line, another MODEM receives the signal and measures the change to convert (DEModulate) the signal back into the binary signals that the other, "listening" computer can understand.

The method of changing the carrier signal used is called FSK (Frequency Shift Keying).

When the computer is not sending anything, it is normally sending an ON (Binary 1) condition to the MODEM. When the MODEM hears the ON, it will transmit a signal at 1270 Hz. When the computer starts to send something it will send an OFF (Binary 0) signal to the MODEM. Hearing this, the MODEM will change the frequency it is transmitting to 1070 Hz.

At the other end of the phone line, the other MODEM does just the opposite: it hears the 1270 Hz signal that the first MODEM sent when it was transmitting the OFF signal and sends a corresponding OFF signal to its attached computer. When the first MODEM changes its signal to 1070 Hz, it recognizes this *shift* in frequency and changes the signal it is sending to its computer from an OFF condition to an ON condition.

The changing of the frequency continues as the computer sends a stream of OFFs and ONs to the other computer until it is done. It then reverts back to the continuous ON condition. This is done so that the other MODEM knows that you are still there, but you just don't have anything to transmit at this time.

This changing, or *shifting*, of the transmitted signal is why it is called frequency shift keying.

One And Two Way Talk

Sending in only one direction is called *simplex* communication. Simplex communication does not expect (and in fact does not even allow!) any communication in the reverse direction. Television is a form of simplex communication: you can watch it all you want, but you cannot directly communicate back to the station through your TV set (at least not yet) no matter how much you might want to at times.

So here we run into a problem. What if the

computer you are communicating with has to send something back? What if you requested that it send a program to you? Since your MODEM is still sending the 1070 Hz signal, the other MODEM can't send the same frequency — it would be drowned out by your signal. You could turn off your transmitter, allowing you to receive the 1070 Hz signal. In fact, some MODEMs do just that, but it requires that you know that you are finished sending and also have a way of turning your transmitter off. What's more, since only one computer can communicate at one time, there is no way of telling the computer that is transmitting to shut up.

Because of these limitations, this partial two way form of communication is called *half duplex.* This term is used because both computers can communicate. It is a dual communication, but only one side can communicate at any one time.

An easier way to solve the problem would be to allow both computers to communicate to each other whenever needed. A different set of frequencies is used for one of the computers to transmit with so that there is no conflict.

When Bell Labs designed the MODEM, they chose a set of frequencies that would minimize any interference with the first set. The frequencies they chose were 2225 Hz for the ONE or ON condition, and 2025 Hz for the ZERO or OFF condition. Since both computers can fully use the communications equipment at the same time, this is referred to as *full duplex.*

Since two different sets of frequencies are used, some means of deciding which MODEM should use which frequencies had to be determined. The original application was for a remote terminal to communicate with the big computer across town.

Talking Full Duplex

Normally, the big computer would be directly attached to the MODEM so that person who wanted to use it could call it up on the telephone and the big computer would automatically answer the phone. It was decided that the set of frequencies that were chosen for the big computer to use would be called the *answer* frequencies. Since the person who called the computer originated the call, the set of frequencies that were chosen for that side's MODEM are called the *originate* frequencies.

The frequencies that were chosen were based on the use to which the MODEMs were to be put. Since the person who called the computer would normally be monitoring the information sent by the computer, it was decided to assign the less reliable set of frequencies to the answer MODEM. The person monitoring it could always ask for the information to be sent again if something went wrong. Conversely, since a computer was not as capable of detecting errors in a transmission, it was decided to give the most reliable set of frequencies to the originate MODEM.

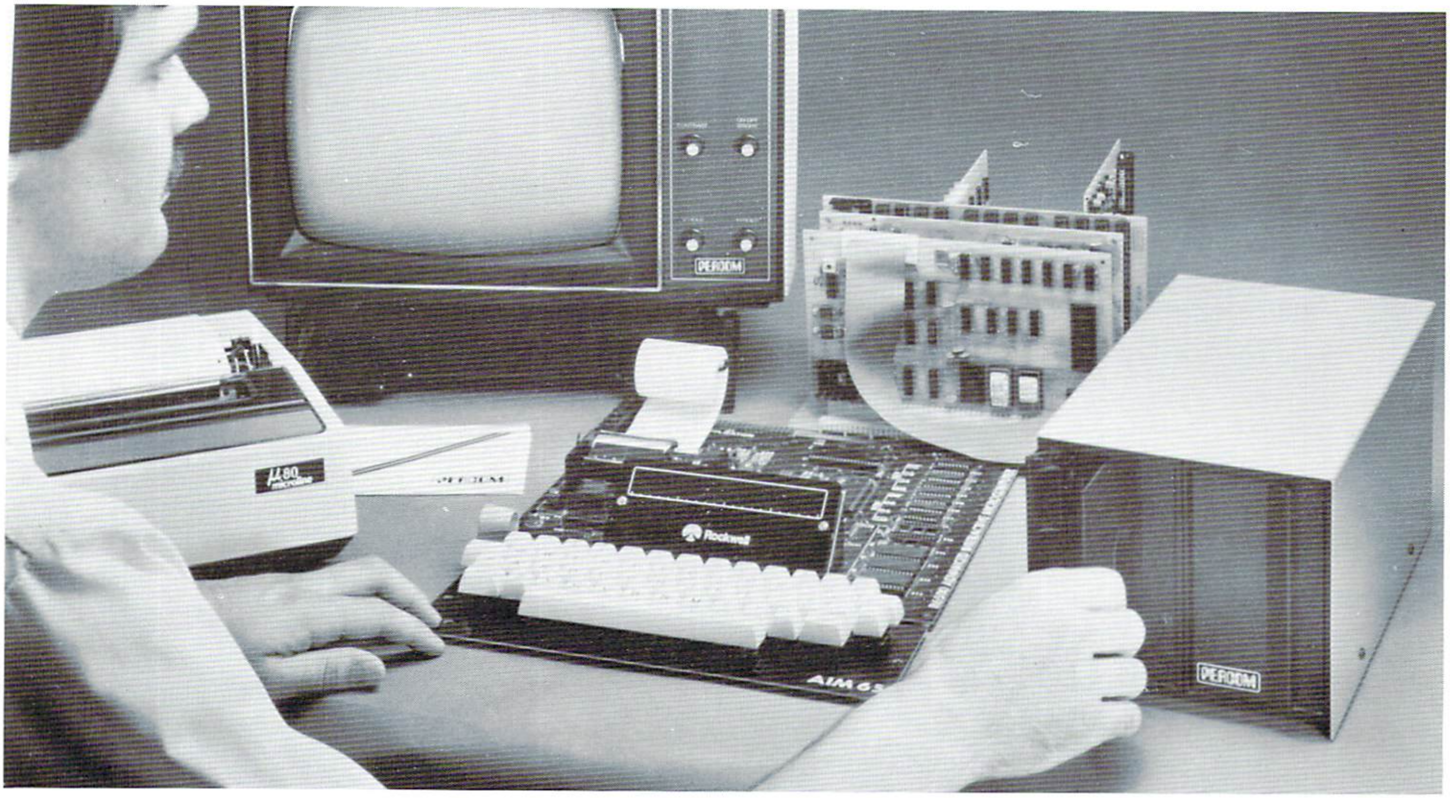
The frequencies chosen for the answer MODEM were 2025 Hz and the frequencies chosen for the originate MODEM were 1070 Hz and 1270 Hz.

Most acoustic MODEMs (sometimes referred to as acoustic couplers) are fixed as originate MODEMs. Some, however, have a switch on them which allows them to operate as an answer MODEM.

Direct-connect MODEMs (the type that is needed if automatic phone answering is desired) come in several different versions. One is *autoanswer only* (it will automatically answer the phone and connect itself to the phone line and then let the computer know that it has done so).

Another style that is available is the *autoanswer/manual originate.* This version allows the same operation as the previous version and, in addition, it can be used as an originate MODEM. Another type that is available is the *autoanswer/autodial* MODEM. This MODEM includes all of the previous capabilities and also allows the computer to make its own calls without operator assistance. Of course, this means that the computer must know how to do this: software must be in the computer to instruct it.

Next month we will explore *asynchronous* communications, another deceptively "ominous" word. ©



Introducing the M line . . .

Now! Drive Systems for AIM, KIM and SYM Computers – from PERCOM.

Assembled and tested systems start at only $599.95, including the drive controller circuit card, disk-operating system, interconnecting cable, drive and comprehensive users manual.

- **The right storage capacity** – Available in 1-, 2- and 3-drive systems, with either 40- or 80-track drives.
- **Flippy storage** – Flippy drives (optional) let you flip a diskette and store data and programs on the second recording surface.
- **High Storage Capacity** – Formatted, one-side storage capacity is 102 Kbytes (40-track drive), 205 Kbytes (80-track drive).
- **Proven Controller** – The drive controller design is the same as the design used in the Percom 680X LFD mini-disk system. This system – introduced in 1977 – has given reliable service in thousands of applications. Two versions are available: the MFD-C65 for the AIM-65 expansion bus, and the MFD-C50 for the System-50 (SS-50) bus.
    - Includes an explicit data separator circuit that's reliable even at the highest bit densities.
    - Provides for on-card firmware.
    - Includes a motor inactivity time-out circuit.
    - Capable of handling up to four drives.
    - Capable of reading both hard- and soft-sectored diskettes.
- **DOS included** – The MFD disk-operating system works with the AIM monitor, editor, assembler, Basic and PL/65 programs; interface is direct, through user I/O and F1, F2 keys.
- **Reliability assurance** – Drives are burned-in 48 hours, under operating conditions, to flag and remove any units with latent defects.
- **Full documentation** – Comprehensive hardware and software manuals are included with each system. These manuals cover details from design to operation and applications.

Now! Expand your AIM-65 with Low-cost System-50 Modules.

The Percom M65/50 Interface Adapter connects your M-65 bus to Percom's System-50 (SS-50) motherboard, allowing you to expand your AIM, KIM or SYM with proven System-50 modules. You can add disk storage, memory modules,even a video display system. The M65/50 provides buffer-amplification of address, data and control lines. On-card decode circuitry lets you allocate address space either to the computer or to the expansion motherboard. Price: only $89.95.

System Requirements: AIM-65, KIM or SYM computer with expansion bus and four Kbytes RAM (min).

PERCOM

PERCOM DATA COMPANY, INC.
11220 PAGEMILL RD. DALLAS, TX 75243
(214) 340-7081

Toll-Free Order Number: **1-800-527-1222**

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

Yes . . . I'd like to know more about Percom MFD drive systems. Rush me free literature.

Send to
PERCOM DATA COMPANY, Inc., Dept. 65-C
11220 Pagemill Rd. Dallas, TX 75243

name ______________________
address ______________________
city ______________ state ______
zip ______ phone number ______________

MAIL TODAY!

Editor's Note: Although some of the applications here are specific to PET computers, the analysis of the Boolean operators is valuable knowledge for any computerist. — RTM

Bits, Bytes And Basic Boole

David O. Williams
Toronto, Canada

Of all the operators which are included in Microsoft BASIC (or in most other implementations of the language) few cause as much confusion among novice programmers as the Boolean words OR, AND and NOT. What does the computer mean when you ask it to PRINT 3 OR 5 and it answers 7? How can 3 AND 5 equal 1? And what can be the possible use of something called NOT 3 which equals -4? In this article, I will describe the meanings and some of the uses of these operators.

I assume that most readers have some idea of what is meant by *binary numbers*. They are numbers which are written in a notation which has only two digits, zero and one. A single zero or a single one means exactly the same as in "regular" (i.e. decimal) numbers. However, to write the next number after one we have to start a new column, just as happens in decimal when we write the next number after nine. Thus, the first few whole numbers in both decimal and binary notations look as follows:

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000

and so on.

By examining this little table you can see that each column of a binary number is related to a power of two, in the same way that the columns in decimal numbers are related to powers of ten. The right-hand digit represents 2^0 which equals one. The next column represents 2^1 which equals 2. The third column represents 2^2 or 4, and so on. Thus, 110 in binary represents 0 times 2^0, plus 1 times 2^1, plus 1 times 2^2, which, in decimal notation, all adds up to six. Take a minute right now to check that all the other numbers in the table can be expressed as sums of powers of two in the same way, and that these powers are reflected in the pattern of 1's in the binary representation of the number.

There is a trivial point about all numbers, binary or decimal, which we ought to get clear before going on to look at the Boolean operators. The number 000101 is exactly the same as the number 101, in either notation. The zeroes before the first non-zero digit (called leading zeroes) are often not written, but in a sense they exist anyway. They represent the fact that the number in question includes zero times high powers of two (or ten). In discussing Boolean operations it is helpful to have the same number of digits in all the numbers in any example. We can do this by writing in some leading zeroes where appropriate.

The OR

Now let's look at the example of the Boolean OR operator mentioned in the first paragraph. 3 OR 5 equals 7. In binary form, with the numbers written beneath each other, it looks like this:

```
    011
 OR 101
    111
```

Here are a couple of other examples, each with numbers of four bits (that's just short for *B*inary dig*ITS*). Before reading on, see if you can spot the pattern and work out what the OR operator does. Hint: look at the ones in the numbers, and think of the English word OR.

```
    1011       0101
 OR 1001    OR 0011
    1011       0111
```

If you still haven't seen the pattern, look at each column in each example. There are no "carries" in Boolean operations, so the columns do not affect each other at all.

I hope you discovered that the answer to an OR operation has a one in each position where one OR the other (OR both) of the starting numbers had a one. That is all the OR operator does. It combines two binary numbers to give a third according to this rule. Perhaps you also noticed that the third example I gave you was the same as the first, except that the two starting numbers were in reversed order. Of course this had no effect on the result.

The AND

Now you should have no difficulty in deciphering the AND operation.

```
     101          10010110
 AND 011      AND 01010101
     001          00010100
```

The answer in each case has a one *if and only if* the first AND the second starting number had a one in the corresponding position.

As an exercise, I suggest you try thinking of some pairs of numbers (restrict yourself to fairly small positive integers) and predict what the results

COMPETENCY EXAM PREPARATION SERIES

This comprehensive set of programs consists of simulated exam modules, a thorough diagnostic package, and a complete set of instructional programs. It is designed to teach concepts and operations, provide drill and practice and assess achievement levels through pre and post testing. The Competency Exam Preparation Series provides a structured, sequential, curriculum encompassing mathematical, reading and writing instruction.

The C.E.P.S. program is designed for individual student use or use in a classroom setting. Programs provide optional printer capability covering worksheet generation and performance monitoring. C.E.P.S. are available in three software formats.

National Proficiency Series $1,299.00
N.Y.S. Regents Competency Test, Preparation Series $1,299.00
California Proficiency Assessment Test, Preparation Series $1,299.00

If desired separate Mathematics and Verbal packages are available for $799.00 ea. A Spanish language version of the Mathematics Instruction Package is available at no extra charge.

COLLEGE BOARD PREPARATION SERIES 81/82

for TRS-80 NORTHSTAR ™
PET, APPLE, OSI, ATARI

Each program confronts the user with a virtually limitless series of questions and answers. Each is based on past exams and presents material of the same level of difficulty and in the same form used in the S.A.T. Scoring is provided in accordance with the formula used by College Boards.

S.A.T., P.S.A.T., N.M.S.Q.T. - Educator Edition set includes 25 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, and Mathematics. Price $229.95

Independent Tests of S.A.T. series performance show a mean total increase of 70 points in students' scores.

G.R.E. Series - Educator Edition includes 28 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, Mathematics, Analytical Reasoning and Logical Diagrams. Price $289.95

ODYSSEY IN TIME

This spectacular adventure game adds a new dimension of excitement and complexity to **Time Traveler.** Players must now compete with the powerful and treacherous adversary in their exacting quest for victory.

To succeed they must vanquish this adversary in combat that rages across 24 time periods.

Odyssey In Time includes all the challenges of Time Traveler plus 10 additional eras, including those of Alexander the Great, Emperor Asoka of India, Attila the Hun, Genghis Khan. Each game is unique, and may be **interupted and saved** for later play.

available for APPLE & TR-80 PET, 32K - $39.95

ISAAC NEWTON

Perhaps the most fascinating and valuable educational game ever devised — ISAAC NEWTON challenges the players to assemble evidence and discern the underlying "Laws of Nature" that have produced this evidence. ISAAC NEWTON is an inductive game that allows players to intervene actively by proposing experiments to determine if new data conform to the "Laws of Nature" in question. Players may set the level of difficulty from simple to fiendishly complex.

In a classroom setting the instructor may elect to choose "Laws of Nature" in accordance with the complete instruction manual provided.

For insight into some of the basic principles underlying **ISAAC NEWTON** see GODEL, ESCHER, BACH by Douglas R. Hofstadler, Chapter XIX and Martin Gardner's MATHEMATICAL GAMES column in **Scientific American,** October, 1977 and June, 1959. $24.95.

TIME TRAVELER

Confronts players with complex decision situations and the demand for real time action. Using the **Time Machine**, players must face a challenging series of environments 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

Send $2.00 for complete Catalogue.

$5.00 Discount Coupon included in Catalogue.

PROGRAMS AVAILABLE FOR
TRS-80, APPLE II & PET
(unless otherwise indicated)

☐ disk or ☐ cassette (please specify)

Krell Software Corp.

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

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

NY State Residents Add Sales Tax

would be of ANDing and ORing them. Then test your predictions with your computer. Remember that the numbers will have to be in decimal notation for the BASIC interpreter in the machine to understand them. At a "deeper" level, however, the numbers are actually processed in binary form.

Perhaps the commonest use of Boolean operators is in what might be called "gentle POKEing".

The NOT

The NOT operation is even simpler, in binary notation, than the ones we have already looked at. There is only one starting number. Here are a couple of examples:

NOT <u>10010110</u>	NOT <u>01101001</u>
01101001	10010110

The digits in the answer are simply the opposites of the ones in the starting number. Each one becomes a zero, and each zero a one. Of course, doing this operation twice gets you back to your starting number. If you look closely at the two examples you'll see that they do just that.

A PET Complication

There is a complication, however, which has to do with the way the PET handles negative numbers. This is done by having an extra bit attached to a number, and this is either a zero or a one according to whether the number is positive or negative. When you NOT a number ALL the bits are reversed, including the sign bit. So the NOT of a positive number is negative, and vice versa. Zero is counted as a positive number, and its NOT is -1. In fact the following simple rule always applies:

NOT(X) = -(X+1)

Remember this rule as a curiosity, if you like. For practical purposes, it is more important to remember the fundamental fact that NOT reverses all the bits. Ones become zeroes and zeroes ones.

Only one more point needs to be considered before we can say we understand the Boolean operators, at least in theory. How are fractions handled? The answer is: they're not. If you ask the PET to PRINT 2.5 OR 5.1 the answer 7 will be returned. The fractional parts of the input numbers were simply dropped, just as if you had used the INT( ) operator. The PET does all the Boolean operations on 16-bit integers, including the sign bit. Fractions can't be fitted into this format, so they are dropped. Also, very large numbers, which would need more bits, cannot be used in Boolean operations. They lead to ILLEGAL QUANTITY ERRORs. In practice, this is not a serious restriction.

Programming With Booleans

By now, I hope, you feel you understand what the Boolean operators do, but you are probably still wondering whether they ever turn out to be useful in realistic programming situations. Indeed they do and, to illustrate this fact, we'll use the rest of this article to explore several uses.

The first of these, which is so simple as to be almost trivial, is a test which distinguishes between odd and even numbers. Every now and again (we'll see an actual example later) it is useful to make such a test on a number which crops up in the middle of some calculation. Of course there are many ways to do it, but none simpler than:

X=N AND 1

If N is odd, X will equal one. If N is even, X will be zero. The reason this works is simple if you think of binary numbers. The right-hand digit of an odd number is always one and, of an even number, is always zero. ANDing the number with 1 has the effect of looking only at this digit (a technique which is aptly known as *masking*), so the answer is zero or one, according to this digit.

Gentle POKEing

Perhaps the commonest use of Boolean operators is in what might be called "gentle POKEing". The POKE command, as I expect you know, forces a byte of memory (that's a set of eight bits) to be loaded with a number which the command contains. Thus the command:

POKE 59468,14

causes the byte at memory location number 59468 to be loaded with the bits 00001110. (That's fourteen in binary notation.) This command allows for no flexibility. Every bit of the number is exactly specified.

But sometimes we may not want to POKE all eight bits in the address. We may want one particular bit to be forced to become a one, but all the other bits should be left at whatever values they have already. For example, suppose we want the letter in the top left-hand corner of the PET video screen to be displayed in reverse field (black on white), but we do not want to change the actual letter from whatever it is already. The command:

POKE 32768,PEEK(32768) OR 128

does the trick. (Try it!). The command forces the eighth bit in the address ($128=2^7$) to become a one (remember the OR operator), but leaves all the other bits alone, so the letter is not changed. The eighth bit is the one which signifies reverse field, so making it a one has the effect we want. Of course,

now we have this character in reverse field, we may want to change it back again. We want to force the eighth bit to become a zero.

```
POKE 32768,PEEK(32768) AND NOT 128
```

is the command we want. This uses two Boolean operators. First the number NOT 128 is calculated. It has a one in every position *except* for the eighth bit, which is zero. Then this number is ANDed with the previous contents of the address. This forces a zero into the eighth bit, and leaves the rest unchanged. As an exercise, try predicting the effect of the following command, then try it several times in succession on your PET.

```
N=32768 : P=PEEK(N) : POKE N,((P OR 128) AND
NOT (P AND 128))
```

Manipulating the reverse-field bit in the video display makes a good demonstration of "gentle POKEing," and it is sometimes useful in programs. However, the technique is more often used in connection with some rather complex trickery to make the computer do these things which it ordinarily could not do. There are some addresses in a computer's memory which contain bits which act as internal *flags* which store information about the machine's status. Often several flags which refer to very different aspects of the computer's condition are grouped together in a single byte.

For example, for the PET, there is a byte which contains a flag which shows whether the Play key on the tape deck has been pressed, but the same byte also contains bits which select a particular row of keys on the main keyboard. Normally all these flags are looked after by the PET's internal routines, and the programmer does not have to think about them. However, advanced programmers sometimes find reasons to change the contents of a flag. This means that the flag is made to contain the "wrong" information, and the PET starts to behave in abnormal ways because of this. Sometimes this abnormal behaviour can be put to good use. Anyway, whatever its purpose, changing a single flag bit in a byte which also contains other flags which are to be left unchanged is a common use of "gentle POKEing."

A very different use of Boolean operations is to translate between the ASCII numbers of characters and the numbers which are used to represent these same characters in the PET's video memory. To demonstrate this, first try keying in and running the following little program:

```
10 GET G$ : IF G$="" THEN 10
20 POKE 32768,ASC(G$)
30 GOTO 10
```

When a program is running, if you press a key on the keyboard a character will appear in the top left-hand corner of the screen. However, this character will often not be the same as the one on the key you pressed. The program is poking the ASCII numbers into video memory, but these are often the wrong numbers. The video numbers are related to the ASCII numbers, but they are often not the same. The following little Boolean formula translates the ASCII numbers into the correct POKE numbers:

```
P=(A AND 63)+(128 AND A)/2
```

If you think about it, (remember that dividing by two shifts each bit in a binary number one place to the right) you will find that this formula has the effect of dropping the seventh bit of the number and of moving the eighth bit into the seventh position. This is exactly the way in which the PET changes ASCII before putting numbers into the video memory. You can incorporate this into the little program as follows:

```
10 GET G$ : IF G$="" THEN 10
20 A=ASC(G$)
30 P=(A AND 63)+(128 AND A)/2
40 POKE 32768,P
50 GOTO 10
```

When you run this program you will find that you always get the same character appearing in the top left-hand corner of the screen as the one on the key you pressed. This trick is sometimes useful in programming. It allows a message to be POKEd onto the screen without using the PRINT command, and thereby disturbing the position of the cursor.

The opposite trick, of translating screen characters into ASCII, can also be useful. For example, you might want to copy whatever is on the screen onto a printer. The printer wants ASCII, but the message, or whatever, is in screen characters. The translation formula has to generate the appropriate seventh bit of the ASCII number, which is not present in the screen representation. The following formula correctly translates every character except one:

```
A=(63 AND P)+2*(64 AND P)+2*((NOT P) AND 32)
```

The last part of the formula is the one in which the seventh bit is generated. The one character which it does not translate correctly is π. This character is a Commodore addition to the standard ASCII set, and is handled by special routines in the PET. If you encounter this character, you will also have to write special routines to translate it.

Fine Plotting

My last example of Boolean operations will be the process of "fine plotting" on the 40-column PET. (I have never tried this on the 80-column machine, but I assume that something similar can be done.) On the 40-column PET there are 1000 character positions on the screen, arranged in 40 columns and 25 rows. Suppose you want to draw (or have the computer draw) an abstract shape, such as a mathematical graph, on the screen. A simple way to do this is to set up a pattern in which a single

character, such as a reverse-field blank, is printed or POKEd into some of the 1000 positions. The overall pattern of these characters on the screen is arranged to have the appearance of the wanted shape. However, the small number of columns and rows gives a crude graph, and for this reason this technique is known as "coarse plotting."

A great improvement can be made by "fine plotting," which effectively increases the number of columns to 80 and the number of rows to 50. This is done by constructing the shape out of some of the PET's graphic characters, such as those on the comma, semicolon and question-mark keys. In each of these characters, the full-size character position (which I shall call the "frame") can be imagined as being divided vertically and horizontally into four quarters or "squares." In each character, some of the squares in the frame are white and the others black, where the word "some" means any number from zero to four. There are a total of sixteen characters which, together, cover all the possible combinations of white and black square in a frame. There are thus 4000 positions for squares on the screen and, by appropriately choosing and combining the 16 characters, any pattern of white and black squares can be generated. The process of fine plotting generates the pattern which most closely represents a wanted shape or graph.

A moment's thought shows that there are some interesting problems involved in doing this. Suppose you want to make a particular square white. You cannot simply calculate an address in the video memory and POKE it to some fixed number. Each address in the memory corresponds to a frame, and hence to four squares. The number to which the address should be POKEd must therefore take into account all four squares. Making one square white must not accidentally make any of the other squares change from white to black or vice versa.

The important clue in recognizing an elegant way of making the required calculations is the similarity between the requirements of fine plotting and those of "gentle POKEing." In each case there is a combination of items (squares in a frame, or bits in a byte) and we want to alter just one item while leaving all the others undisturbed. We saw earlier that the Boolean OR instruction can be used to force a single bit in a byte to become a one, and that AND NOT can be used to make a bit a zero. Perhaps, by putting the 16 graphic shapes into one-to-one correspondence with the 16 possible four-bit binary numbers, so that each bit represents the status (white or black) of a particular square, and by using Boolean operators on the numbers, we can work out a way of modifying the status of one square in a frame without disturbing the others.

Of course it can be done, but you may be surprised by how easily. Here are the necessary routines:

```
10 GOTO 1000
100 REM PLOT/UNPLOT ROUTINE
110 N=N1+40*INT(R/2)+INT(C/2)
120 M=(C AND 1)+2*(R AND 1)
130 POKE N,SC(F,SB(PEEK(N)),M)
140 RETURN
1000 REM INITIALIZATION
1010 DIM SA(15), SB(255), SC(1,15,3)
1020 FOR A1=0 TO 15
1030 READ SA(A1)
1040 SB(SA(A1))=A1
1050 NEXT A1
1060 FOR A$=0 TO 3
1070 A2=
1080 FOR A3=0 TO 15
1090 SC(1,A3,A1)=SA(A3 OR A2)
1100 SC(1,A3,A1)=SA(A3 AND NOT A2)
1110 NEXT A3,A1
1120 N1=32768
1130 DATA 32,126,124,226,123,97,255,236,108,127,
     225,251,98,252,254,160
```

When you RUN the above, it will initialize the arrays, but will do nothing visible on the screen. When it has finished, try entering:

```
F=0 : C=40 : R=25 : GOSUB 100
```

You should see a small white square appear in the middle of the screen. The value of F is a flag. If F=0 the routine will plot a point, i.e. make a square appear white. If F=1 the point will be "unplotted," i.e. made dark. The values of C and R are the coordinates of the point. C is the column number, which goes from zero at the left of the screen to 79 at the right. R is the row number, which is zero at the top of the screen and 49 at the bottom.

Try entering:

```
F=0 : FOR R=0 TO 49 : C=R : GOSUB 100 : NEXT
```

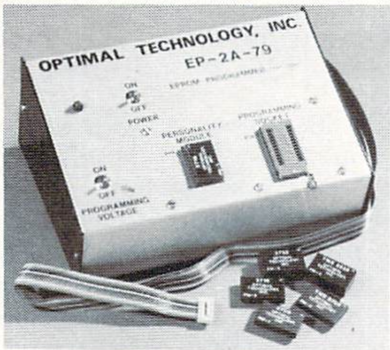
A diagonal line of little squares will be drawn across the screen. Enter the same thing but with F=1 (and without allowing the screen to scroll!) and the line will be neatly erased.

Write a program including these routines, and you will be able neatly and quickly to fine-plot any shape you want.

But how does it work? You work it out! There are lots of things in the routines which you ought to recognize from earlier in this article. In line 120 there are tests for odd and even numbers. In lines 1090 and 1100 there are the OR and AND NOT operators which we know are used to force bits to become one or zero. From line 1070, A2 is clearly a power of two. A little experimentation will show you that the numbers in the DATA line are the screen memory numbers corresponding to the sixteen characters which are used. Look at the order of these characters in the DATA and see how they are READ in lines 1020 to 1059. You will find that the shapes of the characters (i.e. the patterns of light and dark squares in the frame) are related to the bit-patterns in the corresponding (binary) values of A1. ©

Model EP-2A-79
EPROM Programmer

North Star · Apple · S-100 · SS-50 · STD-Bus · Atari · Pet · Kim-1



TRS-80 · H-8 · H-89 · Ohio Scientific · SWTP · Aim-65 · Sym-1

Three years in the field with unsurpassed performance. Software is available for the **EP-2A-79** for most all of the microcomputers including the popular CP/M, FLEX, HDOS operating systems. Write or call for specific hardware/software interfacing. Driver packages available for F-8, 6800, 6809, 8080,8085, Z-80, 1802, 6502 and 2650 based systems.

EP-2A-79 115V 50/60 HZ $169.00

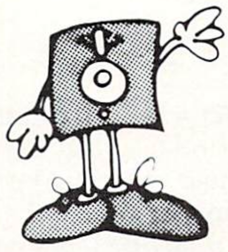
Personality Modules

PM-0	TMS 2708	$17.00	PM-5	2716, 2758	$17.00
PM-1	2704, 2708	17.00	PM-5E	2816	35.00
PM-2	2732	33.00	PM-8	MCM68764	35.00
PM-2A	2732A	33.00	PM-9	2764	35.00
PM-3	TMS 2716	17.00	SA-64-2	TMS 2564	39.00
PM-4	TMS 2532	33.00	SA-64-3	2764	39.00

Optimal Technology, Inc.

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

MEMOREX
FLEXIBLE DISCS



WE WILL NOT BE UNDERSOLD!! Call Free **(800)235-4137** for prices and information. Dealer inquiries invited and C.O.D.'s accepted.

PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401. In Cal. call
(800)529-5935 or
(805)543-1037

VISA

DISK DRIVE WOES?
PRINTER INTERACTION?
MEMORY LOSS?
ERRATIC OPERATION?

Don't Blame The Software!

Power Line Spikes, Surges & Hash could be the culprit! Pat. #4,259,705
Floppies, printers, memory & processor often interact! Our patented ISOLATORS eliminate equipment interaction AND curb damaging Power Line Spikes, Surges and Hash.

- **ISOLATOR (ISO-1) 3 filter isolated 3-prong sockets; integral Surge/Spike Suppression; 1875 W Maximum load, 1 KW load any socket $62.95**
- **ISOLATOR (ISO-2) 2 filter isolated 3-prong socket banks; (6 sockets total); integral Spike/Surge Suppression; 1875 W Max load, 1 KW either bank $62.95**
- **SUPER ISOLATOR (ISO-3), similar to ISO-1 except double filtering & Suppression $94.95**
- **ISOLATOR (ISO-4), similar to ISO-1 except unit has 6 individually filtered sockets $106.95**
- **SUPER ISOLATOR (ISO-11) similar to ISO-2 except double filtering & Suppression $94.95**
- **CIRCUIT BREAKER, any model (add-CB) .... Add $ 8.00**
- **CKT BRKR/SWITCH/PILOT (-CBS) .... Add $16.00**

AT YOUR DEALERS

Master-Card, Visa, American Express
Order Toll Free 1-800-225-4876
(except AK, HI, PR & Canada)

ESP Electronic Specialists, Inc.
171 South Main Street, Natick, Mass. 01760
Technical & Non-800: 1-617-655-1532

Micro Co-op

Micro Co-op is a software co-operative. We publish a bi-monthly newsletter describing various new and useful software products as they become available. We also try to give reasonable comparisons of existing products that help you make more educated buying decisions. Our members may purchase most software through us below retail price.

For more information, write for a free copy of our newsletter.

Micro Co-op
(Specify Apple or Atari) Newsletter
P.O. Box 432
West Chicago, IL 60185
(312) 231-0912

ATTENTION! AIM 65 USERS
AND OTHER 6500 MICRO PROCESSORS

Convert Your 65 Into A More Useful Tool With The New And Reasonable

VD640 VIDEO INTERFACE

Save Expensive Thermol Paper • Save Time • Save Money

FEATURES

- Uses 2K 2716 or 1K 2758 E Prom
- 1K Ram
- Pin compatible with AIM expansion connector. Can be pin adapted to fit Kim & Sim.
- Single +5 volt from connector.
- Reverse video & switch to reverse field.
- Upper & lower case characters.
- Uses standard T.V. with optional modulator.
- 7x11 Dot Format
- 40 characters X 16 lines.
- 32 special graphics characters.
- Software switch for programmable hardware screen flashing.
- Uses Motorola MC6845 CRTC
- Operating system in Eprom interacts with AIM monitor.
- Component layout silkscreened on P.C. Board.

Full operating instructions and schematic with each kit.

- ☐ VD640 Silkscreened P.C. board w/parts list only .... $ 39.00 each
- ☐ VD640 Kit-inc's. Silkscreened P.C. board with sockets, all components and Programmed Eprom. (P.C. Connector Board & mod. opt.) .... $ 119.00 each
- ☐ VD640 Fully assembled with sockets all components and connectors, tested and guaranteed 1 year. (modulator optional) .. 149.00 each
- ☐ Optional R.F. Modulator (allows use of std. T.V.) $39.00
- ☐ Optional P.C. Connector Board 15.00

Insert quantity wanted in box next to kit desired and mail with check or money order to: Sierra Pacific; 1112 Wellington Dr.; Modesto, CA 95350. (Calif Residents add 6% sales tax.) (Allow 4 to 6 weeks delivery.)

The Practical Side Of Assembly Language

Part II: Loops And Arrays

Bruce D. Carbrey, Raleigh, NC

In the last installment (**COMPUTE!** #14) I presented some ideas on how to represent and use flags in 6502 assembly language programs. This time I will discuss methods for programming loop control structures for the manipulation of arrays of data. Let's start by writing a loop which simply initializes all elements of an array to zero. In BASIC, you might write:

```
100 DIM AR(50)
110 FOR I=1 TO 50
120 AR(I)=0
130 NEXT I
```

If you are a neophyte assembly language programmer and try to translate this program segment on a line-by-line basis, you might wind up with something like this:

```
AR     *=*+  50     ;SPACE FOR ARRAY AR (50 BYTES)
I      *=*+  1      ;LOOP COUNTER VARIABLE I
       ...
       LDA   #1
       STA   I      ;INITIALIZE LOOP COUNTER
LOOP   LDA   #0
       LDX   I      ;RECALL CURRENT LOOP
                     COUNTER
       STA   AR,X   ;SET ELEMENT OF ARRAY TO 0
       INX          ;ADVANCE TO NEXT ELEMENT
       STX   I      ;STORE INDEX REGISTER
       CPX   #50    ;CHECK AGAINST LIMIT
       BNE   LOOP   ;REPEAT UNTIL DONE
```

If you run this program you'll be dismayed to find out that it only sets the last 49 elements of the array to 0 and skips the first element, because the first element of the array should be indexed with a zero, not a one.

Rule #7. To access the first element of an assembly language array, you should use an index of 0, not 1. The last element of an array of size N is indexed by N–1.

You may also recognize that it is not necessary to allocate space or save the variable I for the loop. Since it is only needed to control the loop, it can be simply kept in the index register (I chose the X register; the Y register will serve equally well). We can correct and improve the loop as follows:

```
AR     *=*+  50     ;SPACE FOR ARRAY AR (50 BYTES
       ...
       LDA   #0     ;CONSTANT TO FILL ARRAY
                     WITH
       TAX          ;X=0=INITIAL INDEX TO
                     ARRAY AR
LOOP   STA   AR,X   ;ZERO OUT ONE ARRAY
                     ELEMENT
       INX          ;ADVANCE TO NEXT ELEMENT
       CPX   #50    ;CHECK INDEX AGAINST LIMIT
       BNE   LOOP   ;REPEAT UNTIL DONE
```

You may notice some other subtle improvements in this program segment. The A register is only loaded with 0 once, outside the loop, since it does not change inside the loop. This will make the program run faster by eliminating 49 unneeded repetitions of the LDA #0 instruction.

Rule #8. Move code which does not need to be repeated out of the loop if possible.

Also note that one byte of code was saved by using TAX to initialize the X register instead of LDX #0. Naturally if we were filling our array with something other than 0, this trick won't work. We now have a correctly-functioning loop which is equivalent of our BASIC program (strictly speaking, is not exactly equivalent because the BASIC interpreter uses floating point arithmetic which uses four or five bytes for each array element instead of one).

Can our loop be further improved in terms of efficiency? Consider this alternative:

```
AR     *=*+  50     ;SPACE FOR ARRAY (50 BYTES
       ...
       LDA   #0     ;CONSTANT TO FILL THE ARRAY
                     WITH
       LDX   #49    ;INDEX TO LAST ELEMENT OF
                     THE ARRAY
LOOP   STA   AR,X   ;SET AN ELEMENT OF THE
                     ARRAY TO 0
       DEX          ;BACKUP TO PREVIOUS
                     ELEMENT
       BPL   LOOP   ;REPEAT UNTIL DONE
```

This code segment fills the loop backwards, filling the last element first and the first element last. Once the 0th element has been filled, the index register is decremented to -1 ($FF) and the BPL LOOP instruction will exit the loop. Notice that we have eliminated the CMP instruction from the loop, saving two cycles.

Rule #9. Moving backwards through an array will usually be more efficient.

If you try to make the array bigger than 128 elements you will be in trouble! Suppose you increase the dimension of AR to 200. In this case your loop will be executed only once because on the first pass, the DEX instruction will change the index from 199 to 198. But 198 has the hexadecimal representation $C6, which has bit 7 (the sign bit) set to 1. Therefore the 6502 will consider this a negative number (-58 decimal) and the BPL instruction will let control "fall through." Therefore, our BPL instruction will only work right up to +127 decimal, which is the largest *signed* 8-bit number. We can remedy this problem for index values up to 255 with a slightly more "tricky," but equivalent, method:

```
AR     *=*+  200    ;SPACE FOR ARRAY (200 BYTES)
       ...
       LDA   #0     ;CONSTANT TO FILL ARRAY
                     WITH
       LDX   #200   ;INDEX TO LAST ELEMENT OF
                     THE ARRAY+1
```

NEW! IBM COMPATIBLE 8" FLOPPY DISK for CBM/PET,* Aim, Sym

PEDISK II

LOW COST — HIGH PERFORMANCE FLOPPY DISK

5¼" MINI FLOPPY DISK SYSTEMS:

Model 540-1 Single Drive, Double Density (143K) $595.00
Model 540-2 Dual Drive, Double Density (286K) $895.00
Model 580-1 Single Drive, Quad Density (286K) $795.00
Model 580-2 Dual Drive, Quad Density (572K) $1,195.00

8" MINI FLOPPY DISK SYSTEMS:

Model 877-1 Single Drive, IBM standard (295K) $1,095.00
Model 877-2 Dual Drive, IBM standard (590K) $1,695.00

Payable in U.S. Dollars

PEDISK II is a high performance mass storage peripheral to enhance your computer's storage capability. Total storage to 850K bytes is available. The PEDISK II system consists of a small disk controller electronic board that mounts inside the computer and an external disk drive assembly. PEDISK II offers the fastest mass storage system available for the Commodore PET. With a data transfer rate of 250000K bits per second, the PEDISK II loads data directly to memory. This is up to four times faster than any 488 bus-type mass storage device.

One, two, or three drives connect to the PEDISK controller board. The user can choose 5¼" or 8" disk drives. An important feature of the PEDISK II system is its ability to exchange data with other computers. The standard 8" IBM 3740 format allows data exchange with large computers. Data files, mail lists, etc. available on large computers can be processed by the PET/PEDISK system. Remote PET/PEDISK computers can provide data entry for large machines.

PDOS II software links directly to the standard BASIC and operates with BASIC-type commands for easy interfacing. A full set of disk utility commands completes the powerful disk operating firmware.

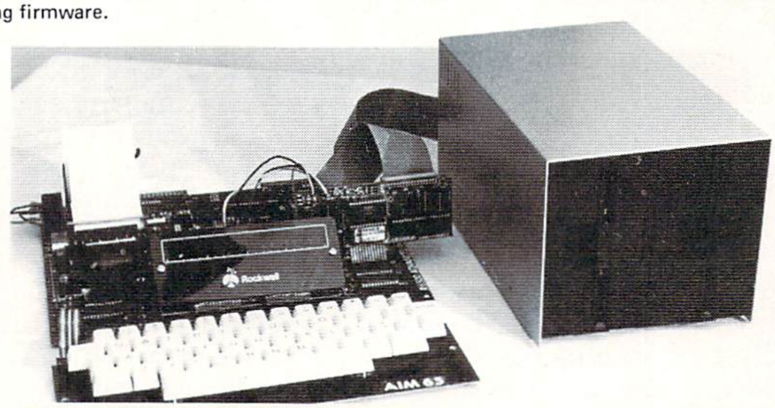
DISK COMMANDS:

BASIC

!LOAD — reads a program file to the computer
!SAVE — stores a BASIC program file to the disk
!OPEN — forms a sequential or relative data file
!INPUT — reads a data record from a file on the disk
!PRINT — stores a data record to a file on the disk
!CLOSE — ends a sequential or relative data file
!LIST — displays a directory of all files on the disk
!RUN — reads a program file and executes

MONITOR-DOS

D — displays contents of memory or diskette.
G — go to program and execute.
H — help user with listing of all commands.
K — kill a file on the diskette (erase file).
L — read program to the computer memory.
M — memory examine and change monitor.
N — name a file differently (rename).
P — print directory of all files on the disk.
R — return to BASIC mode.
S — save program or data from memory to the disk.
U — utility: format, copy, compress, patch diskette.
X — execute program after loading.



fullFORTH +

FULL FEATURE "FORTH" FOR 6502 SYSTEMS

INTERPRETER — can be executed directly in an interpretive mode to speed testing and debugging.

CROSS-COMPILER — words can be individually compiled and tested, the entire program can also be cross-compiled for maximum efficiency.

COND. ASSEMBLER — Machine language modules can be intermixed and conditionally assembled to fullFORTH.

STRING HANDLING — variable length constants and variables are allowed. Processes compare, move, concatenate and sub-string words.

FLOATING POINT — process 5 or 9 digit integer and floating point numbers for arithmetic operations.

SCREEN EDITOR — contains a unique full cursor visible screen editor.

SPECIFY PEDISK II, PET 2040 or 4040 DISK, OR APPLE $75.00

cgrs MICROTECH

DEALER INQUIRIES INVITED

FOR INFORMATION, SEE YOUR DEALER OR:

P.O. BOX 102 • LANGHORNE, PA 19047 • (215) 757-0284
U.S.A.

*PET IS A TRADEMARK OF COMMODORE

```
LOOP  STA   AR-1,X     ;SET AN ELEMENT OF THE
                        ARRAY TO 0
      DEX              ;BACKUP TO PREVIOUS
                        ELEMENT
      BNE   LOOP       ;REPEAT UNTIL DONE
```

Here we have replaced the BPL instruction with a BNE instruction so that the loop will terminate one pass earlier, but will not be stymied by index values greater than 127. Since our last pass through the array will now have an index of 1 instead of 0, we must compensate by changing the destination for our indexed STA instruction to AR-1. Therefore the last element set will be AR-1 + 1 = AR + 0. Finally, the starting index must be bumped from 199 to 200 for the same reason. Note that a starting index of 0 will clear a full 256 byte array.

The same technique can be used to move a block of data of up to 256 bytes from one known location to another:

```
ARA   *=*+  200        ;ARRAY A CONTAINING 200
                        ELEMENTS
ARB   *=*+  200        ;ARRAY B CONTAINING 200
                        ELEMENTS
      ...
      LDX   #200       ;NUMBER OF ELEMENTS TO
                        MOVE
LOOP  LDA   ARA-1,X    ;FETCH ELEMENT OF ARRAY A
      STA   ARB-1,X    ;INSTALL IN ARRAY B
      DEX              ;DECREMENT TO PREVIOUS
                        ELEMENT
      BNE   LOOP       ;REPEAT UNTIL DONE
      ...
```

What happens if you have more than 256 bytes? Throw away your 6502 and get a processor with a 16-bit index register? Nope. The indirect,X and indirect,Y addressing modes will solve this problem.

Rule #10. To use arrays of more than 256 bytes or arrays whose location is not known at assembly time, plan on using indirect,X or indirect,Y addressing.

Unlike the absolute, indexed addressing modes, indirect,X and indirect,Y are not equivalent. You may remember that indirect,X addressing uses pre-indexing and indirect,Y uses post-indexing. As a practical matter, indirect,X addressing will almost always be used with a permanent index of 0, simulating simple indirect addressing. This mode lends itself to manipulating large data arrays in non-time-critical portions of a program. For example, the following loop initializes a 1000-element array to 0:

```
ARRAY *=*+  1000               ;ROOM ARRAY OF 1000
                                ELEMENTS
PTR   *=*+  2                  ;POINTER TO AN ARRAY
                                ELEMENT
      ...
CLR1K LDA   #ARRAY&$FF         ;LOW 8 BITS OF ADDRESS
                                OF START OF ARRAY
      STA   PTR                ;INITIALIZE POINTER
      LDA   #ARRAY/256         ;HIGH 8 BITS OF AD-
                                DRESS OF START OF
                                ARRAY
      STA   PTR+1              ;INITIALIZE HIGH BYTE
                                OF POINTER
      LDX   #0                 ;PERMANENTLY LOAD
                                X WITH 0
LOOP  LDA   #0
      STA   (PTR),X            ;ZERO BYTE POINTED
                                TO BY PTR
      INC   PTR                ;BUMP POINTER UP TO
                                NEXT ELEMENT
      BNE   CHECK              ;BRANCH IF NOT
                                CROSSING PAGE
                                BOUNDARY
      INC   PTR+1              ;ELSE BUMP HI-ORDER
                                BYTE OF POINTER
CHECK LDA   PTR
      CMP   #ARRAY+1000&$FF    ;CHECK POINTER
                                AGAINST LIMIT
      BNE   LOOP               ;REPEAT IF NOT DONE
      LDA   PTR+1
      CMP   #ARRAY+1000/256    ;ELSE CHECK HI BYTE
                                OF POINTER
      BNE   LOOP               ;REPEAT IF NOT DONE
```

Some assemblers use the notation #‹ARRAY to mean the low byte of the address of ARRAY and #›ARRAY for the high byte instead of #ARRAY&$FF and #ARRAY/256. Clearly this program segment is quite a bit "messier" than the one for arrays of less than 256 bytes. When planning sizes for arrays, you should remember this and try to limit arrays to 256 bytes or less whenever practical.

Luckily, the indirect,Y addressing mode is considerably more powerful than indirect,X. For our final problem, let's use the indirect,Y mode to build a subroutine to move a large block of data from one place to another in memory as fast as possible. The source address, destination address, and number of bytes to be moved are to be specified as input to the routine as three 16-bit variables in page 0:

```
FROM  *=*+  2   ;POINTER TO STARTING ADDRESS OF
                 ARRAY TO MOVE
TO    *=*+  2   ;POINTER TO STARTING ADDRESS
                 OF DESTINATION
COUNT *=*+  2   ;NUMBER OF BYTES TO COPY
```

In an earlier example we saved execution time by removing the need for a compare inside the loop. We can apply the same principle to speed up our block move by sub-dividing the routine into two loops, one which moves entire pages (1 page = 256 bytes), and one which moves the final fractional page. This allows us to avoid any compares in the part which moves entire pages(which is part of the routine executed the most when copying large blocks). This will also let us use both 8-bit index registers to maximum effectiveness by allocating one for counting pages and index registers to maximum effectiveness by allocating one for counting pages and the other for indexing bytes within the page. The resulting routine (shown in Program 4) can easily be converted into a block-fill routine instead by removing FROM and all lines that refer to it, and presetting A to 0 (or the value with which to fill the array).

Rule #11. To deal with large arrays, split your program into two loops, one to operate on entire pages and one to operate on the "leftover"

fractional page.

The routine in Program 4 moves data at about 16.1 machine cycles per byte for large blocks, which means a 16K byte array can be moved in 0.26 seconds using a 6502 with a 1MHz clock. In certain applications where speed is of paramount importance, you may wish to improve even this super-fast copy routine. Can it be done? Yes, if you are willing to trade some increased program size for increased execution speed. Again, we employ the same gen-

Program 2: Keyboard Driver with Alpha Lock Flag Using 0 = False and Non-0 = True

Editor's Note: Part of this program was not printed in **COMPUTE!** *#14. we reprint it entirely here. — RTM*

```
                ;
                ;       SUBROUTINE INCH: KEYBOARD DRIVER FOR ASCII-ENCODED
                ;       KEYBOARD WITH PARALLEL INTERFACE.
                ;
                ;       ADDRESSES SHOWN ARE FOR 6530 ON KIM-1 COMPUTER.
                ;       KEYBOARD DATA LINES TO PORT A BITS 0 TO 6,
                ;       NEGATIVE-GOING STROBE TO BIT 7.
                ;
                ;       ON ENTRY: IF ALFALK IS NON-0, THEN ALL LOWERCASE LETTERS WILL
                ;       BE RETURNED AS THE EQUIVALENT UPPERCASE ALPHA.
                ;       ON RETURN: REGISTER A = ASCII CODE FOR KEY PRESSED;
                ;       X AND Y PRESERVED.
                ;
1700            PAD     =       $1700           ;KIM PORT A DATA REGISTER ON 6530
1701            PADD    =       $1701           ;KIM PORT A DATA DIRECTION REGISTER
                ;
0000                    *=      $1780           ;PROGRAM ORIGIN
                ;
1780 A900       INCH    LDA     #$00
1782 8D0117             STA     PADD            ;SET PORT DIRECTION = INPUTS
1785 AD0017     INCH1   LDA     PAD             ;TEST PORT
1788 30FB               BMI     INCH1           ;WAIT FOR STROBE PULSE
178A 2C0017     INCH2   BIT     PAD
178D 10FB               BPL     INCH2           ;WAIT FOR END OF STROBE
                ;
                ;       IF ALPHA-LOCK FLAG IS SET, FOLD ANY LOWERCASE LETTERS TO
                ;       EQUIVALENT UPPERCASE LETTERS.
                ;
178F 48         FOLD    PHA                     ;SAVE CHARACTER TEMPORARILY
1790 ADA317             LDA     ALFALK          ;RECALL "ALPHA LOCK" FLAG
1793 F00C               BEQ     FOLD2           ;BRANCH IF NO FOLDING DESIRED
1795 68                 PLA                     ;ELSE RECALL CHARACTER
1796 C97B               CMP     #$7B            ;LOWER CASE "Z" + 1
1798 B006               BCS     FOLD1           ;BRANCH IF PUNCTUATION
179A C961               CMP     #$61            ;LOWER CASE "A"
179C 9002               BCC     FOLD1           ;BRANCH IF NOT LOWER CASE ALPHA
179E E920               SBC     #$20            ;ELSE FOLD TO EQUIVALENT UPPERCASE
17A0 60         FOLD1   RTS
                ;
17A1 68         FOLD2   PLA                     ;RECALL CHARACTER
17A2 60                 RTS
                ;
                ;       ALPHA LOCK FLAG (DEFAULT = ALLOW LOWER CASE)...
                ;
17A3            ALFALK  .BYTE   0               ;"ALPHA LOCK" FLAG; NON-0=UPPERCASE ONLY.
                ;
0000                    .END

NO ERROR LINES
```

Program 4: Block-Move Memory Routine

```
GENERAL BLOCK-MOVE SUBROUTINE                         MTU 6502 ASSEMBLER 1.0

0000                     .PAGE   'GENERAL BLOCK-MOVE SUBROUTINE'
0000                     *=      0        ;ZERO PAGE ORIGIN
0000            FROM     *=*+    2        ;STARTING ADDRESS OF BLOCK TO BE COPIED
0002            TO       *=*+    2        ;STARTING ADDRESS OF DESTINATION
0004            COUNT    *=*+    2        ;NUMBER OF BYTES TO BE MOVED
0006            ;
0006                     *=      $2000    ;ORIGIN FOR PROGRAM
2000            ;
2000            ;        THIS ROUTINE COPIES A BLOCK OF ANY SIZE FROM ONE
2000            ;        LOCATION TO ANOTHER.
2000            ;
2000            ;        ON ENTRY: FROM (2 BYTES) IS THE STARTING ADDRESS OF
2000            ;        THE BLOCK TO BE COPIED; TO (2 BYTES) IS THE DESIRED
2000            ;        STARTING DESTINATION ADDRESS FOR THE COPY; COUNT
2000            ;        (2 BYTES) IS THE NUMBER OF BYTES TO COPY.
2000            ;
2000            ;        ON RETURN: NO REGISTERS PRESERVED; FROM, TO AND COUNT
2000            ;        ARE CLOBBERED.
2000            ;
2000            ;        NOTE: THE DESTINATION BLOCK MAY OVERLAP THE SOURCE
2000            ;        BLOCK ONLY IF "TO" IS AT A LOWER ADDRESS THAN "FROM".
2000            ;
2000 A000       BLKMOV   LDY     #0       ;INITIAL INDEX WITHIN A PAGE
2002 A605                LDX     COUNT+1  ;NUMBER OF PAGES TO BE MOVED
2004 F00E                BEQ     FRCMOV   ;BRANCH IF ONLY A FRACTIONAL PAGE
2006            ;
2006            ;        THIS LOOP COPIES ENTIRE PAGES...
2006            ;
2006 B100       PAGMOV   LDA     (FROM),Y ;FETCH A BYTE FROM SOURCE
2008 9102                STA     (TO),Y   ;COPY TO DESTINATION
200A C8                  INY              ;BUMP POINTER
200B D0F9                BNE     PAGMOV   ;REPEAT TILL ENTIRE PAGE MOVED
200D E601                INC     FROM+1   ;BUMP HI BYTE OF POINTERS
200F E603                INC     TO+1
2011 CA                  DEX              ;DECREMENT COUNT OF PAGES TO COPY
2012 D0F2                BNE     PAGMOV   ;REPEAT TILL ALL WHOLE PAGES COPIED
2014            ;
2014            ;        THIS LOOP COPIES THE FINAL FRACTION OF A PAGE...
2014            ;
2014 A604       FRCMOV   LDX     COUNT    ;RECALL NUMBER OF BYTES LEFT TO COPY
2016 F008                BEQ     DONEMV   ;BRANCH IF COUNT IS EXACT PAGE MULTIPLE
2018 B100       FRLOOP   LDA     (FROM),Y ;FETCH A BYTE FROM SOURCE
201A 9102                STA     (TO),Y   ;COPY TO DESTINATION
201C C8                  INY              ;BUMP INDEX
201D CA                  DEX              ;DECREMENT COUNT OF BYTES LEFT
201E D0F8                BNE     FRLOOP   ;REPEAT UNTIL DONE
2020            ;
2020 60         DONEMV   RTS
2021            ;
2021                     .END

  0 ERRORS IN PASS 2
```

eral principle of loop optimization:

Rule #12. To optimize loop execution speed, try to remove unnecessary compares from within the loop.

About the only way we can remove more compares from Program 4 is to "unwind" part of the loop and, instead, write some of the loop code "inline." Since we know the first loop will always move exactly 256 bytes, we can move two bytes at a time instead of one before checking for a page crossing:

```
        ...
PAGMOV  LDA  (FROM),Y  ;FETCH A BYTE FROM SOURCE
        STA  (TO),Y    ;COPY TO DESTINATION
        INY            ;BUMP POINTER
        LDA  (FROM),Y  ;FETCH A BYTE FROM SOURCE
        STA  (TO),Y    ;COPY TO DESTINATION
        INY            ;BUMP POINTER
        BNE  PAGMOV    ;REPEAT TILL ENTIRE PAGE
                        MOVED
        ...
```

This loop now takes 14.5 cycles per byte moved versus 16 cycles for the equivalent loop of Program 4, because the three cycle BNE instruction is only executed for every other byte moved. The loop can be unwound still further to move four, eight or more bytes per pass, but the speed improvement gained drops off rapidly as more code is written inline.

In the next installment I will explore some techniques for optimizing jumps and subroutine calls. ©

Part One:

Introduction To Binary Numbers

Charles Brannon
Greensboro, NC

To use machine language, or even to be truly computer literate, requires an understanding of binary numbers. The reason is simplicity — a computer can only understand two states, whether it is +5 or -5 volts, yes or no, or on or off. These simple relationships are expressed in a computer's world as merely a one or a zero. Because such a number has only two elements, one and zero, it is called binary. Mathematically, binary numbers are called base two numbers. We shall attempt to understand the computer more fully in this mathematical way.

The numbers we commonly use, whether we call them integers, counting numbers, or real numbers, are understood to be in base ten. Sometimes base ten is called *decimal* from the Latin word *decem*, meaning ten. We can look at any number on a digit-by-digit basis:

5	3	2	7
1000	100	10	1
10^3	10^2	10^1	10^0

So the number 5,327 can be expressed as $5x10^3+3x10^2+2x10^1+7x10^0$. Remember that any number raised to the zero power is one, so that we have $5x1000+3x100+2x10+7x1 = 5000+300+20+7$, which totals 5,327. It is simply a matter of multiplying each digit by a power of ten. For other number bases, only two things change — the base itself, and the number of digits used. Let's say we have 4302, but in base *five*. It can be shown as being equal to $4x5^3+3x5^2+0x5^1+2x5^0 = 4x125+3x25+0x5+2$, which, when totaled, is equal to 581 in base ten. It can be seen that 10 (pronounced ONE-ZERO) is equal to five. In fact, 10 is always equal to the base itself. Therefore, 10 in base two must be equal to two. Hey! There went our first binary number. In base five, the only digits are (0,1,2,3,4). The digit 5 is not present because it is represented by 10 (remember?). When we jump all the way down to base two, the only digits we'll have are zero and one. That's just what a computer needs. Any number can be converted to decimal in the same way we did for that base five number. Let's take the binary number 1101. It can be expressed as $1x2^3+1x2^2+0x2^1+0x2^1+1x2^0=8+4+0+1=13$. Because we are always multiplying by either a zero or a one, we really only have to "sum up" the ones to get the value. For example:

0	0	1	0	1	1	0	1
128	64	32	16	8	4	2	1

This would give us 32 + 8 + 4 + 1, or 45. Remember, we skip the zeros. The numbers at the bottom could be extended as far left as necessary. Just multiply the current value by two to get the next one.

On most microcomputers, the numbers have only eight digits. Each digit is called a *bit,* which is short for Binary digIT. Eight bits together comprise a *byte.* Since numbers are stored in "little boxes" called *memory locations,* the memory size of your computer now means something. If you have a 16K computer, that means it has roughly 16,000 of these boxes. Each box can store one byte. Since any character (the letter "a" or the number "9") can be stored as a number from zero to 255 (the highest number that can fit in eight bits), memory is often referred to as "characters" of storage. Therefore, if your computer can display 25 lines of 40 characters for a total of 1000 characters, it would take 1000 bytes to store one screen of information.

You should now be able to convert binary numbers to decimal. Now we'll work on going the other way. Basically, the trick is to break the number down into the powers of two. 61 probably has a 32 in it, but not 64 or 128. If it also has 16, then we have 32 + 16 = 48. Subtracting 48 from 61 gives us 13. Of the last possibilities (8,4,2,1), we choose 8 + 4 + 1 = 13. Therefore, we can now total 32 + 16 + 8 + 4 + 1 to get 61. Now we "fill in" the bits to form the binary number.

BITS	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**
POWERS OF TWO	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**

We put a one above the powers of two we used, and a zero above the rest. So now we know that $61_{10} = 00111101_2$.

The previously mentioned method will give you a feeling for how binary numbers work, but it is sometimes easier to use the "division method" to convert a decimal number to binary.

The Division Method

	Number	Remainder
1. Write down your number.	37	
2. Divide it by two.	18	
3. Write down the remainder.		1
4. Continue...	9	0
	4	1
	2	0
5. When you get to one, two can go into one zero times, with a remainder of one.	1	0
	0	1

6. Read the remainders from the bottom up. We have: 100101

7. You can check the number:

0	0	1	0	0	1	0	1
128	64	32	16	8	4	2	

32 + 4 + 1 = 37.

This method takes the guessing out of conversion. However, the easiest way of all is to use your computer to do the conversion. A short program is included at the end of this article which will convert a decimal number to binary. Because it is written to run on any BASIC-speaking computer, you may want to modify it and add any special features unique to your computer.

To reinforce your knowledge, I strongly suggest that you do the exercises included at the end of this article. (Without your computer!) Next month, we'll get into working with these binary numbers — adding and subtracting them.

Exercises

1. Convert to decimal:

a) 10101 b) 110011

c) 0111100 d) 11111111

2. Convert to binary

a) 52 b) 234

c) 66 d) 15

3. Extend the chart to 16 bits:

— — — — — — — — 128 64 32 16 8 4 2 1

```
1ØØ REM  TO CONVERT A DECIMAL NUMBER
11Ø REM  TO BINARY
12Ø REM
13Ø PRINT "ENTER THE DECIMAL NUMBER:"
14Ø INPUT D
15Ø FOR I=7 TO Ø STEP -1
16Ø P=2↑I
17Ø IF INT(D/P)=1 THEN PRINT "1"; : D=D-P: GOTO 19Ø
18Ø PRINT "Ø";
19Ø NEXT I
2ØØ PRINT
21Ø END
```

©

THE apple® GAZETTE

Editor's Note: In February of this year we ran the following Reader's Feedback plea:

> I am a high school science teacher. I am a novice Apple Computer programmer. I would appreciate **COMPUTE!** articles designed to enhance the programming ability of novice Apple programmers... In-depth articles of Apple POKEing, PEEKing, and CALLS would be very helpful....

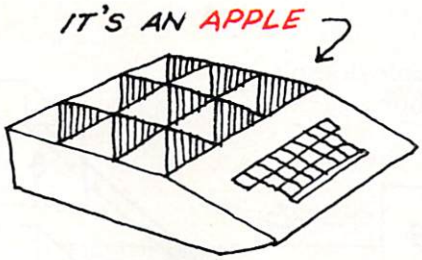
In response, we received the following from Gary who Kathleen and I had the pleasure of meeting at this year's West Coast Computer Faire. Gary, 11, gave us permission to run the response as an article. We think it's an excellent piece for beginners. — RCL

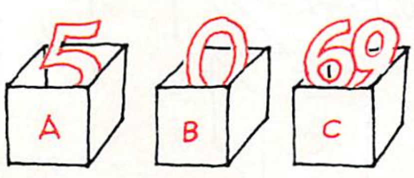
An Apple Primer

Gary Lin
San Jose, CA

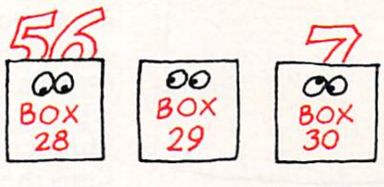
Having trouble with PEEK's, POKE's, and CALL's? Here's a rough explanation:

1. Imagine the memory of an Apple is divided up to a bunch of boxes.

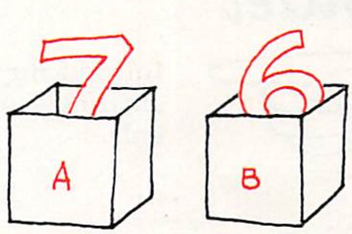




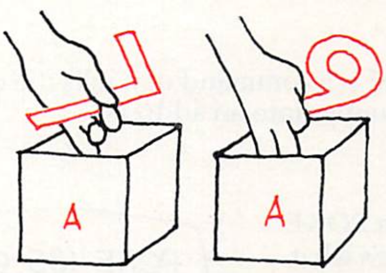
The Apple stores numbers in those boxes. Each box can have only one number assigned to it. Each box has its own personal address.



The Apple, like a mailman, gets a number and delivers it to the box. In this example, Box 29 gets the number 5.



The Apple stores numbers in the boxes of its memory. Okay, suppose the Apple sends a 9 to Box A. Box A holds the number 7.



The Apple takes out the 7 and throws it away. Now Box A is clear.

Then it puts a 9 into Box A.

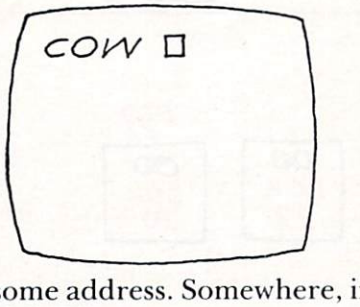
In reality, there are no boxes. Instead there are addresses. Addresses, like boxes, can be assigned a number. For example, address 2 may hold the number 15. The Apple's addresses are numbered in hex, a complicated numbering scheme*. Only the Apple understands HEX, and humans need to know the decimal equivalent.

Don't worry about hex, most beginners say, "What? hex, are you kidding?"

*Base 16

Okay, each address holds a number. If the Apple didn't assign a number to an address, the address automatically holds a 0.

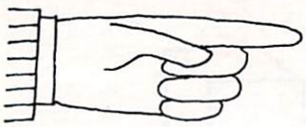
Suppose you type, in BASIC, "COW":



The Apple does a long process and sticks "COW" into its memory. It converts "COW" (or whatever you type in) into little numbers and assigns the numbers to some address. Somewhere, in an address, is "COW."

Well, you can do it a different way!

The Secret

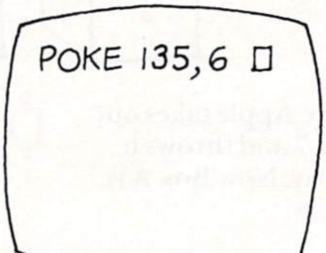


Introducing the amazingly, one and only,

POKE!

Okay, POKE is a command that tells the computer to stick a number into an address.

Let's type in POKE 135,6. Here's what the computer does: It converts 6 into hex and runs over to the address. Then 6 is placed in.



POKE A, B

A is the address (decimal) where you're going to stick a number. B is the number. Try it yourself!

You tell me. What do I do with some number in some address. Here's the next biggie:

PEEK!

Suppose ten is stored someplace, maybe address A. We want to know what is in address A. So we type

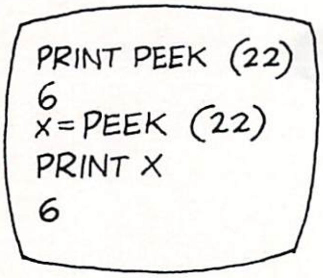
"PEEK (100)" (100 is the address for A). The computer figures out what 100 is in hex and goes to that address and picks out the number stored there.

It runs back and converts the number to decimal. To show what is at 100, we PRINT PEEK (100) and it'll print it.

PRINT PEEK (A)

need to show the number at address.

A is the address where you want to know what's there.

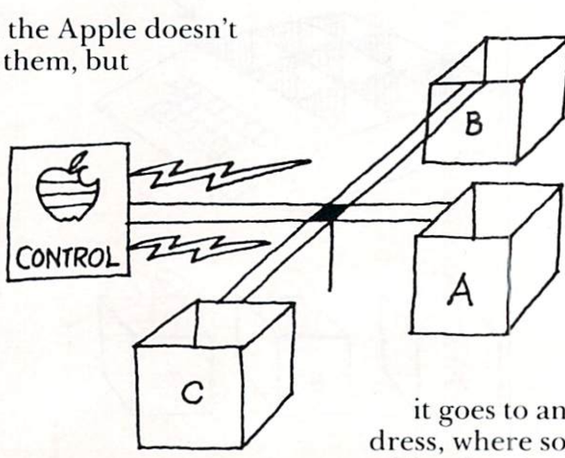


You can assign to a variable (another address, actually) the number which is at location 22 (decimal).

The last, but not least:

CALL!

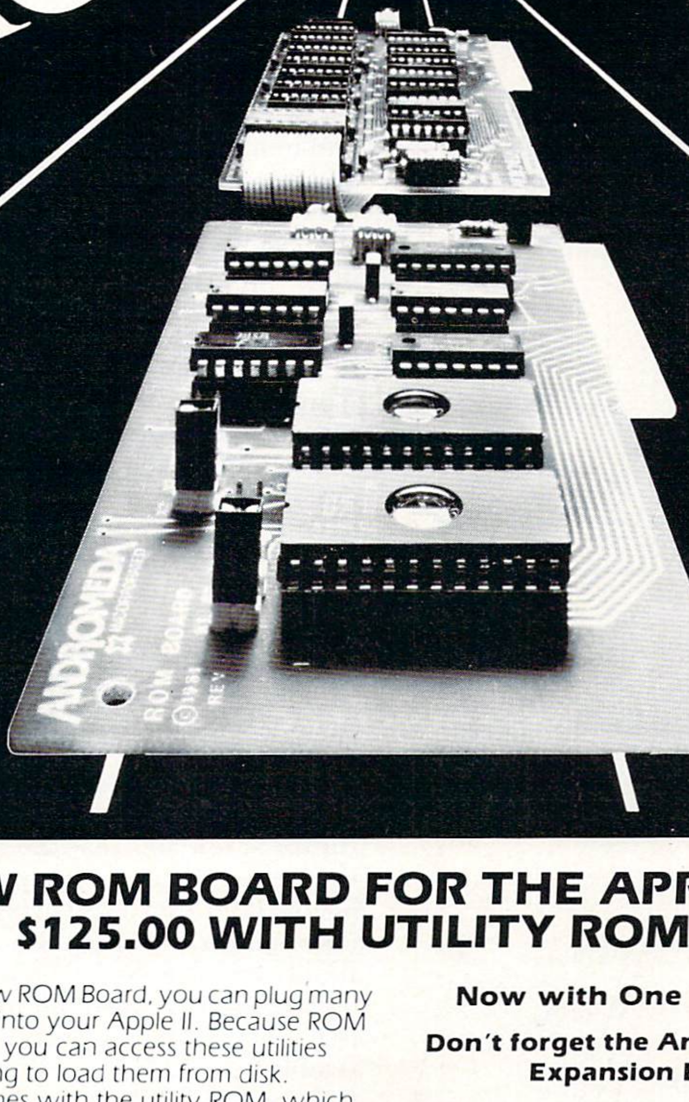
No, the Apple doesn't call them, but



it goes to an address, where something lies, (usually a program) and starts RUNning itself.

A BRIGHT NEW STAR FROM ANDROMEDA!

ROM ☆ RAM



NEW ROM BOARD FOR THE APPLE II*
$125.00 WITH UTILITY ROM.

With Andromeda's new ROM Board, you can plug many useful utility programs into your Apple II. Because ROM memory never forgets, you can access these utilities instantly without having to load them from disk.

The ROM Board comes with the utility ROM, which gives you five powerful options to apply to your Applesoft* programs. With the Utility ROM, you can do automatic line numbering, control a program list with a page mode, restore a crashed Applesoft* program in memory, alphabetize a disk catalogue and create a disk without DOS, giving you an extra 8K on your disk. Any of Soft Control Systems' other ROMS can be used, such as the Dual DOS in ROM, and 'Your'ple ROM.

You can install 2K PROMS, 4K PROMS, or even 2K RAM chips in each of the two memory sockets. So you can even have the Read - Write capability of RAM to develop PROM Programs yourself, or just have an extra 2K RAM for your machine - Language programs. Two 2732 PROMS allow a total of 8K of memory on the Board.

Now with One Year Warranty.

Don't forget the Andromeda 16K RAM Expansion Board $195.00

ANDROMEDA

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

Distributed By:

COMPUTER DATA SERVICES
P.O. Box 696
Amherst, NH. 03031
603 673-7375

*Apple II and Applesoft are trademarks

You say, "that's nice, but what's so big about it?"

The Big Trick of

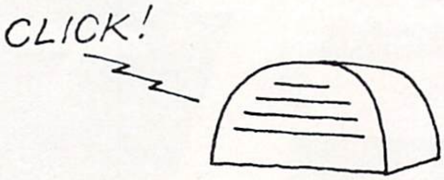
PEEKs, POKEs, and CALLs

Certain addresses in the Apple do nice things, depending on what's stored there.

Like POKE 50,127 (Type it in!)

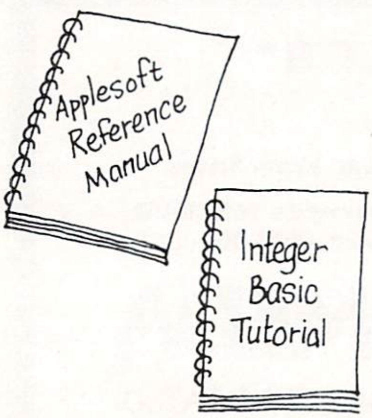
The computer sees 127 is in location 50 and the built-in command tells it to do something. What it does depends on the value stored. Try POKE 50,63 and POKE 50,255. You see, address 50 tells the computer to do something. PEEK does the same, but PEEK doesn't stick a value in — it just activates whatever is at that location.

Like PEEK (-16336)
(Listen carefully!)



CALLs also do stuff like CALL -936 clears the screen. CALL -151 enters Monitor. There are hundreds of POKEs, PEEKs, and CALLs that do special things. That's why people love them.

To find out more, look in these books:



and

I hope this helps you understand it better, although it's not much of a primer. If you have any questions write me:

Gary Lin
1598 Lock Lomond
San Jose, CA 95129

©

Page Flipper: Five Hires And Four Lores Pages For The Apple

Richard Cornelius
Department of Chemistry
Wichita State University

Five high resolution pages? Four pages of text or low resolution? The facility to copy, overlay or xcopy from one page to another? Yes, all of these and more are available on the (48K) Apple II Plus, and here is the program, PAGE FLIPPER, which demonstrates their use.

Simple arithmetic tells us that space is available on the 48K Apple for more than two high resolution pages. Each hires page occupies 8K (8192 bytes) of memory. Let's digress for a moment to see why that much space is required. The resolution on the Apple is 280 dots across by 192 dots high, which means that the Apple must store information regarding 53760 dots. Each dot is controlled by one bit, so we need 53760 ÷ 8 = 6720 bytes to record all of the on/off information for all of the dots on the screen. In addition, we need to record information about the color of the dot.

On the Apple screen, the colors in a horizontal series of seven dots are controlled by a single color control bit. If the color bit is off, then the colors in the seven dots can be any of those given by HCOLOR values of 0 to 3, depending upon the locations of the dots. When HCOLORs 4 to 7 are selected, the color bit is on. The on/off control bits for the seven dots plus the color bit make up one byte. Since each byte controls only seven dots, we need 53760 ÷ 7 = 7680 bytes. Some space in the 8K reserved for each page is not used, but there is simply no way to store all of the necessary information in, say, 6K. Even the space on each hires page that is not used to store graphics information cannot readily be used for other purposes because it is fragmented. After every set of 120 bytes that is used for information storage, there follows a set of eight bytes that is not used. Thus 512 bytes of unused space on each high resolution memory page is divided into 64 pieces of eight bytes each.

If we use 8K of memory to store the information on a single hires page, then an Apple with 48K could store enough information for six pages. Since we need to leave some room for a program, we must limit ourselves to five pages. Hires page one is located beginning at 8K and continuing up to 16K, and hires page two occupies memory from 16K to 24K. These two pages are the ones that are readily accessible through the Applesoft commands HGR and HGR2. Pages three, four, and five can be defined to begin at 24, 32, and 40K.

The area in memory to which the HPLOT and DRAW commands write is controlled by a POKE to position 230. POKEing a 32 specifies page one, 64 says page two, and 96, 128, and 160 are used to direct writing to pages three, four, and five. To see that this works, first scroll to the bottom of the Apple screen and then enter the HGR command. Now tell the computer HCOLOR=1 and HPLOT 0,0 to 279, 191 to put a line on the screen. Next POKE 230,64 to direct plotting to hires page two, set HCOLOR=2 and PLOT 279,0 to 0,191. No line appears on the screen because you are viewing page one and the plotting appeared on page two. If you POKE –16299,0 you will switch the Apple to display page two and presto, you see the second line that you plotted.

The Flipper

Unfortunately, we cannot so simply switch to see the other hires pages. Instead we must actually move the information on these pages down to page one or two so that it can be displayed. To accomplish this, we use a short machine language program for speed. This machine language routine is given in Program 1. You don't need to understand any machine language in order to use this little program because it is entered into memory by POKE statements in PAGE FLIPPER, it is executed by a CALL 768, and its function is controlled by POKE statements. Depending upon the POKEd values the routine can a) erase a page, b) copy information from one page to another, discarding the information originally on the destination page, c) overlay a page onto a different one so that the images from the two pages are superimposed, or d) "xcopy" the contents of one page onto another. "Xcopy" is most easily described as being analogous to the XDRAW routine which handles shapes in Applesoft. If you XDRAW a shape on top of some existing image, you get a composite of both the first image and the shape. If you XDRAW again, the shape disappears and you are left with only the original image. In PAGE FLIPPER, if you "xcopy" the contents of one page onto another and then xcopy it again, you are left with the original image also. For those interested in the machine language, "xcopy" uses an exclusive or (EOR) while overlay uses an inclusive or (ORA).

PAGE FLIPPER can also manipulate the pages of memory which store text or low resolution (lores) graphics. Just as the Apple has two hires pages, it also has two text/lores pages which begin at 1K and 2K. Much less information is required to store the letters that appear on the screen than is needed to store a screenful of hires graphics, so each text/hires page occupies only 1K of memory. Since the text screen offers 40 characters across and 24 down, there are 960 "boxes" where characters can be displayed. The contents of each little box is controlled by one byte of memory, so 960 bytes are all that is needed.

As is the case for the hires screens, the unused memory within the 1K allocated for a text/lores page is fragmented into many 8-byte pieces. The image on the lores screen corresponds to the same information as the text screen, but the image displayed is different when the Apple is in lores mode. Each byte which specifies a character on the text screen determines the colors (COLOR 0 to 15) of two blocks on the lores screen which occupy the same screen location as the corresponding character. Four bits (a nibble) determine the color of the upper block, and four bits determine the color of the lower block. In PAGE FLIPPER page three of text/lores is at 3K and page four is at 4K. Page four is used only to save the instructions. A schematic map of memory usage in the program is given in Figure 1. When the machine language routine is used to move any of the text/lores pages, it has less to move than when it operates on any of the hires pages, so another POKE statement is used to specify the size of the page that is being moved.

In addition to the POKEs used to adapt the machine language routine to different purposes, POKE statements are also used to control the display mode of the Apple. These POKEs are outlined in the Apple manuals. All of the POKE positions used in PAGE FLIPPER are listed in Table 1.

Easily Moved Pages

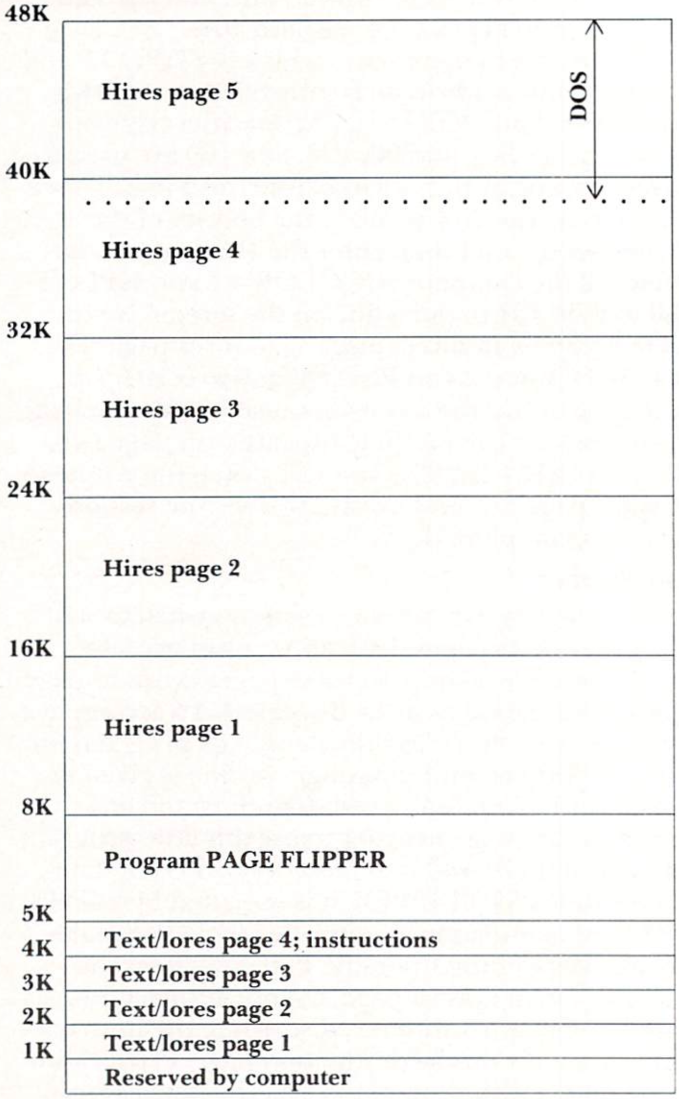
Now that the memory layout and details have been explained, let's look at the PAGE FLIPPER program itself. The first thing to notice is that the program is divided into two parts. The initial part completes a few tasks and then loads the second part. This division is necessary for two reasons. One reason is that the division leaves that part of the program which does most of the work (the second part) small enough that room is left for three pages of text/lores memory plus a fourth page for the directions. The other, more critical, reason is that the first program POKEs certain values into the correct positions so that the second program loads above text/lores page four. Normally an Applesoft program loads starting at 2K, but we want to be able to copy images into that area without overwriting our program.

The first executable statements in the initial part of the program POKE into memory the machine language transfer routine so that it will be there when it is needed. Beginning in line 2000, the instructions are printed, but they are printed on text page one while hires graphics page two is displayed so the user doesn't see them yet. In line 2190, these instructions are moved into the memory area for hires page one for safekeeping while the second half of the program is loaded later. After

Table 1. POKE Positions and Functions

Positions	Values to be POKEd	Function
103,104	1,20	sets spot for beginning of program to above text/lores page 4
230	32,64,96,128, or 160	makes HPLOT write on hires page 1, 2, 3, 4, or 5
768 to 804	values in statement number 1040	puts machine language transfer routine into memory
773	32 x hires page no. or 4 x lores page no.	determines page from which image will be taken
781	32 x hires page no. or 4 x lores page no.	determines page to which image will be written
785	32 for hires, 4 for text/lores	sets size of page to be transferred
790,791	169,0 177,6 17,8 81,8	erase page [LDA #$00] copy a page [LDA ($06),Y] overlay [ORA ($08),Y] xcopy [EOR ($08),Y]
5120,5121,5122	0,0,0	allows execution of program loaded above text/lores page 4
-16297	0	display hires if in graphics mode
-16298	0	display lores if in graphics mode
-16299	0	display page 1 (hires or text/lores)
-16300	0	display page 2 (hires or text/lores)
-16301	0	mix text and graphics if in graphics mode
-16302	0	full screen graphics if in graphics mode
-16303	0	text mode
-16304	0	graphics mode

Figure 1. Schematic RAM Memory Map



this information is moved, text page one is cleared and the introductory screen image is printed. The next to last action in the initial part is moving the pointers which specify where Applesoft programs begin so that when the last statement runs the second part of the program, it will load above text/lores page four.

For convenience in reference, the statements in the second part of the program are numbered higher than those in the first part, but this numbering scheme is not required for its operation. As part of the initialization routine, HIMEM is set to 8192 in order to prevent string variables from being written onto one of the hires pages. The commands IN#0 and PR#0 disconnect DOS so that DOS can be erased by using the machine language transfer routine. Until DOS is disabled hires pages four and five cannot be used. The remainder of the initialization routine plots random lines on the hires pages two through five in different colors, copies the instructions from hires page one to text page one, and then clears and plots random lines on hires page one.

When the program reaches the input routines, we can see the power of being able to readily move pages in memory. The directions that are displayed on the screen are given in Figure 3 except that the inverse characters on the screen are represented only by underlines in the figure. All of the input is controlled by GET statements so that the user never needs to hit return. At any time, an "I" will move the instructions to text page one and display them. A "T" puts the display into text mode, "L" gives lores graphics, and an "H" changes to hires graphics. The commands "M" and "F" specify mixed and full screen graphics. "Q" is used to quit the program. The "Q" reboots DOS which was cleared to make room for hires pages four and five. The command "D" followed by a one or two shows page one or two which may be text, lores, or hires depending on which keys have been pressed previously. An "E" followed by a number can erase any one of the hires pages 1-5 or text/lores pages 1-3 depending upon the current mode of display

Figure 2. Xcopying with Page Flipper

A) Random lines on page 2.
B) Page 3 xcopied onto page 2.
C) Page 3 xcopied onto page 2 again.

A)
B)
C)

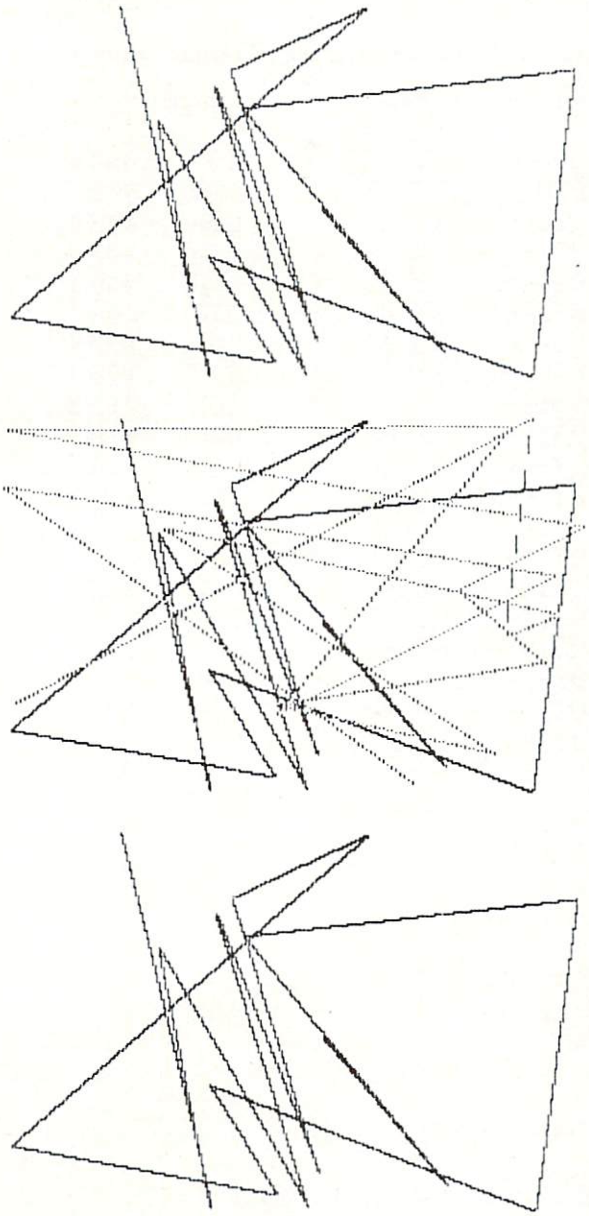


Figure 3. Instruction Page

Options: Instructions

TEXT, LOW — OR HIGH-RES GRAPHICS
MIXED OR FULL SCREEN GRAPHICS
QUIT AND REBOOT

The following commands must be followed by one or two page numbers (represented by X and Y). Accessible pages are Hires 1-5 (1-2 for display) and Text/Lores 1-3 (1-2 for display).

DISPLAY X
ERASE X
COPY X ONTO Y
XCOPY X ONTO Y
OVERLAY X ONTO Y

(hires or text/lores). The commands "C", "X", and "O" each need to be followed by two numbers. The first number gives the source page and the second number the destination page for the transfer routine.

As an example of what PAGE FLIPPER can demonstrate, press the following sequence of letters, observing what (if anything) happens after each entry; L, H, D, 2, X, 3, 2, X, 3, 2. This sequence changes the display first to lores page one, then the white random lines of hires page one, and then to the green random lines of hires page two (Figure 3a). The X command xcopies the violet lines of hires page three onto two to give a result like in Figure 3b and then again xcopies three onto two. The result is exactly the image that was first on page two. Next try to xcopy first the green lines of page two, then the violet lines of page three, the orange lines of page four, and the blue lines of page five onto page one: D, 1, X, 2, 1, X, 3, 1, X, 4, 1, X, 5, 1. Now that all of these lines are on page one, xdraw pages two through five onto page one once again. Color by color, all of the lines disappear except the white ones that were originally on page one. Now try your hand with the lores screens in the same manner. Remember that at any time you can press "I" to return to the instruction page.

This program by itself is fun to play with, but does not actually accomplish much. The accomplishment will come when you make use of the concepts pulled together here and put them in your own programs to improve them. Imagine the enhanced graphics capabilities that you will have with five hires pages. Get to work!

Program 1: Machine Language Transfer Routine

Location	Value	Operation	
0300-	A9 00	LDA	#$00
0302-	85 06	STA	$06
0304-	A9 20	LDA	#$20
0306-	85 07	STA	$07
0308-	A9 00	LDA	#$00
030A-	85 08	STA	$08
030C-	A9 20	LDA	#$20
030E-	85 09	STA	$09
0310-	A2 20	LDX	#$20
0312-	A0 00	LDY	#$00
0314-	B1 06	LDA	($06),Y
0316-	A9 00	LDA	#$00
0318-	91 08	STA	($08),Y
031A-	88	DEY	
031B-	D0 F7	BNE	$0314
031D-	E6 07	INC	$07
031F-	E6 09	INC	$09
0321-	CA	DEX	
0322-	D0 F0	BNE	$0314
0324-	60	RTS	

Program 2: "Page Flipper"

```
100  REM *** PAGE FLIPPER ***
110  REM      INITIAL PART

130  REM  BY DICK CORNELIUS
140  REM      CHEMISTRY DEPT.
150  REM      WICHITA STATE UNIV.
160  REM      WICHITA, KS  67208

170  REM      (316) 689-3120

180  REM  POKES USED BY THE MACHINE LANGUAGE TRANSFER ROUTINE:
190  REM   773:  SPECIFIES THE "FROM" PAGE
200  REM         32,64,96,128, OR 160 FOR HIRES PAGE 1,2,3,4, OR 5
210  REM         4,8,12, OR 16 FOR TEXT/LORES PAGE 1,2,3, OR 4
220  REM   781:  SPECIFIES THE "TO" PAGE (SEE VALUES ABOVE)
230  REM   785:  SIZE OF PAGE:  4-TEXT/LORES, 8-HIRES
240  REM   790:  DETERMINES (WITH 791) THE NATURE OF THE TRANSFER
250  REM         177-COPY, 81-XCOPY, 17-OVERLAY, 169-ERASE
260  REM   791:  0-ERASE, 6-COPY, 8-OVERLAY OR XCOPY

1000  REM
        **INITIALIZATION**

1010  REM    THE NEXT TWO STATEMENTS POKE INTO MEMORY THE MACHINE
1020  REM    LANGUAGE MEMORY TRANSFER ROUTINE
1030  FOR SPOT = 768 TO 804: READ CODE: POKE SPOT,CODE: NEXT
1040  DATA  169,0,133,6,169,32,133,7,169,0,133,8,169,64,133,9,162,32,
    160,0,177,6,81,8,145,8,136,208,247,230,7,230,9,202,208,240,96
2000  REM
        **PRINT INSTRUCTIONS**
```

```
2010  HIMEM: 8192
2020  HOME : HGR2
2030  HOME
2040  PRINT "OPTIONS:";
2050  HTAB 10: INVERSE : PRINT "I";: NORMAL : PRINT "NSTRUCTIONS"
2060  PRINT : HTAB 10: INVERSE : PRINT "T";: NORMAL : PRINT "EXT, ";:
       INVERSE : PRINT "L";: NORMAL : PRINT "OW- OR ";: INVERSE : PRINT
      "H";: NORMAL : PRINT "IGH-RES GRAPHICS";
2070  PRINT : HTAB 10: INVERSE : PRINT "M";: NORMAL : PRINT "IXED OR
      ";: INVERSE : PRINT "F";: NORMAL : PRINT "ULL SCREEN GRAPHICS"
2080  PRINT : HTAB 10: INVERSE : PRINT "Q";: NORMAL : PRINT "UIT AND
      REBOOT"
2090  PRINT : PRINT "THE FOLLOWING COMMANDS MUST BE FOLLOWED"
2100  PRINT "BY ONE OR TWO PAGE NUMBERS (REPRESENTED"
2110  PRINT "BY X AND Y). ACCESSIBLE PAGES ARE HIRES"
2120  PRINT "1-5 (1-2 FOR DISPLAY) AND TEXT/LORES 1-3";
2130  PRINT "(1-2 FOR DISPLAY).
2140  PRINT : HTAB 10: INVERSE : PRINT "D";: NORMAL : PRINT "ISPLAY "
      ;: INVERSE : PRINT "X": NORMAL
2150  PRINT : HTAB 10: INVERSE : PRINT "E";: NORMAL : PRINT "RASE ";:
       INVERSE : PRINT "X": NORMAL
2160  PRINT : HTAB 10: INVERSE : PRINT "C";: NORMAL : PRINT "OPY ";: INVERS
E : PRINT "X";: NORMAL : PRINT " ONTO ";: INVERSE : PRINT "Y": NORMAL
2170  PRINT : HTAB 10: INVERSE : PRINT "X";: NORMAL : PRINT "COPY ";:
       INVERSE : PRINT "X";: NORMAL : PRINT " ONTO ";: INVERSE : PRINT
      "Y": NORMAL
2180  PRINT : HTAB 10: INVERSE : PRINT "O";: NORMAL : PRINT "VERLAY "
      ;: INVERSE : PRINT "X";: NORMAL : PRINT " ONTO ";: INVERSE : PRINT
      "Y": NORMAL
2190  POKE 773,4: POKE 781,32: POKE 785,4: POKE 790,177: POKE 791,6: CALL
      768: REM MOVES PAGE 1 TO HIRES PAGE 1
3000  REM
        **PRINT FIRST SCREEN IMAGE**

3010  TEXT : HOME : VTAB 2: HTAB 13:
      PRINT "/PAGE FLIPPER/
3020  PRINT : PRINT : PRINT "THIS PRO
      GRAM ALLOWS THE EASY DISPLAY OF"
3030  PRINT "THE VARIOUS TEXT AND
      GRAPHICS PAGES AND"
3040  PRINT "DEMONSTRATES THE 'XPLOT'
      UTILITY."
3050  PRINT : PRINT "WRITTEN BY DICK
      CORNELIUS
3060  PRINT " WICHITA STATE UNIVERSITY"
3070  PRINT " WICHITA, KS 67208"
3080  PRINT : PRINT : PRINT : PRINT
      "PLEASE WAIT A FEW SECONDS..."
3090  VTAB 21
4000  REM
        **LOAD SECOND HALF**

4010  POKE 104,20: POKE 103,1: REM MOVES
      STARTING POSITION FOR APPLESOFT
      PROGRAMS
4020  POKE 5120,0: POKE 5121,0: POKE
      5122,0: REM  PUTS ZEROS INTO
      STARTING POSITIONS
4030 D$ =  CHR$ (13) +  CHR$ (4)
4040  PRINT D$"RUN PAGE FLIPPER.FINAL
      PART"
```

```
5000  REM ***PAGE FLIPPER***
5010  REM     FINAL PART
5020  REM   UPDATED 6/22/81

5030  REM  BY DICK CORNELIUS
5040  REM     CHEMISTRY DEPT.
5050  REM     WICHITA STATE UNIV.
5060  REM     WICHITA, KS  67208

5070  REM     (316) 689-3120

6000  REM
        **INITIALIZATION**

6010  HIMEM: 8192: REM  KEEPS ALL VARIABLES STORED BELOW HIRES PAGE 1
6020 BELL$ =  CHR$ (7):MODE$ = "T/L"
6030  IN# 0: REM     THESE TWO COMMANDS
6040  PR# 0: REM     DISCONNECT DOS
6050  REM  THE FOLLOWING STATEMENTS CLEAR THE VARIOUS PAGES
6060  POKE 790,169: POKE 791,0
6070  POKE 785,128: POKE 773,32: POKE 781,64: CALL 768: REM CLEARS HIRE
     S PAGES 2-5
6080  REM DOS HAS NOW BEEN ERASED
6090  REM
        **DRAWING LINES ON DIFFERENT PAGES**

6100  FOR PAGE = 2 TO 5
6110  HCOLOR= PAGE - 1: IF PAGE > 3 THEN  HCOLOR= PAGE + 1
6120  GOSUB 8000: REM LINES PLOTTED ON HIRES PAGES 2-5 HERE
6130  NEXT
6140  POKE 773,32: POKE 781,4: POKE 785,4: POKE 790,177: POKE 791,6: CALL
     768: REM MOVES INSTRUCTIONS ONTO PAGE 1
6150  POKE 781,16: CALL 768: REM MOVES INSTRUCTIONS ONTO PAGE 4
6160  POKE 781,32: POKE 785,32: POKE 790,169: POKE 791,0: CALL 768: REM
     CLEARS HIRES PAGE1
6170 PAGE = 1: HCOLOR= 3: GOSUB 8000: REM LINES PLOTTED ON HIRES PAGE 1
7000  REM
        **INPUT ROUTINES**

7010  GET G$
7020  IF G$ = "I" THEN  PRINT BELL$;: GOSUB 7900
7030  IF G$ = "T" THEN  PRINT BELL$;: POKE  - 16303,0:MODE$ = "T/L"
7040  IF G$ = "L" THEN  PRINT BELL$;: POKE  - 16298,0: POKE  - 16304,0:
     MODE$ = "T/L"
7050  IF G$ = "H" THEN  PRINT BELL$;: POKE  - 16297,0: POKE  - 16304,0:
     MODE$ = "H"
7060  IF G$ = "M" THEN  PRINT BELL$;: POKE  - 16301,0
7070  IF G$ = "F" THEN  PRINT BELL$;: POKE  - 16302,0
7080  IF G$ = "Q" THEN 9000
7090  IF G$ = "D" THEN  PRINT BELL$;: GOSUB 7400
7100  IF G$ = "E" THEN  PRINT BELL$;: GOSUB 7500
7110  IF G$ = "C" OR G$ = "X" OR G$ = "O" THEN  PRINT BELL$;: GOSUB 760
     0
7120  GOTO 7010
7400  REM
        **DISPLAY**

7410  GET G$
7420  IF G$ = "1" THEN  POKE  - 16300,0: PRINT BELL$;: RETURN
7430  IF G$ = "2" THEN  POKE  - 16299,0: PRINT BELL$;: RETURN
7440  POP : GOTO 7030
7500  REM
```

```
        **ERASE**

7510  GOSUB 7800
7520  POKE 781,PLOC
7530  POKE 790,169:POKE 791,0
7540 SIZE = 32
7550  IF MODE$ = "T/L" THEN
      SIZE = 4
  **COPY, XCOPY, OR OVERLAY**
7560  POKE 785,SIZE
7570  CALL 768
7580  RETURN
7600  REM

7610 SIZE = 32
7620  IF MODE$ = "T/L" THEN
      SIZE = 4
7630  POKE 785,SIZE
7640 A1 = 177:A2 = 6
7650  IF G$ = "X" THEN A1 =
      81:A2 = 8
7660  IF G$ = "O" THEN A1 =
      17:A2 = 8
7670  GOSUB 7800
7680  POKE 773,PLOC
7690  GOSUB 7800
7700  POKE 781,PLOC
7710  POKE 790,A1
7720  POKE 791,A2
7730  CALL 768
7740  RETURN
7800  REM
        **PAGE SELECTOR**

7810  GET G$
7820 PAGE =  ASC (G$) - 48
7830 MAX = 5:MULT = 32
7840  IF MODE$ = "T" OR MODE$ = "T/L" THEN MAX = 3:MULT = 4
7850  IF PAGE < 1 OR PAGE > MAX THEN  POP : POP : GOTO 7030
7860 PLOC = PAGE * MULT: REM  PLOC IS PAGE LOCATION
7870  PRINT BELL$;: RETURN
7900  REM
        **GET INSTRUCTIONS**

7910 MODE$ = "T/L"
7920  POKE 773,16: POKE 781,4: POKE 785,4: POKE 790,177: POKE 791,6: CALL
    768
7930  POKE  - 16303,0: POKE  - 16300,0
7940  RETURN
8000  REM
        **PLOT RANDOM LINES**

8010  POKE 230,PAGE * 32
8020  HPLOT 280 *  RND (1),192 *  RND (1)
8030  FOR LINE = 1 TO 15
8040 X = 280 *  RND (1):Y = 192 *  RND (1)
8050  HPLOT  TO X,Y
8060  NEXT
8070  RETURN
9000  REM
        **REBOOT**

9010  PR# 6
```

©

THE ATARI® GAZETTE

Atari Data Management/ Database System

An Atari Database With Application Generation Features

Ronald Marcuse
Freehold, NJ

My initial excursion into the world of microcomputers began several months ago, timed to coincide with the arrival of the carton containing my Atari 800. It took perhaps a month for my data processing (meaning file processing) background to emerge from the myraid of games that I was coding and playing on the Atari, but this was inevitable. Don't misunderstand me, the games are fun, but I personally wanted more from the computer than just zapping Zylon Raiders or hitting a quarter-inch baseball across a 25 inch screen.

Luckily, a second carton arrived during this transition period — the 810 Disk Drive, complete with enough blank disks to store anything that my imagination conjured up. The cassette recorder, which had satisfied the storage needs of my embryonic stage, was too limited in I/O abilities for a confirmed file-processor.

In the ensuing month, information processing systems materialized in all shapes and sizes, covering such important data as telephone/addresses, appointments, and the "Star Wars" figure collection of my offspring (which turned out to be the largest of my numerous files)! When I saw myself coding a catalog system to keep track of all the other catalog/ information systems, I vowed to find a better way.

This leads us to the subject of the article — an Atari Data Management/Database system. There is a great deal of similarity between different systems that are designed to track different data; in fact, the similarities usually far outweigh the differences. The variance in file/record attributes does not require a markedly different approach in getting to and from the storage medium. If one were to store these file/record attributes in, say, a Data Dictionary, one could use the Dictionary to supply the parameters to drive generalized file/screen/ printer IO routines. One need only specify the attributes to generate the application. An information system needed to track paper clips could be implemented in minutes. That, in a nutshell, is the concept.

Converting To Microsoft BASIC

Before we get into a discussion of the software itself, a word about converting this program to the "other" forms of BASIC (e.g. MICROSOFT). Atari Basic has an intrinsic weakness in its handling of string variables as compared to Microsoft and others of that ilk. The inability to dimension a string array as well as the lack of the concatenation flexibility of LEFT$, MID$, and RIGHT$ has, if anything, caused additional complexity in the software. Sub-stringing on the Atari is of the form A$(B,C) where B and C are the starting and ending points of the stated (and DIMensioned) string A$. There are also numerous GOSUB NNNN + I's in the programming to allow retrieval of a particular string where A$(I) would have been much simpler.

The selection of file names would depend on the environment at which the conversion is aimed. Atari's requirement is of the form "Dn:FILENAME.EXT" where n is the drive number (1-4), FILENAME is a maximum of eight characters and the optional EXTender is limited to three. Within this particular effort, the Database files are generically formed as "D:filename.DB". The DOS functions represented by the various XIO commands are reproduced in Table 3. The Atari's TRAP statements are a mechanism to redirect program control during an error that would otherwise cause an abnormal termination.

The discussion of the Data Management/ Database software can best be handled by neatly dividing it up into its three main functions: 1. Data Dictionary; 2. File Management; and 3. Soft Utility. You flowcharters out there may find the diagram in Figure 1 interesting. A primary option menu, located in lines 100-220 of the program listing, controls the flow into and out of the three main functions. Note that the sort utility is *not* a resident

*CYBERSOFT is a registered trademark of Cybersoft Inc.

IF YOU OWN ATARI TAKE A CLOSER LOOK.

Introducing CYBERSOFT™ custom ATARI® programs. The Grand Masters of programming at a mail order price.

Now that you own ATARI®, the finest, most versatile, most powerful home computer on the market, you should have the software to make your ATARI sit up and really function.

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

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

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

Invasion Orion™ *(Automated Simulations) 32K* You are the fleet commander pitted against the Klaatu somewhere in space. You play the leading role in a number of carefully crafted scenarios. This game takes more skill than luck in order to be successful. The documentation is superb. Create your own senarios. Excellent graphics, good sound. Cassette $24.95

Rescue at Rigel *(Automated Simulations) 32K* Ten people from the Orion colony at Ultima Thule (including your favorite female) have been captured. Your mother ship must leave in 60 minutes. As "Sudden" Smith, you must find and release them, but it won't be easy in the maze of corridors, rooms and levels. Oh, by the way, the place is crawling with bad guys. Cassette $29.95

Sands of Mars *(Crystal) 40K* This is the one gamesters are all talking about. A two diskette game, with diskette one putting you in command of the "Starship Herman". You must stock her, select her crew, command the launch and negotiate your way through the hazards of the Universe. Diskette two holds your adventures on the Sands of Mars. The game is saveable. Two Diskettes $39.95

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

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

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

Mission Impossible *(Adventure International) 24K* Another adventure from the great Scott Adams. Beginning "Good morning, your mission is...", your task is to save a nuclear reactor from destruction. No magic help in this one. Addiction is possible. Cassette $14.95

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

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

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

***CYBERSOFT Ironclad Guarantee.**

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

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

1-800-247-2000 EXT. 505

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

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

cybersoft

module and is called in by executing a RUN "D:DMSSORT" statement. This was done purposely to allow the package to run on an Atari configured with 24K RAM and a single disk drive. Those lucky (or rich?) enough to own 32K+ can easily incorporate the sort into the main program by following the procedures at the end of the article. Generally speaking, the File Management routines occupy lines 500-4600 while the Data Dictionary runs from 6000-9320, though several subroutines are shared by both.

The Dictionary

The Data Dictionary function, as the front-seat driver of the entire vehicle, deserves clarification first. Its primary function is to create and monitor the individual dictionary records containing the attributes of the database on that particular disk. Within this scheme, the ADD logic resides in lines 8000-8630, the INDEX in 6500-6600, the INQUIRY or LIST in 7000-7080, and the DELETE from 7500-7640. Subroutines shared by one or more of these modules are: PAUSE 6300; CREATE FILESPEC 6310-6320; PRINT DIC RECORD 8750-8870; LOAD VARIABLES FROM DIC RECORD 9000-9050; and READ DIC RECORD 9200-9320. Additionally, located from 9801-9846 are short subroutines of the variety GOSUB NNNN+I utilized in moving headers and data elements to and from larger strings. The format of the dictionary record (REC$) and other variables used are listed in Tables 1 and 2.

In operation, when creating a new application/database, one would specify a title (25 char. max), data set name (8 char. max) and the number of data elements (6 max) within the proposed record. A loop (lines 8110-8200) would step you through the entering of the heading (12 char.); size (30 max); and editing requirements of each data element. Editing criteria of N–Numeric; D–Date; and $–Dollar may be requested. A nominal limit of six data elements with 30 characters maximum for any one element and a total record length of 120 bytes has been imposed for no other reason than I happened to have liked those numbers on that day. If the Data Dictionary has *not* been initialized on the disk, you will be prompted during the add routine for permission to create it (lines 8610-8630).

There is a conspicuous absence of any UPDATE functions on the data dictionary menu. This was intentional. Changing the attributes of a given data base *after* it has been created and fed huge mounds of raw data may be detrimental to your health. (Picture the File Manager going after a 40 byte record that it now believes is 80 bytes long). If anyone out there adds an update here, please call me after the smoke clears.

The Manager

The File Management system, with its menu residing in lines (500-670), requires that the operator first select one of the databases on that disk (lines 750-892), with an index available by entering a single character of "I". The selection process reads the data dictionary file and, upon finding your request (file name is the key), moves the dictionary record into the applicable variables. Once loaded, these variables control the execution of the other File management modules. These are: ADD RECORD (lines 1000-1200); LIST RECORD/INQUIRY (2000-2440); UPDATE RECORD (3000-3310). Note that typing an "E" on any function menu will return program control to the module one step higher in the network. The variables used in these procedures are listed in Tables 1 and 2. There is a SEARCH procedure generated in lines 4000-4210 that can be utilized by both the INQUIRY/LIST and UPDATE functions. One need only specify the field number and then enter the value of the characters to be used. By entering fewer characters than the size of the field, one can perform a generic search, extracting, in turn, all records that satisfy those requirements.

The previously mentioned input editing is performed in lines 4400-4590. The selection of output medium (screen or printer) is done in lines 680-740. ASCII control characters (for an EPSON printer), based on the attributes of the resident Database, are generated and sent to the device thereby setting character size and tabs. The EPSON MX-80 has software selectable print lines of from 40 to 132 positions. Other printers without this feature may require a modified approach or a limitation in your record size.

The Sort

The SORT routine (Program 2) is an example of the "selection and exchange" variety. Logic similar to that contained in the database manager allows you to select the file to sort or list the index of that particular disk (lines 4000-4340). The choice of sort key and whether "ascending" or "descending" occurs in lines 5000-5070. The file is input and stored in the DIMensioned string X$ in lines 5100-5160. The variables I, J, and K are used as pointers during the loops through X$ as follows: I represents the sorted/not sorted boundary of X$, J is the current comparison to the previously selected lowest or highest value (DS$) and K is the location of this previous value.

An exchange between X$(K,K+L) and X$(I,I+L) occurs if an unsorted condition is detected on any loop. The sort terminates when the sorted boundary is equal to the size of X$. X$ is then written to the disk, the original file is deleted and the new file is renamed.

OK, at this point it would probably be beneficial for us to single-step through the code, but I think that we have neither the time nor the space to accomplish this feat. But we can and will examine at least one aspect of the structure, notably: the interaction of the subroutines that perform the input and handling of an existing Data Dictionary record. Entry to the READ DICTIONARY routine (9200-9310) is from several possible locations: LIST INDEX, both FMS and DIC (via line 6500); DICTIONARY INQUIRY/LIST (line 7080); and SELECT DATABASE (line 800). Two variables (loaded by the calling module) enable this one subroutine to perform several functions. "T" represents the "type" of request (0-index, 1-loop through to find and load data dic. record with key equal to value in FILD$,2-list single data dic. record with key FILD$, 9-browse through all data dic. records. The variable "R" is utilized to return control back to a specific line number in the event of a DISK I/O error being TRAPped to line 9500.

As the program executes each line within the routine, program control is modified based on the value of "T" during that call. Options include loading the full record into all of the Dictionary variables (9000-9050), listing the Dictionary record to the screen (8750-8870), chatting with the operator (9280-9290) and a "quickie" index. The subroutines mentioned above, as well as the balance of the program, work in a similar manner.

Does It All Work?

How well does this all work? Not too bad (would you believe me if I said that it was perfect)? There is one "bug" in Atari's DOS I that has forced me to process updates in a less efficient style. This problem centers around the inability to "rewrite" a record using the NOTE and POINT comands. (These are software pointers that allow random access to a disk file.) The optimum procedure for updating a disk record would be inputting the record, updating the data, and then returning the record to the same location in the file.

Unfortunately, an error in the DOS close routine causes a bad link on the file with a resultant file number mismatch and the loss of all sectors located after the rewritten data. This has been fixed in DOS II. The way around this problem involves rewriting the entire file as a "temporary" data set, stopping when you come to the record to be updated to post the changes, and then deleting the original and renaming the temporary. A little on the slow side, but it *does* work. Another enhancement planned is the inclusion of a sort routine in machine language. The "selection/exchange" sort is fairly quick, but one written in machine language (probably a *bubble* type) would run like a jackrabbit. Oh well, is a program ever really *finished*?

For those 32 + K's out there who must have the sort routine within the main program, move the sort call (lines 150 and 200) to 585 and 655, changing the RUN to a GOTO 5000. Add SORT lines 5000-5360 to the main program, changing line 5360 to GOTO 500. Also, DIMension variable X$ with a size of 8000, or as much as you can spare. The rest of the code can be put into the round file (and I don't mean a floppy). Keeping the sort separate gives you the ability to work with larger files, so think first before you merge the two together.

Typing The Programs In

Both program listings contain unprintable ASCII characters used for screen and printer control. I have taken the liberty of substituting other characters (enclosed by [ ]) in their place. A glance at Table 4 before you start keying may save you much aggravation later. Additionally, the lower case phrases in the programs should be typed as *upper case inverse video*. One final note: because the two programs call each other by name, you must SAVE the two as "D:DMSDB" and "D:DMSSORT".

Table 1. Data Dictionary Record (REC$)

Variable Name	Pos. Within REC$	Description
FILD$	1-8	Application File Name
APP$	9-33	Application Title
DL(1) : : DL(6)	34-45	Lengths of 6 (max) Data Elements Within Record (2 char ea.)
HD1$: : HD6$	46-117	Heading Titles (12 char)
DE(1) : : DE(6)	118-123	Editing Criteria for Elements 0 = Alphanumeric 1 = Numeric (N) 2 = Date (D) 3 = Dollar ($)
	124	Delimiter ("*")

Table 2. Other DMS/DB Variables

Name	Size	Description
RL	(var)	Application Data Set Record Length
NE	(var)	Number of Data Elements
DM$	8	DMS/DB Filespec (D:DMS.DB)
FIL$	14	Application Filespec (D:filename.DB)
I$	1	Operator Response to Questions
IN$	31	General Input of Data and Temp Stor
SF SL SS SE SV$	(var)	Used by Search Function
FD1$: FD6$	(var)	Application Data Elements
T	—	Tran Type Passed to I/O Routines
EOF	—	End of File Counter
I,J,K,L,N	—	Temp Stor for Looping, Length, etc.
ERR	—	Input Data Error Flag
R	—	Error Message Return Line #
P	—	Printer/Screen Indicator (1 = Print)
S$,X$,P$,N$	—	Messages, Prompts

Table 3. Atari DOS I XIO Commands

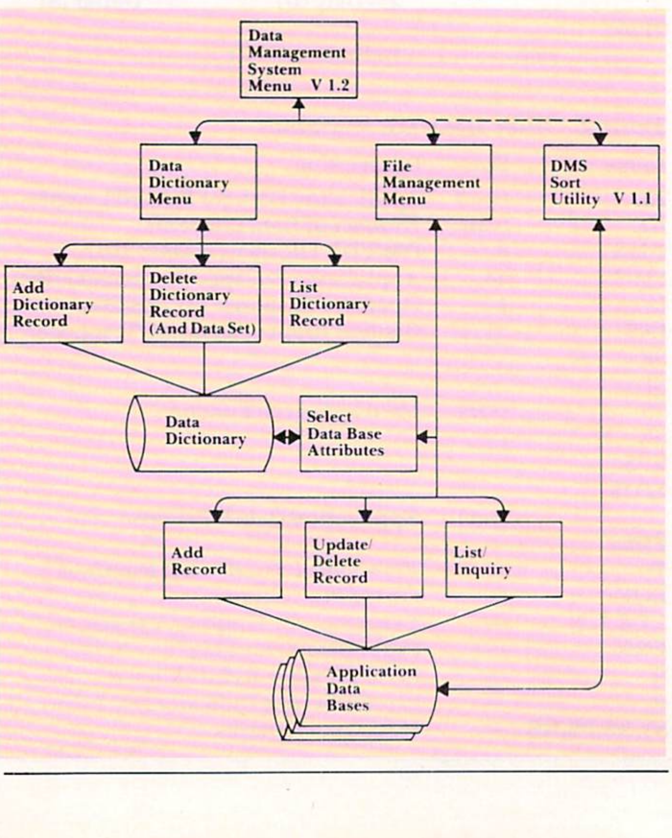
Command #	Description
XIO 32	Rename (XIO 32,#n,0,0,"D:oldname, newname")
33	Delete
35	Lock File
36	Unlock File

Note: The General Format is XIO nn,#n,0,0, "D:filename" where nn = XIO cmnd no. and n = IO control block

Table 4. Control Characters (Atari and EPSON MX-80 Printer)

Symbol	ASCII Val. (Dec)	Key Sequence (Atari)	Description
[A]	0	CON; , (comma)	Null; End of Tab Set Seq
[B]	9	CON; I	Horizontal Tab
[C]	11	CON; K	Vertical Tab
[D]	12	CON; L	Form Feed
[E]	14	CON; N	Print Double Width Characters
[F]	15	CON; O	Print Condensed Characters
[G]	18	CON; R	Cancel Condensed Mode
[H]	20	CON; T	Cancel Double Width Mode
[I]	27,68	ESC;D	Set Tab (followed by Tab Positions and NULL Char)
[J]	27,253	ESC;CON;2	Console Bell
[K]	27,125	ESC;CON;CLR	Clear Screen
[L]	27,29	ESC;CON; =	Move Cursor Down
[M]	27,158	ESC;CON;TAB	Clear Tab (screen)
[N]	27,159	ESC;SHFT;TAB	Set Tab (screen)
[O]	27,127	ESC;TAB	Tab (screen)

Figure 1. Data Management/Database System



Program 1.

```
10 REM * DMS - DATABASE PROTOTYPE    VER
 1.2  *
20 REM * 02/19/81  RONALD MARCUSE, FREEH
OLD NJ *
40 POKE 82,0:POKE 83,39:? "!{=}!":GOTO 1
00
50 DIM APP$(26),FIL$(14),FILD$(9),HD1$(1
2),HD2$(12),HD3$(12),HD4$(12),HD5$(12)
60 DIM HD6$(12),I$(1),REC$(132),IN$(31),
DM$(8),DL(6),DE(6)
65 DIM S$(15),X$(5),P$(24),N$(16):N$="!R
ECORD NOT FOUND!":DM$="D:DMS.DB"
70 P$="PRESS RETURN TO CONTINUE":S$="{DO
WN}SELECT OPTION:":X$="!ERROR!"
75 FOR I=1 TO 6:DL(I)=0:DE(I)=0:NEXT I
80 FOR I=1 TO 132:REC$(I,I)=" ":NEXT I:R
ETURN
100 GRAPHICS 0:? "{DOWN}  DATA MANAGEMEN
100 GRAPHICS 0:? "{DOWN}  DATA MANAGEMEN
T SYSTEM  - VER 1.2":CLR :DIM I$(1)
110 ? "{DOWN}      PRIMARY OPTION MENU":P
OKE 16,64:POKE 53774,64
120 ? "{DOWN} D - DATA DICTIONARY FUNCTI
ONS":?"    (DEFINE NEW APPLICATION & DA
TA)"
130 ? "{DOWN} R - EXEC FILE MANAGEMENT S
YSTEM":? "    (ACCESS EXISTING DATA BASE
)"
150 ? "{DOWN} S - CALL DMS SORT UTILITY"
:? "{DOWN} E - END (TERMINATE DMS)"
160 ? :? "SELECT OPTION:";:INPUT I$:IF I
$="R" THEN 500
180 IF I$="D" THEN 6000
200 IF I$="S" THEN RUN "D:DMSSORT"
210 IF I$="E" THEN GRAPHICS 0:END
220 ? "!INVALID!":GOTO 160
490 ? "{DOWN}SELECT DATA SET 1ST":GOSUB
6300
500 ? "{CLEAR}{DOWN}    DMS FILE MANAGEME
NT MENU"
520 IF NE<>0 THEN ? :? ,APP$:IF P=1 THEN
 ? ,"!PRINTER SELECTED!"
540 ? "{DOWN} S - SELECT DATA BASE":? "
A - ADD"
580 ? " L - LIST / INQUIRY":? " U - UPDA
TE"
590 ? " P - PRINTER (SCREEN DEFAULT)":?
" E - END (RETURN TO DMS MENU)"
600 ? :? "SELECT OPTION:";:INPUT I$:IF I
$="E" THEN 100
620 IF I$="S" THEN 750
630 IF I$="A" THEN 1000
640 IF I$="L" THEN 2000
650 IF I$="U" THEN 3000
660 IF I$="P" THEN 680
670 ? "!INVALID!":GOTO 600
680 IF NE=0 THEN 490
```

```
690 ? :? "TYPE: P- PRINTER S- SCREEN":IN
PUT I$:IF I$<>"P" THEN P=0:GOTO 500
700 P=1:IN$="{R}{ESC}D     {,}":IF RL>55
 OR NE>4 THEN IN$(1,1)="{O}"
710 L=1:FOR I=2 TO NE:J=DL(I-1):IF J<12
THEN J=12
720 L=L+J+2:IN$(I+2,I+2)=CHR$(L):NEXT I
730 R=690:TRAP 9510:LPRINT IN$:TRAP 4000
0:GOSUB 80:GOTO 500
750 ? "{CLEAR}{DOWN}    DATA BASE SELECT
ION":CLR :GOSUB 50:R=760
760 ? "{DOWN}ENTER DATA SET NAME (I FOR
INDEX)":? :? P$;" (END)"
770 INPUT FILD$:IF LEN(FILD$)=0 THEN 500

780 IF LEN(FILD$)=1 AND FILD$="I" THEN G
OSUB 6500:GOTO 760
800 T=1:GOSUB 9200:IF EOF=0 THEN 500
820 ? "{DOWN}DATA SET LOADED":GOSUB 6300

830 RL=1:FOR N=1 TO NE:GOSUB 880+N:RL=RL
+DL(N):NEXT N
850 DIM DS$(RL),SV$(30):FOR I=1 TO RL:DS
$(I,I)=" ":NEXT I:GOTO 500
881 DIM FD1$(DL(N)):RETURN
882 DIM FD2$(DL(N)):RETURN
883 DIM FD3$(DL(N)):RETURN
884 DIM FD4$(DL(N)):RETURN
885 DIM FD5$(DL(N)):RETURN
886 DIM FD6$(DL(N)):RETURN
1000 IF NE=0 THEN 490
1005 R=500:TRAP 9500:XIO 36,#2,0,0,FIL$:
OPEN #2,9,0,FIL$:TRAP 40000
1010 ? "{CLEAR}{DOWN}   ";APP$:? "{DOWN}
 TO ADD RECORD, ENTER:"
1040 FOR N=1 TO NE
1050 GOSUB 9820+N:IF DE(N)<>2 THEN ? " (
";DL(N);" CHAR MAX)":GOTO 1070
1060 ? " (6 CHAR - MMDDYY)"
1070 INPUT IN$:L=LEN(IN$):IF N=1 AND L=0
 THEN 1180
1075 GOSUB 4400:IF ERR<>0 THEN 1070
1080 GOSUB 9810+N:NEXT N:GOSUB 4000
1170 ? #2;DS$:? "{DOWN}TRANSACTION ACCEP
TED"
1180 ? "{DOWN}TYPE E FOR FMS MENU":? P$
1190 INPUT I$:IF I$<>"E" THEN 1010
1200 CLOSE #2:XIO 35,#2,0,0,FIL$:GOTO 50
0
2000 IF NE=0 THEN 490
2020 ? "{CLEAR}{DOWN}   ";APP$:? "{DOWN}
 INQUIRY / LIST":IF P=1 THEN ? ,,"{UP}!P
RINTER SELECTED!"
2030 ? "{DOWN} A - LIST ALL RECORDS":? "
 S - SEARCH ON KEY"
2050 ? " E - END (RETURN TO MENU)":T=9
2060 ? S$;:INPUT I$:IF I$="E" THEN 500
2080 IF I$="A" THEN 2300
2090 IF I$="S" THEN T=1:? "{CLEAR}":GOSU
B 4100:GOTO 2300
2100 ? X$:GOTO 2060
2300 I=0:EOF=0:R=2000:TRAP 9500:OPEN #2,
4,0,FIL$
2310 TRAP 2420:INPUT #2,DS$:TRAP 40000:I
F T>8 THEN 2330
2320 IF SV$<>DS$(SS,SE) THEN 2310
2330 EOF=EOF+1:IF P<>1 THEN 2400
2340 L=1:J=1:IF I>0 THEN 2370
2350 R=2440:TRAP 9510:LPRINT "{I}{N}";AP
P$;"{T}{K}"
2360 LPRINT HD1$;"{I}";HD2$;"{I}";HD3$;"
{I}";HD4$;"{I}";HD5$;"{I}";HD6$;"{K}"
2370 FOR N=1 TO NE:K=DL(N):IF K<12 THEN
K=12
2375 REC$(J,J+DL(N)-1)=DS$(L,L+DL(N)-1):
L=L+DL(N):J=J+K+2:NEXT N
2380 TRAP 9510:LPRINT REC$(1,J):I=I+1:IF
 I<55 THEN 2310
2390 I=0:LPRINT "{L}":GOTO 2310
2400 GOSUB 4300:? :? P$;" (E TO STOP)"
2410 INPUT I$:IF I$<>"E" THEN 2310
2420 IF EOF=0 THEN ? N$
2430 IF EOF>0 THEN ? "REC COUNT- ";EOF
2435 IF P=1 THEN LPRINT "{L}"
2440 CLOSE #2:GOSUB 6300:GOTO 2020
3000 IF NE=0 THEN 490
3020 ? "{CLEAR}{DOWN}   ";APP$:? "{DOWN}
 UPDATE / DELETE";:GOSUB 4100
3040 R=500:TRAP 9500:OPEN #2,4,0,FIL$:OP
EN #3,8,0,"D:TEMP":EOF=0
3060 TRAP 3250:INPUT #2,DS$:TRAP 40000
3080 IF SV$<>DS$(SS,SE) THEN ? #3;DS$:GO
TO 3060
3100 GOSUB 4300
3120 ? "{DOWN}ENTER: FIELD # TO UPDATE;
D TO DELETE":? "PRESS RETURN TO WRITE RE
C"
3130 EOF=EOF+1:INPUT I$:IF LEN(I$)=0 THE
N ? #3;DS$:GOTO 3060
3135 IF I$="D" THEN 3060
3140 TRAP 3120:N=VAL(I$):TRAP 40000:IF N
<1 OR N>NE THEN 3120
3150 ? "{DOWN}ENTER NEW ";:GOSUB 9820+N:
? " "
3170 INPUT IN$:L=LEN(IN$):GOSUB 4400:IF
ERR<>0 THEN 3150
3180 GOSUB 9810+N:GOSUB 4000:GOTO 3100
3250 FOR I=1 TO LEN(IN$):IN$(I,I)=" ":NE
XT I:IN$(1,7)="D:TEMP,"
3260 IN$(8,7+LEN(FIL$)-2)=FIL$(3)
3270 CLOSE #2:CLOSE #3:XIO 36,#2,0,0,FIL
$:XIO 33,#2,0,0,FIL$
3280 XIO 32,#3,0,0,IN$:XIO 35,#3,0,0,FIL
$
3290 IF EOF=0 THEN ? N$
3300 ? "{DOWN}MORE UPDATES? (Y OR N)":IN
```

```
PUT I$:IF I$="Y" THEN 3000
3310 GOTO 500
4000 FOR I=1 TO LEN(DS$):DS$(I,I)=" ":NE
XT I
4010 L=1:FOR N=1 TO NE:GOSUB 9840+N:DS$(
L,L+DL(N)-1)=IN$
4020 L=L+DL(N):NEXT N:DS$(RL)="*":RETURN

4100 ? " SELECT KEY:":? :FOR I=1 TO NE:?
 " ";I;" ";
4110 GOSUB 9820+I:? " ":NEXT I:? :IF SF=
0 THEN 4140
4120 ? "PRESS RETURN FOR KEY:":? " 1ST "
;SL;" POS OF ";
4130 GOSUB 9820+SF:? " (";SV$;")":? " ":
? " OR, ";
4140 ? "ENTER KEY FIELD #";
4160 TRAP 4200:INPUT SF:TRAP 40000:IF SF
<1 OR SF>NE THEN ? X$:GOTO 4140
4170 ? "ENTER VALUE OF ";:GOSUB 9820+SF:
? " (1-";DL(SF);")"
4180 INPUT SV$:SL=LEN(SV$):IF SL<1 OR SL
>DL(SF) THEN ? X$;" LEN":GOTO 4170
4190 SS=1:IF SF>1 THEN FOR I=2 TO SF:SS=
SS+DL(I-1):NEXT I
4200 IF SF=0 THEN ? X$:GOTO 4160
4210 SE=SS+SL-1:RETURN
4300 L=1:FOR N=1 TO NE:IN$=DS$(L,L+DL(N)
-1)
4310 GOSUB 9810+N:L=L+DL(N):NEXT N
4320 ? "{CLEAR}{DOWN}  ";APP$:? :FOR N=1
 TO NE
4330 ? N;" ";:GOSUB 9820+N:GOSUB 9840+N:
? " ";IN$:NEXT N:RETURN
4400 ERR=0:IF L<1 OR L>DL(N) THEN ERR=1
4410 ON DE(N) GOSUB 4500,4520,4560:IF ER
R<>0 THEN ? X$
4420 RETURN
4500 FOR I=1 TO L:J=ASC(IN$(I,I)):IF J<4
8 OR J>57 THEN ERR=1
4510 NEXT I:RETURN
4520 GOSUB 4500:IF L<>6 THEN ERR=1
4530 IF IN$(1,2)<"01" OR IN$(1,2)>"12" T
HEN ERR=1
4540 IF IN$(3,4)<"01" OR IN$(3,4)>"31" T
HEN ERR=1
4550 RETURN
4560 IF IN$(L-2,L-2)<>" " THEN ERR=1
4570 TRAP 4590:I=VAL(IN$(1,L)):TRAP 4000
0

4580 RETURN
4590 ERR=1:GOTO 4580
6000 ? "{CLEAR}{DOWN}  DMS  DATA DICTIO
NARY MENU":CLR :GOSUB 50
6050 ? "{DOWN} A - ADD NEW APPLICATION /
 DATA BASE"
6060 ? " I - LIST DICTIONARY INDEX":? "
L - LIST CURRENT DB DESCRIP"
6080 ? " D - DELETE DATA BASE":? " E - E
ND (RETURN TO DMS MENU)"
6100 ? :? S$;:INPUT I$
6120 IF I$="A" THEN 8000
6130 IF I$="I" THEN R=6000:GOSUB 6500:?
:? P$;:INPUT I$:GOTO 6000
6140 IF I$="L" THEN 7000
6150 IF I$="D" THEN 7500
6160 IF I$="E" THEN 100
6170 ? X$:GOTO 6100
6300 FOR I=1 TO 150:NEXT I:RETURN
6310 FOR L=1 TO LEN(FILD$):IF FILD$(L,L)
<>" " THEN NEXT L
6320 L=L-1:FIL$(1,2)="D:":FIL$(3,2+L)=FI
LD$(1,L):FIL$(3+L)=".DB":RETURN
6500 ? "{CLEAR}{DOWN}  DATA SET INDEX":?
 :T=0:GOSUB 9200:RETURN
7000 ? "{CLEAR}{DOWN}   DATA DICTIONARY
INQUIRY":? "{DOWN} A - ALL FILES":T=9
7020 ? " S - SINGLE FILE":? " E - END (R
ETURN TO MENU)"
7030 ? S$;:INPUT I$:IF I$="E" THEN 6000
7040 IF I$="A" THEN 7080
7050 IF I$="S" THEN ? "ENTER FILE NAME":
T=2:GOTO 7070
7060 ? X$:GOTO 7030
7070 INPUT FILD$:IF LEN(FILD$)=0 THEN 70
60
7080 R=7000:GOSUB 9200:GOTO 7000
7500 ? "{CLEAR}{DOWN} TO DELETE DICTIONA
RY ELEMENT AND":? " RELATED FILE, TYPE F
ILE NAME:"
7510 ? P$;" (CANCEL)":INPUT FILD$:IF LEN
(FILD$)=0 THEN 6000
7520 R=7500:TRAP 9500:OPEN #2,4,0,DM$:OP
EN #3,8,0,"D:TEMP":EOF=0
7530 TRAP 7600:INPUT #2,REC$:TRAP 40000
7540 IF FILD$<>REC$(1,LEN(FILD$)) THEN ?
 #3;REC$:GOTO 7530
7550 EOF=EOF+1:GOSUB 9000:GOSUB 8750
7560 ? "TYPE D TO DELETE";:INPUT I$:IF I
$<>"D" THEN ? "SAVED":? #3;REC$:GOTO 753
0

7570 ? "DELETED":XIO 36,#4,0,0,FIL$:XIO
33,#4,0,0,FIL$:GOTO 7530
7600 IF EOF=0 THEN ? N$
7610 CLOSE #2:CLOSE #3:XIO 36,#2,0,0,DM$
:XIO 33,#2,0,0,DM$
7620 XIO 32,#3,0,0,"D:TEMP,DMS.DB":XIO 3
5,#3,0,0,DM$
7640 GOTO 6000
8000 ? "{CLEAR}{DOWN}  ADD TO DATA DICTI
ONARY":? "{DOWN} ENTER APPLICATION NAME
(1 TO 25 CHAR)"
8010 INPUT APP$:L=LEN(APP$):IF L<1 OR L>
25 THEN ? X$:GOTO 8010
8020 ? "ENTER FILE NAME (1 TO 8 CHAR)":?
 "  FILESPEC WILL BE 'D:XXXXXXXX.DB'"
```

```
8030 INPUT FILD$:L=LEN(FILD$):IF L<1 OR
L>8 THEN ? X$:GOTO 8030
8040 IF FILD$(1,1)<"A" OR FILD$(1,1)>"Z"
 THEN ? X$:GOTO 8030
8060 GOSUB 6310
8080 ? "{DOWN} (MIN REC LEN=10, MAX=120)
":? "{DOWN}ENTER # OF DATA ELEMENTS IN R
EC (2-6)"
8090 TRAP 8080:INPUT NE:TRAP 40000
8100 IF NE<2 OR NE>6 THEN ? X$:GOTO 8080

8110 RL=0:FOR I=1 TO NE:? "{CLEAR}{DOWN}
  FOR DATA ELEMENT # ";I;" OF ";NE;". EN
TER:":?
8120 ? "HEADING (1-12)":INPUT IN$:L=LEN(
IN$):IF L<1 OR L>12 THEN ? X$:GOTO 8120
8125 GOSUB 9800+I
8130 ? "ELEMENT LENGTH (1 TO 30)":? "  (
TOT REC LEN IS ";RL;")"
8140 TRAP 8130:INPUT L:TRAP 40000:IF L<=
0 OR L>30 THEN ? X$:GOTO 8130
8150 DL(I)=L:RL=RL+L
8160 ? "EDITING? (N:NUMERIC, D:DATE, $:D
OLLAR)":? "RETURN TO SKIP"
8170 INPUT I$:DE(I)=0:IF I$="N" THEN DE(
I)=1
8180 IF I$="D" THEN DE(I)=2
8190 IF I$="$" THEN DE(I)=3
8200 NEXT I
8220 IF RL<10 OR RL>120 THEN ? "REC LEN=
";RL;"  ";X$:GOTO 8080
8300 REC$(1,8)=FILD$:REC$(9,33)=APP$
8320 FOR I=1 TO 6:REC$(32+I%2,33+I%2)=ST
R$(DL(I))
8340 REC$(117+I,117+I)=STR$(DE(I)):NEXT
I
8360 REC$(46,57)=HD1$:REC$(58,69)=HD2$:R
EC$(70,81)=HD3$
8380 REC$(82,93)=HD4$:REC$(94,105)=HD5$:
REC$(106,117)=HD6$:REC$(124)="%"
8390 GOSUB 8750:? "{DOWN}TYPE Y TO CREAT
E DATA BASE"
8400 INPUT I$:IF I$<>"Y" THEN 6000
8420 TRAP 8600:XIO 36,#2,0,0,DM$:OPEN #2
,9,0,DM$
8430 ? #2;REC$:CLOSE #2:XIO 35,#2,0,0,DM
$:TRAP 40000
8450 ? "DATA BASE CREATED"
8460 OPEN #3,8,0,FIL$:CLOSE #3:XIO 35,#3
,0,0,FIL$:GOTO 6000
8600 STATUS #2,ST:IF ST<>170 THEN R=8390
:GOTO 9500
8610 ? DM$;" NOT ON DISK":? "TYPE Y TO I
NITIALIZE"
8620 INPUT I$:IF I$<>"Y" THEN 6000
8630 OPEN #2,8,0,DM$:GOTO 8430
8750 ? "{CLEAR}{CLR TAB}{TAB}{CLR TAB}{T
AB}{CLR TAB}{TAB}{CLR TAB}{TAB}{CLR TAB}
{TAB}{CLR TAB}{DOWN}  DATA DICTIONARY RE
CORD"
8760 ? "{DOWN} FILE NAME - ";FILD$:? " A
PPLICATION - ";APP$
8780 ? "{DOWN} EL{SET TAB}EM #  {SET TAB
}HEADING      LE{SET TAB}NGTH  {SET TAB}
 EDIT?":?
8800 FOR I=1 TO NE:? "{TAB}";I;"{TAB}";:
GOSUB 9820+I:? "{TAB}";DL(I);"{TAB}";
8820 IF DE(I)=0 THEN ? " "
8830 IF DE(I)=1 THEN ? "NUMERIC"
8840 IF DE(I)=2 THEN ? "DATE"
8850 IF DE(I)=3 THEN ? "DOLLAR"
8870 NEXT I:? :? " RECORD LENGTH = ";RL:
RETURN
9000 FILD$=REC$(1,8):APP$=REC$(9,33)
9010 RL=0:NE=0:FOR I=1 TO 6
9020 DL(I)=VAL(REC$(32+I%2,33+I%2)):DE(I
)=VAL(REC$(117+I,117+I))
9030 IF DL(I)=0 THEN 9050
9040 NE=NE+1:RL=RL+DL(I):IN$=REC$(34+I%1
2,45+I%12):GOSUB 9800+I
9050 NEXT I:GOSUB 6310:RETURN
9200 TRAP 9500:OPEN #2,4,0,DM$:EOF=0
9210 TRAP 9300:INPUT #2,REC$:TRAP 40000:
IF T>8 THEN 9250
9220 IF T=0 THEN ? " ";REC$(1,8);" ";REC
$(9,33):GOTO 9210
9230 IF FILD$<>REC$(1,LEN(FILD$)) THEN 9
210
9250 EOF=EOF+1:GOSUB 9000:IF T>1 THEN GO
SUB 8750
9280 IF T>1 THEN ? :? P$:IF T>8 THEN ? "
 TYPE E TO END"
9290 IF T>1 THEN INPUT I$:IF T>8 AND I$<
>"E" THEN 9210
9300 CLOSE #2
9310 IF T>0 AND EOF=0 THEN ? N$:GOSUB 63
00
9320 RETURN
9500 STATUS #2,K:? "|{=}CHECK DISK DRIVE
|",X$;K:? P$;:CLOSE #2:INPUT I$:POP :GOT
O R
9510 STATUS #7,K:? "|{=}CHECK PRINTER|",
X$;K:? P$;:INPUT I$:GOTO R
9801 HD1$=IN$:RETURN
9802 HD2$=IN$:RETURN
9803 HD3$=IN$:RETURN
9804 HD4$=IN$:RETURN
9805 HD5$=IN$:RETURN
9806 HD6$=IN$:RETURN
9811 FD1$=IN$:RETURN
9812 FD2$=IN$:RETURN
9813 FD3$=IN$:RETURN
9814 FD4$=IN$:RETURN
9815 FD5$=IN$:RETURN
9816 FD6$=IN$:RETURN
9821 ? HD1$;:RETURN
```

```
9822 ? HD2$;:RETURN
9823 ? HD3$;:RETURN
9824 ? HD4$;:RETURN
9825 ? HD5$;:RETURN
9826 ? HD6$;:RETURN
9841 IN$=FD1$:RETURN
9842 IN$=FD2$:RETURN
9843 IN$=FD3$:RETURN
9844 IN$=FD4$:RETURN
9845 IN$=FD5$:RETURN
9846 IN$=FD6$:RETURN
```

Program 2.

```
11 REM **     DMS SORT UTILITY          **
12 REM **  RM  02/22/81   VER 1.2       **
20 ? "|{=}|":GRAPHICS 0:POKE 82,0:POKE 1
6,64:POKE 53774,64:GOTO 4000
50 DIM X$(8500),DS$(124),APP$(25),FIL$(1
4),FILD$(8),HD1$(12),HD2$(12)
60 DIM HD3$(12),HD4$(12),HD5$(12),HD6$(1
2),I$(1),IN$(12),DL(6)
70 FOR I=1 TO 6:DL(I)=0:NEXT I
80 RETURN
4000 CLR :? "{CLEAR}{DOWN}  DMS DATA BAS
E SORT":GOSUB 50
4010 ? :? " SELECT DATA SET:":? " (I - I
NDEX, D - DMS, E - END)"
4030 INPUT FILD$:L=LEN(FILD$):IF L=1 AND
 FILD$="D" THEN RUN "D:DMSDB"
4040 T=1:IF L=0 THEN 4010
4050 IF L=1 AND FILD$="E" THEN GRAPHICS
0:END
4060 IF L=1 AND FILD$="I" THEN ? "{CLEAR
}{DOWN} DATA SET INDEX":? :T=0
4200 R=4000:TRAP 9500:OPEN #2,4,0,"D:DMS
.DB":EOF=0
4210 TRAP 4300:INPUT #2,DS$:TRAP 40000
4220 IF T=0 THEN ? " ";DS$(1,8);" ";DS$(
9,33):GOTO 4210
4230 IF FILD$<>DS$(1,LEN(FILD$)) THEN 42
10
4240 FILD$=DS$(1,8):APP$=DS$(9,33)
4250 EOF=EOF+1:RL=1:NE=0:FOR I=1 TO 6
4260 DL(I)=VAL(DS$(32+I*2,33+I*2)):IF DL
(I)=0 THEN 4290
4280 NE=NE+1:RL=RL+DL(I):IN$=DS$(34+I*12
,45+I*12):GOSUB 9800+I
4290 NEXT I
4300 CLOSE #2:IF T=0 THEN 4010
4310 IF EOF=0 THEN ? "|NOT FOUND|":GOTO
4010
4330 FOR L=1 TO LEN(FILD$):IF FILD$(L,L)
<>" " THEN NEXT L
4340 L=L-1:FIL$(1,2)="D:":FIL$(3,2+L)=FI
LD$(1,L):FIL$(3+L)=".DB"
5000 IF NE=0 THEN 490
5010 ? "{CLEAR}{DOWN}  ";APP$:? "{DOWN}
SELECT SORT KEY:":? :FOR I=1 TO NE
5020 ? I;" ";:GOSUB 9820+I:? :NEXT I:? "
PRESS RETURN TO CANCEL";
5030 TRAP 4000:INPUT SF:TRAP 40000:IF SF
<1 OR SF>NE THEN ? "|INVALID|":GOTO 5030

5040 T=2:? "ASCENDING OR DESCENDING? (A
OR D)";:INPUT I$:IF I$="D" THEN T=1
5050 SL=DL(SF):? "LOADING ";FIL$:? "SORT
 ON ";SL;" CHAR OF ";:GOSUB 9820+SF:?
5060 SS=1:IF SF>1 THEN FOR I=2 TO SF:SS=
SS+DL(I-1):NEXT I
5070 SE=SS+SL-1:R=5000:EOF=0:N=1:L=RL-1
5100 TRAP 9500:OPEN #2,4,0,FIL$
5110 TRAP 5150:INPUT #2,DS$:TRAP 40000
5120 EOF=EOF+1:X$(N,N+L)=DS$:N=N+RL:GOTO
 5110
5150 CLOSE #2:? "{DOWN}REC LOADED= ";EOF
;", RAM (BYTES)= ";N-1:? "BEGIN SORT"
5160 I=1:N=N-RL
5200 K=0:DS$=X$(I,I+L):J=I+RL
5210 ON T GOTO 5220,5240
5220 IF X$(J+SS-1,J+SE-1)>DS$(SS,SE) THE
N DS$=X$(J,J+L):K=J
5230 GOTO 5250
5240 IF X$(J+SS-1,J+SE-1)<DS$(SS,SE) THE
N DS$=X$(J,J+L):K=J
5250 J=J+RL:IF J<=N THEN 5210
5280 IF K<>0 THEN X$(K,K+L)=X$(I,I+L):X$
(I,I+L)=DS$
5290 I=I+RL:IF I<N THEN 5200
5300 ? "|{=}|SORT COMPLETED"
5320 TRAP 9500:XIO 36,#2,0,0,FIL$:OPEN #
2,8,0,FIL$:TRAP 40000:EOF=0
5330 FOR I=1 TO N STEP RL:? #2;X$(I,I+L)
:EOF=EOF+1:NEXT I
5340 CLOSE #2:XIO 35,#2,0,0,FIL$:? "REC
COUNT= ";EOF
5350 ? "SORT THIS FILE AGAIN? (Y OR N)";
:INPUT I$:IF I$="Y" THEN 5010
5360 GOTO 4000
9500 STATUS #2,K:? "|{=}CHECK DISK DRIVE
|","|ERROR| ";K:CLOSE #2:? "PRESS ENTER"
;:INPUT I$:GOTO R
9801 HD1$=IN$:RETURN
9802 HD2$=IN$:RETURN
9803 HD3$=IN$:RETURN
9804 HD4$=IN$:RETURN
9805 HD5$=IN$:RETURN
9806 HD6$=IN$:RETURN
9821 ? HD1$;:RETURN
9822 ? HD2$;:RETURN
9823 ? HD3$;:RETURN
9824 ? HD4$;:RETURN
9825 ? HD5$;:RETURN
9826 ? HD6$;:RETURN
```

©

A Program For Writing Programs On The Atari 400/800 Computers

David D. Thornburg
Innovision
Los Altos, CA 94022

If you write a lot of programs for your computer, you may have developed a few building blocks which you like to use over again in different applications. While it is a pretty easy task to bring these building blocks into a new program unchanged, you probably need to personalize these program segments for most applications.

Have you ever wished that there were a simple way to get the computer to write customized program segments for you?

Depending on how fancy you want to get, a "Program Writing Program" (PWP) can be made to construct a fairly detailed set of BASIC statements on the basis of your answers to a series of questions. To illustrate how such programs work, I will describe a simple PWP for the Atari computers.

PWP1 — A BASIC Example

This program creates a series of DATA statements which incorporate a word or data list entered from the keyboard. This is a very handy utility, since the user only types the raw data to be used (word lists, numbers, etc.), and the computer generates the proper BASIC program lines with line numbers, the word DATA, and all the field separators. As the DATA statements are composed, they are written to a file on a disk or cassette for retrieval later.

As you can see from the listing below, this program is quite simple.

```
10 DIM A$(40)
20 PRINT "PWP1 — A DATA STATEMENT
   WRITER"
30 PRINT "ENTER YOUR DATA AND PRESS
   RETURN"
40 PRINT "ENTER *** WHEN DONE"
50 OPEN #1,8,0,"D1:WORDS.LST"
60 C=30000
70 PRINT #1;C;"DATA";
80 FOR I=1 TO 8
90 INPUT A$
100 IF A$="***" THEN GOTO 170
110 PRINT #1;A$;
120 IF I<>8 THEN PRINT #1;",";
130 NEXT I
140 PRINT #1
150 C=C+10
160 GOTO 70
170 PRINT #1;A$
180 CLOSE #1
190 END
```

In line 10 we define a string variable which will receive text from the keyboard. After printing the instructions (lines 20-40), the computer opens the file WORDS.LST on disk drive 1. (If you use the cassette recorder instead, you must modify the program as shown below.) Line 60 sets the starting line number to 30000. The beginning of the DATA statement is printed to the file by line 70. This consists of the line number (stored in variable C) followed by the word DATA. Next, a loop using lines 80-130 accepts data from the keyboard and appends up to eight entries after the word DATA, placing commas (,) between them. In line 100 the program checks to see if you are done entering data, at which time the program prints the three asterisks (line 170) and closes the data file (line 180). Otherwise, if there is more data to be entered, the new data is printed to the file and a comma is placed after the entry (line 120) unless it is the last entry in the line. Once eight entries have been made, the DATA statement is completed with a carriage return (by using the PRINT statement in line 140). The line number is increased by 10 (line 150) and the next DATA statement is started (line 160).

...a 'Program Writing Program' (PWP) can be made to construct a fairly detailed set of BASIC statements on the basis of your answers to a series of questions.

If you are using a cassette instead of a disk drive to store the new program, you must make the following changes:

```
50 OPEN #1,8,0,"C:"
52 FOR I=1 TO 64: PRINT #1;"R.";: NEXT I
54 PRINT #1
```

In line 50, the file for the new program is opened on the cassette recorder. Lines 52 and 54 print 128 characters to this file to make sure that there is no gap between the tape header and the first 128 byte block of data. By printing the characters "R." 64

times, we generate a meaningless REM statement which will be ignored automatically when the new program is loaded back into the computer.

Running PWP1

When you RUN this program, you will hear the disk drive turn on and off as the data file is created. If you use the cassette recorder, be sure a blank tape is inserted and press the PLAY and RECORD buttons when you hear the two "buzzes" from the computer. Once you have done this, press the RETURN key and notice that the tape is starting to move.

Once the program is ready to accept data, you will see a question mark on the screen. At this point, start entering your data, pressing RETURN after each entry. Entries of a list of words might look like this:

```
?THIS
?IS
?A
?LIST
?OF
.
.
.
?COMPUTER
?***
```

Once the three asterisks are entered, the data file will be closed, and the program is finished.

Be sure to save PWP1 on disk or tape because we need to erase it is order to look at the new program.

Loading The New Program

If you try to load your new program using LOAD "D:WORDS.LST", or CLOAD (from tape), you will have a most unpleasant surprise. BASIC programs which are SAVE'd or CSAVE'd are stored in a compact "tokenized" format. The program written by PWP1 is saved in the form of text strings, just as if it had been entered from the keyboard. To bring this program into the computer, we need to use the ENTER command. Be sure to type NEW and press RETURN before typing ENTER, otherwise the new program will be *added* to the program already resident in the computer. You should either type ENTER "D:WORDS.LST" or ENTER "C:", depending on whether you used a disk or a cassette. If you used a disk, the program will load automatically. If you used the cassette, you should press the PLAY button when you hear the "buzz" and press RETURN again on the computer. The cassette version will print READY twice before it is done (the first READY shows up when the dummy REM statement has been received). If you now type LIST you should see something like this:

```
30000 DATA THIS,IS,A,LIST,OF,WORDS,WHICH,
      WILL
30010 DATA BE,PLACED,INTO,A,SERIES,OF,
      DATA,STATEMENTS
30020 DATA BY,THE,COMPUTER,***
```

Obviously, the word list will be made up of whatever words *you* used when you run PWP1.

Next, you should write the rest of the program which uses these DATA statements. The following short program is an example which prints words from this list randomly.

```
10 DIM A$(40)
20 N=0
30 READ A$:N=N+1:IF A$<>"***" THEN 30
40 RESTORE
50 C=INT(N*RND(1)+1)
60 FOR I=1 TO C
70 READ A$
80 NEXT I
90 PRINT A$
100 GOTO 40
```

Once you have finished this program (which includes all the DATA statements written by the computer), you should save it using CSAVE or SAVE, and you are all done.

Using the concepts shown in PWP1, you should now be able to write your own Program Writing Programs. You can use PWP's to create all kinds of program segments. If you are really industrious, you might even want to make a PWP to write complete BASIC programs. If you do the job well enough, it just may be the last program *you* ever need to write! ©

Insight: Atari

Bill Wilkinson
Cupertino, CA

In my September column, I mentioned that the subject of graphics and I/O would make a nice series of columns. I wondered if there would be enough interest in the topic to justify the writing effort. Since that time, I have (in the course of talking to our customers) discovered that not only is there interest in the topic, but there is also a woeful lack of information and an abundance of misinformation regarding Atari's OS. So, with this column, we start a three or four part series on assembly language I/O.

Also, this month's column includes a list of major, known bugs in Atari BASIC and how to get around them.

Atari I/O, Part One: Interfacing To OS

Before I get started with the hairy details, I would like to state that Atari has the *best operating system* in the low-end microcomputer market. There is a simple reason for this: Atari has the *only* operating system on the market! Now, admittedly, I am being a purist when I make this contention, but the truth is that the Atari is the only machine I know of that has a *true* operating system in ROM. And, no, neither my company (Optimized Systems Software) nor I were involved in the creation of that operating system; the credit must go straight to Atari.

The operating system is contained in ROM and is identical on both the Atari 400 and Atari 800. The 10K bytes of ROM you may have noticed contain not only the operating system, but also the upper/lower case character set, the floating point mathematical operations, the power-on and cartridge select logic, and the device drivers. Device drivers? Aren't those part of the operating system? No! An emphatic "no." And that's what enables me to say that Atari has the only true operating system.

Believe it or not, the operating system on the Atari occupies less than 700 bytes. And yet it is as complete in its own way as UNIX is on a large time sharing machine. How many times have you read a magazine and seen lists of addresses of I/O subroutines for XYZ computer? You must use this address to output a character to the screen, another to get a character from the keyboard, yet another to talk to the line printer, and disk I/O? A nightmare! Not so Atari. One and only one address need be remembered: Hex E456, Decimal 58454. (Yes, I know, why not E400 or F000 or some such. Well, I didn't say Atari was perfect, only good.)

With only one address that matters, you can imagine that it should be easy for Atari to come out with new versions of the OS without affecting any other programs. They have, and they are, and only programs that have "cheated" (gone outside the OS rules) are in trouble. So, don't get yourself in trouble; follow the OS rules.

Finally, to avoid duplication of effort, I would refer you to the massive program listing for "SHOOT" in **COMPUTE!** #16: the first two pages and first column of the next two pages constitute most of the useful equates when using the Atari from assembly language. We are concerned with the "Operating System Equates" (first column, second page) and "Page three RAM assignments" (third column, first page). I will use the mnemonics given in that listing throughout this series of articles. (Those of you who own our "OS/A+" system will find these equates, with some mnemonics altered slightly for consistency, in the file "SYSEQU.ASM". You may use the EASMD pseudo-op ".INCLUDE #D:SYSEQU.ASM" to include them in any assembly programs. This will save you some typing.)

...the operating system on the Atari occupies less than 700 bytes.

The Structure Of The IOCB's

When a program calls the OS through location $E456, OS expects to be given the address of a properly formatted IOCB (Input Output Control Block). For simplicity, Atari has predefined eight IOCB's, each 16 bytes long, and the program specifies which one to use by passing the IOCB number times 16 in the 6502's X-register. Thus, to access IOCB number four, the X-register should contain $40 on entry to OS. Notice that the IOCB number corresponds directly to the file number in BASIC (as in PRINT #6, etc.). Actually, the IOCB's are located from $0340 to $03BF (refer to the "SHOOT" listing).

When OS gets control, it uses the X-register to inspect the appropriate IOCB and determine just what it was that the user wanted done. Table 1 gives the Atari standard names for each field in the IOCB along with a short description of the purpose of the field. Study the Table before proceeding.

The user program should *never* touch fields ICHID,ICDNO, ICSTA and ICPTL/ICPTH. In addition, unless the particular device and I/O request requires it, the program should not change ICAX1 through ICAX6. The most important field is the one-byte command code, ICCOM, which tells the operating system what function is desired.

The OS itself only understands a few fundamental commands, but Atari wisely provided for extended commands necessary to some de-

Now for adults.

Textwizard™ transforms Atari into a powerfully serious word processor.

Textwizard™ is no kids game. It's a sophisticated and complete word processing program for the **Atari 800 computer.

Here is instant control over the creation, editing and formating of any writing. Insert words. Replace phrases. Delete sentences. Move paragraphs. Automatically repositions surrounding text. *Textwizard™ does it all with over 50 simple commands.

On your command, Textwizard™ will search out and correct mistakes throughout your document. And it protects you from common operating errors by warning you before you make them.

Textwizard™ will print out as many original "copies" as you need. In boldface, elongated or condensed lettering, upper or lowercase, with proportional spacing. And you can store your work on a diskette.

So, whether you write legal briefs, computer programs, or The Great American novel, Textwizard™ can make your work easier. And though it's no toy, it's fun to use.

At $99.95, you don't have to write a best-seller to afford Textwizard.™ Textwizard™ is another of the many creative products from Datasoft.

Datasoft Inc.™
COMPUTER SOFTWARE

Software for people who aren't easy to please.

19519 Business Center Drive / Northridge, California 91324 / (213) 701-5161

Check your local software dealer or
Send check or money order with $2.00 postage/handling. California residents, add 6% sales tax.

*Textwizard will perform on a 32K system with one or more disc drives. It is compatible with the Atari® 825, Centronics® 737 and Epson® MX-80 printers.
**Atari is a registered trademark of Atari Computers Inc.

vices (XIO in BASIC). In any case, each one of these fundamental commands deserves a short description.

OPEN Open a device (synonyms: file, IOCB, channel) for read and/or write access. OS expects ICAX1 to contain a byte that specifies

...the drivers account for over 5K bytes of the ROM code. The screen handler, with all its associated editing and Graphics modes, occupies about 3K bytes of that.

the mode of access: ICAX1 = 4 for read access, 8 for write access, and 12 for both read and write access. (Note: the disk file manager and the screen device handler allow other modes, and they will be discussed in a later section.) The name of the device (and, for the disk, the file) must be given to OS; this is accomplished by placing the ADDRESS of a string containing the name in ICBAL/ICBAH.

CLOSE Terminate access to a device/file. Only the command must be given.

STATUS Request the status of a device/file. The device can interpret this request as it wishes, and pass back a (hopefully) meaningful status. As with OPEN, the ADDRESS of a filename must be placed in ICBAL/ICBAH.

GET TEXT A powerful command, this causes the OS to retrieve ("GET") bytes one at a time from a device/file already OPENed until either the buffer space provided by the user is exhausted or an Atari RETURN character (Hex 9B, Decimal 155) is encountered. The user specifies the buffer to use by placing its ADDRESS in ICBAL/ICBAH and its size (length) in ICBLL/ICBLH.

PUT TEXT The analog of GET TEXT,OS outputs characters one at a time until a RETURN is encountered or the buffer is empty. Requires ICBAL/ICBAH and ICBLL/ICBLH to be specified.

GET DATA Extremely flexible command, this causes OS to retrieve, from the device/file previously OPENed, the number of bytes specified by ICBLL/ICBLH into the buffer specified by ICBAL/ICBAH. *No checks whatsoever are performed on the contents of the transferred data.*

PUT DATA Similar to GET DATA, except that OS will output ICBLL/ICBLH bytes from the buffer specified by ICBAL/ICBAH. Again, no data checks are performed.

Table 2 provides the OS commands and their usage of the various fields of the IOCB's. For convenience, the disk file manager extended commands are also shown, but I must withhold discussion of them until next month.

Device names on the Atari computers are very simplistic; they consist of a single letter (optionally followed by a single numeral). Traditionally (and, in the case of disk files, of necessity) the device name is followed by a colon. You have probably seen these device names in your various Atari manuals, but a quick summary might be convenient:

E: The keyboard/screen editor device. The normal console output.
K: The keyboard alone. Use this device to bypass editing of user input.
S: The screen alone. Can be either characters (à la E:) or graphics.
P: The printer. The standard device driver allows only one printer.
C: The cassette recorder.
D: The disk file manager, which also usually requires a file name.

Other device names are possible (e.g., for RS-232 interfaces) and, in fact, the ease with which other devices may be added is another reason for my claim that Atari has a *true* operating system. The structure of device drivers is material for a later article, but I should like to point out that the OS ROM includes drivers for all the above except the disk. In fact, the drivers account for over 5K bytes of the ROM code. The screen handler, with all its associated editing and Graphics modes, occupies about 3K bytes of that.

Actually, the next column will begin to delve deeper into the ways of using OS, but for those of you anxious and brave enough to get started now we present a very simple example program:

```
PUTMSG                  ; A ROUTINE TO PRINT A
                          MESSAGE
        LDX   #$00      ; WE USE IOCB NUMBER 0, THE
                          CONSOLE (E:)
        LDA   #PUTREC
        STA   ICCOM,X   ; THE COMMAND IS 'PUT TEXT
                          RECORD'
        LDA   #MSG&255
        STA   ICBAL,X   ; LOWER BYTE OF ADDRESS
                          OF 'MSG'
        LDA   #MSG/256
        STA   ICBAH,X   ; UPPER BYTE OF ADDRESS
        LDA   #255
        STA   ICBLL,X   ; LOWER BYTE OF LENGTH OF
                          MSG
        STA   ICBLH,X   ; UPPER BYTE, LENGTH IS ALL
                          OF MEMORY
                        ; BUT 'PUTREC' WILL STOP
                          WITH THE 'RETURN' CHAR
        JSR   CIOV      ; CALL THE OS TO DO THE
                          WORK
        TYA             ; MOVES RETURNED ERROR
                          CODE TO A-REGISTER
        BMI   ERROR     ; ANY NEGATIVE VALUE IS
                          SOME SORT OF ERROR
```

Table 1.

FIELD NAME	OFFSET WITHIN IOCB (BYTES)	SIZE OF FIELD (BYTES)	PURPOSE OF FIELD
ICHID	0	1	SET BY OS. Index into device name table for currently OPEN file, set to $FF if no file open on this IOCB.
ICDNO	1	1	SET BY OS. Device number (e.g., 1 for "D1:xxx" or 2 for "D2:yyy")
ICCOM	2	1	The COMMAND request from user program. Defines how rest of IOCB is formatted.
ICSTA	3	1	SET BY OS. Last status returned by device. Not necessarily the status returned via STATUS command request.
ICBAL ICBAH	4	2	BUFFER ADDRESS. A two byte address in normal 6502 low/high order. Specifies address of buffer for data transfer or address of filename for OPEN, STATUS, etc.
ICPTL ICPTH	6	2	SET BY OS. Address minus one of device's put-one-byte routine. Possibly useful when high speed single byte transfers are needed.
ICBLL ICBLH	8	2	BUFFER LENGTH. Specifies maximum number of bytes to transfer for PUT/GET operations. *Note:* this length is decremented by one for each byte transferred.
ICAX1	10	1	Auxiliary byte number one. Used in OPEN to specify kind of file access needed. Some drivers can make additional use of this byte.
ICAX2	11	1	Auxilliary byte number two. Some serial port functions may use this byte. This and all following AUX bytes are for special use by each device driver.
ICAX3 ICAX4	12	2	For disk files only: where the disk sector number is passed by NOTE and POINT. (These bytes could be used separately by other drivers.)
ICAX5	14	1	For disk files only: the byte-within-sector number passed by NOTE and POINT.
ICAX6	15	1	A spare auxilliary byte.

```
        ...
        ...                          ;CONTINUE WITH MORE
                                      CODE
        ...
MSG     .BYTE   'THIS IS A MESSAGE',$9B
```

Just a very few notes on this routine: (1) If the command had been "GETREC," the OS would have gotten a line from the keyboard and put it into the "buffer" at MSG. (2) If the X-register had been set to $20 and if the printer had previously been OPENed at IOCB number 2, then *this same code* would have sent the message to the printer. (3) If the buffer length had been given as less than 18, the message would have been truncated to the specified length. That's all on I/O for this month. I hope you will hound your mailbox until your next issue of **COMPUTE!** arrives.

Bugs In BASIC

Several people have requested a list of all known bugs in Atari BASIC. The following list may not be complete, but it certainly enumerates all the bugs that may be considered "killers."

1. In the course of editing a BASIC program, sometimes the system loses all or part of the program and/or simply hangs. Often, turning power off and back on is the only solution. Contrary to popular belief, this condition is related to nothing except the size of the program that is being moved by a delete operation (*not* the size of the deleted line). FIX: NONE. Sorry about that. Just be sure and SAVE your programs often, especially if you are doing heavy editing.

2. String assignments that involve the movement of multiples of 256 bytes do not move the first 256 bytes. FIX: don't move multiples of 256 bytes. An easy way to accomplish this is to always move an ODD number of bytes. Usually, moving one extra byte is fairly easy to handle.

3. The cassette handler doesn't always properly initialize its hardware interface. Symptoms: ERROR 138 and ERROR 143. FIX: use an LPRINT before doing a CSAVE, etc. (This isn't a BASIC bug, but BASIC can be used to fix it.)

4. Taking the unary minus of a zero number (e.g., PRINT -0) can result in garbage. Usually this garbage will not affect subsequent calculations, but it does print strangely. FIX: don't use the unary minus in cases where there may be a doubt (e.g., use PRINT 0-x if 'x' might be zero).

5. Strange things can happen if you type in a program line longer than three screen lines long. Reason: the system editor device (E:) cuts off your input at three lines and gives it to BASIC, which processes it as is, and then E: gives the rest of your input to BASIC as the next line! FIX: don't try to put in program

ATARI 400/800 OWNERS: THERE'S A MICROCONNECTION™ FOR YOU! —

Now you can direct-connect to the telephone with or without the 850™ interface, there's also a serial port to drive a printer, optional autodial and autoanswer, and smart terminal software! Prices start at $199.50 For more details write or phone:

the microperipheral corporation

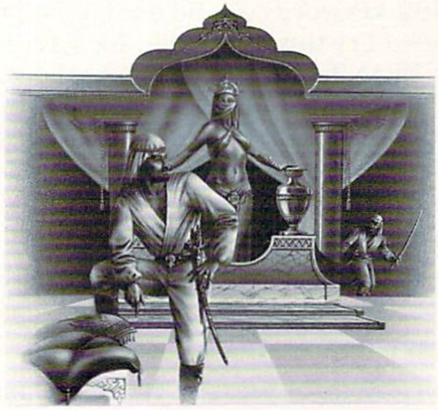
2643A - 151st Pl. N.E.
Redmond, WA 98052
(206) 881-7544

master charge
VISA

Fantasy for your ATARI

Ali Baba and the forty thieves

By Stuart Smith



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

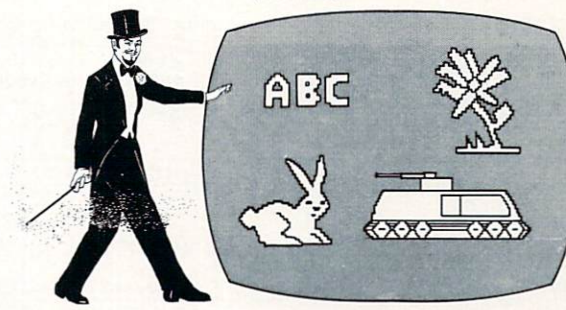
ALI BABA AND THE FORTY THIEVES is written in high resolution color graphics and includes music and sound effects. Adventures can be saved to disk and resumed at a later time. Requires 32K.

On Diskette Only — $32.95

Graphics for your ATARI

Character Magic

By Chris Hull



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

Cassette or Diskette - $29.95

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

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

QS QUALITY SOFTWARE

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

*Indicates trademarks of Atari

ATARI®

Software for Personal Computers

A collection of 10 challenging programs created to provide a unique entertainment value — and 2 personal/business programs with broad functional value. In disk and/or cassette as indicated.

GAMES	• HELICOPTER BATTLE	Req. — 16K RAM/Cassette	$ 9.95
		— 16K RAM/Disk	14.95
	• HORSE RACING	Req. — 16K RAM/Cassette	9.95
		— 16K RAM/Disk	14.95
	• KENO	Req. — 8K RAM/Cassette	9.95
		— 16K RAM/Disk	14.95
	• LIGHTNING BOLTS and REACTION	Req. — 16K RAM/Cassette	9.95
		— 24K RAM/Disk	14.95
	• THE MAD MARBLE	Req. — 8K RAM/Cassette	9.95
		— 16K RAM/Disk	14.95
	• MUSIGAME (2 Games)	Req. — 16K RAM/Cassette	9.95
		— 24K RAM/Disk	14.95
	• SUPERMASTER	Req. — 8K RAM/Cassette	9.95
		— 16K RAM/Disk	14.95
	• TAG	Req. — 16K RAM/Cassette	9.95
		— 16K RAM/Disk	14.95
	• TRACTOR BEAM	Req. — 8K RAM/Cassette	9.95
		— 16K RAM/Disk	14.95
	• WAR AT SEA	Req. — 16K RAM/Cassette	14.95
		— 24K RAM/Disk	19.95
BUSINESS	• CCA Data Management System	Req. — 48K RAM/Disk	99.95
	• LETTER WRITER	Req. — 24K RAM/Disk	19.95

Mastercard & VISA Accepted

CE
DIVISION OF CUSTOM ELECTRONICS, INC.
SOFTWARE

238 Exchange St., Chicopee, Massachusetts 01013
(413) 592-4761

• Dealer And Distributor Inquiries Invited
• Closed Mondays — Open Daily 'Til 5:30 — Fridays 'Til 8

lines bigger than three screen lines.

6. Using an INPUT statement without a variable (i.e., just '10 INPUT') does *not* cause a syntax error (it should) and may cause program lock-up when RUN. FIX: don't do it. (What did you expect? BASIC is in ROM, so *it* can't be fixed.)

7. Most keywords can be used as variable names. (Try this sometime: LET LET=5 : LET PRINT=3 : PRINT PRINT : PRINT LET ... it works!) Some cannot, and BASIC will tell you about them. But 'NOT' cannot be the first three letters of any variable name. Example:

```
10 LET NOTE=5 : PRINT NOTE
```

If you enter that line and then LIST it, you will get

```
10 LET NOTE=5 : PRINT NOT E
```

because in an expression NOT is a unary operator that is never seen as part of a variable name. (In the LET, only a variable name is expected, so NOT is never seen.) This is the only "poison" keyword in Atari BASIC. (Note the use of 'LET' in several instances above. Generally, assignment to a variable name which starts with a keyword requires the use of LET to avoid confusing the syntaxer.)

8. LOCATE and GET do not reinitialize their buffer pointer, so they can do nasty things to memory if used directly after some statements (e.g., they can change the line number of a DATA statement if used after a READ). FIX: reinitialize the pointer by using a STR$ function call (e.g., XX=STR$(0) works fine). Clumsy, but it works. PRINTing a numeric value works also (since PRINT calls STR$ internally). This fix is probably one you can ignore until it happens to you.

9. An INPUT of more than 128 bytes (from disk, cassette, etc.) will write into the lower half of page six RAM ($0600-$0657F). This is *not* a bug, it was designed that way. The lower half of page six was supposed to be available to BASIC, but someone at Atari forgot to tell someone else at Atari (and even two different memory maps in the Atari BASIC Reference Manual don't agree). As a consequence, both Atari and user programmers have come to regard all of page six as their own and have put small assembly language programs there. FIX: don't use the programs from $0600-$067F or don't INPUT such long strings.

There are a few other minor bugs (e.g., you can say DIM A(32766,32766) without getting an error message), but, by and large, they won't affect most programs. If anyone thinks they know of any other major bugs, let me know and I will try to provide a fix. Please let us know what topics you want covered.

IOCB field offset and name / Type of command	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
					BUFFER ADDRESS		PUT-A-BYTE ADDRESS		BUFFER LENGTH								
	ICHID	ICDNO	ICCOM	ICSTA	ICBAL	ICBAH	ICPTL	ICPTH	ICBLL	ICBLH	ICAX1	ICAX2	ICAX3	ICAX4	ICAX5	ICAX6	Mnemonic label used by Atari for this field
OPEN	•	•	3	•	FILENAME		•	•			SEE TEXT						OPEN
CLOSE	•		12	•													CLOSE
DYNAMIC STATUS		•	13	•	FILENAME												STATIS
GET TEXT RECORD			5	•	BUFFER				LENGTH								GETREC
PUT TEXT RECORD			9	•	BUFFER				LENGTH								PUTREC
GET BINARY RECORD			7	•	BUFFER				LENGTH								GETCHR
PUT BINARY RECORD			11	•	BUFFER				LENGTH								PUTCHR
	EXTENDED COMMANDS: DISK FILE MANAGER ONLY																
RENAME		•	32	•	FILENAME												RENAME
ERASE		•	33	•	FILENAME												DELETE
PROTECT		•	35	•	FILENAME												LOCKFL
UNPROTECT		•	36	•	FILENAME												UNLOCK
NOTE			38	•									SECTOR NUMBER		BYTE		NOTE
POINT			37	•									SECTOR NUMBER		BYTE		POINT

—LEGEND—

* — Set by OS when this comand is used.
BUFFER — 16-bit address of a data buffer.
FILENAME — 16-bit address of a filename.
LENGTH — length (in bytes) of a data buffer.
SECTOR NUMBER and BYTE — see text.

©

Atari Timing Delays

Editor's Note: Timer #4 will not work correctly due to an error in the Atari operating system. — RTM

Jim Clark
Seattle, WA

Timing delays are frequently needed in programs, and a common way to implement a delay in BASIC programs is to code a FOR-NEXT loop that does nothing other than to loop a specified number of times. FOR-NEXT loops are difficult to calibrate in Atari BASIC, however, because the time required for a loop will vary, depending on the location of the loop in the program and the size of the program.

I had a need for precise timing delays when I was writing a telephone dialer program for use with an autodial modem. One possible solution was to use Atari's built-in, realtime clock (for example, see "Real-Time Clock on the Atari" by Richard Bills, **COMPUTE!** #12, pg. 88.) Coding the delay would involve setting the clock to zero with POKE statements and then using PEEK statements in a loop to compare the current time with the amount of time that must pass. This procedure is awkward and I discovered that the Atari provides a better way.

In addition to the realtime clock, which measures time in the usual "forward" direction, the Atari also contains several countdown timers, which measure time "backwards" to zero from some initial value, which you set. When a timer has counted down to zero, the Atari "rings a bell," so to speak, just like a kitchen timer used for cooking. The timers are updated 60 times per second, which permits the coding of delays in multiples of 1/60 of a second (.0167 second).

One limitation to the use of the timers is that they can only be accessed from an assembly language program. There is no way to set a timer by way of POKE statements in BASIC. Program 1 shows an assembly language subroutine called TIMER which can be used to set system timer number three. The subroutine is POKEed into memory and then called with a BASIC USR statement. The example program asks for a value in seconds. The program multiplies the seconds by 60 to get the number of "ticks," which is used to set the timer. The TIMER subroutine sets the timer and then waits until it counts down to zero, at which point TIMER returns to the BASIC program. If you run the program and enter, say, 1 second, you will observe a one second delay, and then the program will ask for another delay value. Enter 10 and it will wait for 10 seconds until it asks again, and so on. You must enter a number greater than zero, and the longest delay you can use is 65,536 "ticks," which is 1,092 seconds, or about 18 minutes.

The listing has assembly language codes in DATA statements interspersed with REM statements in order to clearly show the assembly language subroutine. Lines 100 to 140 POKE the subroutine into page six, which is an area of RAM that is reserved for applications programs and which is not used directly by BASIC. You may have other uses for page six, so an alternative way to store the assembly language program is in a string array. You can put the lines shown in Program 2 into your program as an alternative to the lines with the same numbers in Program 1. In Program 2, the USR function is given the address of the string (ADR(TIMER$)) instead of the address of page six (1536) as in Program 1.

Program 2.

```
110 DIM TIMER$ (25)
120 FOR I=1 TO 25
130 READ BYTE:TIMER$ (I)=CHR$ (BYTE)
140 NEXT I
240 Z=USR(ADR(TIMER$),TICKS)
```

TIMER could be modified to return immediately, with the timer ticking, by deleting lines 650 through 690. (Also change the 24 in line 120 of Program 1 to 19). In this case, your BASIC program must PEEK location 554 to find out if the amount of time you set has passed. IF PEEK(554) =0 THEN time is up, otherwise IF PEEK(544)<>0 THEN time is not up yet.

The Atari has five built-in timers, which creates the potential for control of very complex situations. Two of the timers "ring their bells" by calling a subroutine (which you can provide) and three "ring their bells" by setting a memory location to 0. Timer number three works this second way — the TIMER subroutine in Program 1 sets the memory location to a nonzero value at line 610, and then just repeatedly examines the location at line 650. When the value at the location goes to zero, the branch at line 670 will no longer be taken, and the subroutine returns to the BASIC program.

Timers one and two, which call a subroutine upon reaching zero, must have the subroutine entry address placed at a special location in memory, as shown in the following table:

Timer	Subroutine Entry Atari Name	Address Address
1	CDTMA1	$0226
2	CDTMA2	$0228

Timers three, four, and five use flags at the memory locations shown in the following table:

Timer	Atari Name	Flag Address
3	CDTMF3	$022A
4	CDTMF4	$022C
5	CDTMF5	$022E

The Atari Operating System provides a special subroutine called SETVBV to set the timers. (The need to use this subroutine is the reason that the timers can not be set directly from BASIC.) To use SETVBV, either put the address of the subroutine you want to have executed when the time is up into CDTMA1 or CDTMA2 (in the case of timer one or timer two,) or set one of the flags CDTMF3, CDTMF4, or CDTMF5 to a nonzero value (in the case of timers three, four, or five). You must then load the high-order byte of your countdown time into index register X, load the low-order byte of the time into index register Y, and load the number of the timer you wish to use (one through five) into the accumulator (register A). Finally, code a JSR SETVBV to set the timer. The address of the SETVBV subroutine is $E45C. Lines 430 through 640 in Program 1 illustrate this calling procedure.

One final note — you should probably not use timer one. The Atari Operating System itself uses this timer in its input-output routines. If you attempt to use it, you may interfere with the operation of your system.

Program 1.

```
10 REM DELAY.BAS
20 REM DEMONSTRATE USE OF ATARI COUNT-DO
WN TIMER FROM BASIC
30 REM BY JIM CLARK
100 REM PUT ASSEMBLY SUBROUTINE "TIMER"
ON PAGE 6
110 PAGE6=1536
120 FOR ADDR=PAGE6 TO PAGE6+24
130 READ BYTE:POKE ADDR,BYTE
140 NEXT ADDR
200 REM DEMONSTRATION OF CALL TO "TIMER"

210 PRINT "SECONDS DELAY";
220 INPUT SECONDS
230 TICKS=60*SECONDS
240 Z=USR(PAGE6,TICKS)
250 GOTO 200
300 REM SUBROUTINE TIMER
310 REM CALLING SEQUENCE FROM BASIC:
320 REM Z=USR(ENTRY,TICKS)
330 REM
340 REM RETURNS TO BASIC AFTER TICKS/60
SECONDS
350 REM SUBROUTINE IS COMPLETELY RELOCAT
ABLE
360 REM SO ORIGIN FOR ASSEMBLY IS NOT SH
OWN HERE
370 REM SYMBOL EQUATES TO ATARI OPERATIN
G SYSTEM:
380 REM CDTMF3=$022A       ;FLAG FOR SYST
EM TIMER #3
390 REM SETVBV=$E45C       ;ENTRY POINT F
OR SYSTEM ROUTINE
400 REM ;                  TO SET TIMERS
410 REM PLA
420 DATA 104
430 REM PLA                ;GET HIGH-BYTE
 OF TICKS
440 DATA 104
450 REM TAX
460 DATA 170
470 REM PLA                ;GET LOW-BYTE
OF TICKS
480 DATA 104
490 REM TAY
500 DATA 168
510 REM BNE TIMER          ;MAKE SURE TIC
KS>0
520 DATA 208,4
530 REM TXA                ;LOW BYTE=0, H
OW ABOUT HIGH BYTE?
540 DATA 138
550 REM BNE TIMER
560 DATA 208,1
570 REM RTS                ;TICKS=0, JUST
 RETURN
580 DATA 96
590 REM TIMER LDA #3       ;USE SYSTEM TI
MER #3
600 DATA 169,3
610 REM STA CDTMF3         ;SET THE FLAG<
>0
620 DATA 141,42,2
630 REM JSR SETVBV         ;START THE TIM
ER
640 DATA 32,92,228
650 REM LOOP  LDA CDTMF3   ;WAIT TILL THE
 FLAG GOES TO 0
660 DATA 173,42,2
670 REM BNE LOOP
680 DATA 208,251
690 REM RTS                ;TIMER HAS COU
NTED DOWN TO 0, RETURN
700 DATA 96
999 END
```

©

GREAT GAMES FOR ATARI

Gin Rummy 3.0

The classic computer Gin Rummy game now available for Atari, with color graphics and sound. Plays a regulation game, and a tough game that will hold its own against anyone. Keeps score to game level. Disk version keeps a running score in disk file so you can start another session where you left off. 24K Cass. IGR/C $19.95, Disk IGR/D $24.95.

Casino Blackjack/Counter

Play at a very realistic casino table — learn to beat the house at its own game, or just play for fun following the recommended bets. You play one of five hands (the computer plays the others), and practice card counting as the cards are dealt. Choose up to 6 decks to play against, and set the dealing speed to slow, medium or fast. 16K Cass. IBJ/C $19.95, Disk IBJ/D $24.95.

Concentration

Excellent full-color graphics, great fun to play on the screen. Where *was* that sailboat you saw two turns ago? Choose up to 15 pairs of figures—a smaller game is fascinating for children, and 15 pairs will challenge anyone. For 2 players. 16K Cass. ICO/C $14.95.

Atari 3-Game Pack

These three great Atari games — Gin Rummy, Blackjack and Concentration — on a single disk, and at less than the cost of the three separately. $49.95.

ALL PROGRAMS OPERATE WITH JOYSTICKS

AT YOUR DEALER OR DIRECT FROM:

MANHATTAN SOFTWARE
POST OFFICE BOX 1063
WOODLAND HILLS, CA 91365

California residents add 6% sales tax

24-hour Visa and MasterCard order line:

(213) 704-8495

RAM

For ATARI

Get the most from your ATARI 400. Memory expansion to a full 48K is now possible with our 48K Board. Expand your ATARI 800 with our 32K Board.

48K Board $299
32K Board $199

INTEC
Suite # 111
3387 Del Rosa Ave. North
San Bernardino, CA 92404
(714) 864-5269

CA residents add 6 percent tax

ATARI is trademark of ATARI, Inc.

ATARI™ 400* & 800*

Computer Software

Announcing:

KAYOS

$34.95

IN 8K MACHINE LANGUAGE DISK OR TAPE.

*The longest lines at the Video/Electronics show in N.Y. were people waiting to play KAYOS!

CM 1001 — HORSEWORD DERBY**	**$19.95**
CM 1002 — AVOID**	**$19.95**
CM 1003 — TRIBOMB**	**$19.95**
CM 2001 — KAYOS	**$34.95**

**Requires 16K of RAM

TO ORDER SEND $1.50 SHIPPING

COMPUTER MAGIC LTD.
40 Wakefield Ave.
Port Washington, NY 11050

Dealer Inquiries
516-883-0094

educational software for the ATARI* educational software for the APPLE II+ educational software for the ATARI* educational software for the APPLE II+** educational software for the ATARI* educational software for the APPLE II+** educational software for the ATARI* educational software for the APPLE II+** educational software for the ATARI* educational**

HIDDEN WORDS

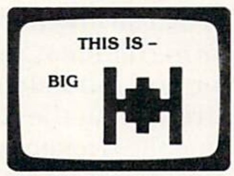
A word-search program for the ATARI*. Move your character over the letters that spell your word. Words may be read from left, right, up, down, or diagonally.

- For one or two players
- Four levels of play
- Comes with word base of the most frequently misspelled words.
- Requires JOYSTICKS

24K BASIC, Grades 1-8+ . . . **$17.50**

SPATIAL RELATIONS

For the ATARI*



Designed to help the child learn words that indicate size and placement. Units cover: big/small, tall/short, right/left, high/low, over/under, in/out. Requires one joystick. Minimal reading. SIDE TWO REQUIRES NO READING - Use with Type-'N-Talk™ speech synthesizer.

16K BASIC, Preschool-1 . . . . **$17.50**

Type-N'-Talk™*** speech synthesizer (VOTRAX).

Let your computer talk with one of the easiest to use speech synthesizers on the market.

Requires RS-232 interface. **$375.00**

T.H.E.S.I.S.

P.O. Box 147 • Garden City, MI 48135
or call (313) 595-4722 for C.O.D.

Some T.H.E.S.I.S. software is now available for the APPLE II+

DEALER INQUIRIES WELCOME

Please add: $1.50 shipping/handling, $1.50 for C.O.D., Mich. res. add 4% sales tax.

WRITE FOR FREE CALALOG

*ATARI is trademark of Atari, Inc. **APPLE is trademark of Apple Computer, Inc. ***Type-'N-Talk is trademark of Votrax.

Printing Numbers That Make Cents

Tina Halcomb
Carrolton, TX

In the *Atari Basic Reference Manual* (page 39), the section on String Manipulations shows a way to print selected portions of a string. That is, sometimes we need to control the size of items that are being printed due to limited space. Or maybe we just want a prettier, more uniform format.

When dealing with numbers — complex computations and so on — accuracy can be very important. Atari gives several digits of accuracy and, in some cases, the numbers are expressed in terms of scientific notation.

If you were to write an accounting program you would want to have the figures as accurate as possible, but, in most cases, you wouldn't need any more than two decimal places printed. This is especially true if you are speaking in terms of dollars and cents. You might also want the fraction rounded to the nearest 100th. The following formula will do this for you:

```
A=INT((A+.005)*100)/100
```

It rounds the number off to the nearest 100th and disregards anything beyond two decimal places. This works very well, however, if you were to enter a number such as 79.00, the Atari automatically deletes the decimal point and trailing zeros. They would be necessary for a consistent printout. Lines 70 and 80 of the example program check for decimal point position and pad with zeros if necessary.

You can also line up the decimal points for a uniform columnar printout. Using an X and Y cursor position statement, subtracting the length of the string from the X (right boundary) coordinate will line the decimal points up very nicely. And, of course, you will increment the Y coordinate according to the desired horizontal spacing.

To see how all of this works, type the example program in and run it.

```
10 DIM B$(20)
20 Y=6
30 GRAPHICS 0
40 INPUT A
50 A=INT((A+.005)*100)/100
60 B$="00":B$(LEN(B$)+1)=STR$(A)
70 IF B$(LEN(B$)-1,LEN(B$)-1)="." THEN B$
   (LEN(B$)+1)="0":REM check for 1 decimal
   place
80 IF B$(LEN(B$)-2,LEN(B$)-2)<>"." THEN B$
   (LEN(B$)+1)=".00":REM check for 0 decimal
   places
100 GOSUB 200
110 GOTO 40
200 X=20-LEN(B$):Y=Y+1
210 POSITION X,Y:PRINT B$(3)
220 RETURN
```

©

Formatting Input

Kenneth J. Freese
East Meadow, NY

Data entry can frequently be simplified by formatting the screen as a table and entering the data into the table. This is in contrast to the usual method of serial prompt statements. Presented here are two simple routines for this technique. Both examples are for numeric input, but can easily be adapted for strings.

Program 1 utilizes the Keyboard (K:) as the input device. The data is entered without prompt. The variable RC in lines 150, 180 and 200 represents the right margin against which the data will be placed. The subroutine beginning at line 500 accomplishes the right justification (see **COMPUTE!** #10, pg. 84). Memory location 84 ("ROWCRS") holds the value of the current cursor row so line 520 will position the screen output on the proper line. One drawback is not seeing the input until the entire is keyed in and the RETURN is pressed. Another problem is that an inadvertently pressed key cannot be DELETEd so that correction requires a routine to alter the disk file.

Program 2 uses the Screen Editor (E:) as the input. This is the more familiar technique to most programmers since it is the default mode of the input statement. Since a semicolon cannot be used to "hold" the line after an input function, the POSITION statements in lines 150 and 160 use a PEEK(84)-1 to move the cursor up to maintain the same line. The screen display is not as appealing as in Program 1 because question marks precede every entry and there is no right margin justification. However, this may be a small price to pay to see entries as they are being typed and allowing for changes in entries prior to hitting the RETURN key.

Program 1.

```
100 DIM X$(10)
110 OPEN #2,4,0,"K:"
120 OPEN #3,8,0,"D:INFO"
130 ? "  NO          CODE          PROCEDU
RE"
140 FOR X=1 TO 37:? "-";:NEXT X:? :?
150 INPUT #2;NO:X=NO:RC=6:GOSUB 500
160 ? NO;
170 IF NO=0 THEN 240
180 INPUT #2;CODE:X=CODE:RC=19:GOSUB 500

190 ? CODE;
200 INPUT #2;PROCEDURE:X=PROCEDURE:RC=34
:GOSUB 500
210 ? PROCEDURE
220 ? #3;NO;",";CODE;",";PROCEDURE
230 GOTO 150
240 CLOSE #2:CLOSE #3
250 END
500 X$=STR$(X)
510 LC=RC+1-LEN(X$)
520 POSITION LC,PEEK(84)
530 RETURN
```

Program 2.

```
100 OPEN #3,8,0,"D:INFO"
110 ? "  NO          CODE          PROCEDU
RE"
120 FOR X=1 TO 37:? "-";:NEXT X:? :?
130 INPUT NO
140 IF NO=0 THEN 200
150 POSITION 15,PEEK(84)-1:INPUT CODE
160 POSITION 28,PEEK(84)-1:INPUT PROCEDU
RE
170 ? #3;NO;",";CODE;",";PROCEDURE
180 GOTO 130
200 CLOSE #3
```

©

Typing SHOOT

Since SHOOT was published in **COMPUTE!** #16, we've received numerous letters from readers who were having problems with it. Rest assured, if you type in the Boot Tape Maker correctly, it *will* make a perfect bootable copy of SHOOT.

It is not easy to type in a long computer program listing, especially one like SHOOT, with its huge table of DATA statements. Here are some tips on how to make it easier, and what to watch out for:

1. Some beginners do not fully realize that a program must be typed in *exactly* as listed. Seemingly trivial punctuation marks like the semicolon are absolutely necessary for proper execution.

2. Look out for numbers, too. For example, in line 200 of the Boot Tape Maker, were the number 125120 mistyped, the whole effort will be flagged as an error.

3. Remember, SHOOT must run on a 16K or greater Atari.

4. Make sure the characters in quotes on line 300 are hhh, reverse-video asterisk, LV, reverse-video d.

5. Before running the Boot Tape Maker, you might want to enter "LPRINT" to insure that the cassette pointers are correct.

6. We reprint the program here in larger type.

```
0 DIM H$(1),B$(2),AD$(4),A$(60),BUF$(1148)
1 POKOFF=4*1024-1
2 GOTO 100
10 IF H$>="0" AND H$<="9" THEN D=ASC(H$)-48:RETURN
12 D=ASC(H$)-55:RETURN
20 H$=B$(1,1):GOSUB 10:B=D:H$=B$(2,2):GOSUB 10:B=B*16+D:CHECK=CHECK+B:RETURN
30 B$=AD$(1,2):GOSUB 20:AD=B:B$=AD$(3,4):GOSUB 20:AD=AD*256+B:RETURN
100 GRAPHICS 0
110 TRAP 900:LINE=1000:LSUM=0
120 READ A$:IF A$="END" THEN 200
130 ? LINE:CHECK=0:B$=A$(1,2):GOSUB 20:NOB=B
140 AD$=A$(3,6):GOSUB 30:FAD=AD
150 FOR I=1 TO NOB:B$=A$(5+2*I,6+2*I):GOSUB 20
160 M=FAD+I-1-POKOFF:BUF$(M,M)=CHR$(B):NEXT I
165 SUM=CHECK-65536*INT(CHECK/65536)
170 AD$=A$(LEN(A$)-3,LEN(A$)):GOSUB 30
180 IF SUM<>AD THEN 900
185 LSUM=LSUM+SUM:LINE=LINE+10:GOTO 120
200 IF LSUM<>125120 THEN ? "Too many/few lines":END
205 CLOSE #1
210 OPEN #1,8,128,"C:"
220 IOCB=832+16
230 POKE IOCB+2,11
240 BUF=ADR(BUF$)
250 POKE IOCB+4,BUF-(INT(BUF/256)*256)
260 POKE IOCB+5,INT(BUF/256)
270 BUFLEN=LEN(BUF$)
280 POKE IOCB+8,BUFLEN-(INT(BUFLEN/256)*256)
290 POKE IOCB+9,INT(BUFLEN/256)
300 DUMMY=USR(ADR("hhh*LVd"),16)
310 CLOSE #1
320 END
900 ? "ERROR IN#";LINE:END
1000 DATA 1810000009001008101860A93C8D02D3A916850AA910850B604C4E06A9
```

```
1010 DATA 18101812707070460018F070F070F070F070F070F070F070F070F00DF0
1020 DATA 18103070F070F070411910284329313938312O4A20482050414C45076D
1030 DATA 1810485649434880908292849486968898C808182838485868788 00A5A
1040 DATA 181060383800010203040506070 87C7C0102030201FFFEFDFE00000615
1050 DATA 181078010203040506070809 0A0B000100010001000100010100 0100E9
1060 DATA 181090FF000003060C1C3C7EFF00C06030383C7EFF00181818183C077E
1070 DATA 1810A87EFF0000F820F29E90F000001F044F79090F0000010D3F7F0844
1080 DATA 1810C01800000080B0FCFE18000018247E817E000001050008FF05070D
1090 DATA 1810D80000020A0118FE0A011003190020FD190020488A48A6B0E80708
1100 DATA 1810F0BD4C108D0AD48D1AD0A6B0AD08D02901F013A9009D62109D0B70
1110 DATA 1811086D10BD78101865B185B18D1ED0E886B0BD6210187D6D109D0ACE
1120 DATA 18112062108D00D0BD57108D12D0BD83108D08D068AA6840A5B1D00B40
1130 DATA 18113808A9808D03D24C901138E90185B1A98A8D03D2A205BD00180A4A
1140 DATA 18115018690109109D0018C91AD009A9109D0018CA4C4D11A5B7D00893
1150 DATA 18116827A6B5E886B5E00FD01EA20086B5A205BD0E1818690109900A95
1160 DATA 1811809D0E18C99A9009A9909D0E18CA4C7811A6B6E886B6E03CD00C6F
1170 DATA 18119822A20086B6A5B7D02EA205BD0E1838E90109909D0E18C99F0A8B
1180 DATA 1811B0D009A9999D0E18CA4CA311A900A2061D0D18CAD0FA290FC90AA9
1190 DATA 1811C800D004A90185B7A900854DAD78024A4AAABD8E1085B3CA8A0B72
1200 DATA 1811E00A0A0AAAA000BD921099601AC899601AE8C8C010D0F0A5B20C55
1210 DATA 1811F81865B385B28D04D0A5B4F026AAA9009D8019CAF011A5B1D00DD2
1220 DATA 1812101286B4A9FF9D80198E00D24C2A1286B44C2A12A2008E00D20A10
1230 DATA 18122886B4A5B7D016AD8402D011A5B4D00DA96285B4A5B30A0A180BE0
1240 DATA 181240698485B2A9FF85B08D1ED04CD1E7A9A88D01D2A9808D03D20E26
1250 DATA 181258A9008D00D2A9308D02D2A280A9009DFF199D7F19CAD0F7A90CB3
1260 DATA 18127000A2089DFFCFCAD0FAA92E8D2F02A9188D07D4A9038D1DD00C27
1270 DATA 181288A9100D6F028D6F028D1BD0A90085B4A90185B7A9408D0ED40A7F
1280 DATA 1812A0A9108D3102A9198D3002A9108D0102A9EA8D0002A211A03508B7
1290 DATA 1812B8A906205CE4A9C08D0ED4A9C68DC402A9368DC502A9188DC60CD2
1300 DATA 1812D002A90A8DC702A214BD3710200E1409C09DFF17CAD0F2A5130AC1
1310 DATA 1812E8186903C513D0FCA214A9009DFF17CAD0FAA9108D0518A9500C3C
1320 DATA 1813008D0C18A9908D1318A90185B7A9088D1FD0AD1FD0C901D0060A1C
1330 DATA 181318202A144C4E12C906D0EFA900A2069D0D189DFF17CAD0F7A90ADB
1340 DATA 181330918D1118A9928D1218A9908D1318A9108D0518A90085B7850952
1350 DATA 181348B185B685B5A218A00020C513C8C008D0F8A207A0D0A9038D0C95
1360 DATA 1813602A02205CE4A9C08D0ED4AD2A02D0034C4D13A008A900196C091D
1370 DATA 1813781088D0FAC900D00AA9321865B185B14C4D13A5B7F0DCA5130C73
1380 DATA 181390186902C513D0FCA200BD0718291FDD0018F005B0084CB0130959
1390 DATA 1813A8E8E006D0EC4C0813A206BDFF17291F09409D0618CAD0F34C0B64
1400 DATA 1813C008130000008EC2138CC313AD0AD22907C906B0F70A0AAABD0975
1410 DATA 1813D8D210996D10BDD3109978 10BDD41099831 0A900996210BDD50BCF
1420 DATA 1813F010AAACC213A9088DC413BDAA1099001AE8C8CEC413D0F3980D45
1430 DATA 181408AAACC31360008C0D14A88A48982A2A2A2A2903AA98299F1D087A
1440 DATA 181420F6FEA868AA98AC0D1460A220A90C9D42032056E4A9149D450B11
1450 DATA 18143803A9759D4403A9039D4203A9089D4A03A9809D4B032056E40900
1460 DATA 181450A9009D4403A9109D4503A9789D4803A9049D4903A90B9D4208D9
1470 DATA 101468032056E4A90C9D42032056E460433A9B0652
1480 DATA END
```

©

TextPlot

Charles Brannon
Editorial Assistant

TextPlot is a machine language graphics utility that lets you mix text and graphics. It is designed to work with the four-color graphics modes three, five, and seven. It will place any ATASCII character — upper/lowercase, graphics, numbers, and special symbols in normal or reverse field — on the graphics screen in any of three colors. The size of the characters varies in proportion to the pixel size: GRAPHICS 3 characters are *four times* as large as those in GRAPHICS 7, whose characters are the same size and proportions as those in GRAPHICS (text) mode two. Through standard display list modification, any of the three sizes of text can be mixed with the other modes. TextPlot enables you to use a total of *eight* text modes. (See the description of the "bonus" text modes later.)

Text On Graphics Lines

TextPlot, unlike the text modes, can be mixed on the same line with normal graphics. You can label charts and graphs, or quickly draw pictures with the graphics characters and then embellish them. TextPlot even works with an alternate character set, so you can design special "shapes" and move them around the screen for high-speed animation. The text in GRAPHICS mode three is *huge*, a real eyecatcher. Unlike the other text modes, TextPlot lets you position any character at any possible vertical resolution (although horizontally it's the same). And all this was without modifying the display list!

Luckily, TextPlot is easy to use. You load it into memory (it goes into the reserved memory at $600 hex) with a BASIC loader or BINARY LOAD, via DOS. You then select the graphics mode to use it in with the ordinary GRAPHICS command. (TextPlot works in either full-screen or window modes.) You then "plot" each character with the command:

A = USR(1536,chr,color,column,row)

Don't let this machine language call intimidate you. It merely enables a USeR command. The other variables for the function communicate with TextPlot. If you leave one out, or add an extra one, TextPlot will ring the bell to warn you.

CHR: The ASCII value of the desired character [like ASC("K")].

COLOR: The color of the character (just like the COLOR statement, 1–3).

COLUMN: The horizontal position of the character. This depends on the mode:

Table 1

Mode	Max Columns	Max Rows
3	5	16
5	10	40
6	10	88
7	20	88
8	20	184

Row: The vertical position of the character. This also depends on the mode (see Table 1), and is the line at which you want the character to start. Remember that each character is just eight lines of dots, so they can start at any pixel position vertically. The horizontal resolution is limited by the internal storage of graphics information on the screen.

So, to place a blue capital letter "A" on the screen in GRAPHICS mode three, at the second column and tenth row, use the command:

A = USR(1536,65,3,2,10)

65 is the ATASCII value of "A"; three is the color; two is the column; ten is the row. Strings of text can be placed on the screen as well:

```
DIM T$(20)
T$ = "That's Incredible!"
GRAPHICS 7 + 16
FOR I = 1 TO LEN(T$)
  A = ASC(T$(I,I))
  V = USR(1536,A,1,I,2)
NEXT I
```

Notice that you can use any variable with the USR function, not just A.

Bonus Text Modes

TextPlot was designed for the four color graphics modes. Strange things can happen if you use it in any other mode. In modes six and eight, however, you will indeed get text. In GRAPHICS 6, the characters are the same size as those in GRAPHICS 5. There is a blank line between each row of dots in each character. A character plotted in COLOR 1 or COLOR 2 will also skip horizontally. COLOR 3 will create characters divided into "bands." The effect is similar to the IBM logo (See Figure 1). This same oddity results in "artifacting" in GRAPHICS 8. What does that mean? You get three colors of text in GRAPHICS 8! Depending on background and dot colors, COLOR 1 is purple, COLOR 2 is green, and COLOR 3 is white. The text is twice the width of GRAPHICS 0 characters, but the same height, just like GRAPHICS 1.

I have included a sample program that lets you type on the screen using a flashing cursor. It works in GRAPHICS mode seven. You can use all the standard keys, but only a few of the editing keys work. What can I say? It's not supposed to be a word processor! The lines from 20000 and up will place TextPlot into memory at page six. You can save them to disk or tape and merge them with other programs using the LIST/ENTER combination.

For Cassette	**For Disk**
Rewind cassette, press PLAY & RECORD, and enter:	**Enter:**
LIST "C:",20000,32767	**LIST "D:TXTPLT.ENT", 20000,32767**
Press RETURN twice.	**and press RETURN.**
To merge with a program already in memory:	
Rewind tape, press PLAY, enter:	**Enter:**
ENTER "C:"	**ENTER "D:TXTPLT.ENT"**
and press RETURN twice.	

Advanced readers may want to know how TextPlot works (if you haven't figured it out already). You are probably familiar with how to plot characters on the GRAPHICS 8 screen by PEEKing the character generator and then placing these bit patterns directly into the screen memory for GRAPHICS 8. It works because each byte in GRAPHICS 8 (and modes four and six, too) displays eight dots, or pixels. A one-bit in the byte means a "lit" pixel and a zero is a dark ("background") dot. The four color modes have to split the load between two bytes. Each byte displays four pixels. Two bits hold the color (binary): 01 color one, 10 color two, 11 color three. (See Figure 2.) TextPlot uses the character generator (indirectly through CHBAS, 756 decimal) to get the bit map and then "pulls" the byte accordian-style into two color bytes. Theoretically, any character could be a mixture of the three colors, but it's hard to implement and use. (Unless you use IR modes 4 or 5...)

Using TextPlot as a BINARY FILE

The Atari DOS lets you store machine language files on the disk and load them back, both by DOS menu selections. You can even have TextPlot load in automatically with the DOS, if you're sure you'll always need it. After placing TextPlot into RAM, go to DOS with the command: DOS. If you have DOS 2.0S, there will be a pause as the Disk Utility Package loads. The DOS menu should be displayed. Type K ‹RETURN›. After the prompt, enter:

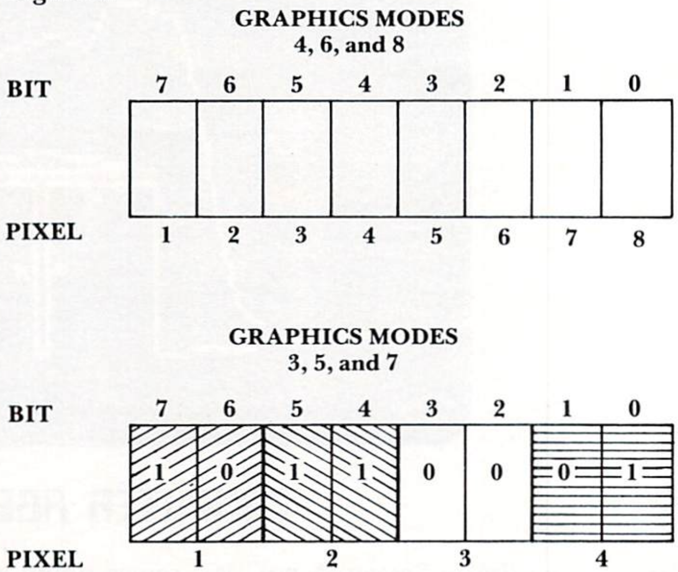
TXTPLT.OBJ,0600,06FF ‹RETURN›

If you want TextPlot to automatically load with DOS, enter:

AUTORUN.SYS,0600,06FF ‹RETURN›

instead. If you don't do this, you'll have to go to DOS and enter L (Load) and reply with TXTPLT.OBJ to load it and B ‹RETURN› to exit to BASIC.

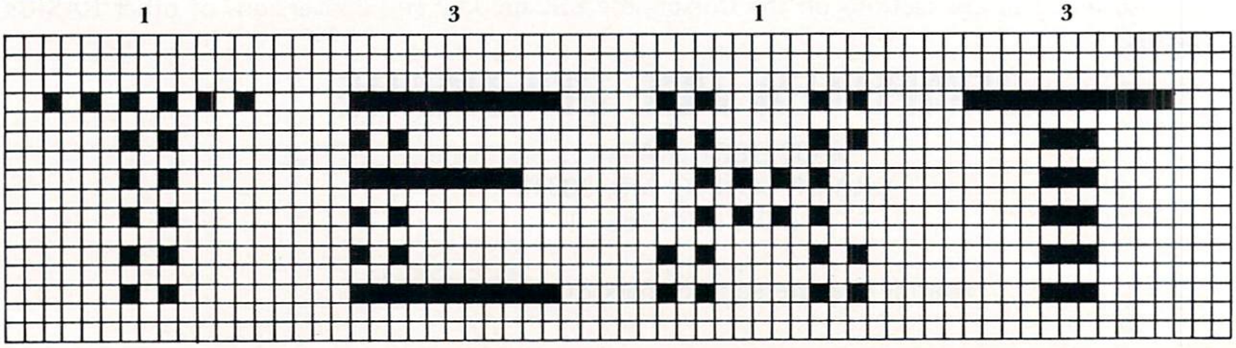
Figure 2.




```
10 REM SUPER SCREEN-TextPlot Demo
20 REM Use all the ATARI characters
30 REM including cursor up/down
40 REM left/right, backspace, RETURN,
50 REM etc.  Press CAPS/LOWR to
60 REM select upper or lower case,
70 REM as usual.  Atari Logo key
80 REM toggles reverse field.
90 REM Press console buttons for
       different colors
100 REM ESC switches modes (7 vs.8)
110 ML=1536
120 IF PEEK(ML)=0 THEN GOSUB 20000
130 XL=19:YL=11:DIM CHAR$(480),C$(480)
140 CHAR$(1)=" ":CHAR$(480)=" ":CHAR$(2)
=CHAR$:C$=CHAR$
```

Figure 1.

GRAPHICS SIX



```
150 GRAPHICS 7+G+16:OPEN #1,4,0,"K:"
155 IF G=1 THEN SETCOLOR 2,0,0:YL=23
160 LM=1:X=LM:Y=0:C=1
170 POS=X+Y*20+1:CHR=ASC(CHAR$(POS,POS))
:RVS=CHR:SC=ASC(C$(POS))-31
180 POKE 20,0:RVS=RVS+128:IF RVS>255 THE
N RVS=RVS-256
190 A=USR(ML,RVS,C,X,Y*8)
200 IF PEEK(764)<>255 THEN 300
201 T=PEEK(53279):IF T=6 THEN C=1
202 IF T=5 THEN C=2
203 IF T=3 THEN C=3
210 IF PEEK(20)<15 THEN 200
220 GOTO 180
300 A=USR(ML,CHR,SC,X,Y*8)
310 GET #1,KEY:E=0:DL=E
320 IF KEY>31 AND KEY<123 THEN 500
325 IF KEY=ASC("{CLEAR}") THEN CLOSE #1:
GOTO 140
326 IF KEY=ASC("{ESC}") THEN CLOSE #1:G=
1-G:GOTO 140
330 IF KEY=ASC("{UP}") THEN Y=Y-1:E=1
340 IF KEY=ASC("{DOWN}") THEN Y=Y+1:E=1
350 IF KEY=ASC("{LEFT}") THEN X=X-1:E=1
360 IF KEY=ASC("{RIGHT}") THEN X=X+1:E=1

380 IF KEY=155 THEN X=LM:Y=Y+1:E=1
390 IF KEY=ASC("{BACK S}") THEN X=X-1:KE
Y=32:DL=1
400 IF X<LM THEN X=XL:Y=Y-1
410 IF X>XL THEN X=LM:Y=Y+1
420 IF Y>YL THEN Y=0
430 IF Y<0 THEN Y=YL
440 IF E THEN 170
500 A=USR(ML,KEY,C,X,Y*8)
510 POS=X+Y*20+1:CHAR$(POS,POS)=CHR$(KEY
):C$(POS,POS)=CHR$(31+C)
520 IF DL=0 THEN X=X+1:IF X>XL THEN X=LM
:Y=Y+1:IF Y>YL THEN Y=0
530 GOTO 170
20000 ML=1536:FOR I=0 TO 252:READ A:POKE
 ML+I,A:NEXT I:RETURN
20010 DATA 104,240,10,201,4,240
20020 DATA 11,170,104,104,202,208
20030 DATA 251,169,253,76,164,246
20040 DATA 104,133,195,104,201,128
20050 DATA 144,4,41,127,198,195
20060 DATA 170,141,250,6,224,96
20070 DATA 176,15,169,64,224,32
20080 DATA 144,2,169,224,24,109
20090 DATA 250,6,141,250,6,104
20100 DATA 104,141,251,6,104,104
20110 DATA 141,252,6,14,252,6
20120 DATA 104,104,141,253,6,133
20130 DATA 186,166,87,169,10,224
20140 DATA 3,240,8,169,20,224
20150 DATA 5,240,2,169,40,133
20160 DATA 207,133,187,165,88,133
```

```
20170 DATA 203,165,89,133,204,32
20180 DATA 228,6,24,173,252,6
20190 DATA 101,203,133,203,144,2
20200 DATA 230,204,24,165,203,101
20210 DATA 212,133,203,165,204,101
20220 DATA 213,133,204,173,250,6
20230 DATA 133,187,169,8,133,186
20240 DATA 32,228,6,165,212,133
20250 DATA 205,173,244,2,101,213
20260 DATA 133,206,160,0,162,8
20270 DATA 169,0,133,208,133,209
20280 DATA 177,205,69,195,72,104
20290 DATA 10,72,144,8,24,173
20300 DATA 251,6,5,208,133,208
20310 DATA 224,1,240,8,6,208
20320 DATA 38,209,6,208,38,209
20330 DATA 202,208,228,104,152,72
20340 DATA 160,0,165,209,145,203
20350 DATA 200,165,208,145,203,104
20360 DATA 168,24,165,203,101,207
20370 DATA 133,203,144,2,230,204
20380 DATA 200,192,8,208,183,96
20390 DATA 169,0,133,212,162,8
20400 DATA 70,186,144,3,24,101
20410 DATA 187,106,102,212,202,208
20420 DATA 243,133,213,96,0,1
20430 DATA 28
```
©

Assembler Update

Eric Brandon

In my previous article, **Assembler in BASIC,** I presented a program that allowed you to enter, edit, and assemble machine language programs. One feature not included in the program was the ability to SAVE and LOAD your source code. I have since realized that this is a very useful capability to have, so I have written this short patch that gives the assembler SAVE and LOAD commands.

LOAD your old version, and type in the program lines you see below.

From now on, when in the menu, a 0 will allow you to LOAD a file from disk, and an S will allow you to SAVE a file to disk. The routine is written for DOS 1.0 disks. To use it with cassette type in:

```
11010 OPEN 8,1,0,FL
12010 OPEN 8,1,1,FL
```

Open last note — 8K users should not use this routine since their memory is quite full with the original program. If however, they can convince a friend to run it through the COMPACTOR program in **COMPUTE!** SEP/OCT 1980, p. 104, they may end up with more bytes free than when they started! ©

Maypole

Louis & Helen Markoya
Shelton, CN

Clyde Spencer's article on graphics modes 9, 10 and 11 **COMPUTE!** #14, pg. 120) is the first documentation I know of concerning these "mystery" modes. As the programs he wrote show, these modes offer great color versatility. For a very attention-getting display, add the following program to his *PALETTE* program with the subroutine TEN.

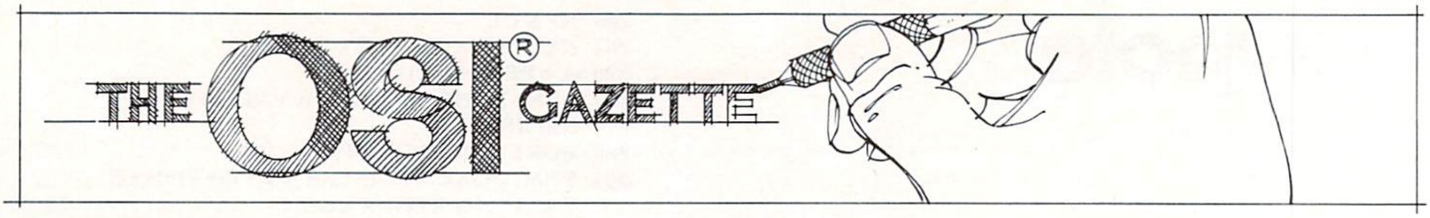
```
200 C=1
201 REM: START COLOR NUMBER
210 FOR I=1 TO 75 STEP 3
211 REM: DRAW TO THESE PLACES
220 COLOR C
221 REM: SETS COLOR VALUE FOR EACH LINE
230 PLOT 40,5
231 REM: TOP OF MAYPOLE
240 DR.I, 180
241 REM: MAYPOLE RIBBONS
250 C=C+1
251 REM INCREMENT COLOR NUMBER
260 NEXT I
261 REM: DRAWS NEXT RIBBON
270 A=PEEK (712)
271 REM: LOOK AT COLOR VALUE IN
    REGISTER 4
280 POKE 712, PEEK (710)
281 REM: POKE REGISTER 4 WITH LOWER
    VALUE IN REGISTER 2
290 POKE 710, PEEK (709)
291 REM: POKE REGISTER 2 WITH COLOR
    VALUE IN REGISTER 1
300 POKE 709, PEEK (708)
301 REM: POKE REGISTER 1 WITH COLOR
    VALUE IN REGISTER 0
310 POKE 708, A
311 REM POKE REGISTER 0 WITH ORIGINAL
    COLOR VALUE IN REGISTER 4
320 FOR I=1 TO 20:NEXT I
321 REM: GIVE TIME TO LOOK AT DISPLAY
330 GOTO 270
331 REM: ROTATE COLOR VALUES
    HIT BREAK TO STOP DISPLAY.
```

Color rotation is one effective way of creating animation easily. The above program not only rotates the Maypole's "ribbons," but also moves the color display from the top of the screen to the bottom. Enjoy and experiment: these bonus graphic modes certainly call for that. ©

Atari Program Listings

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

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



OSI Relocation Or What's NEW?

Mark Guzdial
Royal Oak, MI

After reading Elizabeth Deal's article on "Relocation of BASIC Programs on the PET" in **COMPUTE!** #13, I tried to implement the program relocation on my OSI C1P. After doing some POKEing around, both on and off-line, I found not only how very simple it was to do on an OSI machine, but also some observations about how the NEW command works.

Ms. Deal broke the relocation process down into four steps:

1. Set up one or more partitions
2. Adjust all BASIC pointers
3. Change the tape header information
4. Correct the forward pointers in the relocated program.

Since OSI machines store programs on tape in ASCII (not in the internal format) steps (4) and (5) are unnecessary. All one needs to relocate is to set up the partitions and to adjust the pointers.

In Williams and Dorner's *The First Book of OSI*, doing just that is described. In an example in that book, the authors show how a new workspace may be created by two (or three) POKEs and the NEW statement. One needs to adjust the pointer at $079, $007A (decimal 132,133), the beginning of BASIC workspace pointer, and to POKE in the initial null (Williams and Dorner do this with only two POKEs in the book, but normal use would probably entail three POKEs). So, if one needed to create a workspace at $1000, $0079 should be set to $01 (one byte past the initial null), $0071 should be set to $10, and a null should be inserted at $1000. After these POKEs, NEW should be typed to reset all other necessary pointers.

A normal LOAD should then load the program on tape into the new workspace, provided enough memory is left to hold it.

To return to a program LOADed into a new workspace, change the beginning of Basic workspace pointer with POKEs and type RUN.

What Else Was NEW Doing?

What I became interested in was the effect of the NEW upon the process. What other pointers were being reset? What else was NEW doing? Was it possible to relocate a BASIC workspace merely with POKEs?

I quickly found out that just leaving out the NEW only caused a machine crash. Taking a clue from Harvey Herman's article in the same issue of **COMPUTE!**, I tried modifying the single variable storage pointer, which is the same as the end of text pointer. In a normal cold start the end of text pointer ($007B, $007C) points to a memory address two bytes past whatever address $0079,7A points at. That is, after just cold-starting my C1P, $0079,7A points at $0301 (768) and $007B,7C points at $0303 (770). So, I tried setting my beginning of workspace to $1001 and my end of text pointer to $1003.

Well, it didn't crash — immediately. Obviously, there was something else the NEW statement did.

Checking around, I found that in a cold start, and in a NEW, addresses $0300 to $0302 are nulled out. So, I tried adding the POKEs to null $1001 and $1002 to the POKEs I had already.

Success! By POKEing the beginning of text pointer, the end of text pointer and nulling out the first three bytes of my new workspace, I could seemingly relocate my BASIC workspace without using the NEW command. But seven pokes seemed quite a bit more work than three pokes and a NEW statement, so I decided to see if any of the three nulls could be removed.

I found that the initial null could be left out to enter a program and to LIST it, but the RUN command generated a syntax error. If the null at $0302, or in the modified workspace $1002, was left out, the program would not LIST. But the null at $0301 or $1001 could be left out with no obvious problems.

$0301 and $0302 are pointers to the second line in the program in the 6502 format of least significant byte (LSB), most significant byte (MSB) for addresses. I knew that both being nulled indi-

cated the end of the program, i.e. there are no other lines to be pointed to. But the need for only one null at $0302 convinced me to start experimenting with nulls in the MSB of pointers later on in the program.

I found that regardless of what value the LSB contained, a null in the MSB of any line's pointer had the effect of deleting that line and all lines following.

For example, let's say that a multi-line program has been typed into an OSI C1P at the normal workspace of $0300, and the memory addresses $0301 and $0302 contain $0E and $03, respectively. This means that the second line of the program begins at address $030E, and the pointer to the third line is at ($030E,$030F). If we change $030F to a null ($00), we will not be able to LIST or RUN past the first line of the program. Everything from the second line on will be deleted. Except for changing the pointers to FREe the now unused memory, we have effectively NEWed most of our program. But, if we restored the value contained in $030F before it was nulled, we would also find our program restored, as long as we hadn't added any new lines of code which would overwrite the pseudo-NEWed code.

This means that during a RUN or LIST, BASIC does not check both the LSB and the MSB for nulls to indicate the end of the program-just the MSB. Actually, this makes sense since no real program line would exist in Page Zero memory (addresses that start with $00).

Though relocation without a NEW statement is still six pokes (four for the pointers and two for inserting nulls), it seems to me that knowing exactly what each POKE is doing is a lot safer than using such a dangerous command as NEW. Further, this information gives us a number of practical benefits such as the capability of a reversible NEW.

So these are the results of my excursion into OSI relocation. I hope users of both OSI and other BASIC systems may find my experiences to be helpful.

References

Deal, Elizabeth, "Relocation of Basic Programs on the PET", **COMPUTE!**, June 1981

Herman, Harvey B., "Memory Partition of Basic Workspace", **COMPUTE!**, Jan.–Feb. 1980 (original printing)

Williams and Dorner, *The First Book of OSI*, Aardvark Technical Services, 1980

©

Data Handling

Paul Muller
Ventura, CA

The OSI Superboard is an exceptional value for the computing power it provides. The 8K BASIC in ROM is ideal for writing number crunching programs which is my area of interest. After those first few weeks of wanton programming, however, I soon realized that an organized approach to data handling was needed. This article outlines a method I have used to structure the data handling chores for my numerical analysis programs. It is a series of BASIC routines which form a framework on which programs are built.

A completely flexible numerical program should be able to handle the following:

— Input data into the program from the keyboard
— Read data into the program off a tape
— Write data on to a tape
— Edit array data in the program
  List any part of the string or array
  Add elements to the string or array
  Delete elements
  Change elements
  Insert elements

Program 1 demonstrates these data handling routines. In this simple example, up to 50 numbers can be placed into two one-dimensional arrays, X and Y. The data can be examined, manipulated, stored, or retrieved from tape without leaving the program. To anyone who has written a statistical program, the advantages of these routines soon become evident.

The program consumes about 1800 bytes as written (not including the REMark statements). The amount of memory required will depend on the number and size of the arrays being used. The really determined Superboarder will undoubtedly find a way to squeeze the program into even less memory.

Here is a step-by-step look at the commands available in this program. If it sounds complicated, don't worry. The program takes all of five minutes to master.

INPUT: Calling the INPUT routine allows you to enter the data into arrays X and Y from the keyboard. Simply input the values for each element in the arrays per the program query. The element number of each input is shown also. To exit from the INPUT routine, simply enter END,END in response to the query.

Now that you have entered some data into the program, you will want to check it for accuracy. Calling up the EDIT routine provides the necessary commands.

LIST asks for the beginning and ending array element numbers for the data you want listed. The output shows the element number in the left hand column, and the data in X and Y residing in those elements.

To change an incorrect entry, enter the element number and the correct values of X and Y in response to the program query.

To delete data in X and Y, enter only the element number in response to the DELETE query. The same procedure is used for the insert command, but you must enter X and Y values as well.

The ADD routine picks up where the INPUT routine left off. To exit, enter END,END in response to the query. Note that terminating the ADD routine takes you out of the EDIT mode. The CNTL function in EDIT does the same thing.

To write the data on a cassette, call up the TAPE routine. Upon hitting the WRITE command, the program will ask you to enter a title pad. Simply repeat a suitable title and date until you get an overflow signal. The title pad will be 72 characters long, and will be written on the tape ahead of the data. This serves two purposes; to identify the data and keep recorder noise between data files from being picked up by the program. (The title pad is read into the program in the READ routine, but not used.)

After the title pad is entered, the program will instruct you to start the tape. When the recorder is running, enter any number and hit RETURN; a listing of the data will follow the title pad. When the end of the data in the two arrays is reached, the program will prompt you to stop the tape. All of the SAVE and LOAD commands are taken care of by the program.

The READ routine is similar. Since you are entering new data, exit the program using the STOP command. Using the Superboard's LOAD command, locate the data file in the tape, then bring up the program. The READ routine will prompt you to start the tape, which should be in the title pad of the data file. Enter any number (quickly!), and hit RETURN. The title pad and data will begin listing on your terminal. When END,END is encountered in the data file, the program will prompt you to stop the tape. You can then edit or add to the data using the other routines in the program.

The RUN command simply jumps to the starting line of the analytical portion of the program. In this example, the mean value of the X and Y arrays is calculated and displayed. The STOP command allows you to exit the program.

This program is intended only to demonstrate one technique of handling data. There is a lot of room here for more sophistication. You may want to modify this program to allow the reading of several tape files into the same array. Or maybe you want to read data into different arrays. You could set up your program such that it would take in one set of data and write out a different set to be picked up by another program for further analysis. You could thus chain programs together in modules and increase the power of your Superboard far above its memory size. And who knows, maybe your friends with their dual disk drives and mega-byte memories will actually envy your low-cost, customized "tape operating system!"

```
10 REM***DATA HANDLING PROGRAM***
20 REM***BY PAUL MULLER 8/81*****
30 REM***DIMENSION ARRAYS***
40 DIM X(50),Y(50)
50 REM***START CONTROL ROUTINE***
60 FOR J=1TO5:PRINT:NEXT
70 PRINT"INPUT(1),EDIT(2)":PRINT"TAPE(3)
   ,RUN(4)"
80 INPUT"OR STOP(5)";C
90 ON C GOTO 110,200,480,780,850:GOTO60
100 REM***START INPUT ROUTINE***
110 N=0
120 REM***START ADD ROUTINE***
130 FOR I=N+1TO50
140 PRINT"X";I;",Y";I:INPUTX$,Y$
150 IFX$="END"THEN60
160 X(I)=VAL(X$):Y(I)=VAL(Y$)
170 N=N+1
180 NEXTI
190 REM***START EDIT ROUTINE***
200 PRINT:PRINT:PRINT
210 PRINT"LIST(1),ADD(2)";PRINT"DELETE(3)
    ,INSERT(4)"
220 INPUT"CHANGE(5),CNTL(6)";C
230 ONCGOTO250,130,300,360,430,60
240 REM***ARRAY LIST ROUTINE***
250 PRINT":INPUT"ENTER I1,I2";I1,I2
260 FORI=I1TOI2
270 PRINT" ";I;X(I);Y(I)
280 NEXTI:GOTO200
290 REM***DELETE ROUTINE***
300 PRINT:PRINT"ENTER I FOR"
310 INPUT"DELETED X, Y";I
320 FORG=ITON
330 X(G)=X(G+1):Y(G)=Y(G+1)
340 NEXT G: N=N-1: GOTO200
350 REM***INSERT ROUTINE***
360 PRINT:PRINT"ENTER I,X,Y":INPUTI3,I4,I
    5
370 FORG=I3TON+1
380 I6=X(G):I7=Y(G)
390 X(G)=I4:Y(G)=I5
400 I4=I6:I5=I7
410 NEXT G:N=N+1:GOTO200
420 REM***CHANGE ROUTINE***
430 PRINT:PRINT"ENTER I,X,Y"
440 INPUT I4,I5,I6
450 X(I4)=I5:Y(I4)=I6
460 GOTO200
470 REM***TAPE ROUTINE***
480 PRINT:PRINT"READ(1),"
490 INPUT"WRITE(2),CNTL(3)";C
500 ON C GOTO 640,520,60
510 REM***WRITE ROUTINE***
520 A=PEEK(15):POKE 15,72
530 PRINT"ENTER TITLE PAD":INPUT P$
540 INPUT"START RECORD";C:SAVE
550 PRINTP$
560 P$="-"
570 FORI=1TON
580 PRINTP$;",";X(I);",";Y(I)
590 NEXT I:PRINTP$;",END,END"
600 POKE 517,0
610 POKE 15,A:PRINT"STOP TAPE"
620 GOTO 480
630 REM***READ ROUTINE***
640 A=PEEK(15):POKE15,72
650 PRINT:INPUT"START PLAY";C
660 N=0:LOAD
670 INPUT P$
680 FORI=N+1 TO 50
690 INPUT P$,X$,Y$
700 IFX$="END"THEN 740
710 X(I)=VAL(X$):Y(I)=VAL(Y$)
720 N=N+1
730 NEXT I
740 POKE 515,0
750 PRINT"STOP TAPE":POKE 15,A
760 GOTO 60
770 REM***START ANALYTICAL PART OF PROGRA
    M***
780 Z=0:Z2=0
790 FORI=1TON
800 Z=Z+X(I):Z2=Z2+Y(I)
810 NEXT I
820 PRINT:PRINT"MEAN X=";Z/N
830 PRINT"MEAN Y=";Z2/N
840 GOTO 60
850 END
```

©

Keyboard Conversion Program For The OSI C1P

Ronald C. Whitaker
Salt Lake City, UT

One of the features advertised by OSI for the C1P and Superboard microcomputers is a lower case character set. While this feature is present, trying to make use of it can be both confusing and frustrating. When the shift lock is depressed, the alphabet is decoded as upper case characters and the upper row of keys produces numbers. The symbols on the upper row of keys are produced when either shift key is depressed. This works well when writing BASIC programs since lower case characters are not recognized by the BASIC interpreter except when used in PRINT statements. When the shift lock is released, the alphabet is decoded as lower case and the upper row of keys is decoded as garbage. The numbers can be obtained by depressing the left shift key. This also changes the alphabet to upper case. Depressing the right shift key produces only garbage. There's probably a reason somewhere for this bizarre decoding pattern, but it certainly seems illogical.

This problem with keyboard decoding didn't make too much difference to me until I tried to write a program to teach touch typing skills to my kids. Then it became very apparent that the keyboard must decode as much like a standard typewriter keyboard as possible. This means that, when the shift lock is depressed, the alphabet must decode as uppercase and the upper row of keys must produce the symbols. With the shift lock released, the alphabet will decode as lower case, and the upper row will produce numbers. Finally, both right and left shift keys must decode the same and produce upper case alphabet characters and symbols from the upper row of keys. To do this, I wrote a short machine language program to change the decoding.

This program makes use of the fact that, when a subroutine call is made to $F000, the subroutine polls the keyboard and the ASCII code of the next key pressed is placed in memory location 531. The keyboard conversion routine then examines this value and, if necessary, converts it to a standard typewriter character and places it back in memory location 531. The converted value is then PEEKed from memory by the BASIC program and POKEd to the screen. Below is a BASIC program which demonstrates the use of this subroutine. It works as follows:

Line 5 specifies the end of RAM available to BASIC

Lines 10–30 POKE the subroutine into the protected area

Line 40 clears the screen

"...it became very apparent that the keyboard must decode as much like a standard typewriter keyboard as possible."

Lines 100–115 set up the beginning of the line to be written

Line 120 loads zero page addresses 11 and 12 (USR vector) with the starting address of the machine language subroutine and calls the subroutine via the USR function.

Line 130 provides a line feed/carriage return function whenever the RETURN key is pressed.

Line 140 provides a true backspace whenever the RUBOUT key is pressed

Line 150 erases the page and starts the program at the top of the screen

It is necessary to protect the area at the top of RAM where the machine language subroutine is stored so that the BASIC program will not write over the top of it. The REMARK lines at the beginning of the program give values to use in lines 5 and 10 when other than 8K systems are used. I intentionally avoided the "free memory" from $0222 to $02FF because of its growing popularity with almost everyone who writes short machine code programs. This popularity has led to an ever increasing number of conflicts between otherwise useful programs.

If I were going to improve the program further I would do two additional things: first, I would write another machine language program to erase the page instead of using the PRINT/scroll erase that I have included here. Second, I would write another machine language program that would scroll the screen upward when the cursor reached the bottom of the page. However, these additions are beyond the scope of the rather simple demonstration program listed here and are left for the reader to implement.

The Keyboard Conversion program as described here suggests additional uses for the OSI C1P microcomputer which would be difficult and impractical with the standard keyboard decoding provided by OSI. Two that readily come to mind are typing tutor programs and character oriented word processors or text editors. The conversion subroutine has been an interesting exercise in machine language programming and use of the USR function. Hopefully, the reader will find it useful and I would be interested to hear of any uses found for it.

KEYBOARD CONVERSION PROGRAM

```
5 POKE 133,179:POKE 134,31
6 REM..FOR 16K SYSTEMS CHANGE LINE 5 TO
  "POKE 133,179:POKE 134,63"
7 REM..FOR 24K SYSTEMS CHANGE LINE 5 TO
  "POKE 133,179:POKE 134,95"
8 REM..FOR 32K SYSTEMS CHANGE LINE 5 TO
  "POKE 133,179:POKE 134,127"
10 FOR X=8116 TO 8192:READ Y:POKE X,Y:NEXT
11 REM..FOR 16K SYSTEMS CHANGE LINE 10 TO
   "FOR X=16308 TO 16384:"
12 REM..FOR 24K SYSTEMS CHANGE LINE 10 TO
   "FOR X=24500 TO 24576:"
13 REM..FOR 32K SYSTEMS CHANGE LINE 10 TO
   "FOR X=32692 TO 32768:"
15 DATA 32,0,253,173,0,223,201,255,240,34,201,253,
   240,50,173,19,2,201
20 DATA 47,208,6,169,63,141,19,2,96,201,59,144,1,
   96,201,48,176,1,96,56
25 DATA 233,16,141,19,2,96,173,19,2,201,92,144,1,
   96,201,45,176,1,96,56
30 DATA 233,32,141,19,2,96,173,19,2,201,107,144,1,
   96,201,81,176,217,96
40 FOR N=1 TO 30:PRINT:NEXT
100 PL=53411:S=0
110 IFPL>54171 THEN 40
115 POKE PL+S,128
120 POKE11,180:POKE12,31:X=USR(X):C=
    PEEK(531)
130 IFC=13THENPOKEPL+S,32:S=0:PL=PL+64:
    GOTO110
140 IFC=127ORC=159THENPOKEPL+S,32:S=S-1
    :POKEPL+S,128:GOTO110
150 IFC=10ORC=42THEN40
160 POKEPL+S,C
170 S=S+1:IFS>23THENS=S-1
180 GOTO110
```

©

THE PET® GAZETTE

Editor's Note: In this candid interview, **COMPUTE!***'s Assistant Editor, Richard Mansfield, and Commodore's new US Marketing Director, Kit Spencer, explore Kit's background and goals for Commodore US. During Kit's tenure with Commodore in England and Europe, Commodore reached a point where they held the lead in market share in every country where Kit held marketing responsibility. Many of you Commodore "old-timers" may remember those times as years of promised, but undelivered, increased support in the US. We're curious to see the impact of this and other reorientations of personnel and goals at Commodore US. At the age of 35, with seven years experience at Commodore, Kit himself qualifies as an "old-timer." We welcome him to the US. — RCL*

COMPUTE! Interview:

Kit Spencer – CBM's New US Marketing Director

COMPUTE!: *What was your background in Europe? What sort of things have you done in your life?*

KS: Where would you like to start — with my life in Commodore?

COMPUTE!: *Just a little bit earlier than that.*

KS: Should I take you through a very quick pointed history of Kit Spencer?

COMPUTE!: *Okay, that would be great.*

KS: I'm a physics graduate by original background, but I'm the worst physicist in the business. After leaving university, I worked in the Peace Corps for a year. In Uganda. Then I spent two years with a company called Bowmar which is in the packaging business, if you like. Largely in the marketing sales area with them. I then spent four years with the Phillips Corporation. Primarily in their hi-fi and TV division.

COMPUTE!: *Was this all in London?*

KS: This was, ah well, Cambridge which is about 80 miles north of London. But, basically, it was across the UK that I was concerned with. I used to work in — I was in their marketing group there. A very quick and sort of pointed history, this goes back, that must be 10 years now. We had a lovely boom time in color television. It's wonderful. We could sell every one we could make. But we said the market was going to turn down in a year's time. Let's look at what market we should get into — was there anything we could do? And one of the things that I was looking at was the calculator market at the time. They'd come down, I think, to under

$200. Well, we said we sell electronic products. We sell products around two hundred plus dollars. And I started looking around at the marketplace to see if it was right for us at the time. And I got rather excited at what I saw then. I saw, in that class of companies, three generations — those that grew up on basic electricity — cookers, freezers, lightbulbs, heating appliances, that sort of product. What we call the "white" goods, it was the massive generation of companies which grew out of that — probably General Electric over here, for example. It was the second generation that grew up out of the *valve* which was hi-fi, television, all those sort of things. Maybe RCA or something like that is an example here.

COMPUTE!: *Transistor.*

KS: Actually, the transistor was in between.

COMPUTE!: *You're thinking of the vacuum tube.*

KS: The vacuum tube?

COMPUTE!: *I'm trying to translate this for our audience here.*

KS: The vacuum tube.

COMPUTE!: *Right.*

KS: Which you know you used to have them in radios, you had them in hi-fi's, and in gramaphones. It led to a whole massive industry of recording. What we, in Europe, call the "brown goods" industry — televisions, hi-fi, radios. And most of the people around me, in fact, would say it was better for that business. At least it was a very exciting business when it grew up around wartime, in fact. And I said 'Hey there is something happening here coming out of semiconductors and display technologies.' And you could see a few products. You could see digital watches and they were around $450 for the Pulsar, if you remember that.

COMPUTE!: *That would be back about what year would you say?*

KS: Oh, this has got to be back maybe 10 years — 8 or 9 years. I came to the conclusion there is a whole new generation of companies — a whole new industry coming. And I remember thinking — I felt that within two years the price of calculators could drop to under $25. I was wrong, it went to $10. And I thought I was being very bullish on that. And, actually, I decided I wanted to be a part of the industry. And after about four years there, I became the UK marketing manager for Bowmar where Commodore was my biggest competitor for two years. And it was as if Bowmar in the U.S. didn't have a very happy time. Unfortunately, the UK company was the brand leader dividing the UK when Bowmar went into Chapter Eleven over here ... I got approached to join Commodore, at that time and I joined Commodore to run their UK market operation. That would be seven years ago. Since that time (running through my background with Commodore) I ran the UK marketing operation. It involves the marketing companies in Europe, which is extensive. I'm then general manager of the systems division for the one which sent out the PET computer from the launching pad. I spent the last six months as European marketing director. I moved across here about four weeks ago now.

"...who's to say that in the next few years liquid crystals won't start replacing the vacuum, the cathode ray tube?"

COMPUTE!: *You know, you said something that interested me about the fact that you saw the three waves coming through. The electric, which is what they call "white goods."*

KS: Yes.

COMPUTE!: *The valve, (which I'm going to have to put in parenthesis as the vacuum tube) and then the third one. You called that one "brown goods." So what would be this third category? Microprocessors?*

KS: I guess so. "Silicon goods" almost could be said.

COMPUTE! *Okay.*

KS: Comes out of chips and displays.

COMPUTE!: *Displays? What do you mean by that?*

KS: The liquid crystal displays, LEDs...

COMPUTE!: *Do you see those as being the central aspect?*

KS: Not central, but a definitely associated area. I mean, if you look at the total industry which includes things like calculators in the marketplace and TV games and, ah, who's to say that in the next few years liquid crystals won't start replacing the vacuum, the cathode ray tube?

COMPUTE!: *Yes, that was what I was thinking, maybe...*

KS: And ... oh I'm sorry ... it was like seven years ago, I remember, I drew up a list of companies I thought would be leaders in the business into three categories. The first category I thought would make it; those that did a good job; or those that were unlikely to. Commodore was in the last category at that time.

COMPUTE!: *Well, that's an irony.*

KS: I've had to eat my words.

COMPUTE!: *Do you have anything to which you attribute Commodore's success in particular?*

KS: I think there are several things. Commodore is

a very misunderstood company. It is looked upon in many people's eyes here as a U.S. company and, in many ways, as a newish company, where, in fact, we have actually been going 23 years. And the majority of our business turnover is outside the USA. We are, in that sense, sort of a diversifying company. We are, as you probably know, the brand leader in every country in Europe. And, in fact, in Canada here, which is right next door to the States (*pause*) ... I'm sorry, I'm afraid I was interrupted ...

COMPUTE!: *You were describing what would characterize Commodore's success ...*

KS: One thing that our products have always been known for is value for money. And I've known Commodore now, both as having worked for Commodore and being a competitor of it for a few years before, and always our products have been value for money. They've always been at the leading edge of what's happening. I heard somebody: 'Oh Commodore is lucky. It brought the PET out. It got into the market at the right time.' As you know, it wasn't luck. We've been doing that consistently for the ten years I've been at Commodore. Whether it was a scientific calculator or whether it was a computer and, you know, hopefully we're still there doing it today. Be it the VIC, or whatever.

COMPUTE!: *Is there any significant difference in terms of Commodore's advertising? Do they advertise extensively in Europe and England as opposed to what they do here?*

KS: I think possibly it had a more consistent business approach in Europe than here. This has been, relatively speaking now, an oddly successful market so far. Obviously, I'm hoping that will change.

COMPUTE!: *Well, do you attribute that difference to anything that you can put your finger on?*

KS: A combination of things. One of which was a decision by the corporation. It's saying: 'How much resources have you got in the company? Where do you put them?' If one goes back to the beginning of the microcomputer marketplace, I think that we were all learning, we were all short of resources. Whoever you name in the business. It's so tough getting production, getting finance, everything else going in the business three or four years ago, that some people just put all their eggs into one basket here. We, if you like, set about perhaps a different strategy. We set about continuing to broaden the product range and we put a fair bit of energy into that. If you look at our range now, we've got the VIC, which I think is a very interesting product down on the true personal end. We've sort of taken the original 8K PET, developed what would the personal man want — he'd want color, sound, low cost, and expandability. And that's the VIC. We've said 'What's the difference between them?'

[A buyer at the other end] wanted a desirable keyboard, he wanted more memory backup, he wanted software, etc. and that's really the 8000 series. So we developed up the VIC range, the 4000 series, the 8000 and, now coming out, the SuperPET series. So now I think we have put together ... we've put a lot of our investments into getting a broad market range. And we've put our investments into a world-wide market as well. We did, you know, invest into developing up Germany, England, France, etc., which probably meant that we spent less of our corporate resources — both human and others — in the U.S., compared to some of our competitors. What we did start, just about a year ago, was the beginning of our strategy to increase our presence here in the USA. The heart of that, very much, was the regionalization policy — start to get closer to the dealers, etc. and we've started. About a year now I think it has been going on up there. And we really haven't got that in place. We've got the products in place and we've got the financing in place. We're beginning to look at now the stepping up of our marketing activities here to capitalize on the other factors.

COMPUTE! *Well, I'm sure you're aware that Commodore's developed a reputation, in the short amount of time that it's been selling computers in America, of having, as you point out, a strong price-performance ratio, but a woefully weak support system. And there have been a series of announcements of new distribution centers, new dealer set-ups, new hot-lines. There's a long list of more or less reorganizational announcements. I take it there's going to be another announcement.*

My question would be: 'Why would the dealers and the loyal Commodore supporters be likely, this time, to expect that the kind of full-fledged support that they can get — let's say with the Apple — would be forthcoming from Commodore?'

KS: I'm not going to promise a mountain. I'd promise achieved improvement. And I'd think if you'd look at some of the things: the regions have shown a considerable amount of progress. Speaking to a lot of the dealers here, we seem to have shown a fair amount of progress after the last four months. Some things have worked, some haven't. That's par for the business. But I think, increasingly, we're continuing to move in one direction. I think that occasionally we have over-promised and that maybe, for the best intentions, that's something we've got to be careful of. I think we've demonstrated our fair bid over here the last four months. I think we will continue to demonstrate that by the various things we are doing.

My job here is to build up, and support any operations to build up, our marketing policies. I'm not going to tell you that it's going to transform overnight. I hope to continue and accelerate what we've been doing for, really, the past 12 months, I think. And, hopefully, continue doing some of the things that we've shown we can do, elsewhere in the world.

COMPUTE!: *I suppose that's the curious thing. That there's this strange disparity between — I mean even in a demographic sense — the Canadian and American markets can't be all that dissimilar, and yet there's this strange difference in them in the way the marketing took place. Compare, for example, the Commodore in-house magazines. The* Transactor *in Canada and the American magazine. There is an entirely different reality to those two things. I guess, as you say, there is really no single cause. But, all the way down the line — advertising, dealers, support — everything seems to be quite distinctly different as soon as you cross the border up there. I imagine you are more or less addressing yourself to trying to bring America in line with the level of success, anyway, that Commodore experienced overseas?*

KS: Oh, yes. I would be unrealistic to say that we are going to take the number one position in the US overnight.

COMPUTE!: *But that's your goal?*

KS: No, my goal is to increase our position in the marketplace here. I don't think that we're going to be number one in the next six months. That would be unrealistic. I think the danger is if we often said we were going to do something which is promising too much. I very much expect to take a better, larger share of the marketplace. How large that is, we'll have to see. We've got a good product range. I believe that actually we are in a relatively good position to start building. We *have* got a good product range. I think that some of our competitors may be under greater attack when this competition comes in. Our product range does have a very good price-performance out there so we do have a position in that sense. I think that we have a big user base out there going for us which anybody new coming to business these days doesn't have going for them. There are a lot of PET users. A lot of things like **COMPUTE!** magazine, which write about our product, lend to our support. I think turning ... using our strength out there is going to be important.

COMPUTE!: *Well, that leads me to a somewhat ... actually I would like to revert to something I was thinking I might want to ask you about one of the first things you said. You've gone to university in physics and you didn't like physics? Or didn't like your work...*

KS: I did get my honors degree, but I didn't want to spend my life being a physicist.

COMPUTE!: *Right. And you found the transition to marketing, specifically marketing products having to do with electronics, a relatively easy one?*

KS: Oh, I think so, yes ... my personality and that fits in that area. I've been in the marketing of electronic durables now for about 12 years.

COMPUTE!: *I guess one thing that the readers would be interested in knowing is anything you have in the way of a forecast for what the next year, five years, or ten years is going to be like, for computing, at any level? Can you do that in a sentence?*

KS: In a sentence?

COMPUTE!: *No, no. I'm just kidding.*

KS: (Laughs) It's going to go in many different directions. I think it is a fallacy to say we're in one marketplace. I think that we have to recognize that we are in a variety of marketplaces. That there are the business marketplaces, the educational marketplaces, there are the personal marketplaces. In virtually every ... the instrumentation market places, the PET is very strong in that area as you probably know. There is undoubtedly going to be a large increase in the usage of computers. There will be a large increase in the awareness of them. I think different manufacturers will begin to find different niches in the marketplaces. I believe that it will require traditional industry goals, but I believe it will require other skills, as well, to be successful in business. I think, no question, that value for money will continue to increase in this marketplace. That's been the market ever since I've known it for ten years.

"...my goal is to increase our position in the marketplace here. I don't think we're going to be number one in the next six months. That would be unrealistic."

The price-performance ratio will continue to increase. I think the emphasis in some areas will begin to shift from technology to marketing. I think that we've almost gone through the very first era of the rush of enthusiasm of the products' technicalities. And people are now beginning to see them as tools to be used; and, therefore, the types of dealers, the types of users, the types of software packages in industry will begin to change. The people, in my experience, now are less asking how we did this, how we buy etc., as much as what it will do for me. And that we're going to see distribution broadening and the use broadening.

COMPUTE!: *Do you see, for example, that a computer will be in as many homes as, say, televisions, at some point? Or will, in fact,* become *a television?*

KS: I think, ultimately, the telephones, the televisions, all the technologies are now converging. That, yes, there will be some conversion certainly, but it will take a number of years. It will take...it could be five years before it happens. I think the television is becoming interactive with the TV games, with computers. I think that there is a gen-

eration of people growing up computer literate. I would say that within five years that the business environment the computer will become...will be looked upon like a copy of a typewriter. It is just a business tool. It's not the hallowed sanctuary of the data processor department. It's due to come. And, in the home, it's going to become another electronic tool like the TV, like the HiFi. I think there are interesting areas that will develop through the years in education and in things like computer-age construction. And, potentially, it can be a very interesting area.

COMPUTE!: *Do you foresee any danger that what's happened to most other American industries — such as the automobile, television, and appliance industries — that the Japanese will eventually get into the computer industry?*

KS: I undoubtedly believe we're in international business; that you cannot compete on just a one country area. The Japanese, I have the greatest respect for in electronics. They may find this market harder to get into than the others because of the software. I also see this competition in Europe. I think the companies, if they truly look just at America, will find themselves under increasing pressure. I think that, as myself and as a company, it's one of the reasons why we deliberately have tried to act internationally in terms of distribution and development.

For instance, we have a whole host of software in British, German, French, Spanish. You name it. And I would say, yes, I would think the share of the purely American manufacturers decrease, but will remain the dominant force. And, if you want to choose to include the American manufacturer, it's probably a good question because we have factories in Japan and in Germany. So I mean, in our case, part of our production is here in the States.

COMPUTE!: *We've spoken mostly about hardware. I'd like to ask you just a couple of questions about software. Do you feel that — well, this is actually a question about hardware — do you feel that the peripherals are mature? By that I mean, do you feel that the disk drive is essentially always going to be the disk drive it is today?*

KS: No, if you want one statement about the future from me, there's going to be even more change in the future than there has been in the past. It is, by no means, a static industry in terms of development.

COMPUTE!: *So you feel that there will be a product that will more or less replace the disk drive?*

KS: I believe, in time...I don't believe that anything will be obsolete overnight. But I believe, in time, we will continue to have changes. The greatest technical developments are still going on out there. Our biggest problem is keeping up with the marketing front.

COMPUTE!: *How do you feel in general about the future of software?*

KS: I think it will increasingly become more user-friendly. That's where the biggest breakthrough has got to come. I see the most important areas are use and assimilation by the person who is going to handle it. The days of the spoke software are gone in this industry. You'll have not just pre-packaged software — you have a lot of that. And also the quality has to continue to increase. But you have tools to develop the software much quicker and, increasingly, ones that can be done by the user. Products probably like OZZ, the Manager, things like that, are probably sort of the halfway stage between a purely pre-packaged product and a spoke product.

COMPUTE!: *One last question (unless you have something you think of that I haven't asked). What are the plans in the other direction, on the other side of things from the VIC? What are the plans for the SuperPET?*

KS: I'd say it's a very good market here. I think it's next in addition to our product range. I think it's going to be particularly suitable for areas in education where we're already extremely strong, higher education. But I think it's going to increase when they move us into some of the more traditional areas of business. I want to say that, so far, the micro has not gone into the traditional data processing. Small businessmen, inside the big company, the department manager who couldn't get time on the mainframe put in a micro and did word processing or put up financial planning or kept his own mailing list in his own office. I think that the SuperPET potentially bridges his gap.

I think the traditional computer industry has recognized the micro at last. Maybe IBM's entry signifies that. If you can't beat them, join them. I'm beginning to see rather traditional computer companies not having such a good time as they were. Rather traditional software houses were beginning to find it tougher. When the micro first came out, they could be poo-pooed a little bit. People said where there was no software, there was no service. All those were things that would come in time, but the potential for the machine was there. Now it's there.

I think the SuperPET is going to start appealing to some of the software audience out there — the ability to write in languages they know. That the DP department is beginning to look at micros. They are beginning to get involved with the specification of micros within their company at the department level and that they will. For the SuperPET offers them something very attractive: the ability to potentially transport software, the ability to use it potentially as an intelligent terminal. I think it begins to cross that gap between the traditional computer and the micro. And that could be a very important thing to us. ©

This is the second **COMPUTE!** *Overview. Believing that* **COMPUTE!** *readers deserve the most objective, comprehensive analysis possible, we select new products which are then sent out for Overview. We have established review panels, each consisting of three or more reviewers whose backgrounds qualify them to analyze new software or hardware products.*

To better prepare **COMPUTE!**'*s readers for their buying decisions, and to present fair reviews, we collect the independent opinions of the panelists into one Overview. Lawyers will test legal packages in the environment of a legal office, doctors will test medical systems, and this issue's Overview panel is made up of computing professionals who have each used Commodore computers since they were first introduced. We think that you will find* **COMPUTE!** *Overviews complete, balanced, and informative. — The Editors.*

COMPUTE! Overview: Power

It should be acknowledged at the start: the review panel was unanimous in concluding that POWER is an extraordinary product. It is a ROM chip which goes into the machine and adds several significant commands to the BASIC chips already inside. The key word here is *significant*. One panelist said that POWER made the PET "into an entirely new, much stronger computer." It is not inexpensive ($89.95), but the reviewers agreed that it was an excellent value. One commented, "I think that adding a disk drive or a printer are important expansions to a computer; adding POWER has some of that impact ... it greatly expands the programmer's capabilities."

POWER Commands

POWER was written by Brad Templeton; the manual, by Jim Butterfield. The documentation is written by a knowledgeable user, based on extensive experience (rather than written by a programmer as an afterthought). Butterfield uses his considerable knowledge of the PET system and POWER to make the manual clear and complete.

A major advantage which POWER possesses over a number of the chips available is that it performs in almost precisely the same way on 8032, 4032, and 3032 machines.

POWER consists of four thousand bytes of machine code which provide a great amount of help to the BASIC programmer. It solves many housekeeping problems. It also provides solutions to problems you didn't even know you had. Theoretically, POWER adds at least thirteen commands to PET's BASIC. The exact number is irrelevant in any comparison shopping, because several instructions permit adding your own commands, thus expanding the count well beyond the specified thirteen.

Some of the standard commands, also seen in similar products, are automatic line numbering, deleting all or portions of the BASIC text, and renumbering. The unique feature of this renumber command is that it can renumber not only the entire text, but also *any portion* of a program. Subsequent adding of extra program lines becomes a snap.

This is even more useful when employed in conjunction with POWER's disk or tape MERGE command. The latter is an appending routine which permits you to attach another program or a subroutine to a program already in memory. Once again, a powerful feature has been added: for complicated uses, one may, with little effort, fit subroutines into *any* place within an existing program. A machine code "dynamic keyboard" type of routine performs the job by reading an ASCII format subroutine created earlier. Program lines being appended from a tape or disk are included into the program in the same fashion as you would have typed them in from the keyboard.

Extraordinary Features

Power has "instant phrases." You may tell POWER that when you press shifted-A, for instance, what you really mean is this print loop:

```
FOR J = 1 TO 50: PRINT X(J): NEXT
```

It works like magic. Press shifted-A, the above line appears on the screen, you hit RETURN (after altering the line first, if you wish) and the values of the X array are displayed. Should you wish to use shifted-A for something else, you may redefine the key on the spot.

The unique, and very clever feature of this is that special REMs are used, at the start of the program, so that a BASIC program will continue to run even if POWER is not in place, and yet the phrases are available if required.

Both Instant Phrases and Instant Subroutines represent tremendous advances in capability. It would be ideal if Jim and Brad could be persuaded to produce some form of Advanced Users Supplement, or to update to the manual, dealing with this and the similar question of the more specialized aspects of using the XEC command to allow the disk unit, or some other peripheral to take over the machine in substitution for the keyboard. It is tantalizing to see simple examples. Why not some extremely clever ones too?
for the keyboard. It is tantalizing to see simple examples. Why not some extremely clever ones too?

This instant phrase feature of the keyboard is invaluable. Imagine never having to retype the monitor SAVE instruction. In this instance, assuming you are working within BASIC, you can define a key to mean S"prog",01,1234,5678. Since these single keystrokes stand for more than one keyword it could prove the most useful of the instant features.

On a smaller scale, most shifted keys have been defined to mean BASIC keywords, like OPEN, DATA, READ etc. (NEW has no key, a nice touch). You may change those default assignments, if you wish. Labels are provided for placing on the keys to help you remember which key means what. When you press a shifted-I, for example, the word INPUT appears on the screen. Shifted-L followed by RETURN lists the program.

The debugging potential of the POWER chip is enhanced by virtue of the fact that POWER features can be embodied in instant phrases. This has enabled the major design criterion which Brad Templeton had in mind, namely that program should be transportable and should not be POWER-dependent, to be implemented, whilst allowing the programmer full use of this attractive feature.

A major advantage which is available in the 80 column machine only is the command SEL I which gives the ability for an Instant Phrase to be accepted in response to an INPUT command. This is really great for debugging, because it implies, provided you hit the right key, that the data you input is identical every time.

And on a very large scale, keys can be defined to mean *whole sections* of a program. For example, Upgrade ROM may define a key to mean the well known disk error channel read routine. You may define a key to mean "execute the routine which begins at line 1000." You can convert numbers with it, draw with it, whatever you wish.

This feature is difficult to appreciate fully until you've used it, but it permits much flexibility. One of the panelists tested it by writing a "programming shell" which was LOADed into PET at the start of any session. This consisted of eleven lines (1-11) which held the REM statements which POWER uses to define keys for this feature. The subroutines themselves were placed in lines 50000+.

This easily added the following *single-keystroke* functions to BASIC. Clear screen and display the time for 10 seconds. Get the name and SAVE a program to disk. Get the name and LOAD a program from disk. Enable POWER. VERIFY a SAVE. Display Directory. List a program. Display instant-feature command keys. Delete lines 1-11 and 50000+ (removing the shell from whatever program was written). Search the program for a particular phrase, word, or variable. Translate between hex and decimal.

The panelist observed, "I wrote (as one of these instant subroutines) a short 'dynamic keyboard' routine to permit POWER's own commands to be used within subroutines. This allowed specified line deleting, searching, etc. Whatever you often need to do in BASIC programming you can assign to a shifted key and it's instantaneous. Machine language programmers can slip out of the monitor, press 'H' and enter a decimal number — the correct hex is immediately printed on the screen. Anything you want can be accomplished at a keystroke. It's like having a library of all your utility subroutines built into the computer. You can effortlessly add to BASIC; you can *personalize* your machine in a way you never imagined."

The "instant-key" feature, if not needed or if it interferes with something else you want to do, can be easily turned off.

XEC-Execute

Possibly the most extraordinary POWER command is XEC. One is reminded of COM files on a DEC computer. Sharp programmers may think of clever things to do with XEC that may not have yet occurred to the author. Merge is part of the XEC subset. XEC is a command that can turn PET into a powerful workhorse with exotic things like linking loaders. It is possible to set up an elegant system of program development aids or system execution requests consisting of a series of BASIC commands. These commands are placed on a tape or disk file and recalled by XEC. As the file is read, the statements are placed on the screen and executed on the spot, in the same fashion as if they were typed from the keyboard. This means, for instance, that if there is a repetitive sequence of events that you usually go through each day, you no longer need to type the commands. After buying POWER you can decide to set up such a system by typing the commands just once more and never again.

XEC (execute) is very powerful and totally unique. Because it is an extension of the technique pioneered by Brad Templeton for merging programs on cassette, there is a danger that it will be seen as merely a rather complicated merging method. This is totally wrong. The sequential file created is read by the computer and treated in every way as if the strings were being typed in at the machine. This implies that they will be acted upon as Direct Commands, and their capability is limited only by the fact that the special commands in POWER cannot be included. The manual gives a very tantalizing hint of writing programs which write programs.

POWER permits you to redefine the keyboard in almost any way you wish. For the technical and machine code people there are additional commands that can help, namely MLM and FIX. MLM simply makes a *call* to the monitor as opposed to a *break* obtained by SYS1024. This saves typing effort, crashing due to keybounce or bad typing, and in BASIC 4.0, guarantees that the CMD command remains enabled. FIX is a handy command that aligns BASIC pointers following a BASIC LOAD of machine code. Such LOADs often make a mess of the end of program pointer. You no longer have to worry about it. FIX finds and notes the actual end of the BASIC program. (FIX will not restore pointers after a NEW.)

POWER also provides find and replace functions. A program text can be searched for presence of a keyword, a character string, or a variable. It can be searched for occurrence of patterns or sequences of commands; for instance, a FOR-NEXT sequence. It can be searched for all occurrences of words containing a pattern without necessarily knowing what the rest of a word might be. POWER can replace a variable by a differently named variable and can change a character string to a different character string.

The ability to continue to SEARCH for a particular string, without having to define it again, merely by hitting @ is very convenient.

POWER includes a tracing command which permits you to watch the program executing at a slow pace. It is the best visual debugging show on the PET. POWER can display any or all of: line number, its statements, and the value of a last relevant variable. Three different speeds are permitted, including single step.

The versatility of the Trace command is enhanced by having special SYS commands with parameters enabling you to switch in and out specific types of Traces. This is particularly good in avoiding waste of paper when loops are running, and also makes it easy to bypass sections of the program which have already been successfully debugged. Complex programs can be debugged in stages using a GOSUB to a subroutine for the SYS commands switching the Trace on and off.

The manual shows how to avoid the problems of tracing GET commands. FIX will occasionally be very useful for resetting certain essential pointers which may have been clobbered by loading machine code programs into an unusual part of memory. The manual gives a very full explanation of how this command will prevent loss of variables or of the BASIC program.

A program stopped by a STOP key or by an error can be queried as to the reason, or more precisely, the location of the last executed place by use of the WHY command.

The variable dump command displays values of variables and functions and permits values to be changed on the screen, once again, an unusual feature. Array values are not displayed. Taken together, TRACE and WHY will also make BASIC easier to learn. There is no ambiguity any more about what is wrong, where it is wrong, or how it goes wrong during a RUN.

The chip adds more punch to the already excellent PET screen editor. Cursor keys and INST/DEL keys repeat at a very fast rate. Up and down cursor keys can also be used for scrolling the program. Yes, scrolling the program, WordPro fashion, in both directions. LIST need no longer be used. The scrolling motion is as smooth as PET usually provides in the up direction and perhaps only slightly choppy in the down direction on 40 column PETs, though this may be an illusion due to double BASIC lines on the 40 column screen, coming in backwards on top.

A useful feature of the repeating cursor key and instant key provisions (IRQ driven) is that they disengage when POWER takes notice of LOAD, SAVE and RUN commands. Consequently, the program runs fast and it is no longer necessary to resort to the usual RUN/STOP trick of restoring the IRQ vector prior to input and output. It is easy to re-enable the repeating key feature. Pressing RETURN or issuing any POWER command does the job. The entire chip can be disabled by OFF.

There are two items that might cause trouble for some people. First, a "delete all program" command exists and is written "DEL-". It is equivalent to NEW. Users of TOOLKIT™ might wipe out a program unintentionally since they are accustomed to putting in the line number or getting an error message if the number is missing. ("DEL" does give error message). Second, the Universal Wedge and the old DOS SUPPORT program conflict in the hookup to the CHRGET routine and are, therefore, disabled. Users of Basic 4.0 systems will not feel the loss of the Wedge program. Users of the Upgrade systems without convenient shorthand disk commands might miss it. However, the tradeoff seems obvious: POWER gives far more than the Wedge and, with POWER, you can *add* single-stroke disk commands to replace the Wedge anyway.

The Book

The documentation provided with POWER ought to be considered the standard against which future documentation is measured. It is correct, unambiguous, redundant, thorough, coherent, and fun to read. The redundancy greatly assists understanding. Most of the time, Butterfield shows a command and explains it in one or two sentences. Then, in case you didn't get it on the first pass, he rephrases the explanation to make it absolutely clear. Abundant examples follow each command. Where appropriate, he points out and explains syntax which is different from that used in similar products. Just about every possible, usual, or unusual use of a command, together with its pitfalls, if any, is described. A separate chapter is provided for a list of several unexpected events that might occur. One gets a feel for how the program functions by reading that list and by reading Jim's explanations and methods of preventing trouble and misuse.

Apart from several innocent typos, no errors were discovered. The book contains a table of contents, index, and top-of-the-page indicators of contents on that page, making it easy to find things. It is 74 pages long.

Reading the book, one not only quickly learns how to use POWER, but also gets to see it from the perspective of the whole PET. In the case of the XEC command, a hint of a warning is given to the effect that PET just might make us all obsolete by writing its own programs. Short programs are also provided. For instance, there exist two disk-related routines for users of the Upgrade ROM system, which can be attached to the instant action keyboard. Various instant keyboard illustrations are amusing and educational.

In an unusual twist in documentation practice, the book also includes a listing of the key locations in the chip, making it possible for machine code people to interface to POWER. A user may also add his own commands to expand POWER's power. It should be possible, for instance, to reinstate the WEDGE program back into existence, even though the interfacing addresses conflict.

Command Set Summary

AUTO (line #) (increment) — automatic line numbers as a program is being entered.

DEL (line range) — delete any range of lines (syntax similar to LIST).

DUM — provides a list of all program variables which can then be, if necessary, changed on the screen.

FIX — restores BASIC pointers.

MLM — a call entry to the monitor.

OFF — kills POWER.

REN (increment) (new starting number) (line range) renumbers all or a range of program line numbers.

SEL (R) or (K) or (P) — turns on or off a number of the "instant key" features.

TRC (L) or (N) or (T) or (NT) or (LT) — enters the various Tracing modes.

WHY — prints the line in the program where an error stopped the program (or the STOP key was pressed). It turns on reverse field over the error in the line.

XEC (file number) — turns control of the computer to a tape or disk file.

@(line range) — search.

] (replacement) (line range) — search and replace.

General Overview

•Panelist #1: "By way of summary: POWER is a unique program giving you many useful commands. It is easy to learn to use. It is easy to expand and customize. It is meant to make programming easy. It is 'friendly' in use. It works smoothly and without a glitch.

"Needless to say, Brad Templeton should be congratulated for an extremely difficult job done imaginatively and Jim Butterfield gets a little gold star for writing the most unique set of instructions. More power to both."

•Panelist #2: "The user will find this package very helpful when writing and debugging BASIC programs. Some of the commands are not as useful as others and some will take a little getting used to. On balance, however, this is an excellent value for the price."

•Panelist #3: "With the possible exception of my $400 word processor, I can think of no software product which matches this one: I will use it often every time I'm at the computer. I tried to balance my comments by looking for things to complain about. I can't find any.

"Usually I'm a curmudgeon about this sort of "helper" software. I never used the wedge because I found it more trouble to load than the time it saved. I avoided other CHRGET "aids" because they, too, seemed rarely worth using. POWER is the exception. It contains such a range of *necessary* programming tools (and permits the user to fashion his own tools so easily) that I would compare it to BASIC's PRINT command: after you use it you can't imagine programming without it.

"One final comment. In the large manual, Butterfield does for POWER what he's been doing for the PET in **COMPUTE!** for years. He lucidly teaches you the essentials and then takes you beyond, into new tricks and methods you'd likely never have thought of yourself. This whole package is, without qualification, a splendid addition to the computer."

•Panelist #4: "Power offers so many facilities, that the user, whether beginner or expert, can grow in capability by using it.

It is worth emphasizing the fact that this is no mere run-time improvement to extend BASIC, nor a mixture of such and programmer's aids, but rather a very carefully thought-out set of aids to the serious user of BASIC, designed by a truly expert programmer, to cram as much power as possible into a 4K chip, with as much care taken in choosing what to leave out, as what to include."

POWER. Professional Software Inc., 166 Crescent Road, Needham, MA 02194. $89.95. Versions available for the 2001 (graphics or business), 4000, or 8000 CBM computers. ©

The PET Speaks

Kenneth Finn
Bedford, NY

The following article gives the Commodore PET computer (all ROMs) the ability to reproduce speech from audio cassette tapes. Even though the sound quality is very limited, the principle will allow intelligible speech and the program is by no means optimized — much can be done with it.

This program was taken originally from an idea of Robert Bishop in *Microcomputing*. He wrote about the idea for the APPLE computer. His original program was 18 bytes long. In order to adapt the principle for the PET many changes had to be made. This is evidenced by the present size of 81 bytes. Another problem was that the PET has no built-in speaker like the APPLE so I had to substitute the "CB2 sound" convention which many present PET users have.

For those of you who have not heard about it, purchase a small audio amplifier such as the Radio Shack #277-1088 two hundred milliwatt solid state speaker-amplifier. This is then wired to pins N and M of the user port via a card edge connector. These pins are the CB2 line and ground. This setup can make musical sounds also using the internal shift register of the 6522 VIA. This article will not explain this techniques because it has been often described. This simple setup which has many uses.

Basically, the small program shown below does the following things:

1. It sets the interrupt to stop further interrupts.

2. It starts the motor on cassette #1.

3. It tests to see if the stop key is pressed.

4. It clears the 6520 Keyboard PIA and sets CA1 for a low-high transition.

5. It then waits for that transition. This means that the cassette has sensed some sound. This also represents a digitization of the analog signal from a voice tape. Also note here that the PET cassette electronics has a Schmidt trigger in it which also stops analog signals cold.

6. When it detects the shift, it sets CB2 on the 6522 high. Note that this is tricky since other bits in the 6522 PCR must be left alone.

7. It then sets up CA1 for a high-low transition and sets CB2 low when it occurs. It then returns to step #3 for another cycle.

8. Lastly, when the STOP key is pressed, it turns off the motor and returns via a break instruction.

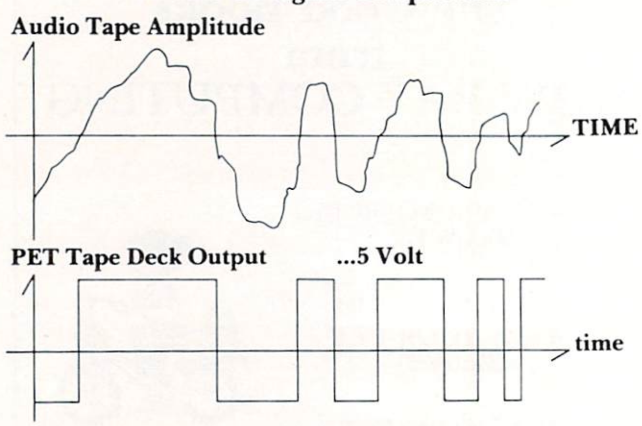
To use the program, first have it in the second cassette buffer. To enter the program, type SYS1024. Then type .M 033A 0390 (or, for 4.0, M 0360 03B0). Then enter the numbers right over the numbers on the screen, hitting RETURN after each line is changed. Then issue the monitor GO command (i.e. .G 033A) and you will hear some semblance of sound through the CB2 amplifier. The STOP key on the PET will get you out of the routine.

What is next for this routine? Well, speeding up the main loop will improve the sampling rate and, thus, the sound quality. Also, putting a simple low-pass filter between the PET and the amplifier will eliminate the switching noise. Other improvements such as adding a way for a BASIC program to time a message and then stop the routine would make it quite useful.

A big question is what to do with it? The answers are many. First it can be used to provide audio directions after a digital program load. Second it can become a speak and spell type of demonstrator. Third it can provide speech processing once limitations in the sampling rate/memory requirement are solved. (Hal Chamberlin where are you?)

I have just begun experimenting with this routine and I am looking forward to discovering both new uses and new modifications to it. The routine also gives many clues on how to use the cassette IO and the CB2 line which are useful all by themself. Happy Experimenting.

Picture Of Signal Comparisons



Program 1: Original and Upgrade PETs

```
.:  Ø33A 78 A9 35 8D 13 E8 A9 F9
.:  Ø342 8D 1Ø E8 A9 1Ø 2D 12 E8
.:  Ø34A DØ Ø3 4C 84 Ø3 AD 1Ø E8
.:  Ø352 A9 3E 8D 11 E8 2C 11 E8
.:  Ø35A 1Ø FB AD 1Ø E8 AD 4C E8
.:  Ø362 Ø9 EØ 8D 4C E8 AD 1Ø E8
.:  Ø36A A9 3C 8D 11 E8 2C 11 E8
.:  Ø372 1Ø FB AD 1Ø E8 AD 4C E8
.:  Ø37A 29 1F Ø9 CØ 8D 4C E8 4C
.:  Ø382 4Ø Ø3 A9 3D 8D 13 E8 58
.:  Ø38A ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
```

Program 2: 4.0 PETs

```
.
.:    0360 78 A9 35 8D 13 E8 A9 F9
.:    0368 4D 10 E8 A9 10 2D 12 E8
.:    0370 D0 03 4C AA 03 AD 10 E8
.:    0378 A9 3E 8D 11 E8 2C 11 E8
.:    0380 10 FB AD 10 E8 AD 4C E8
.:    0388 09 E0 8D 4C E8 AD 10 E8
.:    0390 A9 3C 8D 11 E8 2C 11 E8
.:    0398 10 FB AD 10 E8 AD 4C E8
.:    03A0 29 1F 09 C0 8D 4C E8 4C
.:    03A8 66 03 A9 3D 8D 13 E8 58
.:    03B0 00 00 00 00 00 00 00 00
.
```

Program 3: Source Code

033A	78			SEI		Set interrupt
933B	A9	35		LDA	#$35	
033D	8D	13	E8	STA	$ E813	Start motor for cassette #1
0340	A9	F9		LDA	#$F9	
0342	8D	10	E8	STA	$ E810	Test for STOP key pressed
0345	A9	10		LDA	#$10	
0347	2D	12	E8	AND	$ E812	
034A	D0	03		BNE	$034F	Branch if no key pressed
034C	4C	84	03	JMP	$ 0384	Go to End of program if key pressed
034F	AD	10	E8	LDA	$ E810	Clear 6520
0352	A9	3E		LDA	#$3E	
0354	8D	11	E8	STA	$ E811	Set CA1 for Low/High transition
0357	2C	11	E8	BIT	$ 036F	
035A	10	FB		BPL	$ 0357	Wait for transition
035C	AD	10	E8	LDA	$ E810	Clear 6520
035F	AD	4C	E8	LDA	$ E84C	
0362	09	E0		ORA	#$E8	Set CB2 HIGH
0364	8D	4C	E8	STA	$ E84C	
0367	AD	10	E8	LDA	$ E810	Clear 6520
036A	A9	3C		LDA	#$3C	
036C	8D	11	E8	STA	$ E811	Set CA1 for High/Low transition
036F	2C	11	E8	BIT	$ E811	
0372	10	FB		BPL	$ 036F	Wait for transition
0374	AD	10	E8	LDA	$ E810	Clear 6520
0377	AD	4C	E8	LDA	$ E84C	
037A	29	1F		AND	#$1F	Set CB2 LOW
037C	09	C0		ORA	#$C0	
037E	8D	4C	E8	STA	$ E84C	
0381	4C	40	03	JMP	$ 0340	Jump always for next transitions
0384	A9	3D		LDA	#$3D	
0386	8D	13	E8	STA	$ E813	Stop motor on Cassette #1
0389	58			CLI		Clear interrupt
038A	00			BRK		Return to monitor via break ©

Machine Language: Monitoring Progress

Jim Butterfield
Toronto, Canada

A Monitor is a simple thing. It's the built-in program that allows you to read and change memory. Yet beginners find that developing skills in using their monitor systems can be a major hangup in learning how to cope with machine language. It's partly mechanics, and partly insight.

First Functions

All monitors allow memory to be displayed and changed. The display is usually in hexadecimal, and beginners are urged to learn about hexadecimal numbering systems. The command varies from system to system. On the KIM, you press the AD button and enter the address; memory contents are immediately displayed, and the user can "walk through" memory using the "+" key to step to the next location. The SYM asks you to enter the address first and then press the MEM key to see the contents. The AIM displays memory locations four at a time, and screen-oriented systems usually allow any number of locations to be seen; in the PET it's multiples of eight which means that if you ask for ten addresses to be displayed you'll get sixteen.

There's always a command to change memory, although sometimes it doesn't look like one. On the KIM, it's a mode change: you switch to data entry mode by pressing the DA button and, from that point on, you're entering changes. On the PET, memory is changed by first displaying the contents on the screen and then moving the cursor back and typing over; it's similar to screen editing in BASIC. Remember to press RETURN to make the screen change permanent. Most other machines ask for a specific change-memory command to be typed — on the Apple, it's a colon, for example.

So we can read memory and change memory: that's indispensable if we want to get a program into the machine. The next question is how to run the program, and how to stop it after it's run.

STOP and GO — Traffic Control

The usual command to get a program to run is GO, often abbreviated by the letter G. GO does a number of fascinating things before it actually goes to the selected program; to explain them, we'll need to reverse the order and discuss STOP first.

When a program is Stopped, there's a lot of useful information inside the microprocessor chip. It's not in memory; it's in the chip where you can't get at it, and it will soon be destroyed by other data. It's vital to be able to inspect this data which is held in "registers" within the chip; so the monitor's STOP sequence arranges to dump these values to a fixed place in memory. Now you can look at them if you wish by inspecting memory.

Suppose we have Stopped a program and we want it to continue: that often happens during program testing. We issue the command for GO — but our program wouldn't run properly if the information in the registers were lost ... and the monitor itself has been using and changing those registers as it worked. What to do? The GO command, just before going to the desired location, will bring back the register values that were dumped into memory and restore the microprocessor exactly. Then it goes to your program. Clever? No — vital.

How do we Stop a program? There are several ways. The Break instruction in machine language (BRK — code 00) will almost invariably take you to the Monitor stop program. Some machines are equipped with a STOP key which interrupts the program. Advanced monitors allow the definition of Breakpoints — locations where the program will flip to the Monitor.

Registers

The Stop routine causes the contents of the microprocessor's registers to be dumped to memory. You may examine them there, and you also may change them. Don't become confused: these locations are not the registers themselves; they are just copies made at Stop time.

Many monitors allow registers to be inspected by means of a special command — R is popular. Once again, there is always a means of changing the register image; this is useful when you are checking out a program and want to make shortcuts or special tests.

There aren't a lot of registers in the 6502. There's the Program Counter, usually abbreviated PC, which tells you where the program was operating when it stopped. There are three data registers, A, X, and Y which are the only places in the microprocessor where you hold data. There's the Stack Pointer, which tells you where the temporary stack values are being kept: for example, if SP contains value EC, you should look in locations 01ED to 01FF for the values themselves. Finally, there's the Program Status Word, often abbreviated PSW or SR for Status Register. This last contains several flags; it's often useful to be able to break the hexadecimal value to binary so as to see individual flags. For example, if I see a value of B9 I know immediately that the Carry flag is set. Can you see how? To work it out, you'll need to know that the code for the flags is NVXBDIZC. Don't try to pronounce it — use it to decode the bits.

The PET monitor has an oddity: among the registers it shows a value called IRQ. This isn't a register at all; it's a vector located in memory (at hex 90 and 91, if you must know).

Saving And Loading

On many systems, the only means of saving memory to tape or disk and loading it back to memory are provided by the Monitor. When Saving memory you must give the start and end addresses of memory. Know your system: many Monitors require that the end address be one beyond the last location saved. When loading, the Monitor usually knows where the data came from and puts it back; some Monitors allow loading to a different location.

Some Monitors don't write a program name to tape; on others you must supply the name. When a Monitor is capable of saving to more than one device (e.g., either of two tapes or disk) you'll have to furnish information on where you want your data saved.

Quitting

Many systems are Monitor-oriented. They started as tiny systems with no language but the Monitor and, when extra features were added, the Monitor still was dominant. In this case, you must leave the Monitor with a GO command to enter another system. To return to BASIC on an Apple II system, for example, you're asked to type in the curious command 3D0G. This has nothing to do with man's best friend, but is a command to GO to location hexadecimal 3D0 where BASIC takes over.

The PET considers BASIC as standard and the Monitor as an "extra;" the Monitor returns to BASIC with the simple command: X (exit). Other systems make provision in the Monitor for invoking other systems — the AIM, for example, has special control keys for its text editor, its assembler, and BASIC.

Extras

Most monitors can be extended; some have extra features built in. Some of the features you may find in your system, or may wish to add with an "extension" package or to write yourself include: Breakpoints, single-stepping, slow step, disassemblers, tiny assemblers, memory search, memory fill, tape check, memory block move.

It's quite a luxury to have these extra features. Keep in mind that they're icing on the cake ... you can work well, if not quite as comfortably, with just a basic Monitor. And remember that extra features take up extra memory. It doesn't help to have an 18K super Monitor when you have an 8K machine...©

Directory For 3.0

Stephen Meirowsky
Peabody, KN

A very important part of any disk system is the ability to get the directory of your disks at any time. Commodore has provided a very nice DIRECTORY command for 4.0 BASIC. But what if you don't have 4.0 BASIC? On the Upgrade BASIC machines you must either use LOAD"$0",8, which would destroy your current BASIC program, or use the DOS Wedge from Commodore which can only be used in direct mode. To solve the problem of getting a directory during a program RUN, one must use some other method.

This program allows you to display the directory at any time you want in your program. This comes in handy when loading or saving files from a disk. The ideas for this program came from the article "PET/CBM Disk Formats" **COMPUTE!** #13. In this article, the author says "Remember that you can read the BAM, header, and directory blocks as a single file name of $0 or $1. To read the data contained in these blocks you must use the GET# command since there are no carriage return delimiters". I followed this article and the tables listed in the article to make this program.

Lines 50100-50180 are for the title. If the title is not wanted or needed then replace the lines with this line. 50100 VV=253:GOSUB50320.

I did not check for any disk errors in this program. If needed, modify to your own personal needs. This program works on the 2040 disk and should work on the 3040 and 4040 disk systems.

```
READY.

 50000 REM
 50010 REM DIRECTORY FOR 3.0
 50020 REM
 50030 REM STEPHEN MEIROWSKY
 50040 REM
 50050 INPUT"DRIVE #0,1";M
 50060 A$=STR$(M):B$="I"+A$
 50070 OPEN 14,8,15:PRINT#14,B$
 50080 F$="$"+RIGHT$(A$,1)+",SEQ"
 50090 OPEN 1,8,2,F$
 50100 GET#1,G$:FO=ASC(G$)
 50110 VV=140:GOSUB50320
 50120 PRINT:PRINTM"▮"CHR$(34);
 50130 FORWW=144TO159:GET#1,G$:PRINTG$;
       :NEXT:PRINTCHR$(34)" ";
 50140 VV=2:GOSUB50320:PRINTG$;
 50150 GET#1,G$:PRINTG$" ";
 50160 IFFO=1THENPRINT" 1"
 50170 IFFO=65THENPRINT"2A"
 50180 VV=91:GOSUB50320
 50190 FORYY=1TO8:A$="":GET#1,G$
 50200 IFG$=""THENVV=28:GOSUB50320:GOTO50290
 50210 IFG$=CHR$(13)THEN50310
 50220 TT=ASC(G$):VV=1:GOSUB50320
 50230 FORWW=3TO18:GET#1,G$:A$=A$+G$:NEXT
 50240 VV=9:GOSUB50320:PRINTASC(G$);
 50250 PRINTTAB(5)CHR$(34)A$CHR$(34);
 50260 IFTT=129THENPRINTTAB(24)"SEQ"
 50270 IFTT=130THENPRINTTAB(24)"PRG"
 50280 GET#1,G$
 50290 IFYY<>8THENVV=1:GOSUB50320
 50300 NEXT:GOTO50190
 50310 CLOSE1:CLOSE14:END
 50320 FORWW=0TOVV:GET#1,G$:NEXT:RETURN
```

©

Editor's Note: This program will work on all PETs. For 4.0 (if using disk) you might want to relocate it. — RTM

Inversion Partitioning

David O. Williams
Toronto, Canada

Several people have written for **COMPUTE!** about methods of putting more than one BASIC program into the PET's memory. (See the articles by Elizabeth Deal, Charles Brannon and Harvey Herman in **COMPUTE!** #13.) The methods they use involve "partitioning" the memory into several blocks, each usually 8K in size, and arranging the pointers so that each block becomes the entire working memory space in which a program can be stored and run. While it is working in one block, the pointers prevent the PET from affecting the memory in the other blocks in any way. Programs which are stored there are undisturbed, waiting for the pointers to be realigned by an appropriate machine language routine.

This technique suffers from several disadvantages. The tape SAVE and LOAD commands do not work when a BASIC program is not in the usual position in memory, so various tricks such as saving through the machine language monitor have to be used. A far more serious problem is that each program is confined to 8K of memory. Any program which needs a total of more than 8K, including the space it uses for variables, arrays, and strings, will crash with an OUT OF MEMORY ERROR. Also, any program which performs a lot of string manipulations will run slower in a small memory space because of the greater frequency of garbage-collection delays. There may be plenty of spare memory in the other blocks which is effectively useless.

Providing you don't want more than two programs in your PET, you can avoid all of these problems by using the machine language routine here. It resides in the second cassette buffer, and its main effect is to turn the entire contents of the memory which is accessible to BASIC upside down! A BASIC program which is in the normal place at the bottom of memory is thus moved to the top. (It is also turned upside down, but that doesn't matter since no attempt is made to run it in that position.) A second program can then be written or loaded from tape into the bottom of memory. When this program is RUN it has access to all the unoccupied memory between the two programs which it can use for variables, arrays, and strings. (The machine language routine adjusts the pointers so that the second program cannot over-write the text of the first.) When you want to return to the first program, simply run the machine language routine again. It moves the first program back to the normal position, stores the second one at the top of memory, and adjusts the pointers so that the first program has access to all the working space without over-writing the second one. In this way, you can interchange the two programs as many times as you want. At any time, the program which is in the "running position" at the bottom of memory can be handled in all the normal ways for BASIC programs. You can SAVE it, edit it, delete it with NEW, LOAD another program, anything you want.

It all sounds very simple in theory, and in practice it isn't much more difficult. First, engage the machine language monitor by entering SYS 1024, then display the contents of the second cassette buffer with the command ".M 033A 03F1". The resulting display will look very like the listing except that the numbers in the body of the table will be different. Move the cursor to the first line of the table, change all the numbers to match the listing and press RETURN at the end of the line. Do the same thing to the second line, and so on to the end of the table. You have now written (or at least copied) a machine language program. Save it on tape (before you run it — this is important) with the monitor still engaged by using the command .S "EXCHANGE",01,033A,03F2. Press PLAY and RECORD, as requested by the machine, and wait for the cursor to reappear. (In the future you can get to this point by putting this tape into the deck, entering the monitor, and giving the simple command " .L". You will be told to press PLAY, and the tape will load into the correct memory locations.) Now exit the monitor by entering " .X". The familiar word READY will appear, and you are back to BASIC.

Now all you have to do to invert the memory is enter SYS 826. The process of turning the whole memory over takes a second or so, then the word READY will reappear.

Incidentally, Exchange will work with memories of any size, so even 8K PET owners can use it to get two short programs into their machines. Exchange will also leave intact any machine language you may have located in high RAM, *provided* that Exchange is initialized (i.e. run for the first

time) *after* the high-RAM program is in place and protected with the top-of-BASIC-memory pointer.

Have fun with it....!

```
Ø33A A2 ØØ EØ ØØ DØ 1D A5 35
Ø342 29 FE 48 38 E9 Ø1 8D 78
Ø34A Ø3 68 48 18 6A 69 Ø1 8D
Ø352 94 Ø3 68 69 Ø3 8D AA Ø3
Ø35A 8D B5 Ø3 AØ ØØ A5 Ø1 48
Ø362 A5 Ø2 48 A5 14 48 A5 15
Ø36A 48 A9 Ø1 85 Ø1 A9 Ø4 85
Ø372 Ø2 A9 FE 85 14 A9 3F 85
Ø37A 15 A1 Ø1 48 A1 14 81 Ø1
Ø382 68 81 14 E6 Ø1 C6 14 E4
Ø38A Ø1 DØ EE E6 Ø2 C6 15 A5
Ø392 15 C9 21 DØ E4 68 85 15
Ø39A 68 85 14 68 85 Ø2 68 85
Ø3A2 Ø1 38 A9 FF E5 2A 48 A9
Ø3AA 43 E5 2B 48 A9 FF E5 34
Ø3B2 85 2A A9 43 E5 35 85 2B
Ø3BA 68 85 35 68 85 34 EC 3D
Ø3C2 Ø3 DØ 17 A9 Ø4 8D 3D Ø3
Ø3CA 8E Ø1 Ø4 8E Ø2 Ø4 85 29
Ø3D2 85 2B A9 Ø1 85 28 A9 Ø3
Ø3DA 85 2A A5 2A 85 2C 85 2E
Ø3E2 A5 2B 85 2D 85 2F A5 34
Ø3EA 85 3Ø A5 35 85 31 6Ø ØØ
```

©

A Personal News Service

Ed Steinfeld
Hudson, NH

It's easy to get your home computer to type letters, maintain an inventory, store financial data, and play chess. But try to make it into a dumb terminal — that's difficult!

I have owned a PET 2001 and a TNW 232/488 (dual RS232 port) for about three years and used the RS232 port only to interface my Selectric. (I now own an 8032 CBM, 4040 disk, 2022 printer, TNW 232/488, Harris 1030 Selectric, and an acoustical coupler.) A few weeks ago I reread Jim Butterfield's article, "Basic CBM 8010 Modem Routines" in issue 7 of **COMPUTE!**. The 8010 programming is similar to the programming for the TNW 232/488 device. I entered the code with a few modifications, connected an acoustical coupler to the RS232 port, dialed a Digital Equipment Corporation (DEC) VAX-11/780, and I was on-line to another computer.

The VAX expected a number of characters I couldn't type from the CBM keyboard. I had to define a few keys for the VAX CONTROL characters. This was easy because Jim Butterfield used a table-driven translator in his program. This translator was used to convert CBM ASCII into standard ASCII and vice versa. I used this same table to convert the RVS, OFF, Left Arrow, and Up Arrow keys into CONTROL C, CONTROL Z, CONTROL S, and CONTROL P keys. These control keys abort programs, terminate input, and control scrolling. I got all this to work properly in about three weeks. Once the program worked, I contacted the Dow Jones News/Retrieval Service (DJNRS) (800-257-5114). Dow Jones & Company have an information service available to the home or business. This news service is not unlike the press wire services except the subscriber selects the information he wants rather than receiving everything being sent on the wire. The news includes what you might expect from the publishers of *The Wall Street Journal* and *Barron's*, stock and financial data. It also has general news, industry news, corporate profiles — 10-K extracts, financial statistics, and historical stock quotes. Current day quotes with a 15-minute delay on stocks, options, and corporate bonds as well as current day updates for U.S. Treasury issues, mutual funds, and the national over-the-counter market are all available.

I was given a password to try the service for a weekend. By Monday I had subscribed. My 8032, with a little software, can log into the Dow Jones News/Retrieval Service (DJNRS), request stock quotes and up-to-the-minute news, store it all on my disk, and print it on the 2022 printer. I now have my own personal news service.

The program I use to access the DEC VAX (VMS operating system) and PDP-11/70 (RSX-11M PLUS) software is the same software I use to talk to the DJNRS. However, Dow doesn't require any of the CONTROL characters, it only uses the question mark to stop output. All commands on the DJNRS are punctuation keys. The program first prompts with the terminal attributes that tell the various operating systems the characteristics needed for proper operation of the terminal. The DJNRS is available via a TYMNET system and requires the user to type a single character called the terminal identifier when he first accesses the system. (In some areas, the DJNRS is accessed through a TELENET service.) The terminal identifier lets the host software know when extra fill characters are required after carriage returns and when a line feed is needed between lines. I have found the identifier 'A' works well with my software.

On DEC, two fill characters are required after every return. Then the program asks if the terminal session should be written to disk and, if so, which drive and what file name. If you write the session to the disk, everything on the screen will be written into the file (both what you type as well as what is sent by the host computer). Even with this activity, the CBM can run at 300 baud without missing a character.

After you select the disk option, the program clears the screen except for the top line. This line in reverse video has the telephone numbers of the two systems I use. This leaves 24 lines for display data. Twenty-four lines is the standard display area of most video terminals and is the default page size for both the DJNRS and DEC systems.

A second program, PRINT TERMINAL, is used to print the terminal session. I save myself money by recording the data on the disk and later printing it after disconnecting from the service. The charges for DJNRS are by minute of connect time and the type of data being accessed. There is a one-time connect fee of $50 and the non-prime-time minute rate is between 15 and 20 cents.

The TERMINAL EMULATOR program has one gotcha when you use the option of writing to the disk. I have no way to stop the program other than to press the STOP key. Since I can't determine when I'm through, there is no way to close the disk file. So after I press the STOP key, I must remember to type DCLOSE (CLOSE 7 on PETs) to close the disk file. If I use some other key to determine when the session is finished, more code would have to be added to the lines that decode the input and output. More code would not allow the CBM to run at 300 baud. If you modify the 'business part' of the program, you may begin to lose characters. I have added about all that can be added and still keep up with the hosts. Options such as Command-O, Basic Toolkit, and the Wedge cannot be used because they slow the execution speed of the TERMINAL EMULATOR program.

With the exception of the 80-character reverse video line and setting the window, this terminal emulator program will work equally well with 40 column PETs and CBMs. In fact the news data displayed by DJNRS is 40 columns by 24 lines. You would think they expected all Commodore owners to be DJNRS customers. A Commodore 8010 modem may be used instead of the RS232 device and an acoustical coupler.

When using the DJNRS you can select headlines of current general news or specific headlines where a particular company is mentioned in the article. If one of the headlines is of interest, you then request to have the article displayed. You don't have to waste time reading every article to see if a company you're interested in is mentioned. The attached terminal session printout shows the login, Commodore's stock quotes between April 15 and May 1, an article in which Commodore is mentioned, followed by a list of headlines of articles in which Commodore is mentioned.

With this simple TERMINAL EMULATOR program, your PET or CBM can become a dumb terminal and can access any number of information services. Now your home computer is a window to general and financial news services and you can read the news as fast as they receive it — your own Personal News Service.

Program 1.

```
100 REM**"TERMINAL EMULATOR  3 MAY 1981 ¬
    ¬ COPYRIGHT (C) 1981"
110 REM**"ED STEINFELD"
120 REM**"31 RICHMAN ROAD"
130 REM**"HUDSON, NH  03051"
140 REM**********************************
    ¬*********************************
    ¬*
150 REM**"CBM ASCII TO ASCII AND ASCII ¬
    ¬TO CBM ASCII CONVERTER"
160 REM**"THIS PROGRAM WILL EMULATE A ¬
    ¬VT52 WITHOUT CURSOR CONTROL.
170 REM**"USE TERMINAL IDENTIFIER 'A'"
180 REM**"EITHER A TNW 232D, OR 488/232 ¬
    ¬WITH AN ACOUSTICAL COUPLER
190 REM**"OR A CBM 8010 MODEM MAY BE ¬
    ¬USED.
200 TE=22:REM"     INSERT THE UNIT ¬
    ¬NUMBER OF RS232 OR MODEM DEVICE."
210 O=0:A$="":B$="":C$="":Q=0:I=0:J=0:
    ¬K=0:Q$="":DIM F(255),T(255)
220 PRINT"hhh           TERMINAL ¬
    ¬EMULATOR BY E.F. STEINFELD (C) ¬
    ¬1981"
230 FORJ=32 TO 64: T(J)=J:NEXT:T(13)=13:
    ¬T(18)=3:T(20)=127:T(7)=7:T(8)=20
240 FORJ=65 TO 90:K=J+32:T(J)=K:NEXT:
    ¬T(9)=9
250 FORJ=91 TO 95:T(J)=J:NEXT
260 FORJ=193 TO 218: K=J-128:T(J)=K:NEXT
270 T(160)=32:T(141)=13:T(94)=19:
    ¬T(95)=17
280 FORJ=0 TO 255:K=T(J):IF K THEN ¬
    ¬F(K)=J:F(K+128)=J
290 NEXT:F(94)=94:F(95)=95:F(8)=20:
    ¬T(146)=26:F(27)=147
300 F(141)=13
310 POKE1020,0:POKE59468,14
320 OPEN5,TE
330 A$="    TERMINAL EMULATOR          "
340 PRINT"hhhr DJNS TEL. NO. 889-8618 ¬
    ¬***  DEC TEL. NO. 884-1707"A$
350 PRINTCHR$(15):FORI=1TO80:PRINTCHR$(9
    ¬)CHR$(137);:NEXT:PRINT:REM"CLEARS ¬
    ¬TABS
360 T$=CHR$(137):PRINTTAB(8)T$TAB(16)T$T
    ¬AB(24)T$TAB(32)T$TAB(40)T$TAB(48)T
    ¬$;
370 PRINTTAB(56)T$TAB(64)T$TAB(72)T$:
    ¬REM "SETS TABS EVERY 8 COL."
380 PRINT"      TYMNET    TERMINAL ¬
    ¬IDENTIFIER 'A'
390 PRINT"      VMS        SET TERMINAL/CRF
    ¬ILL=2
400 PRINT"      RSX-11M  SET TERMINAL ¬
    ¬HFILL:2
410 PRINT "↓     RVS = CTRL C, OFF = ¬
    ¬CTRL Z, ^ = CTRL S, ^ = CTRL P"
420 PRINT "↓     AFTER LOGOFF HIT ¬
    ¬rSTOPf TO TERMINATE THE PROGRAM."
430 INPUT"↓DO YOU WISH TO WRITE THIS ¬
    ¬SESSION TO THE DISK (Y OR N)   ¬
    ¬N←←←";Q$
440 Q$=LEFT$(Q$,1):IFQ$="N"THENQ=0:
    ¬GOTO520
450 IF Q$<>"Y"GOTO430
460 PRINT:PRINT"DON'T FORGET TO TYPE ¬
    ¬DCLOSE AFTER YOU STOP."
470 INPUT"↓DRIVE NO. (0 OR 1) _0←←←";D:
    ¬IFD<0ORD>1THEN470
480 INPUT"FILE NAME___←←←";Q$:IFQ$="_"TH
    ¬EN480
490 Q=1
500 DOPEN#7,D(D),(Q$)+",W"
510 IFDS<>0THEN PRINTDS$:END
520 PRINT"h";
530 GETA$:IFA$THENPRINT#5,CHR$(T(ASC(A$)
    ¬));
540 GET#5,A$:IFST=OANDA$<>B$THENC$=CHR$(
    ¬F(ASC(A$))):PRINTC$;:IFQTHENPRINT#
    ¬7,C$;
550 GOTO530
READY.
```

Table 1.

```
please type your terminal identifier
-1040-134-
please log in: dow1;
to> host is online

WHAT SERVICE PLEASE????
djns
ENTER PASSWORD
WWWWWWWWWWWWWWWW
MMMMMMMMMMMMMMMM
@@@@@@@@@@@@@@@@
%    %
DOW JONES NEWS/SERVICES  COPYRIGHT 1981
DOW JONES & CO., INC.
NOW..FREE TEXT SEARCHABLE NEWS DATABASE
LOW INTRODUCTORY RATE CALL 800-257-5114
ENTER QUERY
%    %;1cbu p1

DOW JONES HISTORICAL
STOCK QUOTE REPORTER SERVICE

STOCK 1CBU

 DATE      HIGH      LOW      CLOSE   VOL(100/S)
04/15/81   36        34 3/8   36        752
04/16/81   38 3/8    36 3/8   38 3/8   1717
04/20/81   42 3/8    37 3/4   42       1173
04/21/81   43 1/2    41 1/2   42       1481
04/22/81   44        40 1/2   40 1/2   1152
04/23/81   42 3/8    39 7/8   40 1/2    993
04/24/81   42 1/2    39 3/4   42 1/2   1079
04/27/81   42 3/4    40 3/8   40 5/8    489
04/28/81   40 1/2    37 5/8   39 1/2    928
04/29/81   39        37 1/8   37 3/4   1156
04/30/81   38 3/8    36 1/8   36 1/8    591
05/01/81   37 5/8    35 3/4   37        363

%    %.cbu
 N  CBU        01/02 AN 1/3
  /AAPL  CBU    TAN   DGN  /OFF    /
    05/01 SOFTWARE PIRATES A WORRY
   (WJ) FOR PERSONAL COMPUTER MAKERS
  THE BOOMING MARKET IN PERSONAL
COMPUTERS OWES ITS VITALITY NOT ONLY TO
THE LOW PRICES OF THE MACHINES
THEMSELVES BUT ALSO TO THE QUALITY OF
THE PROGRAMS, OR SOFTWARE. NOW, MANY IN
THE EMERGING PERSONAL COMPUTING
INDUSTRY ARE WORRIED BY A RECENT RISE
IN WHAT  SOME CALL SOFTWARE PIRACY.
INCREASINGLY, INDIVIDUALS ARE
REPRODUCING  MICROCOMPUTER PROGRAMS
THAT AREN'T  SUPPOSED TO BE DUPLICATED.
  MOST OF THE PROGRAMS ARE COPYRIGHTED,
AND THE AUTHORS HAVE USED SOPHISTICATED
RECORDING TECHNIQUES TO GUARD AGAINST
COPYING. MOST PEOPLE MAKING THE
DUPLICATES SEEM TO BE DOING SO AS
INSURANCE IN CASE SOMETHING GOES WRONG
WITH THEIR ORIGINAL, AS IT OFTEN DOES.
  BUT MANY OTHERS ARE GIVING COPIES
AWAY BY THE DOZEN, AND SOME ARE SELLING
%    %
 N  CBU        01/02 AN 2/3
THEM.
  BUT MANY OTHERS ARE GIVING COPIES
AWAY BY THE DOZEN, AND SOME ARE SELLING
THEM.
  BY ONE ESTIMATE, UNAUTHORIZED COPYING
HAS RISEN FROM ECONOMIC INSIGNIFICANCE
TO CUT SALES OF THE $200-MILLION-A-YEAR
MICROCOMPUTER SOFTWARE PUBLISHING
INDUSTRY BY $12 MILLION TO $36 MILLION
ANNUALLY.
  AT PRESENT, THE COPYING AFFECTS ONLY
SUCH RELATIVELY SMALL MAKERS OF
PERSONAL DATA-PROCESSING EQUIPMENT AS
APPLE  COMPUTER, COMMODORE
INTERNATIONAL AND  TANDY CORP. AND THE
EVEN SMALLER  COMPANIES THAT WRITE AND
PUBLISH MOST OF THE PROGRAMS FOR SUCH
MACHINES. BUT IT  ALSO COULD HINDER NEW
VENTURES BY THE  LARGER, TRADITIONAL
COMPUTER COMPANIES,  SUCH AS DATA
GENERAL, DIGITAL EQUIPMENT,
INTERNATIONAL BUSINESS MACHINES AND
WANG LABORATORIES THAT ARE EXECTED TO
%    %
 N  CBU        01/02 AN 3/3
ENTER  THE PERSONAL COMPUTER MARKET
WITH  PRODUCT INTRODUCTIONS AT THE
NATIONAL  COMPUTER CONFERENCE OPENING
MONDAY IN  CHICAGO.
  TO SOME DEGREE, INDUSTRY OBSERVERS
SAY, SOFTWARE PUBLISHERS BROUGHT THE
COPYING PROBLEM ON THEMSELVES BY
FAILING TO PROVIDE ADDITIONAL OR
REPLACEMENT  COPIES OF IMPORTANT
PROGRAMS QUICKLY. IN SOME CASES, USERS
HAVE COME TO DEPEND ON UNCOPYABLE
PROGRAMS THAT WENT AWRY, ONLY TO FIND
THAT THE PROGRAM TOOK WEEKS TO
REPLACE.
  "ANY COMPANY THAT DOESN'T PROTECT ITS
USERS BETTER THAN THAT DESERVES EVER
BOOTLEG COPY MADE," SAYS ALLAN
TOMMERVIK, MANAGING EDITOR OF SOFTALK,
A MAGAZINE FOR OWNERS OF APPLE
COMPUTERS.
%    %
 NO PAGE
%    %.cbu 01
 N  CBU        01/02
 AN 05/01 SOFTWARE PIRATES A WORRY
  (WJ) FOR PERSONAL COMPUTER MAKERS
 AM 04/22 COMMODORE INTL OUTLOOK -2-
  (DJ)
 AL 04/22 COMMODORE INTL SEES 4TH QTR
  (DJ) AS BEST IN COMPANY HISTORY
 AK 04/22 COMMODORE INTL EARNS -2-
  (DW)
 AJ 04/22 COMMODORE INTL LNET 66C A SHR VS 38C
 AI 04/13 UP & DOWN WALL ST: IBM
  (BN) SAID TO HAVE SMALL COMPUTER
 AH 04/02 COMMODORE INTERNATIONAL
  (DW) UNVEILS MICROCOMPUTER
 AG 03/26 TANDON CORP RECEIVES
  (DW) $9.3 MILLION ORDER
 AF 03/05 COMMODORE INTERNATIONAL GETS
  (DW) $6.2 MILLION ORDER FROM ATARI
 AE 03/04 IBM SAID TO WEIGH MATSUSHITA
  (WJ) AS MICROCOMPUTER SUPPLIER
 AD 03/03 COMMODORE INTL ISSUES
  (DJ) STATEMENT ON BARRON'S STORY
%    %
 N  CBU        02/02
 AC 03/02 COMMODORE INTERNATIONAL
  (BN) PROSPECTS UNCERTAIN
 AB 02/18 COMMODORE INTL. SEES HIGHER
  (DW) FISCAL 1981 NET -2-
 AA 02/18 COMMODORE INTL SEES FISCAL
  (DW) 1981 NET OF MORE THAN $2.40 A SH
%    %disc
```

©

FOR/NEXT GOSUB/RETURN, And The Stack

Jim Butterfield
Toronto, Canada


```
100 print "stack sniffer       jim butterfield"
110 data 186,189,3,1,160,1,217,-11,240,4,136,16,248,96
    ,185,-9,32,210,255
120 data 185,-7,32,210,255,169,32,32,210,255,138,24,
    121,-5,141,-1,138,121
130 data -3,170,188,4,1,189,3,1,132
140 data 300,107,137,133,136,32,148,220 : remark
    – orig
150 data 300,27,55,133,54,32,206,220 : remark – upgr
160 data 300,66,55,133,54,32,120,207 : remark – 4.0
170 data 169,13,32,210,255,174,-1,208,189,0,129,
    141,76
180 data 83,80,66,18,7,14,1,0
190 r=peek(65534)
200 v=52: if r=107 then v=134
210 a=peek(v)+peek(v+1)*256
220 b=a-80:b%=b/256
230 poke v,b-256*b%:poke v+1,b%
240 poke v-4,b-256*b%:poke v-3,b%
250 for j=0 to 79
260 read x:t=t+x
270 if x<0 then y=a+x:x=int(y/256):poke j+b,y-256
    *x:j=j+1
280 if x<>300 goto 320
290 read x:t=t+x:if x=r goto 310
300 for k=1 to 6:read x:t=t+x: next k:goto 260
310 for k=1 to 5:read x:t=t+x:poke j+b,x:j=j+1:
    next k: read x:t=t+x
320 poke j+b,x
330 next j
340 print"analyze stack with sys";b
350 print"(checksum =";t;")"
360 print"checksum should be 9965"
```

FOR...NEXT loops are quick and easy to code. GOSUB to a subroutine with RETURN at the end works very nicely. They never give you trouble. Except, once in a while, very subtle trouble. An ?OUT OF MEMORY when you know you have lots of memory left ... a ?NEXT WITHOUT FOR when you can see the FOR in the listing. What's going on here?

We'll try to establish a few rules by means of experiments. Type in the program with the elegant name of Stack Sniffer. Run it, and it will park itself out of the way in high memory; it will remain there until you power down. Make a note of the SYS address; we're going to use it quite a few times. If the checksum is wrong, check your data statements: there must be an error in there.

Checking Out The Program

Type NEW. Now type in:

CBM/PET? SEE SKYLES ... CBM/PET?

"Should we call it Command-O or Command-O-Pro?"

That's a problem because this popular ROM is called the Command-O-Pro in Europe. (Maybe Command-O smacks too much of the military.)

But whatever you call it, this 4K byte ROM will provide your CBM BASIC 4.0 (4016, 4032) and 8032 computers with 20 additional commands including 10 Toolkit program editing and debugging commands and 10 additional commands for screening, formatting and disc file manipulating. (And our manual writer dug up 39 additional commands in the course of doing a 78-page manual!)

The Command-O extends Commodore's 8032 advanced screen editing features to the ultimate. You can now SCROLL up and down, insert or delete entire lines, delete the characters to the left or right of the cursor, select TEXT or GRAPHICS modes or ring the 8032 bell. You can even redefine the window to adjust it by size and position on your screen. And you can define any key to equal a sequence of up to 90 key strokes.

The Command-O chip resides in hexadecimal address $9000, the rightmost empty socket in 4016 and 4032 or the rearmost in 8032. If there is a space conflict, we do have Socket-2-ME available at a very special price.

Skyles guarantees your satisfaction: if you are not absolutely happy with your new Command-O, return it to us within ten days for an immediate, full refund.

Command-O from Skyles Electric Works **$75.00**

Complete with Socket-2-Me **95.00**

Shipping and Handling .... *(USA/Canada)* **$2.50** *(Europe/Asia)* **$10.00**

California residents must add 6%/6½% sales tax, as required.

Skyles Electric Works
231E South Whisman Road
Mountain View, California 94041
(415) 965-1735

Visa/Mastercard orders: call tollfree **(800) 227-9998** (except California). California orders: please call **(415) 965-1735.**

WATCH

THIS

SPACE

CBM/PET? SEE SKYLES ... CBM/PET?

"You mean this one little Disk-O-Pro ROM will give my PET twenty-five new commands?

And for just $75.00? Why, that's only $3.00 a command!"

The Disk-O-Pro in any PET with Version III (BASIC 2.0) ROMs **(### COMMODORE BASIC ###)** will give 19 software compatible disk instructions*: 15 identical with the new BASIC 4.0 (or with 8032 ROMs) compatible with both old and new DOS. Plus 4 additional disk commands...including appending **(MERGE)**, overlaying **(MERGE #________)** and **PRINT USING**, allowing formatting output of strings and numbers on the PET screen or on any printer.

**NOTE: Old DOS doesn't recognize three of the commands.*

Those are just 3 of the important commands—and there are 7 more beauties—on your Disk-O-Pro that have never been available previously to PET/CBM users. (Skyles does it again!)... Beauties like the softtouch key **(SET)** which allows you to define a key to equal a sequence of up to 80 keystrokes; like **SCROLL** whereby all keys repeat as well as slow scrolling and extra editing features; like **BEEP** which allows you to play music on your PET.

The Disk-O-Pro is completely compatible with the BASIC programmer's Toolkit. The chip resides in the socket at hexadecimal address $9000, the rightmost empty socket in most PETS. And for the owners of "classic" (or old) PETS, we do have interface boards.

(For those owning a BASIC 4.0 or 8032, even though the Disk-O-Pro may not be suitable, the Command-O is. Just write to Skyles for additional information. Remember, we have never abandoned a PET owner.)

Complete with 84-page manual written by Greg Yob...who was having so much fun that he got carried away. We had expected 32 pages.

Skyles guarantees your satisfaction: if you are not absolutely happy with your new Disk-O-Pro ROM chip, return it to us within ten days for an immediate full refund.

Disk-O-Pro from Skyles Electric Works **$75.00**

Complete with interface board (for "classic" PETS) **95.00**

Shipping and Handling .... *(USA/Canada)* **$2.50** *(Europe/Asia)* **$10.00**

California residents must add 6%/6½% sales tax, as required.

Skyles Electric Works
231E South Whisman Road
Mountain View, California 94041
(415) 965-1735

Visa/Mastercard orders: call tollfree **(800) 227-9998** (except California). California orders: please call **(415) 965-1735.**

WATCH

THIS

SPACE

```
100 FOR J=1 TO 50
200 FOR M=7 TO 3
```

Type the SYS command that Stack Sniffer wants. The system should respond READY. That means that there are no unclosed loops or unfinished subroutines. We haven't run the program yet.

Conclusion 1: You don't build loops or subroutines until you run the program.

Now type RUN. The system will answer READY immediately. Now give the SYS command. You should get:

```
LP IN 200
LP IN 100
```

Note that the loops are reported from the innermost out.

Conslusion 2: Loops stay open if they are not closed, even if the program is finished.

Add line 150 to the program:

```
150 GOSUB 200:STOP
```

Type the SYS command without RUNning. You'll get READY.

Conclusion 3: Adding or changing a line clears all loops, just as it clears all variables.

Say RUN, and then give the SYS command again. You'll get three lines back:

```
LP IN 200
SB IN 150
LP IN 100
```

Conslusion 4: Loops and Subroutines seem to "nest," one within another.

By this time, Stack Sniffer has been checked out well enough that we know it is OK.

Loops Within Subroutines

Now add the line: 300 RETURN. If you read through the program, you will see that the program calls the subroutine, returns from it, and then stops. The loops at 100 and 200 have never been closed. What will be on the stack? Type RUN, then give the SYS and see.

What? The loop at 200 has been closed!

Conclusion 5: If a FOR loop is opened within a subroutine, it will be terminated when the subroutine returns.

This can be a good way to clean out a FOR-...NEXT loop structure that is no longer needed.

Loops Within Loops

Delete line 150. Type in line 300 NEXT J. LIST the program and see what you think will happen to the loops when you RUN. Now try it.

Result: no loops, even though we didn't close the inner one.

Conclusion 6: Closing an outer loop always closes any loops inside.

Type in 150 IF J=7 THEN STOP. In this case, we're going to stop the program before the outer loop has completed its cycle. What will happen to the inner loop? RUN and then give SYS and see.

Result: the J loop is open, but the K loop is closed. How come? It seems that the NEXT J must close the K loop.

Conclusion 7: Performing NEXT on an outer loop always closes all loops inside.

This rule is what makes it impossible for you to incorrectly nest loops. If you were to type: FOR X=1 TO 4:FOR Y=1 TO 3:PRINT"H":NEXT X:NEXT Y you'd get a ?NEXT WITHOUT FOR error message. Do you see why?

Early Exit From Loops

Say NEW and start over. This time, we'll enter the program:

```
300 FOR A=3 TO 9
350 FOR B=4 TO 15
400 IF A+B=20 GOTO 600
450 NEXT B
500 NEXT A
550 STOP
600 FOR F=1 TO 7
650 NEXT F
```

Think about this one. This kind of coding can happen regularly when you are looking things up in a table. When you find what you want (line 400) you exit the loop and go on to other business.

RUN the program and then give the SYS call.

Surprised? The loops at 300 and 350 were never closed. They are still active.

Conclusion 8: Jumping out of a loop leaves the loop active. This can run you out of stack space very quickly.

What can we do about this? Let's try the following, which doesn't seem to do anything significant: change lines 600 and 650 to read: FOR A ... and NEXT A. RUN and SYS.

Wow! We've cleaned everything up. How did we do it? The only change was in the name of the loop variable. A was previously used for the outermost loop: re-opening a loop using A closed the old one and closed the B loop too.

Conclusion 9: Re-opening a loop closes the previous loop activity and all inner loops.

Picky point for technical tyros: this always works except when you try to do it after calling a subroutine: you may remember that the subroutine "separates" the FOR/NEXT loop entries.

Interlude

We've arrived at a very important coding moral: if you jump out of a loop, be prepared to do something about it. Otherwise, the loop stays around and this can result in one of two possible problems.

First, your stack (where the loop information is kept) will fill up and you'll get ?OUT OF MEMORY even though it seems you have lots of memory. Annoying.

Alternatively, you may accidentally open a loop using the same variable name much later in the program. What's worse (pay careful attention here), by this time we're inside some genuine loops that we are using. Now: as far as BASIC is concerned, you're re-opening the outermost loop; it's outermost because it was opened a long time ago and all subsequent loops were fitted inside. Think about that, and the enormity of what's going to happen will dawn on you ... BASIC will close absolutely every loop it's got before opening the one you're asking for. You'll get ?NEXT WITHOUT FOR when you try to exercise one of the outer loops. Insidious.

How do you police this problem? It's not practical to insist that you never jump out of a loop until it's completed. You have several alternatives:

*Do this kind of loop activity in a subroutine. When you return, the loop will be closed automatically.

*Name all your outermost loops by the same name, I. Then name the next level inside with the same name, J ... and so on. This ensures that every time you exit from a loop you will soon open another with the same name.

*Write a dummy loop on the same variable immediately after you exit. FOR Y = 1 TO 1: NEXT Y will do the trick nicely. The Y loop will end up good and closed.

Subroutines

If you write your subroutines correctly (jump into the beginning, have a neat RETURN by itself at the end, don't jump out), you won't have any trouble. The thing that beginners forget in the heat of the moment is that you can't just GOTO out of a subroutine directly into your main coding.

Type NEW and enter the following program:

```
100 GOSUB 300
110 STOP
300 X=X+1
310 IF X=1 THEN GOSUB 300
```

We deliberately have left off the RETURN to see what's happening here. This is an example of coding beloved to some clever types, but utterly horrible to most of us. It's called recursive programming, with the emphasis on the curse. What that means is that the subroutine at 300 calls itself. Do it and see what happens with SYS.

Hmmm. Both calls are neatly sitting there ... the one from 100 and the subsequent one from 310.

Conclusion 10: Subroutines don't work like loops. Calling the same one twice doesn't clean the old one out.

Add 320 RETURN and RUN it again. Call SYS. Check the value of X and see if it is what you expect.

Conclusion 11: Recursive subroutines will work, if you're that fool hardy.

Now we're ready for our final experiment in this series. Type new and enter the single line:

```
500 GOSUB 500
```

This is the ultimate in recursive subroutines. The subroutine calls itself, and then calls itself ... and so on. It will give you an ?OUT OF MEMORY incredibly fast. Adding a RETURN won't help; the stack is used up in milliseconds. RUN it, read the bad news, and then give the SYS command.

Huh? The stack is clean. How can that be? Well ...

Conclusion 12: Some error messages clean the stack, removing all GOSUB calls and FOR... NEXT loops.

Those of you who love speed contests: type RUN, and then see if you can hit the RUN/STOP key quick enough to catch the program before it bombs. It can be done, but you'll need super nimble fingers. If you do stop it in time, you might like to give the SYS command and see all those subroutine calls stacked up.

Conclusion

FOR ... NEXT and GOSUB/RETURN have a few subtleties. It's well to be aware of them.

It doesn't hurt to keep a copy of Stack Sniffer around and try them on your own programs. How many loops or subroutine calls have you left open? Look through your programs, and draw your own conclusions... ©

New Listing Conventions For CBM

New machines — VIC and "FAT-40" — and 4.0 BASIC have added a host of new editing functions and color control codes. To make **COMPUTE!**'s program listings as easy as possible for you to type in accurately, we will list CBM programs in a new, simpler way. Starting in this issue, you will see that our previous method — reconstructed cursor symbols — has been replaced by bracketed words. [DOWN] will mean the cursor-down key. [3 LEFT] will mean three cursor-lefts, and so on.

We will continue to split program lines with the ~ symbol. It signals that the line is continued below and prevents any spaces from being hidden. All shifted characters and graphics are represented by their underlined non-graphics equivalent. Line 110 in David Swaim's article, "High Resolution Bar Graphics for the PET," is a good example of the new conventions. We hope you will agree that this change will simplify your typing of programs. Let us know how you feel.

Here is a table of the new conventions:

Key To COMPUTE!'s CBM Listings

All Machines

Clear Screen	{CLEAR}
Home Cursor	{HOME}
Cursor Up	{UP}
Cursor Down	{DOWN}
Cursor Right	{RIGHT}
Cursor Left	{LEFT}
Insert Character	{INST}
Delete Character	{DEL}
Reverse field on	{RVS}
Reverse field off	{OFF}

CBM 8032/"FAT 40"

Set Window Top	{SET TOP}
Set Window Bottom	{SET BOT}
Scroll Up	{SCR UP}
Scroll Down	{SCR DOWN}
Insert Line	{INST LINE}
Delete Line	{DEL LINE}
Erase to Beginning	{ERASE BEG}
Erase to End	{ERASE END}
Toggle TAB	{TGL TAB}
TAB	{TAB}
ESCape key	{ESC}

©

Nuts And Volts: Build Your Own Controllers: Part III

Gene Zumchak
Buffalo, NY

In the first installment of this series of articles on building your own controllers, I outlined what you would need as a minimum to develop independent controller systems. Specifically, you need a development system consisting of a computer system with some kind of machine language operating system, an EPROM PROGRAMMER, and an EPROM emulator. A console computer system like an APPLE or PET will do quite well as the computer system, provided some machine language capability has been added. That is, PEEK and POKE are not sufficient for reading and writing memory. While hand assembly of programs can get the job done, the use of text editor and assembler is preferred.

EPROM programmers are available from several sources for a wide range of prices. EPROM emulators, however, are less common. In the second installment of this series, the design of an EPROM emulator was provided. In Part III, we will begin to look at what is required for the target system itself.

Basically, there are two extreme approaches you can take; you can buy everything, or build everything. While the first may look attractive to someone with no hardware experience, even if you can buy a suitable CPU board, there will undoubtedly be some custom I/O that you will have to provide. On the other hand, if you can afford the few extra bucks for a ready-made CPU board, the savings of your time may make the ready-made board a bargain. The advantages of doing it all yourself are usually compactness (low parts count) and low parts cost. You ought to be able to put together a complete system for about $100.

The cost for a ready-made CPU board will start at about $100. John Bell and Brachman Associates are two outfits that come to mind with boards in this range. If your application requires that it be field-programmable, then you should consider a board with a built-in keyboard/display and/or a serial interface. A versatile board having these features and a reasonable price is the SYM. The SYM has a powerful 4K machine language monitor. Two 6522s are availble for I/O and a third can be plugged in. Sockets for extra RAM and EPROM are also available, and the SYM can be jumpered to reset into your program. If ASCII style I/O or hard copy is necessary, the AIM-65 should be considered, although you are now talking over $400.

Power Supply

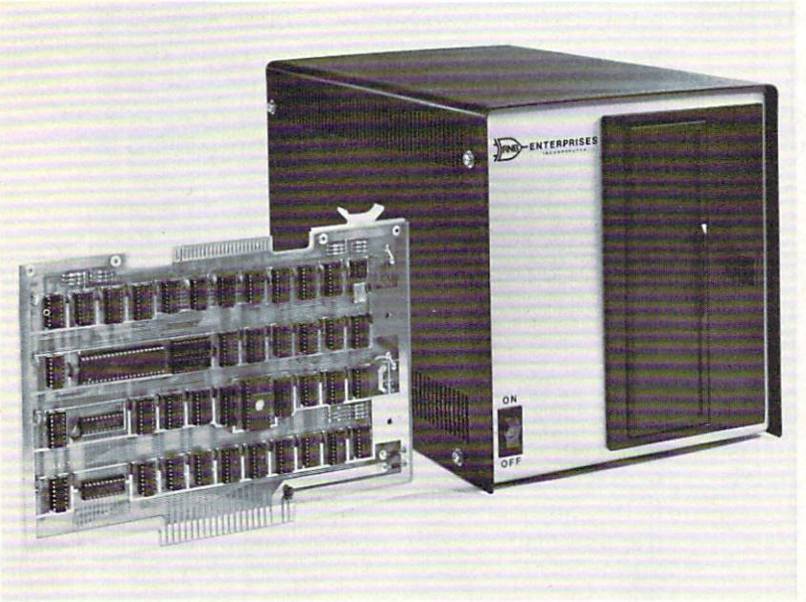
Whether you build your own board, or use a ready-made, single-board computer like the SYM, you'll need to provide a power supply. Kits and assembled 5-volt supplies are reasonably priced, or you can build your own.

Obtaining a regulated voltage from a raw DC voltage is easy with a three-terminal voltage regulator IC. The 7805 can provide one amp if adequately heatsunk and costs less than a buck. Obtaining the raw DC is only a little more work. Having to bring high-voltage AC into your board or system can be hazardous, but this can be avoided by buying what is called a "wall" transformer. This is a molded box containing a transformer that plugs directly into an AC socket. The output is low-voltage AC or DC. For currents 500 ma or less, the rectifiers and filter capacitor are usually included and the output is a DC voltage. Such supplies have been used with calculators for a long time. For larger currents, only AC is available and you will have to provide the rectifiers and filters. Wall transformers are available from a number of mail order outfits. For most controller applications, a single five-volt supply with a one-amp capacity will probably be adequate. Figure 1 shows such a supply made with an AC wall transformer.

While the 7805 can provide 1 amp with adequate heatsinking, since they are so inexpensive, a better idea is to share the load among two or more. Note: this does not mean paralleling the outputs. Instead, the five-volt loads are split up, each piece getting its own regulator. By running the regulators considerably below their capacity, heatsinking can be reduced or eliminated. Another possibility is using a regulator with a larger capacity. The LM323, for example, comes in a metal power

VAK-7 8″ FLOPPY DISK SYSTEM

FOR AIM-65, SYM-1



The VAK-7 Disk System was specifically designed for use with AIM-65 and SYM-1 Microcomputer Systems. The VAK-7 is a complete full size (8″) Floppy Disk System. This system will Read, Write and Format IBM Single and Dual Density diskettes. The VAK-7 is available with both Single and Dual Sided Disk Drives. Each Disk Drive comes with its own cabinet and Power Supply. The VAK-7 can handle up to 4 disk drives, totaling more than 4.98 Megabytes of storage.

The VAK-7 Disk System incorporates both advanced hardware and innovative software designs. The addition of the VAK-7 produces a very powerful and useful computer system. Unlike most other disk systems, there is no requirement for the user to provide RAM to hold the Disk Operating System software. No valuable time is wasted loading in the DOS from cassette tape, because the VAK-7 DOS software is in onboard ROM. The VAK-7 is located above the 32K boundary (8000 HEX), leaving the user with a full 32K bytes of contiguous address space for his own use.

AIM-65—Allows the user to save and load object code thru the AIM Monitor; to load, save, and append Text thru the AIM Editor; to load, save, and append Basic Programs thru the BASIC INTERPRETER; to assemble directly from disk single or multiple file programs.

SYM-1—Allows the user to save and load Files for use with the SYM Monitor, SYM Basic, and RAE-1.

ADDITIONAL COMMANDS:

ACTIVATE A DELETED FILE
COMPRESS A DISK
RENAME A DISK FILE
COPY A DISK
FORMAT A DISK
DELETE A DISK FILE
INITIALIZE A DISK
LIST CATALOG

SPECIFICATIONS:

- Completely assembled, tested, and burned in.
- Occupies address 8000–8FFF for AIM-65, $9000–9FFF for SYM-1, or $E000–EFFF for KIM-1.
- IBM Format; Single Density (128 bytes/sector); Dual Density (256,512, or 1024 bytes/sector).
- All ICs are in sockets.
- Fully buffered address and data bus.
- Standard KIM-4* BUS (both electrical pin-out and card size).
- Designed for use with a regulated power supply, but has provisions for adding regulators for use with an unregulated power supply.
- Dimensions: Board—10″ wide x 7″ high (including card-edge). Cabinet—9.25″ wide x 10″ high x 16″ deep.
- Power Requirements: +5v DC @ 2 Amps.
  117 AC 60Hz @ 2 Amps.
- Shipping Weight: 25 lbs.

**KIM-4 is a product of MOS Technology/C.B.M.*

PRICING:

VAK-7 — **$1,299.00**
Controller and One Single-Sided Drive

VAK-7A — **$599.00**
Additional Single-Sided Drive with Cabinet and Power Supply

VAK-7B — **$1,599.00**
Controller and One Dual-Sided Drive

VAK-7C — **$899.00**
Additional Dual-Sided Drive with Cabinet and Power Supply

CALL OR WRITE FOR FREIGHT CHARGES

RNB ENTERPRISES
INCORPORATED

4030 N. 27th Avenue, Suite D
Phoenix, AZ 85017
(602) 265-7564

transistor package and can supply three amps when heatsunk. The case or mounting tab for most positive, three-terminal regulators is the ground connection, permitting them to be attached directly to a grounded chassis or case.

With a 5-volt only EPROMS, like the 2716, being so cheap, it is now relatively easy to avoid parts that require other than a single five-volt supply.

The CPU

A controller will minimally consist of a CPU, an EPROM, some RAM, and some I/O. The first three items need not vary much from application to application, and you may even wish to lay out a standard controller board. The I/O, of course, will be a function of the application. If you are a dyed-in-the-wool 6502 fan, the choice of CPU is easy. While the 6502 is available in 28-pin packages (6503,6504, and 6505), the couple of square centimeters in board space saved is minimal. While you won't need all of the address space of the 6502, it will make decoding easier. Besides, since they are more common, 6502s are cheaper anyway. If you are open minded, and have assembled programs for more than one processor, you may consider other processor types, perhaps the Z-80 or the 6809, or a ROMless single-chip type. Since we are emulating program EPROM and not the processor, our development system can be used with any processor that can use EPROMs. It is a good idea to pick a processor that has built-in clock circuitry and only needs five volts. For convenience, we'll assume that we are using the 6502.

The 6502 can generate its own two-phase clock in a number of ways. The simplest is to use an RC network. While this method saves the price of a crystal, the loss of the accurate crystal time-base precludes accurate timing using either software or programmable timers, and is a false economy. A crystal may be connected in either a series or parallel mode. The series connection used by the SYM and KIM is shown in Figure 2a. Some asymmetry between the two phases usually occurs with this connection. Symmetry can be assured by applying a TTL square wave to the phase-zero input. The AIM uses this method starting with a four MHz crystal and a classical oscillator circuit. A pair of flip-flops are used to generate quadrature one MHz signals from the four MHz clock. (Only one is used.) A more straightforward circuit that merely divides by two twice is shown in Figure 2b. This circuit permits a two MHz clock to be selected for use with a two MHz 6502.

If some I/O device requires its own crystal of a specific frequency, it may be possible to get a suitable CPU clock from it. For example, a clock of about .9 MHz can be obtained by dividing the 3.58 MHz color-burst frequency by four. Use of a clock frequency that gives other than one-microsecond clocks makes counting time less easy.

The only control output signals required by memory devices will be ϕ2 and R/W. These are applied directly to family devices, and can be combined (in the proper plolarity) to provide read/write signals for non-family I/O. This will be illustrated later. The READY and interrupt inputs should be pulled up, whether used or not, with 3.3K resistors.

Most controllers are designed to be automatically reset when power is applied. The circuit of Fig. 3 will provide a power-on reset and also permits a manual reset capability.

RAM

Some RAM will be necessary for stack. There are many choices. The Motorola 68A10 contains 128 bytes for less than $3. The 6532 contains 128 bytes of RAM, two programmable ports, and a timer ($10). Since the timer has no free-running mode, the 6522 is probably a better choice for a port chip, and it has two timers. A pair of 2114L 1K x 4 chips ($6) will provide 1K of memory and is probably the safest bet.

Figure 4 shows the two basic ways of controlling the chip select and write enable pin of a 2114L. In 4a., the chip select is decoded from addresses only. The write enable gets a write strobe fabricated from R/W and ϕ2. In 4b., the write enable pin gets the R/W direction signal. This means that write strobing action must be applied via the chip select input. This is accomplished by gating an ordinary address select with ϕ2 as shown. The gating may be omitted, if the address select is generated using ϕ2 as a component. The scheme of 4b. should be used if the 6502's data bus is buffered in order to avoid bus contention during ϕ1 of a write operation.

In the next installment, I'll talk about I/O choices.

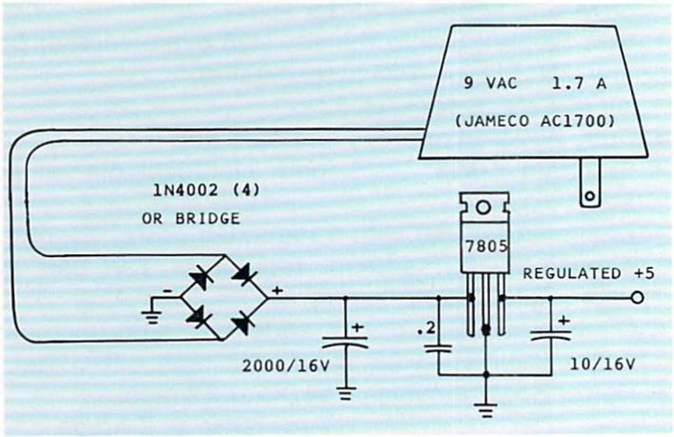
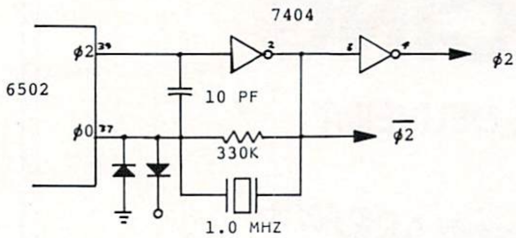
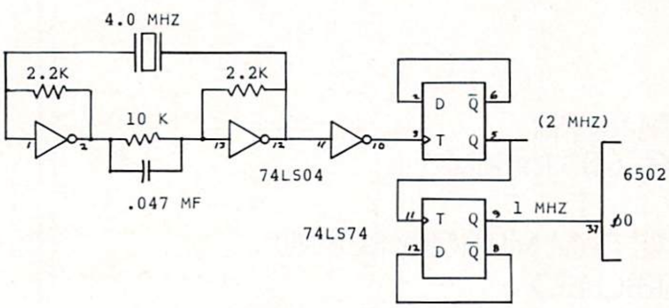


Figure 1. Regulated 5-volt supply using wall transformer.



a. Series Crystal Configuration.



b. Applied TTL Square Wave

Figure 2. Clock Generation for 6502.

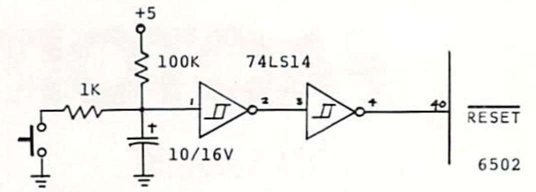
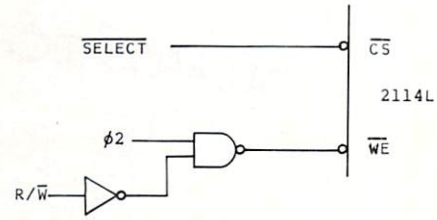
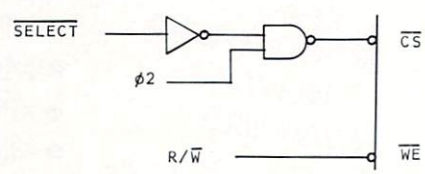


Figure 3. Manual and Power-on RESET for 6502.



a. Strobed Write Enable



b. Strobed Select

Figure 4. Control Signals for 2114L Static RAM.

©

Review:

DOS/65 – A Disk Operating System

(6502 Software)

Harvey B. Herman
Associate Editor

Ever since I installed an 8" disk on my KIM system (see **COMPUTE!** #11) I have been using KMMM disk operating software (DOS), sold by Willi Kusche. Recently, I saw an ad for a general purpose 6502 DOS, sold by Richard Leary.

From the beginning, Richard handles your order in a systematic and professional manner. His software can be run on a variety of machines, so you must give him, for example, the type of disk controller (Versafloppy in mine) and the location of the console I/O routine (my system uses standard KIM with echo defeated).

The package I promptly received in the mail contained an extensive instruction manual, a loader on cassette tape, and a diskette recorded with the operating system, an assembler/text editor and several utility programs. The utility programs include (among others): disktest, copy, debug, and format (not as yet for the Versafloppy, however). Users will appreciate that he has included the source code of many of the utility programs so they can be easily modified or enhanced if desired.

I had some trouble getting the system up for the first time as my read head is not aligned properly with track 0. Most people will not have this problem. The procedure required to bring the software up is trivial. First, load the tape. Second, run from location $200 (for the KIM system). However, using DOS/65 properly is a little more difficult than getting the first prompt ('A›'). It definitely helps to be familiar with CP/M, as Richard has emulated most of its functions. I found it much easier to read his manual after I had done some reading about CP/M. (So that's what "A›" means!)

An important advantage of DOS/65 is that it normally comes with an integral text editor and assembler. As a test, I typed in a program which transfers a table of data from one memory location to another. The editor features a variety of commands which allow easy correction of errors. It is even possible to edit programs which are larger than available memory. I had no trouble with any of the commands and in a short time I had created a file suitable for input to the assembler.

Running the two-pass assembler is a snap. The program just grinds away on its own and, in no time, two output files are produced. The first, a printer file, can be listed at some later time if desired. The second, a so-called "KIM" file, can be made into an executable file by a standard DOS command or loaded with the utility debugger program. I tested the later method and confirmed (by the debugger's disassembler) that the program was indeed in memory. I was surprised at how smoothly everything went once I got past some mental hangups.

DOS/65 appears to me to be a quite sophisticated program. It is complicated enough that, I believe, it should only be considered by serious assembly language programmers. It can be used with Microsoft BASIC, but the user may have to develop his own software interface. I confess this was beyond me. However, I feel certain Richard will offer his assistance if others are interested. The program does have a few awkward areas. For example, the user must specify, in decimal, the number of pages to save. However, some of that may be due to emulation of CP/M. An experienced user should have no problems with this software.

Richard A. Leary
1363 Nathan Hale Drive
Phoenixville, PA 19460 *$100*

©

ATARI SOFTWARE at DISCOUNT PRICES from

COMPUTER HOUSE

	Sug. retail	Our Price
Adventure International		
Adventures 1 thru 10 each (c)	19.95	16.00
Star Trek 3.5 (C)	19.95	16.00
Angleworms (C)	14.95	11.95
Sunday Golf (C)	14.95	11.95
Mountain Shoot (C)	14.95	11.95
Deflection (C)	14.95	11.95
Automated Simulations		
Datestones of Ryn (c)	19.95	16.00
Invasion Orion (c)	24.95	20.00
Rescue at Rigel	29.95	24.00
The Code Works		
Iridis - 1 (C)	9.95	7.95
(D)	12.95	10.35
Iridis - 2 (C)	15.95	12.75
(D)	18.95	15.15
Crystal Computer		
Galactic Quest (D)	29.95	23.95
House of Usher (D)	24.95	20.00
Imperial Walker (D)	29.95	23.95
Lazar Wars (D)	29.95	23.95
Little Crystal (D)	39.95	31.95
Sands of Mars (D)	39.95	31.95
Summer 4000 BC (D)	19.95	15.95
Dynacomp, Inc.		
Valdez (C)	14.95	11.95
(D)	18.95	15.15
Flight Simulator (C)	17.95	14.35
Optimized Systems Software		
Basic A+ (D)	80.00	70.00
OS/A (formerly CP/A) (D)	80.00	70.00
Personal Software		
Visicalc Atari	199.95	149.00
Program Design, Inc.		
Bowling (C)	16.95	13.55
Code Breaker (C)	16.95	13.55
Mini Crossword (C)	16.95	13.55
Preschool IQ Builder (C)	16.95	13.55
Word Search: Spanish	16.95	13.55
Quality Software		
Assembler (C)	24.95	20.00
6502 Disassembler (C)	11.95	9.95
Fastgammon (C)	19.95	15.95
Tank Trap (C)	11.95	9.55
(D)	14.95	11.95
Tari Trek (C)	11.95	9.55
(D)	14.95	11.95
QS Forth	79.95	63.95
Swiftly Software		
Programming Aids I (C)	14.95	11.95
(D)	19.95	15.95
File-It (D)	34.95	27.95
File-It 2 (D)	49.95	39.95
T.H.E.S.I.S.		
Word Scramble (C)	15.00	12.00
Spellbound (C)	15.00	12.00
SB Data Tapes 1 thru 10 each tape	5.00	4.00
All 10	50.00	36.00

(C) Cassette (D) Diskette

IF YOU DON'T SEE IT, write and ask!

Send orders to:

COMPUTER HOUSE

P.O. Box 369, Dept. 10 Mammoth Lakes, CA 93546

(714) 934-6538

Terms: FOR FAST DELIVERY, send certified checks, money orders or Visa or MasterCard number and expiration date. Personal checks require 3 weeks to clear. ADD $1.50 for postage. Orders over $100 we pay shipping. All foreign orders ad $10.00 CALIFORNIA residents add 6% tax. Prices subject to change.

LO•BALL COMPUTERS LO•BALL COMPUTE

apple®

ATARI®

EPSON

... and more

Apple® II

32K	$1099.00
48K	$1149.00
Disk II w/3.3 DOS	$ 529.00
Disk II	$ 450.00
Apple III w/128K	$3600.00

EPSON

MX-70	$ 399.00
MX-80	CALL
Apple card & cable	$ 99.00

ATARI®

400 w/16K	$ 439.00
800 w/16K	$ 775.00
810 Disk Drive	$ 449.00

We'll attempt to meet or beat any advertised price.

HEWLETT PACKARD

HP-85A w/16K	$2697.30

NEC

5510-2 w/Tractor	$2550.00
5520-2 w/Tractor	$2850.00

SOROC

IQ 120	$ 725.00
IQ 135	$ 799.00

C.ITOH

Comet	$ 499.95
Starwriter	$1450.00

Northstar, Altos and Zenith All 25% Discount

LO-BALL COMPUTERS

7677 S.W. Cirrus Dr.
Beaverton, OR. 97005

TO ORDER CALL (503) 641-0211

Ordering Information: For fastest service, send money order, cashier's check or bank wire. Visa and MC orders, add 3%. Personal checks accepted (allow minimum 10 days to clear). Hours 9-5, M-F.

Call for our Free Catalog.

LO•BALL COMPUTERS LO•BALL COMPUTE

Home Accounting System For The Atari

Sunnyvale, CA — A new home accounting system for the ATARI® 800™ Home Computer will replace the ATARI Accountant™, a small business accounting system, which will not now be offered for sale.

The new system will be lower-priced and easier to use than the current system, since it will be designed expressly for the home market. It will be ready for delivery in the first calendar quarter of 1982. The ATARI 815™ Dual Disk Drive, which was required for use of the professional accounting package, will not be offered for sale.

"The major thrust of our marketing efforts is toward use of our products in the home. We feel that our resources are better spent developing products aimed specifically at this market and segments that support this market, such as institutional education. It makes the most sense to convert the existing accounting package for home use," Roger H. Badertscher, president of Atari's Computer Division, said.

The new system will work with the ATARI 810™ Disk Drive, which is priced at $599.95. The dual disk drive was priced at $1,499.95, and offered "more capacity and a higher price than we feel is necessary for most home applications," Badertscher added.

No further details of the new accounting system are available at this time, except that it will include those features of the previous system that are best for the home market.

Business Package For Accounting And Tax Planning

Dakin5 Corporation's newest business application software package, The Depreciation Planner™ has been released to coincide with recent federal tax changes.

The Depreciation Planner is for use on the Apple microcomputer. It is designed to keep track of depreciable assets for accounting and tax planning purposes. This comprehensive package incorporates both the previous depreciation methods (to be used for assets purchased before January 1981), as well as the new depreciation methods (to be used for assets purchased after January 1981). It is faster than manual record keeping. It reduces chances of error and alleviates excess paperwork.

The user determines cost, salvage amount, useful life, and special restrictions or conditions pertaining to the asset and depreciation method. Once these figures are entered, The Depreciation Planner will automatically keep track of each asset.

The Depreciation Planner provides the following features:

— Automatically calculates current month depreciation, year-to-date and life-to-date amounts.

— Provides a printed list of assets in five formats.

Tag numbers
Location
Depreciation type
Depreciation Method
Asset Life

— Prints a list of all items that are potentially eligible for investment tax credit in each fiscal year.

— Prints a depreciation projection report that lists the projected current year and next year depreciation for each asset.

— Has a unique modeling feature that allows the user to experiment with any or all forms of depreciation for each asset. The information can be printed to allow thorough evaluation and comparison before a final selection is made.

The Depreciation Planner can work independently or will interface with The Controller® or The Business Bookkeeping System™. If used with either The Controller or The Business Bookkeeping System, The Depreciation Planner will automatically post depreciation amounts to the General Ledger and update current month, year-to-date and life-to-date amounts for each asset.

The documentation includes a complete glossary and modeling workbook to introduce users to the concept of depreciation and to give them hands-on experience with the software package.

The Depreciation Planner will provide significant benefits not only to businesses, but also to accountants. Accountants can use The Depreciation Planner to record all assets and depreciation information for their individual clients.

Owners of The Depreciation Planner may take advantage of Dakin5 Corporation's toll-free customer service line. This service, which is available to users of all Dakin5 products, is designed to answer user as well as dealer questions.

For further information contact Dakin5 Corp., 7475 Dakin St., 4th fl., Denver, CO 80221. (303) 426-6090.

The Controller is a registered trademark

THE PROGRAM STORE

Our Most Popular

Unbelievable Realtime 3-D Graphics!

FLIGHT SIMULATION

From Sub-Logic
A graphic tour de force that will truly captivate you. If you haven't seen this brilliant program, you haven't fully utilized the capabilities of your computer! During FLIGHT SIMULATION, you instantly select instrument flight, radar, or a breathtaking pilot's-eye-view. But be sure to strap yourself in -- you're liable to get dizzy!

Once you put in some air time learning to fly your computer head for enemy territory and try to bomb the fuel depot while fighting off five enemy warplanes. Good Luck!

Apple II 16K tape....$25.00
Apple II 32K disk....$33.50
TRS-80 16K tape......$25.00

MICROSOFT ADVENTURE

Prepare yourself for the adventure of your life as this classic fantasy/logic game takes you into the world of the Colossal Cave. Your computer is your guide as you search for treasures, solve puzzles, explore, and avoid the dangers that lurk within.

Complete version of the original Adventure, originally written for the DEC PDP-10 in FORTRAN. The program has been translated to bring you all the enjoyment in your home computer.

TRS-80 (32K disk), Apple (32K disk) ~~$29.95~~ Special

SPECIAL SAVE $10: $19.95 thru 12/31!

DEATH-MAZE 5000

From Med Systems
A new breed of adventuring! Venture through a graphically represented 3-D maze, with halls that could dead end -- or recede to infinity. Step through the doors or drop into the pits. Will you encounter monsters and mayhem, or will you be treated to useful objects and information? Will you ever get out alive?

TRS-80 (16K tape), Apple (32K tape) $14.95
TRS-80 (32K disk) $19.95

Also Available: ASYLUM for TRS-80 (16K tape) $14.95

TRS-80® · APPLE® · ATARI®

The Program Store carries more than 1000 programs for TRS-80, Apple, and Atari. If the program you want is not listed here, call us: we probably have it in stock!

NEW

VOYAGE TO VALKYRE

By Leo Christopherson from AOS
Combine the animation and music techniques pioneered by Christopherson with the challenge of his first fast-moving arcade game and you have VOYAGE TO VALKYRE!

You speed through a magical maze guarded by ferocious birds that swoop down to attack if you don't get them first. To list all the play and options of this exciting game would take the 16 pages of instruction included.

Tape: TRS-80 (16K) $34.95
Disk: TRS-80 (16K), Apple (48K) $39.95

LUNAR LANDERS

From Adventure & Stoneware
These are realtime action simulations of a descent to the Moon. In common, they all have super graphics, realistic movement and control, and sound. The skillful are rewarded with high scores, the clumsy can look forward to spectacular crashes!

LUNAR LANDER: TRS-80 & Atari (16K), $19.95
TRANQUILITY BASE: Apple (32K disk), $24.95

FILE MANAGER 800

From Synapse Software
An extremely powerful and versatile database manager for use in both professional and personal applications. You define the format of the records to be filed and FILE MANAGER 800 gives you full control over sorting, searches, and retrieval.

You can store up to 1000 records on each disk and the program will allow up to four drives. Access to any record takes less than 1.5 seconds, and most commands can be entered with self-prompting single keystrokes.

Atari (40K disk)...$99.95

NEW

Crush, Crumble and Chomp!

From Epyx
It's a monster movie, and you are the monster! You can be The Glob, Kraken, Mantra, Mechismo, Arachnis, or Goshilla -- or even design your own "custom" monster (disk version only). This hilarious action game is loaded with graphics and sound as you practice your villany. With 6 monsters, 4 cities, and 5 game objectives, you get a choice of more than 100 possible scenarios. A monster's life is not all carnivorous crunching, though: The combined resources of the police, science, and armed forces are bent on your destruction.

TRS-80 (16K tape or 32K disk), Apple (48K disk)...$29.95
Coming soon for Atari.

GALACTIC SAGA

By Douglas Carlston from Broderbund/A.I.
Take control of the Galactica as you navigate through an uncharted 3-D universe. In GALACTIC EMPIRE, you attempt to unify a star system that is randomly created each time you play. TRS-80 (16K tape) $14.95, Apple (48K disk) $24.95, Atari (32K tape) $19.95.

GALACTIC TRADER pits your bartering skills against those of the other inhabitants as you try to accumulate riches and power. But watch out for the assassins and the energy cartel -- they're out to getcha! TRS-80 (16K tape) $14.95, Apple (48K disk) 24.95.

Diplomacy and deviousness play equal parts in GALACTIC REVOLUTION. It's a game that combines tactics, social manipulation, and Machiavellian ruthlessnes. For more intrigue, this game allows more than one player. Sound effects. TRS-80 (16K tape) $14.95, Apple (48K disk) 24.95.

TAWALA'S LAST REDOUBT makes you the rebel leader. You must intercept and decipher the messages of the dethroned Emporer Tawala in order to initiate an assault on his exile kingdom. Apple (48K disk) $24.95. Coming soon for TRS-80.

Control The Whole Universe -- Get All Three!

Visit our other stores: Seven Corners Center · Falls Church, VA & W. Bell Plaza · 6600 Security Blvd. · Baltimore, MD

TO ORDER CALL TOLL FREE 800 424-2738 — For information Call (202) 363-9797

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

MAIL ORDERS: Send check or M.O. for total purchase price, plus $1.00 postage & handling. D.C. residents, add 6% tax. Charge card customers: include all embossed information on card.

Get the most from your micro with software and accessories from one of the world's largest selections.

The Program Store

CALL TOLL FREE 800 424-2738

EPYX COMPUTER GAMES THINKERS PLAY

TEMPLE OF APSHAI
The first of the DunjonQuest series, and still one of the most popular. In exploring over 200 rooms in the magical labyrinth, you will encounter more than 30 kinds of fearsome monsters guarding over 70 treasures. Some of the treasures will help you in your quest, but you must still watch out for the many monsters and traps that spring out from the walls and shadows.
Apple (48K disk) TRS-80 (16K tape, 32K disk) $39.95

LOWER REACHES OF APSHAI: Four more levels. Requires above program. $19.95

HELLFIRE WARRIOR
Sequel to APSHAI, this thriller gives you four levels to explore with sixty rooms each. New monsters, new traps, new challenge!
Apple (48K disk), TRS-80 (16K tape, 32K disk) $39.95

KEYS OF ACHEREON: Four more levels. Requires above program. $19.95

MORLOC'S TOWER
A completely different 6-level "dunjon" hides treasures and trouble. Defeating the dreaded Morloc takes wit and determination.
Apple (32K tape, 48K disk), TRS-80 (16K tape, 32K disk), Atari (32K tape or disk) $19.95

DATESTONES OF RYN
Rex the Reaver and his band of 24 cutthroats have made off with the treasured datestones. They're hiding in an labyrinth of caves in the Haunted Mountains. It is your job to find them before they find you.
Apple (32K tape, 48K disk), TRS-80 (16K tape, 32K disk), Atari (32K tape or disk) $19.95

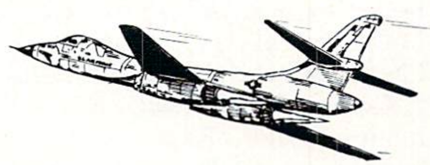
TUESDAY MORNING QUARTERBACK
An amazingly realistic football game that lets you captain the team of your choice against the team you choose for the computer. 96 possible offensive/defensive combinations are available. A database includes the teams' rosters as well as their strengths and weaknesses -- just like the real thing!
Apple (48K disk), TRS-80 (48K disk) $29.95

STARQUEST

RESCUE AT RIGEL
First of the "Starquest" series, it's an interplanetary adventure. The evil High Tollah has captured ten prisoners; if you don't find and save them they will be transformed into enemy aliens -- and set out after you!
Apple (32K tape, 48K disk), TRS-80 (16K tape, 32K disk), Atari (32K tape or disk) $19.95

STAR WARRIOR
You're on your own, light years from Earth in intergalactic space. If you thought saving the princess in RESCUE AT RIGEL was a challenge, wait 'til you pit yourself against STAR WARRIOR! Order the second edition of the Starquest Series now.
Apple (32K tape, 48K disk), TRS-80 (16K tape, 32K disk), Atari (32K tape or disk) $19.95

MICROCOMPUTER GAMES



From Avalon Hill
Fine war and strategy games for the home computer that pit your skill against the program. 16K tape unless otherwise noted.

MIDWAY CAMPAIGN - relive the battle as you control our naval forces. TRS-80, Apple, Atari (32K) $14.95
B-1 NUCLEAR BOMBER - avoid MiGs & missiles as you pilot this advanced aircraft. TRS-80, Apple, Atari (16K) $14.95
NORTH ATLANTIC CONVOY RAIDER - lets you simulate the Bismark convoy raid of 1941. TRS-80, Apple, Atari (16K) $14.95
NUKE WAR - choose espionage or arms buildup to control a nuclear confrontation. TRS-80, Apple, Atari (16K) $14.95
PLANET MINERS - one to four players compete, staking claims in the solar system. TRS-80, Apple, Atari (24K) $14.95
CONFLICT 2500 - star battle in the 26th century for up to 10 players. TRS-80, Apple, Atari (32K) $14.95
LORDS OF KARMA - adventure in a land where good deeds score more than treachery. TRS-80, Apple, Atari (40K) $20
TANKTICS - combines gameboard & computer in a strategic WWII tank battle. TRS-80, Apple, Atari (24K) $23.95
MAJOR LEAGUE BASEBALL - play solitaire or head-to-head, using names & statistics of real players. Apple (48K disk) only, $24.95

THE PROGRAM STORE
Franchises Available

SCOTT ADAMS'

ADVENTURE

By Scott Adams from Adventure International
Twelve different adventures make up this acclaimed series. Written in machine language for fast response, they support lower case (if installed) and have over 100 words in their vocabularies.

Until you've played an Adventure, you can't appreciate the hours of challenge and fun built into each program. Each tests your powers of reason and deduction as you attempt to accomplish your mission using the implements you have, find or devise. Tape for TRS-80 16K, Apple 24K, Atari 24K (Adventures 1-9 only for Atari).

1. ADVENTURELAND - caves, pits, magic words and the dragon... $19.95
2. PIRATE'S ADVENTURE - go from your London flat to Treasure Island... $19.95
3. MISSION IMPOSSIBLE - complete your mission or the reactor is doomed... $19.95
4. VOODOO CASTLE - Save Count Cristo from his fiendish curse... $19.95
5. THE COUNT - when you awaken in Transylvania, watch your neck... $19.95
6. STRANGE ODYSSEY - find alien treasures at the edge of the galaxy... $19.95
7. MYSTERY FUN HOUSE - try to leave the most bizarre funhouse you've seen... $19.95
8. PYRAMID OF DOOM - search the labyrinth for ancient treasures... $19.95
9. GHOST TOWN - just because something moves doesn't mean it's not dead... $19.95
10. SAVAGE ISLAND #1 - first part of a new series for advanced adventurers...$19.95
11. SAVAGE ISLAND #2 - part two: requires the purchase of part one (above)...$19.95
12. GOLDEN VOYAGE - you have just 3 days to rejuvenate the near-dead king...$19.95

Three Adventures on disk (TRS-80 32K, Apple 48K), choose:
1-3 4-6 7-9 10-12 $39.95 per disk

SARGON II

By Dan & Kathe Spraklen from Hayden
Acclaimed the best of the microcomputer chess programs. SARGON II came in third in the 9th North American Computer Chess Championship, playing against much bigger machines! You haven't really played chess against your computer until you've tried this brilliant program.

Tape: TRS-80 (16K), Apple (24K) $29.95
Disk: TRS-80 (32K), Apple (24K) $34.95

Prices Subject to Change

THE PROGRAM STORE · Dept UH Box 9609 · 4200 Wisconsin Ave, NW · Washington, D.C. 20016

Item ____ Price ____ Postage $1.00 Total ____ name ____ addr ____
□ CHECK □ VISA city ____ state __ zip ____
□ MASTERCARD MC Bank # ____ Card # ____ Exp ____

of Dakin5 Corporation. The Depreciation Planner and The Business Bookkeeping System are trademarks of Dakin5 Corporation. The Controller is marketed by Apple Computer, Inc. Apple is a registered trademark of Apple Computer, Inc.

80 Column Adaptor

EXECOM CORP. announced a new product for Commodore Business Machines' PET/CBM series computers. The product is an 80 column adaptor, circuit board and ROM combination, that allows the user to switch between the original, 40 column display, and the new 80 column display, from the keyboard, or through program control. Price of this modification is $275.00 plus installation. This circuit board and ROM combination is for the PET/CBM computers that do not have a CRT, or display controller chip, in the 2000/3000/4000 series models, which are designed for version 3.0, or 4.0 Basic. The installation involves cutting circuit traces and installing 4 sockets, thus making it a product that should be purchased from a dealer, if you are not technically oriented. EXECOM CORP. offers the installation for $75.00. This requires the user to send in their computer circuit board for modification.

For further information contact Execom Corp., 1901 Polaris Ave., Racine, WI 53404.

Information Systems Education To Be DPMAEF Conference Theme

Chicago: The Education Foundation (EF) of the Data Processing Management Association (DPMA) has announced that it will sponsor a National Conference on Information Systems Education to be held March 22-24, 1982 in Chicago. The meeting will consist of one-day in-depth Workshops followed by a two-day general Conference.

On the first day, invited experts will conduct special Workshops in major areas of importance in Information Systems Education. The Conference, which begins on the second day, will deal with issues of practical importance to the providers, recipients, and end users of Information Systems Education. It will provide comprehensive and authoritative updates on the DPMA Model Curriculum, reports on practical experiences in implementing Information Systems Education, and approaches to overcome financial and other obstacles to implementing such programs.

The DPMAEF was established in 1975 by the Association for the purpose of expanding educational opportunities for systems professionals and to conduct research and programs of benefit to the DP industry,

commodore spectacular

8032-32K 80 COL CRT
REG. $1495 **$1095**

4032-32K 40 COL CRT
REG. $1295 **$995**

4016-16K 40 COL CRT
REG. $995 **$795**

8050-DUAL DISK 950K
REG. $1795 **$1345**

4040-DUAL DISK 343K
REG. $1295 **$995**

C2N-CASSETTE DRIVE
REG. $95 **$69**

4022-80 COL PRINTER
REG. $795 **$629**

8024-MANNSMAN TALLEY
REG. $1995 **$1595**

25CPS-STARWRITER
REG. $1895 **$1445**

CBM-IEEE MODEM
REG. $279 **$229**

VOICE SYNTHESIZER
REG. $395 **$329**

VIC 20 ★★★★
REG. $299 **$269**

PET TO IEEE CABLE
REG. $39.95 **$34**

IEEE TO IEEE CABLE
REG. $49.95 **$39**

SOFTWARE

	LIST	YOU PAY
OZZ—The Information Wizard	$395	$299
Wordcraft 80	$395	$299
IRMA—Info Retrieval & Mgmt. Aid	$495	$399
Dow Jones Portfolio Mgmt.	$149	$119
Pascal Development Pkg.	$295	$229
EBS—Receivables, Inventory	$750	$579
BPI—General Ledger	$395	$299
Word Pro 3—40 Column	$250	$179
Word Pro 4—80 Column	$375	$269
Word Pro 4 Plus	$450	$329

CUSTOM PRINTED CONTINUOUS MINI-FORMS

- LOOK PROFESSIONAL
- LOW PRICES FOR SMALL RUNS

1000-11"x9½" BLACK

PARTS	LETTERHEAD	INVOICE	STATEMENT
1	37.50	—	37.50
2		63.50	63.50
3		79.50	
5		115.50	

- SPECIAL LOGOS, 2 COLOR PRINTING & SPECIAL STOCK AVAILABLE

MINI-FORMS PRICES & ORDERS:
(800) 523-3442
PA & West of Rockies (215) 455-9705

Main Line Computer Discount

P.O. BOX 170
ST. DAVIDS, PA 19087

MAIL ORDER PHONE
800-345-8112
IN PA CALL 1-800-662-2444

- PREPAID ORDERS SHIPPED FREE
- VISA & MASTERCARD ADD 3%
- COD-UPS
- PA RESIDENTS ADD 6% PA TAX

educators, business, government, and the public.

DPMAEF has issued a "Call for Papers" for this Conference. For a copy of the "Call" or for further information on the Conference, contact the Conference Manager, USPDI, 12611 Davan Drive, Silver Spring, MD 20904. Telephone (301)622-0066.

Low Cost Floppy Disk For Rockwell AIM

The PEDISK II Floppy Disk System from CGRS Microtech, Inc. is now available for the Rockwell AIM microcomputer. This system, originally developed for the Commodore PET, now provides the AIM owner with a high performance mass storage peripheral. It is available with either 5¼" or 8" drives with a storage capacity to 858 Kbytes in a three drive system. The 8" drive offers standard IBM 3740 compatibility allowing data exchange with most microcomputers and mini-computers.

A new software package, ADOS, provides a full set of disk utilities including format, copy, display, patch and directory. Additional disk commands allow the user to load and save programs, data files and source files. A convenient user interface is provided to utilize the disk system in custom software. Several software tools are available for use with the AIM/PEDISK. These include full FORTH+, a versatile high level language, a Macroassembler/Editor for machine language programs and a full BASIC interface to allow the disk to be used with AIM BASIC.

The single drive dual density Model 540-1 offers 143 Kbytes of storage and sells for $595.00. The dual drive quad density Model 580-2 offers 572 Kbytes of storage and sells for $1195.00. For additional information, contact your dealer or CGRS Microtech, P.O. Box 102, Langhorne, PA 19047. 215-757-0284.

Hi-Res Graphics For Atari 400/800 Computers

Newbury Park, CA — Versa Computing, Inc. announces the release of *GRAPHICS COMPOSER*, the complete joystick/paddle graphics software package for ATARI 400/800 Computers.

With GRAPHICS COMPOSER, you use paddles or joystick to draw a picture outline on hi-res screen Mode 8 or 7. Then use color fill-in, color brushes and add Text to complete your graphic designs. *Save your graphics to disk or cassette.*

GRAPHICS COMPOSER allows easy creation of PLAYER/MISSILE shapes which may be used in other programs. The

we beat the price...

Exclusive One Year Warranty Included

800™ $779

WITH ONE YEAR EXTENDED WARRANTY

ATARI®
Computers for people.™

ATARI 810 DISC DRIVE

$444

Available without warranty for even less.

ATARI SOFTWARE

CX401 General Accounting	$399.00
CX402 Accounts Receivable	$399.00
CX403 Inventory Control	$399.00
CX404 Word Processor	$119.00
CX405 PILOT	$68.00
CX413 MICROSOFT BASIC	$68.00
CX4101 Invitation to Programming 1	$17.00
CX4102 Kingdom	$13.00
CX4103 Statistics	$17.00
CX4104 Mailing List	$17.00
CX4105 Blackjack	$13.00
CX4106 Invitation to Programming 2	$20.00
CX4107 Biorhythm	$13.00
CX4108 Hangman	$13.00
CX4109 Graph It	$17.00
CX4110 Touch Typing	$20.00
CX4111 Space Invaders	$17.00
CX4112 States & Capitals	$13.00
CX4114 European Countries & Capitals	$13.00
CX4115 Mortgage & Loan Analysis	$13.00
CX4116 Personal Fitness Prog	$59.00
CX4117 Invitation to Programming 3	$20.00
CX4118 Conversational French	$45.00
CX4119 Conversational German	$45.00
CX4120 Conversational Spanish	$45.00
CX4121 Energy Czar	$13.00
CX4125 Conversational Italian	$45.00
CX6001 U.S. History	$23.00
CX6002 U.S. Government	$23.00
CX6003 Supervisory Skills	$23.00
CX6004 World History	$23.00
CX6005 Basic Sociology	$23.00
CX6006 Counseling Proced	$23.00
CX6007 Principal of Act	$23.00
CX6008 Physics	$23.00
CX6009 Great Classics	$23.00
CX6010 Business Comm	$23.00
CX6011 Basic Psychology	$23.00
CX6012 Effective Writing	$23.00
CX6014 Principals of Econ	$23.00
CX6015 Spelling	$23.00
CX6016 Basic Electricity	$23.00
CX6017 Basic Algebra	$23.00
CX8106 Bond Analysis	$20.00
CX8107 Stock Analysis	$20.00
CX8108 Stock Charting	$20.00
CXL4001 Education System Master	$21.00
CXL4002 Basic Computing Language	$46.00
CXL4003 Assembler Editor	$46.00
CXL4004 Basketball	$24.00
CXL4005 Video Easel	$24.00
CXL4006 Super Breakout	$30.00
CXL4007 Music Composer	$45.00
CXL4009 Chess	$30.00
CXL4010 3-D Tic-Tac-Tow	$24.00
CXL4011 Star Raiders	$32.00
CXL4012 Missle Command	$32.00
CXL4013 Asteroids	$32.00
CXL4015 TeleLink	$20.00
Compuhome	$74.95
Visicalc	$149.00
Letter Perfect (Word Processor)	$119.00
Source	$89.00

Atari® Peripherals:

400 16K	$329.00
410 Recorder	$59.00
822 Printer	$359.00
825 Printer	$ CALL
830 Modem	$159.00
850 Interface	$ CALL

Atari® Accessories

New DOS 2 System	$21.00
CX70 Light Pen	$64.00
CX30 Paddle	$18.00
CX40 Joy Stick	$18.00
CX853 16K RAM	$89.00
Microtek 16K RAM	$75.00
Microtek 32K RAM	$169.00

CBM 8032 $1099

4016	$799.00
4032	$999.99
8096	$1795.00
CBM4022 Printer	$629.00
Tally 8024	$1699.00
CBM C2N Cassette Drive	$69.00
CBM4040 Dual Disk Drive	$999.00
CBM8050 Dual Disk Drive	$1349.00

commodore

VIC 20 $259

Vic-TV Modual	$19.00
Vic Cassette	$69.00
Vic Disk Drive	$ Call
Vic 6 Pack program	$44.00

CBM Software

WordPro3 Plus	$199.00
WordPro4 Plus	$299.00
Commodore Tax Package	$399.00
Visicalc	$149.00
EBS Accts. Rec./Inventory Interactive Syst	$595.00
BPI General Ledger	$329.00
OZZ Information System	$329.00
Dow Jones Portfolio	$129.00
Pascal	$239.00
Legal Time Accounting	$449.00
World Craft 80	$289.00
Word Check	$180.00
Create-A-Base	$219.00
Power	$89.00
Socket-2-Me	$20.00
Jinsam	Call

Disks

CX8100 Blank Disk (5)	$22.00
Sycom Blank Disk (10)	$29.00
Maxell Blank Disk (10)	$36.00
Maxell Blank Disk (10)	$46.00

Printers

Epson MX-70	Call for Prices
Epson MX-80	Call for Prices
Epson MX-80 FT	Call for Prices
Diablo 630	Call for Prices
Nec 5530	$2495.00
TEC 1500 Starwriter 25cps	$1495.00
TEC 1500 Starwriter 45cps	$1795.00

No Risk, No Deposit On Phone Orders, COD or Credit Card, Shipped Same Day You Call *

* on all in stock units

IN PA, CALL (717) 327-9575

(800) 233-8950

To Order:

Phone orders invited (800 number is for order desk only). Or send check or money order and receive free shipping. Pennsylvania residents add 6% sales tax. Add 3% for Visa or M.C. Equipment is subject to price change and availability without notice. Please call between 11 AM & 6 PM.

COMPUTER MAIL ORDER
501 E. 3RD ST., WILLIAMSPORT, PA 17701

GEOMETRIC FIGURES program lets the user define circles, triangles, polygons, parallelograms, and even trigonometric curves!

Loading routines are provided so that pictures can be used in other programs or traded with friends.

GRAPHICS COMPOSER is available on cassette or disk for only $39.95, and requires 32K RAM. Call Versa Computing at (805) 498-1956, or ask your local computer retailer for GRAPHICS COMPOSER.

Hayden Announces DENTISTAID™

DENTISTAID, the lowest priced dental office management program on the market is designed to streamline all major time-consuming tasks performed in the dental office.

This menu-oriented program will automatically print standard ADA insurance forms, prequalification and actual services, monthly statements, patient recall notices, accounts receivable aging reports, daily summary of work performed and payments received, daily, weekly, monthly, and yearly totals, production analysis using standard ADA insurance forms, prequalification and actual services, monthly statements, patient recall notices, accounts receivable aging reports, daily summary of work performed and payments received, daily, weekly, monthly, and yearly totals, production analysis using standard ADA categories and display of individual accounts.

One of the primary advantages of DENTISTAID is the simplicity of operation used to increase office efficiency through automatic generation of many forms and reports that give the dentist better practice control.

With many systems installed in the past two years, this error-free, operator-proof program can be installed in a dental office in two days.

DENTISTAID is a trademark of the Hayden Book Company, Inc., located at 50 Essex Street, Rochelle Park, NJ 07662.

New Books

Experiments in Artificial Intelligence for Small Computers,
by John Krutch
Howard W. Sams & Co.
112 pages, $8.95 softbound

This new book is a resource for anyone wanting to conduct interesting and exciting experiments on the small computer.

The author presents programs written in Microsoft's Level II BASIC that can be easily converted to most BASIC dialects.

The book begins with an explanation of artificial intelligence — its scope and problem areas. A short BASIC program that involves moving a chess king on a small chessboard is used to illustrate the discussion. Game-playing programs, with checkers as a major topic, are presented in the second chapter.

Problem-solving is covered with emphasis on a program that predicts a human player's choices. The author then focuses on reasoning, primarily by means of a program which stores data and makes deductions from this data. Also, creativity is treated in the form of computer poetry and computer-generated prose.

The book devotes a chapter to natural-language processing or verbal communication. It is illustrated by the program DOCTOR, which simulates a psychiatrist's counsel.

An appendix describes BASIC keywords to help make the book's programs more easily translated to the version of BASIC to which the user is accustomed.

Your Own Computer (2nd ed.)
by Mitchell Waite & Michael Pardee
Howard W. Sams & Co.
224 pages, $7.95 softbound

In this book, authors Mitchell Waite and Michael Pardee have removed much of the complexity and mystery that surrounds the microcomputer and have succeeded in producing a simple, easy-to-understand book about these amazing devices.

This new edition provides the newcomer with the knowledge and confidence needed to use today's personal computer. The text has been updated and explained to include a chapter that objectively compares 30 popular personal and small business computers now on the market.

What are personal computers? Where did they come from? What's coming in 1990? An introductory chapter provides these answers for you.

Personal computing applications for your home, office, or classroom are explained. The chapter on programming has been expanded in this edition to help you understand the heavy emphasis on computer software.

To help you better understand computer concepts, a glossary of the most often encountered buzz words, complete with definitions and a practice sentence using the word, is presented in Chapter Two. Also included is a list of key acronyms.

The book presents the nuts and bolts — from input/output units and peripherals to buses and memories — that make up the personal computer.

The authors give tips on how you can get started in computing for your home, business, or hobby. An appendix on computer number systems is included for your reference.

Webster's Microcomputer Buyer's Guide
by Tony Webster
Hayden Book Co.
326 pages, $25.00 peperback

This is a detailed reference guide listing approximately 113 private vendors for microcomputer users.

The book's 16 chapters contain four parts: Theory and Application, Independent Software Vendors, Microcomputers and Microcomputer Systems and CRT Displays, Printers and Printing Terminals.

Part I introduces ten chapters, the first five concerning theory and application. Also, outlined are the differences between microprocessors, microcomputers and microcomputer systems. The remaining five chapters examine the future potential of microcomputers, provide guidelines for selecting microcomputers and the capabilities of microcomputers in word

Lyco Computer Marketing & Consultants

We Specialize in Quality, Knowledge, Service, and Microcomputers

717 - 435-5197

ATARI

800 16K	$744.00
400 16K	$329.00

ATARI HARDWARE

800	$744.00
400	$329.00
410 CASSETTE RECORDER	$57.00
810 DISK DRIVE	$444.00
850 INTERFACE	$159.00
825 PRINTER	$525.00
830 PHONE MODEM	$135.00

PROGRAMMING SOFTWARE

CX4101 INVITATION TO PROG.	$17.00
CX4106 INVITATION TO PROG. 2	$19.00
CX4117 INVITATION TO PROG. 3	$19.00
CXL4003 ASSEMBLER EDITOR	$42.00
CXL4002 ATARI BASIC CART	$42.00
CXL4015 TELELINK	$19.00

EDUCATIONAL SOFTWARE

CXL4001 EDUCATIONAL SYSTEM	$19.00
CX 6001 U. S. HISTORY	$22.00
CX 6002 U. S. GOVERNMENT	$22.00
CX 6003 SUPERVISORY SKILLS	$22.00
CX 6004 WORLD HISTORY	$22.00
CX 6005 BASIC SOCIOLOGY	$22.00
CX 6006 COUNSELING PROCED.	$22.00
CX 6007 PRINCIPLES OF ACCT.	$22.00
CX 6008 PHYSICS	$22.00
CX 6009 GREAT CLASSICS	$22.00
CX 6010 BUS. COMMUNICATION	$22.00
CX 6011 BASIC PSYCHOLOGY	$22.00
CX 6012 EFFECTIVE WRITING	$22.00
CX 6014 PRINCIPLES OF ECON.	$22.00
CX 6015 SPELLING	$22.00
CX 6016 BASIC ELECTRICITY	$22.00
CX 6017 BASIC ALGEBRA	$22.00
CX 4108 HANGMAN	$12.00
CX 4112 STATES & CAPITALS	$12.00
CX 4114 EUROPE COUNT. & CAP.	$12.00
CX 4121 ENERGY CZAR	$12.00
CX 4123 SCRAM	$12.00
CX 4102 KINGDOM	$12.00

ENTERTAINMENT

CXL4004 BASKETBALL	$29.00
CXL4005 VIDEO EASEL	$29.00
CXL4006 SUPER BREAKOUT	$29.00
CXL4009 CHESS	$29.00
CXL4010 3-D TIC-TAC-TOE	$29.00
CXL 4011 STAR RAIDERS	$30.00
CXL 4012 MISSILE COMMAND	$30.00
CXL4013 ASTEROIDS	$30.00
CX 4015 BLACKJACK	$12.00
CX 4111 SPACE INVADERS	$16.00

ACCESSORIES

CX 30 PADDLES	$17.00
CX 40 JOYSTICKS	$17.00
CX853 16K RAM	$85.00
****** 32K RAM by MICROTEK	$169.00

WE CARRY MANY OTHER LINES OF MICROCOMPUTERS YOU CAN CALL FOR PRICES ON:

CPU's	PRINTERS
COMMODORE	TEC
ALTOS	DIABLO
TEXAS INSTRUMENT	HEC
ZENITH	EPSON
INERTEC DATA	ANADEX
AND OTHERS	AND OTHERS

At Lyco Computers we offer our expert services to help customers make their first computer purchase schools establish a computer program, or evaluate multiterminal systems.

CALL US AT 717 - 435-5197

to help evaluate your needs, or if you wish to make a purchase.

Lyco Computers
P.O. Box 10
COGAN STATION, PA 17728

Note the prices in this advertisement are subject to change.

processing and education.

Part II, covering Chapters 11 and 16, provides descriptions of many software operating systems, utilities, languages and application packages from independent vendors. Part III provides a summary of many major microcomputer systems available, under the headings, Overview, Central Processing Systems, Peripherals, Software, Pricing and Head Office. Part IV lists in detail the specifications of some of the popular CRT displays, printers and printing terminals available for connection to microcomputers.

Learning With Computers
by Alfred Bork
Digital Press
304 pages, $25.00 hardbound

Learning With Computers is a key publication for anyone interested in designing, using, or studying the computer as an educational tool. It focuses specifically on using the computer to build a more responsive learning environment.

An original collection of articles, *Learning With Computers* offers a comprehensive overview for using the computer as an interactive teaching and learning device. Written in a non-technical style, it covers such crucial topics as:

- Using programs prepared by others
- Personal computers versus time-sharing systems
- Types of hardware and the efficient use of hardware
- How computer graphics can assist learning
- Authoring graphic dialogs
- Attitudes towards learning simulations
- On-line tests and quizzes
- Motivation for computers in the classroom
- Conversational language criteria
- Computer availability
- Self-pacing and student choice
- Course management

Instant (Freeze-Dried Computer Programming in) BASIC
by Jerald Brown
Dilithium Press
200 pages, $10.95

The 2nd Edition keeps the style and flavor of the 1st edition. BASIC, a computer programming language, is favored by most beginners because it is easy to learn and use, and because of the proliferation of programs written in BASIC. The author, Jerry Brown, appreciates who his audience is (the absolute beginner) and writes specifically for them.

The 2nd Astounding! Edition has more annotations, additions activities and text, and has been updated. In the tradition of the first edition:

- It is microcomputer oriented for Microsoft-like versions of BASIC as used on the Apple, TRS-80, PET, Microexpander, and any brand of computer using Microsoft BASIC 80, with annotations for Northstar BASIC, Atari BASIC, and DEC BASIC Plus.
- There is no heavy math.
- There are neat little boxed summaries of BASIC statements throughout the book.
- There are end-of chapter activities to see how well the reader is learning BASIC.
- It is full of graphics to relieve boredom and stimulate interest.

Instant BASIC is for every beginner, and is especially suited for the educational market.

File System For Floppy Disks Described In Brochure

Los Altos, CA — Amaray Corporation announces the immediate availability of a four-page, two-color brochure describing its DiskBank floppy disk filing and storage system.

Available for either 5¼" or 8" floppy disks, each DiskBank unit holds ten floppy disks in a dust-free, secure environment. The DiskBank's interlocking, modular design allows the user to structure the system horizontally or vertically with maximum space efficiency.

The DiskBank's design allows it to be used either on a desk top or in a file drawer application.

Contact Rich Koch at Amaray Corp., 2251 Grant Road, Suite H, Los Altos, CA 94022 (415) 968-2840.

Low Power IEEE-488 Controller/Talker/ Listener Module Plus Custom I/O

Ontario, CA, September 29, 1981 — A General Purpose Interface Bus (GPIB) module which meets all IEEE-488 specifications, with control software, a wire-wrap section for custom input/output, operating on a single +5V DC power source, is now available from General Micro Systems, Inc.

The module, GMS6514, is fully compatible with the Motorola EXORciser/ Micro Module and Rockwell AIM 65 and SYSTEM 65 bus structures. It has an on-board device address switch and a special GPIB connector to allow bus control with minimum effort. Using low power Schottky technology, the module also offers over voltage and reverse polarity protection.

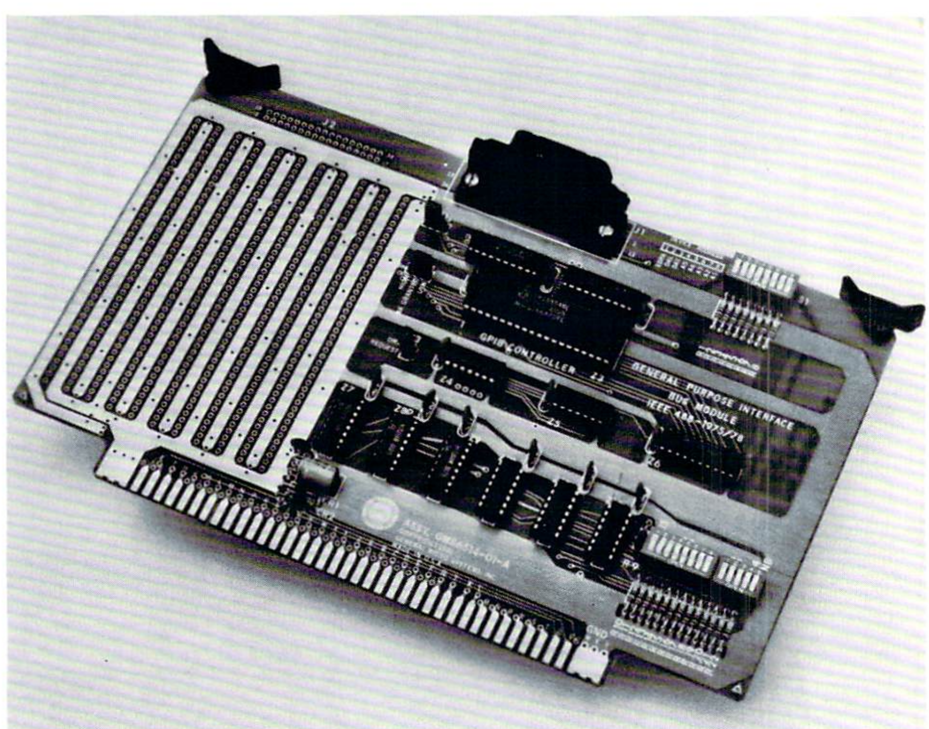
The GMS6514 is designed with a TMS9914 LSI bus controller device from Texas Instruments, which meets all specifications for signal levels and timing for IEEE-488, 1975/78 standards. It has pass control and system control capabilities with device clear and trigger functions, parallel and serial poll, service request and remote/local selection with local lockout.

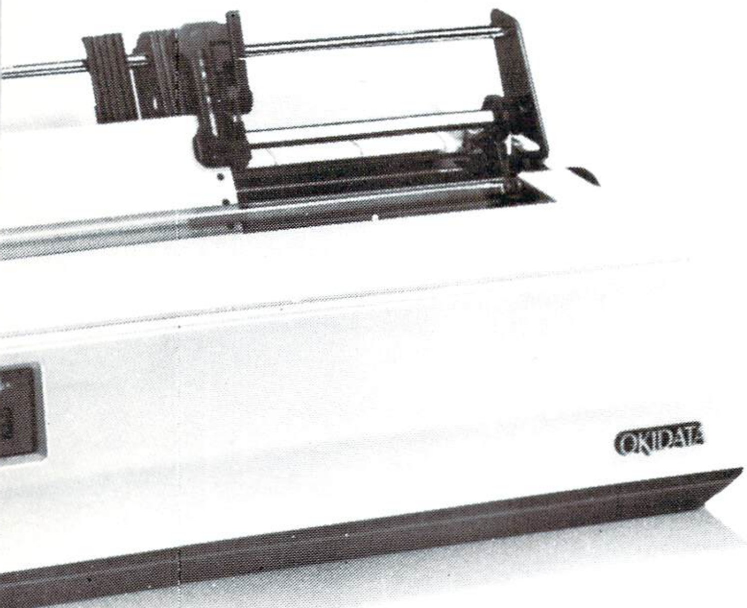
The board has base address and enable/disable switches. A device address switch, with secondary addressing capability, is accessible from the top of the module.

A large wire-wrap section for custom interface circuitry has provisions for a 34-pin connector. A software package, written for the Rockwell SYSTEM 65 and AIM 65, gives complete IEEE-488 bus control.

With a 72-hour burn-in and a full year warranty, the GMS6514 module is $250 in single piece quantity. Available off-the-shelf in the US, Canada, South America and Europe, normal quantity discounts apply.

General Micro Systems, Inc., located at 1320 Chaffey Ct., Ontario, Calif. 91726, (714) 621-7532, designs and manufactures a family of microcomputer modules and systems directly compatible with the 6500/6800 bus, plus associated software. ©





$749

CBM 8032 COMPUTER	**$ 1149**
CBM 8050 DISK DRIVE	1349
CBM 4032 COMPUTER	1029
CBM 4040 DISK DRIVE	1029
CBM 402	649
CBM VIC-20	269
LEEDEX / AMDEK 100	**139**
LEEDEX / AMDEK 100G	169
LEEDEX / AMDEK COLOR-1 13" Color Monitor	329
MICROTEK 16K RAMBOARD for Atari	**79**
MICROTEK 32K	149
ATARI 800 16K	**749**
ATARI 400 16K	349
ATARI 810 Disk Drive	449
ATARI 820 40 Column Printer	299
ATARI 822 40 Column Thermal Printer	349
ATARI 825 80 Column Printer	599

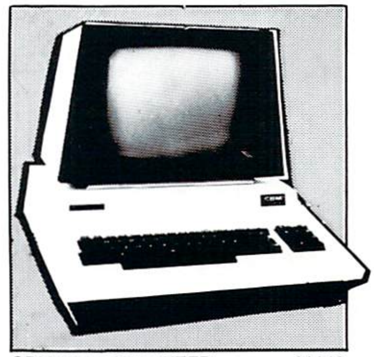
OMEGA SALES CO.

EAST COAST
1-800-556-7586
OMEGA SALES CO.
12 Meeting St.
Cumberland, RI 02864
1-401-722-1027

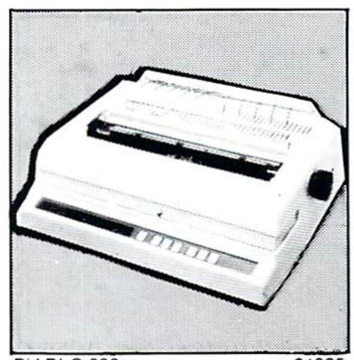
WEST COAST
1-800-235-3581
OMEGA SALES CO.
3533 Old Conejo Rd. #102
Newbury Park, CA 91320
1-805-499-3678
CA. TOLL FREE 1-800-322-1873

OMEGA SALES CO.

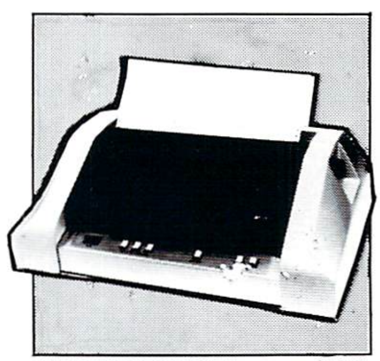
CHRISTMAS SPECIALS!



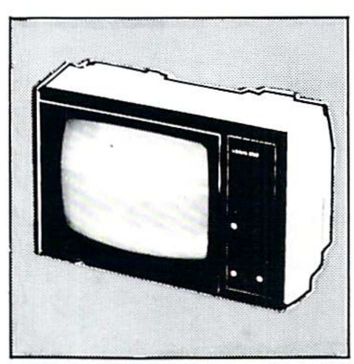
CBM 8032 COMPUTER $1149



DIABLO 630 $1995



ANADEX DP-9500/9501 $1249



LEEDEX/AMDEK COLOR-1 13" $ 329

Advertiser's Index

BEFORE YOU BUY A FINE PRINTER, READ THE FINE PRINT.

That's where you'll find the prices—in the tiny type under the brief descriptions.

Which is only appropriate. Because with C. Itoh's full line—from the workhorse business printers to the racehorse word-processing printers—the descriptions could be much bigger.

But the prices couldn't be much smaller.

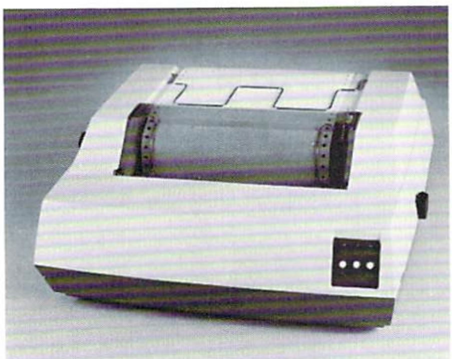
Moreover, as part of the Leading Edge Inventory Bank,™ our printers (along with dozens of other popular computer products) are available for 24-hour "withdrawal." Within a day of when we receive your order, your order will be on its way from our nearest warehouse (Boston, New York, Los Angeles, etc.), thus saving you both time and freight. So you can use our space, and our money, to inventory your product, and still be sure of having exactly what you need, exactly when you need it.

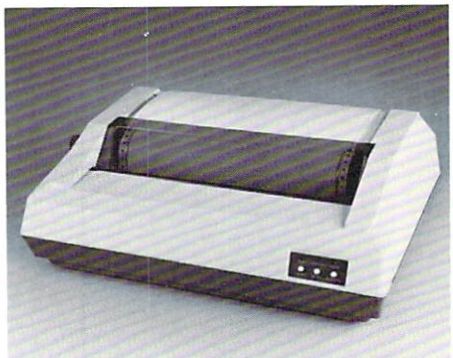
All the printers on this page are warranteed by Leading Edge™ for 3 months on parts and labor.

For the name of your nearest dealer—just pick up the nearest telephone.

TOLL-FREE 1-800-343-6833.

In Massachusetts, call collect (617) 828-8150.



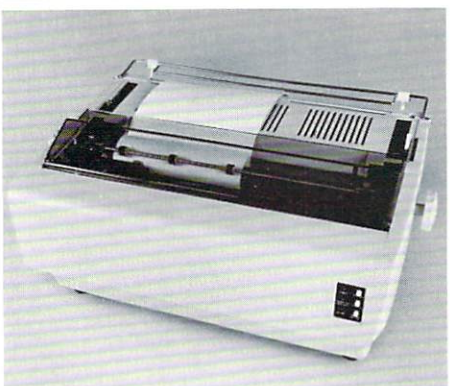


Big business
(Comet II)

The first 15" printer to retail for under a grand. Full 136 columns for standard-width accounting reports, giving you easy-to-read 10 cpi printout up to 66 lines deep. And, like with the Comet I, ribbons are available everywhere. Retail: about $995.

Beyond business
(Starwriter I)

Lowest-priced letter-quality printer on the market. Comes complete and ready to use, requiring no changes in hardware or software. Industry-standard daisy wheels and ribbon cartridges. Plug-in compatible with a wide variety of systems. 25 cps; 163 columns; precise character placement; full graphics capability. Retail: about $1,895.



Busy business
(Starwriter II)

The racehorse of letter-quality daisy wheel printers. Typical 45 cps throughput is about equal to (or, in bi-directional mode, actually *faster* than) "thimble" printers rated at 55 cps. Like the Starwriter I, it uses easily available daisy wheels and ribbon cartridges, and it's compatible with software packages, like Wordstar and Magic Wand, programmed for systems like the Qume Sprint 5. Retail: about $2,395.

LEADING EDGE.™

Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021.

DEALERS: For immediate delivery from the Leading Edge Inventory Bank,™ call toll-free, **1-800-343-6833.**
In Massachusetts, call collect (617) 828-8150, Telex 951-624.

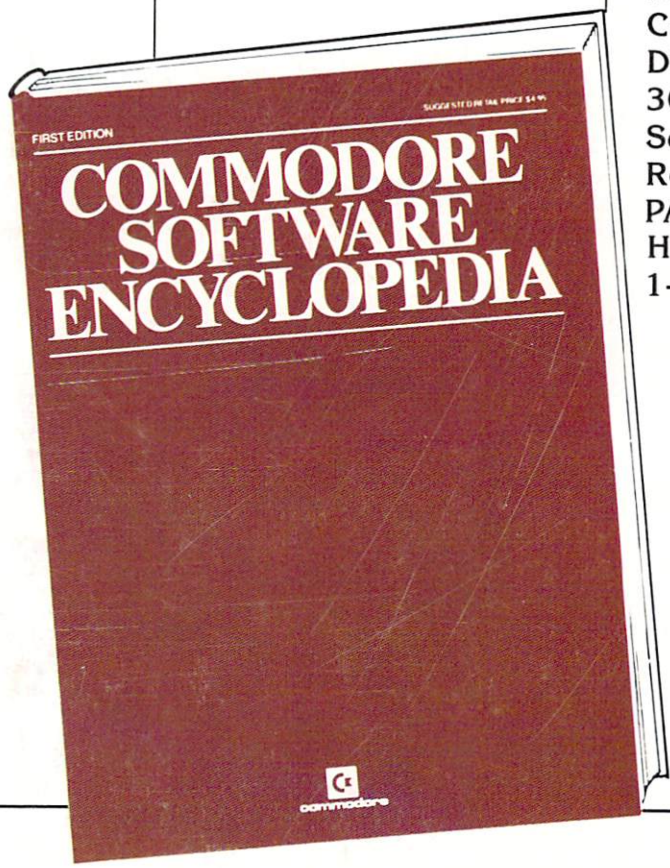
Commodore Software Encyclopedia

The COMMODORE SOFTWARE ENCYCLOPEDIA is now available from your authorized COMMODORE Dealer, for . . .

$4.95

The next edition of the COMMODORE SOFTWARE ENCYCLOPEDIA will be available in 90 to 120 days. If your software is not listed and you would like to have it listed in the next edition, please submit details to:

Software Department
Commodore Business Machines, Inc.
Computer Systems Division
300 Valley Forge Square, 681 Moore Road, King of Prussia, PA 19406.
HOTLINE Number
1-800-523-5622



commodore