Atari:
The Fluid Brush

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

Kids For
Computers

COMPUTE!

$2.50
January,
1980
Issue 8
Vol. 3, No. 1
63354

The Journal For Progressive Computing™

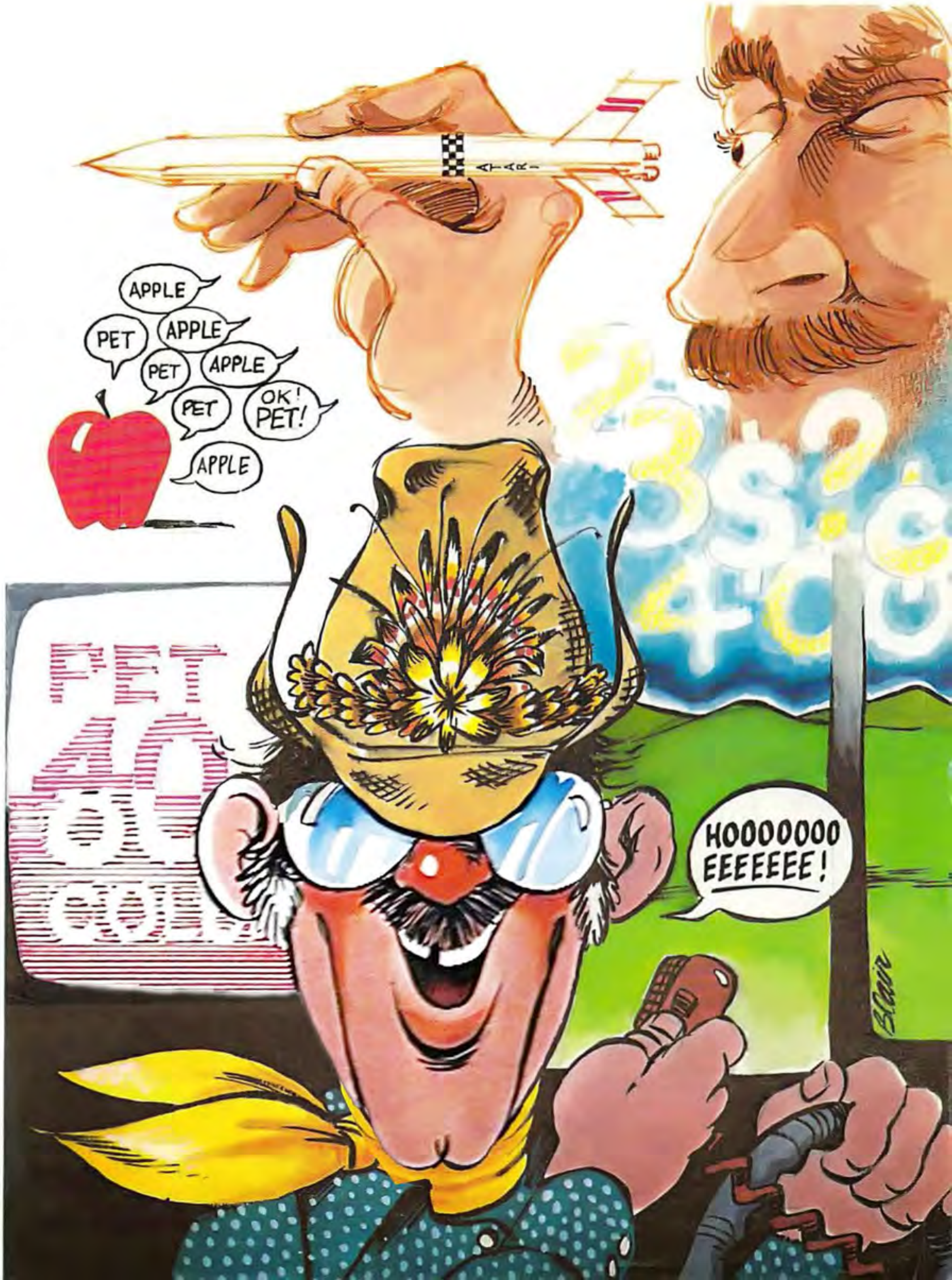
Player Missle Graphics On The Atari

Load PET Tapes To Your Apple II

Financial Fuzzies

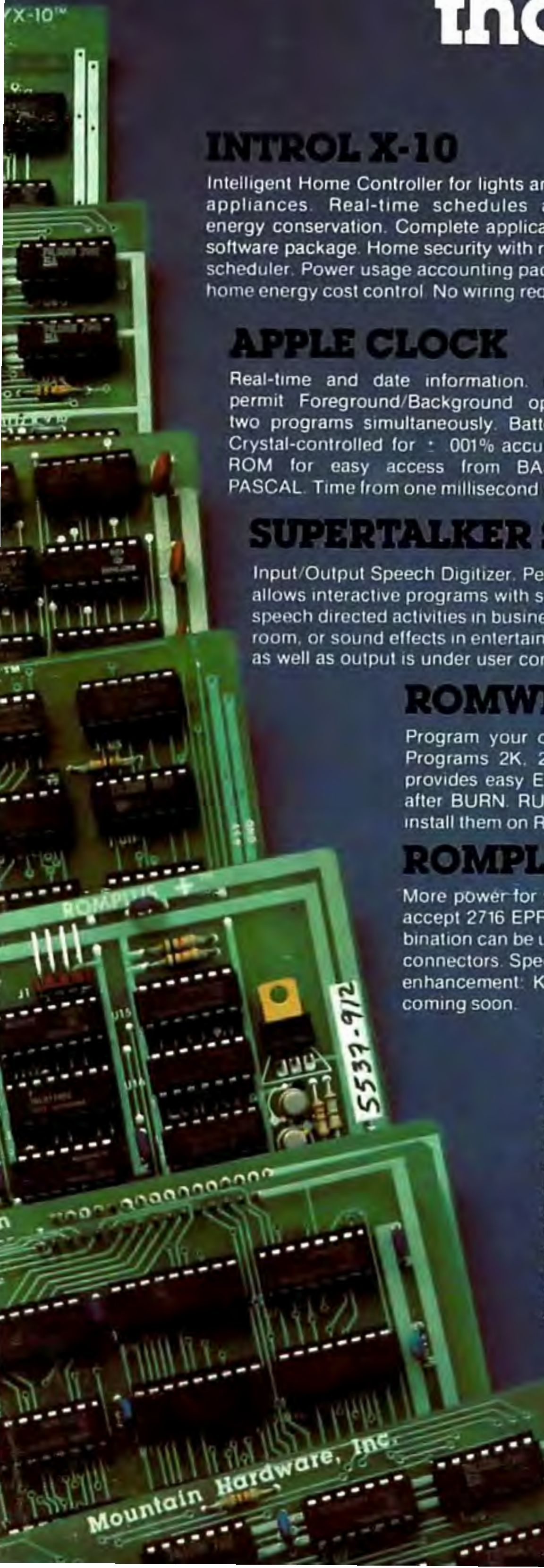
Screen Squeeze Fix For The CBM 8000

Dr. Chip Meets The Silver Streak



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

and . . . a place to put them



INTROL X-10

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

APPLE CLOCK

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

SUPERTALKER SD200

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

ROMWRITER

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

ROMPLUS+

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

MusicSystem

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

A/D+D/A

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

EXPANSION CHASSIS

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

CardReader

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

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

Available at Apple Dealers worldwide

Mountain Computer
INCORPORATED

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

*Based on Manufacturer's published catalogs [illegible]

Apple is a trademark of Apple Computer Inc

IT'S A DAISYWHEEL COMPUTER PRINTER & AN ELECTRONIC TYPEWRITER BUT . . .

IT WON'T WORK!

...UNLESS YOUR PRINTER & YOUR SOFTWARE ARE TOTALLY COMPATIBLE

Only
$2850
Suggested Retail

Lowest Priced
Daisywheel Printer with KSR Option
on the Market!

The TYPRINTER 221 T.M.

Is a TYPEWRITER QUALITY, DAISYWHEEL PRINTER that is Totally Compatible with All Word Processors. That's because the TYPEPRINTER 221 may be PROGRAMMED in PLAIN ENGLISH, Imbedded within The Text File of All Word Processing Software!

Use the 221 as your...

Electronic Typewriter

When not being used as a Computer Printer, the 221 becomes a fully functional Electronic Typewriter.

Computer Printer

It's a Daisywheel Computer Printer with more standard features and available options than any other machine.

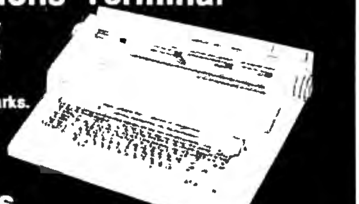
Stand Alone Terminal

Available options allow the 221 to Communicate with Distant Computer or Information Services such as Source, Micronet & others.

Tele-Communications Terminal

Option available to allow your 221 to access the Teletype & Telenex networks.

Telex & Teletype are registered trademarks.



Additional Options

4K or 16K RAM Memory which can be used as INPUT or OUTPUT Buffers. Also use as an Automatic Spooler to your computer. Bi-Directional Communicatons from The 221 to your Pet, Apple or TRS-80. Nothing else to buy. Lawyers, Accountants and others will find our Automatic Strike-Out Type and High Density Spacing options very useful.

Built-in Features

The 221 Centers Copy Automatically, Sets Columns, Prints in Reverse, Bold Face and Underlines Automatically. The 221 also Justifies Right, Types in Three Pitches and does Proportional Spacing. It Types in Spanish, French, German, Italian and Portugese as well as English. And much, much more!

Call 714/778-3443 for the distributor in your area.

HOWARD INDUSTRIES

2031 E. Cerritos Ave. 7K
Anaheim, California 92806

NOW THE SOFTCARD™ CAN TAKE YOU BEYOND THE BASICS.

You probably know about the SoftCard — our ingenious circuit card that converts an Apple II® into a Z-80® machine running CP/M.®

You may even know that with the SoftCard, you get Microsoft's powerful BASIC — extended to support Apple graphics and many other features.

Now, whenever you're ready to get beyond the BASICs, the SoftCard can take you into whole new realms. Starting with two advanced language packages from Microsoft.

FORTRAN AND COBOL TO GO.

Now you can run the world's most popular engineering/scientific language and the most popular business language on your Apple. Think what that means: you can choose from literally thousands of "off-the-shelf" applications programs, and have them working with little conversion. Or design your own programs, taking advantage of all the problem-solving power these specialized languages give you.

FORTRAN-80

A complete ANSI-standard FORTRAN (except COMPLEX type), with important enhancements. The extremely fast compiler performs extensive code optimization, and, since it doesn't require a "P-code" interpreter at run time, your programs will typically execute 2-3 times faster than with Apple FORTRAN.

FORTRAN is easy to learn if you know BASIC, and the package includes a huge library of floating point, math, and I/O routines you can use in all your programs.

COBOL-80

Virtually the only choice for serious business data processing. It's ANSI 1974 standard COBOL, with many user-oriented features added: formatted screen support for CRT terminals, simple segmenting of very large programs, powerful file handling capability, trace debugging, and much more. A separate Sort package is coming soon.

FORTRAN-80 and COBOL-80 are just two more reasons why the Apple with SoftCard is the world's most versatile personal computer. Get all the exciting details from your Microsoft dealer today. And start getting beyond the BASICs.

MICROSOFT Consumer Products, 400 108th Ave. N.E., Suite 200, Bellevue, WA 98004. (206) 454-1315.

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

MICROSOFT

Table of Contents

January, 1980. Issue 8, Vol. 3. No. 1

Page 22

Page 50

Page 66

Page 92

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

Domestic Subscriptions: 12 issues, $16.00. Send subscription orders or change of address (P.O. Form 3579) to Circulation Dept., COMPUTE. Magazine, P.O. Box 5406, Greensboro, NC 27403. Controlled circulation postage paid at Greensboro, NC 27403. Entire contents copyright © 1980 by Small System Services, Inc. All rights reserved. ISSN 0194-357X.

Robert C. Lock, Publisher/Editor
Joretta Klepfer, Manager, Dealer Marketing
Carol Holmquist Lock, Circulation Manager
Kathleen Martinek, Publication Assistant
J. Gary Dean, Art Direction/Production Assistance

COMPUTE receives continuing editorial assistance from the following persons:
Harvey Herman, University of North Carolina at Greensboro
Jim Butterfield, Toronto, Canada
Larry Isaacs, Raleigh, NC

The following writers contribute on a regular basis as Contributing Editors:
Al Baker, 2327 S. Westminster, Wheaton, IL 60187
Gene Beals, 115 E. Stump Road, Montgomeryville, PA 18936
Len Lindsay, 5501 Groveland Terrace, Madison, WI 53716
Roy O'Brien, P.O. Box 426, Beaumont, CA 92223

Subscription Information (12 Issue Year):
COMPUTE. Circulation Dept.
P.O. Box 5406
Greensboro, NC 27403 USA
U.S. $16.00
Canada $18.00 (U.S. funds)
Europe: Surface Subscription, $20.00 (U.S. funds) if ordered direct, or available in local currency from the following distributors:
United Kingdom
Contact L. P. Enterprises,
8-11 Cambridge House
Cambridge Road
Barking, Essex
England IG1 18NT
Germany,
Switzerland,
Austria
Contact Ing. W. Hofacker GMBH
8 Munchen 75
Postfach 437
West Germany
Canadian Retail Dealers should contact:
Micron Distributing
409 Queen Street West
Toronto, Ontario M5V 2A5
(416) 361-0609

The Editor's Notes

Robert Lock,
Editor/Publisher

COMDEX '80

This second annual dealer/vendor show was well attended, and pleasantly well-organized. Most interesting was being "New Kids On The Block". I suppose in part because of recent acquisitions (Hayden buying Programma International and Personal Computing Magazine), and recent stock market moves (Apple going public and Commodore stock soaring), there was a great deal of interest in the "microcomputer" people there. Dealer interest was normal to slow, but "big company" interest was strong. Everybody seems to be noticing this "new" market these days, and it was nice to meet those nice people from far-away places like Digital Equipment and Hewlett-Packard. You readers don't have any mis-impressions about such markets... Sol Libes, in his December BYTE column, cities an annual DATAMATION survey of the top 100 Computer Companies in US volume of sales. Tandy, Apple and Commodore were there (for the second year in a row), and guess which computer companies (no "micro" here) were the three fastest growing from the year before? Ah, what a wonderful "discovery" we're all "sharing in".

Atari: No Show

I was surprised that Atari didn't appear as an exhibitor at COMDEX. After all, corporate exhibitors included Apple and Commodore, as well as OSI. With market competition heating-up, we missed them! And numerous dealers commented about their absence. Sales, however, are moving along, we hear, and that's what translates into increased support and sustinance for you Atari owners.

The last day of the show was marred by the tragic MGM Grand fire. As far as we know, all exhibitors who were staying there got out safely. Among them were Bob Crowell and the crew from NEECO, and Bob Pierce from Quality Software. I have never experienced a more tragic incident, and strongly advise (after the fact) that you be cautious in your selection of, and placement in hotels while travelling. Most fire department equipment cannot reach beyond the eighth or ninth floor.

OSI Sells, But Not Out

Ohio Scientific has been sold to a company called M/A COM, Inc. Judging from recent OSI sales figures (according to the Boston Globe, $14.8 million in the ten months ending October 30), I'm sure the sale commanded a hefty price. No word yet on major implications of the change, but we have heard that key marketing personnel will remain with the company. I'll try to keep you posted on any developments as they relate to various portions of the current product line.

Computer Programs and Your Ethics
The ATARI Software Scam...

Almost one year ago, I sat here writing an editorial defending Ron Jeffries and his PET programs on tape magazine, CURSOR. I wrote in response to an editorial Ron had written regarding the number of his cassette magazines that were apparently being ripped off. Schools were high on that list, and after the editorials by the two of us, several dealers, educators, and others wrote about the problem. At that time, COMPUTE! was a brand new fledgling magazine. The editorial was written to appear in our second issue. We've grown a lot since then, and now find ourselves in the same boat. A company in Iowa is selling an article, with program, literally lifted from the pages of our Issue Number 5. The package marketed under the name "Add A Voice", has been sold to dealers around the country as original work. If you were an Issue 5 COMPUTE! reader, you'd notice a strange similarity between the documentation for that package and our published article:
"Adding A Voice Track To Atari Programs" by John Victor, President of Program Design, Inc. If you delved further into such mysteries, you'd realize that the documentation, sans our masthead and lead paragraph, were word for word indentical. And if you were especially prudent, you'd notice that the words were not even retyped, merely a pasted-up and copied reproduction of COMPUTE!.

Now I must admit, there's some personal gratification in seeing a magazine's articles making in onto Atari dealer shelves all across the country. I mean, after all, that's what we've been working on ourselves. But to sell a single COMPUTE! article, in lifted form (with cassette tape), for $19.95? A subscription for a whole year is only $16.00! Our notion of growth never included someone else doing our growing for us.

One Small Favor

If you're a dealer, or user, who's seen such software, and articles, copied from COMPUTE!, please let us know. Drop me a card, a letter, or call me. We want to know what's going on and in part rely on you for the information. I'm not going on about someone "stealing" our software: I'm talking about basic violations of copyright laws. And even if you're not clear on the laws of software, it's quite clear that magazines are different. COMPUTE! does not sell software! We sell information for a living, and that's a whole different problem. Especially when we sell a whole year of it for less than the copied article. I think the thing that really makes me furious is just that point. Some Atari owners around the country have paid $19.95 to get a single article from COMPUTE!

The Other Side Of The Scam

At the same time we discovered that our article was being sold, we discovered that Atari programs from software vendors were also being sold. Among these was an apparent version of Atari's own mailing list program and a version of Quality Software's Atari disassembler. In both cases, the programs were being sold for much more than the original vendor's asking price. Am I making it clear? We're not talking about someone "sharing" programs; we're looking at basic selling of copied material without permission. Quality Software, for example, sells their Atari disassembler for $11.95. The pirated version was selling for $29.95.

The Final Note

In this case, contrary to the initial appearance, it looks as if the company in Iowa may have been a victim of fraud as well. A late breaking article in Infoworld indicates that some other parties had sold the material from COMPUTE! and Quality Software, as original material, to the Iowa company...who in turn sold it not realizing it was copied.

I can't say more now, but vendors are going after software pirates, and hard. Our industry is old enough now and mature enough to protect the legitimate businesses from the not-so-legitimate ones. And our industry is becoming large enough so that the not-so-legitimate ones will be surfacing more frequently. Happy anniversary to "Computer Programs and Your Ethics"; I'm glad COMPUTE! has grown enough to help.

Post-Script: During all of this we've recently learned that other programs from COMPUTE! are being sold as part of "collections" of programs. One defense raised to me in a recent example was that $10.00, with a user-provided diskette, is a "reasonable" copying fee. When members of a local users group take turns keying in the longer programs from COMPUTE!, and providing them to other club members, who are also COMPUTE! subscribers, at meetings, no charge... that's a reasonable copying fee. When individuals take the same programs and sell copies for a charge, that's clearly different. And we expect to treat it differently from here. If you've been covered in the above discussion, please take a moment to read our current copyright notice. We have chosen not to sell software. We take the best of the software we see and print it on these pages. Given this choice, we don't expect others to sell it either. ©

COMPUTE! Is Looking For Good Articles

Send Program Listings, Articles, Hints, Odds and Ends, etc. to
The Editor **COMPUTE!**
P.O. Box 5406 Greensboro, NC 27403 USA

The Readers Feedback

Robert Lock,
Editor/Publisher and Readers

I should have known better. Remember way back in the November/December issue when I said we would drop back to 112-120 pages with our first monthly issue? Oh well.

Best Articles Last Issue:

Small Computers And Small Libraries
Times Squari For Your Atari
Feed Your PET Some Applesoft
Keyprint
Interfacing KIM/SYM/AIM/OSI With BASIC

On the Editor's Feedback Question:

Should we organize by content rather than Gazette's?

A **resounding** no! I defer to your collective decision-making. Not one of you who answered that question felt the idea had any merit.

The Readers Respond

How about headlining the start of each Gazette, e.g. "The SBC Gazette"?

We do; well, normally. Last time wasn't quite clear. This time we're back to norm, and each Gazette has its own masthead. You newer readers should note that the magazine starts with a "general interest" area, and continues with Gazettes devoted to specific machines. Articles are continuous and sequential (none are ever continued to any other part of the magazine). You will find articles of interest to you in other Gazettes.

Please address articles to novices for Atari. . .especially translation to Atari from other BASICs (Microsoft) and novice assembly language programming.

You'll be happy to know that we're starting a tutorial series on Atari machine language. We expect it to be up and running by next issue. Your translation problem will in part be solved when Atari's Extended (Microsoft) BASIC cartridge is introduced this spring.

From A PET Reader:

Better explaining of assembly language programs. . .how to convert from new to old ROM and vice versa. . .don't think that we know the basics. . .

And another reader. . .

HELP the Beginner

We're gradually evolving an initial section of the magazine into a beginners corner. Here you'll find Basically Useful BASIC, Odds and Ends (a new series, implemented this time by Jim Butterfield on PET cassette tape, but expanding (with your help) to cover all machines), and hopefully a Stat Lab, providing useful BASIC programs you may all use.

11/04/80
ATTN: Editor

Dear Sir,

I have been a reader of **Compute!** *from the first issue. I enjoy your magazine so much that I had to write and let you know. I just finished the latest issue (Nov./Dec.) that I picked up at your booth at the computer show in the NY Coliseum.*

The article by Charles Brannon, "Keyprint" was the first article that discussed a machine language utility that I was able to use. I am just starting in programming and am only marginal in BASIC. My system is a 32k Commodore PET and Commodore printer (2022).

Most M/L programs in magazines are listed with an assembler. In his article Mr. Brannon lists the program using the machine language monitor that comes as part of the operating system of the PET/CBM. I feel that many programmers may want to use the M/L utilities that are published in your magazine, but are scared away by the use of assemblers. I wonder if it would be possible for program authors to include a listing using the Commodore monitor so that people, such as myself without an assembler can enter these programs and utilize them even though it may take longer to type in.

Finally I want to tell you that of all the computer magazines, yours is my favorite, keep up the good work.

Very Truly Yours,
Michael Schiller

IMPORTANT ANNOUNCEMENT

We Are Sold Out of All Back Issues

Please bear with us. We expect to have certain articles available soon in reprint form.

An Interview With Dr. Chip

Robert Lock, Editor/Publisher

Editor's Note: By way of introduction to you newer readers, Dr. Chip is a frequent apparition on these pages. He is Professor of 6502 Science at Figment University, and closely monitors a group of Figment U. Users Groups all over the world. From time to time he agrees to these momentous interviews...

RCL: Well Dr. Chip, it's good to have you back. Your readers were growing concerned.

Chip: Nothing of concern, that's my whole problem now! I tried to stow away on a shipment of Commodore parts to Japan...

RCL: Why ever for, Dr.?

Chip: I figured it would be months at your current speed before you reviewed VIC so I thought I'd take a look at the Japanese version and see how it was going.

RCL: Well?

Chip: I stowed away in a new Commodore regional warehouse in California and they closed it down the next day. I ended up in limbo for eleven weeks before somebody finally shipped me to Dallas!

RCL: Sounds like a familiar problem. You must have been in an 8050 box.

Chip: Hrumph.

RCL: Well, on with it. Any word on VIC?

Chip: VIC has run into a set of new FCC regulations that they're currently trying to figure out. Word is that everybody is in the same boat (no pun intended). I hear they'll still be showing the new unit in January with a strong push for whatever approval's necessary and product introduction by February.

RCL: Sounds good. Any pre-introduction fixes involved there?

Chip: Funny you should ask. Some founding chapter Figment U. people are reporting that the machine doesn't really have the promised 32K expansion capability. And that one of the expansion connectors may need to be slightly retooled.

RCL: We're still hoping to get ours by the time this goes to press.

Chip: Enough on VIC. I've got more pressing news.

RCL: I can't wait...

Chip: You'll remember our interview in the spring where we discussed the upper level marketing turnovers at Commodore?

RCL: Yes. I said everything was settling down and Dick Powers was the new Sales and Marketing Manager.

The Terminal Madness of the Silver Streak . . .

Hey there Dr. Chip, got your screen on? This here's the Silver Streak ready for a little 'modemulation' at a baudacious 300 baud! Yessir, this electronic mail is the most fun I've had since the FCC pulled my ticket for runnin' a 2 KW CB rig. Just think, the only Tee Vee's I interfere with now are your's and mine.

I guess you know I'm not drivin' my truck anymore. The judge thought I had too much technology in the cab when he saw the microwave repeater I used to send a new speed signal back to the fuzz. It appears I wiped out every doppler radar on Highway 1! Anyway, now that I'm drivin' a desk, I got set up with an account on both Source and Micronet - I'm gettin ready to type in stereo.

I got me a whole new rig too - an Atari 800 with a printer, the RS-232 box, a modem, a Telelink I communications cartridge, and one of those new projection Tee Vees with a six foot screen. My setup is so fancy, I had to add on a waiting room for folks who come over to see the equipment. Don't that just flush your buffer?

And to think that I was told I'd never get the hang of this stuff just because I thought a tri-state bus driver was someone who worked for Greyhound - well, everyone has to start somewhere, right?

So I was chatting with this lady on the Source the other night and she asked if I had read the Third Wave by this Alvin Toffler fella. I told her I had heard about it and, as near as I can figger, the first wave was telephones, the second wave was CB, and the third wave was this electronic mail stuff. Well sir, she was so impressed with the novelty of my answer that she said I should get in touch with you 'cause you are interested in unique characters. She even said I was the most unformattable character she ever met. Wasn't that nice?

Whoops! I just went into prime time. Guess I better pop my stacks and make some tracks!

Catch you around the net, Dr. Chip.
So long for now.
.SEND

Dr. Chip seems to have acquired a new (and strange) friend. Chip informs me that we can expect to see occasional communications from the Silver Streak. RCL

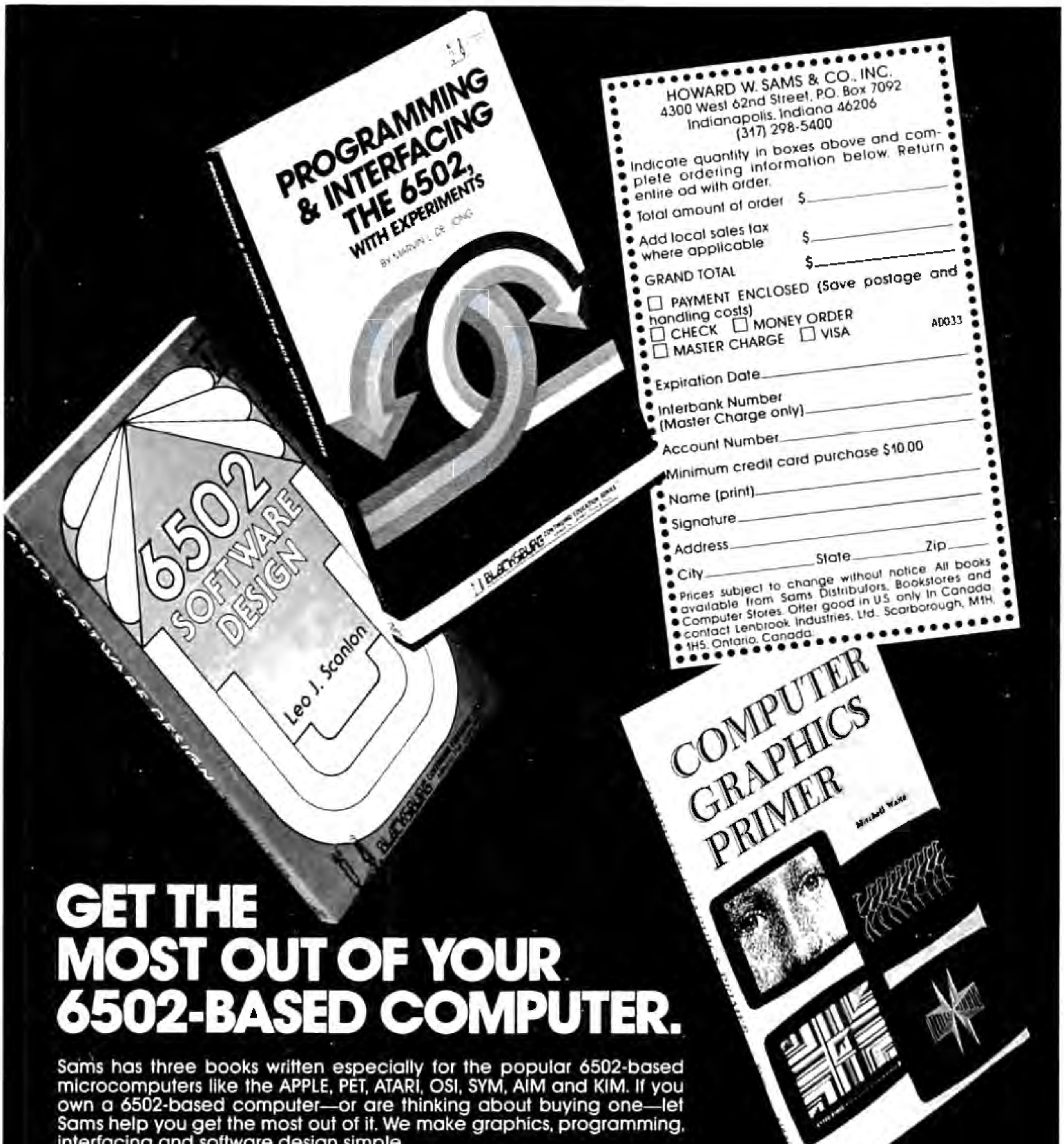
Chip: Right. You blew it. Powers has moved to head up OEM Sales. He was replaced by Bill Robinson...

RCL: Okay, so now there's a new Marketing Director...

Chip: You got it. His name's Mike Tomczyk. He assumed some of Robinson's duties when Bill left.

RCL: Well, at least the seven regional distribution centers are lending some new cohesiveness.

Chip: I hear they'll only now getting things up and rolling in some regions, and several dealers have complained with a bitterness reminiscent of years



HOWARD W. SAMS & CO., INC.
4300 West 62nd Street, P.O. Box 7092
Indianapolis, Indiana 46206
(317) 298-5400

Indicate quantity in boxes above and complete ordering information below. Return entire ad with order.

Total amount of order $______

Add local sales tax where applicable $______

GRAND TOTAL $______

□ PAYMENT ENCLOSED (Save postage and handling costs)
□ CHECK □ MONEY ORDER
□ MASTER CHARGE □ VISA

AD033

Expiration Date______

Interbank Number (Master Charge only)______

Account Number______

Minimum credit card purchase $10.00

Name (print)______

Signature______

Address______

City______ State______ Zip______

Prices subject to change without notice. All books available from Sams Distributors, Bookstores and Computer Stores. Offer good in U.S. only. In Canada contact Lenbrook Industries, Ltd., Scarborough, M1H, 1H5, Ontario, Canada.

GET THE MOST OUT OF YOUR 6502-BASED COMPUTER.

Sams has three books written especially for the popular 6502-based microcomputers like the APPLE, PET, ATARI, OSI, SYM, AIM and KIM. If you own a 6502-based computer—or are thinking about buying one—let Sams help you get the most out of it. We make graphics, programming, interfacing and software design simple.

■ **PROGRAMMING & INTERFACING THE 6502, WITH EXPERIMENTS.**
By Marvin De Jong. The more you know about programming and interfacing, the more performance you can get out of your microcomputer. This hands-on guide to 6502 presents 80 carefully graded experiments to help you get the most out of your AIM, KIM or SYM. NO. 21651. $13.95

■ **6502 SOFTWARE DESIGN**
Leo Scanlon—a leading computer expert—simplifies software design. Takes you from fundamentals into more complex topics. Get more versatility out of your computer by learning to program it yourself. INCLUDES 89 TEST PROGRAMS! NO. 21656. $10.50

■ **COMPUTER GRAPHICS PRIMER**
Mitchell Waite—one of the most popular computer authors—brings computer graphics into sharp focus. Shows you how to use a 6502-based computer to create complex drawings, plans, maps and schematics on a video screen. NO. 21650. $12.95

Sams
BOOKS

past. And their biggest beef is that there's no one to complain to. Management keeps changing so fast they can't keep up. Three regional managers have been replaced since the seven centers were so highly touted just months ago.

RCL: Ouch. Maybe now that COMPUTE!'s going monthly, we should start printing a running who's who for readers and dealers.

Chip: Not a bad idea.

RCL: By the way, APPLE III's are now shipping.

Chip: So I heard. I also understand that "Word Craft", the British word processor for Commodore machines, was not present at the COMDEX booth, but that Commodore was instead showing off Word Pro from Professional Software, Inc. An interesting turn of events indeed.

RCL: Thanks Chips, we'll see you next time. By the way, who's your new friend Mr. Streak?

Chip: No comment.

Late breaking news: Chip reports two potentially significant new products. MOSAIC Electronics is introducing a 32K RAM board that will work on both the Atari, Inc. 400 and 800 machines. Microtechnology Unlimmited is developing a 6502 based computer that will retail for under $1,000.00 Standard equipmment, reports Chip, includes: black annd white monitor; stringy floppy type tape drive; standard keyboard, enclosure and power supply; Microsoft BASIC; graphics capability of 320 by 200; and 48K RAM. COMPUTE! looks forward to reviewing these two introductions. ©

VIC!

Commodore's new $299 personal computer made an appearance in Las Vegas. The following description of VIC is based on the two units present in the Hilton. These were not final production units, but they should be fairly close.

VIC is not much larger than its keyboard, which is about the same size as the keyboard on current PETs. New additions to the keyboard include a CTL key and four function keys. The VIC doesn't have the numeric pad that the current PETs have.

For display the VIC can use any normal color TV. The display will be 22 characters by 23 lines. Line wrap will allow line lengths of 88 characters. Each character cell in the display has its own background color and character color. An advantage of the 22 character display is that the characters are readable regardless of the character and background colors, as long as they aren't the same. The character set for U.S. units will be the standard PET character set found in the current PET.

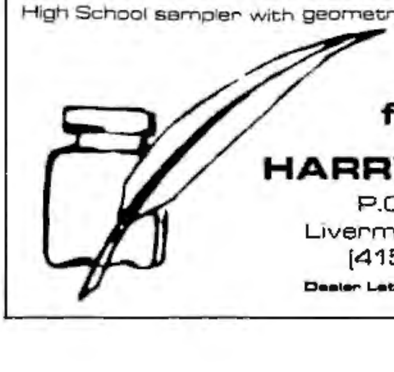
The unexpanded VIC will contain BASIC, and user RAM from $1000 to $1FFF. The top 512 bytes are reserved for character memory (like screen memory), leaving 3584 bytes free for BASIC programs. This RAM may be expanded down to $400 which gives 6656 bytes. Or it may be expanded above character memory to give around 28K bytes.

For connection to the outside world there is a video connector. In addition there is a cassette and user port that is much the same as on current PETs. For connection to peripherals other than cassette, there is a serial port connector. This port serves the same purpose as the IEEE port on PETs and CBMs. A second method of serial I/O is provided on two of the bits of the user port. A small amount of circuitry is required to convert the TTL output levels to true RS232. For system expansion there is a 44 pin edge finger connector. This is where cartridges will plug in. I was told that one of the expansion products will be an expansion motherboard to allow more than one cartridge to be plugged in at one time.

VIC will contain a BASIC that is essentially the same as BASIC 2.0. It has been modified, however, to allow cartridges with ROM to add new commands to BASIC. One such cartridge will add a set of GRAPHIC commands. Creating color displays in the unexpanded VIC is done using PRINT statements, with special control characters to select colors. Not much information was available on the high resolution display modes of VIC. At the time of the Las Vegas show, the full power of VIC's color display chip have yet to be explored.

Finally, concerning FCC approval, we were told that VIC was currently undergoing tests to determine what needs to be done to meet FCC requirements. If all goes well, I would expect VIC to make its appearance in the marketplace sometime in the first quarter of 1981.

Larry Isaacs ©



SEND FOR OUR FREE CATALOG

COMPUMART LETS YOU PUT APPLE ON TRIAL

WITH OUR EXCLUSIVE 10-DAY FREE RETURN ON ALL THESE GREAT APPLE COMPUTERS AND ACCESSORIES.

CompuMart carries complete lines of microcomputers from Apple to Zenith, as well as thousands of microcomputer peripherals and accessories. Write for our FREE 36 page catalog.

Apple Computer

We carry the most complete inventory of Apple computers, peripherals and software. CALL!

Our Christmas Apple Special: Save over $250 on our most popular Apple System. System includes a 48K Apple II, Apple Disk & Controller, and a Sup R Mod RF Modulator.

List: **$2,020**

Compumart Sale Price: **$1,769**

New from Apple for the Apple II:

DOS 3.3 Convert disks to 16 sector format for 23% more storage and faster access $60

Apple Plot The perfect graphic complement for Visicalc	$70
Dow Jones News & Quotes	$95
Adventure (Uses 48K)	$35
DOS Tool Kit	$75
Apple Fortran	$200
Silentype Printer w/Xface	$595
Visicalc	$149
Tax Planner	$120
From Symtech & Info Unlimited	
Super Sound Generator (mono)	$159
(stereo)	$259
Light Pen	$249
X-IO Controller (plugs into paddle port)	$49
Apple Sync Controller	$49
From Personal Software	
Visicalc	$149
Desk top plan	$99

New from Videx! — Video Term 80 Col. x 24 line 7 x 9 matrix, plug in compatible board for the Apple II. Price	$325
without graphics EPROM.	
With graphics EPROM	$350.
New from MUSE	
The Voice	$39.95
Super Text	$99.00
Address Book	$49.95
Mountain Hardware — Expansion accessories for your Apple	
Introl/X-10 System	$289
Super Talker	$299
The Music System	$545
ROM plus board w/keyboard filter	$199
Clock Calendar	$280
16 Channel A to D Converter	$350
Apple Expansion Chassis	$650
ROM Writer	$175
Miscellaneous Apple II Accessories	
Easy Writer (80 col. need Videx)	$249
Easy Mover	$49
Easy Mailer	$69
Dysan Diskettes	ea. $5
SSM Serial & Parallel Apple Interface	$225
ABT's Numeric Key Pad	$110

270 THIRD ST. DEPT 307 **COMPUMART** CAMBRIDGE, MA 02142

TOLL FREE 1-800-343-5504 IN MASS 617-491-2700

apple computer
Authorized Dealer

Computers And Society

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

Happy New Year! Now that Compute is being published monthly, I have decided to stay on a two-month cycle, so you will hear from me every other issue - at least for a while. As always, please feel free to contact me at the above address or through the source at TCE132.

In thinking about the role of computers in society, there is one application which stands well above the rest in its potential impact on the general public. This application is computer communication, be it electronic mail, computer conferencing, or just on-line chatting.

The Europeans have been pioneers in this area through various videotext systems in England, France, Germany, and elsewhere. In our continent, the Canadian Telidon effort stands out as a particularly nice piece of technology for the home information market. We in the United States have access to the Source and Micronet on a national scale, and there are many local experiments ranging from replicas of the European systems to the distribution of video game software through the TV cable (the Mattel/Jerrold Playcable system).

This field grows more exciting in its potential every day. The acquisition of a majority of Source Telecomputing by the Reader's Digest, and the existence of the Better Homes and Gardens menu data base on Micronet serve as strong testimony to the idea that these systems are gearing themselves to the needs and desires of the general public.

It is my opinion that we are experiencing a communications revolution whose impact is likely to be as great, if not greater, than that of the telephone.

While I encourage you to give me your perspectives on this topic so I can share then with the rest of our readers, I thought that it might be valuable to review some recent books which deal with various aspects of this topic. The books I selected range from history to science fiction, with the common thread that they deal with the use of computers as communication aids for human beings.

The Third Wave by Alvin Toffler, William Morrow and Co.

It is almost impossible these days to pick up a newspaper or magazine without seeing reference to Toffler's ideas. Before rushing out to buy a copy of this book, however, be aware that an up-to-date technologist might be mildly disappointed in it since Toffler devotes his major effort to chronicling the work of others and threading these efforts onto a conceptual latticework rather than probing deeply into the motivation behind these developments.

The real value of this book, in my opinion, comes from the clarity with which the author is able to convey these developments to the general public in a manner which generates excitement while maintaining a good deal of objectivity. Toffler writes like a newspaper reporter, which may be of questionable value for a 500 + page book, but this style may appeal to those of you who enjoy reading only a few pages at a sitting.

The title of the book derives from the three major overlapping stages in the development of society from 8000 BC to the present time. Beginning with the idea that the rise of agriculture was the first turning point in human social development, and that the industrial revolution was the second breakthrough, then the development of what others have called the "information age" heralds the coming of the Third Wave.

As Toffler points out, there is much more than technology separating these waves. Family structures, concepts of work, time, space, even life itself, all are influenced by the nature of the society in which one lives. Life gets particularly exciting during transitionary times (as we are seeing now, for example), and one of Toffler's themes is that we should be aware of the larger context in which these changes are taking place, in order to accept them gracefully.

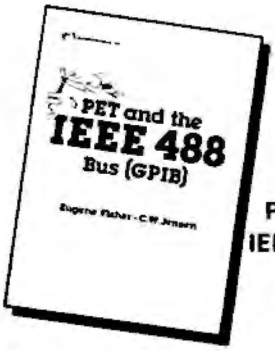
As for the Third Wave society, it is suggested that the advent of low-cost communications and distributed computing will be the nucleus of the new cottage industries. As in First Wave societies, many Third Wave workers will operate out of their homes. Toffler sees this in a most positive light. I must confess that the prospect of spending less of my life in airplanes appeals to me as well.

In Toffler's mind, you who are using personal computers, and talking to each other through electronic mail, represent the leading edge of the Third Wave. His view of the personal computer world is so favorable that I would bet that personal computer sales showed an upturn when this book was published.

If you give after dinner talks to the general public on personal computing, or otherwise address groups on this topic, you should read this book. You can safely bet that some of the people in your audience have!

The Micro Millenium by Christopher Evans, Viking Press

This book shares a similar theme with The Third Wave - that computer technology will have an extraordinary impact on our lives. Unlike Toffler, however, Evans presents a more detailed background for his concepts, thus making his book more



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

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

#31-4 $15.99

NEW PET/CBM edition

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

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

#40-3 $14.99

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

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

Practical BASIC Programs
ed. Lon Poole

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

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

#38-1 $15.99

6502 Assembly Language Programming
by L. Leventhal

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

#27-6 $16.99

New for your PET
from
OSBORNE/McGraw-Hill

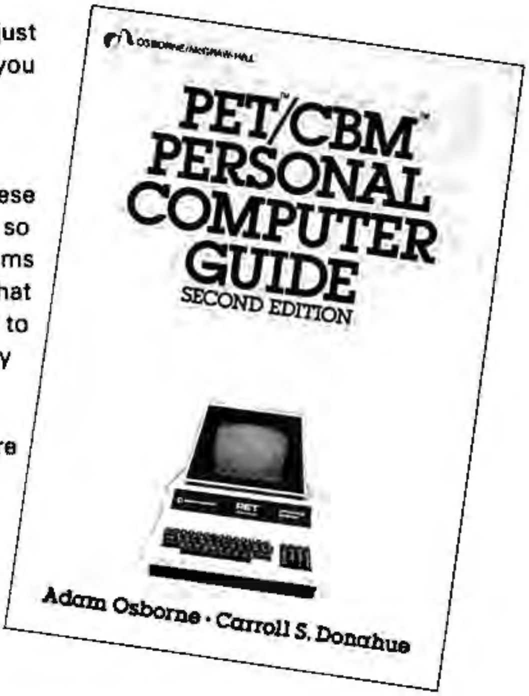
NEW Second Edition

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

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

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

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



Included are:

Complete operating instructions for

- keyboard
- tape cassette
- disk

Description of all CBM BASIC statements
Optimal programming techniques including

- input/output programming
- file handling
- screen editing

Solutions to programming problems
CBM capabilities and limitations

#55-1 $15.00

Book/Cassette/Disk	Price	Quanity	Amount
27-6 6502 Assembly Language Programming	$16.95		
30-6 PET Personal Computer Guide	$15.00		
31-4 PET and the IEEE 488 Bus (GPIB)	$15.99		
40-3 Some Common BASIC Programs PET/CBM ed. (book)	$14.99		
25-X Some Common BASIC Programs PET Cassette	$15.00		
33-0 Some Common BASIC Programs PET Disk	$22.50		
38-1 Practical BASIC Programs	$15.99		

California residents add 6% sales tax
S.F. BART residents add 6 ½% sales tax

California resident tax ____
Shipping ____
Total Amount Enclosed ____

Shipping: (Shipping for large orders to be arranged)

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

Cassettes and Disk

- No additional charge in the U.S.
- $1.50 each foreign airmail

OSBORNE/McGraw-Hill
630 Bancroft Way, Dept.G3
Berkeley, California 94710
(415) 548-2805 · TWX 910-366-7277

√06

Name ____________________

Address ____________________

City ____________________

State ____________ ZIP ____________

Phone ____________________

□ VISA □ Master Card

Credit Card No ____________________

McGraw Hill

1110

readable. The Micro Millenium presents the astounding advances in computer technology in historical sequence, starting with the developments of Pascal, Jacquard, Babbage, and others.

As Evans gazes into his crystal ball, he sees much of the same sort of thing that Toffler predicts:

- Reduced work weeks and the rebirth of the cottage industry
- The disappearance of books, with their contents being stored on single silicon chips which can be "read" with special "viewers"
- Replacement of the postal system with electronic mail
- Replacement of much business travel with computer conferencing
- The ultimate success of EFT (Electronic Funds Transfer) which will eliminate the need for currency

Along with these predictions (for which he provides technological justification), there is another thread carefully woven into this book - that of machine intelligence. Early on he quotes Ada, the Countess of Lovelace (who had the distinction of being the first computer software specialist). Her work with Babbage included studying his plans for the "Analytical Engine" in depth. On the issue of whether the machine could become creative, she wrote:

> *The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform. It can follow analysis; but it has no power of anticipating any analytical relations or truths. Its province is to assist us in making available what we are already acquainted with.*

This topic comes up with increasing frequence as the book progresses until one reaches a three chapter section which deals entirely with the concept of the Ultra Intelligent Machine.

Chris Evans' views on the feasibility of machine intelligence are not without controversy. His untimely death prior to the publication of this book precludes his continued participation in this discussion, but this fine book stands as a most articulate exposition of his views.

The Network Nation by Starr Roxanne Hiltz and Murray Turoff, Addison-Wesley Publishing Co.

One of the features of personal information utilities such as the Source or Micronet, is the potential they have for creating distributed discussion groups. These discussion groups become the nucleus of a massive interconnected conferencing scheme in which the participants can take part in several ongoing conferences on various topics without leaving the comfort of their own homes. Computer conferencing is not a particularly new idea, and it is the published research on early conference experiments which gives us a glimpse of what this environment will be like for us.

The Network Nation is an interesting book for several reasons. First, the authors have widely different fields of expertise: sociology and computer science. This diversity of background gives the book far more depth than might be present had both authors been experts in the same field. Second, this book views the future from a solid base of research on computer conferencing conducted by the authors since 1970. While the systems they studied were generally limited to specific government funded projects, and had less than a thousand users, the nature of this new communications medium was carefully studied from many perspectives.

Whether it is called computer conferencing, electronic mail or chatting, it is clear from this book that communication between people through the medium of the computer is markedly different from communication through any other medium. In the case of the telephone or CB, for example, it is required that all participants in a conference be on-line simultaneously. On the other extreme, a remote conference which takes place through the mails has very long delays associated with each round of messages. Computer conferencing has a niche of its own.

What Hiltz and Turoff found was that this niche has some interesting characteristics. Deprived of the body language and verbal nuances which accompany face-to-face meetings, participants in computer conferences have had to find new ways to express emotion. Also, since a single message may be sent to hundreds of people with a single keystroke, the accumulation of messages in ones "in-basket" can be quite disequilibrating. Hiltz and Turoff found that people tended to read their mail often enough to prevent the accumulation of more than seven new messages. As with any other technological "toy", some users became addicted to the system, logging on as many as three times a day. If this work can be extrapolated to the public at large, our own "Network Nation" will definitely have its fraction of information "junkies" who can hardly wait for their next "fix".

There doesn't seem to be an aspect of computer conferencing which the authors overlooked, whether it was privacy, regulation, foul language, or ways of making the system better for the novice user. As for the impact of this technology on society, the authors say:

> *It is our belief that we are entering an era in which the ability of an individual to function as a citizen of that society will depend on adequate access to computerized information and communication systems. Imagine today a person trying to function without being able to use the telephone. We believe that within 10 to 20 years computer terminals will be as necessary as telephones are today.*

This is a scholarly work which will take on increased importance as the home information utility increases in popularity.

SORT is a 6502 machine language in-memory sorting algorithm of commercial quality for PET and APPLE owners. Most sorts are accomplished in less than a second and large sorts take only a few seconds. The algorithm is a diminishing increment insertion sort, with optionally chosen increments. There are no conditions under which SORT performance degenerates or fails.

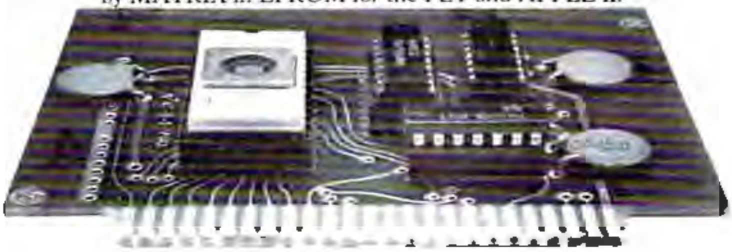
SORT requires almost no user set-up operations. SORT handles integer, floating-point, and string arrays plus arrays of more than one dimension. In addition, multi-key sorting of string arrays has been enabled. The user may specify the character within a string to begin sorting on and how many characters are to be evaluated. SORT is capable of performing up to twenty of these multi-key sub-sorts (on matches found) at the same time.

SORT on the PET: SORT is available for large-keyboard PETS only. One EPROM fits all newer 40 & 80 column PETS. SORT EPROM comes at hex $9000, $A000, or $B000 socket. EPROM with SORT and text dump is $55.00 (postpaid).

SORT on the APPLE II via a quality slot independent EPROM board. Board includes function driver that supports 16 EPROM based functions for user EPROMS. APPLE EPROM card with SORT, text screen dump and function driver is $110.00 (postpaid). MASTERCHARGE & VISA accepted.

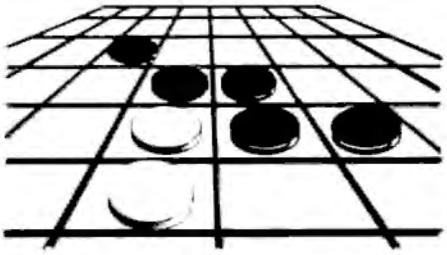
SORT

by MATRIX in EPROM for the PET and APPLE II.



MATRIX software, inc. 315 Marion Avenue, Big Rapids, MI 49307 (616) 796-2483
A Complete Line of Software for Small Businesses.

Hayden's Going Great Games for 1981!



REVERSAL (Spracklen) Winner of the software division of the First International Man-Machine OTHELLO™ Tournament, this version of the 200-year-old game Reversi, features 27 levels of play and high-resolution color graphics. *07004, Apple II tape, $29.95; 07009, Apple II Disk, $34.95.*

ROYAL FLUSH: Competitive Poker Solitaire (Wazaney) A game you can play alone or with any number of players. High score wins in this poker-based, fun-filled card game. Choose from possible game variations. *07101, PET; 07103, TRS-80 Level II, each tape $14.95.*

BLACKJACK MASTER: A Simulator/Tutor/Game (Wazaney) A serious program that performs complex simulations and evaluations of any playing and betting strategies that you enter into the computer. And, it will tutor you in the playing of these strategies and perform an evaluation of the results. *05303, TRS-80 Level II tape, $19.95; 05308, TRS-80 Disk Version, $24.95.*

Available at your local computer store!

SARGON II (Spracklen) The Champ of champs. "This program represents a giant step forward in microcomputing Chess . . . an excellent program which will provide a true challenge for many players. We are impressed with the program's speed, its opening book, and its much improved end game . . . Save your money and buy SARGON II . . ." *80 Software Critique. 03403, TRS-80 Level II; 03404, Apple II; 03401, OSI C1P; 03420, OSI C2P; 03440, OSI C4P; 03410, PET; each tape $29.95; 03408, TRS-80 Level II Disk; 03409, Apple II Disk; 03414, OSI C1P Disk; 03424, OSI C2P Disk; 03444, OSI C4P Disk; 03484, C8P Disk; each $34.95.*

Call Toll Free

24 hours a day.
(1 800 821 3777, ext. 302)* TO CHARGE YOUR ORDER TO Master Card or Visa
Minimum order is $10.00, customer pays postage and handling
From Missouri call (1 800 892 7655, ext. 302)

Hayden
50 Essex Street, Rochelle Park, NJ 07662 **Book Company, Inc.**

Videotext: the Coming Revolution in Home/Office Information Retrieval by Efrem Sigel, Colin McIntyre, Max Wilkinson and Joseph Roizen, Knowledge Industry Publications

As many of you already know, the Europeans have been experimenting with home information utilities for several years. These activities, grouped under the generic heading of Videotext, actually comprise two very different services: Teletext and Viewdata. While both of these systems use specially modified television sets as the display medium, Teletext is a receive-only service in which textual data is encoded on several unused scan lines of the television broadcast signal, and Viewdata is a fully bi-directional system on which the data is carried over the phone lines.

Rather than concentrating in the details of the technology associated with these systems, the authors of this book describe the development of these systems from a programming, marketing, and regulatory perspective. Since we in the United States are starting to see experiments with these systems in regional trials, it is most interesting to see how these systems were developed in the United Kingdom and elsewhere.

The authors warn that these systems are not easy to get up or to sell to the public, and that the established habit of watching TV as a pastime, for entertainment only, is a major obstacle to the growth of these systems. For example, in the first two years of Teletext service in England, only 15,000 specially equipped sets were sold. This service provides captioning for the deaf, news bulletins, subtiles, and other services.

It is my feeling that the high cost of televisions in the UK may be a major contributor to the lack of enthusiastic response, but one can hardly argue with the authors' viewpoint that the initial growth will be among business and professional users who are used to paying a premium for rapid information retrieval.

This book is a good introduction to the home information terminal environment.

Viewdata and Videotext, 1980-81: A Worldwide Report, Knowledge Industry Publications

This report contains more than fifty technical papers which were presented at Viewdata 80, the first world conference on videotext. This report provides detailed technical expositions on the various home information utility services being provided all over the world. While the previously reviewed book gave an overview of this topic from a historical perspective, this report gives a raw up-to-the-minute report on the services being tried in Canada, France, the United Kingdom, the United States, Japan, Germany, the Netherlands and Finland. All facets of the technology seem to be represented, from the choice of display format to the problems of carrying advertising on the medium.

One might think that an international commission should set a single standard so that all countries can use the same system. The problem with this approach is that the television formats differ from country to country. Our televisions, which use the 525 line NTSC signal, are considered primitive by European standards, for example. Beyond the issues of display format, there is the related issue of how graphics are displayed on the screen. The original British experiments used "alphamosaic" characters for graphics (somewhat like the method used for drawing pictures on the PET). At the other extreme, the Canadian Telidon system sends "picture description information" which the local processor must decode and use to generate graphics signals itself. While this approach is presently more expensive than that using a special character generator (as is used in the alphamosaic scheme), the results are breathtaking.

What is especially heartening about this report is that researchers came from all over the world to share their views and results with the goal of building as much mutual compatibility into their individual systems as possible. In reading these papers it is apparent that thousands of people are working towards making their own "Network Nation" part of the "Wired World".

The Medusa Conspiracy by Ethan Shedley Viking Press

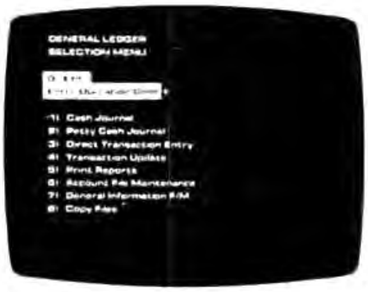
While it is easy to see much benefit from a "Wired World", one must always consider the potential impact of a massive system failure, it is even more chilling to think of the consequences of a subtle failure whose presence may go undetected for some time. When this failure affects international security, the results can be deadly.

The Medusa Conspiracy is a novel which deals with this very topic in such a realistic manner that it is obvious that the author is a computer scientist who writes novels on the side. Revolving around topical international intrigue (involving the Russians, United States, Israel, and the Arabs), this story deals with a data base error which brings the world to the brink of war. The President of the United States has a terminal in his office with which he contacts Medusa, a massive relocatable program which serves as his principal source of foreign intelligence information. Medusa shifts itself around the ARPANET, residing in various computers with available resources. Slowly, almost imperceptibly at first, the system starts to break down in devious ways. The services of Dr. Seth Miller, one of Medusa's implementers, are brought in to fix the problem. The story of his efforts (told in accurate technical detail), coupled with the attempts of others to thwart his success, makes for a gripping tale.

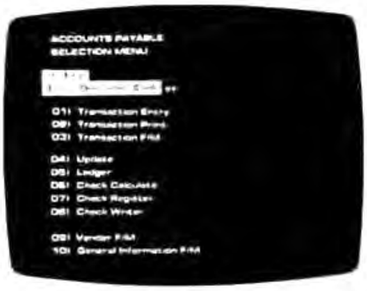
As mentioned above, the author is a computer scientist. The technical accuracy of this book adds to its suspense and Compute! readers should have no trouble with the jargon.

Professional Business Software

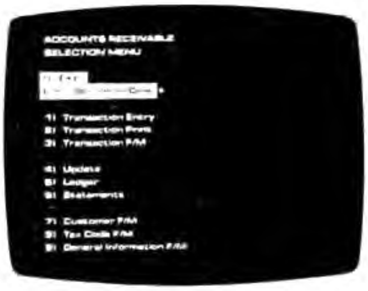
For The Commodore 32K Microcomputer System
With 2040 Dual Drive Disk & 2022 Tractor Feed Printer



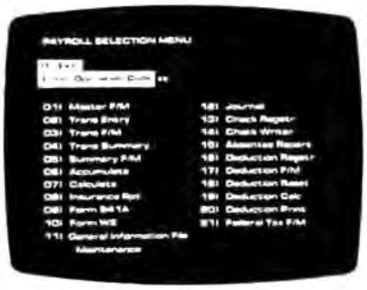
```
GENERAL LEDGER
SELECTION MENU

0) Exit
Enter Operation Code

1) Cash Journal
2) Petty Cash Journal
3) Direct Transaction Entry
4) Transaction Update
5) Print Reports
6) Account File Maintenance
7) General Information F/M
8) Copy Files
```



```
ACCOUNTS PAYABLE
SELECTION MENU

0) Exit
Enter Operation Code

01) Transaction Entry
02) Transaction Print
03) Transaction F/M

04) Update
05) Ledger
06) Check Calculate
07) Check Register
08) Check Writer

09) Vendor F/M
10) General Information F/M
```



```
ACCOUNTS RECEIVABLE
SELECTION MENU

0) Exit
Enter Operation Code

1) Transaction Entry
2) Transaction Print
3) Transaction F/M

4) Update
5) Ledger
6) Statements

7) Customer F/M
8) Tax Code F/M
9) General Information F/M
```



```
PAYROLL SELECTION MENU

0) Exit
Enter Operation Code

01) Master F/M                    12) Journal
02) Trans Entry                   13) Check Register
03) Trans F/M                     14) Check Writer
04) Trans Summary                 15) Absentee Report
05) Summary F/M                   16) Deduction Register
06) Accumulate                    17) Deduction F/M
07) Calculate                     18) Deduction Reset
08) Insurance Rpt                 19) Deduction Calc
09) Form 941A                     20) Deduction Print
10) Form W2                       21) Federal Tax F/M
11) General Information File
      Maintenance
```

General Ledger

- Holds Up To 300 Accounts
- Accepts Up To 3000 Transactions Per Month
- Cash Disbursements Journal, Cash Receipts Journal, and Petty Cash Journal for simplified data entry
- Maintains Account Balances For Present Month, Present Quarter, Present Year, Three Previous Quarters, And Previous Year
- Complete Financial Reports Including Trial Balance, Balance Sheet, Profit & Loss Statement, Cash Receipts Journal, Cash Disbursements Journal, Petty Cash Journal and more
- Accepts Postings From External Sources Such As Accounts Payable, Accounts Receivable, Payroll, Etc

Accounts Payable

- Interactive Data Entry With Verified Input And Complete Operator Prompting.
- Automatic Application Of Credit And Debit Memos
- Maintains Complete Purchase Records For Up To 200 Vendors
- Invoice File Accepts Up To 400 Invoices
- Random Access File Organization Allows Fast Individual Record Updating
- Multiple Reports Provide A Complete Audit Trail
- Check Printing With Full Invoice Detail
- Full Invoice Aging
- Automatic Posting To General Ledger

Accounts Receivable

- Maintains Invoice File For Up To 300 Invoices
- Accomodates Full Or Partial Invoice Payments
- Customer File Maintains Purchase Information For Up To 1000 Customers.
- Allows For Automatic Progress Billing
- Provides For Credit And Debit Memos As Well As Invoices
- Prints Individualized Customer Statements.
- Interactive Data Entry With FullOperator Prompting
- Complete Data Input Verification And Formating
- Automatic Posting To General Ledger

Payroll

- Maintains Monthly, Quarterly, And Yearly Cumulative Totals For Each Employee.
- Payroll Check Printing With Full Deduction And Pay Detail
- Sixteen Different Reports Including W2 And 941
- Interactive Data Entry With Easy Correction Of Entry Errors.
- Automatic Data Verification
- Complete Job Costing Option With Cumulative Totals And Overhead Calculations
- Random Access File Organization For Fast Updating Of Individual Records
- Automatic Posting To General Ledger

Structured around the time tested and reliability proven series of business software systems developed by Osborne and Associates, these programs have been designed to fill the need of a comprehensive accounting package for the new Commodore PET micro computer system. Each program can either stand alone, or be integrated with the others in a total software system.

Designed with the first time user in mind, these programs lead the operator through step by step, verified data entry. It is impossible to 'crash' a program due to operator error or invalid data input. Design consistency has been maintained from program to program to greatly increase operator familiarity and confidence.

Documentation, normally a problem for small systems users, is provided by the comprehensive series of Osborne and Associates user manuals. These three manuals together total over 800 pages of detailed step by step instructions written at three levels for DP Department Managers, Data Entry Operators, and Programmers. You don't have to worry about getting 'promises' instead of documentation because the documentation was written before the programs were developed. A second set of manuals details any changes required during conversion. Each program provided on disk with complete documentation. Packaged in a handsome three ring binder with pockets and twelve monthly dividers for convenient storage of reports.

See your nearest Commodore dealer for a demonstration of this outstanding business software system.

CMS Software Systems

5115 MENEFEE DRIVE • DALLAS, TX 75227 • 214-381-0690

NEECO

"Your complete source
for all CBM Hardware
and Software Products"

PROUDLY ANNOUNCES OUR NEW ONE YEAR WARRANTY ON ALL CBM COMPUTERS!

The 8032 CBM Computer is now available!

commodore

CBM™ 8000 SERIES BUSINESS COMPUTERS

The new Commodore 8000 series computers offer a wide screen display to show you up to 80-character lines of information. Text editing and report formatting are faster and easier with the new wide-screen display. The 8000 series also provides a resident Operating System with expanded functional capabilities. You can use BASIC on the 8000 computers in both interactive and program modes, with expanded commands and functions for arithmetic, editing, and disk file management. The CBM 8000 series computers are ideally suited for the computing needs of the business marketplace.

CBM™ 8050 DUAL DRIVE FLOPPY DISK

The CBM 8050 Dual Drive Floppy Disk is an enhanced version of the intelligent CBM 2040 Disk Drive. The CBM 8050 has all of the features of the CBM 2040, and provides more powerful software capabilities, as well as nearly one megabyte of online storage capacity. The CBM 8050 supplies relative record files and automatic diskette initialization. It can copy all the files from one diskette to another without copying unused space. The CBM 8050 also offers improved error recovery and the ability to append to sequential files.

HARDWARE SPECIFICATIONS

Dual Drives
Two microprocessors
974K Bytes storage on two 5.25" diskettes (single sided)
Tracks 70
Sectors 17-21
Soft sector format
IEEE-488 interface
Combination power (green) and error (red) indicator lights
Drive Activity indicator lights
Disk Operating System Firmware (12K ROM)
Disk Buffer (4K RAM)

FIRMWARE

DOS version 2.1
Sequential file manipulation
Sequential user files
Relative record files
Append to sequential files
Improved error recovery
Automatic diskette initialization
Automatic directory search
Command parser for syntax validation
Program load and save

CBM	PRODUCT DESCRIPTION	PRICE
4008N	8K RAM-Graphics Keyboard-40 col	$ 795.00
4016N	16KN RAM-Graphics Keyboard-40 col	$ 995.00
4016B	16K RAM-Business Keyboard-40 col	$ 995.00
4032N	32K RAM-Graphics Keyboard-40 col	$1295.00
4032B	32K RAM-Business Keyboard-40 col	$1295.00
8016	16K RAM-80 Col.-4.1 O/S	$1495.00 •
8032	32K RAM-80 Col.-4.1 O/S	$1795.00
2023	Friction Feed Printer	$ 695.00
2022	Tractor Feed Printer	$ 795.00

NOTE:
All current CBM production computers/disks now contain operating system 4.1/DOS 2.1

CBM	PRODUCT DESCRIPTION	PRICE
2040	Dual Floppy-343K-DOS 1.0	$1295.00
4040	Dual Floppy-343K-DOS 2.0	$1295.00
8050	Dual Floppy-974K-DOS 2.0	$1695.00 *
C2N Cassette	External Cassette Drive	$ 95.00
CBM to IEEE	CBM to 1st IEEE Peripheral	$ 39.95
IEEE to IEEE	CBM to 2nd IEEE Peripheral	$ 49.95
8010	IEEE 300 Baud Modem	$ 395.00 *
2.0 DOS	DOS Upgrade for 2040	$ 50.00
4.0 O/S	O/S Upgrade for 40 Column	$ 100.00

*Asterisks indicate fall delivery—all others are immediately available

SPECIAL OFFER ON CBM COMPATIBLE BUSINESS SOFTWARE!

Purchasing software has always been difficult due to the "you buy it - you own it" attitude of most vendors. We at NEECO, recognize this problem and can now, on all of the Software Packages listed, offer a full 30 day refund policy to NEECO's customers. Now you can purchase with confidence. Buy it - try it; if the program package is not suitable for any reason, send it back to us within 30 days and we will refund the full purchase price—less shipping charges!

NEW!

SOFTWARE	APPLICATION	REQUIRES	AUTHOR	AVAILABILITY	PRICE
Word Pro I	Word Processing	8K + cassette	Professional Software	Immediate	$ 29.95
Word Pro II	"	10K + 2040	"	"	99.95
Word Pro III Plus	"	32K + 2040	"	"	395.00
Word Pro IV Plus	"	8032 + 2040/8050	"	"	595.00
BPI Integrated G/L	Business	32K/8032 + 2040	BPI	"	360.00
BPI Inventory	"	"	"	"	T B A
BPI Payroll	"	"	"	"	
BPI Enhanced A/R	"	"	"	"	
CMS G/L	"	"	CMS Software	"	295.00
CMS A/R	"	"	"	"	195.00
CMS A/P	"	"	"	"	195.00
CMS Customer Mail List	"	"	"	"	195.00
CMS Payroll	"	"	"	"	350.00
Datasource 1	All Business	32K/8032 + 2050/8050	BMB	August/Sept	295.00

*Wordprocessing Software requires output printer. We recommend the NEC Spinwriter ($2995) for letter quality.
*PET is a registered trademark of Commodore Business Machines. Small Keyboard PETS require a ROM Retrofit Kit.
Multi-Cluster is available in Canada from BMB Compu Science, P.O. BOX 121, Milton, Ontario, L9T2Y3
All prices and specifications are subject to change without notice.

NEECO
679 HIGHLAND AVE.
NEEDHAM, MA 02194

NEW ENGLAND ELECTRONICS CO., INC.
"NEW ENGLAND's Largest Computer Showroom"

(617) 449-1760
MASTERCHARGE OR VISA ACCEPTED
TELEX NUMBER 951021, NEECO
MON-FRI, 9:00-5:30

NEECO announces the MIPLOT Plotter from WATANABE

MIPLOT WX4671

	Code(ASCII)	Name	Function
Vector commands	D	DRAW	Draw a straight line to the point specified by absolute coordinates
	I	RELATIVE DRAW	Draw a straight line to the point specified by relative coordinates
	M	MOVE	Move with pen up to the point specified by absolute coordinates
	R	RELATIVE MOVE	Move with pen up to the point specified by relative coordinates
	L	LINE TYPE	Specify solid or broken line
	B	LINE SCALE	Specify the pitch of a broken line (0.1 – 12.7mm)
	X	AXIS	Draw X or Y coordinate axis
	H	HOME	Return to the origin with the pen up
Character commands	S	ALPHA SCALE	Specify character size (1 to 16 times basic 0.7mm x 0.4mm)
	Q	ALPHA ROTATE	Specify character orientation (Four directional)
	P	PRINT	Draw ASCII code characters
	N	MARK	Draw mark centered on the pen position (5 x kinds)

- 26 cm X 36 cm Plotting area
- .1 mm Addressing Resolution
- Full ASCII in any of 15 different character sizes
- Uses standard felt tip pens

$1250.00*

Includes interface to CBM, Atari, Apple, or TRS•80**. Please specify.

Please CALL or WRITE for specifications and information.

*Atari is a registered trademark

**TRS•80 is a registered trademark of Tandy Corp.

ORIGINAL 8K PET 2001* OWNERS TAKE NOTE!

The following peripherals and accessories are IN STOCK AT NEECO:

1. AXIOM PRINTER

- Complete PET graphics
- Plug compatible
- Electrostatic paper
- 40 or 80 columns

$299.00

2. 16 or 24K EXPANDAMEM

INTERNAL MEMORY EXPANSION UNIT

- Plug compatible
- Dynamic low heat memory
- Proven reliability
- No adaptor needed

16K - $299.00
24K - $379.00

3. FULL SIZE KEYBOARD

- Complete PET graphics
- Separate keypad
- Plug compatible
- With cover

$99.95

*8K-2001 with original keyboard and built-in cassette

NEECO
679 HIGHLAND AVE.
NEEDHAM, MA 02194

NEW ENGLAND ELECTRONICS CO., INC.
"NEW ENGLAND's Largest
Computer Showroom"

(617) 449-1760
MASTERCHARGE OR VISA ACCEPTED
TELEX NUMBER 951021, NEECO
MON-FRI. 9:00-5:30, E.S.T.

Basically Useful BASIC

Financial Fuzzies

Jim Butterfield

Small computers are ideal for many small business applications. But a puzzling problem crops up which could cause pennies to occasionally disappear.

It's tied into the fact that almost all computers do their arithmetic in binary. This seems innocent enough until you realize that some fractions which are exact in decimal notation turn into an endless repeating fraction in binary. For example, .1 decimal translates into binary as .000110011 ...

You can see the nature of the problem instantly by typing on the PET: PRINT 8.13 .. the number which is actually printed is slightly different from 8.13. Or try PRINT 2.23-2.18 and see if the answer is the .05 you expect.

Don't feel too smug if you use a different make of computer and the above examples turn out all right. All binary number machines have this problem; it's there, waiting for you.

Almost all decimal fractions change to repeating binary numbers; and since you don't have infinite memory space, the number must be chopped off somewhere. Many Basics are very clever, and have rounding routines that trim the number upward or downward as seems best. Numbers are usually trimmed slightly before printing, which makes the error disappear. But the problem is still there. And the more numbers you add and subtract, the greater your error will be.

A Detailed Look

Here's a small Basic program which will allow you to look at numbers stored in your system.

```
100 INPUT"AMOUNT";A
110 B = INT(A) : C = A-B : ?A;" = ";B;".";
120 C = C*10 : D = INT(C) : C = C-D
130 PRINT D; : IF C>0 GOTO 120
140 PRINT
150 GOTO 100
```

Line 100 gets the input number. You may change this to calculate values for printing, if you wish.

Line 110 takes the integer part of the value and prints it, followed by a decimal point. Variable C becomes equal to the fractional part of the value.

Line 120 multiplies C by 10; the integer value is the next digit to be printed. For example, a value of .125 would become 1.25; we'd print the 1 and keep computing using the .25 as the next C value.

You may be surprised to find that the only financial fractions that work out exactly in binary are .00, .25, .50 and .75!

By playing around a little, you'll quickly spot the fact that integers never give any trouble. It's only the decimal fractions that misbehave.

How to fix it.

The quickest way around this problem is to round the numbers when you're printing them. If you have a PRINT USING statement on your computer, or an equivalent subroutine, it should round the numbers as it formats them.

A quick rounding formula is: X = INT(X*100 + .5)/100 which will give a value to the nearest cent.

Rounding isn't the best way, however. The error is still there - the value will seldom be exact. As you add and subtract more and more numbers, the error grows. It will be particularly noticeable for large numbers (amounts over $100,000, say), and eventually a penny may get away from you even using rounding.

The best method is to change all financial numbers to pennies, and work exclusively in integer pennies. $3.21 becomes 321 pennies, for example.

Convert the numbers at the time of input. For example:

```
340 INPUT "AMOUNT";A : A = INT(A*100 + .5)
```

During your computation, remember to round percentage calculations. Add .5 to round; don't add to drop the fractional penny:

```
470 T = INT(A*7/100 + .5) : REM 7% TAX, ROUNDED
480 S = INT(A/3) : REM ONE THIRD, DROP FRACTION
```

Note that T and S remain as exact pennies.

Finally, convert back to pennies just before printing:

```
760 PRINT A/100 : REM OUTPUT AS DOLLARS-AND-CENTS
```

©

Editor's Note:
B. J. Deemer's interesting article presents some general hints for keying in a long program. *Basically Useful BASIC* would be interested in other such efforts.

Spend Time, Save Money!

B. J. Deemer

For those of us who spent their last dime to purchase their micro, a couple of hints for hand entry of printed listings. As my micro is a PET, the procedure is described as done on that unit. But it is compatible with any micro.

First find a blank Cassette, a c-30 size is what I use. Next dig out that 200 line listing that you want to put on tape. I will use "HAT IN THE RING" from VOL. I, ISSUE 6, to explain the hints. For those of you who use this listing, a small change to make that program more interesting will be at the end of the article.

First hint, of course, everyone knows that you are supposed to save material every 20-25 lines in case of some malfunction...Right? Next hint, follow the computer through the program, and do the listing as the computer would. This means, when you come to a GOTO, GOSUB, READ, or any other statement that will cause the computer to jump elsewhere in the program, go to that point and enter what is necessary for the computer to do that branch.

Now lets start on the listing. First, find the highest numbered statement in the listing, and if it is not an end statement, add a statement that will end your program. In this case 19030END. Get your micro all set with the tape positioned properly to SAVE and start your entry with the end statement, then do lines 9-120. That is to the first branch, GOSUB19000. Go to line 19000 and enter until you encounter a RETURN. THIS is a good place to SAVE. On the PET SAVE"1", rewind the tape and VERIFY"1". Now if you get OK you have a portion of the listing that can be RUN. So enter RUN and see what your computer does. If you get any errors correct them now, and then do another SAVE and verify. Use progressive numbers and make pencil notations on the listing. You can also get the graphics centered at this time. Now go to line 130, where the computer was returning to and enter it. Here you will find two READ statements. They require DATA statements; so find the Data and enter it/them. In this case lines 7000-7210. Do the save routine again. Don't rewind your tape to the beginning, start it right where it stopped. Now you can run again and check the Data statements..note if you get a syntax error it is not in the data but in the READ or DIM statements.

Proceed through the listing, doing the SAVE, VERIFY, and RUN routine, correcting as you go. In this listing, I recommend this order: 150-285,: 4000-4060,:290-360,:365,1000-1070,:370,2000-2050,: 375,3000-3050,:380,4000-4060,:385,5000-5010,:390, 6000-6750..... Now if you have made all your corrections as you went along you have a running program, with no more errors.

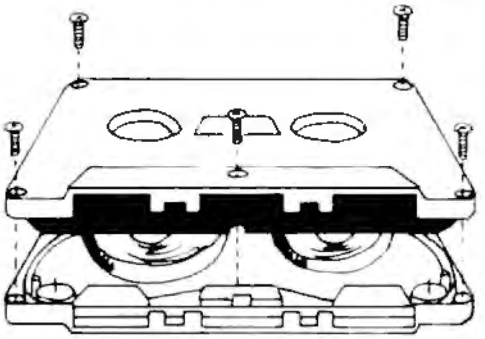
Now the last hint. Rewind your tape and set it to the beginning of the usable tape or end of the leader. Now do as many saves as it takes to fill one side of the tape, then rewind and do all of the verifys. Now you have a tape that doesn't need rewinding every time you want to load this program, just load from where it stopped, unless that was the last one. Now take the cassette out and look at the back edge, you will find a breakout tab near each corner, remove these and you will not accidentally erase your hard work. If you ever want to make a change you can put a small piece of tape over the hole and remove it afterwards.

For those of you who did enter "HAT IN THE RING": in lines 2050,3050,4060, and 5010--Delete the GOTO 150 and substitute this: PRINT "clr home":GOTO 320..... This change will allow a player to see the statistics, and then make a move, before his turn is over.

Now while I have put these on paper for your use I do not claim to be the originator of them, but do not recall seeing them in print recently, if ever, or if I just picked them up in conversations with other users. Happy computing! ©

Introducing the compact graphic printer that performs like the big guys.
Only $450
(plus shipping & handling)
Apple or PET interface available ... $80

Don't lose your message because of the medium...



RAMWARE™

CASSETTES

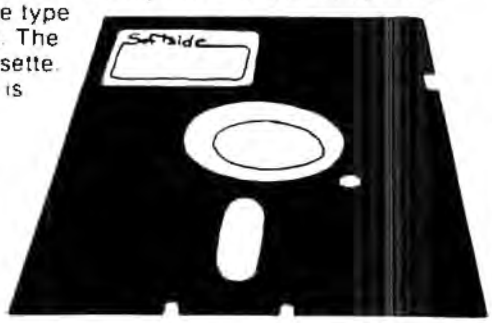
The cassette tapes used for recording data are composed of two parts: the cassette shell and the tape loaded into the shell. The shell can be either a 5-screw or sonic welded type with a non-magnetic leader or a magnetic leader (so called leaderless cassettes). The shell used in our cassettes is of premium quality 5-screw, with non-magnetic leader. The choice of non-magnetic leader may confuse some people, but there is a valid reason. There is a splice required to connect the magnetic tape to the leader at both ends of the tape. A person recording program material or data, using a leaderless tape, stands to drop a bit of data at the splice point. Not all leaderless tapes have the splice and you have to be very careful when buying this type of data tape. We use standard leader to avoid the confusion, and unhappy customers when the first recording on the tape is always bad.

The tape used in our cassettes is of studio quality. The same type of tape is used by some studios for making master recordings. The magnetic tape used in the cassette is the true heart of the cassette. You can have the best shell made, but with low quality tape it is still junk.

The cassettes offered here have been chosen for the highest quality components consistent with a practical cost level.

Cassettes come packaged in boxes of 10. They are offered in 10 and 20 minute lengths.

C-10	$6.95 + $1
C-20	$7.95 + $1



DISKETTES

We offer two levels of diskettes: certified and non-certified. The certified diskettes have been put through a test to check the entire working surface for bad spots. These diskettes are certified error-free by the manufacturer. If you require assurance of every diskette being perfect, then the Dysan certified diskette is for you.

The BASF company invented magnetic tape from which the very large and varied industry of today has grown. We offer the BASF premium quality (non-certified) Diskette. These diskettes enjoy one of the lowest reject rates of any manufacturer (all our disk-based software is duplicated on BASF).

We are also offering diskettes from 3-M SCOTCH. These come encased in a touch (PVC) jacket which resists handling damages. They are certified 100% error-free. Their low modulation provides better signal stability.

BASF:	
Box of 5	$19.95 + $1
Box of 10	$34.95 + $2
Box of 100	$299.00 + $3
3-M SCOTCH:	
Box of 10	$39.95 + $2
DYSAN:	
Box of 5	$29.95 + $1

The Software Exchange

6 South St. Milford, NH 03055

TO ORDER TOLL-FREE: (in NH call 673-5144)

1-800-258-1790

INTRODUCING *THE NEW IMPROVED* BUSINESS ENHANCEMENTS COMPUSERVICE BUSINESS SOFTWARE FOR COMMODORE AND APPLE COMPUTER SYSTEMS

Micro Mini Computer World Inc. is an execlusive distributor for the BUSINESS SOFTWARE developed by Business Enhancements Compuservice of Escondido, California.

If you are selling or using the COMMODORE BUSINESS MACHINES or the APPLE computer systems, then you should provide yourself and your customers with the MOST COST EFFECTIVE and COMPREHENSIVE business software for a business computer system.

CURRENT B.E.C. SOFTWARE

- General Ledger--Master File 1000 Accounts and Journal File 4400 Entries
- Accounts Receivable--Master File 1170 Accounts and Invoice File 1430 Entries
- Accounts Payable--Master File 1170 and Invoice File 1430 Entries
- Payroll--440 Employees
- Job Costing--1100 Items Per Disk
- Inventory--1100 Items Per Disk
- Mail List/Customer Information--1000 Entries Per Disk

Above figures apply to CBM 2001 computer system with 32K CPU and 2040 dual disk. With the new CBM 8050 Megabyte disk the volumes will be increased significantly.

B.E.C. SOFTWARE FEATURES:

- Complete and total documentation
- Step by step walk through on every program operation, with examples
- Each package is MENU driven and uses dynamic load and overlay once the initial menu is loaded.
- Examples are provided for all reports and other printed forms. All forms are available from New England Business Services Inc. (NEBS).
- All input/output operations use random access
- Sorts are machine language sorts
- Programs are interactive with the General Ledger and update the GL automatically.

B.E.C. VALUE ADDED BENEFITS

- Total commitment to the development of excellent business software for the COMMODORE and APPLE computer systems.
- At reasonable rates Micro Mini Computer World Inc. will provide software modifications to meet customer requirements. (Call MMCWI for further information)
- EXTENDED WARRANTY which entitles users to any enhancements to accounting software during the year of coverage. (Cost is $100 per year)

Dealers and Interested Parties may obtain a copy of the B.E.C. software documentation for $25. If after review you are not interested, send the documentation back, in re-saleable condition, for a full refund or apply the $25 toward your first software purchase.

The NEW B.E.C. BUSINESS SOFTWARE requires a special ROM chip for proper operation.

Suggested Retail Prices are:

1. Rom chip $70 (required on any software package)
2. Individual software package $150
3. All seven software packages $995 (save $55)

DEALER INQUIRIES ARE INVITED

micro mini COMPUTER WORLD Inc.

TOWN & COUNTRY SHOPPING CENTER

74 ROBINWOOD AVE. COLUMBUS, OHIO 43213
(614) 235-5813 (614) 235-6058

Micros With The Handicapped

Susan Semancik,
The Delmarva Computer Club

Our activities at the Marine Science Center (see Issue #7) also involved the testing of several devices that we hope will have further applictions, not only for the handicapped, but for any serious computer user. One complication that arises when trying to use alternative inputs for the handicapped, is that only one device at a time can have access to the eight data lines of the User Port. When working with the blind, especially, it is often desirable to have not only an alternative input device, but also a vocal output device controlled by the computer. It is possible on the User Port to simultaneously assign some of the data lines as input and some as output, but this is not sufficient if eight lines are needed in each direction.

What we tried, and have had success with, is a simple, inexpensive device that converts the User Port's edge into a dual edge. One edge is used by an output device, and the other by an input device that are alternately serviced by the PET.

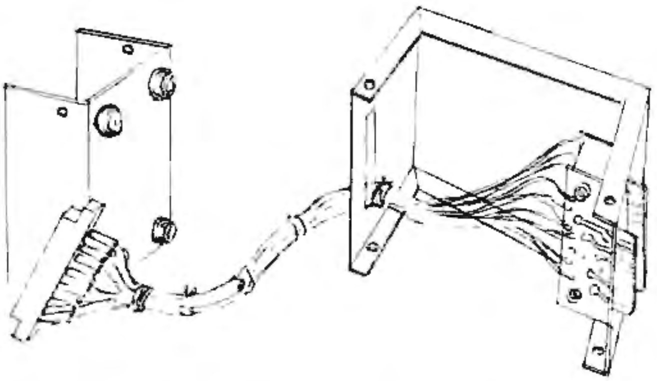
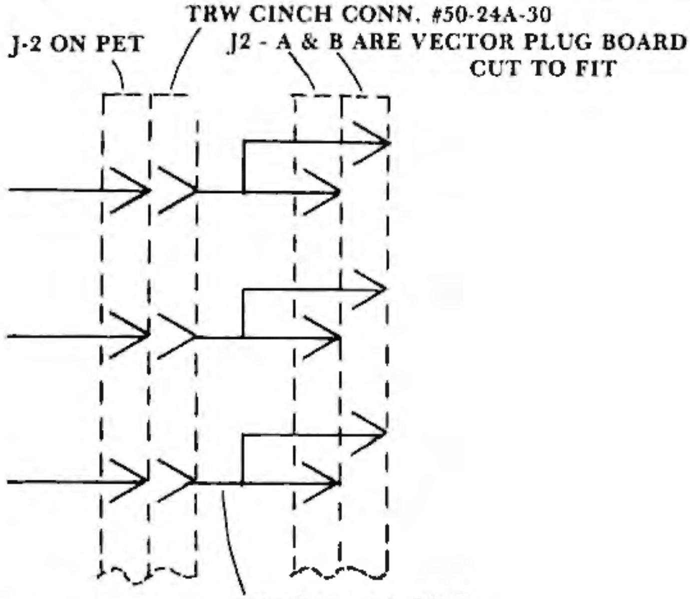


Diagram 1 shows how this device is connected to the PET, and Schematic 1 shows how it is wired. In diagram 1, the bottom of the metal box containing the wiring has four foot pads on it and is shown on the left side of the diagram, removed from the box, with an edge connector to the User Port in front of it. Each pin of this edge connector is connected with either #22 or #24 wire to two printed circuit cards that were cut from prototype boards that can be purchased at Radio Shack. Three typical pins from this connection are shown in Schematic 1, showing how the female connection from the PET is wired to the two male connection on the printed circuit cards. This device can be built for approximately $10. To use the device, plug its edge connector into the User Port, and attach the edge connectors from two of your peripherals to the exposed portions of the printed circuit cards. Make sure one peripheral will be used only as input and the other only as output, and that each uses TTL I/O Logic Levels and has buffered outputs.

The input device we chose to use is Innovision's Prestodigitizer Board, for which we have written programs to enter either Braille code or Morse code. (See Issues #5 and #6.) The output device we used is a modified TI Speak-and-Spell, whose keys can be controlled by a PET computer. We are currently investigating accessing the Speak-and-Spell's vocabulary by computer, and hope to eventually use it with the PET as an inexpensive speech synthesizer.


TRW CINCH CONN. #50-24A-30
J-2 ON PET
J2 - A & B ARE VECTOR PLUG BOARD CUT TO FIT
TYPICAL ALL PINS

Schematic 1

We modified the Braille program for the digitizer board so that as a letter is entered in Braille code on the board, the Speak-and-Spell vocalizes that letter. Because of the active, noisy environment, the students stayed close to the Speak-and-Spell's speaker. This system was especially appreciated by the blind students because they could use a code they were familiar with to enter messages on the PET's screen. This allowed communication one way between the blind who couldn't sign and the deaf who couldn't speak. The program will be modified further so that communication can flow the other way, too. That is, when the deaf type on the PET's keyboard, the letters will appear on the screen and be simultaneously vocalized. Certainly this second way could also be used by a blind person familiar with a typewriter keyboard. We would also like to modify this program so that a message entered from the PET's keyboard or from the digitizer board could be repeated with a single keystroke.

We gained many valuable insights from our experiences at the Marine Science Center's summer workshop for the handicapped. All of our programs were greeted by the intended users with a great deal of enthusiasm and positiveness. We hope to continue exploring these and other ideas we have for aiding the handicapped, and will share our experiences with you through this column.

We have heard from many readers who wish to encourage our efforts, and some who wish to support us in any way they can. Some have even offered to share their experiences as handicapped computer users with us through this column. We hope others will also volunteer to share their experiences so that a much needed dialog can be encouraged in this area with many problems and solutions. We would also like to encourage feedback from readers on programs we are offering here for the handicapped.

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

©

HARD WORKING SOFTWARE

for PET/CBM™ operating systems 1.0 to 3.0

MATRIC™ expands Commodore BASIC with fourteen new commands for handling arrays. Algebraic style syntax. Checks for conformability. Extended error messages.

The 5K machine language program lets you - - Display a matrix on the screen and change its values. Transfer data between matrices or fill a matrix with a constant. Transpose. Transfer diagonals between matrices, or from a matrix to a vector, from a vector to a matrix, or fill a diagonal with a constant. Do vector or matrix addition, subtraction, multiplication; elementwise multiplication, division, squares, and square roots. Inversion. Determinant. Eigenvalues and eigenvectors of a square, symmetric matrix.

Specify size and ROM set of your machine. Tape or disk, 32-page manual. Price: $125.

PRO-GRESS™ multiple regression BASIC program reads unlimited records from tapes or CBM disk. Up to 45 variables in 32K. Permits transformations. Provides means, standard deviations, correlations; R, R-square, F, degrees of freedom; constant and coefficients, betas, Student's t's. Output to screen, or to ASCII or CBM printer.

Manual and two programs. Tape: $45. Disk: $50.

TEXTCAST™ **II** 8K machine language word processor. Easy typing and screen editing. Produces ASCII files on tapes or disks. Prints with ASCII or CBM printer. Centers, underlines, right justifies, numbers pages. Creates data files for PRO-GRESS.

Old/New ROM versions of program, revised manual. Tape: $75. Disk: $80.

ORDER YOUR WORKERWARE FROM: T

Cognitive Products
P.O. Box 2592
Chapel Hill, NC 27514

Apple Disk Fixer

APPLE II
32K, DISK **13 OR 16 SECTOR**

If you care enough to back up critical programs and files, Disk Fixer™ will give additional peace of mind. This powerful utility for experienced Apple users is a tool kit for manipulating, repairing, and protecting all data on disk.

Use the high-speed full screen editor to examine and easily change any portion of a disk, correct space usage within files, and save money by locking out bad tracks on disks. Directories are alphabetized, if you choose.

The display and search capabilities show where specific HEX or ASCII data is located and you can modify any data including binary files. DOS 3.2, DOS 3.3 & LANGUAGE SYSTEM DISK

Written by Jeffrey P. Garbers
©1980 The Image Producers, Inc., All Rights Reserved

IMAGE COMPUTER PRODUCTS™
615 Academy Drive
Northbrook, IL 60062
312/564-5060

What?
You own
a PET and you
haven't received this
brand new catalogue?

...being ye compleat catalogue of peripherals available for your PET

Software.
Peripherals. Books.
Over 60 items. From
$1.00 to $1,250. 24 Pages.
Write to Skyles today for
your FREE catalogue.

Skyles Electric Works

Skyles Electric Works
231 E South Whisman Road
Mountain View, CA 94041

KIDS FOR COMPUTERS

Alan B. Walker

For two years, I had asked the school district in the small community where I teach for a personal computer for my fifth grade classroom. Although they were interested in the idea, the answer was always no. I could see their point, what would the people say if the school district 'wasted' the taxpayers money on an expensive 'toy' for some 'kids'. I tried to get a classroom computer by writing to the state and the federal government for grants. Still no luck.

I had purchased one of the first PETs on the market and for three years had been bringing it into my classroom periodically. Although student interest was unbelievable, it was always a hassle to cart it and some programs to the school and then take it home again. It was during one of the times that my computer was in the classroom that one of my students asked why we didn't get one for the classroom. I explained my dilemma to the class and it was brought up that they could raise the money themselves. That option had crossed my mind before, but I had dismissed it. How could a group of fifth graders raise enough money to purchase even one of the cheapest computers?

During the discussion that followed it became evident that the students were really excited about raising the money. We started talking about ways to raise money, and found that most of the ideas we came up with would only raise a small amount of money. After much discussion we came up with three ideas that had a good possibility of raising us a lot of money.

The first idea we called 'Operation Donation'. In this phase the students would go from door to door asking the people in the community for donations. We planned to run an ad in the local newspaper listing the people that gave over two dollars. There would be special recognition in the newspaper for people who gave $10, $20, $50, and $100 or more. Also for over $100 they would get their name engraved on the side of the computer.

We set up some guidelines on what the students should say and how they should act. The most important was to be friendly even if the people didn't give. Some rules for safety were also set up. The students had to get permission from their parents. They weren't to go out after dark. They were not to enter the person's house and the students were to go out with a partner. To keep track of where the students were to go, and where they had been for donations, we made a large map of the city.

The second idea was to go to the area service clubs and ask for donations. We wrote letters to the presidents of all the clubs and explained to them what we were trying to do. We told them that we would attend a meeting and answer any of their questions. We also promised to present a program on the computer after it arrived.

The third idea involved writing letters to everybody that we thought might give us a donation. We designed some letterhead to make the letters look official and came up with a base letter that the students could use as a guide to follow when writing their letters. Some research got us the addresses of over 100 foundations that give educational grants. We found addresses for about 200 corporations that we thought might be interested in our project. Included in this list of addresses were the addresses of all the major computer manufacturers.

We were now set to begin the campaign. The students came up with the name 'Kids for Computers' and elected officials to take care of the minor details and the paperwork. The local newspaper ran an article explaining what we were trying to do. An area business put 'SUPPORT KIDS FOR COMPUTERS' on their billboard. And 'Kids for Computers' got down to work.

We took two class periods and the students wrote letters. When you have 32 students writing letters, it doesn't take long to write A LOT of letters. The students also wrote letters in their free time at school and at home. We took another class period and came up with answers for any questions that they might be asked when they went out asking for donations.

'Operation Donation' was a real learning experience for the students. Not only did the students develop new social skills by talking and working with others, they also learned first hand about many of the people in our community. Most of the people were really kind and friendly even if they didn't donate.

Two weeks after the students began asking for donations it became evident that we were easily going to receive enough money for a personal computer. We had already collected almost $1000 and we hadn't received any responses from our letter writing campaign. Now 'Kids for Computers' had to make a decision as to which type of computer to purchase.

After much research and talking with the owners of some of the micro-computers available we narrowed the choice down to either a PET or an Apple. It was the opinion of the group that these computers were the most dependable and had the best features for the money.

Our final choice was the Commodore PET for two reasons. Commodore was offering their three computers for the price of two. We figured that we could buy one Apple and a color monitor for the

Feed your PET®

educational software from MICRO-ED

We specialize in programs with these features:

- Instructional lessons end with a summary of student performance.
- Our programs are attractive and motivating to students.
- We will gladly replace any tape that fails to load or run properly.
- They have been written by professional educators. Our main author is Thorwald Esbensen, named in 1980 as one of North America's top school executives by Executive Educator magazine.
- Our tapes are independent modules, each one a complete lesson in itself.

Unless otherwise specified, each tape can be purchased for $7.95

You may wish to order tapes by the MICRODOZEN. Any twelve $7.95 tapes can be purchased for $84.00.

Send for free catalogue:

MICRO-ED, Inc. • P.O. Box 24156 • Minneapolis, MN 55424
or telephone us at (612) 926-2292

All programs work with any 8K PET, old or new.

PET is the registered trademark for Commodore Business Machines, Santa Clara, CA.

same amount that we would have to pay to get three PETs. But what really turned us in favor of the PET was a call that we got from Commodore. The Commodore Educational Director called and asked for the president of 'Kids for Computers''. John, the President of 'Kids of Computers' had written a letter to Commodore asking for a donation. After talking with John, Commodore decided to donate a used PET to the group.

The local Kiwanis club donated $100, which put pressure on the other service clubs. And soon it looked as though buying two PETs would be possible. At about this time the letters from the corporations and foundations started coming in. Again, we were amazed by the response. Most sent back a nice personal letter explaining the reason that they couldn't donate. Some sent us cash donations; others donated items to help our cause. Texas Instruments sent us a calculator, General Mills donated a case of Cheerios, and Milton Bradley sent us one of their computer games. Because of a student letter, Boeing Computer Services in a nearby town decided to donate a computer to the group. They also decided to start a program 'Computers for Kids', in which they will donate a computer a year to different area elementary schools.

With the money we had collected we put in an order for the two Commodore PETs and within two weeks they arrived at the school. The one Commodore donated arrived soon after.

The students came up with some rules on when the computers could be used and by whom. It was decided that before a student could use a computer they had to learn the basics of loading and running a tape. The student would then take a test and if they passed the test they would earn a Computer Operator License. A student had to be finished with all his schoolwork, have a Computer License, and have permission of the teacher to use the computer. The license could be cancelled if the student broke any of the rules or abused the computers.

It didn't take long for all the students to get their licenses and it was amazing how they started getting all their schoolwork done. A schedule was set up that took advantage of every spare moment during the day. The students signed up on the schedule in a order chosen at random by the computer, and the students first to choose one week are the last to choose the next week.

I brought in all the computer programs that I had and the first two weeks with the computers was 'Computer Play'.

By playing the computer games, students learned more about the keyboard and overcame much of their fear of using the computer.

After the students had a chance to use the computer games and see what the computer could do, I began teaching them the basics of the BASIC language. Although the students still enjoyed playing the games, they started spending more and more time typing in prewritten programs and trying to write and debug programs of their own. The students had fun trying to get the computer to do what they wanted it to do.

As of this writing, we have had three computers in the classroom for two months and just received the fourth. We are expecting the fifth (from Boeing) in about two months. I have tried many classroom projects to build student interest in learning, but none have had the effect that the computers have had. Computers turn kids on to learning and school as a whole. Students like coming to school and enjoy the excitement that computers create. Since the first day that the computers arrived, I have been met at the door daily by students wanting to use the computers and at the end of the day, students use the computers until I leave.

Kids for Computers have many more plans on the drawing board. We are developing ways of giving other students in the school exposure to the computers. We are also trying to set up an evening class for interested parents, taught by Kids for Computers. Other plans include raising money for a printer and a floppy disk (the cassette recorder uses too much of a students valuable user time). Don't be surprised if someone knocks on your door and asks for a donation to 'Kids for Floppy Disks'!

If you are interested in starting a similar program and have some questions, contact:

KIDS FOR COMPUTERS
Attn: Alan Walker
Prosser Heights School
Alexander & Miller St.
Prosser, WA 99350

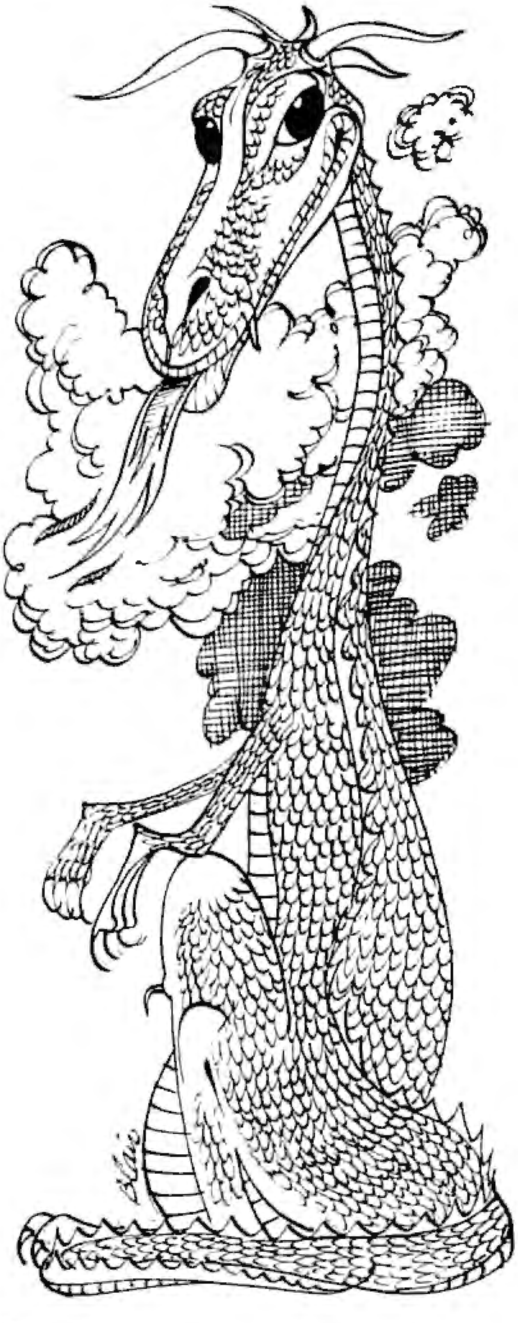
©

Part 1 of several . . .

The Mysterious And Unpredictable RND

Bob Albrecht and
George Firedrake

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



Preface

Bob Albrecht and his friend, George Firedrake, have joined together to bring you this *teach yourself* guide to the PET's world of random numbers and the RaNDom (RND) function. In this self contained description of how the function works, they lead you step-by-step through examples and exercises that help you learn the full capabilities of RND.

Using small programming examples, Bob and George explore with you a series of seemingly simple applications. Don't let the apparent simplicity fool you! The material that is covered deals with probabilities, sampling, general mathematics, proportions, and fantasy gaming.

For the classroom teacher, the booklet is a rich resource on how to present concepts using random numbers - *with or without* a computer. Bob and George, long time teachers of kids, show you how to imaginatively and creatively lead a person to an understanding of randomness without getting entrapped in jargon and complexity.

For the newcomer to programming, or anyone who knows how to program but doesn't have a lot of experience with the RND function, there are exercises and experiments to try. Of course the answers to the exercises are given in the back of the booklet, but Bob and George encourage you to work out your own solutions. There is an old and ancient dragonsong that George sings that tells you why he believes this to be best. A line from the song goes like this, "Work done by the Self is a lasting Work, and will last beyond all Time."

What does this booklet have to do with dragons? Well, that is quite a story, and only a small part can be told here. Bob Albrecht *is* known throughout the world, by both children and adults, as the Dragon. He is fond of dragon-things, and there are people who believe that he may indeed be a real dragon disguised for a while as a human, so that we will listen to what he has to say about kids and learning. Dragons are wise creatures but rather imposing when encountered on a one-to-one basis.

George Firedrake and Bob are old friends. They are said to have met and adventured together several thousand years ago. The chronicle of their escapades, according to George, forms the core of an elaborate dragonsong that takes at least 1000 nights to sing.

Recently awakening from a long sleep, George recalled a dream that he had experienced. In the dream, he had been given a vision of an important task to be done. He was to find Bob, renew their friendship, and begin an adventure that involved the children of the planet Earth. He and Bob were to help the children to acquire knowledge and mastery of the planet's new technologies.

When George found Bob, Bob was already at work on the task. Bob had dreamed the same dream.

All of this diversion probably still doesn't tell you what this booklet has to do with dragons. That's true. Dragons are often reluctant to divulge their larger plans. They like to let events unfold slowly. A few hundred years to a dragon is hardly any time at all.

The only clue Bob and George would give is that everyone should spend some time with the fantasy gaming and fantasy adventure portions of the booklet. They said that *many* kids already know about

ONE MILLION BYTES

THE FLOPPY DISK ALTERNATIVE

PET/BETA-1

If your data and program handling requirements are minimal, a mini-disk may be for you. If you want to access large amounts of data, program libraries, flexible user-defined data formating, and easy to understand documentation, CONSIDER the PET/BETA-1.

The PET/BETA-1 is a fast digital tape system incorporating the BETA-1 drive, with a flexible user-oriented operating system. The double-density drive has high-speed random access to over 1 million bytes per digital cassette with a transfer rate of 1K/second.

The PET/BETA-1 operating system extends your BASIC with 25 new commands executable from the keyboard or from a BASIC program. The new commands add sophisticated data and programming capabilities easily exceeding those of the Commodore disk.

For programs PET/BETA-1 offers program chaining and merging with dynamic memory management. Machine language files can be directly linked to user-defined commands, and any memory segment can be saved and relocated to any RAM location.

PET/BETA-1 offers sophisticated data management with userdefined record and field formats, up to 26 named fields per record, with true random access to any record in the file. These functions are available with easy to use commands.

With the PET/BETA-1 you control huge amounts of data with no operator intervention. The operating system supports up to 4 BETA-1 drives.

This is a complete hardware/software package, assembled and complete with all connectors and extensive user-oriented documentation. We provide technical assistance by phone and mail.

PET/BETA-1 System

Single density (512 k/drive)	**$555.00**
Double density (1 meg/drive)	**$700.00**

PET/CBM SOFTWARE

FORTH Interactive high level compiler and operating system 5–10 times faster than PET BASIC. High level block structured language. This is a true fig-FORTH implementation for 16 and 32K PETs.

fig-FORTH, with editor, assembler	**$50.00**
with floating point and strings	**$60.00**
with floating point, strings, macro-assembler	**$70.00**

cassette and disk interface available.

TERMINAL Turn your PET into an intelligent terminal with one of our terminal packages. These are complete assembled hardware and software packages. All include line editing resend, repeat key, shift lock, output to CBM printer, and more. Delivered on PET cassette with manuals. Inquire for modem prices.

PETTERM I All features above	**$ 80.00**
PETTERM II All features of I, plus local text editor with down-loading capability	90.00
PETTERM III All features of II, plus 80/132 column scrolling window for viewing formatted outputs wider than 40 columns	100.00

FSS

Software for small computers

1903 Rio Grande, Austin, Texas 78705
P.O.B. 8403, Austin, Texas 78712
1-512-477-2207

FSS is an authorized MECA Inc. dealer
Texas residents add 5% sales tax
PET is a registered trademark of Commodore Inc.

fantasy gaming, and that the ability to play fantasy games and experience fantasy adventures was important to what the dragons were here for this time. In fact, George confided that the fantasy gaming material presented in this booklet is just part of an entire series that the two dragons have in preparation.

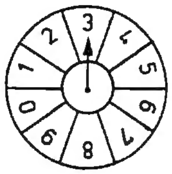
As always, even a dragon's musings about randomness and random numbers is anything but random.

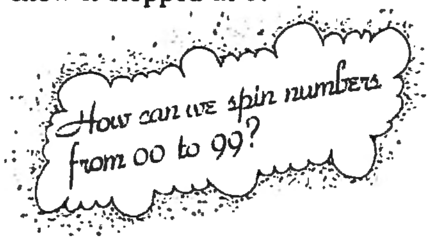
Ramon Zamora

Random Numbers

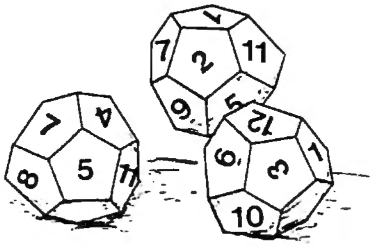
Random numbers? A random number is a number chosen at random from a given set of numbers. Many board games include dice or a spinner for generating random numbers.

- Roll the dice - they come up 7. Move 7 spaces.
- Want a random number from 0 to 9? Spin the spinner. We show it stopped at 3.



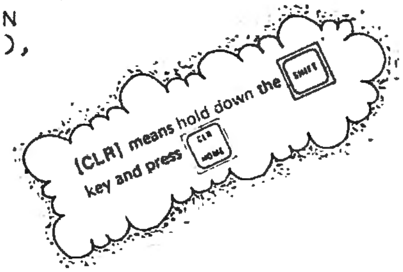


Have you played fantasy adventure games such as *Runequest* or *Dungeons and Dragons* or *Tunnels and Trolls?* These games use special dice to roll random numbers.



PET BASIC has a special function, called the RND functon, which makes it easy to compute numbers that seem to be chosen at random. Let's try some, using the following program.

```
1ØØ REM::::RANDOM NUMBER SAMPLER

11Ø PRINT "[CLR]";
12Ø INPUT "HOW MANY RANDOM NUMBERS" ; N
13Ø PRINT

2ØØ REM::::PRINT N RANDOM NUMBERS
21Ø FOR K = 1 TO N
22Ø   PRINT RND(1),
23Ø NEXT K
24Ø PRINT

999 END
```



We ran the program and asked for 10 random numbers. Here is what happened.

```
HOW MANY RANDOM NUMBERS? 1Ø

 .423845266          .Ø1Ø3Ø99732
 .22449Ø519          .177593567
 .Ø841791929         .Ø611899468
 .Ø1Ø9476861         .7Ø222491
 .815469Ø86          .549286173

READY
■
```

We got a sample of 10 random numbers. Let's get another sample of 10 random numbers. Again, we type RUN and press RETURN.

```
HOW MANY RANDOM NUMBERS? 1Ø

 .51Ø481989          .966ØØ4475
 .98410Ø1932         .577Ø93285
 .927432526          .73445306 2
 .234Ø26Ø46          .Ø876111783
 .936769Ø68          .614376942

READY
■
```

Aha! This bunch of numbers is different from the first bunch. Well, we told you that these are random numbers. In fact, if you put our program into your PET you will probably get yet another bunch of random numbers. That's the idea of random numbers. They are, well, random!

The statement: **220 PRINT RND(1),**
tells the PET to compute and print one random number. We use the number 1 in parentheses following RND. However, any positive number is oK.

RND (   )
Put any positive number here.

EXPERIMENT! What happens if you put zero or a negative number in parentheses. Try it and find out.

From now on, we will call the numbers computed by the RND function RND numbers. We will suggest experiments for you to try to find out things about RND numbers and use RND numbers in programs to simulate, or imitate, events such as flipping coins or rolling dice.

How Big? How Small?

Look at the RND numbers in our two samples.

Exercise 1.

(a) What is the smallest RND number?

____

(b) What is the largest RND number?

____

Hmmm... how could we find the smallest or largest RND number in a large sample? For example, suppose we want to know the largest number in a sample of 1000 RND numbers? Sounds like a lot of

INTRODUCING

Teaching Tools:™ MICROCOMPUTER SERVICES

EDUCATIONAL SOFTWARE

Teaching Tools™ PROGRAMS ARE DESIGNED BY EDUCATIONAL PSYCHOLOGISTS AND EXTENSIVELY TESTED IN ELEMENTARY SCHOOL CLASSROOMS.

Teaching Tools™ PROGRAMS:

- **ADAPT THE COMPUTER TO THE CHILD**
  easy-to-follow instructions, readable displays, child-proof programs
- **FOLLOW SOUND EDUCATIONAL PRACTICES**
  immediate feedback, graphics to reinforce correct responses, informative feedback for incorrect responses, child-controlled pacing, attention-maintaining features
- **HAVE SPECIAL FEATURES FOR TEACHERS AND PARENTS**
  can be easily adapted to meet individual needs, designed for children to use without supervision, provide summary of child's performance

SPECIAL INTRODUCTORY OFFER
ADDITION AND SUBTRACTION PROGRAMS
NOW AVAILABLE FOR ALL PET* COMPUTERS
GUARANTEED TO BE THE BEST YOU'VE SEEN

The **ADDITION** program has 24 levels of difficulty ranging from problems with 2 single digit addends without carrying, to problems of 5 addends with 7 digits each. The **SUBTRACTION** program has 12 levels of difficulty ranging from single digit numbers without regrouping to 7 digit numbers with regrouping. **BOTH** programs can be easily changed to offer different levels of difficulty, number of problems, time limits, and feedback options.

UNDER DEVELOPMENT: MULTIPLICATION, DIVISION, LETTER AND NUMBER MATCHING, SPELLING, AND OTHER FINE PROGRAMS

SEND FOR FREE BROCHURE

ADDITION ALL LEVELS	Cassette—$20.00	Both Programs On
SUBTRACTION ALL LEVELS	Cassette—$20.00	One Floppy disk $40.00

MASTER CHARGE, VISA ACCEPTED. Send card number, expiration date, and signature.

(INCLUDE $1.00 POSTAGE AND HANDLING PER ORDER.)
ALL PROGRAMS WILL BE MAILED FIRST CLASS

MONEY BACK GUARANTEE

NORTH CAROLINA RESIDENTS ADD [illegible] SALES TAX

*PET is a trademark of Commodore Business Machines

Teaching Tools™ MICROCOMPUTER SERVICES

DEPT. CJ81 P.O. BOX 12679 RESEARCH TRIANGLE PARK
NORTH CAROLINA 27709 PHONE (919) 851-2374

Programs available in Canada at $24.00 (CDN) Per Cassette or $48.00 (CDN) Per Floppy Disk. (Includes $1.00 Postage And Handling Per Order.)
FROM: S.E.S. Computing Inc., 267 Bain Avenue, Toronto, Ontario
M4K-1G2 Phone (416) 463-5302

work! Well, that is just the kind of work that the PET loves to do.

Here is a program to generate a bunch of RND numbers and print only the largest number.

```
1ØØ REM***LARGEST RND NUMBER IN A SAMPLE

2ØØ REM***FIND OUT HOW BIG A SAMPLE
21Ø PRINT, "[CLR]" ;
22Ø PRINT
23Ø INPUT "HOW MANY RND NUMBERS" ; N

3ØØ REM***SET BIG EQUAL TO FIRST RND NUMBER
31Ø BIG = RND(1)

4ØØ REM***DO REST OF SAMPLE. COMPARE EACH RND
41Ø REM***NUMBER WITH BIG. IF BIGGER, REPLACE.
42Ø FOR K = 1 TO N-1
43Ø   X = RND(1)
44Ø   IF X>BIG THEN BIG = X
45Ø NEXT K

5ØØ REM***PRINT BIG AND GO BACK FOR MORE
51Ø PRINT "LARGEST NUMBER IN SAMPLE IS" BIG
52Ø GOTO 22Ø

999 END
```

We tried it. Here is what happened.

```
HOW MANY RND NUMBERS? 1Ø
LARGEST NUMBER IN SAMPLE IS .921818131

HOW MANY RND NUMBERS? 1Ø
LARGEST NUMBER IN SAMPLE IS .6749Ø3427

HOW MANY RND NUMBERS? 1ØØ
LARGEST NUMBER IN SAMPLE IS .9962Ø2748

HOW MANY RND NUMBERS? 1ØØØ
LARGEST NUMBER IN SAMPLE IS .997Ø77Ø99

HOW MANY RND NUMBERS? 1ØØØØ
LARGEST NUMBER IN SAMPLE IS .999999136

HOW MANY RND NUMBERS?■
```

Anyone want to try for 100000?

Well, there seems to be some evidence (not proof) that RND numbers are always less than 1. Do notice, though, that with a big sample such as 10000, the largest RND number sort of creeps up on 1.

Yes, .999999136 is close to 1.

Exercise 2. Smallest RND number in a sample. Your turn. Write a program to find and print the smallest RND number in a sample. We suggest you do it by modifying our program to find the largest RND number in a sample. Use SMALL instead of BIG. Later, SMALL and BIG will get together in the same program. In the meantime. . . .

When we ran our program to find the smallest number in a sample, here is what happened.

```
HOW MANY RND NUMBERS? 1Ø
SMALLEST NUMBER IN SAMPLE IS .1526Ø9562

HOW MANY RND NUMBERS? 1Ø
SMALLEST NUMBER IN SAMPLE IS .Ø962136245

HOW MANY RND NUMBERS? 1ØØ
SAMLLEST NUMBER IN SAMPLE IS Ø.132498463

HOW MANY RND NUMBERS? 1ØØ
SMALLEST NUMBER IN SAMPLE IS 5.48253455E-Ø3

HOW MANY RND NUMBERS? 1ØØØ
SMALLEST NUMBER IN SAMPLE IS 1.17754321E-Ø3

HOW MANY RND NUMBERS? 1ØØØØ
SMALLEST NUMBER IN SAMPLE IS 7.84862186E-Ø6

HOW MANY RND NUMBERS?
```

and so on. Take over!

The smallest number can be very small. But, according to the above evidence, it is never zero. Beware! Evidence is not proof.

However, if you run lots of big samples and the smallest number is usually close to zero (but never zero) and the largest number is usually close to one (but never one), you begin to feel secure in making a conjecture, such as the following.

Conjecture 1. RND numbers are greater than zero and less than one. Or to put it into more math-like jargon,

$$0 < \text{RNDA}(1) < 1$$

which says "RND(1) is greater than zero and less than one."

And all this stuff leads into our next program, something for you to do.

Exercise 3.. The small and the big.

Write a program to find and print both the smallest number and the largest number in a sample of RND numbers.

A RUN might look like this.

```
HOW MANY RND NUMBERS? 1Ø
SMALLEST NUMBER IN SAMPLE IS .Ø1Ø3Ø9732
LARGEST NUMBER IN SAMPLE IS  .849286173

HOW MANY RND NUMBERS? 1ØØ
SMALLEST NUMBER IN SAMPLE IS .Ø1533Ø1686
LARGEST NUMBER IN SAMPLE IS  .9841Ø1932

HOW MANY RND NUMBERS? 1ØØØ
SMALLEST NUMBER IN SAMPLE IS 4.234Ø8483E-Ø4
LARGEST NUMBER IN SAMPLE IS  .995799597

HOW MANY RND NUMBERS?■
```

©

Cursor Classifications Revisited

Marlene R. Pratto

Editor Note: Although Marlene Pratto is no longer a COMPUTE! columnist, she continues to serve as a computing volunteer/aide at a local elementary school. This excellent summary of CURSOR magazine programs for the PET provides useful information for teachers currently using CURSOR.

Cursor magazine continues to be a source of inexpensive and quality programs for PET/CBM microcomputers. Cursor has completed its 22nd issue.

The Cursor programs are classified by grade level and by the skills the children may gain from using the programs. Some games may seem to be classified as appropriate for an age which may seem too young. I think, particularly, of RATRUN on issue 13. This is a simulation of a rat running through a maze seeking its reward of cheese. Only a portion of the maze is shown at any one time (which is all that would be visible to the rat in the actual maze). With this limited amount of information, the children have been able to play and to master this game.

Children are not always able to comprehend the written instructions provided by the program or the printed sheet that comes with Cursor, but the children are able to play these games and to play them well. They learn the rules and strategies by playing the games. GODZILLA is a good example of this.

I have classified some of the classic games such as GOMOKU and KALAH as DT, drill and tutor, since the playing of these games with the PET/CBM as programmed on Cursor can help the player develop playing strategies against a consistent player, the PET. The player may stop the game at any time and start over in order to test his or her new hypothesis about how to best play the game. When playing with a human, the play may be more random and a clear method of play is harder to see. I think that this is a good use of computer games since the student can begin to perceive the pattern of the play.

There is a wide range of programs available on Cursor. These programs have helped to maintain a lively interest in the microcomputers at Erwin Open School. The teachers have benefited from the Cursor programs by having classroom enchancements provided in a new and stimulating way. The children have developed new skills and have learned more about programming and the possible uses of computers.

Note: There are two different MATCH games as indicated in the table.

HE = hand-eye
DT = drill or tutor
LS = logical skills
PS = problem solving
FF = fun & familiarity

Level	Program	Issue	HE	DT	LS	PS	FF
K-2	Aliens!	16	•			•	•
	Bonzo!	16					•
	Dance!	21					•
	Drag	21		•			
	Dromeda!	18					•
	Frog!	19	•			•	•
	Letter	17		•			
	Music!	20					•
	Nab!	16	•			•	•
	Spot	17			•		
3-4	All of the above						
	Bat!	14	•			•	
	Bets	20			•		•
	Boswain	21			•	•	
	Capture!	21	•			•	
	Catch!	16	•			•	
	Checkers!	20		•		•	•
	Dungeon	15				•	•
	Everest	15				•	
	Ferry!	13	•			•	
	Fire!	16	•			•	
	Godzilla!	19				•	
	Hawaii!	15					•
	Joust	18	•				•
	Leap	13			•	•	
	Miner!	19				•	
	Ouranos!	21			•	•	
	Poker	22			•		
	Police!	17	•			•	
	Rail	19	•			•	
	Search	14		•	•		
5-6	All of the above						
	Cops	14				•	
	Fifteen	15			•		
	Gomoku	15		•	•		
	Hi-Res	18			•	•	
	Kalah	22		•	•		
	Match	22			•	•	
	Morse	14		•			
	Ratrun	13				•	
	Ruler	17		•			
	Sheep	18	•		•		
	Thunt!	22				•	•
7-8	All of the above						
	Curves	20		•			•
	Match	14			•	•	
	Weather	18			•		

©

SPECIAL: Additional 10% off on all CBM hardware

INTRODUCING ROM PET RABBIT OR CASSETTE

The PET RABBIT contains high-speed cassette routines, auto-repeat key feature, memory test, decimal to hex, hex to decimal, and other features. Coexists with the BASIC PROGRAMMERS TOOLKIT. Works with 3.0 ROMS (New) and new style cassette deck.

Cassette versions configured for $1800, $3000, $3800, $7000, and $7800. (3.0 ROMS only)
Cassette and manual – $29.95

ROM version configured to plug into P.C. board at $A000. (Specify 3.0 or 4.0 ROMS)
ROM and manual – $49.95

FREE ROM RABBIT with purchase of 8K PET and tape deck.
SPECIAL - ROM RABBIT and cassette deck – only $134.95

MACRO ASSEMBLER AND TEXT EDITOR

Now Atari

Macro and conditional assembly, string search and replace, 10 char./label, AUTO line numbering. MOVE, COPY, DELETE, NUMBER, and much more 20+ commands, and 20+ pseudo ops.

PET cassette version (ASSM/TED) – $49.95.
(Specify 2.0, 3.0, or 4.0 ROMS.)
PET disk version (MAE) – $169.95
(Specify 3.0, 4.0, or 8032)
ATARI cassette version with machine language monitor – $53.95

FREE ASSM/TED and ROM RABBIT with purchase of 32K PET and cassette deck.

FREE MAE with purchase of 32K PET and disk drive

TINY-C FOR PET

An adaptation of the TINY-C interpreter sold by Tiny-C Assoc. Useful for learning a modern structured programming language. Diskette – $50.00. Owners manual – $50.00

FREE MAE and TINY-C with purchase of 32K PET, disk drive, and printer.

COMPILERS

Graphics Drawing Compiler for PET and SYM. Works with Macro ASSM/TED. The GDC is composed of a number of macros which emulate a high-level graphics drawing language. In addition to the macros, GDC provides some very useful enhancements to the ASSM/TED. Manual and Cassette – $29.95.

Music and Sound Composer for PET. Works with Macro ASSM/ TED. The MSC is composed of a number of macros which emulate a high-level computer music language. In addition to the macros, MSC provides some very useful enhancements to the ASSM/TED. Manual and Cassette – $29.95.

I/O KIT

PET I/O Experimenters Kit. Allows easy access to IEEE or user port for the construction of external circuits. Kit – $39.95.

ORDERING TERMS

Send check or money order in U.S. dollars. Add 2% for postage for CBM orders. Overseas software orders add $5.00. All software mailed free in USA and Canada. Purchase orders acceptable.

EHS IS NOW A COMMODORE DEALER

EHS offers a number of software products for PET, ATARI, APPLE, and other 6502 computers. Now we sell CBM hardware. If you're in the market for PET products, be sure to look for our FREE software offers. Note: Be sure and deduct 10%.

CBM	PRODUCT DESCRIPTION	PRICE
4001-8KN	8K RAM-Graphics Keyboard	$ 795.00
4001-32KN	32K RAM-Graphics Keyboard	$1295.00
8032	32K RAM 80 Col.-4.0 O/S	$1795.00
2022	Tractor Feed Printer	$ 795.00
4040	Dual Floppy-343K-DOS 2.1	$1295.00
8050	Dual Floppy-974K-DOS 2.0	$1695.00
C2N Cassette	External Cassette Drive	$ 95.00
CBM to IEEE	CBM to 1st IEEE Peripheral	$ 39.95
IEEE to IEEE	CBM to 2nd IEEE Peripheral	$ 49.95
8010	IEEE 300 Baud Modem	$ 395.00
2.0 DOS	DOS Upgrade for 2040	$ 80.00
4.0 O/S	O/S Upgrade for 40 Column	$ 110.00
VIC Computer	New Commodore	$ Call or Write

EDUCATIONAL DISCOUNTS

BUY 2 — GET 1 FREE

TRAP 65

TRAP 65 prevents the 6502 from executing unimplemented instructions. Have you ever had your system to crash on a bad upcode? This is a real machine language debugging tool and time saver. Also useful for teaching trap vectoring and extension of instruction set in schools. 3½ X 4¾ printed circuit board which plugs into 6502 socket of any PET, APPLE, SYM. Only $149.95.

ATARI M.L. MONITOR

Load and save binary data on cassette. Display and change 6502 registers. Will coexist with BASIC. Monitor uses the screen editing capabilities of the ATARI to allow easy use. Cassette and manual – $9.95 (specify memory size).

ATARI MEMORY TEST

When you purchase a new ATARI or add on new RAM modules, you need to be sure that the memory is working properly. (Remember, you only have a short guarantee on your memory!) Cassette and manual – $6.95.

APPLE PRODUCTS

Macro ASSM/TED – includes manual, on cassette – $49.95, on disk – $55.95
Apple MAE – similar to PET MAE. A powerful assembly development system on diskette for 48K APPLE II or plus. (Requires license agreement) – $169.95.
PIG PEN – 100% M.L. word processor for use with Apple ASSM/TED. Fast text formatting, vertical and horizontal margins, right and left justification, centering, titles, foots, shapes, etc. Manual and source included, on cassette – $40.00, on diskette – $45.00

Apple Mail List System. Provides sorting on zip code or last name. Approximately 1000 names/diskette. Manual and Diskette – $34.95.

EASTERN HOUSE SOFTWARE

3239 Linda Drive, Winston-Salem, N.C. 27106
Ph. Orders – 9-4 EST (919) 924-2889
After 4 pm 748-8846
Send SASEfor free catalog

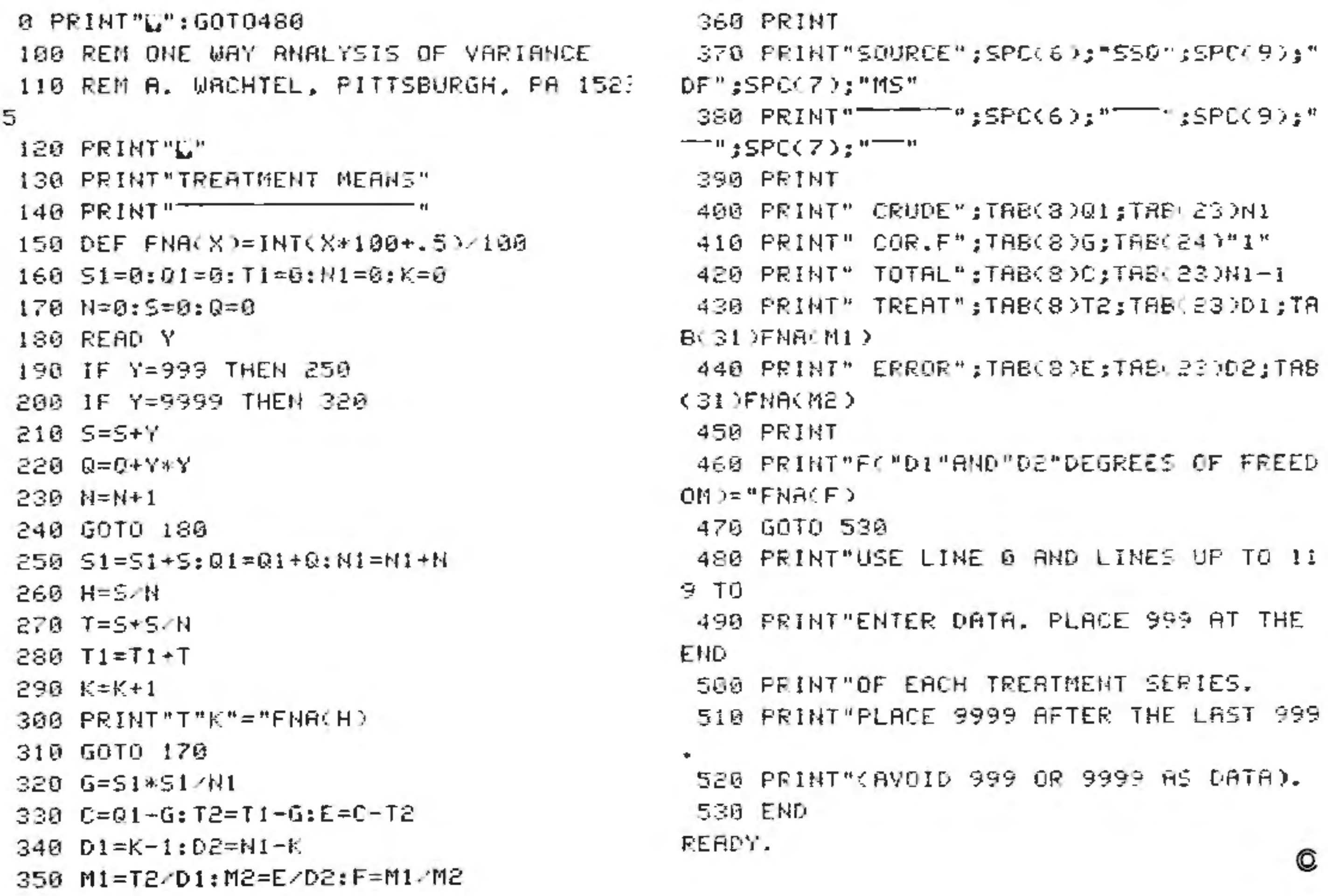
Stat Lab

Analysis of Variance

A. Wachtel

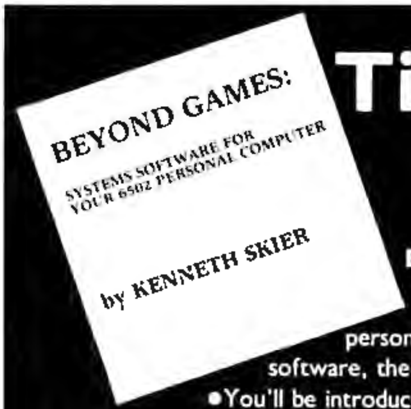
ANOVA is an acronym for Analysis of Variance. It calculates the F-statistic which is the ratio of variances between the data of individual "treatment" groups and the means which belong to these groups. Depending on the "degree of freedom" (DF) associated with each, a confidence limit can be obtained from tables of F-values. The "1" stands for "one way" which means that different tratments are compared with each other under presumably equal (more accurately randomized) conditions. This is not the place to present an exhaustive treatment on ANOVA. Suppose, however, that you wished to compare the average prices of a few different makes of cars, and went to several dealers. Enter the prices for each make followed by 999, then go on to the next make. Finally, enter 999,9999. Type RUN and obtain the F-statistic for x and y degrees of freedom where x = number of makes -1, and y = total number of data -1. Suppose now, that you wish to be 95% confident of these results. Look up a table for F at the 95% confidence level and go into x and y degrees of freedom. Compare the F value obtained by the program with that shown in that place in the table. If it is higher, then you can be 95% sure that the difference between any two means (average prices) for different makes is real rather than due to the differences charged by different dealers for a particular make.

In this program, I have provided for the entry of the data in DATA statements rather than by INPUT (or worse, INPUT#). DATA become part of the program and are therefore portable. They can be edited and appended as needed. I strongly believe that counting is a job for computers, not people, so you won't have to create a FOR-NEXT loop for an array to permit you to correct an input error which usually occurs near the end of a long series of entries. There is lots of room, and if you need more, you can provide it by renumbering the program. Instead of using tables, the confidence limit (percentile) can also be obtained from a program by Lon Poole and Mary Borchers ("Some Common Basic Programs") - Osborne Associates.



```
0 PRINT"▙":GOTO480
 100 REM ONE WAY ANALYSIS OF VARIANCE
 110 REM A. WACHTEL, PITTSBURGH, PA 1523
5
 120 PRINT"▙"
 130 PRINT"TREATMENT MEANS"
 140 PRINT"───────────────────"
 150 DEF FNA(X)=INT(X*100+.5)/100
 160 S1=0:Q1=0:T1=0:N1=0:K=0
 170 N=0:S=0:Q=0
 180 READ Y
 190 IF Y=999 THEN 250
 200 IF Y=9999 THEN 320
 210 S=S+Y
 220 Q=Q+Y*Y
 230 N=N+1
 240 GOTO 180
 250 S1=S1+S:Q1=Q1+Q:N1=N1+N
 260 H=S/N
 270 T=S*S/N
 280 T1=T1+T
 290 K=K+1
 300 PRINT"T"K"="FNA(H)
 310 GOTO 170
 320 G=S1*S1/N1
 330 C=Q1-G:T2=T1-G:E=C-T2
 340 D1=K-1:D2=N1-K
 350 M1=T2/D1:M2=E/D2:F=M1/M2
 360 PRINT
 370 PRINT"SOURCE";SPC(6);"SSQ";SPC(9);"
DF";SPC(7);"MS"
 380 PRINT"──────";SPC(6);"───";SPC(9);"
──";SPC(7);"──"
 390 PRINT
 400 PRINT" CRUDE";TAB(8)Q1;TAB(23)N1
 410 PRINT" COR.F";TAB(8)G;TAB(24)"1"
 420 PRINT" TOTAL";TAB(8)C;TAB(23)N1-1
 430 PRINT" TREAT";TAB(8)T2;TAB(23)D1;TA
B(31)FNA(M1)
 440 PRINT" ERROR";TAB(8)E;TAB(23)D2;TAB
(31)FNA(M2)
 450 PRINT
 460 PRINT"F("D1"AND"D2"DEGREES OF FREED
OM)="FNA(F)
 470 GOTO 530
 480 PRINT"USE LINE 0 AND LINES UP TO 11
9 TO
 490 PRINT"ENTER DATA. PLACE 999 AT THE
END
 500 PRINT"OF EACH TREATMENT SERIES.
 510 PRINT"PLACE 9999 AFTER THE LAST 999
.
 520 PRINT"(AVOID 999 OR 9999 AS DATA).
 530 END
READY.
```

©



BEYOND GAMES:

SYSTEMS SOFTWARE FOR YOUR 6502 PERSONAL COMPUTER

by KENNETH SKIER

Tired of playing games?

If you're serious about personal computing, here's the book for you: **Beyond Games: Systems Software for Your 6502 Personal Computer.**

Written for owners of Apple, Atari, Commodore, OSI, and Panasonic/Quasar personal computers, this indispensable guidebook is a self-contained course in systems software, the "other side" of your computer that lets you take advantage of its full power.

- You'll be introduced to the 6502 microprocessor and assembly language programming.
- Learn about structured programming and top-down design.
- Learn how to add an extended monitor, disassembler, text editor, and hexadecimal dump routine to your system.

Written by one of the experts in the field, **Beyond Games** is sure to become one of the most useful tools in your software library.

This and other BYTE/McGraw-Hill
books are available from BYTE
BOOKS or your local computer store

ISBN 0-07-057860-5
Price $14.95

Available in Spring

Please send □ __________ copies of **Beyond Games: Systems Software for Your 6502 Personal Computer**

Name Title Company

Street City State/Province Code

□ Check enclosed in the amount of $__________
□ Bill Visa □ Bill Master Charge
Card No. ______________ Exp. Date _______
Add 75¢ per book to cover postage and handling.
Please remit in U.S. funds or draw on a U.S. Bank.

BYTE BOOKS
Bill
70 Main St.
Peterborough, NH 03458

Active Electronic Sales Corp.

Your Source For 6502 Support

MICROPROCESSORS

6502	1 mhz CPU	$ 9.45
6502A	1.5 mhz CPU	12.95
6520	PIA	6.45
6522	VIA	9.95
6532	RAM 1/0	13.95
6551	ACIA	13.95
6850	ACIA	3.50

UARTS

COM8017/AY3-1015
40 khz Single 5V 3.95

6502 MICROCOMPUTERS

SYM-1 Single Board $229

SUPER SPECIAL: AIM-65 with 4K Memory at 1K Price: $425

6502 BOOKS

6502 Assembly Language Programming	$9.59
6502 Software Design (Rockwell Aim-65)	8.95
6502 Applications Book	14.04
6502 Games Book	12.56
Programming the 6502	13.30

MEMORIES

STATIC

2114L (Low Power 1K x 4) 300NS	3.45
6514 (CMOS 1K x 4) 450NS	7.95

DYNAMIC

4116-3 (16K x 1) 200NS	3.95
4116-5 (16K x 1) 300NS	2.95

EPROMS

2532	(4K x 8) 450NS, Single 5V, TI Pinout	24.95
2732	(4K x 8) 450NS, Single 5V, Intel	24.95
2716	(2K x 8) 450NS, Single 5V	7.95
2758	(1K x 8) 450NS, Single 5V	6.95
2708	(1K x 8) 450NS, 3 power	4.95

BUILD-IT-YOURSELF VECTOR

4112-4DP	4.5 x 6.5" Plugboard (25 16 Pin Dip)	$16.58
3682-2DP	4.5 x 6.5" Plugboard (20 16 Pin Dip)	13.25
32X1DP	Positive Photo Resist Kit	19.90
R407DP	Rub-on Transfers, Etch directly or make Mylar Film Master	3.95
ETCHANT	Crystal, Instructions Makes 4 Litres (Approx. 1 gal.)	16.95
H411121-22	22/44 Pin Solder Tail Edge Connector	1.84
H431121-50	50/100 Pin Solder Tail Edge Connector	3.30

SOCKETS

SOLDER TAIL		WIRE WRAP	
C850801	.07	C810804	.29
C851401	.11	C811404	.47
C851601	.13	C811604	.54
C851801	.17	C811804	.56
C852001	.19	C812004	.73
C852401	.23	C812404	1.11
C852801	.27	C812804	1.21
C854001	.39	C814004	1.70

SPECIAL: SPECIFY THIS AD WITH YOUR ORDER, AND RECEIVE FREE — ONE 6 x 4½" DOUBLESIDED PC BOARD LIMIT ONE PER CUSTOMER

ALL PRIME MAJOR BRAND NEW PARTS

Active Electronic Sales Corp.

MAIL ORDERS SHOULD BE SENT TO:

U.S.A. P.O. Box 1035 Framingham, Massachusetts 01701
Telephone Orders & Inquiries: (617) 879-0077
CANADA & FOREIGN 5651 Ferrier St., Montreal, Quebec H4P 2K5, Canada

Foreign customers please remit payment on an international bank draft or international postal money order payable in U.S. dollars.

Prices are in U.S. dollars. Minimum Order: $10.00
Add $3.00 to cover Postage & Handling

VISA AND MASTER CARD ACCEPTED

Why

PEDISK ?

FLOPPY DISK SYSTEM FOR PET, AIM, SYM, KIM

FAST – loads programs at least 3 times faster than 2040. It is the fastest disk available for PET.

VERSATILE – With its built-in S100 expansion chassis, any peripheral imaginable can be added to your computer.

SOPHISTICATED – The CRS/DOS op system provides advanced file handling, indexed-sequential, but very easy to use.

RELIABLE – single chip disk controller using IBM 3740 recording format gives maximum timing margins + super reliability. Auto error recovery.

SIMPLE – An enhanced BASIC command set makes PEDISK very easy to use. !LOAD, !SAVE, !OPEN, !CLOSE, !RUN, etc. No initialization required.

COMPATIBLE – read diskettes from other systems – 6502PDS, AIM, SYM, and even RADIO SHACK (special software). IBM 3740 soft sector single density.

PEDISK PACKAGE 1 . . . . $799.95
5" DISK SYSTEM, CASE AND POWER SUPPLY
PEDISK PACKAGE 2 . . . . $895.00
5" DISK SYSTEM, S100 CARD CAGE, CASE AND POWER SUPPLY
PEDISK PACKAGE 4 . . . . $1495.00
8" DISK SYSTEM, S100 CARD CAGE, CASE AND POWER-SUPPLY

EXS100 DISK CONTROLLER BOARD $49.95
BARE BOARD
EXS100 DISK CONTROLLER KIT $225.00
AIM, SYM, KIM ADAPTOR KIT $25.00
CRS/PDOS SOFTWARE SYSTEM $75.00
SPECIFY OLD OR NEW ROMS, MEMORY SIZE 8K, 16K 32K

Introducing . . .

6502PDS !

6502 PROFESSIONAL DEVELOPMENT SYSTEM

6502PDS: MINI $2695.00

The 6502 PDS is a versatile multi-card microcomputer designed and programmed for professional engineering and program development work, scientific computing, and general processing. This system provides the maximum in capability at the lowest possible cost by utilizing the industries must widely used computer bus - the S100. With a choice of over 500 peripherals including telephone interface, speech synthesizers, vocoders, and even associate memory, the potential end use is unlimited. The 6502 PDS is housed in a sturdy S100 mainfrain containing the 6502 MPU, Multiple I/O Card, RAM, and Disk Controller Board. This leaves room for future expansion. The system can be connected to any RS232 terminal or used with the optional internal Video Board.

6502PDS: 8" DISK $3495.00

Standard 6502 PDS except that full size 8" floppy disks are included instead of minifloppy disks. The S100 Disk Controller can handle up to four 8" disk drives. At 250 Kbytes of capacity each, the total system capacity could reach 1 Megabyte.

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, 6502PDS, COMMODORE 2040 DISK $65.00

The Spacemaker

*SPACEMAKER. $29.00

Spacemaker is a utility device for new PETs which allows user selection between ROMs that occupy the same address space. Spacemaker is a verticle 4.25 inch by 2.5 inch board that plugs into a ROM expansion socket inside the PET. The user simply installs a pair of conflicting address ROMs on Spacemaker, plugs it into the appropriate socket, and is up and running. ROM selection is provided by flipping a switch.

USER I/O $12.95

This option for Spacemaker allows the user software selection and switching of ROMs. User I/O consists of a user port connector, specially designed Spacemaker jumpers for connection to Spacemaker, and our utility software Spacectl supplied on Commodore or PEDISK diskett.

*NEED MORE ROM ROOM?

ROMDRIVER $39.00

A second mode of Spacemaker provides greater flexibility. with ROMDRIVER, you can control ROM switching and selection from software control. PET owners can daisy chain Spacemakers together, selecting (under software control) one of two, one of four, even one of six ROMs.

ROM I/O $9.95

A utility software package for ROMDRIVER owners allowing software switching of ROMS. Available on commodore or PEDISK diskettes.

FOR INFORMATION, SEE YOUR DEALER OR:

cgrs MICROTECH

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

ETC ELECTRONIC TECHNOLOGY CORPORATION

BATTERY BACKUP FOR PET/CBM

BACKPACK ™

Designed to install within the CPU Case, **BACKPACK** gives 6 to 10 minutes of full power to the computer (32K, CRT, Tape Drive) during times of power shortages and gliches in power lines thus allowing the user time enough to save data or programs to tape (or disk with **FLOPPY BACKPACK**). Batteries are recharged from CPU's own power supply and **BACKPACK** requires no wiring changes to install.

Single Quanity $225.00

FLOPPY BACKPACK ™

FLOPPY BACKPACK is a battery backup system for the Commodore Dual Floppy Disk Drive. Used in conjunction with **BACKPACK** for the pet, the user can save data and programs during power shortages. It also eliminates possible disk crashes. Designed to install within the disk cabinet with no wiring changes, the batteries are recharged from the disk's own power supply.

Single Quanity $135.00

AIM-65 POWER SUPPLY (with battery backup) & ENCLOSURE

AIM POWER 4 ™

AIM POWER 4 is a total power supply with rechargeable battery backup for the Rockwell AIM 65. **AIM POWER 4** supplies full voltage output for the AIM under normal operating mode and with batteries can keep the AIM up and running for one hour. Batteries are recharged during normal operation. **AIM POWER 4** is on a single board and will mount inside the **AME 1** enclosure (from ETC), and Enclosure Group Products.

Single Quanity $250.00

AME 1 ™

AME 1 is a totally metal enclosure of lightweight aluminum construction. Designed to house the AIM 65 and a power supply, with easy access to switches and connectors. **AME 1** will have an integral reset button, easy paper feed and totally visible 20 character LED readout. Designed for industrial applications, but has professional appearance.

Single Quanity $100.00

FOR MORE INFORMATION OR TO ORDER BY PHONE: (919) 362-4200 OR (919) 362-5671
INCLUDE $5.00 FOR SHIPPING AND HANDLING • VISA AND MASTERCHARGE ACCEPTED
DEALER INQUIRIES INVITED

All Products are shipped fully assembled and Ready to Install • Instructions Included.

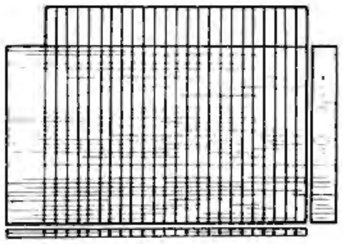
P.O. Box G, Apex, North Carolina 27502
(919) 362-4200

Etcetera International Software

COL U pad ™
THE ELECTRONIC COLUMNAR PAD

A Problem Solving Computer Software Program

COLUPAD™ is a unique computer tool which allows you to create, store, selectively retrieve and perform math operations upon data and then generate reports using that data. COLUPAD™ can best be visualized as the typical columnar accounting pad found in most offices. Powerful, but simple to use, COLUPAD™ has such features as:



- 24 columns and 40 rows for data manipulation
- Space for row and column descriptions
- Ability to reference decimal places for each column
- Ability to perform math functions on one column and store in another
- Ability to add, subtract, multiply, etc., one column to another and then place resulting data in a different column
- Print a whole page, selective columns or rows
- Save whole page to disk
  Retrieve whole pages, selective rows, or columns from disk
- Plus much, much more

One page of the pad, the worksheet, is in computer memory. The pad consists of worksheets stored on disk and linked by name. Pad size is only limited to diskette capacity.

If you find yourself spending hours (or days) copying, adding, subtracting, or whatever one column by another or just compiling data on a columnar pad, then COLUPAD™ can save you time, and in your business, time is money. COLUPAD™ requires a 32K PET, Disk, and printer. – $150.00

FOR MORE INFORMATION OR TO ORDER BY PHONE: (919) 362-4200 OR (919) 362-5671
INCLUDE $1.50 FOR SHIPPING AND HANDLING • VISA AND MASTERCHARGE ACCEPTED
DEALER INQUIRIES INVITED

P.O. BOX G, APEX, NORTH CAROLINA 27502 (919) 362-4200

ODDS & ENDS

.. re PET cassette tape

(ODDS & ENDS is a collection of quick notes on a variety of topics. If you have a quick hint on using any system, send it in to COMPUTE. It need not be new: anything that old timers might have forgotten or newcomers might not have learned is fair game for ODDS and ENDS).

If you're not sure if a cassette tape is blank, try this: press PLAY on the cassette unit, and hold down the < (less-than) key. Don't type LOAD. You'll see right away if there is something on the tape. This works on Original and Upgrade ROMs, but not on the new 4.0 systems.

Original ROM machines have two problems with cassette tape sequential files. First: there's barely enough space between data blocks, and there is a danger of missing a complete block during reading. A quick program can be written to stretch the inter-block space. Second: the end-of-file, as signalled in the ST status word, is non-standard .. it's not compatible with later ROM programs. Watch for it.

You can check with Commodore or with your dealer for fixes to these. If you're going to do much with cassette files, however, you'd be better off moving up to upgrade ROMs.

Original ROM machines perform a Basic SAVE by saving all data from 1024 up. Newer ROMs do the SAVE from 1025 up. No big deal, but the newer ROMs are better.

During cassette tape reading or writing, the keyboard is disabled. This is because the normal interrupt program has been disconnected. The clock is kept up to date, and the RUN/STOP key is still honoured .. special provision is made to do this in the tape routines.

Before writing on a new tape, rewind it completely. This will help unstretch the tape after a long period of storage.

Tape is more delicate and more prone to damage near the ends. It's a good idea to leave extra "leader" by running the tape ahead a short distance before recording.

If you have a speaker connected to CB2 of the Parallel User Port for sound effects, etc., you might find it useful to make an extra connection which allows you to listen to cassette tape. The tape signal is on either pin 6 or pin 8, depending on the PET unit and which cassette you are using. A resistor from the appropriate pin to the same output that CB2 feeds will do the trick. I use a 330K ohm value, which gives a quiet but audible signal; try your own values to fit your system. It's handy for spotting all sorts of tape problems: bad tapes, missed headers, and many others.

Programs are written as two tape blocks: a header of 192 bytes and the program itself which can be any length. Data files can be any number of blocks: again, there's a header of 192 bytes; then the data in 192-byte blocks. The first byte of each data block is reserved, so there are only 191 characters of actual data in each block.

There's a special block called an EOT (end-of-tape) block that can follow either programs or data files. When used, it stops the PET from searching further - the PET stops the tape and reports ?FILE NOT FOUND. This block can be created by giving the value of 2 as the third parameter of a SAVE or OPEN command: for example, SAVE "PROGRAM",1,2 will write an EOT block behind the program.

Two characters can never be written onto a cassette data file. The Linefeed (CHR$(10)) never reaches tape; it's blocked by the software. The Null (CHR$(0)) must not be used because it will cause a false end-of-file signal during file reading.

Everything that goes on tape - headers, programs, data, everything - is written twice; each block contains two closely-joined pieces. (If you listen to a tape, you'll hear a quick "blip" between the two halves of each block). During a read, the first block is the one that is actually read; the second block is used only to pick up errors spotted in reading the first block. PET can log up to 30 character errors for the second block to correct.

Don't depend on the PET message ?LOAD ERROR to tell you whether or not a program load was good; it won't always tell you about a bad load. Instead, type PRINT ST and look at the value that PET gives you. Zero means a good load; anything else spells trouble.

(Above items were submitted by Jim Butterfield) ©

THE STAR MODEM

From Livermore Data Systems

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

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

WE CARRY THE BLACK APPLE

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

MIPLOT Intelligent Plotter
by Watanabe Instruments (Digiplot)

SPECIAL $1095

- Incorporates all intelligent functions required for producing graphs and drawings including 8 vector and 4 character commands.
- Distance accuracy within 1%, repetition accuracy within 01 inches, programmable step size 004 inches, internal interpolation in 002 steps.
- Solid and broken line types can be specified.
- Character generator for letters, numerals and symbols. Characters can be enlarged and rotated in four orientations.
- Coordinate axes can be drawn by simply specifying the graduation interval and number of repetitions.
- Self test mode automatically draws complete test pattern.
- Printer mode outputs character data in 16 sizes with 4 orientations.
- Connection to any microcomputer using parallel 7-bit ASCII code.
- Can use simple cable to parallel port, or special interface to IEEE or other.
- Modular control circuit and mechanical construction.
- Uses any hard fiber-tip pen.
- Uses 11 x 17 paper.

Includes power supply, I/O connector, 2 pens, 50 sheets of paper, and complete manual.

EBS Business System for PET/CBM $795

Extremely comprehensive package for small business. Fully integrated inventory and accounts receivable system including invoices, packing slips, mail labels, statements, bank deposit slips, and 17 reports. Allows any of 10 standard letters to be merged with customer record info on either a selective or complete file basis. Demo disk and system description available for $3.00.

DZZ Data Base System for CBM 8032 $335

Flexible file handling and report writing package for such applications as: Inventory Control, Management Information, Mailing List; Scheduling; Medical Record Keeping, Accounting.

KMMM Pascal for PET $75

Subset of standard Pascal with true machine language translator for faster execution. 16K with tape or disk.

EARL for PET (disk file based) $65

Editor, Assembler, Relocater, Linker to generate relocatable object code.

fullFORTH+ for PET/CBM $65

A full-featured FORTH with extensions conforming to Forth Interest Group standards. Includes assembler, string processing capabilities, disk virtual memory multiple dimensioned arrays, floating point and integer processing.

6502	7.45	10/6.95	50/6.55	100/6.15
6502A	8.40	10/7.95	50/7.35	100/6.90
6520 PIA	5.15	10/4.90	50/4.45	100/4.15
6522 VIA	6.45	10/6.10	50/5.75	100/5.45
6532	7.90	10/7.40	50/7.00	100/6.60
2114-L450		3.95	20/3.75	100/3.45
2114-L300		4.45	20/4.25	100/3.95
,2716 EPROM		13.45	5/12.75	10/11.85
2732 EPROM (TMS2532)				29.50
4116-200 ns RAM (NEC)				**8 for 35.00**
S-100 Wire Wrap $2.65				Solder Tail $2.15

CASSETTES — AGFA PE-611 PREMIUM

High output, low noise, 5 screw housing, labels.

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

All other lengths available. Write for price list.

ATARI 800 $777

All Atari Modules 20% OFF

SPECIAL–purchase ATARI 800, receive extra 8K memory FREE.

EDUCATIONAL PLAN - buy 2 ATARI Computers, receive 1 ATARI 400 FREE!

DISKS

(write for quantity prices)

SCOTCH (3M) 8"	10/3.10	50/2.85	100/2.75
SCOTCH (3M) 5"	10/2.95	50/2.80	100/2.70
Maxell 5" Double Dens	10/4.25	50/4.10	100/3.95
Maxell 8" Double Dens	10/4.65	50/4.50	100/4.35
Verbatim 5"	10/2.40	50/2.35	100/2.30

(add .75 for plastic storage box for Verbatim 5")

Verbatim 8" Dbl. Dens.	10/3.35	50/3.25	100/3.15
BASF 5"	10/2.60	20/2.50	100/2.40
BASF 8"	10/2.40	20/2.35	100/2.30
Diskette Storage Pages			10 for 3.95
Disk Library Cases		8" - 2.85	5" - 2.15

commodore

CBM-PET SPECIALS

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

		FREE
8032 32K - 80 Column CRT	$1795	235
8016 16K - 80 Column CRT	$1495	205
8050 Dual Disk Drive-1,020,000 Bytes	$1795	235
CBM Modem - IEEE Interface	$395	50
CBM Voice Synthesizer	$395	50
8N Full size graphics keyboard	$795	100
16K Full Size Graphics or Business Keyboard	$995	150
32K Full Size Graphics or Business Keyboard	$1295	205
2040 Dual Disk Drive - 343,000 bytes	$1295	205
2022 Tractor Feed Printer	$795	100
C2N External Cassette Deck	$95	12
Used PETs (8, 16, and 32K)		CALL

WRITE FOR SYSTEM PRICES

****EDUCATIONAL DISCOUNTS****

Buy 2 PET Computers, get 1 FREE

CBM Full Size Graphics Keyboard	$ 74
WordPro III - 32K CBM, disk, printer	$170
WordPro IV - 8032, disk, printer	$255
VISICALC for PET, ATARI	$170
BPI General Ledger, A/P, A/R for PET	$270
Programmers Toolkit - PET ROM Utilities	$34.90
PET Spacemaker Switch	$24.90
Dust Cover for PET	$6.90
IEEE-Parallel Printer Interface for PET	$65.00
IEEE-RS232 Printer Interface for PET	$149.00

EPSON MX-80 Printer	$545
C. Itoh Daisy Wheel Printer	$1575
Centronics 737 Proportional Spacing Printer	$795
NEC Spinwriter - parallel	$2450
XYMEC HI-Q 1000 Intelligent Daisy Wheel	$1975
Leedex Video 100 12" Monitor	$129
ZENITH DATA SYSTEMS	
Zenith Z19 Terminal (factory asm.)	$735
Zenith Z89 with 48K	$2150
SYM-1	$209
SYM BAS-1 BASIC or RAE-1/2 Assembler	$ 85
KTM-2/80 Synertek Video Board	$349
KIM-1 (add $34 for power supply)	$159
Seawell Motherboard - 4K RAM	$195
Seawell 16K Static RAM - KIM, SYM, AIM	$320
KL-4M Four Voice Music Board and Visible Music Monitor (4 Voice) for PET	$59.90

FLEXFILE Database-Report Writer by Michael Riley
Flexible file handler for PET/CBM

MICRO-REVERSI for PET by Michael Riley $9.95
Machine language version—you can't win at Level 5.

PAPER-MATE 60 Command PET Word Processor $29.95
Full-featured version by Michael Riley

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

ALL BOOK and SOFTWARE PRICES DISCOUNTED

The 8086 Book (Osborne)	$12.75
Z8000 Assembly Language Programming	$10.60
PET Personal Computer Guide (Osborne)	$12.75
PET and the IEEE-488 Bus (Osborne)	$12.75
6502 Assembly Language (Osborne)	$ 9.90
Programming the 6502 (Zaks)	$10.45
6502 Applications Book (Zaks)	$10.45
6502 Software Cookbook (Scelbi)	$ 9.45
CP/M Handbook (w/ MP/M) Zaks	$11.85

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

WRITE FOR CATALOG.

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

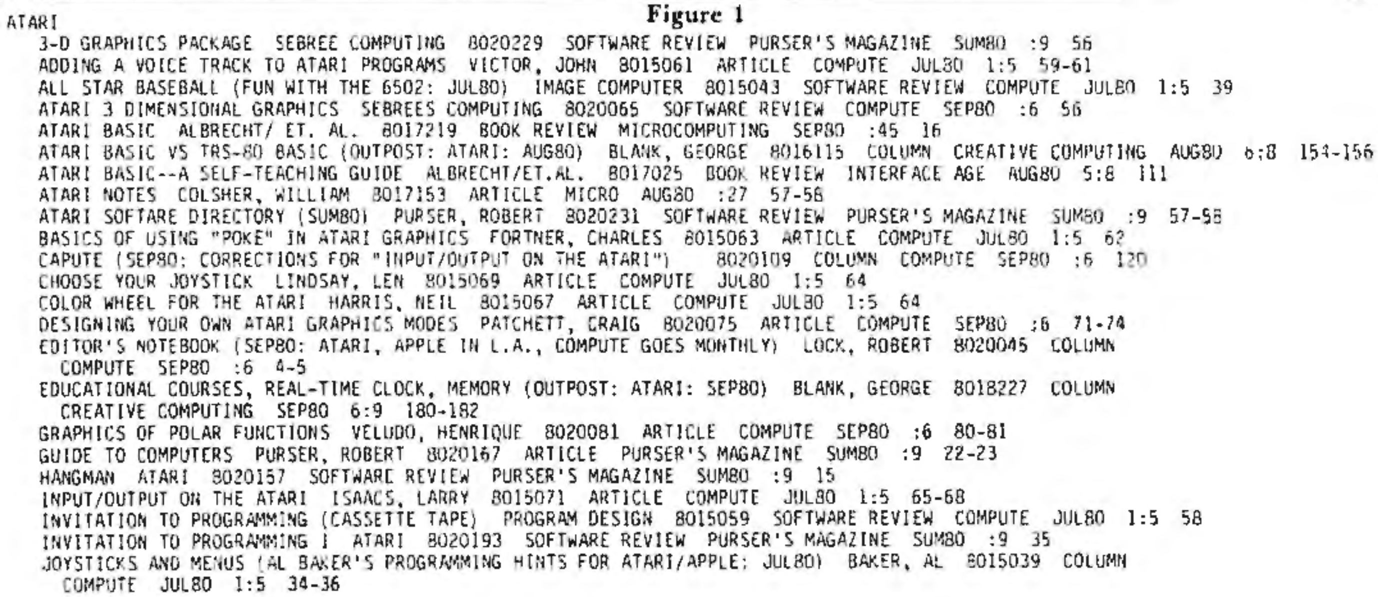
Review

Microcomputer Index

Robert Lock

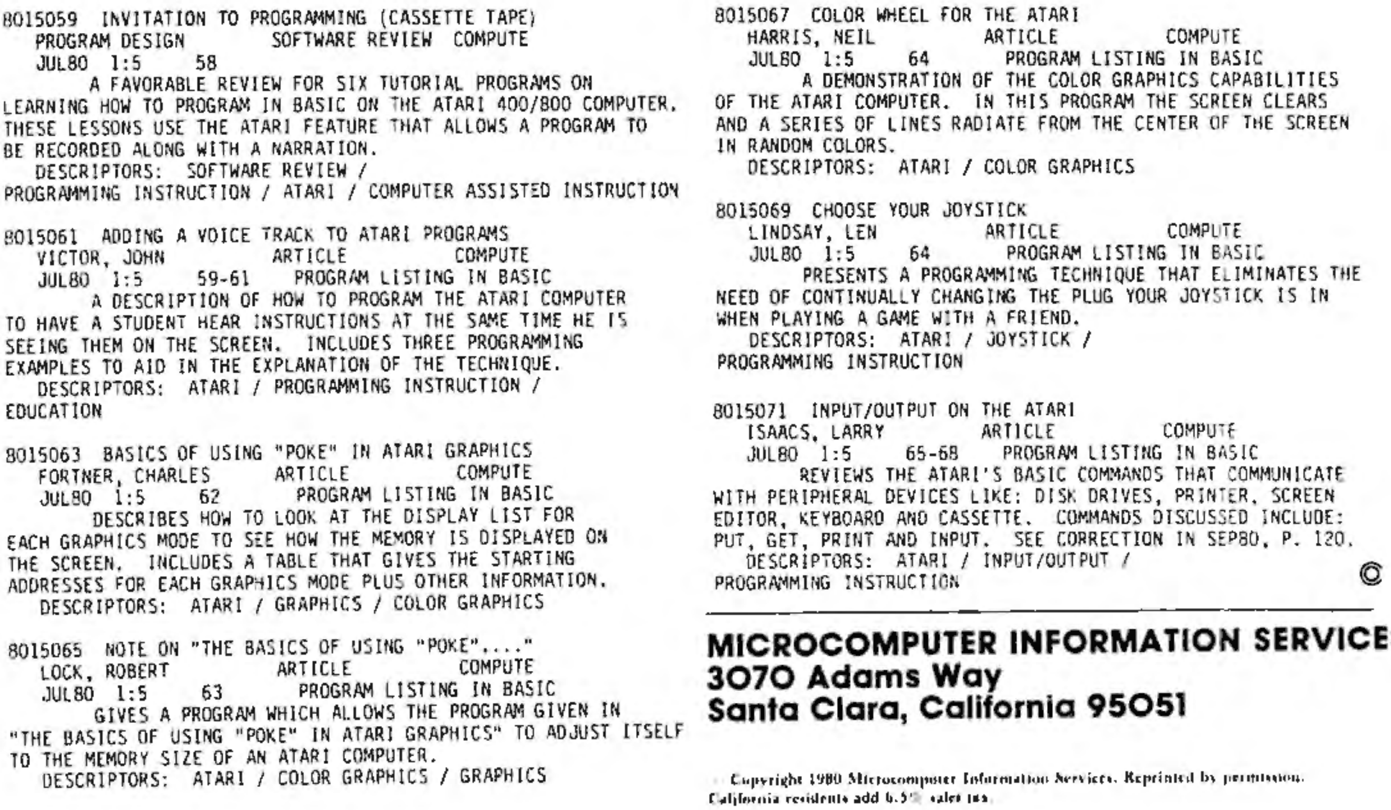
I was quite impressed with this new index when it arrived in early December. Joseph Ward, President of Microcomputer Information Services, is currently indexing 18 magazines in the microcomputer industry. Below are samples excerpted from the July-September, 1980 edition. Figure 1 shows a portion of the entries under "Atari". Figure 2 shows a portion of the entires for an issue of COMPUTE. The Index appears quarterly and is available through dealers or by direct mail. Cost is $5.95 per issue; $11.00 for July - December 80; $22.00 for an annual subscription. If you're desperate to remember where you saw that "Graphics" Article, this index might be quite useful. Most articles are indexed by up to four descriptors.

Figure 1



```
ATARI
   3-D GRAPHICS PACKAGE  SEBREE COMPUTING  8020229  SOFTWARE REVIEW  PURSER'S MAGAZINE  SUM80  :9  56
   ADDING A VOICE TRACK TO ATARI PROGRAMS  VICTOR, JOHN  8015061  ARTICLE  COMPUTE  JUL80  1:5  59-61
   ALL STAR BASEBALL (FUN WITH THE 6502: JUL80)  IMAGE COMPUTER  8015043  SOFTWARE REVIEW  COMPUTE  JUL80  1:5  39
   ATARI 3 DIMENSIONAL GRAPHICS  SEBREES COMPUTING  8020065  SOFTWARE REVIEW  COMPUTE  SEP80  :6  56
   ATARI BASIC  ALBRECHT/ ET. AL.  8017219  BOOK REVIEW  MICROCOMPUTING  SEP80  :45  16
   ATARI BASIC VS TRS-80 BASIC (OUTPOST: ATARI: AUG80)  BLANK, GEORGE  8016115  COLUMN  CREATIVE COMPUTING  AUG80  6:8  154-156
   ATARI BASIC--A SELF-TEACHING GUIDE  ALBRECHT/ET.AL.  8017025  BOOK REVIEW  INTERFACE AGE  AUG80  5:8  111
   ATARI NOTES  COLSHER, WILLIAM  8017153  ARTICLE  MICRO  AUG80  :27  57-58
   ATARI SOFTARE DIRECTORY (SUM80)  PURSER, ROBERT  8020231  SOFTWARE REVIEW  PURSER'S MAGAZINE  SUM80  :9  57-58
   BASICS OF USING "POKE" IN ATARI GRAPHICS  FORTNER, CHARLES  8015063  ARTICLE  COMPUTE  JUL80  1:5  62
   CAPUTE (SEP80: CORRECTIONS FOR "INPUT/OUTPUT ON THE ATARI")    8020109  COLUMN  COMPUTE  SEP80  :6  120
   CHOOSE YOUR JOYSTICK  LINDSAY, LEN  8015069  ARTICLE  COMPUTE  JUL80  1:5  64
   COLOR WHEEL FOR THE ATARI  HARRIS, NEIL  8015067  ARTICLE  COMPUTE  JUL80  1:5  64
   DESIGNING YOUR OWN ATARI GRAPHICS MODES  PATCHETT, CRAIG  8020075  ARTICLE  COMPUTE  SEP80  :6  71-74
   EDITOR'S NOTEBOOK (SEP80: ATARI, APPLE IN L.A., COMPUTE GOES MONTHLY)  LOCK, ROBERT  8020045  COLUMN
     COMPUTE  SEP80  :6  4-5
   EDUCATIONAL COURSES, REAL-TIME CLOCK, MEMORY (OUTPOST: ATARI: SEP80)  BLANK, GEORGE  8018227  COLUMN
     CREATIVE COMPUTING  SEP80  6:9  180-182
   GRAPHICS OF POLAR FUNCTIONS  VELUDO, HENRIQUE  8020081  ARTICLE  COMPUTE  SEP80  :6  80-81
   GUIDE TO COMPUTERS  PURSER, ROBERT  8020167  ARTICLE  PURSER'S MAGAZINE  SUM80  :9  22-23
   HANGMAN  ATARI  8020157  SOFTWARE REVIEW  PURSER'S MAGAZINE  SUM80  :9  15
   INPUT/OUTPUT ON THE ATARI  ISAACS, LARRY  8015071  ARTICLE  COMPUTE  JUL80  1:5  65-68
   INVITATION TO PROGRAMMING (CASSETTE TAPE)  PROGRAM DESIGN  8015059  SOFTWARE REVIEW  COMPUTE  JUL80  1:5  58
   INVITATION TO PROGRAMMING I  ATARI  8020193  SOFTWARE REVIEW  PURSER'S MAGAZINE  SUM80  :9  35
   JOYSTICKS AND MENUS (AL BAKER'S PROGRAMMING HINTS FOR ATARI/APPLE: JUL80)  BAKER, AL  8015039  COLUMN
     COMPUTE  JUL80  1:5  34-36
```

Figure 2



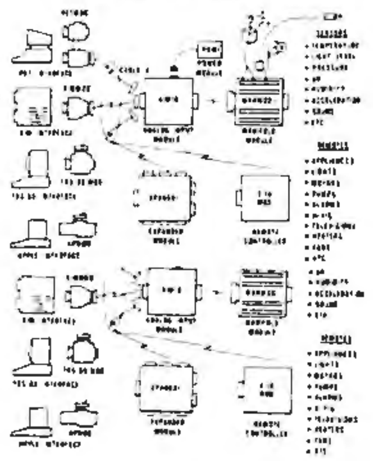
```
8015059  INVITATION TO PROGRAMMING (CASSETTE TAPE)
   PROGRAM DESIGN          SOFTWARE REVIEW  COMPUTE
   JUL80  1:5      58
        A FAVORABLE REVIEW FOR SIX TUTORIAL PROGRAMS ON
LEARNING HOW TO PROGRAM IN BASIC ON THE ATARI 400/800 COMPUTER.
THESE LESSONS USE THE ATARI FEATURE THAT ALLOWS A PROGRAM TO
BE RECORDED ALONG WITH A NARRATION.
   DESCRIPTORS:  SOFTWARE REVIEW /
PROGRAMMING INSTRUCTION / ATARI / COMPUTER ASSISTED INSTRUCTION

8015061  ADDING A VOICE TRACK TO ATARI PROGRAMS
   VICTOR, JOHN            ARTICLE          COMPUTE
   JUL80  1:5      59-61     PROGRAM LISTING IN BASIC
        A DESCRIPTION OF HOW TO PROGRAM THE ATARI COMPUTER
TO HAVE A STUDENT HEAR INSTRUCTIONS AT THE SAME TIME HE IS
SEEING THEM ON THE SCREEN.  INCLUDES THREE PROGRAMMING
EXAMPLES TO AID IN THE EXPLANATION OF THE TECHNIQUE.
   DESCRIPTORS:  ATARI / PROGRAMMING INSTRUCTION /
EDUCATION

8015063  BASICS OF USING "POKE" IN ATARI GRAPHICS
   FORTNER, CHARLES        ARTICLE          COMPUTE
   JUL80  1:5      62        PROGRAM LISTING IN BASIC
        DESCRIBES HOW TO LOOK AT THE DISPLAY LIST FOR
EACH GRAPHICS MODE TO SEE HOW THE MEMORY IS DISPLAYED ON
THE SCREEN.  INCLUDES A TABLE THAT GIVES THE STARTING
ADDRESSES FOR EACH GRAPHICS MODE PLUS OTHER INFORMATION.
   DESCRIPTORS:  ATARI / GRAPHICS / COLOR GRAPHICS

8015065  NOTE ON "THE BASICS OF USING "POKE"...."
   LOCK, ROBERT            ARTICLE          COMPUTE
   JUL80  1:5      63        PROGRAM LISTING IN BASIC
        GIVES A PROGRAM WHICH ALLOWS THE PROGRAM GIVEN IN
"THE BASICS OF USING "POKE" IN ATARI GRAPHICS" TO ADJUST ITSELF
TO THE MEMORY SIZE OF AN ATARI COMPUTER.
   DESCRIPTORS:  ATARI / COLOR GRAPHICS / GRAPHICS

8015067  COLOR WHEEL FOR THE ATARI
   HARRIS, NEIL            ARTICLE          COMPUTE
   JUL80  1:5      64        PROGRAM LISTING IN BASIC
        A DEMONSTRATION OF THE COLOR GRAPHICS CAPABILITIES
OF THE ATARI COMPUTER.  IN THIS PROGRAM THE SCREEN CLEARS
AND A SERIES OF LINES RADIATE FROM THE CENTER OF THE SCREEN
IN RANDOM COLORS.
   DESCRIPTORS:  ATARI / COLOR GRAPHICS

8015069  CHOOSE YOUR JOYSTICK
   LINDSAY, LEN            ARTICLE          COMPUTE
   JUL80  1:5      64        PROGRAM LISTING IN BASIC
        PRESENTS A PROGRAMMING TECHNIQUE THAT ELIMINATES THE
NEED OF CONTINUALLY CHANGING THE PLUG YOUR JOYSTICK IS IN
WHEN PLAYING A GAME WITH A FRIEND.
   DESCRIPTORS:  ATARI / JOYSTICK /
PROGRAMMING INSTRUCTION

8015071  INPUT/OUTPUT ON THE ATARI
   ISAACS, LARRY           ARTICLE          COMPUTE
   JUL80  1:5      65-68     PROGRAM LISTING IN BASIC
        REVIEWS THE ATARI'S BASIC COMMANDS THAT COMMUNICATE
WITH PERIPHERAL DEVICES LIKE: DISK DRIVES, PRINTER, SCREEN
EDITOR, KEYBOARD AND CASSETTE.  COMMANDS DISCUSSED INCLUDE:
PUT, GET, PRINT AND INPUT.  SEE CORRECTION IN SEP80, P. 120.
   DESCRIPTORS:  ATARI / INPUT/OUTPUT /
PROGRAMMING INSTRUCTION
```

©

MICROCOMPUTER INFORMATION SERVICES
3070 Adams Way
Santa Clara, California 95051

California residents add 6.5% sales tax.

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



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

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

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

Connectors

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

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

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

Analog Input Module

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

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

Power requirements are 12 volts DC at 60 ma.

POW1

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

TEMPSENS

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

Remote Controller-Clock and Calendar

AN INEXPENSIVE CONTROL SOLUTION FOR

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

SUPER X-10 MOD SPECS

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

XPANDRI

The XPANDRI1 allows up to eight Input/ Output modules to be connected to a computer at one time. The XPANDRI is connected to the computer in place of the AIM16 or X10 MOD. Up to eight AIM16s or seven Aim16s and one X10 MOD are then connected to each of the eight ports provided using a CABLE A24 for each module.

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

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

For your convenience the X10 MOD comes as part of a number of sets. The minimum configuration for a usable system is the X10 MOD Starter Set which includes one X10 MOD, one ICON and one OCON. This set requires that you have a hardware knowledge of your computer and computer interfacing.

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

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

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

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

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

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

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

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

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

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

PETSET1a (Commodore PET - 110 VAC)	295.00
PETSET1e (Commodore PET - 230 VAC)	305.00
KIMSET1a (KIM,SYM,AIM65 - 110 VAC)	285.00
KIMSET1e (KIM,SYM,AIM65 - 230 VAC)	295.00
APSET1a(APPLE II - 110 VAC)	295.00
APSET1e(APPLE II - 230 VAC)	305.00
TRS-80 SET1a (Radio Shack TRS-80 - 110 VAC)	295.00
TRS-80 SET1e(Radio Shack TRS-80 - 230 VAC)	305.00
AIM65 SET1a(AIM65-110 VAC)	285.00
AIM65 SET1e(AIM65-230 VAC)	295.00

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

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

CBM/PET Printer And Computer Interfaces

SADI For Serial Two-way Communication And Parallel Printers

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

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

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

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

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

ADA1600 ● For Parallel NEC and Centronics Standard Printers

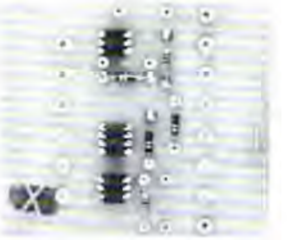
The ADA1600 is a low cost easy to use interface for the Commodore Computers. It allows the PET and CBM computers to use standard Centronics type printers (including the NEC 5530) for improved quality printing. The ADA1600 has a two foot cable which plugs into the PET IEEE port. Another IEEE card edge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

ADA1450 ● Serial Printer Adapters

The ADA1450 is a low cost, easy to use serial interface for the Commodore Computers. It allows the PET and CBM computers to use standard serial printers for improved quality printing. The ADA1450 has a two foot cable which plugs into the PET IEEE port. Another IEEE card edge connector is provided for connecting disks and other peripherals to the PET. The ADA1450 is addressable and does not tie up the bus. The address is switch selectable. A six foot RS-232 cable is provided with a DB25 connector. Pin 3 is data out. Pins 5,6 and 8 act as ready lines to the printer. Pins 4 and 20 act as ready lines from the printer. These lines can be switched for non-standard printers. Baud rate is selectable to 9600 baud. A switch selects upper/lower case, upper/lower case reversed (needed for some Commodore machines) and upper case only for clearer program listings. Works with WORDPRO, BASIC and other software. No special programming is required. The case measures 3 1/2 x 5 3/4 inches. Comes complete, assembled and tested, with case, cables, 120/230 VAC power supply and software on cassette, graphing functions, formatting data etc. The ADA1450 has a female DB25 connector at the end of the RS-232 cable for most standard printers. The ADA1450N has a male DB25 at the end of the RS-232 cable for the DIABLO serial printers. Retail price for the ADA1450 or 1450N is $139.

ADA730 Parallel ● For the Centronics 730 and 737 Printers

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



ADA400 RS-232 To Current Loop Adapter

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

ADA400S - Solder Pads 24.50
ADA400B - Barrier Strips with screw terminals 29.50

Word Processor Program ●

PET Word Processor. On tape - $39.50. On disk - 49.50
Compose and print letters, flyers, ads, manuscripts, etc. Uses disk or tape. 30 page manual included.

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

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

QUANTITY	DESCRIPTION	PRICE	TOTAL

SUBTOTAL
Handling and shipping — add per order $3.00
Foreign orders add 10% for AIR postage
Conn. residents add 7% sales tax
TOTAL ENCLOSED

NAME ______________________
COMPANY ______________________
ADDRESS ______________________
____
CITY ______________________
STATE ____________ ZIP __________
VISA ○ M/C ○ Expiration date ________
Card number ______________________

Mention this magazine and deduct 2% from the TOTAL

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

Load PET Program Tapes Into The APPLE II

Keith Falkner

Applesoft BASIC and COMMODORE BASIC are very similar languages. For this reason it is feasible to load PET programs into an APPLE II running Applesoft BASIC in either ROM or RAM. There are problems every step of the way, but this program solves most of them, with the result that the APPLE can actually run very many of the fine programs available for the PET. Of course there are some profound differences between the two computers, and programs which exploit features of the PET which are absent from the APPLE may be impossible to run.

I call this program "PET Loader", and it requires an APPLE II with Applesoft BASIC. There are two versions, one for the APPLE II PLUS as shown in the listing, and one for an APPLE II with Applesoft in RAM locations $0800-$3000. To load a program which just filled an 8K PET, an APPLE II PLUS uses 10K, and an APPLE II uses 20K. The PET program can then be saved and loaded exactly as an Applesoft program. PET Loader is not needed again for that program.

The operating instructions for PET Loader are in Table 1. If it detects an error, it prints a single digit to identify the error, followed by the message "ERR" and a beep, then it quits. Table 2 shows what has caused the error, and what to do then. Because the PET writes tape very dependably, and because there are two copies of the program on the tape, poor reads are rare, and the usual result is a cheerful 'OK', just after the traditional beep signalling 'stop the tape'. PET Loader then returns to Applesoft, and the "]" prompt appears.

Now the fun begins, or so the saying goes. There are many differences between the PET and the APPLE, which the program could not resolve. When real intelligence is needed, it all depends on you! A PET program loaded into an APPLE is still not an APPLE program. Here are the major differences which you will have to consider, when you try to produce a usable APPLE program:

1. No Equivalent Verb in the APPLE.
OPEN, CLOSE, VERIFY, and CMD are commands in COMMODORE BASIC. Each of these is translated "STOP", and you will need to decipher the programer's intent and program the equivalent for the APPLE. Refer to COMMODORE's excellent manuals for descriptions of these commands.

2. Specific Device Reference.
Programs containing OPEN and CLOSE will also contain either PRINT# or INPUT#, which are simply translated to PR# and IN# respectively, and will require substantial rework. The devices, by number, are conventionally these:

#0 the keyboard
#1 first cassette drive
#2 second cassette drive
#3 the screen
#4 the printer (or maybe not)
#8 the dual disk drive

3. Reference to Actual Memory Locations.
PEEK, POKE, CALL (SYS in the PET), WAIT, and USR refer to specific locations in memory, and you will need more help than I can offer here.

4. Keys to Move the Cursor.
The PET has ten keys to control the position or action of the cursor. Eight of these are missing from the APPLE. PET Loader translates these as follows:
Two functional equivalents-

CURSOR-LEFT becomes BACKSPACE, and appears as
"GR" in the program.

CURSOR-DOWN becomes LINE-FEED, and appears as
"PR#" in the program.

(Odd as they appear, these actually move the cursor exactly as stated.)

One destructive approximation:
CURSOR-RIGHT becomes SPACE, which obliterates what it should space past. This appears as "COLOR = " in the program.

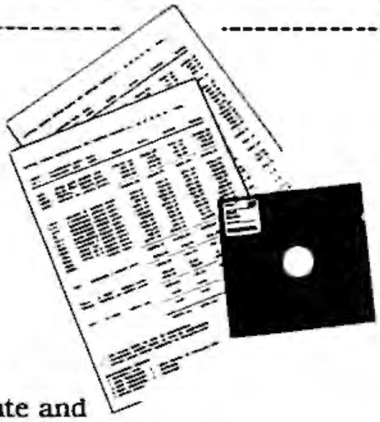
Seven non-functional comments:
When the program is listed, these are visible, looking like genuine verbs, and it looks as if the name of the key will be printed. For example,

```
100 PRINT "CLEAR"
110 INPUT "INVERSE INSTRUCTIONS NORMAL"
;Z$
```

FOR: **apple***

FROM: innerface BUSINESS SYSTEMS

Capital Asset Managment System™



SIMPLE,

CONCISE,

FUNCTIONAL,

ACCURATE,

EFFICIENT,

USABLE

SOFTWARE from I.B.S.

Having developed accurate and understandable software for both government and business, I.B.S. has at last Gone Public, with C.A.M.S. —

CAPITAL ASSETS MANAGEMENT SYSTEM, a simple, easy-to-use system designed for the APPLE Computer* and you.

- DEPRECIATES assets according to one of the eight approved schedules
- INVESTMENT CREDIT and RECAPTURE are computed automatically
- PROJECTIONS aid in the selection of method and term for new assets
- PERSONALIZED REPORTS in either 80 or 132 column formats
- AUTOMATIC SUBTOTALING selected by the user
- SAFTEY ROUTINES check input for most errors
- AUTOMATIC DISK FILING with user control

CAMS:

With just a few clear and simple keystrokes, you can view any of two hundred assets depreciated FROM any time TO any time, using any of the eight schedules.

By SEEING the results, you can make accurate decisions NOW, when it counts.

CAMS will automatically search your records for assets which qualify for either Investment Credit or Investment Credit Tax Recapture.

CAMS prompts you with clear and simple messages on the screen for all entries. Special safety routines check all practical inputs for errors. Because CAMS maintains all files automatically, it serves not only as a tool for projections, but as a RECORD FILING system as well.

And since no one should be limited by software, INNERFACE has made CAMS for user back-up copies as well. Because of this, you can record an UNLIMITED number of assets on individual disks for privacy and safe-keeping.

CAMS records 23 pieces of information on each asset: GL accounts; memos; dates; costs; method & life; full description; user ID/dept.; other deductions.
CAMS reports on depreciation: Straight-line; 125%, 150%, 200% w/wo automatic switch to Straight-line, Sum-of-year-digits AND Investment Tax Credit & Recapture amounts for each year.
CAMS requires a 48K Apple with disk and (at least) 80-column printer.
Cams is fully supported by professional documentation and by I.B.S.

* APPLE is a trademark of Apple Computer Inc.

innerface BUSINESS SYSTEMS

CAMS is 99.50 $2.00 SHIPPING
CA RES ADD 5.97 TAX

box 834, pacific grove CA 93950

In fact, line 100 will neither clear the screen nor print "CLEAR". It will merely print an equal-sign (=). Line 110 will print "INSTRUCTIONS", and no trace will be seen of the INVERSE and NORMAL commands shown in the listing. This behavior can be perplexing, because usually with Applesoft, what you see is what you get. The purpose of these translations is to disclose the intent of the program, so the cursor-keys of the PET are translated thus:

```
PET key: RUV OFF HOME CLR UP DEL INST
Lists as: INVERSE NORMAL HOME CLEAR VLIN
     DEL IN#
Result: nil nil nil " = " nil nil nil
```

... so when you list the program and discover:

```
300 INPUT "HOME PLAY INVERSE
AGAIN NORMAL" ; X$
```

substitute the equivalent APPLE code, which in this case is:

```
300 VTAB 1: PRINT "PLAY";: INVERSE:
PRINT "AGAIN";: NORMAL: INPUT "?";X$
```

For all those keys except INST and DEL, the equivalent in Applesoft is easy to devise, but logic to produce the function of PET's INSERT and DELETE keys is extremely difficult, and APPLE's convoluted screen addressing makes this task truly hairy. Fortunately very few PET programs print INSERT or DELETE characters.

5. Printing of numbers is slightly different.

```
290 X = 4 : Y = - 6
300 PRINT "X IS" ; X ; "Y IS" ; Y ; "."
```

The PET prints: X IS 4 Y IS-6.
THE APPLE PRINTS: X IS4Y IS-6.

The PET prints a blank before positive numbers, and a CURSOR-RIGHT after all numbers; the APPLE does neither. By the way, all four semicolons (;) in line 300 above are optional in both computers.

6. A side effect of TAB.

A PET can TAB over data already on the screen; the APPLE wipes it out, so use an HTAB verb in place of a TAB phrase if this difference mars the display.

```
PET: 40 PRINT TAB(11) "XYZ"
APPLE: 40 HTAB 12 : PRINT "XYZ"
```

7. Computations in Boolean Arithmetic.

In the following lines,

```
400 X = 11 : Y = 6 : Z = X > Y
410 PRINT Z : IF Z THEN 500
```

The PET will set Z to -1, and line 410 will print this result, then go to 500. The APPLE will set Z to +1, and print this different result, then go to 500. In the above example, the difference may not be crucial, but it often can be. Because the PET does bit-by-bit evaluation of the operators OR and AND, in

```
700 X = 11 : Y = 6 : Z = X AND Y
```

PET's result will be Z = 2, because the bit pattern of 11 is 0000 1011 and the bit pattern of 6 is 0000 0110, and these two patterns, ANDed, give 0000 0010, arithmetically 2. APPLE, on the other hand, merely sees that neither X nor Y is FALSE (zero), calls the result TRUE, and gets Z equal to 1. This can be a very subtle pitfall.

8. Random numbers.

RND (0) gives a genuine random number each time in the PET, but in the APPLE, it repeats the previous random number. Simply replace the 0 with a 1.

9. The GET command.

In the PET, GET does not wait for a key to be struck, so the sequence

```
333 GET P$ : IF P$ = "" THEN 333
```

is the customary way for a PET program to wait for a key. Oddly enough, this is completely appropriate in the APPLE, because if the key struck is CTRL-@, then P$ will be the null string. Ignorance of this is an obscure bug in some Applesoft programs. When the PET program is testing for a key but not waiting when no key has been struck, a different approach is needed. For example,

```
PET: 60 GET A$ : IF A$ = "" THEN 100
    APPLE: 60 ON PEEK (-16384) < 128 GOTO 100 :
    GET A$ : IF A$ = "" THEN 60
```

10. Graphics Characters and Lower Case.

The PET can produce lower-case letters and many graphic symbols which the APPLE cannot, and there are two display-modes in the PET. $C1 might mean "a" (old PET), "A" (new PET), or the symbol for the Spade suit (any PET). PET Loader looks for $CF, probably a lower-case "o", in the PET program. If $CF is found, all letters are translated to upper case; if not, graphic symbols are translated into a similar character the APPLE can produce.

11. Direct Screen Addressing.

In both computers, the video screen occupies a part of memory, and a POKE to a storage location in the screen memory will produce a character on the screen. The relationship between memory location and screen position in the PET is straight-forward, but it is quite complicated in the APPLE, and therefore not often used. Nevertheless, it is worth mastering, because there are hordes of PET programs which use this technique, and a lot of them are attractive games. The PET has 25 lines of 40 columns, and the memory location of each byte of the screen can be computed thus (the expression is not written in BASIC):

LOCATION = 32768 + 40 * LINE + COLUMN

where the upper left corner is LINE 0, COLUMN 0.

The APPLE has 24 lines of 40 columns, and the memory location of each byte can be calculated by:

LOCATION = XL% (LINE) + COLUMN

where the array XL% has been initialized thus:

```
1000 DIM XL% (23) : FOR I © 0 TO 7 :
        XL% (I) = 1024 + 128 * I :
        XL% (I + 8) = 1064 + 128 * I :
        XL% (I +16) = 1104 + 128 * I : NEXT
```

As before, the upper left corner is LINE 0, COL-

UMN 0. In applying this tactic take care not to let COLUMN exceed 39, or you will cause destruction of some important values in memory. For example, a POKE to valid LINE 23, and invalid COLUMN 49, will likely cause loss of your BASIC program.

12. Numeric Keypad.
Programs which use screen-POKEs to move pieces of a game around the screen use the keys 1-9 to indicate the direction of motion. This is satisfactory on the PET, because these keys form a square, with 7,8,9 above 4,5,6 above 1,2,3 and it is natural that, if the 5-key means 'stop', the 8-key means 'up', and the 3-key means 'down and right'. APPLE'S numeric keys do not form a square, so some substitute must be devised. None is immediately obvious, but perhaps the parallelogram formed by R,T,Y above F,G,H above V,B,N would serve, since the 'BELL' on the G-key can be easily remembered as being a 'home' position. Often the game can be improved by substituting use of the game paddles for these keys; why should you accept the limitations of the PET?

13. Sound.
Without an accessory, the PET is silent, but a simple way of making noises has become very popular. Discovery of a POKE to 59466 is strong evidence that this technique is in use. Here is how to deduce the programmer's intent:

POKE 59467,16	startmaking noises (musical tones)
POKE 59467,0	stop making noise altogether
POKE 59466,15 (or 51 or 85)	typical values to define an octave
POKE 59464,X	high X is a low note, and vice versa.

It is important to remember that once the PET issues three POKEs, the sound is continuous, whereas the APPLE must continually address the speaker to keep making noise. This part of the conversion can be left for last, because the POKEs address APPLE's Read-Only Memory, which is unaffected by POKE.

14. Real-time clock.
The PET has a genuine timer, which programs can address in two ways. Variable TI increases by 60 every second, and can be read but not written; string TI$ is six numeric characters, in the format HHMMSS, and can be read and written.

```
400 PRINT "THE TIME IS" ; TI$
```

TI$ is computed from the instantaneous value of TI, and formatted as up to six digits. If you try running a PET program in your APPLE, and it just stalls, doing nothing at all, press CTRL-C to stop it, and you may find lines like

```
700 X = TI + 60
710 IF TI < X THEN 710
```

Line 710 is merely waiting for a second to elapse, and in the timeless APPLE, it never will. Substitute a FOR-NEXT loop of the appropriate duration.

15. PI.
A single key on the PET provides the number PI, 3.14159265. No such facility exists in the APPLE, so PI is translated into a character which prints as "UNDEF'D FUNCTION". This causes "?SYNTAX ERROR", and is simple to correct.

16. The INPUT statement has subtle differences.
The PET provides a question mark after the prompt for INPUT, and for neatness, you should supply one. If the PET issues an INPUT statement and the user merely presses RETURN, execution of the program ceases at once. This is such a nuisance that programs with any elegance guard against it in a variety of ways, for example:

```
50 INPUT "WHAT NOW > > + < < < " ; X$
```

where '>' and '<' stand for keys to move the cursor right and left. You can tidy this up as you supply the question mark for the prompt.

As you ponder all the differences and incompatibilities listed, and the other differences which you will find, you may wonder if the resulting program will be worth the work. In fact, PET Loader is a potent and reliable tool, and you will appreciate it. The tape-reading process has proven particularly sound, and has error-recovery procedures even better than those in the PET.

I received a lot of help in this project, all of it from Gordon Campbell of Toronto, who provided manuals, clippings, advice, and many PET pro-

grams. He received some Applesoft programs in exchange, and he loaded them into his PET! But that is his story, and you can read it in the Sep/Oct 1980 issue of COMPUTE.

To get the source and object of both versions of PET Loader, as well as two accessory programs I have developed, send me ten dollars, and I will mail them to you on a cassette, together with printed copies of Table 1 and Table 2.

TABLE 1.
OPERATING INSTRUCTIONS

1. Before loading PET Loader, ensure the BASIC pointers in page zero are normal, by either issuing the DOS command "FP", or powering off and on.
2. Load PET Loader from tape or disk and run it. It will ask you to "PLAY" a PET program tape and press a key. Do so (or press CTRL-C to quit).
3. Any clicks you hear indicate unreadable bytes on the tape. Before or between programs on the tape, these are perfectly normal events.
4. After 10-40 seconds, the name of a program will appear near the top of the screen. If you wish to bypass this program, press CTRL-X promptly, and the name will disappear, then you can press a key to try the next program.
5. Many error conditions are tested at this point. Refer to Table 2 if any of them occurs. The signal is a digit, the letters "ERR", and a "BEEP".
6. The image of the program will be read into memory, and simultaneously be displayed in a 32-character window. Some of it will appear to make sense.
7. Relax. PET programs load slowly, for example 20K in nine minutes. If the speaker clicks during the load, the wait will be doubled, because PET Loader will have to read the second copy of the program as well.
8. When the whole program is loaded, a "BEEP" is issued. Stop the tape.
9. There is a very brief pause while tokens and data are translated, then the message "OK" is issued, and the Applesoft prompt "]" reappears.
10. You now have a PET program in your APPLE. Refer to the article, to turn it into an APPLE program (there's always a catch, isn't there?).
11. To rerun PET Loader, just type "&". You don't need to reload the program.

TABLE 2.
ERRORS AND RECOVERY

If PET Loader detects an error it cannot handle, it issues an error message and quits, returning control to Applesoft. The error message is a digit and "ERR".

MESSAGE	MEANING AND RECOVERY
1 ERR	A long search for a Header Label has not found one. If this really is a PET tape, perhaps the volume is improperly set. PET tapes are louder than APPLE tapes, so adjust the volume by ear and try again.
2 ERR	Neither copy of the label could be read correctly. This might also be improper volume, but is probably a bedraggled tape. Either try again, or acquire another copy of the program.
3 ERR	The header label was read correctly, and a code in it says that the contents of this file is not a BASIC program, but data. PET Loader reads programs only. Try another file or another tape.
4 ERR	This program is too big to fit in memory. Perhaps the pointer at location 115-116 ($73-74) was altered by HIMEN:, so issue the DOS command "FP", or HIMEN:16384 (or what have you), and try again.
5 ERR	After the header label was read correctly, the program should follow. It was not found, perhaps because the PET user aborted the "SAVE". If repeated tries cause the same message, that is most likely.
6 ERR	This program loads at an unusual address in the PET. Possibilities are: proprietary program, machine-language or partly so, or some attempt to make the program difficult to copy (or use?). Forget it!

To try again, just type "&" and press RETURN. With RAM Applesoft "CALL 1013". Use this tactic even if you have typed "NEW" or "FP", or have pressed RESET!

MAKING IT WORK

The best way to enter PET Loader into your APPLE is to have an Assembler. I used Programma's mighty ASM/65, whose only idiosyncracy is the use of "<" and ">" to denote the low and high halves of a two-byte address. (The ".FILE" lines serve to connect pieces of a program which will not all fit in memory.) Next best is the Mini-Assembler which is part of Integer BASIC. Worst is byte-by-byte keying through the Machine-Language Monitor. If you are capable of doing any of these, you already know how (this is not the program to learn the technique on!).

To package PET Loader as an Applesoft program, execute these instructions after loading it into locations $800-$BF6, using the Machine-Language Monitor:

```
*67:01 08 00 0C 00 0C 00 0C
*AF:00 0C N E003G
(which gets you back to Applesoft)
```

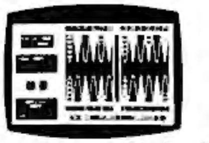
then LIST the program; you should get only one line: 65535 CALL 2064 : END PET Loader can now be saved, loaded, and run exactly as an Applesoft program, except that PET Loader cannot be saved after it has been run. The program occupies locations 2048-3071 ($800-$BFF), and makes them unavailable to Applesoft. This costs 1024 bytes of memory, but saves having to reload PET Loader to convert the next program. To restore the pointers to normal, issue the DOS command "FP", or type "CALL -151" then enter Applesoft "cold" with CTRL-B.

The version of PET Loader which operates with RAM Applesoft occupies memory locations

Software for the Apple II and Apple II Plus*

ASTEROIDS IN SPACE™
By Bruce Wallace
An exciting space action game! Your space ship is traveling in the middle of a shower of asteroids. Blast the asteroids with lasers, but beware — big asteroids fragment into small asteroids! The Apple game paddles allow you to rotate your space ship, fire its laser gun, and give it thrust to propel it through endless space. From time to time you will encounter an alien space ship whose mission is to destroy you, so you'd better destroy it first! High resolution graphics and sound effects add to the arcade like excitement that this program generates. Runs on any Apple II with at least 32K of RAM and one disk drive.

On Diskette - $19.95



FASTGAMMON™ By Bob Christiansen.
Sound, hi res, color, and musical cartoons have helped make this the most popular backgammon playing game for the Apple II. But don't let these entertaining features fool you — FASTGAMMON plays serious backgammon. Runs on any Apple II with at least 24K of RAM.

Cassette - $19.95 Diskette - $24.95

ASTROAPPLE™ by Bob Male.
Your Apple computer becomes your astrologer, generating horoscopes and forecasts based on the computed positions of the heavenly bodies. This program offers a delightful and stimulating way to entertain friends. ASTROAPPLE produces natal horoscopes (birth charts) for each person based on his or her birth data. Any two people may be compared for physical, emotional, and intellectual compatibility. The program is written in Applesoft BASIC with machine language subroutines. It requires either RAM or ROM Applesoft and at least 32K of memory.

Cassette - $14.95 Diskette - $19.95

FRACAS™ by Stuart Smith
A fantastic adventure game like no other! Up to eight players can participate in FRACAS at the same time. Journey in the land of FAROPH, searching for hidden treasure while warding off all sorts of unfriendly and dangerous creatures. You and your friends can compete with each other or you can join forces and gang up on the monsters. Your location is presented graphically and sound effects enliven the battles. Save your adventure on diskette or cassette and continue it at some other time. Both integer BASIC and Applesoft versions included. Requires at least 32K of RAM.

Cassette - $19.95 Diskette - $24.95

BENEATH APPLE MANOR™ by Don Worth.
Descend beneath Apple Manor into an underground maze of corridors, rooms, and secret passages in quest of rich and powerful treasures. The dungeon complex consists of many floors, each lower level more dangerous than the last. If you can reach the lowest level you may even find the ultimate object of your quest, the fabled golden apple of Apple Manor. Strategy is extremely important as you deal with a variety of monsters, each with its own characteristics. Written in integer BASIC with machine language subroutines. Requires integer BASIC and at least 16K of RAM on cassette or 32K of RAM on diskette.

Cassette - $14.95 Diskette - $19.95

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

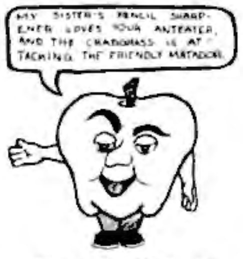
Now exclusive distributors for products from The Software Factory, Newhall, California
*Apple II and Apple II Plus are trademarks of Apple Computer, Inc.

BATTLESHIP COMMANDER™ by Erik Kirk and Matthew Jew

A game of strategy. You and the computer each start out by positioning five ships of different sizes on a ten by ten grid. Then the shooting starts. Place your volleys skillfully — a combination of logic and luck are required to beat the computer. Cartoons show the ships sinking and announce the winner. Sound effects and flashing lights also add to the enjoyment of the game. Both Applesoft and integer BASIC versions are included. Requires at least 32K of RAM.

Cassette - $14.95 Diskette - $19.95

BABBLE™ by Don Worth.
Have fun with this unique software. You write a story, entering it as a BABBLE program. As you write the story you specify certain words to be selected by the computer or entered from the keyboard at execution time. Run the program and watch BABBLE convert your story into an often hilarious collection of incongruities. The ways in which BABBLE can entertain you are limited only to your imagination. You can compose an impressive political speech or write poetry. You can plan a dinner menu. You can even form images on the screen or compose musical tunes with the help of BABBLE. The cassette version requires at least 16K of RAM and the diskette version requires at least 32K of RAM. BABBLE is written in machine language and runs on any Apple II computer.


MY SISTER'S PENCIL SHARPENER LOVES YOUR ANTEATER, AND THE CRABGRASS IS ATTACKING THE FRIENDLY MATADOR.

BABBLE

Cassette - $19.95 Diskette - $24.95

L
I
N
K
E
R

LINKER by Don Worth
Turn your Apple II or Apple II Plus into a powerful and productive software development machine with this superb linking loader/editor package. LINKER does the following and much more:

- Dynamically loads and relocates suitably prepared machine language programs anywhere in RAM.
- Combines a main program with subroutines. You can assemble a subroutine once and then use it with as many main programs as you wish.
- Produces a map of all loaded routines, giving their location and the total length of the resulting module.
- Contains a library of subroutines including binary multiplication and division, print text strings, delay, tone generator, and random number generator.

Linker works with virtually any assembler for the Apple II. Requires 32K of RAM and one disk drive.

Diskette - $49.95
Manual Only - $19.95

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

12288-13311 ($3000-$33FF). The bogus Applesoft program within it says "CALL 12304 : END", and references to locations within the program are $2800 bytes higher than shown in the listing of the version for ROM Applesoft.

```
LINE# LOC CODE   LINE

0000             ; R.O.M. APPLESOFT REQUIRED FOR THIS.
0000             ;
0000                     *=$800
0800             BEGIN**                    ;LOADER STARTS
0800             ;
0800             ;-----> BOGUS APPLESOFT PROGRAM:
0800             ;
0800             ; 65535 CALL 2064:END
0800             ;
0800 00                  .BYT 0          ;TRADITION
0801 0D08                .WOR $080D      ;ADDR CHAIN
0803 FFFF                .WOR $FFFF      ;LINE # 65535
0805 8C                  .BYT $8C        ;TOKEN FOR 'CALL'
0806 3230                .BYT '2064:'    ;PLACE CALLED
0808 36343A
080B 80                  .BYT $80        ;TOKEN FOR 'END'
080C 00                  .BYT 0,0,0,0    ;END OF PGM
080D 00
080E 00
080F 00
0810             ;

0810              .FILE 'PETLOAD1'
0810             ;
0810             ; I ENABLE AN APPLE II COMPUTER
0810             ; TO LOAD PET PROGRAM CASSETTES.
0810             ;
0810             ; COPYRIGHT (C) 1980
0810             ; BY: KEITH FALKNER    (416) 466-3734
0810             ;      76 ROOSEVELT ROAD
0810             ;      TORONTO CANADA  M4J 4T7
0810             ;
0810             ;-----> PAGE-ZERO WORKAREAS
0810             ;
0810             ERRORS=$A5                 ;# BAD BYTES & PARITY
0810             QUOTE=$A6                  ;AM I INSIDE QUOTES?
0810             LINSIZ=$A7                 ;BYTES IN BASIC LINE
0810             GRAPHO=$A8                 ;TRANSLATE GRAPHICS?
0810             MEMLO=$06                  ;WHERE I AM
0810             MEMHI=$07                  ; LOADING
0810             MAXLO=$08                  ;END OF
0810             MAXHI=$09                  ; PGM.
0810             ;
0810             BYTE=$18                   ;INPUT BYTE
0810             ECHO=$19                   ;ITS INVERSE
0810             PARITY=$1A                 ;PET PARITY IS ODD
0810             MAX=$1B                    ;TIME LONG BITS
0810             MIN=$1C                    ;AND SHORT BITS
0810             RETRY=$1D                  ;$80 IF RETRYING
0810             SYNCSX=$1E                 ;LOCATES A BLOCK
0810             STACK=$1F                  ;FOR FINAL RETURN ONLY
0810             ;
0810             ;-----  LABEL INPUT AREA
0810             ;
0810             LABEL=$230                 ;THIS SHOWS DATA TYPE:
0810             ; 1=LABEL OF DATA FILE
0810             ; 2=BLOCK OF DATA FILE
0810             ; 4=LABEL OF BASIC PGM
0810             ;
0810             LMEMLO=LABEL+1             ;START OF
0810             LMEMHI=LABEL+2             ; PROGRAM
0810             LMAXLO=LABEL+3             ;END OF
0810             LMAXHI=LABEL+4             ; PROGRAM
0810             LNAME=LABEL+5              ;NAME OF PGM
0810             ;
0810             ;-----  INITIALIZE
0810             ;
0810             BASIC=BEGIN+$400            ;PET PGM STARTS
0810             START=*
0810 BA                  TSX                ;FOR FINAL RETURN
0811 861F                STX STACK
0813 202FFB              JSR $FB2F          ;INIT TEXT
0816 2058FC              JSR $FC58          ;CLEAR SCREEN
0819 A911                LDA #17            ;HAND MADE
081B 8525                STA $25            ; VTAB 17
081D A98D                LDA #$8D           ;(CR) MOVES
081F 20EDFD              JSR $FDED          ;CURSOR THERE
0822             ;
0822 A94C                LDA #$4C           ;JMP OPCODE
0824 8DF503              STA $3F5           ;RESTART &-SPOT
0827 A910                LDA #<START
0829 8DF603              STA $3F6
082C A908                LDA #>START
082E 8DF703              STA $3F7
0831             ;
0831             ; CRUCIAL TIMING FIGURES!
0831             ;
0831 A925                LDA #$25        ;  THIS MEANS 'LONG'
0833 851B                STA MAX
0835 A919                LDA #$19        ;  THIS MEANS 'SHORT'
0837 851C                STA MIN
0839             ;
0839             ;-----  POKE GREETING & RULES TO CRT
0839             ;
0839 A000                LDY #0
083B B9C00A HELLO        LDA GREET,Y
083E 0980                ORA #$80
0840 990005              STA $500,Y
0843 B9040B              LDA RULES,Y
0846 0980                ORA #$80
0848 990006              STA $600,Y
084B B92C0B              LDA AGAIN,Y
084E 0980                ORA #$80
0850 992806              STA $628,Y
0853 C8                  INY
0854 C028                CPY #40            ;LINE FULL
0856 D0E3                BNE HELLO
0858             ;
0858             ;-----  WAIT FOR A KEYSTROKE
0858             ;
0858 2C10C0              BIT $C010          ;NO KEY YET!
085B A960                LDA #$60           ;FLASHER
085D 8D0007              STA $700
0860 AD00C0 WAIT         LDA $C000          ;ANY KEY HIT?
0863 10FB                BPL WAIT           ;NO=WAIT
0865 48                  PHA
0866 A9A0                LDA #$A0           ;BLANK
0868 8D0007              STA $700
086B A000                LDY #0
086D 990006 WCLR         STA $600,Y         ;CLEAR RULES
0870 C8                  INY
0871 C028                CPY #40
0873 90F8                BCC WCLR
0875 2C10C0              BIT $C010          ;RESET STROBE
0878 68                  PLA
0879 C983                CMP #$83           ;CONTROL-C?
087B D001                BNE OPEN1          ;NO=GO AHEAD
087D 60                  RTS                ;ABORTED!
087E             ;
087E             ;-----. READ LABEL OF PET TAPE
087E             ;
087E A900   OPEN1        LDA #0             ;1ST TRY
0880 851D   OPEN         STA RETRY
0882 20520A              JSR BLOCK          ;FRONT OF IT
0885 A000                LDY #0             ;RESET INDEX
0887 20800A OPBYTE       JSR RDBYTE         ;GET A BYTE
088A 993002              STA LABEL,Y        ;STASH BYTE
088D C005                CPY #5             ;BEFORE NAME?
088F 9005                BCC OPSTEP         ;YES=GET MORE
0891 0980                ORA #$80           ;SIGN BIT ON
0893 99FB06              STA $6FB,Y         ;PRINT NAME
0896 C8     OPSTEP       INY                ;NEXT BYTE
0897 C016                CPY #22            ;21 ALREADY?
0899 90EC                BCC OPBYTE         ;N=GET MORE
089B             ;
089B             ; LABEL IS 192 BYTES. I NEED 21.
089B             ;
089B A5A5                LDA ERRORS         ;HOW DID I DO?
089D F00D                BEQ OPTYPE         ;0=PERFECT!
089F A51D                LDA RETRY          ;1ST TRY?
08A1 3004                BMI OPFAIL         ;NO
08A3 A980                LDA #$80           ;MEANS 2ND TRY
08A5 30D9                BMI OPEN
08A7 A902   OPFAIL       LDA #2             ;ERROR-TYPE 2
08A9 4CCC0A OPEXIT       JMP CRASH          ;ABORT!
08AC 851D   OPTYPE       STA RETRY          ;0=1ST TRY
08AE AD3002              LDA LABEL          ;TYPE OF FILE
08B1 C901                CMP #1             ;BASIC PGM?
08B3 F004                BEQ SIZTST         ;YES=CONTINUE
08B5 A903                LDA #3             ;NOT BASIC
08B7 D0F0                BNE OPEXIT         ;ABORT
08B9             ;
08B9             ;-----  WILL PROGRAM FIT IN SPACE
08B9             ;
08B9 AD3202 SIZTST       LDA LMEMHI         ;PGM START
08BC C904                CMP #4             ;AT $04??
08BE D007                BNE SIZODD         ;NO
08C0 AD3102              LDA LMEMLO         ;DOES IT
08C3 C902                CMP #2             ;00 OR 01?
08C5 9004                BCC SIZMAX
08C7 A906   SIZODD       LDA #6             ;FUNNY ADDR
08C9 D0DE                BNE OPEXIT         ;ABORT!
08CB AD3302 SIZMAX       LDA LMAXLO         ;END OF PGM
08CE 8508                STA MAXLO          ;SAVE IN PG 0
08D0 AD3402              LDA LMAXHI
08D3 18                  CLC
08D4 6908                ADC #>BEGIN
08D6 8509                STA MAXHI
08D8 C974                CMP #$74           ;FP HIMEM HI
08DA 900C                BCC LOAD           ;YES IT FITS
08DC D006                BNE SIZERR         ;NO IT DOESN'T
08DE A508                LDA MAXLO          ;IT MIGHT FIT
08E0 C973                CMP #$73           ;FP HIMEM LO
08E2 9004                BCC LOAD           ;IT JUST FITS
08E4 A904   SIZERR       LDA #4             ;ERROR CODE 4
08E6 D0C1   SIZEXT       BNE OPEXIT         ;ABORT!
08E8             ;
08E8             ;-----  LOAD THE PROGRAM IMAGE
08E8             ;
08E8 A90C   LOAD         LDA #>BASIC        ;INIT ADDR TO
08EA 8507                STA MEMHI          ;LOAD PGM INTO
08EC A900                LDA #$00           ;BASIC AREA
08EE 8506                STA MEMLO
08F0 A000                LDY #0
08F2 84A8                STY GRAPHO         ;SET TO TRANSLATE
08F4 9106                STA (MEMLO),Y      ;0 NEEDED THERE
08F6 85A7                STA LINSIZ
08F8             ;
08F8             ;-----  BEGIN IMAGE BLOCK
08F8             ;
08F8 20520A LDBLOK       JSR BLOCK          ;BEGIN BLOCK
08FB AD00C0              LDA $C000          ;HEY, ANY KEYS
08FE C983                CMP #$83           ;CTRL-C TO ESCAPE
0900 D003                BNE LDGO           ;NO, LOAD IT.
```

```
0202 0902 4C1008        JMP START        ;NEXT PGM
0203 0905 20800A LDGO   JSR RDBYTE        ;READ THE 1ST BYTE
0204 0908 C901          CMP #1           ;2ND LABEL?
0205 090A 900E          BCC LDNOTZ       ;BYPASS POSSIBLE ZERO
0206 090C D008          BNE LD1ST        ;N: BASIC IMAGE
0207 090E 241D          BIT RETRY        ;LOST?
0208 0910 10E6          BPL LDBLOK       ;N: OK
0209 0912 A905          LDA #5           ;ERROR-CODE 5
0210 0914 D0D0          BNE SIZEXT       ;ABORT!
0211 0916 E606   LD1ST  INC MEMLO         ;BASIC+1
0212 0918 D00D          BNE LDMOVE       ;(ALWAYS)
0213 091A        ;
0214 091A        ;-----> LLONG LLOADING LLOOP!
0215 091A        ;
0216 091A E606   LDNOTZ INC MEMLO         ;STEP TO $0C01-$3401
0217 091C 20800A LDBYTE JSR RDBYTE
0218 091F 241D          BIT RETRY        ;2ND TRY?
0219 0921 1004          BPL LDMOVE       ;NO
0220 0923 C9FC          CMP #$FC         ;BAD NOW?
0221 0925 F004          BEQ LDSHOW       ;YES
0222 0927 A000   LDMOVE LDY #0            ;NO INDEXING
0223 0929 9106          STA (MEMLO),Y    ;STASH BYTE
0224 092B 48     LDSHOW PHA
0225 092C A506          LDA MEMLO        ;WHERE IT WENT
0226 092E 291F          AND #$1F         ;LIMIT SPREAD
0227 0930 AA            TAX              ;PREP INDEX
0228 0931 68            PLA
0229 0932 48            PHA
0230 0933 0980          ORA #$80
0231 0935 9D2C04        STA $42C,X       ;SHOW THE BYTE
0232 0938 E6A7          INC LINSIZ
0233 093A A905          LDA #5           ;IN 1ST 4 BYTES
0234 093C C5A7          CMP LINSIZ       ;IGNORE 00 & CF.
0235 093E 900D          BCC LDPULL
0236 0940 68            PLA
0237 0941 AA            TAX              ;END-OF-LINE?
0238 0942 D002          BNE LDLOWO       ;NO
0239 0944 85A7          STA LINSIZ       ;RESET
0240 0946 C9CF   LDLOWO CMP #$CF          ;LOWER CASE O?
0241 0948 D004          BNE LDDONE
0242 094A 85A8          STA GRAPHO       ;NO TRANSLATION
0243 094C 48            PHA
0244 094D 68     LDPULL PLA
0245 094E        ;
0246 094E        ;-----> HAVE I DONE IT ALL?
0247 094E        ;
0248 094E A507   LDDONE LDA MEMHI         ;WHERE IT IS
0249 0950 C509          CMP MAXHI        ;END OF PGM
0250 0952 D006          BNE LDSTEP       ;MORE TO DO
0251 0954 A506          LDA MEMLO
0252 0956 C508          CMP MAXLO
0253 0958 F008          BEQ LDTEST
0254 095A E606   LDSTEP INC MEMLO         ;STEP 1 BYTE
0255 095C D0BE          BNE LDBYTE       ;SAME PAGE
0256 095E E607          INC MEMHI        ;NEW PAGE
0257 0960 D0BA          BNE LDBYTE       ;(ALWAYS)
0258 0962        ;
0259 0962        ;-----> HOW WELL DID IT LOAD?
0260 0962        ;
0261 0962 A5A5   LDTEST LDA ERRORS        ;# BAD BYTES
0262 0964 F00B          BEQ LDING        ;NONE!!!!!!!
0263 0966 A51D          LDA RETRY        ;2ND PASS?
0264 0968 3007          BMI LDING        ;YES
0265 096A A980          LDA #$80         ;NOW IT IS!
0266 096C 851D          STA RETRY
0267 096E 4CE808        JMP LOAD
0268 0971 20E2FB LDING  JSR $FBE2         ;RING BELL
0269 0974        ; BELL MEANS  DONE WITH TAPE .
0270 0974        ;
0271 0974        ;-----> PREPARE PAGE ZERO POINTERS
0272 0974        ;       FOR APPLESOFT INTERPRETER.
0273 0974        ;
0274 0974 A901          LDA #$01
0275 0976 8567          STA $67          ;START OF BASIC
0276 0978 A90C          LDA # BASIC
0277 097A 8568          STA $68
0278 097C A608          LDX MAXLO
0279 097E 86AF          STX $AF          ;END OF
0280 0980 A409          LDY MAXHI        ;PROGRAM.
0281 0982 84B0          STY $B0
0282 0984 E8            INX
0283 0985 D001          BNE POINT1
0284 0987 C8            INY
0285 0988 E8     POINT1 INX               ;2 ABOVE THAT
0286 0989 D001          BNE POINT2       ;IS START OF
0287 098B C8            INY              ;BOTH ...
0288 098C 8669   POINT2 STX $69
0289 098E 846A          STY $6A          ;VARIABLES
0290 0990 866B          STX $6B
0291 0992 846C          STY $6C          ;& ARRAYS.
0292 0994 866D          STX $6D
0293 0996 846E          STY $6E          ;END OF VARS.
0294 0998        ;
0295 0998        ;-----  ADJUST THE ADDRESS-CHAIN.
0296 0998        ;       BECAUSE THIS PROGRAM LIVED
0297 0998        ;       AT $0401 IN THE PET, AND
0298 0998        ;       WAS LOADED JUST ABOVE ME.
0299 0998        ;
0300 0998 A567          LDA $67
0301 099A 8506          STA MEMLO        ;START OF
0302 099C A568          LDA $68          ;ADDR-CHAIN
0303 099E 8507          STA MEMHI
0304 09A0 A000   ADJUST LDY #0            ;NO INDEXING
0305 09A2 B106          LDA (MEMLO),Y    ;LO ADDR LINK
0306 09A4 AA            TAX              ;HOLD IT!
0307 09A5 C8            INY              ;INDEX BY 1
0308 09A6 B106          LDA (MEMLO),Y    ;HI ADDR LINK
0309 09A8 F00B          BEQ CONVRT       ;0=ALL DONE
0310 09AA 18            CLC
```

```
0311 09AB 6908          ADC #>BEGIN
0312 09AD 9106          STA (MEMLO),Y    ;PUT IT BACK.
0313 09AF 8606          STX MEMLO        ;UPDATE BOTH
0314 09B1 8507          STA MEMHI        ;LINK BYTES.
0315 09B3 D0EB          BNE ADJUST       ;(ALWAYS)
0316 09B5        ;
0317 09B5        ;-----> NOW TO CONVERT TOKENS
0318 09B5        ;
0319 09B5 A567   CONVRT LDA $67
0320 09B7 8506          STA MEMLO        ;START OF
0321 09B9 A568          LDA $68          ;PROGRAM.
0322 09BB 8507          STA MEMHI
0323 09BD        ;
0324 09BD A000   COSCAN LDY #0            ;NO INDEXING!
0325 09BF B106          LDA (MEMLO),Y
0326 09C1 AA            TAX
0327 09C2 C8            INY
0328 09C3 B106          LDA (MEMLO),Y
0329 09C5 F078          BEQ APPLE        ;IF DONE CONV.
0330 09C7 88            DEY              ;Y=0 FROM HERE DOWN!
0331 09C8        ;
0332 09C8 84A6          STY QUOTE        ;NOT IN QUOTES
0333 09CA 8608          STX MAXLO        ;-> FINISH OF
0334 09CC 8509          STA MAXHI        ;CURRENT LINE.
0335 09CE        ;
0336 09CE 18            CLC
0337 09CF A506          LDA MEMLO        ;STEP PAST
0338 09D1 6904          ADC #4           ;ADDR-CHAIN
0339 09D3 8506          STA MEMLO        ;& LINE NO.
0340 09D5 A507          LDA MEMHI
0341 09D7 6900          ADC #0
0342 09D9 8507          STA MEMHI
0343 09DB        ;
0344 09DB B106   COBYTE LDA (MEMLO),Y     ;PICK A BYTE
0345 09DD C922          CMP #$22         ;IS IT QUOTE?
0346 09DF F043          BEQ COQUOT       ;YUP.
0347 09E1 24A6          BIT QUOTE        ;IN QUOTES?
0348 09E3 3016          BMI CODATA       ;YUP
0349 09E5        ;
0350 09E5        ; IS THIS BYTE A PET TOKEN?
0351 09E5        ;
0352 09E5 C9FC          CMP #$FC         ;BADLY READ?
0353 09E7 F041          BEQ COSTEP       ;YES. KEEP IT.
0354 09E9 C9CB          CMP #$CB         ;>LARGEST PET TOKEN?
0355 09EB B008          BCS COUNDF       ;YES: SUBSTITUTE.
0356 09ED AA            TAX              ;PREP INDEX
0357 09EE 103A          BPL COSTEP       ;IF NOT TOKEN
0358 09F0 BD2C0B        LDA TOKENS-$80,X ;APPLE TOKEN
0359 09F3 D002          BNE COTOKE       ;ALWAYS
0360 09F5 A9FB   COUNDF LDA #$FB          ;UNDEF'D FUNCTION
0361 09F7 9106   COTOKE STA (MEMLO),Y     ;FIX PET TOKEN
0362 09F9 D02F          BNE COSTEP       ;ALWAYS
0363 09FB        ;
0364 09FB        ; DEAL WITH DATA INSIDE QUOTES:
0365 09FB        ; SOME ARE TRANSLATED. THE REST
0366 09FB        ; HAVE THEIR SIGN-BITS REMOVED.
0367 09FB        ;
0368 09FB A218   CODATA LDX #GRAFIC-DATRAN
0369 09FD CA     CODTRY DEX
0370 09FE DD530B        CMP DATRAN-1,X ;TEST BYTE
0371 0A01 F019          BEQ CODSUB       ;YES: TRANSLATE IT.
0372 0A03 CA            DEX              ;STEP BACK
0373 0A04 D0F7          BNE CODTRY       ;IF MORE TO TEST
0374 0A06        ;  THE BYTE WAS NOT IN 'DATRAN'.
0375 0A06        ;  SEE IF I TRANSLATE GRAPHICS.
0376 0A06 A6A8          LDX GRAPHO       ;ANY LOWER-CASE 'O'?
0377 0A08 D00E          BNE COPLUS       ;YES. NO TRANSLATE.
0378 0A0A C9E0          CMP #$E0         ;GRAPHIC KEY?
0379 0A0C B00A          BCS COPLUS       ;NO
0380 0A0E C9A0          CMP #$A0         ;FOR SURE?
0381 0A10 9006          BCC COPLUS       ;NO
0382 0A12 E9A0          SBC #$A0         ;CALC INDEX
0383 0A14 AA            TAX
0384 0A15 BD6C0B        LDA GRAFIC,X
0385 0A18 297F   COPLUS AND #$7F          ;KILL SIGN
0386 0A1A 1003          BPL CODSTA       ;ALWAYS
0387 0A1C BD540B CODSUB LDA DATRAN,X      ;GET SUBSTITUTE
0388 0A1F 9106   CODSTA STA (MEMLO),Y     ;REPLACE DATA.
0389 0A21 4C2A0A CONEXT JMP COSTEP
0390 0A24        ;
0391 0A24        ;-----> FOUND QUOTE, SO FLIP SWITCH.
0392 0A24        ;
0393 0A24 A5A6   COQUOT LDA QUOTE
0394 0A26 4980          EOR #$80         ;INVERT SIGN
0395 0A28 85A6          STA QUOTE
0396 0A2A        ;
0397 0A2A        ; STEP TO NEXT BYTE IN LINE.
0398 0A2A        ;
0399 0A2A E606   COSTEP INC MEMLO
0400 0A2C D002          BNE COTEST
0401 0A2E E607          INC MEMHI
0402 0A30 A506   COTEST LDA MEMLO
0403 0A32 C508          CMP MAXLO        ;DONE A LINE?
0404 0A34 D0A5          BNE COBYTE       ;N: NEXT BYTE
0405 0A36 A507          LDA MEMHI
0406 0A38 C509          CMP MAXHI
0407 0A3A D09F          BNE COBYTE
0408 0A3C 4CBD09        JMP COSCAN       ;Y: NEXT LINE
0409 0A3F        ;
0410 0A3F        ;-----  FINISHED! PRINT  OK  (CR)
0411 0A3F        ;
0412 0A3F A9CF   APPLE  LDA #$CF          ; O
0413 0A41 20F6FD        JSR $FDF6
0414 0A44 A9CB          LDA #$CB         ; K
0415 0A46 20F6FD        JSR $FDF6
0416 0A49 A98D          LDA #$8D         ;(CR)
0417 0A4B 20F6FD        JSR $FDF6
0418 0A4E        ;
0419 0A4E        ;-----  THE ONLY WAY OUT
```

```
0420 0A4E            ;
0421 0A4E A61F   EXIT   LDX STACK
0422 0A50 9A            TXS
0423 0A51 60            RTS
0424 0A52        ;

0425 0A52         .FILE 'PETLOAD2'
0426 0A52        ;
0427 0A52        ;-----> POSITION TO START OF BLOCK
0428 0A52        ;
0429 0A52 A900   BLOCK  LDA #0          ;ZERO COUNT
0430 0A54 85A5          STA ERRORS     ;OF ERRORS
0431 0A56 20800A BLPASS JSR RDBYTE      ;GET A BYTE
0432 0A59 451D          EOR RETRY      ;(OR 04)
0433 0A5B C984          CMP #$84       ;HEX 84?
0434 0A5D F008          BEQ BLSYNC     ;FOUND IT!
0435 0A5F        ;
0436 0A5F        ; EACH BLOCK BEGINS THUS:
0437 0A5F        ; 89,88,87,86,85,84,83,82,81, DATA
0438 0A5F        ; (DUPLICATES ARE 09,08,...)
0439 0A5F        ;
0440 0A5F        ; I LOOK FOR $84, THEN THE REST.
0441 0A5F        ;
0442 0A5F 24A5          BIT ERRORS     ;MANY ERRORS?
0443 0A61 10F3          BPL BLPASS     ;<128 = CONTINUE
0444 0A63 A901   BLFAIL LDA #1          ;ERROR CODE 1
0445 0A65 D065          BNE CRASH      ;ABORT'
0446 0A67        ;
0447 0A67        ; INSIST ON 83, 82, 81:
0448 0A67        ;
0449 0A67 851E   BLSYNC STA SYNC8X      ;SAVE THE 84
0450 0A69 A900          LDA #0         ;FORGIVE ANY
0451 0A6B 85A5          STA ERRORS     ;ERRORS SO FAR
0452 0A6D        ;
0453 0A6D C61E   BLWALK DEC SYNC8X      ;COUNTDOWN
0454 0A6F A980          LDA #$80
0455 0A71 C51E          CMP SYNC8X     ;PAST THE $81
0456 0A73 F046          BEQ RBEXIT     ;YES; EXIT.
0457 0A75 20800A        JSR RDBYTE     ;GET NEXT
0458 0A78 451D          EOR RETRY      ;ALLOW FOR RETRY
0459 0A7A C51E          CMP SYNC8X     ;RIGHT BYTE?
0460 0A7C F0EF          BEQ BLWALK     ;Y; CONTINUE
0461 0A7E D0D6          BNE BLPASS     ;N; STROLL ON
0462 0A80        ;
0463 0A80        ;-----> READ A BYTE FROM TAPE
0464 0A80        ;
0465 0A80 20BC0A RDBYTE JSR RDBIT       ;GET A BIT
0466 0A83 E41B          CPX MAX        ;BIG FAT ONE?
0467 0A85 90F9          BCC RDBYTE     ;WAIT FOR IT
0468 0A87 20BC0A        JSR RDBIT      ;IGNORE NEXT
0469 0A8A A980          LDA #$80       ;SET TO GET 8
0470 0A8C 8519          STA ECHO
0471 0A8E        ;
0472 0A8E        ; WHEN THAT $80 FALLS OUT THE
0473 0A8E        ; RIGHT SIDE OF  ECHO , EIGHT
0474 0A8E        ; BITS HAVE FLOWED INTO BYTE.
0475 0A8E        ;
0476 0A8E 851A          STA PARITY     ;START EVEN
0477 0A90 20BC0A RD8BIT JSR RDBIT       ;SET OR CLR CARRY
0478 0A93 9002          BCC RDROR      ;IS IT A ZERO-BIT?
0479 0A95 E61A          INC PARITY     ;COUNT ONE-BITS.
0480 0A97 6618   RDROR  ROR BYTE        ;ASSEMBLE DATA
0481 0A99 20BC0A        JSR RDBIT      ;GET PARTNER
0482 0A9C 6619          ROR ECHO       ;ABSORB A BIT
0483 0A9E 90F0          BCC RD8BIT     ;GO FOR 8 BITS
0484 0AA0 20BC0A        JSR RDBIT      ;READ PARITY
0485 0AA3 9002          BCC RDTEST     ;IF IT'S ZERO
0486 0AA5 E61A          INC PARITY     ;ELSE COUNT
0487 0AA7 A519   RDTEST LDA ECHO
0488 0AA9 49FF          EOR #$FF       ;INVERT ECHO
0489 0AAB C518          CMP BYTE       ;MUST = BYTE
0490 0AAD D004          BNE RDBAD      ;NOPE.
0491 0AAF 661A          ROR PARITY     ;ODD PARITY?
0492 0AB1 B007          BCS RDEXIT     ;YUP.
0493 0AB3 E6A5   PDBAD  INC ERRORS
0494 0AB5 A9FC          LDA #$FC       ;TOKEN FOR ERROR
0495 0AB7 8D30C0        STA $C030      ;TOGGLE SPEAKER
0496 0ABA AA     RDEXIT TAX             ;SET FLAGS Z,N
0497 0ABB 60     PBEXIT PTS
0498 0ABC        ;
0499 0ABC        ;-----  READ A BIT FROM TAPE.
0500 0ABC        ;
0501 0ABC        ; THERE ARE  BELIEVE IT OR NOT,
0502 0ABC        ; 4 TRANSITIONS IN A BIT. THIS
0503 0ABC        ; CODE IGNORES POSITIVE ONES,
0504 0ABC        ; BECAUSE THE NEGATIVE ONES ARE
0505 0ABC        ; MORE DRAMATIC. NOW, A LONG AND
0506 0ABC        ; A MEDIUM STARTS A BYTE, AFTER
0507 0ABC        ; WHICH A SHORT AND A MEDIUM IS
0508 0ABC        ; A ZERO-BIT, WHILE A MEDIUM AND
0509 0ABC        ; A SHORT IS A ONE-BIT. ALL THE
0510 0ABC        ; NEGATIVE TRANSITIONS ARE READ
0511 0ABC        ; AND TESTED, AND ONLY VALID BIT
0512 0ABC        ; COMBINATIONS ARE PERMITTED.
0513 0ABC        ;
0514 0ABC A200   PDBIT  LDX #0          ;START TIMER
0515 0ABE 2C60C0 RDBITW BIT $C060       ;LOOK @ INPUT
0516 0AC1 10FB          BPL RDBITW     ;IGNORE +VE.
0517 0AC3 E8     RDBITC INX             ;TIME -VE WAVE
0518 0AC4 2C60C0        BIT $C060      ;LOOK @ INPUT
0519 0AC7 30FA          BMI RDBITC     ;WAIT FOR +VE
0520 0AC9 E41C          CPX MIN        ;SET CARRY BIT
0521 0ACB 60            RTS
0522 0ACC        ;
0523 0ACC        ;-----  DISASTER' ACCUM CONTAINS:
0524 0ACC        ;
0525 0ACC        ;  1 -- CAN T FIND START OF LABEL
0526 0ACC        ;  2 -- BAD BITS/PARITY IN LABEL
0527 0ACC        ;  3 -- THIS ISN T A BASIC PGM
0528 0ACC        ;  4 — PGM IS TOO BIG FOR APPLE
0529 0ACC        ;  5 — CAN'T FIND START OF PGM
0530 0ACC        ;  6 -- IT LOADS @ A FUNNY ADDR
0531 0ACC        ;
0532 0ACC 09B0   CRASH  ORA #$B0        ;NOW ASCII
0533 0ACE 20F6FD        JSR $FDF6      ;PRINT IT.
0534 0AD1 A9A0          LDA #$A0       ;MAKE A SPACE
0535 0AD3 20F6FD        JSR $FDF6      ;PRINT IT TOO
0536 0AD6 202DFF        JSR $FF2D      ;'ERR  & BELL
0537 0AD9 4C4E0A        JMP EXIT       ;EXIT
0538 0ADC        ;
0539 0ADC 5020   GREET  .BYT  P E T   L O A D E P
0539 0ADE 4520
0539 0AE0 5420
0539 0AE2 2020
0539 0AE4 4C20
0539 0AE6 4F20
0539 0AE8 4120
0539 0AEA 4420
0539 0AEC 4520
0539 0AEE 5220
0540 0AF0 2020          .BYT      BY FALKNER  V2.0
0540 0AF2 2020
0540 0AF4 4259
0540 0AF6 2046
0540 0AF8 414C
0540 0AFA 4B4E
0540 0AFC 4552
0540 0AFE 2020
0540 0B00 5632
0540 0B02 2E30
0541 0B04 504C   RULES  .BYT  PLAY PET PROGRAM TAP
0541 0B06 4159
0541 0B08 2050
0541 0B0A 4554
0541 0B0C 2050
0541 0B0E 524F
0541 0B10 4752
0541 0B12 414D
0541 0B14 2054
0541 0B16 4150
0542 0B18 4520          .BYT 'E AND PRESS ANY KEY.
0542 0B1A 414E
0542 0B1C 4420
0542 0B1E 5052
0542 0B20 4553
0542 0B22 5320
0542 0B24 414E
0542 0B26 5920
0542 0B28 4B45
0542 0B2A 592E
0543 0B2C 2852   AGAIN  .BYT '(RESTART FROM APPLES
0543 0B2E 4553
0543 0B30 5441
0543 0B32 5254
0543 0B34 2046
0543 0B36 524F
0543 0B38 4D20
0543 0B3A 4150
0543 0B3C 504C
0543 0B3E 4553
0544 0B40 4F46          .BYT 'OFT BY TYPING  &  .)
0544 0B42 5420
0544 0B44 4259
0544 0B46 2054
0544 0B48 5950
0544 0B4A 494E
0544 0B4C 4720
0544 0B4E 2726
0544 0B50 272E
0544 0B52 2920
0545 0B54        ;
0546 0B54        DATRAN=*                ;DATA TO TRANSLATE:
0547 0B54 118A          .DBY $118A     ;DOWN         - LINEFEED
0548 0B56 129E          .DBY $129E     ;RVS          - INVERSE
0549 0B58 1397          .DBY $1397     ;HOME         - HOME
0550 0B5A 1485          .DBY $1485     ;DEL          - DEL
0551 0B5C 1DA0          .DBY $1DA0     ;RIGHT        - SPACE
0552 0B5E 918F          .DBY $918F     ;UP           - VLIN
0553 0B60 929D          .DBY $929D     ;OFF          - NORMAL
0554 0B62 93BD          .DBY $93BD     ;CLR          - CLEAP
0555 0B64 948B          .DBY $948B     ;INST         - IN#
0556 0B66 9D88          .DBY $9D88     ;LEFT         - BACKSPACE
0557 0B68 FCFC          .DBY $FCFC     ;ERROR BEEP - ERROR BEEP
0558 0B6A A227          .DBY $A227     ;SHIFTQUOTE - APOSTROPHE
0559 0B6C        ;
0560 0B6C        GRAFIC=*                ;APPROX GRAPHIC KEYS
0561 0B6C 205B          .BYT   [       ;$A0-A1
0562 0B6E 3D2D          .BYT  '=-      ;$A2-A3  I HAVE TO MAKE
0563 0B70 5F28          .BYT  '_('     ;$A4-A5  A GUESS:
0564 0B72 2A29          .BYT   *)      ;$A6-A7
0565 0B74 3D2F          .BYT   =       ;$A8-A9  DOES THE PROGPAM
0566 0B76 5D21          .BYT   ]'      ;$AA-AB  DISPLAY LOWER-
0567 0B78 2E2E          .BYT   ..      ;$AC-AD  CASE LETTERS, OR
0568 0B7A 2E3D          .BYT   .=      ;$AE-AF  DOES IT USE THE
0569 0B7C 2E2D          .BYT   .-      ;$B0-B1  PET GRAPHICS
0570 0B7E 2D21          .BYT   -'      ;$B2-B3  CHARACTERS?
0571 0B80 5B5B          .BYT   [[      ;$B4-B5
0572 0B82 5D3D          .BYT   ]=      ;$B6-B7  I LOOK FOR A
0573 0B84 3D3D          .BYT   ==      ;$B8-B9  LOWER-CASE  O
0574 0B86 5D2E          .BYT   ].      ;$BA-BB  WHILE LOADING.
0575 0B88 2E2E          .BYT   ..      ;$BC-BD
0576 0B8A 2E3F          .BYT   .?'     ;$BE-BF  IF I FIND ONE,
0577 0B8C 2D53          .BYT   -S      ;$C0-C1  I PRESUME THAT
0578 0B8E 212D          .BYT   !-      ;$C2-C3  THE PROGRAM IS
0579 0B90 2D2D          .BYT   --      ;$C4-C5  PRINTING LOWER
0580 0B92 2D28          .BYT  '-(      ;$C6-C7  CASE LETTERS.
0581 0B94 292E          .BYT   ).      ;$C8-C9  AND I DO NOT
0582 0B96 2E2E          .BYT   ..      ;$CA-CB  USE THIS TABLE.
```

```
0583 0B98 5B5C          .BYT  [\        ;$CC-CD
0584 0B9A 2F5B          .BYT  /[        ;$CE-CF   OF COURSE, I
0585 0B9C 5D2A          .BYT  ]*        ;$D0-D1   MIGHT BE WRONG
0586 0B9E 5F43          .BYT  _C        ;$D2-D3   EITHER WAY.
0587 0BA0 282E          .BYT  (.        ;$D4-D5   AND INDEED SOME
0588 0BA2 582B          .BYT  X+        ;$D6-D7   PROGRAMS USE
0589 0BA4 4329          .BYT  C)        ;$D8-D9   THE KEYS BOTH
0590 0BA6 442B          .BYT  D+        ;$DA-DB   WAYS.
0591 0BA8 5B21          .BYT  [!        ;$DC-DD
0592 0BAA 235C          .BYT  #\        ;$DE-DF
0593 0BAC         ;
0594 0BAC         ;------ THE MIGHTY TOKEN TABLE!
0595 0BAC         ;       AT (TOKENS - $80 + PET TOKEN)
0596 0BAC         ;       IS THE SAME APPLE TOKEN.
0597 0BAC         TOKENS=*
0598 0BAC 80            .BYT $80        ;END
0599 0BAD 81            .BYT $81        ;FOR
0600 0BAE 82            .BYT $82        ;NEXT
0601 0BAF 83            .BYT $83        ;DATA
0602 0BB0 8B            .BYT $8B        ;INPUT#-IN#    <-HELP!
0603 0BB1 84            .BYT $84        ;INPUT
0604 0BB2 86            .BYT $86        ;DIM
0605 0BB3 87            .BYT $87        ;READ
0606 0BB4 AA            .BYT $AA        ;LET
0607 0BB5 AB            .BYT $AB        ;GOTO
0608 0BB6 AC            .BYT $AC        ;RUN
0609 0BB7 AD            .BYT $AD        ;IF
0610 0BB8 AE            .BYT $AE        ;RESTORE
0611 0BB9 B0            .BYT $B0        ;GOSUB
0612 0BBA B1            .BYT $B1        ;RETURN
0613 0BBB B2            .BYT $B2        ;REM
0614 0BBC B3            .BYT $B3        ;STOP
0615 0BBD B4            .BYT $B4        ;ON
0616 0BBE B5            .BYT $B5        ;WAIT
0617 0BBF B6            .BYT $B6        ;LOAD
0618 0BC0 B7            .BYT $B7        ;SAVE
0619 0BC1 B3            .BYT $B3        ;VERIFY-STOP
0620 0BC2 B8            .BYT $B8        ;DEF
0621 0BC3 B9            .BYT $B9        ;POKE
0622 0BC4 8A            .BYT $8A        ;PRINT#-PR#    <-HELP!
0623 0BC5 BA            .BYT $BA        ;PRINT
0624 0BC6 BB            .BYT $BB        ;CONT
0625 0BC7 BC            .BYT $BC        ;LIST
0626 0BC8 BD            .BYT $BD        ;CLEAR
0627 0BC9 B3            .BYT $B3        ;CMD-STOP
0628 0BCA 8C            .BYT $8C        ;SYS-CALL      <-HELP!
0629 0BCB B3            .BYT $B3        ;OPEN-STOP
0630 0BCC B3            .BYT $B3        ;CLOSE-STOP
0631 0BCD BE            .BYT $BE        ;GET
0632 0BCE BF            .BYT $BF        ;NEW
0633 0BCF C0            .BYT $C0        ;TAB(
0634 0BD0 C1            .BYT $C1        ;TO
0635 0BD1 C2            .BYT $C2        ;FN
0636 0BD2 C3            .BYT $C3        ;SPC(
0637 0BD3 C4            .BYT $C4        ;THEN
0638 0BD4 C6            .BYT $C6        ;NOT
0639 0BD5 C7            .BYT $C7        ;STEP
0640 0BD6 C8            .BYT $C8        ;+
0641 0BD7 C9            .BYT $C9        ;-
0642 0BD8 CA            .BYT $CA        ;*
0643 0BD9 CB            .BYT $CB        ;/
0644 0BDA CC            .BYT $CC        ;↑
0645 0BDB CD            .BYT $CD        ;AND
0646 0BDC CE            .BYT $CE        ;OR
0647 0BDD CF            .BYT $CF        ;>
0648 0BDE D0            .BYT $D0        ;=
0649 0BDF D1            .BYT $D1        ;<
0650 0BE0 D2            .BYT $D2        ;SGN
0651 0BE1 D3            .BYT $D3        ;INT
0652 0BE2 D4            .BYT $D4        ;ABS
0653 0BE3 D5            .BYT $D5        ;USR
0654 0BE4 D6            .BYT $D6        ;FRE
0655 0BE5 D9            .BYT $D9        ;POS
0656 0BE6 DA            .BYT $DA        ;SQR
0657 0BE7 DB            .BYT $DB        ;RND
0658 0BE8 DC            .BYT $DC        ;LOG
0659 0BE9 DD            .BYT $DD        ;EXP
0660 0BEA DE            .BYT $DE        ;COS
0661 0BEB DF            .BYT $DF        ;SIN
0662 0BEC E0            .BYT $E0        ;TAN
0663 0BED E1            .BYT $E1        ;ATN
0664 0BEE E2            .BYT $E2        ;PEEK
0665 0BEF E3            .BYT $E3        ;LEN
0666 0BF0 E4            .BYT $E4        ;STR$
0667 0BF1 E5            .BYT $E5        ;VAL
0668 0BF2 E6            .BYT $E6        ;ASC
0669 0BF3 E7            .BYT $E7        ;CHR$
0670 0BF4 E8            .BYT $E8        ;LEFT$
0671 0BF5 E9            .BYT $E9        ;RIGHT$
0672 0BF6 EA            .BYT $EA        ;MID$
0673 0BF7               .END

ERRORS = 0000  0000
```

```
SYMBOL  TABLE

SYMBOL    VALUE

ADJUST  09A0   AGAIN   0B2C   APPLE   0A3F
BASIC   0C00   BEGIN   0800   BLFAIL  0A63
BLOCK   0A52   BLPASS  0A56   BLSYNC  0A67
BLWALK  0A6D   BYTE    0018   COBYTE  09DB
CODATA  09FB   CODSTA  0A1F   CODSUB  0A1C
CODTRY  09FD   CONEXT  0A21   CONVRT  0965
COPLUS  0A18   COQUOT  0A24   COSCAN  098D
COSTEP  0A2A   COTEST  0A30   COTOKE  09F7
COUNOF  09F5   CRASH   0ACC   DATRAN  0B54
ECHO    0019   ERRORS  00A5   EXIT    0A4E
GRAFIC  0B6C   GRAPHO  00A8   GREET   0ADC
HELLO   083B   LABEL   0230   LDIST   0916
LDBLOK  08F8   LDBYTE  091C   LDDONE  094E
LDGO    0905   LDING   0971   LDLOWO  0946
LDMOVE  0927   LDNOTZ  091A   LDPULL  094D
LDSHOW  092B   LDSTEP  095A   LDTEST  0962
LINSIZ  00A7   LMAXHI  0234   LMAXLO  0233
LMEMHI  0232   LMEMLO  0231   LNAME   0235
LOAD    08E8   MAX     001B   MAXHI   0009
MAXLO   0008   MEMHI   0007   MEMLO   0006
MIN     001C   OPBYTE  0887   OPEN    0880
OPEN1   087E   OPEXIT  08A9   OPFAIL  08A7
OPSTEP  0896   OPTYPE  08AC   PARITY  001A
POINT1  0998   POINT2  099C   QUOTE   00A6
RBEXIT  0ABB   RD8BIT  0A90   RDBAD   0AB3
RDBIT   0ABC   RDBITC  0AC3   RDBITW  0ABE
RDBYTE  0A90   RDEXIT  0ABA   RDROR   0A97
RDTEST  0AA7   RETRY   001D   RULES   0B04
SIZERR  08E4   SIZEXT  08E6   SIZMAX  08CB
SIZODD  08C7   SIZTST  08B9   STACK   001F
START   0810   SYNC8X  001E   TOKENS  0BAC
WAIT    0960   WCLR    096D
END OF ASSEMBLY
```

The Musicsystem

Keith Harrell

The Musicsystem from Mountain Hardware gives the Apple owner the ability to create original compositions or transcribe sheet music. The Musicsystem is comprised of two boards and one disk that contains the Music Player, Editor, Music Merger and some demo compositions to play. Also included with the system is a light pen to use in creating and editing music scores.

The Musicsystem hardware consists of two music boards that have sixteen programmable digital oscillators. The sixteen programmable oscillators are divided into two groups which represent the left and right channels. Each oscillator includes a software based volume control and an overall volume control for all sixteen oscillators. The system provides two jacks which are connected to the auxiliary input of your stereo. The system is also capable of driving most high quality stereo headphones without a amplifier.

The Musicsystem *editor* is the tool that allows you to enter your musical score. The editor displays the regular music staff to create your composition on the high resolution graphic screen. To enter notes you may use the paddles, the light pen, or the keyboard. After loading the editor, the screen is divided into two areas. They are the upper screen, which displays the score of the composition being edited, and the lower screen, which can display up to four different menus.

Each menu contains different options that can be selected by the light pen or paddles. The main menu contains a choice of twelve different note durations (from a whole note to a dotted thirty-second note) with each displayed in standard music notation symbols. The main menu allows the selection of note or chord mode and a delete forward or backward . You may also select six different rest durations. Accidentals may be selected from the main menu to be added to a note. The main menu allows for three other menus to be selected, one of which is the signature commands menu. This menu contains the commands that will allow you to select the choice of time, clef or key for the composition. The third menu contains a choice of chord accents, and a choice of dynamic accents, and the ability to tie notes together. The fourth menu provides the capability of applying dynamics to the score and specifying the tempo. You may also define the spatial location to determine which speaker the composition will come from. To make a selection from one of the menus, place the light pen on that box in the menu, rotate the game paddle until the marker is positioned in the box that is desired, or simply key in the command.

The editor also has a save and load command for future editing of a composition. A composition may have multiple parts and each part may be assigned a different instrument and a different speaker. There are six basic instruments available to choose from depending on the octave you have selected. They are a bass, organ, brass, gong, string, and woodwind. The brass and the gong can sound like two different instruments. If a high octave is assigned, then the brass will sound like a horn and if a low octave is assigned, then it will sound like a tuba. The gong will sound like a chime for low octaves and a bell for high octaves.

The *music merger* is used to merge composition files together to form a longer composition. This allows you to create small segments of a composition with the editor and merge them together to form a large composition file that can be played with the music player.

The *music player* is used to perform the composition that was created with the music editor. The music player also allows you to change the instruments assigned to each part of the composition and to reassign the speaker location for each part. Any composition to be played through the COMP file, which is the file created by the music editor, must be compiled first. This also is the job of the music player. After the music player compiles the composition, the file may be saved so that it does not have to be compiled each time you wish to listen to the piece. The file is now saved out on disk but as a play file which cannot be edited. The comp file is still on disk as the source file to which addition or changes may be made and then recompiled.

You do not need to know how to read music to use the Musicsystem, as I have never played any instrument and was able to copy sheet music into the editor and play it with satisfactory results. The manual supplied does not explain in detail how to read music, but with it and another book I found at a music store, I was able to do almost anything I wanted without much difficulty.

The present software is not compatible with the new DOS3.3. The minimum hardware required to use the Musicsystem is one disk drive and 48k of memory. The suggested retail price is $525.00. ©

Programming And Interfacing The Apple, With Experiments

Marvin L. De Jong
Department of Mathematics – Physics
The School of the Ozarks
Pt. Lookout, MO 65726

Introduction

When the Apple microcomputer is compared with other popular microcomputers, one of its most attractive features is the ease with which it can be interfaced to devices in the outside world. Particularly important in this connection are those eight beautiful card slots in the Apple. The "black box" philosophy of the designers of the TRS-80 leaves much to be desired in scientific, educational, or industrial applications.

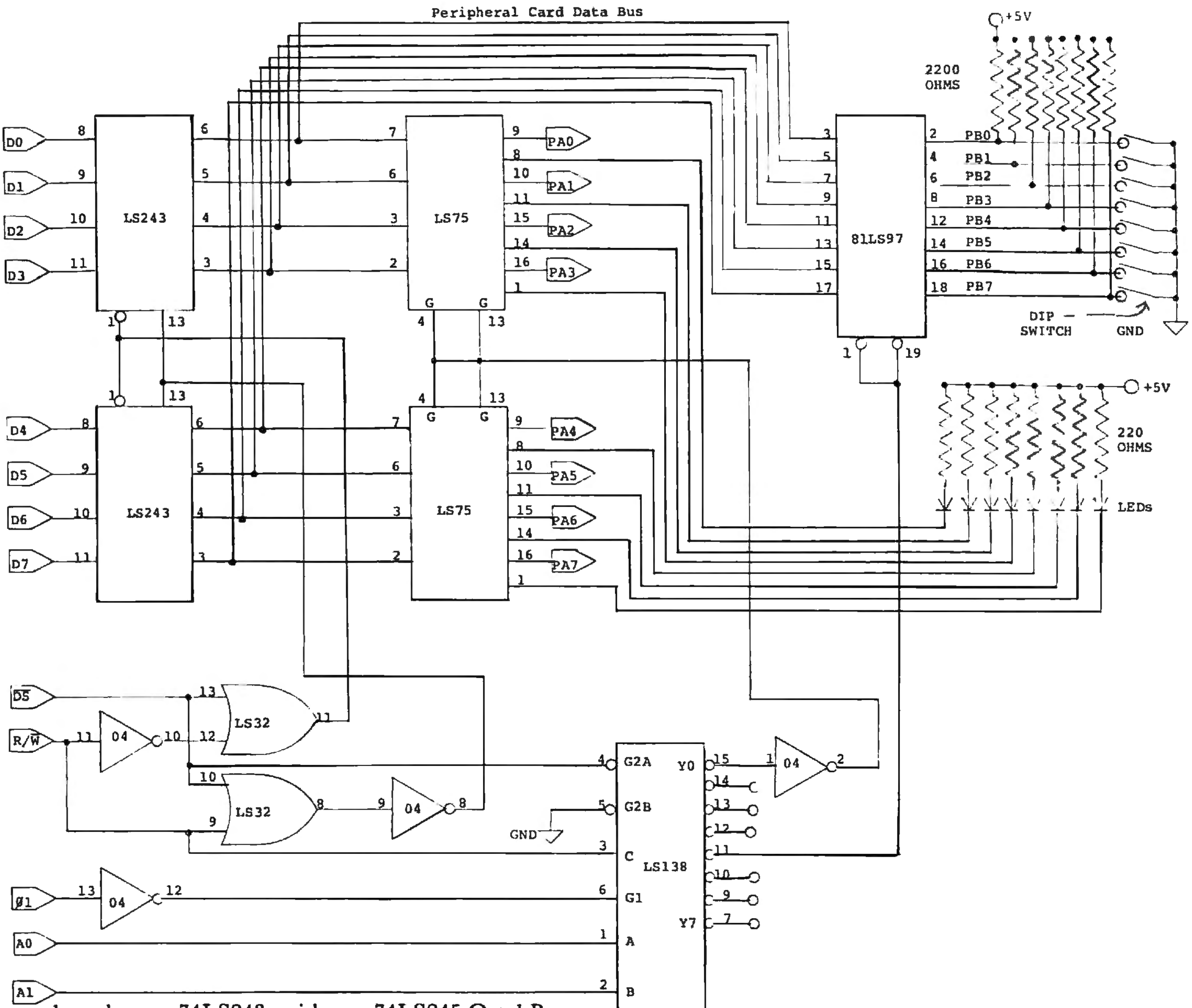
In this article we will describe a circuit to be built on a peripheral card that fits in any of the eight card slots in the Apple. The circuit provides the user with one eight-bit input port and one eight-bit output port (with possibilities for expansion). The circuit is built with readily available components, and the output port is also attached to eight LEDs so the user can visualize the state of the bits. The bit values of the input port may be controlled with an eight element DIP switch, or by devices of the user's own choice, such as an A/D converter.

My main reason for designing this circuit was to provide Apple owners with the experiments in my book **Programming & Interfacing the 6502, With Experiments.** This book was originally based on the KIM-1, SYM-1, and the AIM 65, but with the I/O board described in this article, the book can be used in conjunction with Apple computers. So, if you are interested in learning assembly language programming in conjunction with my book, this I/O board may make the task a little easier. If you are not interested in the book, the I/O board described here will be of interest if you wish to interface your Apple computer to devices like A/D and D/A converters, stepper motors, transmitters, and a variety of other devices.

The Circuit

The circuit diagram is shown in Figure 1. The circuitry on the Apple microcomputer develops a $\overline{\text{DEVICE SELECT}}$ pulse for each of the eight peripheral cards. For card number 0, this pulse occurs whenever addresses $C080 through $C08F appear on the address bus. For card number 1, the corresponding addresses are $C090 through $C09F, and so on for the other card numbers. That is, the device select pulse, $\overline{\text{DS}}$, is at logic zero for each address in the range $C0N0 through $C0NF, where N = $8 + CN and CN is the card number expressed in hexadecimal. These device select pulses can activate up to 16 I/O ports on each card.

In the circuit shown in Figure 1, the device select pulse ($\overline{\text{DS}}$) generated by the Apple is combined with the R/$\overline{\text{W}}$ signal generated by the 6502 to produce a signal that activates the 74LS242 bus transceivers in the direction from the Apple data bus to a peripheral card data bus when the R/$\overline{\text{W}}$ line is at logic zero. The lower 74LS32 OR gate and the 74LS04 INVERTER generate this signal. When the R/$\overline{\text{W}}$ line is at logic one during a 6502 READ cycle, the 74LS243 bus transceivers drive the Apple data bus from the peripheral card data bus, provided the $\overline{\text{DS}}$ signal is at logic zero, its active state. Thus, all data bus buffering is handled by the two 74LS243s and it is controlled by the $\overline{\text{DEVICE SELECT}}$ pulse and the R/$\overline{\text{W}}$ line coming from the Apple computer. If you want to reduce the chip count,

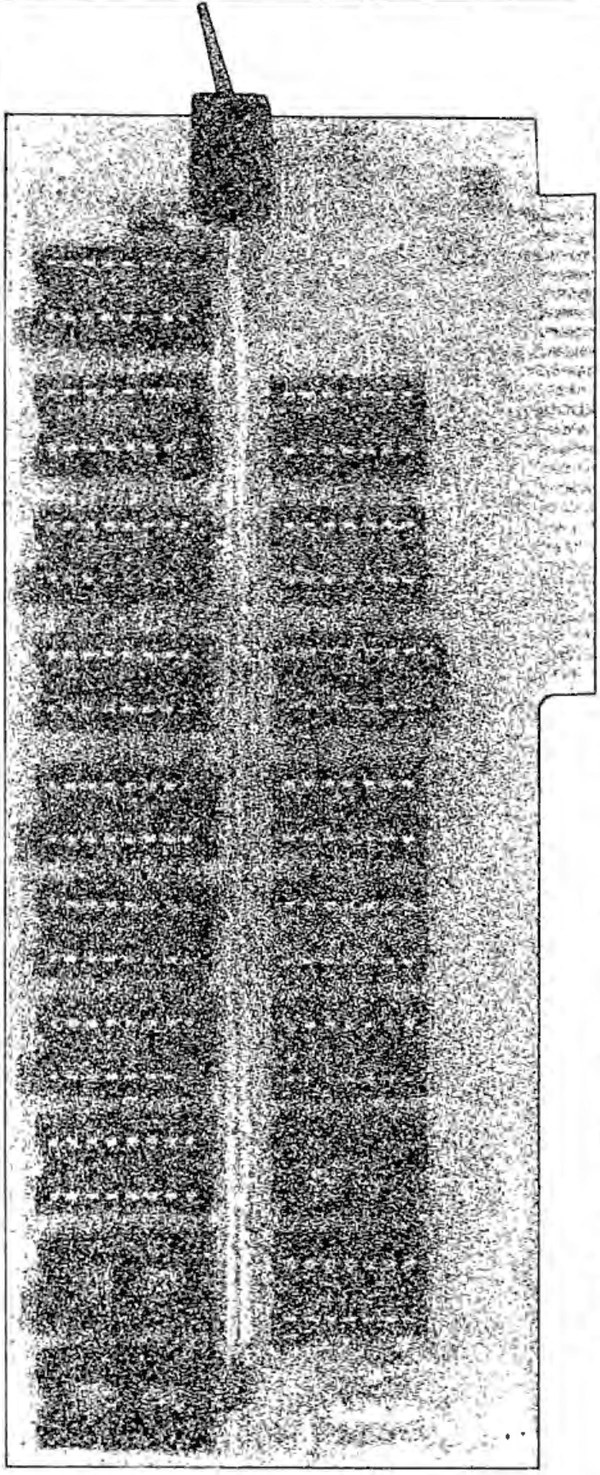


replace the two 74LS243s with one 74LS245 Octal Bus Transceiver, a 20-pin chip.

The $\overline{DS}$ pulse also activates the 74LS138 3-to-8 line decoder that is used to produce up to four $\overline{WRITE}$ pulses for output ports (Y_0 - Y_3 pins on the LS138) and four $\overline{READ}$ pulses for input ports (Y_4 - Y_7 pins on the LS138). In this application, only one $\overline{WRITE}$ pulse and one $\overline{READ}$ pulse are used. You may add more ports. Note that the R/$\overline{W}$ line is one of the lines decoded by the 74LS138. This idea originated in Gene Zumchak's Nuts and Volts column in Issue 2 of **compute II**. Thus, the lowest four output lines, Y_0 - Y_3, on the 74LS138 will be active only on WRITE cycles. The highest four output lines from the 74LS138, namely Y_4 - Y_7, will be active on READ cycles. In particular, Y_0 is active when the last nibble in the address is $0 ($C080, for example) and the 6502 is in a WRITE cycle with the R/$\overline{W}$ line at logic zero. The Y_4 line from the 74LS138 is active during a READ cycle and when the low-order nibble in the address is $4 ($C084, for example).

The two 74LS75 4-bit bistable latches are transparent to the data bus when the G inputs are at logic one. At the conclusion of the WRITE cycle the G inputs are brought to logic zero and the data on the data bus are latched into the Q outputs of the two 74LS75s. These eight pins for the output port we have called PORT A. The Q outputs activate the LEDs in the sense that the LED will glow if the bit value in Port A is one; otherwise they will not glow. If you want to reduce the chip count, replace the 74LS75s with octal latches such as the 74LS363 or 373, but you will have to eliminate the pretty LEDs or find another way to drive them. This completes the description of the output port.

The input port is an 81LS97 octal three-state buffer. It drives the peripheral card data bus when the Y_4 line on the 74LS138 is at logic zero. Thus, a LDA $C094 will result in a "read" of the input port if the peripheral card is in slot 1 of the Apple. We have called the input port, PORT B. The bit



16K Ram Expansion Board for the Apple II*

$195.00

- expands your 48K Apple to 64K of programmable memory
- works with Microsoft Z-80 card, Visicalc, LISA ver 2.0 and other software
- eliminates the need for an Applesoft* or Integer Basic ROM Card
- switch selection of RAM or mother board ROM language
- includes installation and use manual
- fully assembled and tested

Visa and MasterCard accepted
Shipping and handling will be added unless the order is accompanied by a check or money order
N.C residents add 4% sales tax

*Apple II and Applesoft are trademarks of Apple Computer, Inc

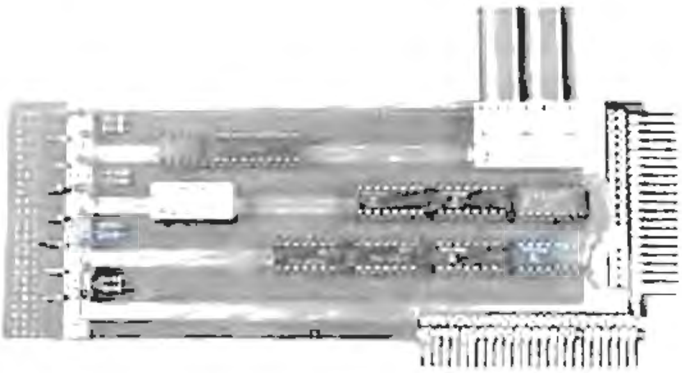
ANDROMEDA
INCORPORATED**
P.O. Box 19144
Greensboro, NC 27410
(919) 852-1482

**Formerly Andromeda Computer Systems

values of Port A many be controlled manually by the eight switches on the DIP switch package. If the input pin is connected to ground through the switch. a logic zero results; otherwise you get a logic one. If the input port is to be driven by some other circuit, then the pull-up resistors (2200 ohms) are not necessary. A 74LS244 octal buffer may be substituted for the 81LS97, but they are not pin-for-pin compatible. This completes our description of the input port, and the entire circuit.

Construction

You will need a peripheral card, the integrated circuits, a pin-out diagram for the peripheral card slots (see your Apple manual), soldering equipment, wire-wrap equipment, and several other parts. If you have never wired or built a circuit before, be sure you have someone with experience around who can help you. Two photographs indicate the parts layout that we used. Figure 2 shows the I/O card with the LEDs and switches installed. The pull-up resistors had not yet been wired. Figure 3 shows the I/O card with a ribbon cable DIP jumper connected to a 24 pin socket on the peripheral card. The 24 line cable can handle the I/O port lines and whatever power connections (+ 5V, GND, + 12V, or -12V) that you might want to steal from the Apple. Another cable alternative would be to use an edge connector on the peripheral card and have the cable exit through one of the slots behind the Apple.



I chose to wire-wrap most of the board, and I purchased my wire-wrap kit (WK-2) and my peripheral card (Vector #4609) from Jameco Electronics, 1355 Shoreway Road, Belmont, CA 94002. The remainder of the parts are available from a variety of mail-order houses, Jade Computer Products for example.

The best way to start is to make all of the power connections first. These are not shown in Figure 1, but they are completely described in Table 1. Also, although none are shown in the photographs, it is good practice to install one 0.01 microfarad capacitor between + 5V and ground for each two integrated circuits before doing the remainder of the wiring. I installed mine later because I did not have any available on the day that I wired the circuit and took the pictures. Next, wire the lines from the edge connector to the appropriate pins on the wire-wrap sockets. These lines all appear to the left of the circuit diagram in Figure 1, and a table of the pin numbers is given in Table 2. Finally, wire the connections between the pins on the sockets. The procedure just described insures that all the soldering is done at an early stage, when you are less likely to burn through several innocent wires while trying to solder another one. Again, if you have had little experience in wiring, find someone to help you lay out the circuit and get started wiring. Most people are anxious to demonstrate their expertise and be helpful.

Testing and Operation

To test the board first turn off the Apple and install the board in one of the slots, say slot 6 in which case the address of the output port is $ C0E0 (decimal 49376) and the address of the input port is $C0E4 (decimal 49380). (Actually, both ports respond to other addresses also, but that will not be of any concern here.) With the card installed and the Apple running in its monitor, write some number to the output port; that is, enter *C0E0:55 RETURN. The LEDs should display $55. Try some other numbers such as $01, $02, $04, $08, $10, $20, $40, and $80 if you wish to try all the LEDs in turn. To test the input port with the monitor simply use it to examine location $C0E4 by entering *C0E4 RETURN. The number you get should correspond to the switch settings. Try each switch in the logic one position in turn to verify that everything is working properly.

To test the I/O card using BASIC you must PEEK at the input port and POKE data to the output port. For example, a statement

```
10 Y = PEEK(49380)
```

returns the data at the input port as Y. The statement

```
20 POKE 49376, Y
```

will put the contents of Y in the output port, provided Y is not greater than 255. Try running this program:

```
10Y = PEEK(49380)
20 POKE 49376, Y
30 GO TO 10
```

It reads the input port and writes that number to the output port. Thus, the LEDs should follow the input switches. In the above test procedure, we are still assuming that the peripheral I/O card is in slot 6. If you select some other slot, then the addresses used above must be modified.

Concluding Remarks

You should be aware that you cannot use read-modify-write assembly language instructions (e.g., INC, DEC, ASL, LSR, etc.) to reference the output port because it does not include an output register. The best way to reference the output port is with STA, STX, or STY instructions. To demonstrate the other instructions, perform all your read-modify-write instructions on some R/W memory location (RAM), then transfer the contents of this location to the output port. For example this program segment demonstrates the ASL instruction.

```
ASL MEM1 Modify the MEM1 location.
LDA MEM1 Transfer it to the accumulator.
STA PAD Output the data to Port A
```

If you have any questions, please include a self-addressed stamped envelope, and I will be glad to respond.

Table 1. Integrated Circuit Information.

IC NUMBER	DESCRIPTION	QUANTITY	+5V	GND
74LS243	Quadruple Bus Transceiver	2	Pin 14	Pin 7
74LS32	Quadruple 2-Input Or Gates	1	Pin 14	Pin 7
74LS04	Hex Inverter	1	Pin 14	Pin 7
74LS138	3-to-8 Line Decoder	1	Pin 16	Pin 8
74LS75	4-bit Bistable Latch	2	Pin 5	Pin 12
81LS97	Octal Three-State Bus Driver	1	Pin 20	Pin 10

Table 2. Some Pin Numbers for the Apple Bus.

Label (Figure 1)	Pin Number on the Apple Bus
D0	49
D1	48
D2	47
D3	46
D4	45
D5	44
D6	43
D7	42
DEVICE SELECT DS	41
01	38
R/W	18
A0	2
A1	3

©

Player-Missile Graphics with the ATARI™ Personal Computer System

Chris Crawford

Anybody who has seen ATARI's Star Raiders™ knows that the Atari Personal Computer System has vastly greater graphics capabilities than any other personal computer. Owners of these computers might wonder if they can get their machines to do the fabulous things that Star Raiders does. The good news is that you can indeed write programs with graphics and animation every bit as good as Star Raiders. In fact, I think it's possible to do better. The bad news is that all this video wizardry isn't as easy to use as BASIC. The Atari computer is a very complex machine; mastering it takes a lot of work. In this article I will explain just one element of the graphics capabilities of the Atari Personal Computer System: player-missile graphics.

Player-missile graphics were designed to meet a common need in personal computing, the need for animation. To understand player-missile graphics you need to understand the old ways of doing animation on machines like the Apple. These machines use what we call pure playfield graphics, in which bits in RAM are directly mapped onto the television screen. You move an image across the screen by moving a pattern of bits through RAM. The procedure you must use is as follows: calculate the current addresses of the bit pattern, erase the bit pattern from these addresses, calculate the new addresses of the bit pattern, and write the bit pattern into the new addresses.

This can be a terribly slow and cumbersome process, particularly when you are moving lots of bits (large objects or many objects) or when the motion is complex. Consequently, most animation on computers like the Apple is limited to pure horizontal motion, pure vertical motion, small objects, or slow motion. Animation like you get in Star Raiders is utterly impossible.

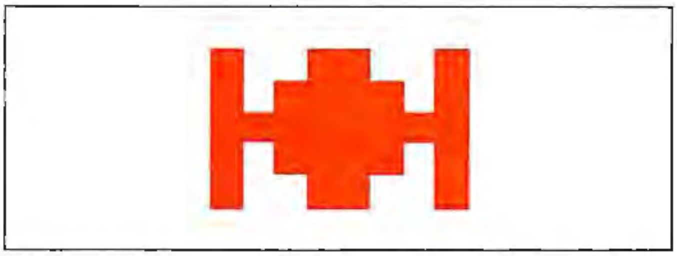
To understand the solution to this problem you must understand its fundamental cause. The screen image is a two-dimensional entity, but the RAM that holds the screen image is a one-dimensional entity. Images that are contiguous on the screen do not necessarily occupy contiguous RAM locations (see Figure 1). To move an image you must perform messy calculations to figure out where it will end up in RAM. Those calculations eat up lots of time. We need to eliminate these calculations by shortcutting past the 2d-to-1d transformation logjam. What we need is an image that is effectively one-dimensional on the screen and one-dimensional in RAM.

Let's set aside a table in RAM for this one-dimensional image. We'll call this table and its associated image a player. We'll have the hardware map this image directly onto the screen, on top of the regular playfield graphics. The first byte in the table will go onto the top line of the screen. The second byte will go onto the second line of the screen, and so on down to the bottom of the screen. Although I'm calling the image one-dimensional, it's actually 8 bits wide, because there are 8 bits in a byte. It's a straight bit-map; if a bit in the byte is turned on, then the corresponding pixel on the screen will be lit up. If the bit in the byte is turned off, then the corresponding pixel has nothing in it.

We can draw a picture with this scheme by turning the appropriate bits on or off. The picture we can draw is somewhat limited; it is tall and skinny, only 8 bits wide but stretching from the top of the screen to the bottom. Let's say we want to draw a picture of a little spaceship. We do this by storing zeros into most of the player RAM. We put the bits that form the spaceship into the middle of the player RAM so that it appears in the middle of the screen. See Figure 2 for a depiction of this process.

So far we don't have much: just a dinky image of a little spaceship. How do we get animation? We move it vertically with the same technique that other computers use. First we must erase the old image from RAM, then we draw in the new image. This time, however, the problem is much simpler. We move the image down by moving its bit pattern one byte forward in RAM. We move the image up by

This is the two-dimensional screen image



Here are the corresponding bytes in RAM (hexadecimal)

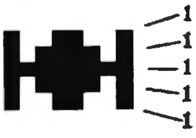
```
0  0  0
0  0  0
0  99 0
0  Bd 0
0  FF 0
0  Bd 0
0  99 0
0  0  0
0  0  0
```

This is how the bytes would be placed in one-dimensional RAM. Note how the bytes that make up the spaceship are scattered through the RAM. What a headache!

```
0
0
0
0
0
0
0
99
0
0
Bd
0
0
FF
0
0
Bd
0
0
Bd
0
0
99
0
0
0
0
0
0
0
```

moving its bit pattern one byte backwards in RAM. We use no crazy two-dimensional calculations, just a simple one-dimensional move routine. It's trivial in BASIC and easy in assembly language. Horizontal motion is even easier. We have a hardware register for the player called the horizontal position register. When we put a number into the horizontal position register, the player is immediately moved to that

Figure 2
How to draw in binary

graphical representation one byte 8 bits	binary representation	hexadecimal representation	decimal representation
	1 0 0 1 1 0 0 1	99	153
	1 0 1 1 1 1 0 1	Bd	189
	1 1 1 1 1 1 1 1	FF	255
	1 0 1 1 1 1 0 1	Bd	189
	1 0 0 1 1 0 0 1	99	153

horizontal position on the screen. Put a big number in and POW!--the player is on the right side of the screen. Put a little number in and POW!--the player is on the left side of the screen. Horizontal motion is achieved by changing the number you put into the horizontal position register. The two techniques for horizontal and vertical motion can be mixed in any way to produce any complex motion you desire.

The capabilities I have described so far are nice, but taken alone they don't give you much. That's why Atari added a long list of embellishments to this basic system which enormously extend its power. The first embellishment is that you have not just one, not two, not three, but FOUR (count 'em, FOUR) players available. This means that you can have four little spaceships flying around on the screen. They are all independent and so can move independently. The next embellishment is that each player has its own color register. Thus, you can set each player to a different color, completely independent of the colors in the playfield. This gives you the capability to put up to nine colors onto the screen, depending on your graphics mode. Next, you have the capability to make a player double or quadruple width. This doesn't change the eight-bit resolution of the player, but it does allow you to make him fatter or skinnier as you please. Next, you can select the vertical resolution of the player to either single line resolution (each byte occupies one scan line on the screen) or double line resolution (each byte occupies two scan lines on the screen. Next, you can select the image priorities of players versus playfield. Since both players and playfield will be imaged onto the same location on the screen you have to decide who has priority in the event of a conflict. You can set players to have higher priority than playfield, playfield to have higher priority than players, or several mixtures of player-playfield priority. This allows you to have players disappear behind playfield or vice-versa. Finally, you have tiny two-bit players called missiles. Each player has one missile associated with him. The missile takes the same color as the player but can move independently of the player. This allows bullets or other small graphics items. If you want, you can group the four missiles together to form a fifth player. They then get a separate color.

How do you use all of these fantastic capabilities? You might think that it would be terribly difficult to put all of this together into a program, but it isn't. Listing 1 shows a program that puts a player onto the screen and moves it around with the joystick. As you can see, the program is ridiculously short. Here's how it works:

Line 10 sets the background color to black (the better to see the player by). It also sets up our starting positions, X being the horizontal position and Y being the vertical position.

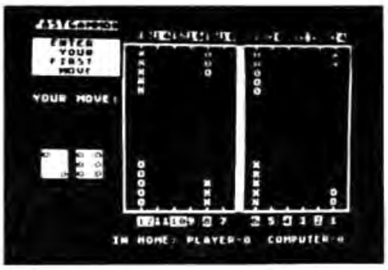
Line 20 finds the top of RAM and steps back eight pages to reserve space for the player-missile RAM. It then pokes the resultant page number into a special hardware register. This tells the computer where it will find the player-missile data. The players are arranged in memory as shown in Figure 3. Finally, line 20 keeps track of where the player memory is through the variable PMBASE. Because of this arrangement, this program will work on any Atari Personal Computer System, regardless of the amount of RAM in place. The number of pages by which you must step back (8 in this case) depends on how much memory your graphics mode consumes and whether you are in single-line resolution or double-line resolution. In any event, the number of pages

Listing 1
Program to demonstrate player-missile graphics

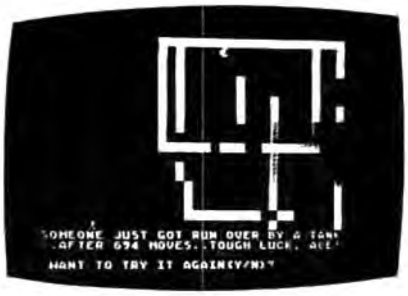
```
10 SETCOLOR 2,0,0:X=120:Y=48:REM Set background color and player position
20 A=PEEK(106)-8:POKE 54279,A:PMBASE=256*A:REM Set player-missile address
30 POKE 559,46:POKE 53277,3:REM Enable PM graphics with 2-line resolution
40 POKE 53248,X:REM Set horizontal position
50 FOR I=PMBASE+512 TO PMBASE+640:POKE I,0:NEXT I:REM Clear out player first
60 POKE 704,216:REM Set color to green
70 FOR I=PMBASE+512+Y TO PMBASE+516+Y:READ A:POKE I,A:NEXT I:REM Draw player
80 DATA 153,189,255,189,153
90 REM Now comes the motion routine
100 A=STICK(0):IF A=15 THEN GOTO 100
110 IF A=11 THEN X=X-1:POKE 53248,X
120 IF A=7 THEN X=X+1:POKE 53248,X
130 IF A=13 THEN FOR I=6 TO 0 STEP -1:POKE PMBASE+512+Y+I,PEEK(PMBASE+511+Y+I):NEXT I:Y=Y+1
140 IF A=14 THEN FOR I=0 TO 6:POKE PMBASE+511+Y+I,PEEK(PMBASE+512+Y+I):NEXT I:Y=Y-1
150 GOTO 100
```

SOFTWARE FOR THE ATARI 800* AND THE ATARI 400*

```
QUADRANT 8,2

COMMAND >
```



```
ENTER YOUR FIRST MOVE
YOUR MOVE:
IN HOME: PLAYER-0 COMPUTER-0
```



```
SOMEONE JUST GOT RUN OVER BY A TANK
AFTER 694 MOVES..TOUGH LUCK, ACE!
WANT TO TRY IT AGAIN(Y/N)?
```

TARI TREK™
By Fabio Ehrengruber

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

Cassette - $11.95 Diskette - $14.95

FASTGAMMON™
By Bob Christiansen

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

On cassette only - $19.95

TANK TRAP
By Don Ursem

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

Cassette - $11.95 Diskette - $14.95

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

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

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

On diskette only - $79.95

★ ★ ★ ★ ★ ★ ★ ★ ★

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

On cassette only - $24.95

★ ★ ★ ★ ★ ★ ★ ★ ★

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

Cassette - $11.95 Diskette - $14.95

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

WHERE TO GET IT: Call us at (213) 344 6599 for the name of the Quality Software dealer nearest you If necessary you may order directly from us Mastercard and Visa cardholders may place orders by telephone Or mail your check or bankcard number to Quality Software, 6660 Reseda Blvd, Suite 105 Reseda, CA 91335 California residents add 6% sales tax SHIPPING CHARGES Within North America orders must include $1 50 for first class shipping and handling Outside North America the charge for airmail shipping and handling is $5 00 Pay in U S currency

*Indicates trademarks of Atari Inc

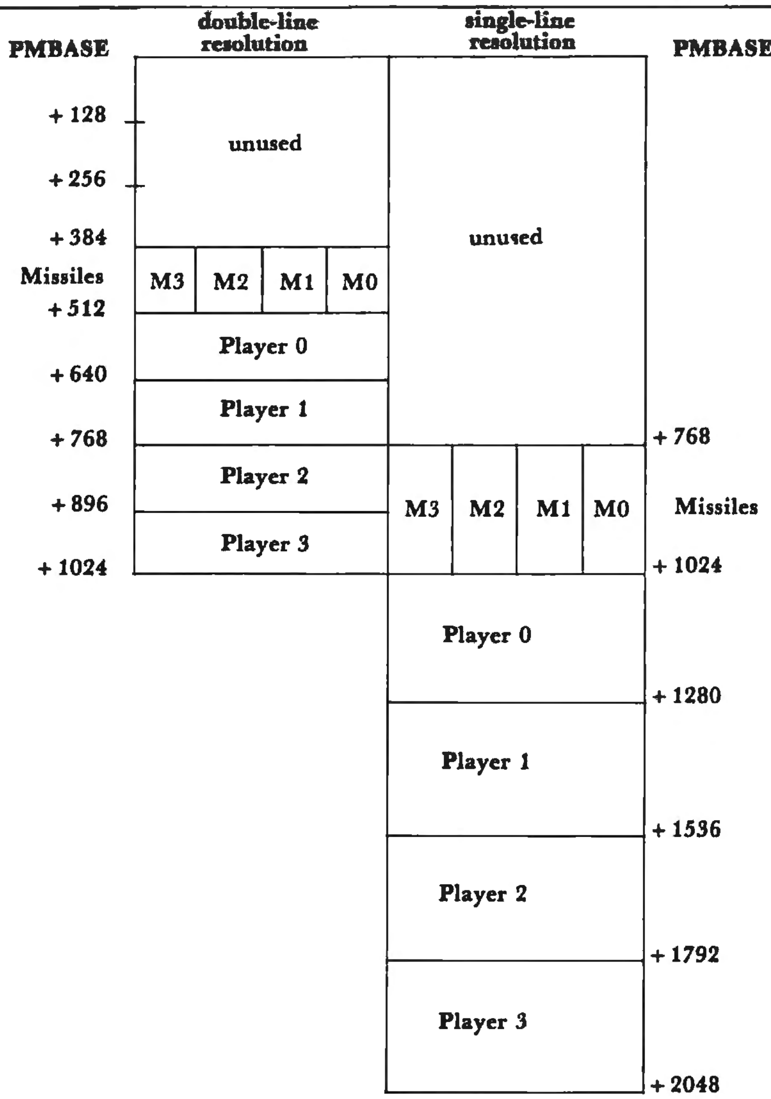


Figure 3
Player-missile graphics RAM positioning
PMBASE must be on 1K boundary for double-line resolution,
2K boundary for single-line resolution

Useful addresses
(all values in decimal)

559 put a 62 here for single line, a 46 for double line resolution
623 sets player/playfield priorities (only one bit on!)
1: all players have priority over all playfield registers
4: all playfield registers have priority over all players
2: mixed. P0 & P1, then all playfield, then P2 & P3
8: mixed. PF0 & PF1, then all players, then PF2 & PF3
704 color of player-missile 0
705 color of player-missile 1
706 color of player-missile 2
707 color of player-missile 3
53248 horizontal position of player 0
53249 horizontal position of player 1
53250 horizontal position of player 2
53251 horizontal position of player 3
53252 horizontal position of missile 0
53253 horizontal position of missile 1
53254 horizontal position of missile 2
53255 horizontal position of missile 3
53256 size of player 0 (0 = normal, 1 = double, 3 = quadruple)
53257 size of player 1 (0 = normal, 1 = double, 3 = quadruple)
53258 size of player 2 (0 = normal, 1 = double, 3 = quadruple)
53259 size of player 3 (0 = normal, 1 = double, 3 = quadruple)
53277 A 3 here enables player-missile graphics, a 0 disables them.
54279 put high byte of PMBASE here

you step back by must be a multiple of 4 for double-line resolution and a multiple of eight for single-line resolution.

Line 30 first informs the computer that this program will use double-line resolution. Poking a 62 into location 559 would give single-line resolution. The next instruction enables player-missile graphics; that is, it authorizes the computer to begin displaying player-missile graphics. Poking a 0 into location 53277 revokes authorization and turns off the player-missile graphics.

Line 40 sets the horizontal position of the player.

Line 50 is a loop that pokes 0's into the player 0 RAM area. This clears the player and eliminates any loose garbage that was in the player RAM area when the program started.

Line 60 sets the player's color to green. You can use any color you want here. The colors here correspond exactly to the colors you get from the SETCOLOR command. Take the hue value from the SETCOLOR command, multiply by 16, and add the luminosity value. The result is the value you poke into the color register.

Line 70 reads data bytes out of line 80 and pokes them into the player RAM area. The bytes in line 80 define the shape of the player. I calculated them with the process shown in Figure 2. Here you have lots of room for creativity. You can make any shape that you desire, so long as it fits into eight bits. You want more bits? Use four players shoulder to shoulder and you have 32 bits. You can make the loop longer to give more vertical height to your player.

These seven lines are sufficient to put a player onto the screen. If you only put in this much of the program, and ran it, it would show the player on the screen. The next lines are for moving the player with the joystick plugged into port 0.

Line 100 reads the joystick.

Line 110 checks to see if the joy the left. If so, it decrements the horizontal position counter and pokes the horizontal position into the horizontal position register. The line does not protect against bad values of the horizontal position (X<1 or X>255).

Line 120 checks to see if the joystick is pressed to the right. If so, it increments the horizontal position counter and pokes the horizontal position into the horizontal position register.

Line 130 checks to see if the joystick is pressed down. If so, it moves the player image in RAM forward by one byte. There are six bytes in the player image that must be moved. When it has moved them it increments the vertical position counter.

Line 140 performs the same function for upward motion.

Line 150 starts the joystick poll loop over again.

This program was written to help you understand the principles of player-missile graphics; as such it has many weaknesses. It also has much potential for improvement. You might want to soup it up in a variety of ways. For example, you could speed it up with tighter code or an assembly language subroutine. You might add more players; perhaps each could be controlled by a separate joystick. You could change the graphics shapes. You could make the colors change with time or position or how much fuel they have left or whatever. You could add missiles for them to shoot with. You could change width to give the impression of 3d motion that Star Raiders gives. You could add playfield priorities so they could move behind some objects but in front of others. The possibilities are almost limitless. ©

The Fluid Brush

Al Baker

This month I got carried away. I had so much fun changing and improving this program that it incorporates several hints on how to use the Atari plus another way to use the joysticks. Before digging into the program, though, let us see what it does.

Type in the program and run it. All but the bottom four lines of the screen turn black. Near the center of the black area is a white dot. Move joystick 0 and you can paint with the dot, just as if it was a brush dipped in white paint. The motion of the dot on the screen is quite slow. This is intentional. If the dot moved too fast, it would be hard to control. As your skill increases, you can speed the dot up.

Hold down the joystick button and move the joystick. Now the dot moves much faster, but it doesn't paint. It erases. You have a paint brush which moves quickly from one area of the screen to another, and yet paints slowly enough to give you complete control!

Unless you are Tom Sawyer, painting with a white brush can be quite boring. Let's change colors. As you probably know, you have three color registers available in graphics mode 7. When the program starts, you are painting with register 1 set to white. To pick another register, press either the 1,2, or 3 key. You are now using this register. But before you can paint, you must choose a color. Type in a color number between 0, for white, and 15, for light gold. Press RETURN. Table 1 lists the 16 color possibilities. Now as you move the joystick, you are painting with this new color.

What's Going On

The program is initialized between lines 1000 and 2030. The beginning location of the dot is position (X = 90, Y = 48); its color, C, is 0; its color register, R, is 1; and its brightness, L, is 10.

Line 1090 opens the keyboard for input. This statement is necessary if you want to read single Ascii characters from the keyboard without using the INPUT statement. The number 1 is my choice for the file number. I could have chosen anything between 1 and 5. The 4 means input, the 0 is required, and the "K:" means the input is from the keyboard.

Look at Diagram 1. The joystick returns numbers between 5 and 15 depending on its position. The actual number is used as the subscript of arrays XD and YD to determine how the X and Y positions of the dot on the screen are to be changed. For example, if the joystick is pushed away from the user and to the right, the number is 6. XD(6) is equal to 1 and YD(6) equals -1. The dot would move one position right (+ 1) and one position up (-1) on the screen. The arrays XD and YD are initialized in lines 1110 to 2030.

The main program loop starts at line 150. Look closely at line 160. This statement determines the speed of the dot on the screen. If the button on joystick zero is pushed, STRIG(0) = 0 and S will be 0. If the button isn't pressed, STRIG = 1 and S = 100. Line 170 uses the variable S as the alarm on the delay timer. Finally, lines 150 and 180 cause the dot to blink. Line 200 makes sure that if the button is pressed, then the dot erases, or leaves a black spot, when the joystick is moved.

The rest of the program loop is between lines 250 and 310. Line 250 gets the value of the joystick. This value is used to modify the X and Y positions of the dot as previously discussed. Once these values are computed, line 290 places the dot in its new location.

The statement on line 280 keeps the dot from running off the TV screen. If the PLOT statement tries to put the dot off the screen, an error results. The trap on line 280 branches the computer to the routine at line 3000. That routine adds one to Y if Y is above the screen (Y<0) or subtracts one from Y if Y is below the screen (Y>79). Likewise, it adjusts X if X is to the left (X<0) or right (X>159) of the screen. Finally, the routine jumps back to line 280 to set the trap again and plot.

The user can type in a new register number and color. Line 300 scans the keyboard each time the program loops to see if the artist is ready to change colors. If location 764 isn't equal to 255, then a key has been pressed. The routine beginning on line 4000 is called on to respond to the artist's request.

The first thing done by the keyboard routine is get the Ascii value of the key pressed. The GET statement must use the same number as the open statement on line 1090 and it puts the value of the key in the variable R. The Ascii value for a one is 49 and for a three is 51. If the key is not between these two, it is ignored and another is required. Once a proper key has been pressed, line 4020 converts it into the numbers 1, 2, or 3 and line 4030 sets plotting to that color register.

The POSITION statement does not control the location of print statements in the text window when graphics modes 1 thorugh 8 are chosen. This is done by poking values into memory locations 656 and 657. Poking a number between 0 and 3 into location 656 will position a statement vertically on the bottom four text lines. Poking a number between 0 and 39 into location 657 will position a print statement between columns 0 and 39 horizontally on the screen. Line 4040 positions the next print statement on the third line of the text area at the bottom of the screen.

Since the print statement on line 4050 is always printed in the same location, it is necessary to erase any previous answers. This is done by including four spaces followed by four back-arrows after the word COLOR. To insert a back-arrow, or any arrow, in a print statement, press the ESC key before typing the arrow key.

Conclusion

I would like to thank Dick Ainsworth for his idea about using two different speeds on the joystick to control different functions, and I'd also like to thank William Bailey for his idea on using arrays to simplify the conversion of joystick values into directions. If you would like to share your ideas with other readers, send them in. If I use them, you will also be acknowledged.

Al, Baker, Programming Director, The Image Producers, Inc.

Table 1: The Atari Colors

NUMBER	COLOR	NUMBER	COLOR
0	Gray	8	Blue
1	Gold	9	Gray blue
2	Orange	10	Turquoise
3	Red	11	Olive green
4	Pink	12	Green
5	Violet	13	Yellow green
6	Purple	14	Brown
7	Light blue	15	Light gold

Diagram 1: The Joystick Control Arrays

CHANGE IN X -- XD

XD(10) = -1 XD(14) = 0 XD(6) = 1
XD(11) = -1 XD(15) = 0 XD(7) = 1
XD(9) = -1 XD(13) = 0 XD(5) = 1

CHANGE IN Y -- YD

YD(10) = -1 YD(14) = -1 YD(6) = -1
YD(11) = 0 YD(15) = 0 YD(7) = 0
YD(9) = 1 YD(13) = 1 YD(5) = 1

```
10 REM THE FLUID BRUSH
20 REM
30 REM
40 REM GO SET UP CONDITIONS
50 REM
60 GOSUB 1000
120 REM
130 REM GET BUTTON FOR SPEED
140 REM
150 COLOR 4:PLOT X,Y
160 S=100*STRIG(0)
170 FOR I=1 TO S:NEXT I
180 COLOR R:PLOT X,Y
190 REM
200 IF S=0 THEN COLOR 4:PLOT X,Y:COLOR R
220 REM
230 REM MOVE DOT IF JOYSTICK MOVED
240 REM
250 J=STICK(0)
260 Y=Y+YD(J)
270 X=X+XD(J)
280 TRAP 3000
290 PLOT X,Y
300 IF PEEK(764)<>255 THEN GOSUB 4000
310 GOTO 150
970 REM
980 REM SET UP CONDITIONS
990 REM
1000 GRAPHICS 7
1010 X=90
1020 Y=48
1030 C=0
1040 L=10
1050 R=1
1060 SETCOLOR R-1,C,L
1070 COLOR 1
1080 PLOT X,Y
1090 OPEN #1,4,0,"K:"
1100 DIM XD(15),YD(15)
1110 FOR I=1 TO 15
1120 READ N:XD(I)=N
1130 READ N:YD(I)=N
1140 NEXT I
1150 RETURN
2000 DATA 0,0,0,0,0,0,0,0
2010 DATA 1,1,1,-1,1,0,0,0
2020 DATA -1,1,-1,-1,-1,0,0,0
2030 DATA 0,1,0,-1,0,0
2970 REM
2980 REM CATCH MOTION OFF THE SCREEN
2990 REM
3000 Y=Y+(Y<0)-(Y>79)
3010 X=X+(X<0)-(X>159)
3020 GOTO 280
3970 REM
3980 REM CHANGE COLOR
3990 REM
4000 GET #1,R
4010 IF (R<49)+(R>51) THEN 4000
4020 R=R-48
4030 COLOR R
4040 POKE 556,2
4050 PRINT "COLOR     ←←←←";
4060 INPUT C
4070 SETCOLOR R-1,C,L
4080 RETURN
```

©

An ATARI Tutorial
Atari Disc Menu

Len Lindsay

Anyone with an ATARI disk will really appreciate this program. You will probably put a copy of MENU on each of your diskettes.

MENU will display the programs on the diskette along with an ID number (1-44). It then asks you which program you wish to RUN. If you wish to RUN program number 8, you simply answer 8. It then LOADS and RUNs that program. No more hassles trying to remember exactly what name you used for the program, or typing the name exactly. MENU does it all for you.

Since MENU uses some special techniques, I will explain how it works. You should be able to apply many of these concepts to your own programs.

LINE 10-11 — Dimension the STRINGs. ARRAY$ will hold all the names of the programs on the disk (12 characters per name). FILE$ and NAME$ are used for the program names. DISK$ is used to hold the drive number prefix.

LINE 15 — Set the margins to default, in case the previous program used different ones.

LINE 20 — Use GRAPHICS 0 full screen text mode. It also clears the screen for you.

LINE 30 — Turn the cursor off — it looks nicer while writing the program names on the screen.

Line 40 — Set the color registers to the preferred colors. A light orange background with warm brown letters is the easiest on your eyes (so that is what I use).

LINE 50 — Set DISK$ to the disk drive to be used. See modification notes to make this more flexible.

LINE 60-70 — "D1:*.*" will refer to the disks directory. It is a two step process to add the DISK$ with "*.*" and call it NAME$.

LINE 100 — Open the disk directory for a READ. This line should be useful for other applications.

LINE 110 — Initialize the counter which counts each program as it is read from the directory. This also acts as the program ID number.

LINE 120 — READ one file from the directory. A program entry is 17 characters long. It is two spaces, 8 characters for name, 3 characters for extension, one space, 3 characters for sectors used. After all the programs is a separate record of the number of free sectors left on the diskette.

LINE 130 — Check if this is the short record of tracks left on diskette, if it is then we are done and should go on to the next part starting at line 500.

LINE 140 — Since we read in another program name, add one to the counter.

LINE 150 — If this is the 23rd program, we must switch to the right half of the screen (prevent scrolling and fit more on the screen this way). To do this we set the left margin to 20 and position the cursor at the top line, 20th spot.

LINE 160 — Check if the screen is completely filled with program names (44 is the maximum display allowed on one screen). If it is full, ignore all the rest, adjust the counter accordingly. See modification notes for other ideas.

LINE 200 — Initialize the name field. To manipulate the string by character position, the positions all must exist -thus initializing to " " (null) will not work. Note the extention dot is in position 9.

LINE 210 — If there is no extension get rid of the dot in position 9.

LINE 220 — Assign the program name from FILE$ which we just READ from the diskette. This is only the first 8 characters of the name, not including the extension.

LINE 230 — Assign the extension of the program name. If there is no extension, we still can assign it since the dot has already been removed.

LINE 300 — To keep a justified column of ID numbers, we must allow for ONE digit numbers. So if the number is less than 10 print an extra space.

LINE 310 — Print the ID number followed by) and a space.

LINE 320 — Print the program name.

LINE 400 — Add the name on to the ARRAY$ we are building. It can now be referenced by number times 12 (since every name is exactly 12 characters long).

LINE 410 — Processing complete for the program just read. Go and do the next one.

LINE 500 — Set the trap to come back and redo the input if an error occurs.

LINE 505 — CLOSE the file used to input the programs from the directory.

LINE 510 — Turn the cursor back on for the INPUT request.

LINE 520 — Position the cursor on the message line (line 22). First print a line delete to erase the previous message. Then print the current message. End the message with a BEEP (control 2).

LINE 525 — Set the left margin back to default so the next program will not be affected.

LINE 530 — INPUT the ID number of the program to be RUN.

LINE 540 — Get rid of any fractions.

LINE 550 — If the choice was not in the range available, go and ask again.

LINE 600 — Start FILE$ with the disk number. The rest of the name is assigned in line 630.

LINE 610 — Assign the name of the program chosen to NAME$ (taken from the ARRAY$ we just put together).

LINE 620 — Start a loop to go through the whole 12 character program name and remove all spaces (spaces cannot be imbedded within a program name when you ask for a LOAD or RUN).

LINE 630 — Add the characters in the program name one at a time to FILE$. Ignore spaces.

LINE 640 — Do the next character.

LINE 700 — Set the trap to line 900 to print a can't load message if there is a disk error.

LINE 720 — Position the cursor to the message line. First do a line delete to erase the previous message. Then print the message LOADING with the file name. Then print a BEEP (control 2).

LINE 730 — RUN the program and spring the trap.

LINE 900 — Print message the program can't be run (maybe diskette was switched or removed since the directory was read).

LINE 910 — Pause to allow message to be read.

LINE 920 — Go and ask for program to RUN again.

Possible Modifications

MENU is set up to work with disk drive 1. It is easy to have it work with both drive 1 and drive 2, and even alternate between them for a wider MENU choice. Line 50 sets the disk drive number prefix to be used by the MENU program. Some possible modificiations follow; the first asks you which drive to use for the MENU, while the second can flip back and forth from drive to drive. I have implemented the second set of modifications and find it works quite well. Either way, it seems that it doesn't like trying to give you a MENU for an empty drive.

Modification Set 1 — Ask Which Drive

```
50 PRINT "[CLEAR] WHAT DISK DRIVE TO USE)";
51 OPEN #1, 4, 0, "K:" :REM OPEN KEYBOARD FOR
GET
52 TRAP 52 : GET#1, DRIVE : REM GET KEY PRESSED
ATASCII VALUE
53 NAME$ = CHR$(DRIVE) : REM CONVERT TO
STRING - USE NAME$ SINCE IT IS DIMed
54 IF NAME$<"1" OR NAME$>"4" THEN 52 : REM
TRY AGAIN
55 PRINT NAME$ : REM PRINT THE REPLY
56 CLOSE #1 : REM CLOSE THE FILE
57 DISKS$ = "D1:" :REM INITIALIZE STRING
58 DISK$(2,2) = NAME$ :REM INSERT DRIVE NUMBER
```

Modifications For Alternating Drives

```
17 DRIVE = 2 : REM INITIALIZE FOR A TWO DRIVE
SYSTEM - DRIVE 1 WILL BE FIRST
18 DISK$ = "D1:" : REM INITIALIZE
50 DRIVE = 3-DRIVE : REM SWITCH DRIVES, WILL DO
DRIVE 1 FIRST
55 DISK$(2,2) = STR$(DRIVE) : REM PUT CORRECT
DRIVE NUMBER INTO DISK$
59 TRAP 50 : REM TRAP DISK ERROR
105 ARRAY$ = " " : REM INITIALIZE
115 PRINT "[CLEAR]"; : REM CLEAR SCREEN
520 POKE 82,2 : REM LEFT MARGIN TO DEFAULT
525 POSITION 2,22 : PRINT "[DELETE LINE]0 = NEXT
DRIVE WHICH TO RUN[CONTROL 2]";
535 IF CHOICE = 0 THEN 50 : REM SWITCH DRIVES
ON CHOICE OF 0
```

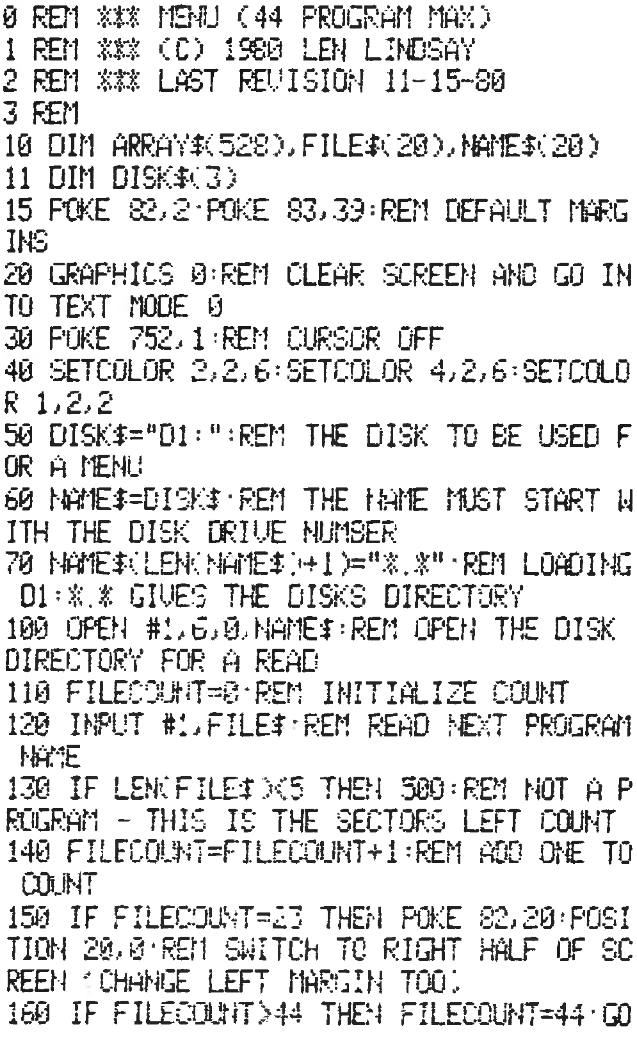
Another modification you may wish to make has to do with the ability to jump into DOS directly from MENU. If you try to RUN it as your MENU choice, it will say "can't run dos". Thus if you think you may need to jump directly to DOS add this line:

```
615 IF NAME$ = "DOS .SYS" THEN DOS
```

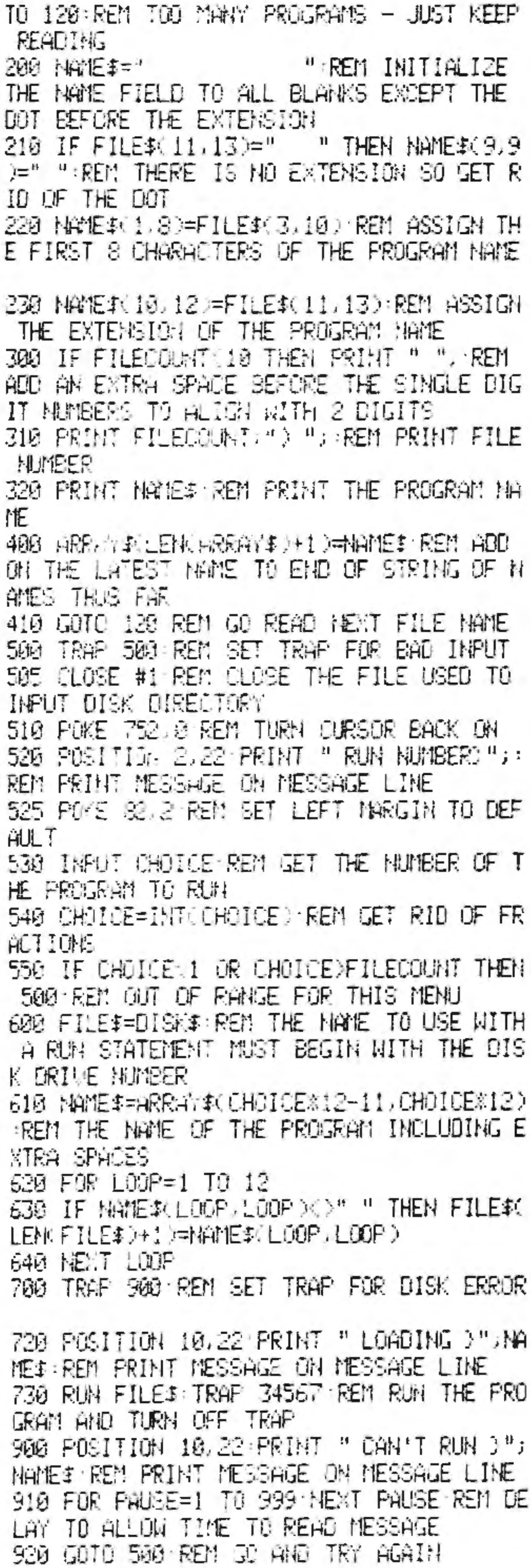
Modifications To Overcome The 44 Program Limit

The MENU can only display 44 program choices at one time, thus line 160 checks if the screen is full(44). If it is, it skips all the rest of the programs. In practice this should not be a problem since most diskettes will be filled before they reach the 45th program unless the programs are all short.

Modifications might be made so that after 44 programs, they no longer are printed on the screen, but still are added to ARRAY$ with FILECOUNT continuing its count. The DIM in line 10 for ARRAY$ should be increased accordingly. The message line (520-525) should also be appropriately changed. Perhaps a choice of 99 would mean "display second screen of menu". A subroutine could calculate what program number to start with (filecount minus 43) and another subroutine could print the menu from ARRAY$ as appropriate. ©



```
0 REM *** MENU (44 PROGRAM MAX)
1 REM *** (C) 1980 LEN LINDSAY
2 REM *** LAST REVISION 11-15-80
3 REM
10 DIM ARRAY$(528),FILE$(20),NAME$(20)
11 DIM DISK$(3)
15 POKE 82,2:POKE 83,39:REM DEFAULT MARG
INS
20 GRAPHICS 0:REM CLEAR SCREEN AND GO IN
TO TEXT MODE 0
30 POKE 752,1:REM CURSOR OFF
40 SETCOLOR 2,2,6:SETCOLOR 4,2,6:SETCOLO
R 1,2,2
50 DISK$="D1:":REM THE DISK TO BE USED F
OR A MENU
60 NAME$=DISK$:REM THE NAME MUST START W
ITH THE DISK DRIVE NUMBER
70 NAME$(LEN(NAME$)+1)="*.*":REM LOADING
 D1:*.* GIVES THE DISKS DIRECTORY
100 OPEN #1,6,0,NAME$:REM OPEN THE DISK
DIRECTORY FOR A READ
110 FILECOUNT=0:REM INITIALIZE COUNT
120 INPUT #1,FILE$:REM READ NEXT PROGRAM
 NAME
130 IF LEN(FILE$)<5 THEN 500:REM NOT A P
ROGRAM - THIS IS THE SECTORS LEFT COUNT
140 FILECOUNT=FILECOUNT+1:REM ADD ONE TO
 COUNT
150 IF FILECOUNT=23 THEN POKE 82,20:POSI
TION 20,0:REM SWITCH TO RIGHT HALF OF SC
REEN (CHANGE LEFT MARGIN TOO)
160 IF FILECOUNT>44 THEN FILECOUNT=44:GO
```



```
TO 120:REM TOO MANY PROGRAMS - JUST KEEP
 READING
200 NAME$="            ":REM INITIALIZE
THE NAME FIELD TO ALL BLANKS EXCEPT THE
DOT BEFORE THE EXTENSION
210 IF FILE$(11,13)="   " THEN NAME$(9,9
)=" ":REM THERE IS NO EXTENSION SO GET R
ID OF THE DOT
220 NAME$(1,8)=FILE$(3,10):REM ASSIGN TH
E FIRST 8 CHARACTERS OF THE PROGRAM NAME

230 NAME$(10,12)=FILE$(11,13):REM ASSIGN
 THE EXTENSION OF THE PROGRAM NAME
300 IF FILECOUNT<10 THEN PRINT " ",:REM
ADD AN EXTRA SPACE BEFORE THE SINGLE DIG
IT NUMBERS TO ALIGN WITH 2 DIGITS
310 PRINT FILECOUNT;") ";:REM PRINT FILE
 NUMBER
320 PRINT NAME$:REM PRINT THE PROGRAM NA
ME
400 ARRAY$(LEN(ARRAY$)+1)=NAME$:REM ADD
ON THE LATEST NAME TO END OF STRING OF N
AMES THUS FAR
410 GOTO 120:REM GO READ NEXT FILE NAME
500 TRAP 500:REM SET TRAP FOR BAD INPUT
505 CLOSE #1:REM CLOSE THE FILE USED TO
INPUT DISK DIRECTORY
510 POKE 752,0:REM TURN CURSOR BACK ON
520 POSITION 2,22:PRINT " RUN NUMBER? ";:
REM PRINT MESSAGE ON MESSAGE LINE
525 POKE 82,2:REM SET LEFT MARGIN TO DEF
AULT
530 INPUT CHOICE:REM GET THE NUMBER OF T
HE PROGRAM TO RUN
540 CHOICE=INT(CHOICE):REM GET RID OF FR
ACTIONS
550 IF CHOICE<1 OR CHOICE>FILECOUNT THEN
 500:REM OUT OF RANGE FOR THIS MENU
600 FILE$=DISK$:REM THE NAME TO USE WITH
 A RUN STATEMENT MUST BEGIN WITH THE DIS
K DRIVE NUMBER
610 NAME$=ARRAY$(CHOICE*12-11,CHOICE*12)
:REM THE NAME OF THE PROGRAM INCLUDING E
XTRA SPACES
620 FOR LOOP=1 TO 12
630 IF NAME$(LOOP,LOOP)<>" " THEN FILE$(
LEN(FILE$)+1)=NAME$(LOOP,LOOP)
640 NEXT LOOP
700 TRAP 900:REM SET TRAP FOR DISK ERROR

720 POSITION 10,22:PRINT " LOADING >";NA
ME$:REM PRINT MESSAGE ON MESSAGE LINE
730 RUN FILE$:TRAP 34567:REM RUN THE PRO
GRAM AND TURN OFF TRAP
900 POSITION 10,22:PRINT " CAN'T RUN >";
NAME$:REM PRINT MESSAGE ON MESSAGE LINE
910 FOR PAUSE=1 TO 999:NEXT PAUSE:REM DE
LAY TO ALLOW TIME TO READ MESSAGE
920 GOTO 500:REM GO AND TRY AGAIN
```

Using the Atari Console Switches

James L. Bruun

The colored console switches to the right of the typewriter keyboard are just the ticket for programs with special features. The names seem to indicate just the kind of things one might wish to do in a program. OPTION - What better key to step through a choice of options. SELECT - After stepping through the options, this key could be used to select the current option. START - This key might be used to transfer control back to the beginning of a sequence or to start the program over again.

The problem is, how does one read these keys? Well, read on, here is a method that works well for me. First, we note that memory location 53279 is used to indicate the condition of all three switches. It's done like this. If we just PEEK(53279) with no switches pressed, we find a seven. Holding down one or more of the keys while doing our PEEK returns us a different number. The table below summarizes the values returned when a console key is pressed. X means that the key or keys are pressed.

Table 1

KEY VALUE	0	1	2	3	4	5	6	7
OPTION	X	X	X	X				
SELECT	X	X			X	X		
START	X		X		X		X	

Now lets use this knowledge in a program.

```
10 DIM DISPLAY$(23)
20 PRINT "(CLEAR)": POKE 752,1
30 POSITION 5,5
40 KEYS = PEEK(53279)
50 ON KEYS + 1 GOSUB 100,110,120,130,140,150,160,170
60 PRINT DISPLAY$
70 GOTO 30
100 DISPLAY$ = "OPTION + SELECT + START" :
    RETURN
110 DISPLAY$ = "OPTION + SELECT         " :
    RETURN
120 DISPLAY$ = "OPTION + START          " :
    RETURN
130 DISPLAY$ = "OPTION                  " :
    RETURN
140 DISPLAY$ = "SELECT + START          " :
    RETURN
150 DISPLAY$ = "SELECT                  " :
    RETURN
160 DISPLAY$ = "START                   " :
    RETURN
170 DISPLAY$ = "NO KEYS ARE PRESSED     " :
    RETURN
```

Of course the subroutines here are very simple, but this method can easily be expanded to fit your needs. ©

The ATARI Disk Operating System

Roger Beseke

Now that you have your ATARI 810 disk system up and running and have undoubtedly saved and entered numerous programs and data, you are probably wondering what else this machine can do? Well, to date Atari has not released their DOS system manual, but there is a preliminary manual which is available and contains a wealth of information. The purpose of this article is to bring into the light some of the features hidden away in the preliminary manual.

As we all know, after having the disk up and running, there is a disk system menu which is displayed upon entry of the command "DOS,RETURN". Some of these commands are straight forward and require little or no explanation, but we are going to take a look at all of them.

There are two neat characters we must discuss before we go into the DOS menu of commands. They are the asterisk (*) and the question mark (?). When these characters are used in a DOS command, they are referred to as wild carding. They allow excellent flexibility which can be used to great benefit or dismay depending what the operater is using the wild card character for. It probably goes without saying that these characters should not be used in a file label.

In the ATARI DOS, the (*) is used to free form a file name for most of the commands. The asterisk can follow a portion of a label in either the main file label or the extension. Note: The file name does not have to be eight characters to use the extension. The asterisk can be used in numerous ways to provide as many results. I will cover a few here and leave the rest to your imagination. By the way, all the commands in this article are in quotes. If the command requires quotes, there will be double quotation marks. Also when return is spelled out in caps, it means the "RETURN" key is to be pressed.

A command of the form "*.*" will display all files on the screen if used with the disk directory command (A). A command of the form "PROG*.*" would list all programs that met the first four character format. Similarly the command "*.U*" would only list files that had an extension that met that format. Note: Once an asterisk is used in either the main or extension field, all characters following it are ignored.

The (?) in the ATARI DOS is used to set a character to a do not care condition when wild carding is used. The following example, "WORD?S.*", shows that all files having the form "WORD" and any character in the don't care character field will be operated on. These wild card characters can be used any place in a legal label field.

Disk directory (A): The disk directory takes care of finding and listing the files of a diskette. The files may be listed on the screen or on your ATARI 820 printer. It is common knowledge that to get a display of the files on a particular disk you must issue the command "A RETURN RETURN" and they are displayed on the monitor. This is fine if you do not have many files or if you want to see all the files there are on the disk. If you do not want to see them all, there are commands that can be sent to select a certain group of files. They also can be printed on the printer. To get a hardcopy, issue a command of ",P:" before the second "RETURN". A command of the form "RA*.B?,P:RETURN" will list all files with the first two characters "RA" in the main field and characters in the extension which begin with a "B" followed by one character.

Run cartridge (B). This command exits the DOS and executes in the left cartridge if one is inserted. It will not exit the DOS if a cartridge is not inserted in the left slot.

Copy (C). The copy command enables the operator to copy a file from one device to the disk or copy a file from one disk to another file on another disk. For instance, a command of the form "D1"FILE,D2:PROG" will copy a file named file from disk one to a file named prog on disk two. You can write a file from the screen editor to a disk file by a command similar to one of the form "E:,NAME". This command must be terminated with a "CTRL 3" key entry.

Delete (D). The delete command does allow wild card commands and can take the form of any of the previous examples. The DOS displays a cue to the operator to delete the file shown. The operator makes the appropriate entry and the DOS brings up another file if there are wild cards used and files that meet the wild card form. A typical deletion of all files with an extension of B1 thru B34 could be deleted one at a time with the command "*.B?? RETURN". If "/N" is appended to the command, it will delete the appropriate files without a cue, so be careful. It must be remembered that locked files cannot be deleted.

Rename (E). The rename command allows you to change the name of a file to another and wild cards are allowed. A typical command would be "FILE,KEEPFILE RETURN". This command will change the name of the present file "FILE" to "KEEPFILE". It must be noted that extreme care is recommended with this command when using wild cards because you can end up with a group of files with the same name.

Lock (F). The lock command as mentioned previously keeps you from inadvertantly writing to or deleting those files. A locked file can be recognized readily in the disk directory mode because of the asterisk ahead of the file name. Wild cards are also allowed in this command. A typical command to lock all files would be "*.*RETURN".

Unlock (G). The unlock command is the inverse of the previous lock command and the same protocol is allowed. But again a word of caution using wild cards, you may be unlocking something you do not want to.

Write "DOS" File (H). This command writes the DOS on a formatted disk so that it can be booted into the computer at turn on. This command allows you to make all your disks boot loadable and gives you a backup for the DOS.

Format Disk (I). This command is required for all new disks before they can be written on. The DOS ques the operator as to which disk to format. Again a double check is made to make sure that is the disk the operator wants formatted because if the wrong one is formatted, all files are lost on that disk.

Duplicate Disk (J). The duplicate disk command allows you to make a copy of your present disk on another even if you do not have 2 drives to copy with. A typical entry might be "1,2RETURN" where 1 is the source disk and 2 is the destination disk. If you do not have two drives, the DOS will issue commands of what disk to insert for writing or reading. Programs in the random access memory are destroyed when using this command and the DOS reminds you of that fact when this command is entered.

Binary Save (K). Binary save is the command that one can use to save all those machine code programs you generate if you have an assembler. The binary save, unlike most of the other commands utilized by the ATARI, uses hex numbers as opposed to decimal. I suppose if you want to save those machine code programs, you can count to sixteen using letters anyway. A typical save binary program appears like "D2:MACHINE.CDE,4FEO,6BAC RETURN". This would write a file called "MACHINE.CDE" on disk 2. The data would be saved from addresses 4FEO to 6BAC inclusive. This command also allows the append syntax by placing it immediately following the file name. An example is as follows: "D:OPCODE/A,54E2,2BC3.

Now I am going to give you a clue as to how to automatically execute your program from a binary load command. Before you become too elated, there are some pains with all neat things even in the world of ATARI. You have to poke addresses 736 and 737 with the starting address of your binary program. Address 736 is the low order byte of the starting address and 737 is the high order byte. For you machine code users, the addresses are 02E0 and 02E1. Now just append this to your program and away you go.

Binary Load (L). This is the command you use to load the previously saved binary program. There really is not too much to say about it especially if you append the starting address of your program. You just enter the file name and let the system do the rest.

Run at Address (M). Run at address is for those of us who did not have the book of how to do it. The DOS asks you run from what address and you enter the address in hex, of course. After all we are binary programmers, are we not.

Define Device (N). The preliminary manual does not recommend using this command as it is not perfected. Rumor has it that there will be a revision out soon to fix it, however. To me that is a challenge to find out what about it works and if it is useful. The intent was to essentially change the name of a device and create pseudo files and names. One example is "P:FILE" where whenever "P" is referenced, it will write to a file "FILE" of whatever you directed it to.

Duplicate File (O). Duplicate file is like the J command of duplicating the disk except you do not duplicate as much, to be exact, a file at a time. Again, if you only have one drive like some of us, you can do it the same way as the duplicate disk command.

This has been a very quick and brief description of what you can do with the DOS and how it can work for you. I am sure that when the DOS operators manual comes out, it will explain everything much better but until then maybe this will keep some of you file manipulators out there happy.

Atari Sounds Tutorial

Jerry White

This program was designed to help you discover some of the amazing sounds of Atari. You will enjoy experimenting with this program and learn at the same time. Here's how it works:

We will use two FOR NEXT loops to alter the volume and pitch variables of the SOUND command. You will be prompted to type the required data. The program will then execute using your data and you will hear the sound you created. Here is sample data for you to use to get the feel of the program. Respond to the prompts with Dist 10, Pitch 20, L1 from 15, L1 to 0, L1 step -0.5, L2 from 3, L2 to 0, L2 step -1. Notice how the sound seems to vibrate as it fades. If you want to hear it again, just hit the option key.

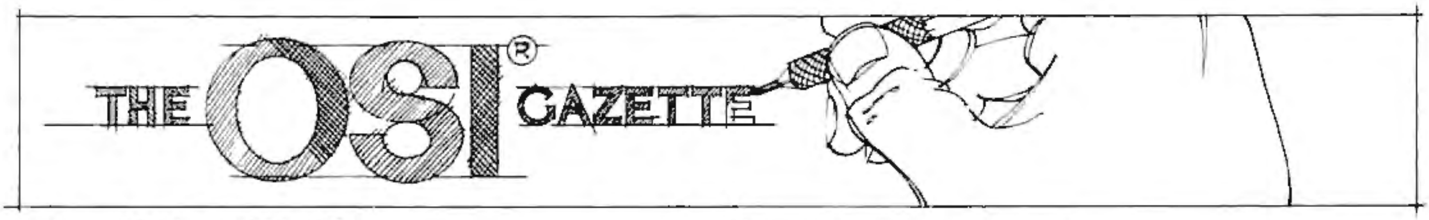
You may want to use that sound in a program you write. At this point you will notice a Basic subroutine is displayed near the top of the screen. Make note of it and any other interesting sounds you come up with. Start a library of subroutines. When you're ready to try a new sound, hit the START key.

There are a few other useful routines in this program you may want to study. Lines 12 and 14 will show you how to use random color. You will find extensive error trapping of input routines. The loop from line 410 to 440 shows how to make use of the OPTION and START keys. To see if the SELECT key has been pressed, PEEK at 53279 and see if it equals 5.

When you type in line 340, type those messages using inverse video. The routine from line 300 to line 380 will cause these messages to flash. To further dress up your display, I suggest you also use inverse video for the messages at lines 10, 130, and 6000.

After you've used and studied this program for a while, you will begin to realize the variety of possible sounds is almost endless. Now consider this. You have been using only one of the four voices available. The four voices can be used at the same time. You control the volume, pitch, and distortion of each voice. Take it away, imagination!

```
0 REM SOUNDS PROGRAM BY JERRY WHITE 8/28/80
1 GRAPHICS 0:DIM X$(1),BL$(20):BL$="                    "
2 ? :? " SOUND TEST ABBREVIATIONS ":? :? "DIST=DISTORTION VARIABLE (0 THRU 14)":
? "EVEN NUMBERS ONLY."
3 ? :? "PITCH=ANY NUMBER FROM 0 THRU 255.":? "WE WILL MOVE THE PITCH IN LOOP 2."

4 ? :? "L1=OUTER LOOP 1 VOLUME.":? "TYPE ANY NUMBER FROM 0 THRU 15":? "AT PROMPT
 FROM, TO, AND STEP."
5 ? :? "L2=INNER LOOP 2 PITCH.":? "TYPE ANY NUMBER FROM 0 THRU 255":? "AT PROMPT
 FROM, T0, AND STEP."
7 ? :? "HIT RETURN TO BEGIN";:INPUT X$
10 GRAPHICS 0:? :? "  SOUND TEST  "
12 POKE 752,1:C=RND(0)*16:REPEAT=0
14 SETCOLOR 1,C,2:SETCOLOR 2,C,8:SETCOLOR 4,C,2
30 POSITION 2,3:? "TYPE DIST    ";:TRAP 34:INPUT D:TRAP 40000
32 IF D=0 OR D=2 OR D=4 OR D=6 OR D=8 OR D=10 OR D=12 OR D=14 THEN 36
34 POSITION 2,3:? BL$:GOTO 30
36 POSITION 2,5:? "TYPE PITCH   ";:TRAP 40:INPUT P:TRAP 40000
38 IF P<255 THEN 42
40 POSITION 2,5:? BL$:GOTO 36
42 POSITION 2,7:? "TYPE L1 FROM ";:TRAP 46:INPUT F1:TRAP 40000
44 IF F1<33 THEN 48
46 POSITION 2,7:? BL$:GOTO 42
48 POSITION 2,9:? "TYPE L1 TO   ";:TRAP 52:INPUT T1:TRAP 40000
50 IF T1<33 THEN 54
52 POSITION 2,9:? BL$:GOTO 48
54 POSITION 2,11:? "TYPE L1 STEP ";:TRAP 58:INPUT S1:TRAP 40000
56 IF S1<33 THEN 60
58 POSITION 2,11:? BL$:GOTO 54
60 POSITION 2,13:? "TYPE L2 FROM ";:TRAP 64:INPUT F2:TRAP 40000
62 IF F2<256 THEN 70
64 POSITION 2,13:? BL$:GOTO 60
70 POSITION 2,15:? "TYPE L2 TO   ";:TRAP 74:INPUT T2:TRAP 40000
72 IF T2<256 THEN 80
74 POSITION 2,15:? BL$:GOTO 70
80 POSITION 2,17:? "TYPE L2 STEP ";:TRAP 84:INPUT S2:TRAP 40000
82 IF S2<256 THEN 100
84 POSITION 2,17:? BL$:GOTO 80
100 IF REPEAT>0 THEN GOSUB 5000:GOTO 400
120 GOSUB 5000:SOUND 0,0,0,0:? CHR$(125)
130 ? :? "  YOUR SOUND SUBROUTINE: "
140 ? :? "100 FOR L1=";F1;" TO ";T1;" STEP ";S1
160 ? "110 FOR L2=";F2;" TO ";T2;" STEP ";S2
180 ? "120 SOUND 0,";P;"-L2,";D;",L1"
200 ? "130 NEXT L2":? "140 NEXT L1":? "150 RETURN"
280 FOR DELAY=1 TO 500:NEXT DELAY
300 FOR TIME=1 TO 5:POSITION 2,20:? "  HIT START TO RESTART  ":? "  HIT OPTION T
O REPEAT  "
320 FOR DELAY=1 TO 10:NEXT DELAY
340 POSITION 2,20:? "  HIT START TO RESTART  ":? "  HIT OPTION TO REPEAT  "
360 FOR DELAY=1 TO 10:NEXT DELAY
380 NEXT TIME
400 SOUND 0,0,0,0
410 IF PEEK(53279)=6 THEN 10
420 IF PEEK(53279)=3 THEN 500
440 GOTO 410
500 REPEAT=REPEAT+1:GOTO 100
5000 FOR L1=F1 TO T1 STEP S1
5100 FOR L2=F2 TO T2 STEP S2
5200 TRAP 6000:SOUND 0,P-L2,D,L1:TRAP 40000
5300 NEXT L2:NEXT L1:RETURN
6000 ? :? "     INVALID SOUND, TRY AGAIN.    ":SOUND 0,0,0,0
6100 FOR DELAY=1 TO 250:NEXT DELAY
```



A 6502 Disassembler

Thomas G. Gordon

As the proud owner of an OSI Superboard II, I was immediately curious to see what made it tick. A little peeking into the Basic-in-Roms via the monitor was enough to convince me that I needed a good disassembler if I was going to get anywhere. Listing 1 is the resultant 6502 disassembler. Although written in Microsoft Basic to run on my OSI, it should run on any 6502 based system with minor modifications (8K memory needed).

Although I feel the program is fairly straight-forward, a few words may be in order to explain its operation.

Lines 5 - 30 print a greeting and ask for a starting address at which to begin disassembling.

The subroutines at 900-990 and 1000-1095 are hex to decimal and decimal to hex conversion routines, respecitvely.

the subroutine at lines 250-340 inserts the mnemonic op codes into the array R$, dimensioned by line 5. Each mnemonic contains a fourth letter which I call a tag code. The purpose of the tag code is to identify the addressing mode associated with that particular op code. For example, the tag R indicates relative more addressing.

Lines 40-75 fetch the numerical op code, print the current address in hex, determine if it is a legal op code, (if not, the operator is requested to enter another starting address) and print the hex op code along with its three letter mnemonic.

Lines 85-150 determine the addressing mode by examining the tag code, and jump to the proper routine to print any associated arguments (data or address) following the op code. These routines are located at lines 600-795.

The disassembler will continue to run until killed by the operator or an invalid op code is found.

```
LISTING 1
5 DIMR$(257)
10 PRINT"6502 DISASSEMBLER":PRINT:PRINT
15 GOSUB250
20 PRINT"ENTER START ADDRESS"
25 PRINT"IN HEX,USE 4 DIGITS."
30 INPUTA$
35 GOSUB900
40 Z=PEEK(S):A=S
55 GOSUB1000:REM-GET HX ADR
60 PRINTFR$;TH$;TW$;DE$;" ";
65 IFR$(Z)=""THENPRINT"INVALID CODE":GOTO20
70 A=Z:GOSUB1000
75 PRINTTW$;DE$;" ";LEFT$(R$(Z),3);" ";
85 U$=RIGHT$(R$(Z),1)
90 IFU$=" "THENPRINTU$:S=S+1:GOTO40
95 IFU$="N"THENPRINT"A#";:GOTO600
100 IFU$="A"THENPRINTU$:S=S+1:GOTO40
105 IFU$="Z"THENPRINT"A#";:GOTO625
110 IFU$="#"THENPRINT"#$";:GOTO625
115 IFU$="X"THENPRINT"A#";:GOTO645
120 IFU$="Y"THENPRINT"A#";:GOTO665
125 IFU$="B"THENPRINT" (";:GOTO685
130 IFU$="C"THENPRINT" (";:GOTO700
135 IFU$="U"THENPRINT"A#";:GOTO715
140 IFU$="R"THENPRINT"TO ";:GOTO765
145 IFU$="J"THENPRINT" (";:GOTO735
150 IFU$="V"THENPRINT"A#";:GOTO755
250 FORX=0TO255:READM$:R$(X)=M$:NEXT
255 DATABRK ,ORAB,,,,ORAZ,ASLZ,,PHP ,ORA#,ASLA,,,ORAN
260 DATAASLN,,BPLR,ORAC,,,,ORAU,ASLU,,CLC ,ORAY,,,,ORAX
265 DATAASLX,,JSRN,ANDB,,,BITZ,ANDZ,ROLZ,,PLP ,AND#,ROLA,,BITN
270 DATAANDU,ROLN,,BMIR,ANDC,,,,ANDU,ROLU,,SEC ,ANDY,,,
275 DATAANDX,ROLX,,RTI ,EORB,,,,EORZ,LSRZ,,PHA ,EOR#,LSRA,,JMPN
280 DATAEORN,LSRN,,BVCR,EORC,,,,EORU,LSRU,,CLI ,EORY,,,
285 DATAEORX,LSRX,,RTS ,ADCB,,,,ADCZ,RORZ,,PLA ,ADC#,RORA,,JMPJ
290 DATAADCN,RORN,,BVSR,ADCC,,,,ADCU,RORU,,SEI ,ADCY,,,,ADCX,,,
295 DATASTAB,,,STYZ,STAZ,STXZ,,DEY ,,TXA ,,STYN,STAN,STXN,,BCCR
300 DATASTAC,,,STYU,STAU,STXV,,TYA ,STAY,TXS ,,,STAX,,,LDY#
305 DATALDAB,LDX#,,LDYZ,LDAZ,LDXZ,,TAY ,LDA#,TAX ,,LDYN,LDAN,LDXN
310 DATA,BCSR,LDAC,,,LDYU,LDAU,LDXV,,CLV ,LDAY,TSX ,,LDYX,LDAX
315 DATALDXY,,CPY#,CMPB,,,CPYZ,CMPZ,DECZ,,INY ,CMP#,DEX ,
320 DATACPYN,CMPN,DECN,,BNER,CMPC,,,,CMPU,DECU,,CLD ,CMPY,,,
325 DATACMPX,DECX,,CPY#,SBCB,,,CPXZ,SBCZ,INCZ,,INX ,SBC#,NOP ,
330 DATACPXN,SBCN,INCN,,BEQR,SBCC,,,
335 DATASBCU,INCU,,SED ,SBCY,,,,SBCX,INCX,
340 RETURN
600 A=PEEK(S+2):GOSUB1000
605 PRINTTW$;DE$;
610 A=PEEK(S+1):GOSUB1000
615 PRINTTW$;DE$
620 S=S+3:GOTO40
625 A=PEEK(S+1):GOSUB1000
630 PRINTTW$;DE$
632 S=S+2:GOTO40
645 A=PEEK(S+2):GOSUB1000
650 PRINTTW$;DE$;
655 A=PEEK(S+1):GOSUB1000
660 PRINTTW$;DE$;",X":S=S+3:GOTO40
665 A=PEEK(S+2):GOSUB1000
670 PRINTTW$;DE$;
675 A=PEEK(S+1):GOSUB1000
680 PRINTTW$;DE$;",Y":S=S+3:GOTO40
```

The format of the resultant printout pretty much follows standard assembler notation, with one exception. When relative addressing mode is encountered, the program prints the hex address to which the branch occurs, rather than the hex offset. I found this to be much more convenient when disassembling. Since it is a one pass only disassembler, the use of labels was out, but this works just as well in my opinion.

Finally, listing 2 shows the resultant printout of some of the OSI code beginning at hex FD00, which is the start of the keyboard monitor routine.

```
685 A=PEEK(S+1):GOSUB1000
690 PRINTTW$;OE$;",X)":GOTO632
700 A=PEEK(S+1):GOSUB1000
705 PRINTTW$;OE$;"),Y":GOTO632
715 A=PEEK(S+1):GOSUB1000
720 PRINTTW$;OE$;",X":GOTO632
735 A=PEEK(S+2):GOSUB1000
740 PRINTTW$;OE$;
745 A=PEEK(S+1):GOSUB1000
750 PRINTTW$;OE$;")":S=S+3:GOTO40
755 A=PEEK(S+1):GOSUB1000
760 PRINTTW$;OE$;"Y":GOTO632
765 A=PEEK(S+1):IFA<128THEN790
770 A=255-A
775 A=S+1-A:GOSUB1000
780 PRINTFR$;TH$;TW$;OE$:GOTO632
790 A=S+A+2:GOSUB1000
795 GOTO780
900 B$=LEFT$(A$,1):C$=MID$(A$,2,1):D$=MID$(A$,3,1)
915 E$=MID$(A$,4,1):F$=B$
925 FORX=1TO4
930 IFF$="A"THENA=10:GOTO965
935 IFF$="B"THENA=11:GOTO965
940 IFF$="C"THENA=12:GOTO965
945 IFF$="D"THENA=13:GOTO965
950 IFF$="E"THENA=14:GOTO965
955 IFF$="F"THENA=15:GOTO965
960 A=VAL(F$)
965 IFX=1THENS=A*4096:F$=C$
970 IFX=2THENS=S+A*256:F$=D$
975 IFX=3THENS=S+A*16:F$=E$
980 IFX=4THENS=S+A
985 NEXTX
990 RETURN
1000 F=INT(A/4096):REM-D TO H CONVERT
1005 R=A-F*4096
1010 TH=INT(R/256)
1015 R=R-TH*256
1020 TW=INT(R/16)
1025 OE=R-TW*16:H=F
1030 FORX=1TO4
1035 IFH=10THENF$="A":GOTO1070
1040 IFH=11THENF$="B":GOTO1070
1045 IFH=12THENF$="C":GOTO1070
1050 IFH=13THENF$="D":GOTO1070
1055 IFH=14THENF$="E":GOTO1070
1060 IFH=15THENF$="F":GOTO1070
1065 F$=STR$(H)
1070 IFX=1THENFR$=F$:H=TH
1075 IFX=2THENTH$=F$:H=TW
1080 IFX=3THENTW$=F$:H=OE
1085 IFX=4THENOE$=F$
1090 NEXTX
1095 RETURN
```

A last minute correction from the author: Line 1065 works as is for APPLE; for OSI it should read 1065 F$ = MID$(STR$(H),2)

```
LISTING 2
RUN
6502 DISASSEMBLER
FNTER START ADDRESS
IN HEX,USF 4 DIGITS.
? FD00
FD00 8A TXA
FD01 48 PHA
FD02 98 TYA
FD03 48 PHA
FD04 A9 LDA #$01
FD06 20 JSR A#FCBE
FD09 20 JSR A#FCC6
FD0C D0 BNE TO FD13
FD0E 0A ASL A
FD0F D0 BNE TO FD06
FD11 F0 BEQ TO FD66
FD13 4A LSR A
FD14 90 BCC TO FD1F
FD16 2A ROL A
FD17 E0 CPX #$21
FD19 D0 BNE TO FD0E
FD1B A9 LDA #$1B
FD1D D0 BNE TO FD40
FD1F 20 JSR A#FDC8
FD22 98 TYA
FD23 8D STA A#0213
FD26 0A ASL A
FD27 0A ASL A
```

©

The First Book Of Ohio Scientific Vol I

By J. Clothier and W. Adams
Elcomp Pubishing Company
3873-L Schaifer Avenue
Chino, CA, 91710
188 Pages, $7.95

Review by Charles L. Stanford

Considering the very serious lack of documentation available for the OSI line of microcomputers, it's almost impossible for me to conceive of writing a negative review of any book with the title "The First Book of Ohio Scientific". But that's what you're going to get! This is an attractive soft cover manual, and on the surface appears to be well laid-out and chock full of goodies. Until you realize that several of the topics mentioned in the credits (such as Aardvark's Joystick Instructions) aren't included in the text, and that what *is* there is a hodgepodge of old magazine articles, sales brochures from various software and hardware houses, and a bunch of poorly organized ROM listings.

To put my viewpoint into context, I am running a very bare-bones C1P with 8K of RAM. Nothing fancy. But I'm into hardware, and have delved pretty thoroughly into the physical innards of the machine, as well as into the BASIC and Monitor ROMs. Maybe the more sophisticated owners, with C4P's, disks, printers, and A/D ports will gain more from this book. I sincerely doubt it.

Several sections may be of use to disk users, including the one on copying diskettes on single drive systems (curiously not listed in the index - it's on page 133). There are also well-written instructions on adding the RS-232C components to the C1P, and on converting to the 600 Baud cassette and printer capability. But I've seen these elsewhere in much the same form.

I think the crowning blow to me was finding that the article entitled "High Resolution Display Conversion for Challenger 1P" is merely a sales pitch for the instructions, at $12.00, from an outfit named Silver Spur. Their address, by some remarkable coincidence, is the same as the publisher's. You can also get a crystal from them for $6.95 more.

There are a number of so-called memory maps included. But they are seemingly scattered at random, and are annotated in an almost incomprehensible style. The various Monitor ROM listings are not annotated at all, which seriously compromises their usefulness.

And finally - why on earth would the "authors" include almost fifty pages of material directly copied from OSI sales brochures and owners manuals? And many more pages from previously published articles and from Aardvark's catalog? And why is there absolutely no organization by subject, computer model, etc.? Quite frankly, I can't conceive of any OSI owner getting his monies' worth from this book. I hardly await Volumes II and III with anticipation.

Editor's Note: This book should not be confused with the book sold by Aardvark Technical Services of similar name: "The (Real) First Book of OSI". RCL

©

Part 1
A Small Operating System: OS65D The Kernel

T R Berger

You switch on your computer, insert a disk, press the RESET button, then press "D". After a second or two of whirring and clicking, a menu flashes on the screen asking what you wish to do. At the time of turn-on, your computer knew nothing. Now BASIC LANGUAGE is in control. How did this happen? How does the machine get data and programs to and from the disk? How does the computer know if you are using a video monitor with a polled keyboard or an expensive serial monitor terminal? How does the computer decide to send its messages to the printer, the monitor, or to memory? How does the computer find its way among BASIC, the Assembler, and the Extended Monitor? The glib answer to all these questions is that the disk operating system makes all decisions and performs all control operations.

It is the task of the operating system to:

1. Start the computer on RESET (BREAK);
2. Manage and control all external input and output devices including keyboards, monitors, printers, and so on;
3. Manage the functioning of the disk (the single most important function of an operating system);
4. Manage loading and execution of system software in the software segment of memory, including BASIC, Assemblers, etc.; and
5. In general, keep tidy control over all transfers between these various functions.

The diagram in FIGURE 1 illustrates this mediating function of an operating system.

I hope, in several articles, to describe some of the general features of a small operating system by describing in some detail how the Ohio Scientific OS65D disk operating system functions. OS65D is a minimal function, small sized operating system. Therefore, mere mortals can comprehend its structure. I hope to convey not only a general understanding of this system, but also to provide you with some nuts and bolts to use in your own programming efforts. This includes memory maps of all subroutines.

The OS65D operating system is divided into several parts.

1. Cold Start ROM

This program takes about 256 bytes (one page) of ROM and accomplishes an absolute minimum of functions. Its major role is to load Track 0 of the disk into memory and start it running.

2. The Preparation Program

This program is in memory only long enough to do its job. All the various bits and pieces hanging on your microprocessor wake up either turned off or in some random state. Some of these need preparation before they will function properly. For example, a serial input/output port (such as used by a terminal) operates through an ACIA (Asynchronous Communication Interface Adapter....who thought up that mouthful?) which must be prepared for proper functioning. This program carries out these preparations.

3. The Operating System Kernel

This program is the 'BOSS'. It contains all the commands by which you may direct the operating system. It directs the functioning of the remaining parts of the operating system.

4. Disk Routines

These are the programs which make the disk the magic storage medium which it is. These routines start and stop the clicking and whirring you hear when the disk operates.

5. The Input and Output Routines

Input may come from a keyboard, a serial terminal, or any of many sources. Similarly, output may go to a serial terminal, a video monitor, a printer, or some other device. Each such device needs its own input or output program. Further, there must be one supervisory program which can choose from as many or as few of these input/output devices as are desired at any one time. These programs constitute the Input and Output Routines.

6. Utility Programs

Certain programs are needed only occasionally. These include disk copying programs, and Track 0 modification programs. These utility routines are only loaded into memory when needed. They hide in sectors placed after the major system software on various tracks of the disk.

In this article we will explore the 'BOSS', i.e. the Operating System Kernel. The obvious part of the kernel is the set of commands. Not so obvious, but very useful, are the line input routine, the line buffer reader, and various Hex to ASCII and ASCII to Hex conversion and other routines. Let's go through the various commands availalbe in OS65D and see how they function.

OSI

SOFTWARE FOR OSI

OSI

A JOURNAL FOR OSI USERS!!

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

1) Using String Variables.

2) High Speed Basic On An OSI.

3) Hooking a Cheap Printer To An OSI.

4) An OSI Disk Primer.

5) A Word Processor For Disk Or Tape Machines.

6) Moving The Disk Directory Off Track 12.

Four back issues already available!
$9.00 per year (6 issues)

ADVENTURES

Adventures are interactive fantasies where you give the computer plain English commands (i.e. take the sword, look at the control panel.) as you explore alien cities, space ships, ancient pyramids and sunken subs. Average playing time is 30 to 40 hours in several sessions. There is literally nothing else like them — except being there yourself. We have six adventures available.

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

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

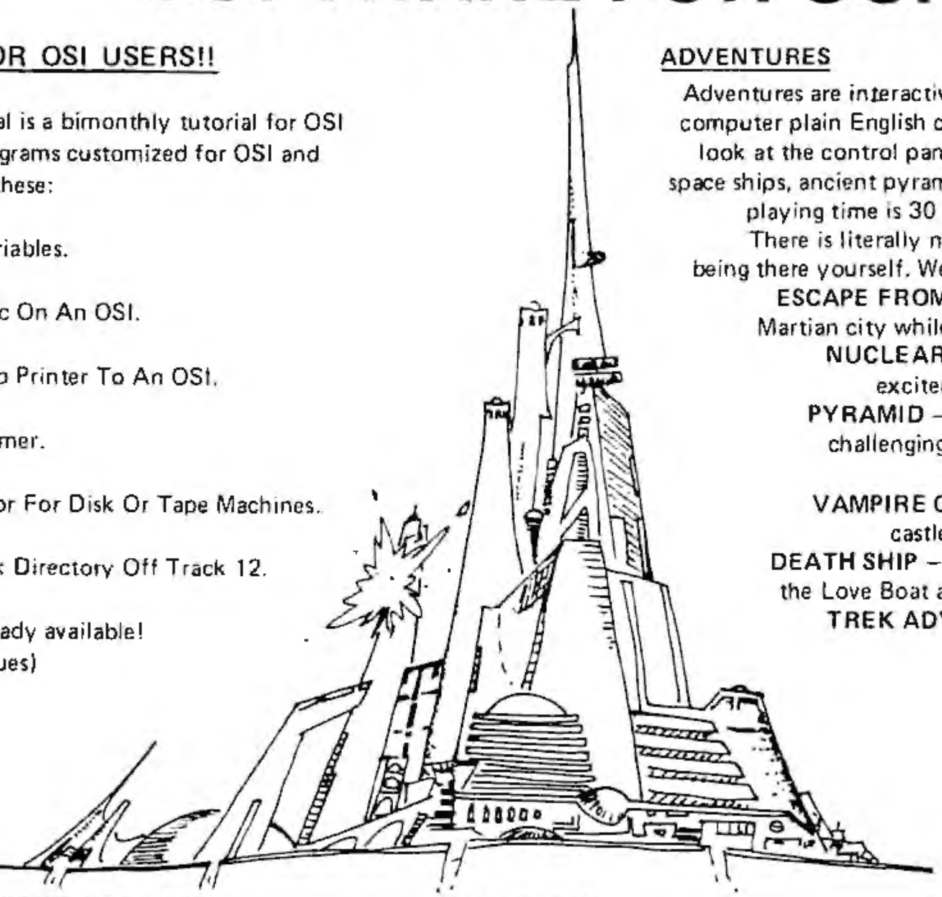
PYRAMID — Our most advanced and most challenging adventure. Takes place in our own special ancient pyramid.

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

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

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

$14.95 each



NEW SUPPORT ROMS FOR BASIC IN ROM MACHINES

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

C1E/C2E for C1/C2/C4/C8 Basic in ROM machines.
This ROM adds full screen editing, software selectable scroll windows, keyboard correction (software selectable), and contains both an extended machine code monitor and a fix for the string handling bug in OSI Basic!! It has breakpoint utilities, machine code load and save, block memory move and hex dump utilities. A must for the machine code programmer replaces OSI support ROM. Specify system! $59.95

STRING BUG FIX (replaces basic ROM chip number 3)
All this chip does is to replace the third basic ROM and correct the errors that were put into the ROM mask. $19.95

DATA SHEETS

OS65D LISTING
Commented with source code, 83 pages. $24.95

THE (REAL) FIRST BOOK OF OSI
65 packed pages on how OSI basic works. Our best selling data sheet. $15.95

OSI BASIC IN ROM
Ed Carlson's book of how to program in basic. Now available from Aardvark. $8.95

P.C. BOARDS

MEMORY BOARDS!! — for the C1P. — and they contain parallel ports!
Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

REAL SOUND FOR THE C1P — and it's cheap! This bare board uses the TI sound chip to give real arcade type sound. The board goes together in a couple of hours with about $20.00 in parts. Bare board, plans, and sample program — $15.95

ARCADE AND VIDEO GAMES

ALIEN INVADERS with machine code moves — for fast action. This is our best Invaders yet. The disk version is so fast that we had to add selectable speeds to make it playable.
Tape - $10.95 — Disk - $12.95

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

STARFIGHTER — a real time space war where you face cruisers, battleships and fighters using a variety of weapons. Your screen contains working instrumentation and a real time display of the alien ships. $6.95 in black and white - $7.95 in color and sound.

SEAWOLFE — this one looks like it just stepped out of the arcades. It features multiple torpedoes, several target ships, floating mines and real time time-to-go and score displays. — $6.95 in black and white $7.95 in color and sound.

SCREEN EDITORS

These programs all allow the editing of basic lines. All assume that you are using the standard OSI video display and polled keyboard.

C1P CURSOR CONTROL — A program that uses no RAM normally available to the system. (We hid it in unused space on page 2). It provides real backspace, insert, delete and replace functions and an optional instant screen clear.
$11.95

C2/4 CURSOR. This one uses 366 BYTES of RAM to provide a full screen editor. Edit and change lines on any part of the screen. (Basic in ROM systems only.)

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

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

DISK UTILITIES

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

DISK CATALOGER
This utility reads the directory of your disks and makes up an alphabetic list off all your programs and what disks they are on. $14.95

MACHINE CODE RENUMBERER
(C2/4-MF only)
Renumbers all or part of a program at machine code speeds — $15.95

VISA

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

master charge

OSI

Aardvark Technical Services • 1690 Bolton • Walled Lake, MI 48088
(313) 669-3110 or (313) 624-6316

OSI

OS65D Kernel Command Descriptions

The kernel has 18 user commands. These may be divided into four categories as follows: (1) Commands which move data or programs from the disk to memory; (2) Commands which reverse this process and move data or programs from memory to the disk; (3) Commands used for disk diagnostics and preparation; and (4) Other commands. With this division in mind, let's discuss the function of each command by category.

Transfers from the disk to memory

The 6 commands (BA, AS, EM, XQ, LO, CA) in this category can be subdivided into four which load and run: BA, AS, EM, XQ, and two which just load: LO, CA. The ones which load and run have their vital statistics listed in Table 1 where the track numbers are for 8'' diskettes. After the tracks have been loaded to memory, program control is transferred to the location listed under 'jump' in the table. The commands BA (BASIC), AS (Assembler), and EM (Extended Monitor) are self-explanatory. They load and run languages and systems supplied with your computer. A general command much like BA, AS, and EM is XQ (EXECUTE). If you develop machine language systems to run at $317E then XQ NAME or XQ TRACK will load and run these sytems where NAME is the name of your file and TRACK is the first track number of the file. These commands load integral numbers of tracks, and thus will not load sectors from within tracks. They offer great ease of operation but practically no versatility.

To add versatility, we need two more general commands (LO, CA). We need a command LO (LOAD) to accomplish what the previous commands do for whole tracks without adding the 'run' feature at the end. This one additional command is not enough. Each track stores 3K bytes of data. It is rather inefficient to store a 200 byte program on one full track. Therefore, the operating system allows us to divide each track into sectors. We are still limited by the fact that a sector must be an integral multiple of pages (1/4 K or 256 bytes) up to 3K. However, it is less wasteful to store 200 bytes in a 256 byte sector than to store it in a 3K track. The CA (CALL) command allows this sector type of operation.

For full tracks and for sectors we have two load commands listed in Table 1. First, LO NAME or LO TRACK loads a file named NAME or a file beginning at track number TRACK to memory. Second, CA MEMORY = TRACK, SECTOR calls sector number SECTOR on track number TRACK to memory, starting the load at memory address MEMORY. Note that LO specifies no starting address. Further, 'LO NAME' specifies no track number for the disk. When a file is named, a track number is found in the disk directory which resides in Track 8. The load vector (memory start address) is usually $3179 for the LO command. Since BASIC disk buffers are kept between $317E and the start of your program, this means that any BASIC program with a buffer will use disk space to preserve buffer space. Disk space is wasted, but the operating system remains very simple. Sectors could also be named in a directory with load vectors written into the first few bytes, but that would enlarge the memory requirements of the operating system and add to its complexity. The authors of OS65D chose to forgo enhancements. Thus the CA command requires all of the load data except the length of the sector, which is stored among the first few bytes of the sector.

The six commands just described (BA, AS, EM, XQ, LO, CA) provide a small, yet very powerful set for obtaining files from the disk. For simplicity and compactness of the system, the user is asked to suffer a little inconvenience in loading sectors. Further, since most data and program files will reside in named files, some disk inefficiency is accepted as the price of a compact operating system. In particular, no matter how long or how short a BASIC program is, it will always be stored on an integral number of tracks. A 1K program will use a 3K Track (or more if there are buffers). To change this would require more elaborate programming of sectors and directories. Under such a more elaborate system the disk would appear to be much larger. On the other hand, because more elaborate programming is necessary, the disk would run more slowly. However, compared to cassette tape, even these more elaborate programs would seem jet propelled.

Transfers from memory to the disk

There are no commands for saving memory which might be analogous with a 'load and run' command. Thus the operating system need only have commands which perform functions opposite to LO and CA. These are PU (PUT) and SA (SAVE) and are also given in Table 1. In analogy with LO, PU NAME or PU TRACK will put memory onto an integral number of disk tracks. If the file is named NAME, the directory will specify the starting track and how many tracks are available. The transfer always starts at memory location $3179 and will save T tracks (about T X 3K of memory) where T is given in $317D.

Similarly, SA TRACK, SECTOR = MEMORY/PAGE will SAVE memory beginning at memory address MEMORY and continuing for PAGE number of pages on track number TRACK in sector number SECTOR. The number of pages in a sector is saved on the disk, but is usually not stored in memory. Therefore, when saving memory, the length of the segment to be saved must be given in the command. The symbols ',', ' = ', and '/' used in the SA command serve only to separate addresses and numbers and to complicate your life. They all can be changed easily to ',' or spaces. The disadvantage of making such a change is that the order of the numbers is vital. Presumably ',', ' = ', followed by '/' help you keep the numbers in the right order. If you don't, you get an error message rather than a disastrous SAVE which might overwrite some of your more beautiful programming efforts.

Commands used for disk diagnostics

Being mechanical devices, disks are not perfect. Occasionally you need to manipulate the disk or examine the entire contents of a given track. Further, you need to copy old and initialize new disks. There are commands for doing these things in the operating system kernel. We may divide these commands into 3 sets: (a) Reading from the disk, (b) Writing on the disk, and (c) Manipulating the disk.

These commands are listed in Table 1. First come the diagnostic read commands EX (EXAMINE) and DI (DIRECTORY). The command EX MEMORY = TRACK reads everything for examination from track number TRACK to memory beginning at address MEMORY. If you are encountering disk trouble or suspect a bad diskette, this is a very useful command. If you have inadvertently erased or overwritten part of a disk, this command may help salvage some of the remaining programs. If you are just trying to learn how your disk stores memory, this is a helpful command.

On the other hand, if you just wish to learn the status of a particular track (i.e. how many and how long are its sectors) then using EX can prove to be very tedious.

The command DI TRACK will print out a sector number and length directory (in pages) for track number TRACK. The disk directory tells us that OS65D occupies Tracks 0 - 8, but does not give us information as to how many sectors reside in, say Track 8. On the other hand, DI 08 tells us there are 4 sectors of length 1 page each on Track 8. Unfortunately, OS65D does not allow us to name individual sectors within a track. We can, however, name the track in which these sectors reside by using the BASIC 'CREATE' program.

There is one diagnostic command IN (INITIALIZE) for writing on the disk. It allows us to initialize a whole disk by IN or an individual track number TRACK by the command IN TRACK. When a track is initialized, the beginning of the track is found and track identification data are placed on the disk. Then the rest of the track is completely erased. No sector identification marks are placed on the disk so the track is not useable by LO or PU as it stands. The BASIC CREATE program will fix this problem.

Finally, there are three diagnostic disk manipulation commands (HO, SE, D9). The disk changes tracks by stepping the read head outward toward Track 0 or inward toward Track 76 one track at a time. The head moves when the stepper motor spins a fixed fraction of a revolution. This process is not perfect and occasionally the head will be misplaced on the disk. There is only one track (Track 0) where there is a sensor to detect whether or not the head is correctly positioned over the track. All other tracks are found by counting steps inward from Track 0, or counting up or down from the present track number. The present track number is saved in memory ($265D). Usually when the head is misplaced, it is only very slightly off the circular data stream on the disk. You may have noted this phenomenon in another context with music filled cassette tapes. A friend loans you his great sounding 'BOOMBAH' cassette which he made live. It sounds great on his HIFI but lousy on yours. The reason is that his recorder put the music track onto the tape in a position differing slightly from the place where your recorder is trying to find it.

If the disk head is slightly out of position on the disk, the same thing occurs, i.e. a lousy read. Your disk will detect this and step the head down one track then back to try again. Even though this process occurs very quickly, it is imperfect at best. Memory tells where the head is supposed to be. But in many jumps back and forth between tracks, 'supposed to be' an 'really is' could differ. If after a few tries at repositioning the head, the disk still fails to find the track, it quits and sends an error message. The solution to this problem is to start all over again. Move the head to Track 0 where it can mechanically sense its position then start up again. The HO (HOME) command does this by homing the read head to Track 0.

If you have run a BASIC program which requires a disk read midway and have been thrown out of your program with the Error #5 then you know how annoying this can be. The cure is to find the step in the BASIC program where the disk read occurs. Just preceding this step, insert a step with DISK!"HO". This instruction assures you that if the track requested in the next step can be found, it will be found without error. A more elaborate operating system would incorporate such a step in its track seeking logic (i.e. if the head fails to find the track after several tries, it would go to Track 0 and start over).

If you own more than one disk drive, (lucky you!) you may select any one by the command SE (SELECT) via SE DRIVE where DRIVE is A,B,C, or D (OS65D can control up to 4 drives). When you select a drive it is automatically homed and thus starts out aligned at Track 0.

Older versions of OS65D did not properly find the disk index hole at the beginning of a track. Newer versions do not have this problem, and go further to incorporate an error if the beginning of a track cannot be found quickly (i.e. within one revolution of the disk). Since older disks may take several revolutions before data synchronization takes place, OS65D will refuse to read these disks. Command D9 (DELETE 9) is supposed to eliminate this condition by short circuiting the new error. Even though the D9 subroutine is included in my version of OS65D, it is not connected. If I enter command D9, my command table sends the computer to the 'syntax error' subroutine instead of the D9 subroutine. This can be corrected by putting the D9 subroutine address (minus one) into the command table in place of the 'syntax error' address. I own no old OS65D disks, so I have not changed anything.

At this point, it might be worth alluding to diagnostic features of OS65D not in the kernel. Ohio Scientific was mortally afraid you might damage the vital kernel information on Track 0. Thus the kernel mightily protects Track 0 against your invasions.

If you happen to load a program into memory, to save it back onto the disk, and in the middle of the save, to change your mind and quickly to remove the diskette from the drive, then you will certainly cause an erasure somewhere on the diskette. This procedure (which you should avoid) places a very strong, rapidly varying magnetic field at an undetermined place on the diskette. Rapidly varying magnetic fields erase diskettes. If the undetermined place is in Track 0, part of Track 0 is lost. Therefore, every single part of the diskette must be changeable by the computer user, including Track 0. OS65D has a Track 0 read/write utility to accomplish this.

Most people have only one disk drive. In order to copy a disk one moves programs from an old diskette to memory and from there onto the new diskette. It's tedious, but it works. It would be very helpful to one-drive owners if you wrote a machine language program to simplify this process as much as possible.

There are others (with spare money) who have two or more disk drives. Two drives have the advantage that it is easy to copy from an old diskette in one drive to a new diskette in another drive, if you have a program. OS65D also contains a disk copying utility.

The copier and Track 0 utility programs are available in Sector 2 of Track 1 on the disk. In order to further protect you from the ways of error, and to save memory, these programs are not normally in memory. They are not part of the kernel. Thus we will discuss them in another article of this series. However, these programs are available for diagnostic purposes. They can be loaded by CA 0200 = 01,2. They can be run by GO 0200. I advise you to know what this program does and how it works before you try it. (Either wait for me or read your manual carefully.)

Other commands

There are 4 additional commands (RE, GO, IO, ME) in the kernel not associated with the disk:

The first of these is the restart command RE (RESTART). If you have just entered a BASIC program from the keyboard and wish to know how many tracks it will occupy on the disk, you type EXIT. This puts you in the command mode of the operating system kernel. If you typed BA (BASIC) to return to BASIC, a minor disaster would occur. BASIC would be loaded from the disk and the source file initialized. In simple terms, your program would be gone. (It really is salvageable, but that is a complicated process.) To avoid this problem we have a restart command. To restart BASIC, the command is RE B.

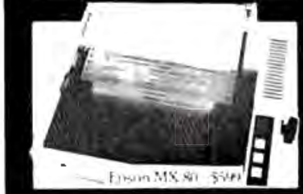
When BASIC is in memory, the Assembler and Extended Monitor are not. If you try to restart the Extended Monitor with RE E when BASIC is loaded, you receive a syntax error message. Using the RE command you may restart BASIC (RE B), the Assembler (RE A), the Extended Monitor (RE E), or the ROM Monitor (RE M) if they are in memory.

At this point it is worth discussing a rather subtle matter. Anytime you are someplace else in memory and able to GO at an arbitrary address, then you may restart OS65D by starting at $2A51. However, if you have used the keyboard without using the keyboard I/O routine in OS65D, you will have crashed BASIC or the Assembler, whichever is in memory. The reason is that the keyboard polling routine was written for ROM BASIC machines and as such uses storage locations $0213-$0216. Unfortunately, these locations are vital to BASIC and the Assembler. Thus, the I/O routine in OS65D swaps these locations out before going to the keyboard polling routine in ROM. After completing the keyboard poll, these locations are swapped back in again.

Intertec Superbrain
32K Ram $2449
64K Ram $2649

NEC spinwriter
5510 5530 $2449

Apple II 16K $ 959
48K $1099


Epson MX-80 $599

Atari 800 $749

Diablo 630 $2295
(with tractor feed)

When you use RE M, these locations are swapped out since the ROM monitor uses the ROM keyboard polling routine. To swap these locations back in again you do not type $2A51G from the ROM monitor. Instead, you use a routine in the I/O section of OS65D which first swaps the keyboard back again and then goes to $2A51. So from the ROM Monitor, you restart OS65D by $2547G.

Through its various programs, the computer transfers control from one program to another. For example, RE B causes the computer to leave the kernel at the address $2C0D and enter BASIC at its WARM START location $20C4. If you have written your own machine programs, you may start them from the ROM monitor, the Extended Monitor, or the Operating System Kernel. To start a program from the kernel at address $4C00, the command is GO 4C00.

The final two OS65D kernel commands (IO, ME) control input to and output from the computer in a very simple way. One byte of memory consists of eight bits; each bit is either a 0 or a 1. One byte of memory is allocated as an input flag ($2321) and one as an output flag ($2322). Each of the eight bits represents an input (or output) device. If a particular device bit is 1, then that device is connected; if it is 0, that device is disconnected. We may imagine the bits arranged in a row as follows:

7	6	5	4	3	2	1	0
a	b	c	d	e	f	g	h

The bit itself is denoted by a letter in a box, and the number above is its position. The positions 0-7 stand for devices. These are given in Table 2. You may not recognize some of the devices because they are not part of your computer. However, if you so choose, you may buy these devices from OSI.

If bit 1 is 1 (g = 1) and all other bits are 0 in the input flag ($2322) and then input is taken from device 1, the keyboard. If bits 1 and 3 are 1 (e = 1 and g = 1) and all others are 0 in the output flag ($2321) then output is sent to the video monitor and the parallel printer. We may change the bits in the IO (INPUT/OUTPUT) flags ($2321 and $2322) via the INPUT/OUTPUT command IO INPUT,OUTPUT where INPUT and OUTPUT are the hexadecimal versions of the bits in the boxes. (IO, OUTPUT changes just the output flag and IO INPUT just the input flag.)

There is one intriguing device (bit 4 for both input and output) called MEMORY. How can memory be an input or output device? (Actually, memory is a storage device, just as cassette tape or a disk is. Thus we can put stuff into it and take it back out. As long as we do not erase memory, it will remain there. Usually material is put into and taken out of memory under program control. There may be circumstances where we do not want memory under program control. For example, suppose you have a long BASIC program that works on a large amount of text stored as strings (such as a justification program for a text editor). Assume the final text is to be sent out via a MODEM to a distant printer. Your justifier will chomp away producing and sending a string every now and then wasting a great deal of telephone time. A better approach would be to temporarily justify into memory, then send the resulting text. A computer has no idea where it gets its input or sends its output except via a subroutine. It does not care if it sends to the video monitor, the disk, a telephone, memory, or the moon.

The memory input/output capability is also used by the Indirect File. This program resides in the Input/Output section of OS65D and will be discussed in another article. One of the many uses of the Indirect File is to append many short BASIC programs end to end to make one long one.

To make use of memory as an input/output device via the command ME (MEMORY) we must know which part of memory to address. ME INPUT,OUTPUT sets the start address of the input to INPUT and the start address of the output to OUTPUT.

Hopefully, these descriptions of the OS65D commands, in conjunction with your OS65D USER's GUIDE will help you to make better use of the commands in your operating system. BASIC can execute any operating system command via DISK! "any OS65D command string". For example, if we have the following program lines then a program allows the user to select disk drive A or B.

```
100 INPUT "WHICH DRIVE (A/B)";A$
110 IF A$<>"A" AND A$<>"B" THEN 100
120 DISK!"SE" + A$
```

The Extended Monitor and Assembler can also send operating system commands via any OS65D command string.

TABLE 1

COM- MAND	NAME	TRACKS	ADDRESS	JUMP
AS	ASSEMBLER	5 - 6	0200-1700	1300
BA	BASIC	2 - 4	0200-22FF	20E4
EM	EXTENDED MONITOR	7	1700-1FFF	1700
XQ	EXECUTE	USER	3179-	317E
CA	CALL	USER	USER	-
LO	LOAD	USER	3179-	-
PU	PUT	USER	3179-	-
SA	SAVE	USER	3179-	-
		FUNCTION		
DI	DIRECTORY	SECTOR DIRECTORY		
D9	DISK #9	DISABLE ERROR #9		
EX	EXAMINE	EXAMINES A FULL TRACK		
HO	HOME	HOME THE DISK TO TRACK 0		
IN	INITIALIZE	INITIALIZE A DISK		
SE	SELECT	SELECT A DRIVE		
GO	GOTO	EXECUTE A MACHINE PROGRAM		
IO	INPUT/OUTPUT	SET INOUT/OUTPUT FLAGS		
ME	MEMORY	SET MEMORY IO VECTORS		
RE A	RESTART ASSEMBLER			
RE B	RESTART BASIC			
RE E	RESTART EXTENDED MONITOR			
RE M	RESTART ROM MONITOR			

TABLE 2
INPUT/OUTPUT

BIT NUMBER	INPUT FLAG
0	SERIAL INPUT (ACIA)
1	POLLED KEYBOARD
2	CASSETTE INPUT ON 430 BOARD
3	NULL (0) INPUT
4	MEMORY INPUT
5	DISK BUFFER #1 INPUT
6	DISK BUFFER #2 INPUT
7	SERIAL INPUTS FROM 550 BOARD
BIT NUMBER	**OUTPUT FLAG**
0	SERIAL OUTPUT (ACIA)
2	VIDEO MONITOR
3	LINE PRINTER
4	MEMORY OUTPUT
5	DISK BUFFER #1 OUTPUT
6	DISK BUFFER #2 OUTPUT
7	SERIAL OUTPUTS FROM 550 BOARD

Next time: Subroutine descriptions.

OSI C1P Fast Screen Clears Revisited

Charles L. Stanford

Since writing the article on Screen Clear Routines for the OSI C1P for Compute II, Issue 1, I've been particularly sensitive to variations on machine language programming methods which could be used to improve the use of the computer. Several publications have been of considerable help, especially Compute and Compute II, Micro, the Aardvark and Progressive Computing Catalogs, and of course Edward Carlson's fine book on OSI BASIC. Mr. Carlson recently published an article which has led, indirectly, to a way of tapping into the Monitor and BASIC routines which input from the keyboard and write to the screen, ACIA, etc. Certainly, these techniques are well known to the more advanced C1P owners. Unfortunately, these people, with few exceptions, aren't writing for publication. So most information is being passed (slowly) by word of mouth or by club newsletters.

There are at least four points at which you can "break into" routines which are actively treating inputs or outputs. These are the Subroutines at $00BC and $0207, and the Jump vectors at $0218 and $021A. I'm sure there are more there for the finding. For this article, the Input vector at $0218 will be used.

Normally, this location holds a Jump Indirect to the routines starting at $FFBA in the monitor ROM which input a character from the keyboard or cassette. But it's no trick to poke a new address into this location, then do a little modifying of the routine. In this case, as shown in the listings, we are changing the vector from $FFBA to $00D8. This is near the end of zero page, which is not used by BASIC. Note, however, that it is used by the Monitor, so a Break to the Monitor followed by a Warm Start will require that the vector be reset and that the program be reentered.

The program is short and simple in operation. Essentially, it Goes sub to FFBA, which inputs a character. Next, the character is tested, and if it is a $7F, the RUBOUT key code, one of the more efficient machine language screen clear routines is effected. If it is any other character, this is skipped, and the program goes on about its business.

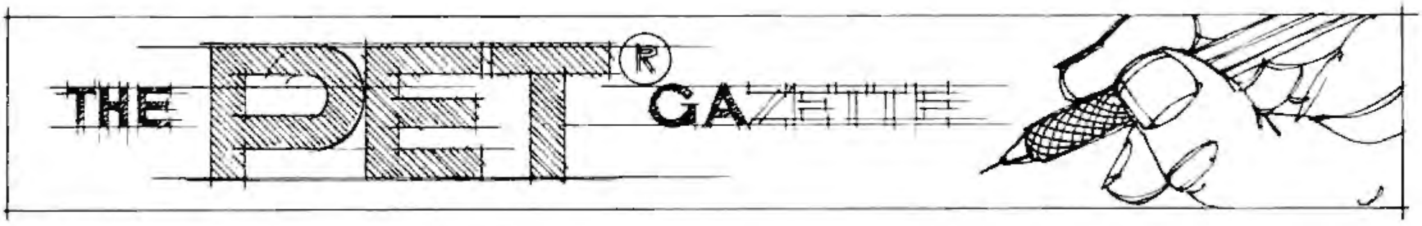
Note also that line 2010 in Listing II also POKEs the vector into location $0B, the USR vector. Thus, you will have both a single key screen clear by pressing the rubout and a programmable one by calling X = USR(X).

LIST 1

```
00D8 20 BA FF  JSR $FFBA   GET A CHARACTER
00DB C9 7F     CMP #$7F    IS IT A RUBOUT?
00DD D0 15     BNE $00F4   IF NO SKIP TO END
00DF 48        PHP         SAVE THE CHAR
00E0 A0 00     LDY #$00
00E2 A9 20     LDA #$20$   BLANK CHAR
00E4 99 00 D3  STA-Y       STORE BLANK AT 256
00E7 99 00 D2  STA-Y       LOCATIONS IN FOUR
00EA 99 00 D1  STA-Y       PAGES OF VIDEO RAM
00ED 99 00 D0  STA-Y
00F0 C8        INY         NEXT ADDRESS
00F1 D0 F1     BNE $00E4   PAGE DONE?
00F3 68        PLA         RETRIEVE CHAR
00F4 60        RTS         EXIT SUBROUTINE
```

List 2

```
47000 REM-ONE KEY SCREEN CLEAR
47010 POKE 11, 223:POKE 12, 0:POKE 536, 216:POKE
    537, 0
47020 FOR M = 216 TO 244:READ D:POKE M, D:NEXT
47030 DATA 32, 186, 255, 201, 127, 208, 21, 72, 160, 0
47040 DATA 169, 32, 153, 0, 211, 153, 0, 210, 153, 0, 209
47050 DATA 153, 0, 208, 200, 208, 241, 104, 96
```



The Screen Squeeze Fix For CBM 8000

Richard Mansfield

Screen Squeeze is a fix for the many programs written for the regular 40-column PETs. Since the new 80-column series has virtually no software yet, this routine will permit games and graphics to work normally on a simulated 40-column matrix within the 80 column display.

When there is a ROM upgrade, there is a good reason for it: added BASIC power, eliminated bugs, new system features. The negative side is that much established software no longer works right. The problems are mainly in the SYS, POKE, and PEEK statements within programs.

To update your software, you must know, for example, that a certain POKE put you into the graphics mode, that a certain PEEK told you where to find the cursor, and that a certain SYS gave you a warm start. Then, after you locate all these commands in the program, you must assign the new, upgrade addresses to them. If, in addition, there are machine language routines, they will probably need adjustments too. Space in zero page, the first 256 bytes in the computer, is frequently important in M.L. programs and it has been getting progressively more scarce with each upgrade.

These are the main problems in software updating. But with the introduction of the new 80-column screen, extra adaptations are required.

It is true that there are several additions to screen formating in BASIC 4.0 in the 8000 series (double-screen) computers. You can define any screen size you want by printing CHR$(14) and CHR$(143) for the top left and the bottom right extremities respectively. This means that a LIST could scroll with nothing lost within a one-inch window, if you wish.

Or you can POKE 213,40 and the LISTS and PRINTS will be confined to the left half of the screen (exactly as if you has a 40-column display). But none of this solves our problem--any pokes outside of these boundaries will still be incorrect. And many graphs and games involve just such POKE statements.

The problem is best illustrated with an example. Let's assume that a ball rolls from left to right across the screen. It would be possible to PRINT such action, but it would be slower and less vivid, less animated. So, the ASCII number of the ball is POKED into column one, then extinguished with a POKE X,32 (a blank), then POKE X + 1,ball. And so on, across the screen. The important thing, here, is that the POKE addresses are calculated in the program and, in a complex game, you could spend days unraveling simultaneous equations trying to get things right for the bigger screen.

Screen Squeeze simply allows the wrong POKEs to take place and then, seeing them, corrects them in a flash. It is fast enough to make itself little noticed (there is some flickering on the right half of the screen).

But it will not solve everything, only the display itself. It cannot detect programmed blanks since all blanks look the same. This means that additional fiddling with the code (involving POKE X, 32) is necessary. But this is less frequent and more obvious than other graphics POKEs. ScreenSqueeze will simplify software adaptation, but not solve it.

Since this machine language routine is applicable to many tasks involving alterations to the screen, I will explain it in detail. The best way to learn the valuable art of programming in M.L. is to get a good book on the subject and then study examples. M.L. is valuable for two reasons--it is very much faster in execution than BASIC and it can do things not possible with BASIC code. Many real-time games (games where there is a time limit) involve animation which must be written in M.L. code to be fast enough.

896-916 set up two addresses (32768 and 32808) as the first targets. In addition, we enter the number 80 which will be added to the above two addresses each time as we finish checking a line for illegal POKEs.

918-920 Initialize our counters (the x and the y will keep track of how many columns we have checked and how many of the 25 possible lines).

922 get the potentially wrong POKE from address 32808.

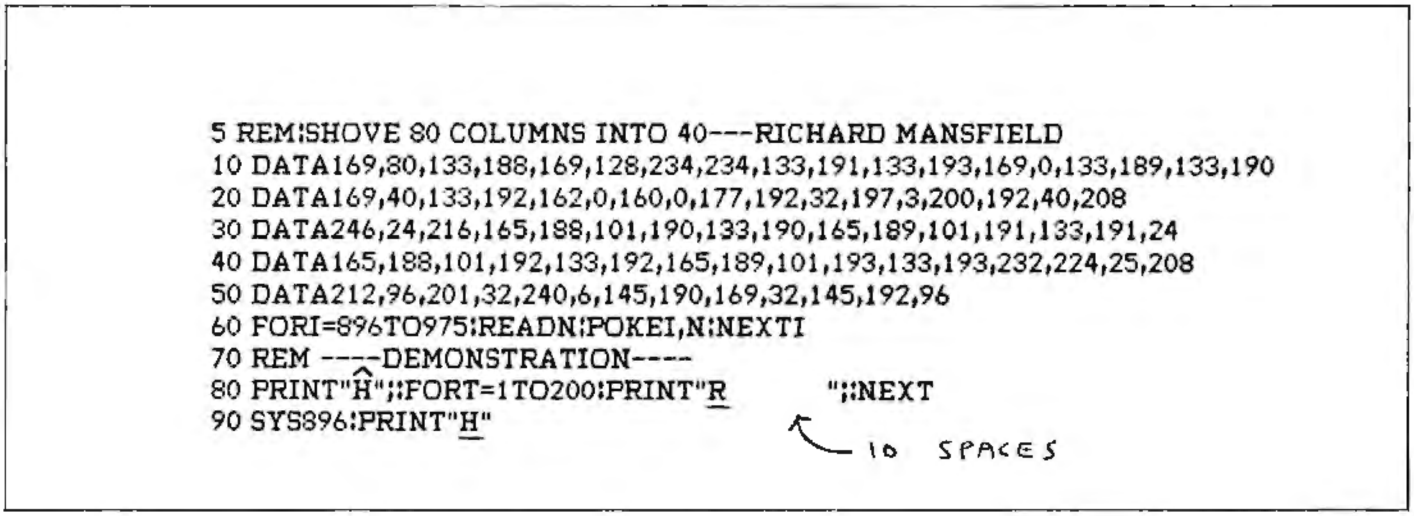
924 go to the subroutine which checks legality and adjusts for it.

927-930 increase the columns counter and, if less than 40, return to 922 for the second column check.

932-957 if all 40 columns were checked in that line, then add 80 to the target addresses.

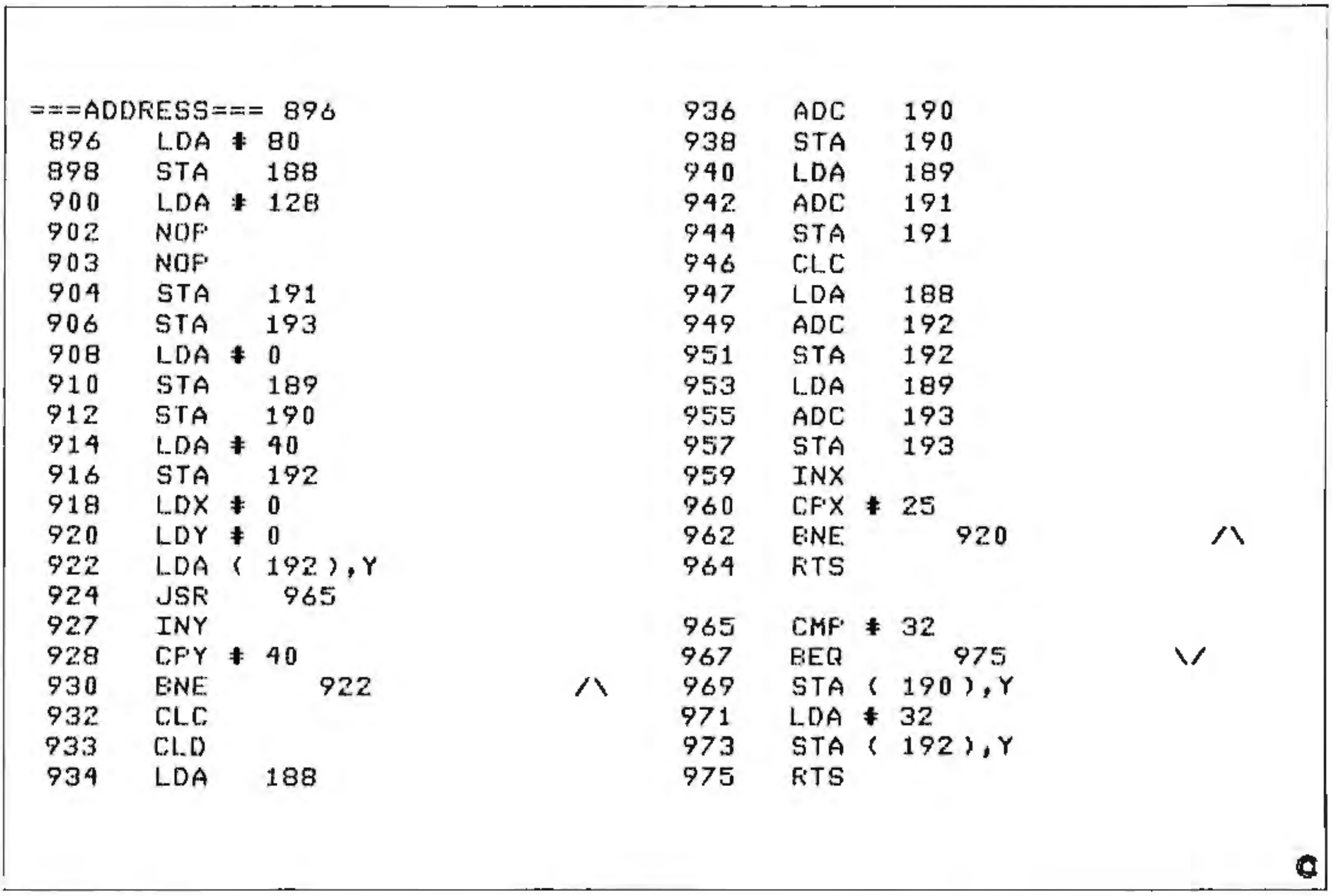
959-964 increase the lines counter and, if it is 25, return to basic. If not, go back to where the loop begins and start on the next line.

965-975 the actual task being performed--see if the screen address was blank. If it was, then go back (via return) to 927 where the columns counter is increased, and continue checking further addresses. If the address was not blank, then put the found character over 40 spaces to the left where it belongs. And, also, put a blank where this character was found. Then return to 927.



```
5 REM:SHOVE 80 COLUMNS INTO 40---RICHARD MANSFIELD
10 DATA169,80,133,188,169,128,234,234,133,191,133,193,169,0,133,189,133,190
20 DATA169,40,133,192,162,0,160,0,177,192,32,197,3,200,192,40,208
30 DATA246,24,216,165,188,101,190,133,190,165,189,101,191,133,191,24
40 DATA165,188,101,192,133,192,165,189,101,193,133,193,232,224,25,208
50 DATA212,96,201,32,240,6,145,190,169,32,145,192,96
60 FORI=896TO975:READN:POKEI,N:NEXTI
70 REM ----DEMONSTRATION----
80 PRINT"H";:FORT=1TO200:PRINT"R          ";:NEXT
90 SYS896:PRINT"H"
```

10 SPACES



```
===ADDRESS=== 896                                  936    ADC    190
 896    LDA # 80                                   938    STA    190
 898    STA    188                                 940    LDA    189
 900    LDA # 128                                  942    ADC    191
 902    NOP                                        944    STA    191
 903    NOP                                        946    CLC
 904    STA    191                                 947    LDA    188
 906    STA    193                                 949    ADC    192
 908    LDA # 0                                    951    STA    192
 910    STA    189                                 953    LDA    189
 912    STA    190                                 955    ADC    193
 914    LDA # 40                                   957    STA    193
 916    STA    192                                 959    INX
 918    LDX # 0                                    960    CPX # 25
 920    LDY # 0                                    962    BNE       920          /\
 922    LDA ( 192 ),Y                              964    RTS
 924    JSR     965
 927    INY                                        965    CMP # 32
 928    CPY # 40                                   967    BEQ       975         \/
 930    BNE        922            /\               969    STA ( 190 ),Y
 932    CLC                                        971    LDA # 32
 933    CLD                                        973    STA ( 192 ),Y
 934    LDA    188                                 975    RTS
```

Skyles Electric Works

BASIC Programmer's, Toolkit™, Disk-O-Pro™, Command-O™

For PET™ Owners Who Want More Fun And Fewer Errors with Their Programs

Here are thirty-five commands you'll need, all on dual chips you can install in two minutes without tools, **on any PET or PET system.** 2KB or 4KB of ROM firmware on each chip with a collection of machine language programs available from the time you turn on your PET to the time you shut it off. No tape to load or to interfere with any running programs.

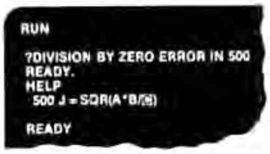
For PET/CBM 2001-8, -8N, -16N/B, -32N/B, 3016 and 3032

BASIC Programmers Toolkit™ commands

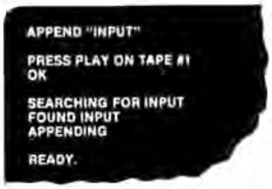
AUTO^{ed} DELETE^{ed} RENUMBER^{ed} HELP^{ed} TRACE^{ed}
STEP^{ed} OFF^{ed} APPEND^{ed} DUMP^{ed} FIND^{ed}

BASIC Programmers Disk-O-Pro™

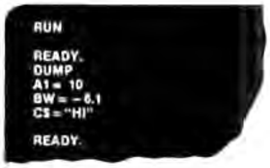
CONCAT^{B80} DOPEN^{B80} DCLOSE^{B80} RECORD^{B80} HEADER^{B80} COLLECT^{B80}
BACKUP^{B80} COPY^{B80} APPEND^{B80} DSAVE^{B80} DLOAD^{B80} CATALOG^{B80}
RENAME^{B80} SCRATCH^{B80} DIRECTORY^{B80} INITIALIZE^{BS} MERGE^{BS} EXECUTE^{BS}
SCROLL^{ed} OUT^{ed} SET^{ed} KILL^{ed} EAT^{ed} PRINT USING^{BS} SEND^{BS} BEEP^{BS}



```
RUN

?DIVISION BY ZERO ERROR IN 500
READY.
HELP
 500 J = SQR(A*B/C)

READY
```



```
APPEND "INPUT"

PRESS PLAY ON TAPE #1
OK

SEARCHING FOR INPUT
FOUND INPUT
APPENDING

READY.
```



```
RUN

READY.
DUMP
A1 = 10
BW = -6.1
C$ = "HI"

READY.
```

NOTES:

ed — a program editing and debugging command
B80 — a BASIC command also available on Commodore CBM™ 8016 and 8032 computers.
BS — a Skyles Electric Works added value BASIC command.
BASIC Programmers Toolkit™ is a trademark of Palo Alto IC's.
BASIC Programmers Disk-O-Pro™, Command-O™ are trademarks of Skyles Electric Works.
PET™, CBM™ are trademarks of Commodore Business Machines.
AVAILABLE: USA/CANADA: Please contact your local dealer
England: Calco Software Lakeside House, Kingston Hill, Surrey KT2 7QT
GERMANY: Unternehmensberatung, Axel Brocker Lennebergestr 4, 6500 Mainz
Japan: Systems Formulate, 1-8-17 Yaesu Shinmaki-cho Bldg. 11F Chuo-ku, Tokyo JAPAN 103

Phone or write for information. We'll be delighted to answer any questions and to send you the complete information package.

Skyles Electric Works

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

Skyles Electric Works

BASIC Programmer's, Toolkit™, Disk-O-Pro™, Command-O™

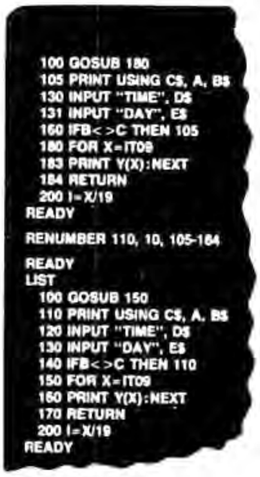
For CBM™ Owners Who Want More Fun And Fewer Errors with Their Programs

Here are nineteen commands you'll need, on a single chip you can install in two minutes without tools, **on any CBM or CMB system.** 4KB of ROM firmware on each chip with a collection of machine language programs available from the time you turn on your PET to the time you shut it off.

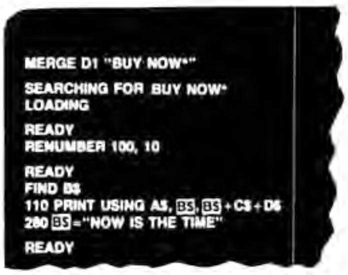
For CBM 8016 and 8032; BASIC 4.0

BASIC Programmers Command-O™

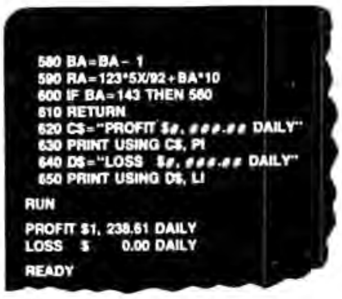
AUTO^{ed} **DUMP**^{ed} **DELETE**^{ed} **FIND**^{ed} (improved) **HELP**^{ed} **KILL**^{ed} **OFF**^{ed} **TRACE**^{ed} (improved) **RENUMBER**^{ed} (improved) **INITIALIZE**^{BS} **MERGE**^{BS} **MOVE**^{BS} **EXECUTE**^{BS} **SCROLL**^{ed} **OUT**^{ed} **SET**^{ed} **SEND**^{BS} **PRINT USING**^{BS} **BEEP**^{BS}



```
  100 GOSUB 180
  105 PRINT USING C$, A, B$
  130 INPUT "TIME", D$
  131 INPUT "DAY", E$
  160 IFB<>C THEN 105
  180 FOR X=IT09
  183 PRINT Y(X):NEXT
  184 RETURN
  200 I=X/19
READY

RENUMBER 110, 10, 105-184

READY
LIST
  100 GOSUB 150
  110 PRINT USING C$, A, B$
  120 INPUT "TIME", D$
  130 INPUT "DAY", E$
  140 IFB<>C THEN 110
  150 FOR X=IT09
  160 PRINT Y(X):NEXT
  170 RETURN
  200 I=X/19
READY
```



```
MERGE D1 "BUY NOW*"

SEARCHING FOR BUY NOW*
LOADING

READY
RENUMBER 100, 10

READY
FIND B$
110 PRINT USING A$, BS, BS +C$+D$
280 BS ="NOW IS THE TIME"

READY
```



```
  580 BA=BA- 1
  590 RA=123*5X/92+BA*10
  600 IF BA=143 THEN 580
  610 RETURN
  620 C$="PROFIT $#,###.## DAILY"
  630 PRINT USING C$, PI
  640 D$="LOSS  $#,###.## DAILY"
  650 PRINT USING D$, LI

RUN

PROFIT $1, 238.61 DAILY
LOSS   $     0.00 DAILY

READY
```

◀ NOTICE ▶

When you order **Command-O,** we will **loan** you a **Toolkit** until we deliver **Command-O.**

◀ NOTICE ▶

PRICES:

BASIC Programmers **Toolkit**™ (chip only)	$40.00
BASIC Programmers **Disk-O-Pro**™ (chip only)	$75.00
BASIC Programmers **Command-O**™ (chip only)	$75.00
Interface boards (needed sometimes)	$20.00-$50.00
Instruction Manual (with redeemable $5.00 coupon)	$5.00

Shipping and handling $2.50 USA/Canada, $10.00 Europe/Asia

California residents please add 6% or 6-1/2% sales tax as required

*Reserve your **Disk-O-Pro, Command-O** today*

Toolkit™ immediate delivery. Disk-O-Pro delivery in December. Command-O delivery in January

VISA, MASTERCHARGE ORDERS CALL (800) 538-3083 (except California residents)
CALIFORNIA ORDERS PLEASE CALL (408) 257-9140

Skyles Electric Works

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

Hooray for SYS

Harvey B. Herman
Department of Chemistry
University of North Carolina at
Greensboro
Greensboro, North Carolina 27412

The PET User's Club Newsletter (Vol. 1, Issue 3) reprinted an interesting article by Karl Hildon entitled "Probing PET's Memory". PETters were encouraged to experiment with the SYS command. Many of the routines in the PET are written as subroutines which terminate with the machine language instruction, RTS (return from subroutine). When a routine is initiated by the SYS command execution of the RTS instruction returns the PET to BASIC command mode or continues with the BASIC program. Using this idea we may be able to pick and choose useful segments of PET's Code and, in effect, make a new operating system.

This article describes three examples of tape operations which are not possible with just the normal BASIC commands. I hope readers will find them useful and are encouraged to develop similar ones on their own with the information supplied here. For convenience I have summarized in the Table the SYS calls and memory locations that I refer to in this article for both original and upgrade ROMs.

Occasionally I receive a tape of a machine language program with no information on its load limits. It is not possible to make a backup copy without this data. My first example is a BASIC program (see listing for TAPE DIRECTORY) which continually reads a tape and lists the start and end (+ 1) addresses for each program it finds. The idea behind this program and other examples here come from an article by Jim Butterfield, "Watching a Cassette Load", PET User Notes, Vol. 2, #1. He discussed several SYS commands in the article. The TAPE DIRECTORY program uses one of the SYS calls to load a tape header, containing among other information, the start and end addresses of the tape load. The addresses PEEKed from the beginning of the first cassette buffer are then converted to hexadecimal, using another Butterfield idea, and printed out. Note the use of the dynamic keyboard in statement 10 (cf., Mike Louder, "Best of PET Gazette") and the changes necessary for upgrade ROMs in statement 9.

More than once I have received a tape which would not load into my PET. For example, the program may have been saved from $4000 up in the originating PET and the highest location in my computer is $3FFF. My second BASIC example is a program (see listing for RELOCATE) which loads a cassette program into any area of RAM memory specified by the user. The loading addresses on the tape header are bypassed. I did the same trick manually in an article I wrote for MICRO ("MOVE IT", 16:17, with update 17:18). This program described here completely automates the procedure by using the dynamic keyboard idea. It asks for input of the starting location, loads the tape header, corrects the header information in the first cassette buffer, and completes the rest of the load. Note the changes necessary for upgrade ROMs (statements 165, 330, and 345). After relocation machine language programs will probably need some changes to reflect the new location before executing successfully.

Appending one BASIC program to another is a very useful operation. The final BASIC example (see listing for APPEND) has appeared before in many different guises. For example, as a wedge (Commodore PET User's Club Newsletter, Vol. 1, #4-5, p. 24), or with a SYS call to a machine language program (PET User's Notes, Vol. 1, #7). The BASIC Programmer's Toolkit (Palo Alto ICs) also has a built in tape append function. The BASIC programs described here (APPEND and APPEND NEW PETS) uses a similar set of SYS calls as RELOCATE. The programs first determine the end of BASIC from pointers in page zero. After the header is loaded the start/end information in the tape buffer is updated to reflect a remaining load which starts at the end of the first BASIC program. The program can be run repeatedly to append as many programs as desired. However, each program should have successively higher line numbers with no overlap. The append program can be deleted manually after use or the task could be automated using the dynamic keyboard.

Several problems came up when I tried to adapt the append program for upgrade ROMs. Jim Butterfield was kind enough to send me the locations corresponding to "load next header" and "load rest of tape". However, the load next tape routine did not have exactly the same effect as in the older model PETs. The new routine did not correct the chaining (links between BASIC lines) and update various pointers (e.g., to end of BASIC program). My APPEND program for new PETs needed an additional SYS call to correct the chaining and a separate update POKE so BASIC can keep track of the larger merged program.

The examples given here were all for cassette tape operation. There is no reason why examples have to be limited to tape. In the future, I hope to read about other examples where selective parts of PET's code BASIC are utilized in BASIC programs. Most of us, including myself can understand BASIC more easily than machine language and as long as speed is not a requirement, I see no reason why we can't use the 14K bytes of code in ways never dreamed of originally.

MTU

CODOS

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

HARDWARE

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

SYSTEM FEATURES

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

CODOS SOFTWARE

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

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

MASTERCARD & VISA accepted

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

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

TABLE
(hex locations in parenthesis)

Function	**Original ROMs**	**Upgrade ROMs**
load header	62894 (F5AE)	62886 (F5A6)
pointer to end of BASIC program	124/125 (7C/7D)	42/43 (2A/2B)
Header Buffer	635-638 (27B-27E)	same
load rest of tape	62403 (F3C3)*	62393 (F3B9)
correct chaining	50224 (C430)	50233 (C439)
current device #	241 (F1)	212 (D4)
LOAD/verify flag	523 (20B)	157 (9D)
# Chars. in keyboard	525 (20D)	158 (9E)
start of keyboard buffer	527 (20F)	623 (26F)

*Note: With original ROMs, the chaining is corrected and end of BASIC program pointer updated automatically after BASIC program load. With upgrade ROM, chaining and pointer update must be done manually.

```
2 REM TAPE DIRECTORY
5 REM HARVEY B. HERMAN
8 REM SET CURRENT DEVICE NUMBER TO TAPE ¬
      ¬1:LOAD NEXT HEADER USING DYNAMIC ¬
      ¬KEYBOARD
9 REM NEW PET-POKE212,1:SYS 62886:
      ¬POKE 158,1:POKE623,13
10 POKE 241,1:PRINT"↓↓↓SYS 62894:
      ¬GOTO20↑↑↑":POKE525,1:POKE527,13:
      ¬END
19 REM 635-638 TAPE BUFFER START AND ¬
      ¬END LOCATION
20 A=PEEK(635):B=PEEK(636)
30 GOSUB 110
35 PRINT
40 PRINT "TAPE START ";
50 GOSUB 120
60 A=PEEK(637):B=PEEK(638)
70 GOSUB 110
80 PRINT "TAPE END ";
90 GOSUB 120
100 GOTO 10
109 REM CONVERT HIGH/LOW BYTES TO ¬
      ¬LOCATION
110 C=256*B+A:RETURN
119 REM DECIMAL TO HEX CONVERSION-JIM ¬
      ¬B. IDEA
120 X=C/4096:FORJ=1TO4:A=INT(X)
130 IFA>9THENPRINT CHR$(A+55);:GOTO 150
140 PRINT CHR$(A+48);
150 X=(X-INT(X))*16:NEXTJ:PRINT:RETURN
```

```
100 REM RELOCATE
110 REM HARVEY B. HERMAN
120 REM INPUT START OF RELOCATION
130 INPUT "PROGRAM START LOCATION";C
140 REM CONVERT TO HIGH/LOW BYTES:
      ¬SAVE FOR LATER
150 GOSUB 400:SL=A:SH=B:NS=C
160 REM SET CURR. DEVICE NUMBER TO TAPE ¬
      ¬1:LOAD NEXT HEADER USING DYNAMIC ¬
      ¬KEYBOARD
165 REM NEW PET-POKE212,1:SYS 62886:
      ¬POKE 158,1:POKE623,13
170 POKE 241,1:PRINT"↓↓↓SYS 62894:
      ¬GOTO200↑↑↑":POKE525,1:POKE527,13:
      ¬END
180 REM 635-638 TAPE BUFFER START AND ¬
      ¬END LOCATION
190 REM FIND TAPE START
200 A=PEEK(635):B=PEEK(636)
210 REM SAVE FOR LATER
220 GOSUB 380:S=C
230 REM FIND TAPE END+1
240 A=PEEK(637):B=PEEK(638)
250 REM SAVE FOR LATER
260 GOSUB 380:F=C
270 REM CALCULATE NEW TAPE END
280 C=NS+(F-S):GOSUB 400:FL=A:FH=B
290 REM CORRECT TAPE BUFFER START AND ¬
      ¬END
300 POKE 635,SL:POKE 636,SH
310 POKE 637,FL:POKE 638,FH
320 REM FLAG-0/LOAD,1/VERIFY
330 POKE 523,0:REM NEW PET-157
340 REM LOAD REST OF TAPE WITH DYNAMIC ¬
      ¬KEYBOARD
345 REM NEW PET-SYS 62393:POKE158,1:
      ¬POKE623,13
350 PRINT:PRINT:PRINT"SYS 62403↑↑↑":
      ¬POKE 525,1:POKE 527,13
360 END
370 REM CONVERT HIGH/LOW BYTES TO ¬
      ¬LOCATION
380 C=256*B+A:RETURN
390 REM CONVERT LOCATION TO HIGH/LOW ¬
      ¬BYTES
400 B=INT(C/256):A=C-B*256:RETURN
```

```
100 REM APPEND
110 REM HARVEY B. HERMAN
120 REM SET CURR. DEVICE NUMBER TO TAPE ¬
      ¬1:LOAD NEXT HEADER USING DYNAMIC ¬
      ¬KEYBOARD
130 POKE 241,1:PRINT"↓↓↓SYS 62894:
      ¬GOTO150↑↑↑":POKE525,1:POKE527,13:
      ¬END
140 REM 635-638 TAPE BUFFER START AND ¬
      ¬END LOCATION
150 A=PEEK(635):B=PEEK(636)
160 GOSUB 400
170 REM SAVE TAPE START
180 S=C
190 REM FIND END OF BASIC PROGRAM
200 A=PEEK(124):B=PEEK(125)
210 GOSUB 400
220 REM CALCULATE NEW START LOAD
230 C=C-3:T=C:IFPEEK(635)=0THENC=C-1:
      ¬T=T-1
240 GOSUB 420
250 REM CORRECT LOAD POINT
260 POKE 635,A:POKE 636,B
270 REM FIND TAPE END(+1)
280 A=PEEK(637):B=PEEK(638)
290 GOSUB 400
300 REM CALCULATE NEW TAPE END
310 C=C-S
320 C=T+C
330 GOSUB 420
340 REM CORRECT END LOAD:SET LOAD/VERIFY
      ¬ FLAG TO LOAD
350 POKE 637,A:POKE 638,B:POKE 523,0
360 REM LOAD REST OF TAPE WITH DYNAMIC ¬
      ¬KEYBOARD
```

NORMAL
HITH
PLEXI-VUE HIGH CONTRAST VIEWING SCREEN! DECREASE GLARE & INCREASE CONTRAST! EASY ON THE EYES! INSTALLS IN 1 MINUTE ON PETS CBMS & TRS-80! ONLY $14.95
commodore
PET
2001 Series

PLEXI — VUE™
SOLAR SCREEN

DRAMATIC IMPROVEMENT?

YOU BE THE JUDGE!

FREE TRIAL OFFER

We urge you to test the PLEXI-VUE Screen now. Order one for our 30-day no obligation trial. See the dramatic difference it makes on your computer. See how much easier it is to read text with the higher contrast, and how much more you enjoy your computer. Your friends will notice the new appearance as the above photos show. We can make this offer because we have a QUALITY PRODUCT, at a REASONABLE PRICE that we feel will meet with your approval, if you will GIVE IT A TRY!

To order your PLEXI-VUE for our free trial, simply send your personal check or money order for $14.95 + $1.00 Shipping. We accept MASTERCHARGE or VISA! Give ACCOUNT NUMBER, INTERBANK NUMBER, EXPIRATION DATE, AND SIGN your order. Give Model Number needed from CHART:

PX1	=	PETs/CBMs with METAL CRT Cases.
PX2	=	PETs/CBMs with PLASTIC CRT Cases.
PX3	=	NEW 80 Character CRT CBMs.

★ ★ FREE PLEXI-VUE! ★ ★

Order a SOFTPAC-1 at the Regular price of $34.95 + $1.00 Shipping and we'll GIVE you the PLEXI-VUE FREE! The 30 day TRIAL OFFER applies to both! (You must return BOTH for a refund.)

SOFTPAC-1 contains 17 programs on DISC or TAPE (SPECIFY) in a Notebook with Back-up copies, printed instructions & program listings! GAMES W/SOUND too!

ORDER FROM:

COMPETITIVE
SOFTWARE
21650 Maple Glen Drive
Edwardsburg, MI 49112

Cyberia™ introduces two very practical additions for PET/CBM systems. . .

SuperBus™

Greatly multiplies system capabilities

$195
per computer

This is an active, integrated system—not just a passive network. SuperBus gives schools, banks, laboratories and businesses control of the way their networks operate.

- Up to 18 computers, disk drives or printers can be interconnected.
- Complete file security (program and data) and BASIC security.
- Built-in error detection and convenience features.
- Can both read and write to disk.
- All BASIC commands can be used.

Auto-Boot™

Simplifies PET/CBM operation

$49⁹⁵
per computer

- Auto-Boot automatically loads and runs the first program on disk (initializing if necessary).
- Completely compatible with WordPro™, CMS™,Visicalc™ and most other programs.
- Automatically detects BASIC errors and causes last program to be loaded and run.
- Easily disabled, returning the computer to normal operation.

CYBERIA Inc. 515-292-7634
2330 LINCOLN WAY, AMES, IOWA 50010

MasterCard, VISA and C.O.D. orders accepted. Specify N or B keyboard when ordering SuperBus. Dealer inquiries invited.
Visicalc is a trademark of Personal Software, Inc.; WordPro, of Professional Software, Inc.; and CMS, of CMS Software, Inc.

```
370 PRINT:PRINT"↓↓SYS 62403↑↑↑":
      ¬POKE 525,1:POKE 527,13
380 END
390 REM CONVERT HIGH/LOW BYTES TO ¬
      ¬LOCATION
400 C=256*B+A:RETURN
410 REM CONVERT LOCATION TO HIGH/LOW ¬
      ¬BYTES
420 B=INT(C/256):A=C-B*256:RETURN
```

```
100 REM APPEND NEW PETS
110 REM HARVEY B. HERMAN
120 REM SET CURR. DEVICE NUMBER TO TAPE ¬
      ¬1:LOAD NEXT HEADER USING DYNAMIC ¬
      ¬KEYBOARD
130 POKE 212,1:PRINT"↓↓↓SYS 62886:
      ¬GOTO150↑↑↑":POKE158,1:POKE623,13:
      ¬END
140 REM 635-638 TAPE BUFFER START AND ¬
      ¬END LOCATION
150 A=PEEK(635):B=PEEK(636)
160 GOSUB 400
170 REM SAVE TAPE START
180 S=C
190 REM FIND END OF BASIC PROGRAM
200 A=PEEK(42):B=PEEK(43)
210 GOSUB 400
220 REM CALCULATE NEW START LOAD
230 C=C-3:T=C:IFPEEK(635)=0THENC=C-1:
      ¬T=T-1
240 GOSUB 420
250 REM CORRECT LOAD POINT
260 POKE 635,A:POKE 636,B
270 REM FIND TAPE END(+1)
280 A=PEEK(637):B=PEEK(638)
290 GOSUB 400
300 REM CALCULATE NEW TAPE END
310 C=C-S
320 C=T+C
330 GOSUB 420
340 REM CORRECT END LOAD:SET LOAD/VERIFY
      ¬ FLAG TO LOAD
350 POKE 637,A:POKE 638,B:POKE 157,0
354 REM UPDATE ALL POINTERS
355 POKE42,PEEK(637):POKE43,PEEK(638):
      ¬CLR
360 REM LOAD REST OF TAPE AND CORRECT ¬
      ¬CHAINING WITH DYNAMIC KEYBOARD
370 PRINT:PRINT"↓↓SYS 62393":PRINT"↓↓↓SY
      ¬S 50233↑↑↑↑↑↑↑"
371 POKE158,2:POKE623,13:POKE624,13
380 END
390 REM CONVERT HIGH/LOW BYTES TO ¬
      ¬LOCATION
400 C=256*B+A:RETURN
410 REM CONVERT LOCATION TO HIGH/LOW ¬
      ¬BYTES
420 B=INT(C/256):A=C-B*256:RETURN
```

©

Machine Language
Scanning the Stack

Jim Butterfield, Toronto

The stack is a group of locations from hexadecimal 0100 to 01FF that can be quickly and conveniently used by the 6502. In the PET, the stack range is limited to the area from 0140 to 01FA; but most of the time you don't need to know where the stack is working: you may just use it.

When you have something that needs keeping for a few moments, you can put it on the stack and call it back later. So long as you're neat, you don't even need to know where it will go - the processor keeps track of that with a special register called a Stack Pointer. It will put information away to the stack and bring it back later without any special information from you.

But you must be neat. The slogan "Leave these premises as clean as you found them" applies critically to the way you use the stack. If you put something in there, you must be sure to call it back or you'll be in trouble.

The stack is appropriately named. It's like a stack of dishes: the first thing that comes off will be the last thing that was put on. It's called LIFO (Last-in-first-out) storage.

Standard usage

If you have a value in the A register that you want to put aside for a moment or so, you can push it to the stack using the PHA (48) instruction. Now you can use the A register for something else, and when you're finished you can call back the original value by pulling it from the stack with PLA (68).

Sometimes you might want to defer a decision. You've just done a comparison or some other activity, and the results are important - but you don't want to act on those results just yet. You can push the status word - all the various flags, such as Carry, Overflow, etc. - to the stack with PHP (08). Now you can tidy up your registers without worrying about losing those flags. They will come back as soon as you give PLP (28) and you can then proceed with the Branch commands that will test the condition you previously set up.

When you call a subroutine with a JSR command, the stack is called into play automatically. The return address, minus one, is placed on the stack. Later, when the RTS is given, that address is called back from the stack and program execution resumes at the instruction following the JSR.

An example here might be worth while. If you are at location hex 1234, and give the instruction JSR $4455, the address 1236 will be placed on the stack. That's not your return point - you'll return to 1237 since the JSR command is 3 bytes long - but the RTS instruction will sort everything out correctly. Here's a little more detail: when the address 1236 is placed on the stack it will use two locations. The high-order part (12) goes onto the stack first, followed by the low-order portion (36).

When the 6502 receives an interrupt - and on the PET this happens 60 times a second - the current machine language instruction completes; the address of the next instruction is pushed to the stack; and finally, the processor Status Word is pushed to the stack. Then the processor starts to handle the interrupt by going to a new location and executing instructions there. When it's finished, it gives a Return from Interrupt instruction (RTI) which restores the original Status Word and instruction address. The original program picks up exactly where it left off when it was interrupted.

You can see that three locations are used in the stack this time: two for the return address and one for the status word. They go onto the stack in that order: address-high, address-low, and status.

It's a little like a JSR followed by a PHP, since we store address and status word. Note, however, that the address is the exact return address; with a JSR the address is one less than the return address.

An example: if the processor is executing a three-byte instruction at hex 1234 and an interrupt is signalled, address 1237 is pushed to the stack, followed by the status word. Later, when RTI is executed, the status word is restored and execution resumes at address 1237.

Finally, the BRK instruction (hex 00) causes an interrupt type of action, with this difference: the address which is placed on the stack is two locations behind the Break instruction. This is odd, since the BRK command is only one byte long. In this case, if we use an RTI to continue executing the code following the BRK, we'll skip one byte.

Tabular Summary

The following table summarizes the instructions that handle the stack.

Number of bytes stored or recalled	Store command	Recall command
1	PHA	PLA
1	PHP	PLP
2	JSR	RTS
3	interrupt	RTI
3	BRK	

Here's where it gets interesting. "Ordinary" programming assumes that you use the companion instruction to restore the stack. That is, if you used a PHA you should use a PLA to bring the information back. If you used a JSR, you should use an RTS.

But by breaking this unwritten law, we can do some pretty fancy things. We must be careful, of course.

JINSAM

DATA MANAGER

SAVE TIME. SAVE MONEY. Let JINSAM work for you.

★CUSTOM DATA FILES
★CUSTOM REPORTS/LABELS
★KEYED RANDOM ACCESS
★FAST/EASY/MENU DRIVEN
★MULTIPLE SEARCH KEYS
★PRIVACY ACCESS CODES
★WILD CARD SEARCH

"JINSAM is the best Database Management System for the Commodore Computers !"

JINSAM data manager assists you by intellectually manipulating records.

No more will hundreds of valuable hours be spent searching for needed information. No more will hundreds of hours be spent entering and re-entering information for various reports.

With JINSAM you can truly transform your Commodore Computer into the "state of the art" data processing machine with sophisticated features and accessories found nowhere, even at 10 times the price.

There are three disk based JINSAM. JINSAM 1.0 allows fast and easy file handling, manipulation and report generation. JINSAM 4.0 was designed for the professional and contains features needed in the business environment, such as: JINSORT, a user accessible machine language sort; compaction/expansion of databases, merging databases and much much more. JINSAM 8.0 is our best. JINSAM 8.0 runs on the new Commodore 8032, 80 column display computer. JINSAM 8.0 has all the functions of 4.0 plus additional features found only on the most sophisticated and expensive database management systems.

JINSAM is a new breed of data processing software. Powerful, sophisticated and easy to use. JINSAM has been thoroughly field tested. JINSAM is now installed and saving its users valuable time and money in educational institutions, research institutions and offices nationwide.

JINSAM was designed with the user in mind. It is a forgiving system with help commands, prompts and utilities for recovering the bulk of data even after power failure, security passwords for privacy, editing, reclaiming space, auto recall, restructuring, unlimited report formats, label printing and a choice of accessory modules all accomplished by a few keystrokes.

JINSAM has 5 accessory interfacing modules:

WORDPROPACK - Intelligent interface for WORDPRO 3 or WORDPRO 4 which creates variable block with data or up to 10 conditions based on database contents. Produce "dunning letters", form letters, report to parent, checks, invoices, etc.

MULTI-LABEL - Prints multiple labels per record with up to 2 lines for messages and consecutive numbering. Produce inventory, bulk mail labels, etc.

MATHPACK - global +, -, x, ÷, by another field or a constant, or zero a field. Sum fields in each record or running sum of single field in all records. Extract information or effect permanent change. Replace in same field or place in a waiting field.

DESCRIPTIVE STATPACK - Determine mean, median, mode, standard deviation, variance, range. Generate histogram and produces Z-Score report.

ADVANCED STATPACK - (you must also acquire DESCRIPTIVE STATPACK). Generates CROSSTABS (number of occurances); CHI SQUARE, LINEAR REGRESSION with graphic representation and prediction. LINEAR CORRELATION and SIMPLE ANALYSIS OF VARIANCE.

All JINSAM accessories are accessed thru the JINSAM menu and require a security password to gain entrance.

JINSAM gives the user **FREEDOM OF CHOICE.** Start with JINSAM 1.0 and upgrade at any time. Choose from the accessory modules available at any time. JINSAM Newsletter brings the latest updates, user input and keeps an eye on the future.

JINSAM alone is reason enough to own a computer. JINSAM can be found at Commodore dealers. Write for the dealer nearest you.

The many features of JINSAM 1.0 – 8.0

JINSAM 1.0 for 16K/32K CBM 2001. Requires CBM 2040 or COMPU/THINK disk — including oldest ROMs. Menu Driven, ISAM - Indexed Sequential access method ● Encripted PASSWORDS for privacy ● Unlimited fields ● unlimited search criteria ● 3 deep subsorts ● .5 – 3 sec retrevial ● editing ● Auto Recall ● Wild Card Capabilities; Reports: multiple headings ● paging ● page numbering ● item count. Labels: any size ● 1-5 across ● sheet or continuous. Utilities: Help commands ● Recover ● Key Dump ● Record Dump ● Descriptor Dump ● Restructure.

JINSAM 4.0 for 32K CBM 2001 with BASIC 4.0. Requires CBM 2040 with DOS 2.1. Has most of JINSAM 1.0 functions Plus + machine sort with user access instructions ● sort 1000 records in apx 10 secs ● Global Compaction/Expantion ● Create new database from existing database ● merge databases. Includes MULTI-LABEL ● 4 deep subsorts. (Available Jan. 13, 1981)

JINSAM 8.0 for Model 8032 with 80 Column screen. Requires 2040 or 8050 disk. Commercial Disk version for 80 Columns, JINSAM 4.0 functions Plus + Displays report formats to screen, 4 deep subsorts. (Available Jan. 1, 1981)

JINSAM is a trademark of JINI MICRO-SYSTEMS, Inc.
WordPro is a trademark of Professional Software Inc.
CBM is a trademark of Commodore Business Machines.

JINSAM Data Manager
for Commodore Computers

— Additional Information

— Jinsam Demo Disk ($10, plus tax)

— Users Guide 1.0 ($25 plus tax)

Please send to:

Name ______________________
Position ______________________
Company ______________________
Address ______________________
City, State, Zip ______________________
Phone ( ) ______________________
Computer, Disk ______________________

JINI MICRO SYSTEMS, INC.
Box 274 ● Riverdale, NY 10463

Dealer inquiry welcome

The Fun Begins

Suppose you are writing a subroutine. Normally, you'll want to return control to the calling point by giving the RTS command. On occasion, however, you don't want to go back; perhaps there's an error in the data so that the calling routine couldn't continue

We can handle this. Just pop the return address from the stack with two PLA commands, and you'll never go back.

A little more detail on the tricks we can use here. Suppose you have a main routine at A, which calls a subroutine at B. Subroutine B, in turn, calls subroutine C several times. Subroutine C, which might for example be digging out a parameter for the SAVE command, decides it doesn't want to go back to subroutine B for some reason; perhaps there are no more parameters left (e.g., SAVE "PGM" instead of SAVE "PGM",1,2). In this case, it wants to go straight back to the main routine A.

When subroutine C is called, the stack will contain four values: two for the return address to A, and two for the return address to B. If subroutine C executes: PLA PLA RTS, it will throw away the return to B and go straight back to A.

Decimal Quickie

Want to find out if you're in Decimal mode? It's unusual in the PET, but the 6502 processor can switch to a special mode for addition and subtraction. There's a flag in the Status word that signals this. You could try a sample addition and see whether the result is calculated in decimal or not: CLC - LDA #$05 - ADC #$05 .. the result will be hexadecimal 0A if you're in binary mode, and hexadecimal 10 if you are in decimal mode.

There's a more straightforward way. Push the Status Word to the stack, and pull it back to the A register - execute PHP, PLA. You can now examine the bits of the Status Word at your leisure. Decimal mode is flagged in bit 3; you could mask it with AND #$08, for example.

The Computed Jump

You can jump to any single location you choose by using the JMP instruction. There are times, however, when you want to jump to one of several locations depending on some value you have calculated. For example, you might be writing a system which would jump to one routine if it detected an A (add) character; another routine for D (delete); a third for C (change); and so on.

This could be done, of course, with a series of compare, branch and jump instructions; but if the list is long, the whole thing becomes tedious and inefficient.

You can set up the equivalent of a very powerful computed jump by clever use of the stack. The principle is to manufacture an address; push it to the stack with PHA ... PHA; and then give RTS.

This seems puzzling at first. How can you return to a place you never came from? It works this way: by pushing the address to the stack, you simulate a non-existent subroutine call. The stack doesn't care. If you issue an RTS instruction, the stack will deliver up that address, and that's where you will go. The stack ends up unchanged: it has pushed two values and delivered them back.

Remember that the RTS instruction expects the address to be one lower than the real return address. If you want to go address hex 3456, you must push the values 34 and 55.

A quick example may help illustrate this powerful technique. Suppose the X register contains a value from 0 to 5. Depending on this value, we wish to jump to one of six different locations. We have built the destination addresses into a set of address tables, each with six entries. The low order part of the addresses are in a table starting at hex 2320 and the high order part of the addresses are in a second table starting at hex 2326. We've carefully remembered to subtract one from each address, and the table looks like the following:

```
2320 41 72 A3 C4 E5 F6
2326 24 25 27 29 2B 2C
```

If X contains zero, we want to jump to hex 2442; if one, we go to 2573; and so on. Let's do it.

```
BD 26 23 LDA $2326,X ; high order first
48       PHA
BD 20 23 LDA $2320,X ; low order last
48       PHA
60       RTS          ; go there
```

It's easy, it's fast, it's compact, and it's one of the most powerful tricks in the repertoire of the 6502 programmer. Microsoft Basic uses it to get to the various routines - PRINT, LET, FOR ... etc. The Machine Language Monitor uses it to interpret its commands - .M, .R, and so on.

The Stack Pointer

There are a couple of commands that tell you where the stack is working, or allow you to control where it is. You won't need to use them very often, but a little detail is worth while.

The stack works in a downwards direction. As you push things, the stack pointer gets lower. As you pull them back, the pointer goes back up. If the stack pointer gets down to its bottom value, pointing at address hex 0100, it will wrap around to 01FF if you try pushing more things in; but your program will be in serious trouble long before you reach this point.

The stack pointer indicates the next location that will be used. If the pointer has a value of 92, you know that the next value that you push will go into address 0192; or the next value that you pull will come from address 0193.

The command TXS - Transfer X to Stack Pointer, hex 9A - is most often used to reset the stack completely. This cancels everything: subroutines, interrupts, the whole works. On most PETs, you should set the pointer to hex FA with: LDX #$FA, TXS.

P
I
E
■
C

The command TSX - Transfer Stack Pointer to X, hex BA - is used for a couple of things.

You can check to see if you have too many things on the stack by coding TSX, CPX #$40 ... use whatever pointer limit you think is reasonable, but hex 40 is about as low as you should ever allow it to get.

The stack is in memory, of course; so you can look through the stack directly by examining the contents of locations 0100 to 01FF. You'll need to know where to start, of course, and TSX comes in handy here. If you give TSX followed by LDA $0100,X you will load the location to which the stack points. That may not be too useful, since it's the next location to be filled, and isn't part of the "active" stack. By incrementing X, however, or by looking higher with an instruction such as LDA $0101,X you can get to whatever part of the stack interests you.

Summary

Most of the time, the stack will take care of itself. Occasionally, however, you'll find that digging a little deeper into the mechanics of the stack can make it possible to do some very effective coding. ©

NEW LIFE for old Pets™?

Is it a dream...is it fantasy?

Switch from old ROMs to new Retrofit ROMs or from the new 24 pin ROMs to the new Commodore® Basic 4.0 with the Basic Switch.

- Not sure about the ROM Retrofit Kit from Commodore? Now you can use all three sets of Commodore ROMs and others as well.
- The Basic Switch allows switch selection of either ROM set (your original set or your retrofit set) from Commodore. Plus, Models 15-A and 15-B include an additional zero insertion force socket allowing easy use of ROMs like the BASIC Programmer's Toolkit...concurrently.
- Models 15-A and 15-B The Basic Switch plus...includes expanded cable assembly and zero insertion force socket. Your 15th ROM simply plugs in...enabled while either ROM set is selected. Socket 15 may be readdressed by the user for additional flexibility.
- The Basic Switch is sold in assembled form only. All models are designed for easy attachment to your PET with a convenient cable assembly. No soldering or drilling is required. The Basic Switch mates with a cable assembly at your primary board, and does not use the physical connectors of any PET ports.
- Our prices and complete product specifications are available by contacting APPLIED MICRO SYSTEMS, Mishawaka, Indiana; or any Commodore Dealer.

Now available for Commodore® Basic 4.0

Other APPLIED MICRO SYSTEMS Products include:

KEYTONE: Audio Feedback for your PET

- Monitors PET® keyboard, emitting a short beep when any key is pressed.
- Easy, solderless installation.
- Works with old and new PETs.
- Completely self contained, with speaker and externally accessible volume control.

Price $49.95 FOB Mishawaka, IN

PET® to Centronics® Printer Interface

- No software drivers required.
- Retains IEEE port to facilitate use with other devices, i.e., disk systems and/or plotters, etc.
- No external power supplies required, negligible power needed is derived from the printer.
- PET/CBM software compatible.
- Attractive metal case.

Price $139.95 FOB Mishawaka, IN

MEMORADD: Memory Expansion for your PET

- Adds 16K bytes to your present 16K PET.
- Fast, easy, solderless installation.
- Small (6¼" x 1⁹⁄₁₆" x ⅜") or (16cm x 3.5cm x .9cm).
- Derives power from PET.
- Can be used only with 16K PETs that presently use 4116 RAMs.

Price $199.95 FOB Mishawaka, IN

Dealer inquiries are encouraged. Free MACHINE LANGUAGE MONITOR COMMANDS List will be included with pricing and product specification requests.

APPLIED MICRO SYSTEMS

3502 Home Street, Mishawaka, Indiana 46544 • 1-219-259-3787 (Indiana) • 1-800-348-7208

Book Reviews

The Pet Revealed and Library of Pet Subroutines

By Nick Hampshire
Computabits Ltd., England
Available at Commodore dealers
Reviewed by Elizabeth Deal

It looks like the Pet's library is growing in the right direction. Commodore should be congratulated for making two new books available. These books supply some much needed, useful and correct documentation.

THE PET REVEALED

This book brings together some useful materials that have appeared previously and adds to it some very fine new material. The text clearly shows the relationship between technical and programming information. In terms of difficulty it falls between books that only teach Basic and books that describe the engineering aspects of the Pet. The text is written clearly and has a nice rhythm to it - just as you begin to come to grips with some hardware concept, the author gives an illustration of its use.

The book begins with a summary of the system hardware: address bus, data bus, clock, interrupt system, memory organization and video system. This is followed by a description of the 6502 CPU. This clear and fascinating introduction to the architecture of the Pet, followed by the logic of machine code programming will help me understand the more difficult text - Leventhal's *Programming The 6502*. The description of SYS, WAIT and USR commands is more thorough than any other I have seen. The book does not go into great detail on the basics of machine code programming, but it does show how to break up a task into steps much smaller than we do in BASIC, shows a few examples and gives hints for debugging.

The next chapter describes the operating system, again merging the hardware with software. It contains an accurate description of array storage, the first text I have seen to do so. It is full of hints on how to do unusual things on the Pet. It shows, for instance how to use the interrupt system, how to beat the old Pet's array size limitation, how to insert commands before a BASIC command is interpreted, and so on.

And finally, the two ports. Every register and its function is described and tied to the previous sections in the book. It is a gold mine of interfacing examples. Enough illustrations are given to help you understand how the Pet can be used in situations other than just grinding data. A short machine code program for music making is included. Use of serial CB2 line to do bit parallel I/O looks interesting. The description of how cassette units function looks good.

The computer user must be alert to confusion in designating operating states, such as: high-low voltage, 0-1 logic and event on-off. For example, there is a routine that allows the Pet to monitor an outside event, such as a mouse breaking a photoelectric cell beam, while the Pet is running another program. I had to make a minor change in coding, since I wanted to count how many times the line was grounded rather than the other way around.

Index, circuit diagrams, list of machine code instructions and a short errata sheet make information quite easy to look up. A table of Pet's six codes compactly lined up in decimal order is invaluable to me (decimal, hex, ASCII, screen, Basic and 6502 machine code). All in one place!. The printing has been done by use of a word processor, and the output is of type quality. The book is tightly written with little wasted space.

I have a few minor complaints:

1. I would have liked to see more circuit diagrams that show how to interface the user port for OUTPUT to devices that consume more power than the Pet can provide.
2. Several routines have been typeset rather than taken directly from the Pet printer. This invites errors. I found one on page 99. Two "F0 16" should be replaced by "F0 17" and "F0 15", in that order.
3. In chapter 3 several section subtitles are missing and there are few typographical errors. But you should have no trouble understanding what the author is saying. Commodore, Butler and Butterfield are cited as information sources, but other sources are not identified. For example, Tessler's Unlist, Cooke's IEEE handshake procedure, and Merrit's hooking up a TV monitor are familiar to me from the Pet User Notes. A map of VIA registers was published in Commodore User Club notes.

LIBRARY OF PET SUBROUTINES

This book contains 55 subroutines. Among them are input edit, screen input edit, trace, repeating key, sequential and random access systems, sorting by various methods, including a machine code sort, graphics in high resolution, and plotting from point to point, to cite just a few. Some routines are in BASIC, some in machine code, and many are mixes of the two. All routines are short, thus you can design your system for whatever application you want. All are written with a complete explanation of their purpose and methods of calling from the main program.

Skyles Electric Works

"If you could own only one peripheral for your PET, it should be this. It opens the whole world to your PET."

The Cat: Switches for mode selection and operation. LEDs display unit status. Acoustic self-test is standard. Compact powerpack plugs directly into wall socket.

Now $325.00* complete with membership in The Source*, Skyles six-foot cable/interface to the PET user port, together with Skyles cassette program in machine language and in BASIC.
(If bought separately: $180.00, modem; $80.00, cable/interface and program; $100.00, membership in The Source)*

The Cat and D-Cat have been specially prepared by Skyles for interfacing to the PET user port (*not* to the IEEE port) and with a special cassette program, allowing communication

...from modem to disk and disk to modem
...from modem to terminal; read on screen, save on disk
...from disk to printer through IEEE
...from disk to screen

The D-Cat: FCC-approved for handset jack connection with any modular phone, either single or multi-line. No need for adapters. Can fit under phone; installs in seconds. Mode switch to monitor voice or data transmission. Special "hold" function; complete self-test. Power pack plugs directly into wall socket.

Now $350.00* complete with membership in The Source*, Skyles six-foot cable/interface to the user port and Skyles cassette program in machine language and in BASIC.

(If bought separately: $210.00, the modem; $80.00, cable/interface and program; $100.00, membership in The Source)*

*What About the Source?

Sometimes called "The Information Utility," it's a telecomputing network that gives you, through your Cat or D-Cat modem, thousands of programs and data bases and allows you to communicate with other users interactively and through electronic mail.

*California residents please add 6% or 6.5% sales tax as required

VISA, MASTERCHARGE ORDERS CALL (800) 538-3083 (except California residents)
CALIFORNIA ORDERS PLEASE CALL (408) 257-9140

Skyles Electric Works

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

There is a program for appending a series of routines from a disk to form a larger program. The disk allocation is about 9K so that the entire package can be saved in the same place.

All program listings are copies of printer output and therefore free from typesetting errors. The listings are very readeable, the code is not tight and comments are provided. This allows modifications to be made without much trouble. There is a thumb index on the side of the page which helps you locate any listing in a jiffy.

I would like to recommend that in the routines that change the pointer to top of the Pet you perform the change in direct mode. There is always a danger that the machine code may get wiped out by strings. **POKE52,xxx:POKE53,xxx:CLR** will guarantee the safety of the code.

I've tried many of these routines and they worked without any problems. They are well prepared, fun to use, and will help in better program development.

It is my understanding that these subroutines will be provided on disks or cassette tapes (about 50 minutes total length) without extra cost to those who purchase both books. ©

Review

A VISIBLE MUSIC MONITOR FROM AB COMPUTERS

Arthur B. Hunkins
School of Music, UNC-G
Greensboro, N.C. 27412

The Visible Music Monitor (VMM) is a unique and remarkable machine language program for the 8K PET/CBM that codes, saves, edits, displays in musical notation, and performs up to four-voice music. Dr. F. Levinson is the ingenious author. VMM is available in both old and new rom versions; it will work with any D/A converter, including A B Computers' new KL-4M. The extensive features of this program are too numerous even to list here; suffice it to say that they are all highly useful and oriented toward the user.

In addition to a nifty "record changer" playback mode, which permits an entire series of arrangements to be performed without user intervention, two features of VMM are particularly notable: one is the user-definable keyboard. The user may define the PET keyboard in any way he likes for convenient pitch entry (a standard default option is also available). The other especially important feature is the extreme ease of editing, which could alone justify use of the system. With such capability one can be a true electronic arranger, trying various alternatives, creating related versions, transposing, deleting and adding segments or measures, adding individual notes, etc. It is all done by editing (using the PET cursor/edit keys) the musical notation on screen.

Both coding system and musical notation are highly abstract, and take some getting used to. Much of this is due to inherent limitations of the PET in keyboard data-entry and character graphics. The notation is oversized; treble and bass staves cover the entire screen and ledger lines cannot be accommodated (upward arrows are used instead!) In other ways too the notation is abstract and sometimes simplified: there are no ties, no beams (only flags for individual notes; flags go in the wrong direction), no rhythmic spacing, no clef signs, "F" for a flat sign, and " = " for natural. Perhaps most difficult to read are the spread-out chords, notated one note after the other, each with a separate stem, with little slashes to show which notes "go together." In most of these cases there are good (computer-oriented) reasons for departure from traditional notation. The idiosyncrasies do, however, affect the legibility and perhaps the usefulness of the system to the arranger. (Preliminary documentation maintains that the difficulties associated with abstractness in coding and notation can be easily overcome.)

Musical limitations of VMM are by and large those of all four-voice synthesis programs currently available. One of these limitations is the lack of amplitude or timbre envelope; loudness and tone color remain constant for each note (most interesting sounds change). Another is the absence of dynamics; all notes are the same loudness, and voices cannot readily change level. (It is theoretically possible to make one voice softer than another, but doing so is not easy, and the procedure is not explained in the documentation.) Finally, there is the omnipresent bugaboo of clicks--in this case, very audible clicks. The chord-by-chord synthesis approach ("Chamberlin-style" music) shared by all software synthesis systems causes all notes to be reinitialized (i. e., "clicked") whenever *any* chord tone changes. All notes are effectively repeated whenever any note changes. Thus there is no essential independence of rhythm. These limitations, in one form or another, are likely to be with us for some time. Overcoming them will require sophisticated programming, larger amounts of memory, and perhaps additional hardware. They *must* be overcome, however, for serious creative musicians to become interested in true microcomputer music synthesis.

I leave it to the individual to decide whether the Visible Music Monitor is sufficiently worthwhile to use as a musical arranging/coding tool. Its editing and debugging capability is clearly the most significant feature. Let me reiterate that VMM is a novel and sophisticated system, which should prove lots of fun for the devoted musical hobbyist/arranger. It is a most inventive piece of software, and author Levinson is to be highly commended for a difficult job well done. Cost of the cassette, with its substantial documentation, is a reasonable $29.90. ©

DR. DALEY'S *BEST* Mailing List Is Now Better!

DR. DALEY has taken his best selling mailing list and made it even better! This version has been totally revised to increase the reliability of the files and make it even easier to operate. Several new features have been added:

- Goof-proof input routine. Eliminates the irritating results of accidentally pressing some cursor control keys. This is a machine code routine so it is as fast as you are! BONUS—Auto repeat on all keys!
- Interface to allow output of the entire mailing list or virtually *ANY* subset to WORDPRO III and WORDPRO IV format files so you can use these to generate personalized form letters. *YOU* can format the structure of this output!
- Routines to merge files and to minimize the number of duplicate entries in a file.
- More machine code routines to speed up processing.
- In addition you have the same powerful file formatting options where *YOU* can determine the structure of the files. *YOU* can format your label output with up to 11 lines per label and from 1 to 8 (yes EIGHT) labels per line.

This system is completely menu driven. It includes 100 pages of user documentation. This documentation is for the end user and is not padded with listings, flow charts, and other such extraneous material.

This program will be available for a short time at the introductory price of $159.95. It is available for the 32K PET and CBM 2000, 3000 and 8000 series computers. You can order through your dealer or directly from us. We will accept VISA or MASTERCARD or your check or money order. Overseas orders include 10% to cover shipping.

Charge to
your
MC/VISA

master charge
THE INTERBANK CARD

VISA

DR. DALEY'S Software

425 Grove Avenue, Berrien Springs, MI 49103
Phone (616) 471-5514
Sunday - Thursday noon to 9 p.m., Eastern Time

Review: DISK-O-PRO™, Skyles Electric Works, Mountain View, CA. Price: $75.00

Review by Jim Butterfield, Toronto

Disk-O-Pro is a ROM chip which plugs into your PET/CBM. It is intended to be used with upgrade ROM to produce a system which recognizes a whole battery of new commands.

Many of these new commands are the same as on the 80-column CBM machines or on 40-column PETs fitted with new 4.0 ROM systems. Other Disk-O-Pro commands are extra, and don't have counterparts in any standard Commodore system.

Disk-O-Pro is designed to fit into a PET without disturbing certain other "enhancement" systems. For example, it will peacefully coexist with such products as the Toolkit (tm). That's a neat trick, since both packages zero in on the same "wedge" area of Basic, and it needs some work to avoid conflict.

A good part of the attractiveness of this package is the lure of being able to use the new Commodore commands without having to abandon existing program tools that the user has purchased. Compatibility is not 100%, however; and it may be worth a brief discussion on the nature of Basic-extenders and the role that Disk-O-Pro and like packages may play.

Editors vs. Interpreters

Many of the existing software aids help a program in its development stage, but are not needed after the program is complete. Your Toolkit may be great for writing and debugging a program; but when the program is finished, it will run on anybody's PET -even those not equipped with a Toolkit. In a similar way, the DOS support program (the "wedge") helps you load and save programs, but isn't needed when the program runs.

Such packages effectively disconnect during a program run. As a result, Basic programs run at virtually full speed.

Disk-O-Pro, on the other hand, participates in the actual running of a program. This means two things: your program will run somewhat slower; and you may not be able to run the same program in a system which does not have Disk-O-Pro installed.

Compatibility with the new Basic 4.0 is quite good. Disk-O-Pro follows many of the internal 4.0 procedures - the tokens are written identically, for example - so that there's a good chance of a program using these features being portable. But the user should be aware that there are minor differences; and sometimes a minor item can hang up your program.

The Compatible Features

All the new disk-oriented commands are there: CATALOG, COPY, SCRATCH, and so on. They may be abbreviated in the usual way: C, shift-A may be used as a short form of CATALOG, for example. Some features such as APPEND won't be needed unless you have the new disk system which supports this function, but they are supplied just in case.

The special variables DS and DS$ are provided, which make error checking from disk a snap. There's a slight difference in the way these variables are handled between the two systems. Basic 4.0 checks the disk status only when the variables are referenced. Disk-O-Pro checks with every normal disk activity, and stores DS and DS$ as ordinary variables. Such variables will disappear after a program change, a CLR, or a LOAD.

An extra disk command, INITIALIZE, is provided for users of the first 2040 disk system (DOS 1.0); they will find it very handy.

What's Missing?

There are a few things that have been implemented in Basic 4.0 that Disk-O-Pro can't handle. The changes to Basic are too fundamental for an add-on package to be able to cope with them.

Garbage collection delay is still there with Disk-O-Pro. The old methods of reducing this delay still work (changing and restoring Top-of-Basic at exactly the right times); but the definitive 4.0 solution just can't be patched in.

Basic 4.0 has a nice touch which is a big help when writing disk files: in normal operation, the Linefeed character is completely eliminated from output. On Disk-O-Pro, you'll still have to use the traditional " .. :CHR$(13); .. " phrase at the end of each PRINT# line.

There are other minor discrepancies which are likely to be seen on rare occasions. There's a problem with unwanted "flashing" of the EOI line on the IEEE-488 bus whenever the screen scrolls. Basic 4.0 has eliminated this; Disk-O-Pro hasn't. Not important unless you're using two computers to run one disk. On the other hand, if you happen to have two disks on one computer, you may run into another small discrepancy: 4.0 always defaults to device 8, whereas Disk-O-Pro goes to the last device referenced. None of this is likely to impact the average user, but it's well to be aware that the differences are there.

Extra Features

Disk-O-Pro is more than just a Basic 4.0 imitation, however: it has a added features of its own. Remember that if you use these features in a program, you'll have trouble running on a system which is not fitted with Disk-O-Pro.

PRINT USING is very useful; it's the answer to printing financial and other numbers neatly in columns. It's nicely done in Disk-O-Pro, and allows numeric values, strings, and literal characters.

Numerics can have commas inserted and values which are too large to fit are flagged in the printout, usually with an asterisk character. One minor problem on PRINT USING: if you don't leave room for the sign character it will be dropped, and a negative value will print as if it were positive.

SCROLL invokes several very useful editing features. After the command is given, the cursor movement keys will repeat automatically after an initial pause. Even better: if you have a program listing on the screen and run the cursor down to the bottom, the screen will scroll and new lines of the program will appear. Scroll to the top, and earlier lines of the program come into view. This can save you the trouble of typing LIST over and over again.

SCROLL also invokes a new set of keyboard operations. Most noticeable is the "softkey" feature: this allows you to redefine any key by using the SET command. How do you redefine a key? Type SET "GOSUB 800" ON "@" and whenever you strike the @ key you'll get GOSUB 800 on the screen. Once you get used to it, you'll find this very useful.

There are several other commands available, including BEEP to beep over the CB2 line, MERGE to stitch two programs together, and KILL to disconnect Disk-O-Pro.

Speed Considerations

The Disk-O-Pro remains active during a Basic program run. It has to, since it must detect special commands. DOPEN, SCRATCH, or BEEP, for example, call for appropriate action during a run, and just searching for them takes some time. As a result, programs won't run as fast when Disk-O-Pro is in place - even when these special commands are not used in the program. If your program used none of them, you could disconnect with KILL before running.

Time penalty varies depending on the type of work done. It can be as low as 10% for simple FOR/NEXT loops with arithmetic inside, to as high as several hundred percent (!) when using GET# statements and concatenation to drag material from disk (all those DS updates...). The typical run time penalty is around 15%; but this must be seen in perspective. Many programs spend a lot of time inputting or outputting; actual computation speed won't make much difference to these. Most of the time will be spent in waiting for the user to type a response at the keyboard, or waiting for the printer to output a line: any delays due to Disk-O-Pro won't be noticeable.

For short computation jobs, an increase in run time from 20 seconds to 23 seconds won't be serious. For longer jobs, Disk-O-Pro may need to be KILLed .. but there will be plenty of other things that the time-sensitive programmer will need to go after, too.

Summary

Disk-O-Pro is an interesting new concept that will certainly be useful to many PET users. It's nicely put together and convenient to use.

Many users will have to choose between upgrading to the new Basic 4.0 system and fitting Disk-O-Pro. They will need to weigh the alternatives carefully.

In some cases, elimination of garbage collection delays, potentially faster run time, and compatibility with future products will rule in favor of Basic 4.0. In other cases, the lower cost of adding Disk-O-Pro, compatibility with an existing Toolkit, and extra features such as PRINT USING and scroll/edit features will tip the balance toward Disk-O-Pro. ©

Detecting Loading Problems And Correcting Alignment On Your PET

Dan Isaacson

Loading

When you load a program into your PET do you **?PEEK(630),ST?**

As you know, when you RUN an incompletely loaded program your PET will crash, forcing you to turn it off and power up again. When you are loading a program for copying it would be nice to know that you've made a copy of the complete program--one with no dropped bits.

The PET recording system uses a dual redundant method. Programs are recorded twice on cassette--a block of data followed by an identical block, followed by a block of different data, followed by a block identical to that data, etc.

DATA BLOCK 1 DATA BLOCK 1
DATA BLOCK 2 DATA BLOCK 2

In LOADing a tape, the computer compares the two blocks, bit for bit, and stores in location 630 a count of unmatched (or not identical) bits in the two blocks. After a program LOADs, if you

```
?PEEK(630),ST
0            0
```

will be returned if there were no bits dropped and if the STatus bit was read properly at the end of the LOAD, thus informing the computer of a satisfactory load. As long as location 630 has a 4 or less stored in it you have a satisfactory load which can be copied with confidence.

Tape-reading problems

Location 630 also can tell you if you have a head alignment problem--and, incidentally, can help you to re-align your tape head without sending it back to your supplier.

If your PET is not loading properly, LOAD several tapes and ?PEEK(630). If *all* the tapes give non-zero responses either your tape head is out of alignment or the recording level is too low on all of them. Low recording level is not a problem with PET recordings as it may be with the TRS-80. "Low" level usually comes from a blank tape which you *thought* was recorded on, from a tape which got too near a source of magnetism and was partially or wholly erased, or from a tape which got too near a source of magnetism and was partially or or wholly erased, or from a tape which has been twisted in the cassette and which now has its non-magnetic side facing outward.

It's easy to test volume level of a suspect tape on an audio tape recorder against a tape that works. You can hear the volume difference easily. The odds are very low, however, that more than one tape in a group has a low recording level.

Field-alignment

To field-align your cassette head you will need a thin, Phillips-head screwdriver and several commercially produced tapes with which to verify tape-head alignment. The strategy I'll describe (before I tell you where to find the screw) is to:

1. LOAD a tape

2. ?PEEK(630),ST

3. Write down the result of (2)

4. If the result is not both zeros, turn the head-adjusting screw slightly counter-clockwise (about 1 minute on a clock, or about 6 to 10 degrees).

5. Write down which way you turned the screw (clockwise or counter-clockwise).

6. Do steps 1 through 3 again.

7. If your PEEK gives a higher number of dropped bits then you turned your Phillips-head screwdriver the wrong way.

8. Turn the screwdriver in small increments, repeating steps 1 through 3 until PEEK gives you 0 0.

9. Finally, LOAD 3 or 4 commercial and also locally-made tapes to make sure the results are always 0 0. Your tape head is then aligned to read the tapes you need to read.

CAUTION: To take the slack out of the adjusting screw when turning it clockwise, first turn the screwdriver clockwise slightly past the position you want and then turn it counter-clockwise back to the position you want to end at.

Although the alignment procedure described takes time, it should be possible for any handy youngster following these directions to do a satisfactory job.

Where is the adjustment screw?

The PET has been delivered with two different models of tape recorders: one has a cover you lift by hand (see Figure 1), the other has a push-button-eject cover (Figures 2 and 3). The location of the adjusting screw for each is shown in the photos.

To get at these screws, the PLAY button must be depressed. The power may be present or off during adjustment, but the PLAY button must be depressed so the screw moves to a position under the hole where you insert your screwdriver.

Once you've found the screw, go back to follow the instructions in the field-alignment section.

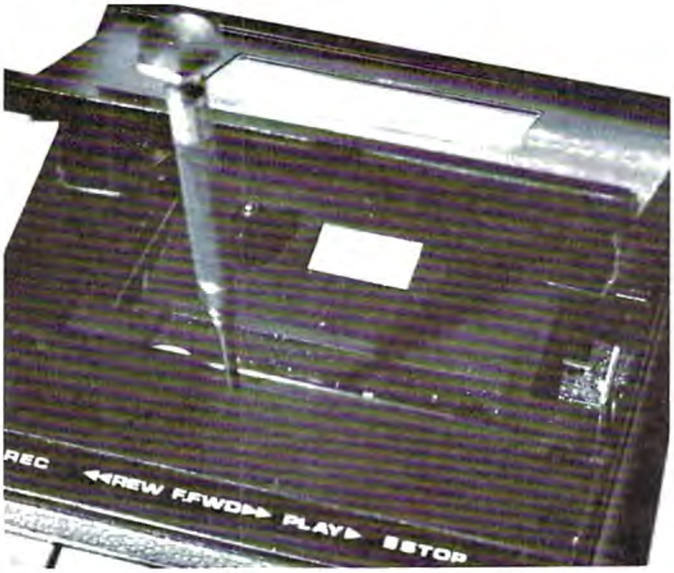


Figure 1 - Location of the adjustment screw hole for the hand-lifted cover model.

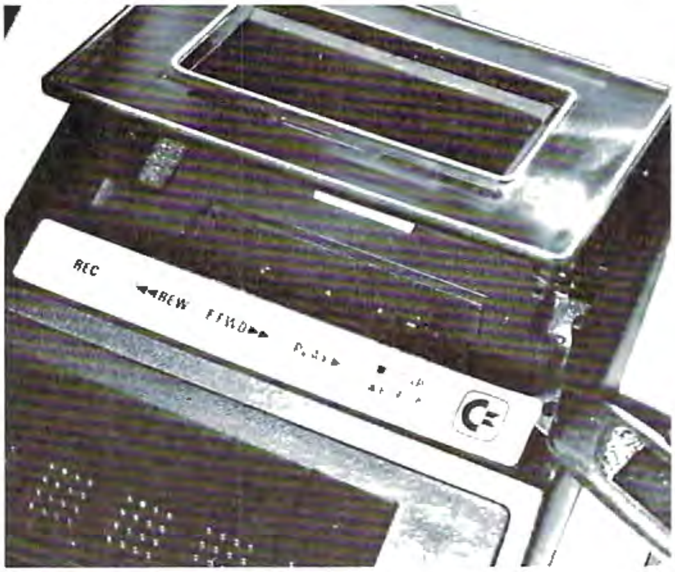


Figure 2 - To get at the screw hole in the push-button-eject model you must remove the metal label strip as shown.

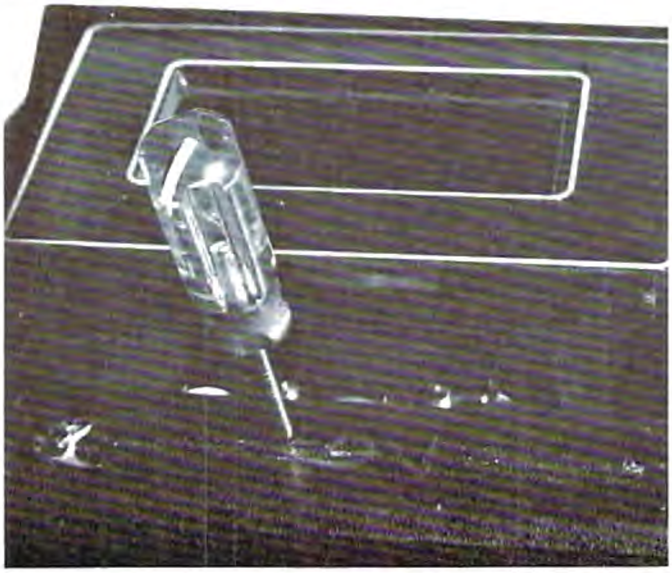
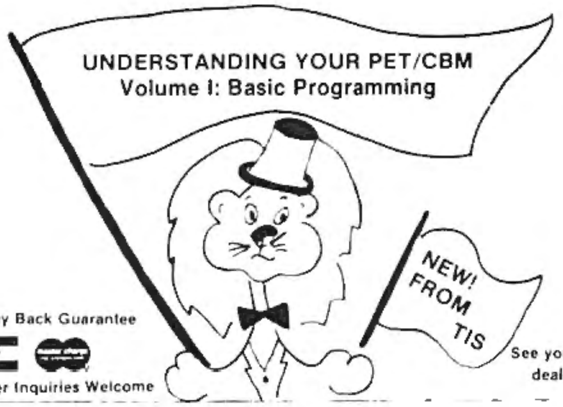
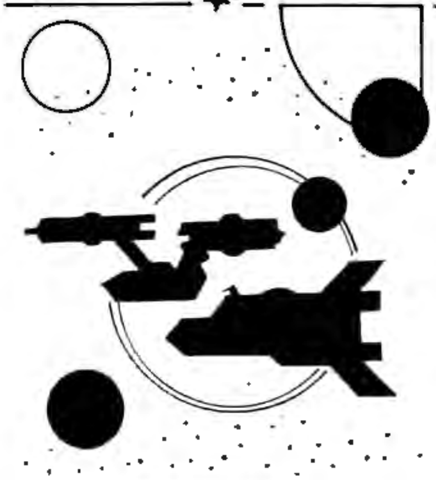


Figure 3 - Location of the adjustment screw hole for the push-button-eject model.



Get Fireworks From Your PET*!



TREK-X

TREK-X Command the Enterprise as you scour the quadrant for enemy warships. This package not only has superb graphics, but also includes programming for optional sound effects. A one-player game for the PET 8K. **Order No. 0032P $7.95.**

DUNGEON OF DEATH

DUNGEON OF DEATH Battle evil demons, cast magic spells, and accumulate great wealth as you search for the Holy Grail. You'll have to descend into the Dungeon of Death and grope through the suffocating darkness. If you survive, glory and treasure are yours. For the PET 8K. **Order No. 0064P $7.95.**

ARCADE I

ARCADE I This package combines an exciting outdoor sport with one of America's most popular indoor sports:

•**Kite Fight**—It's a national sport in India. After you and a friend have spent several hours maneuvering your kites across the screen of your PET, you'll know why!

•**Pinball**—By far the finest use of the PET's exceptional graphics capabilities we've ever seen, and a heck of a lot of fun to boot.

Requires an 8K PET. **Order No. 0074P $7.95.**

CODE NAME: CIPHER

CODE NAME:CIPHER

Enjoy that same feeling of intrigue and discovery with the Code Name: Cipher package. Included are:

•**Memory Game**—Would you like to match your memory against the computer's? You can with the Memory Game.

•**Codemaster**—One player types in a word, phrase, or sentence, and the PET translates that message into a cryptogram. The other player must break the code and solve the cryptogram in the shortest time possible.

•**Deceitful Mindmaster**—This isn't your ordinary Mastermind-type game. You must guess the five letters in the hidden code word.

•**Code Breaker**—Cracking this code won't be as easy as cracking walnuts. You'll need to flex your mental muscles to win this game.

If you want a mental challenge, then Code Name: Cipher is for you. For the 8K PET. **Order No. 0112P. $7.95.**

Instant Software™

*A trademark of Commodore Business Machines

PETERBOROUGH, N.H. 03458
603-924-7296

Captivate Yourself.

Santa Paravia and Fiumaccio

The most captivating and engrossing program ever made for the PET*

SANTA PARAVIA AND FIUMACCIO Become the ruler of a medieval city-state as you struggle to create a kingdom. Up to six players can compete to see who will become the King or Queen first. This program requires a PET 16K. **Order No. 0175P.$9.95.**

It is the dawn of the 15th Century; you rule a tiny Italian city-state. Your goal: The Crown!

Up to six players can compete as rulers of neighboring cities. You control the grain harvest, feed your serfs, set tax rates, dispense justice and invest in public works.

The future of your realm will depend on your decisions. If they are wise, your city-state will grow and you will acquire loftier titles. If your rule is incompetent, your people will starve and you may be invaded by your neighbors.

How will *you* rule your kingdom? Will you be an enlightened leader—or an unscrupulous despot? Only you can answer that question—with Santa Paravia and Fiumaccio.

CHIMERA

CHIMERA If you think the legendary Chimera was hard to handle, wait until you try the Chimera package. Included are:

•Reflex—Round and round the little white ball rolls. Only fast reflexes can guide it into the center of the maze.

•Dragon—You'll have to shoot down those pesky, fire-breathing dragons with your cannon. If you succeed your castle will be safe, if not it will mean a call to your fire insurance company. For one player.

•Dungeon—A very punctual guard comes down to the dungeon every day to torture you. This means that you have only thirty seconds to dig your way under the castle to freedom. For one player.

•Dragon Hunt—You must go forth and slay a fire-breathing dragon. The only thing that will protect you from the flames is your shield, if you know when to use it. For one player.

•Dropoff—You must make your opponent's men "dropoff" the board by moving and firing your own men. For one or two players. **Order No. 0110P. $9.95.**

PET DEMO I

PET DEMO I You can give yourself, your family, and your friends hours of fun and excitement with this gem of a package.

•Slot Machine—You won't be able to resist the enticing messages from this computerized one-armed bandit.

•Chase—You must find the black piece as you search through the ever-changing maze.

•Flying Pheasant—Try to shoot the flying pheasant on the wing.

•Sitting Ducks—Try to get your archer to shoot as many ducks as possible for a high score.

•Craps—It's Snake Eyes, Little Joe, or Boxcars as you roll the dice and try to make your point.

•Gran Prix 2001—Drivers with experience ranging from novice to professional will enjoy this multi-leveled race game.

•Fox and Hounds—It's you against the computer as your four hounds try to capture the computer's fox.

For true excitement, you'll need a PET 8K. **Order No. 0035P $7.95.**

DOW JONE$

DOW JONES Up to six players can enjoy this exciting stock market game. You can buy and sell stock in response to changing market conditions. Get a taste of what playing the market is all about. Requires a PET with 8K. **Order No. 0026P $7.95.**

We Guarantee It!

Instant Software Guarantee

OUR PROGRAMS ARE GUARANTEED TO BE QUALITY PRODUCTS. IF NOT COMPLETELY SATISFIED YOU MAY RETURN THE PROGRAM WITHIN 60 DAYS. A CREDIT OR REPLACEMENT WILL BE WILLINGLY GIVEN FOR ANY REASON.

Prices subject to change without notice.

TO ORDER

SEE YOUR LOCAL
INSTANT SOFTWARE
DEALER

OR

call now
Toll-Free
1-800-258-5473

Instant Software™

PETERBOROUGH, N.H. 03458
603-924-7296

* A trademark of Commodore Business Machines

Spooling For PET With 2040 Disk Drive

T. M. Peterson

Tired of waiting for your slow (aren't they all?!) printer to output that long listing, letter, or whatever; so you can get back to playing STAR DREK? Well, why not let that "intelligent" peripheral (the 2040) help you out? In IBM-speak it's called 'spooling': you send the print file out to some lesser entity (in this case, the Commodore 2040 floppy disk drive) and let the big-shot CPU get on with the important stuff--like balancing your checkbook(?).

The idea is very simple, just tell the printer to 'listen' and the 2040 to 'talk,' and stand back so they can do their respective things. Of course, there are a couple preliminaries: you have to set up a disk file containing the image of the file you want printed and then tell the 2040 that's the file it's supposed to 'talk.' The first part is very easy--let's take a program listing, for example: You probably now do something like this

```
OPEN 4,4: CMD 4: LIST
PRINT#4,;: CLOSE 4
```

The only change necessary would be to open the output file to the disk with an appropriate filename, e.g.:

```
OPEN 4,8,8,"0:◄any name►,S,W": CMD 4: LIST
PRINT#4,;:CLOSE 4
```

and, in this case, DON'T FORGET TO CLOSE THE FILE or you may lose it entirely!

The next part is to tell the 2040 you want to access this file:

```
OPEN 4,8,8, "◄whatever name you used►,S,R"
```

Now you're ready for 'no hands' hardcopy! First, you tell the 2040 to 'talk' the file you've just opened:

```
POKE 165,72: SYS 61668: POKE 165,104: SYS 61668
```

To explain the above line: 165 is the address of the IEEE bus output character buffer. 72 (64 + 8) is the 'talk' address for device 8 (That's the 2040, unless you've changed the address jumpers). 104 (96 + 8) is the secondary address (or channel) corresponding to the second '8' in the last OPEN statement above. And 61668 is the beginning of the ROM routine which outputs a command to the IEEE bus. This routine conveniently leaves the ATN line on the bus down, so the 2040 won't start 'talking' just yet.

Next you tell the printer (device #4 in this example) to 'listen' and you make the PET forget it was supposed to be sending its output to the IEEE bus:

```
OPEN 5,4: CMD 5: POKE 176,3: POKE 174,0
```

At this point the 2040 will start 'talking': the printer, printing; and the PET will be 'READY.' for whatever you want--except you can't use the IEEE bus! Therefore, unless you're going to load from cassette (YUK!), you'd better LOAD whatever disk program file you need before executing the last two lines of 'direct' code above.

Finally, when the clatter over at the printer stops, you can close the open disk file (and turn off that pesky LED) either by re-initializing the drive or with the following:

```
OPEN 1,8,8: CLOSE 1
```

Here, the important thing is to use the same secondary address as was used in the OPEN which accessed the disk file. Notice the filename is unnecessary. ©

Variable Dump For New Rom Pets

Frank R Levinson V M.D

The following routine lists all defined basic variables and gives their current values. It can be used after execution of a Basic program has been halted with the Stop key, a "Stop" command, or "End". "Continue" will resume execution. The routine lists simple variables only, no arrays.

Since PET Basic only uses the first two letters of variable names, these are the only ones used by the routine. The existing variables are output in alphabetical order, with single letter names preceeding all double keystroke names. Hence, the order of output is: A, B, ... Z, A0, A1, ... A9, AA, ... AZ, B0, ... BZ, ... ZZ. Floating point variables are listed first. Then interger variables (%) are listed and finally string variables ($). The Stop key is functional during the listing, and may be needed to prevent the strings from scrolling other data off the screen.

The routine sits in the second cassette buffer. Enter it using the Machine Language Monitor by listing memory addresses 0338-03C7 and then changing the values of 033A-03C2 as shown in Figure 1. Save the routine for future use with the following MLM command: S"VARIABLE DUMP",01,033A,03C3

The routine should be loaded before your Basic program or else note and reset the end of basic program pointers at 42 and 43 decimal after loading. Follow this with "Clear"

The routine can be relocated anywhere in memory with one alteration. The location

MICRO COMPUTER INDUSTRIES, LTD.

WORDCHECK

WordCheck is the secretary's lifesaver!

Our newest and already one of our fastest selling programs.

This program interacts with WordPro 3 or 4*. Run your letters and documents through WordCheck it checks EVERY SINGLE WORD for spelling or typographical errors.

WordCheck contains a spelling list of most commonly used words. Any words that do not match this list will show up on your screen. If these flagged words are all right pass them by with the pressing of a single key or AUTOMATICALLY add them to the spelling list without having to retype them.

WordCheck is ideal for doctors, lawyers and anyone else doing technical writing. WordCheck is so simple to learn to use your secretary can be working with it in a matter of minutes.

Your worries are over! No more scrambling for the dictionary when you have to write "fluorescent", "nucleotide" or "receive". WordCheck does the work for you quickly, thoroughly and accurately.

Available for CBM and PET 32K** machines with dual disk drives. List price is only $200.00.

* Word Pro is a registered trademark of Professional Software Inc. and Pro-Micro Software Ltd.
** CBM and PET are registered trademarks of Commodore Business Machines.

INVENTORY CONTROL

Disk based for CBM or PET 32K
Inventory, Point of Sale, Accounts Receivable

Inventory a minimum of 2000 items per diskette (a lot more with the 8050 Disk Drive.)

Complete records of merchandise purchased and sold. Update files and supply cost values of stocked items.

Update cash and credit sales, write invoices, remove sales from inventory and keep running total of sales tax. Cash sales and credit sales.

Maintain a complete record of items charged, payments on account, print bills, sort fiels and print out summary reports.

List price is only $200.00.

MICRO COMPUTER INDUSTRIES, LTD.

1520 East Mulberry Suite 170
Fort Collins, Colorado 80524
1-303-221-1955

CREATE-A-BASE

This data base management program for CBM or PET 32K handles most business data processing chores with one program. No computer experience required. Just turn it on and go!

EXTREMELY FLEXIBLE FEATURES

Create records with up to 24 fields of data of your choosing.
File up to 650 records on each floppy diskette (1800 if you own the 8050 Disk Drive.)
Change or add fields at any time.
Change data disks with out dumping operating program.
Sort or search by any one or two fields.
Data can be added as $ amounts, with right hand justification.
Perform arithmetic operations on fields with $ amounts (+,-,*,/).
Merge files, change or scratch records, output mailing labels.
Completely interactive with WordPro 3 or 4*, output form letters, mailing lists, accounts receivable, invoices, statements, inventories, even reports on your favorite fishing holes (and have more time to go fishing too.)

Start the new year off right with Create-A-Base. You'll pat yourself on the back for months.

Create-A-Base runs on CBM 8032 or 2001 32K machines.

Available on disk only
Price $200.00 For the 8050 Disk Drive $300.00

PET-TERM
ONLINE TERMINAL SOFTWARE FOR THE 8010 OR TNW MODEM

Proven tested software for the PET 8010 MODEM

Machine language routines for speedy performance.

All necessary screen and keyboard character conversions. Control key and special key functions.

Terminal to Disk Storage.
Sequential or Program file transmission capability.

Return to BASIC at will. Operate Half or Full Duplex.

EXTRAS

Support programs, such as a SEQ FILE READ/EDIT/PRINTER which allows you and your customers to read, edit and printout those data files you will be receiving from the SOURCE and other such data bases.

This program and complete operating documentation lists for only $35.00.

pointed to by the instruction at $0341 (currently BD B7 03) must point to the $22 near the end of the program (currently at $03B7). Change the B7 and 03 to point to the new location of this $22. (Caution, there are two $22s near the end of the program. Get the correct one.) Call the routine at its new location by SYSing to the first byte of the relocated program.

Figure 1. HEX DUMP OF ROUTINE TO DUMP BASIC VARIABLES

```
0338: -- -- A5 13 48 A0 20 A2
0340: 0B BD B7 03 9D 20 02 CA
0348: 10 F7 8C 29 02 C0 20 F0
0350: 1C 8C 24 02 D0 17 8E 22
0358: 02 AD 12 E8 C9 EF D0 08
0360: 68 85 13 68 68 4C 89 C3
0368: A2 30 8E 23 02 A9 20 85
0370: 77 A9 00 85 45 20 84 CD
0378: A5 45 F0 13 A9 20 85 77
0380: AD 22 02 8D 27 02 AD 23
0388: 02 8D 28 02 20 A8 C9 AE
0390: 23 02 E8 E0 3A 90 D3 E0
0398: 41 90 F7 E0 5B 90 CB AE
03A0: 22 02 E8 E0 41 90 FB E0
03A8: 5B 90 AB A0 24 CC 29 02
03B0: F0 AE 90 8B C8 D0 88 22*
03B8: 20 20 41 92 3D 22 20 41
03C0: 20 3B 00
```

*Asterisk denotes the $22 that is referenced by the instruction at location $0341 ©

The 32K Bug

Earl H. Wuchter

There is something about the 32K PET that makes it different from any other size PET, and if you are not aware of this difference, the graphic program you have developed and debugged on a smaller machine may not run correctly in the big one.

The difference is subtle, but quite important. With 32K of RAM, the upper end of memory available to BASIC (where it stores string variables) is immediately followed by the lower end of the screen RAM (top of the screen). Any lesser amount of RAM provides a dead area adjacent to the screen.

If your program is to run correctly in 32K you must pay strict attention to the screen boundary and not allow any characters to spill over into the strings.

Consider, for example, a program that is going to create an explosion effect with characters radiating upward from a computed position (variable M). Characters are to be poked to locations M-40, M-39, M-41, M-80, M-78, M-82, etc. You would certainly make tests to ensure that location M is on the screen, but you might not want to slow the graphic effect by testing all of the other locations. When location M is near the top of the screen, some of the characters will go out of bounds, and will not be displayed. In the 32K PET, they may show up later as part of other strings.

If this condition is troubling an existing program, cure it by lowering the top of available memory. Begin the program with IF PEEK(53) = 128 THEN POKE 53,127 :CLR ©

An 'Ideal' Machine Language Save For The Pet

Arthur C Hudson

To paraphrase Peter Ustinov: *there are some things that can be done, and some things that can not be done. There are even ways of doing the things that can not be done.*

There is a way to partition the PET, so that it thinks that it is two (or more) computers. Among the things that we can do with this technique is to write a Basic program which efficiently incorporates and saves its own machine language.

Let us start immediately with an example of what to do, leaving the explanation until later. If you want to follow this procedure on your PET, use exactly the order given.

Suppose that we need space for 239 or less bytes of ML (machine language). Start with a cold or reset PET, and enter, using no spaces:

10POKE041,5:RUN (This is for upgrade ROM, if you have old ROM use 123 in place of 041.)

Now examine the contents of the SB (start-of-basic) pointer. If you have upgrade ROM this pointer is at bins $0028,0029 (40,41 in decimal). On the old ROM it is at bins $007A,007B (122,123). You will find that this pointer contains 01,04; in other words it points to bin $0401, the second bin in page 4. (In decimal, 1025 = 1 + 4*256). Using a direct statement, or the resident monitor if you have it, change the pointer to point one full page higher, i.e. change it to read 01,05. Also change the three immediately following pointers to page 5 instead of page 4.

When the PET was first turned on (but not now) the three bins starting at $0400 each contained zero. We must now poke or load a similar group of three zeros in higher RAM at bins which are higher than $0400 by exactly the amount we have incremented the SB pointer. In the example this means insert three zeros starting at bin $0500 (1280 in decimal).

ML routines may now be loaded anywhere between bins $0410 and $04FF. For our example we borrow Raynor Taylor's 'Tiny' (PET User Notes V. 1. No. 4). Starting at bin $0410 (1040) we load the following:

A2 00 8A 9D 50 81 E8 D0 F9 60

Now enter any basic statements you wish, you may even re-use 10 as a line number. Include at least one SYS1040 to access the ML. Here is a short suggested program which demonstrates that For Loops, String Arrays etc. and also the machine language function normally:

```
10 A$ = "PARTITIONED BY A. HUDSON 8 SEPT. '80
20 DIML$(35)
30 FORI = 1TO35:L$(I) = LEFT$(A$,I):NEXT
40 FORI = 1TO35:?L$(I):NEXT
50 SYS1040:END
999 POKE41,4:SAVE "PARTITION DEMO
```

Now just key in RUN and stand well back!

All that remains now is the Save procedure. The only departure from the usual Basic Save is that we must first change the SB pointer back to its normal value which is 01,04. In the above program we do this automatically with RUN999. In a fully developed program (if there is such a thing) statement 999 could be omitted.

One word of caution: RAM should contain three zeros starting at $040D. If you have inadvertently disturbed these, put them back before the Save.

Comments

What we have done is partition the PET into two parts. When the SP pointer points to $0401 we are in partition 1. The only Basic we permit here is the simple Poke and Run. Do not try to change this in any way. You can list immediately after loading and you will see this statement. You may admire it, but don't change it!

Partition 1 extends from the fourth bin beyond the last bin used by the Poke and Run to the last bin *but one* before the bin pointed to by the modified SB pointer. This last bin must contain a zero. We could dispense with the Poke and Run, but then we would have to poke the SB pointer manually after each load. ML may be loaded and altered at will, when the pointer points to partition 1, but Basic can not.

Our schizophrenic PET is in a much more amenable mood in partition 2, here it can be regarded as completely normal - we can fiddle with the Basic to our heart's content, just remembering to poke back to partition 1 before saving. Basic here can reach into partition 1 to access any ML programs which reside there.

Now what about the size of partition 1. To increase it by 256 bytes, just up the pointer high addresses to 6 instead of 5 which was used in the example. This can be continued until you run out of memory. To allot a smaller ML space the low order addresses of the pointers must be augmented by a constant, but the same principle applies. If you are very short of memory and want to tailor partition 1 to exactly the amount of ML needed, both parts of the pointers may have to be changed and a double Poke and Run used. This is trickier, but there is no real barrier.

You will almost certainly get into trouble if you try to change horses in mid stream, that is if you try to change the size of the partition after you have started to program. The best way to stay out of trouble is to decide initially that 239 bytes, or 239 plus some multiple of 256 is adequate. Then you never need to touch the low order parts of the four pointers.

With care, a tape made for old ROM can be loaded on a PET using new ROM, but considerable pointer modification is necessary. In the Poke and Run, I used 041 rather than 41 so that the statement is the same length for old and new ROM. ©

PET Metronome

Elizabeth Deal

The Pet can double as the most expensive metronome in the world at no additional cost other than typing seven lines of code. Register 59468 (the same one that controls graphic or lower case mode) is used to produce clicking sounds which are heard through a speaker-amplifier connected to the user port. The usual "music" connection is required via pins M and N of that port.

The instructions for using the gadget are in the REM lines. The program begins clicking at the rate of 120 beats per minute. The rate can be adjusted quickly or slowly by pressing the appropriate keys.

```
100 J=3600:T=360:B=30:R=120:P=1:
       ¬M=59468:L=236:N=204:D=12:G=5:
       ¬C=255:Q=64:H=10-P
110 K=151:S=152:A=0:V=12:TI$="000000
       ¬"
120 X=TI:PRINT"ħ"R;TI$:Z=PEEK(K):
       ¬F=PEEK(S):I=P+F*H:IFZ=QTHENPOK
       ¬EM,V:END
130 R=R-I*(Z=D)+I*(Z=G):R=R+(R-T)*(R
       ¬>T)+(R-B)*(R<B):IFZ<>CORF<>AGO
       ¬TO120
140 POKEM,N:POKEM,L:W=J/R
150 IF(TI-X)<WGOTO150
160 GOTO120
170 REM==============================
       ¬==
180 :             PET METRONOME
190 :ELIZABETH DEAL, MALVERN,
       ¬ PA 19355
200 REM==============================
       ¬==
210 RATE IS SHOWN IN BEATS PER ¬
       ¬MINUTE
220 < > FOR SLOW CHANGE IN RATE
230 SHIFTED < > FOR FAST CHANGE
240 Q TO QUIT & RESET REGISTER
250 T AND B GIVE RANGE OF RATE, B>0
260 MAX RESOLUTION 7-8 JIFFIES
270 MAX ERROR 0.6 JIFFIE/BEAT,
       ¬I THINK.
280 :   (LINE 160 IS NOT TIMED)
290 FOR USE AN ACCURATE LONG TERM
300 COUNTER SET RATE 1% LOWER THAN
310 DESIRED.
320 :   REL.2 PETS - K=515, S=516
```

©

The IEEE Bus - Standing Room Only?

Jim Butterfield

Before we discuss problems on the IEEE bus, let's take a quick overview and say some good things.

First, the basic bus structure is sound. It can handle many devices, and is capable of transferring data with full handshake at a pretty good speed. It is asynchronous, which means it will adapt to whatever speed the connected devices need.

There's a myth going around that the bus is slow: not true. With a typical operating speed of over 5000 bytes per second, it can transmit a full 64K in about 13 seconds - not bad for full handshaking. There are certain ways that the bus can be used, for example with Basic GET# statements, that will indeed be slow; but you can't blame the IEEE bus for that.

The real problems, as I see them, are in the way that Commodore uses the IEEE bus. There's a lot of potential that is being thrown away.

Lockout

The current Commodore disk units and printers lock out the bus when they are busy. If the disk is doing a NEW/HEADER or a VERIFY/COLLECT it will hold up everything else on the bus. Similarly, if the printer is moving to the next page, or even just printing a line, the bus is out of business.

This is fair enough if you wanted to use the device that is busy. You can't write on the disk until it has finished its last job, and it's reasonable enough to wait for the printer to stop moving before you give it new stuff. But if the disk is busy, you can't use the printer; or if the printer is busy, you can't use the 8010 modem ... this kind of thing can hurt.

Item: a software vendor buys four 2040 disk units, strapping the device numbers to allow their simultaneous use. He wants to NEW four blank disks, one on each unit - and discovers to his amazement that he can't start a NEW on unit two until one is finished and releases the bus.

Item: a hobbyist wants to use an 8010 modem together with a printer as a printing communications terminal. He finds that he can't use the modem during the time that the printer is operating, and he loses characters from the transmission line during the time that the bus is locked out.

The bus is not the problem: it's the way that it is being used. Commodore have taken the old-style data processing approach that only one device should be working at a time. This doesn't hurt in many data processing situations, especially if you have only a disk and printer connected. But a more sophisticated system will start to hurt when lockouts start to interfere with real-time sensing or communications applications.

Multiple processors

It should be possible for several PETs to work a single disk unit, or for that matter, a whole set of common peripherals. As small computers find use in larger commercial environments or in schools, there's a need for several operators to read from and write to a common data base.

This is now being done with commercially available interfaces, such as the MUPET, but Commodore didn't make the job easy.

The biggest problem is that ROM systems prior to 4.0 would flash the EOI control line every time the screen scrolled on the PET. This would break up anything else that might be happening on the bus. This problem is now eliminated on 4.0 and (presumably) subsequent ROMs.

Of course, without a commercial interface system, you also have to make sure that two operators don't attempt to use the bus at the same time. The usual practice is to have the operator call to the others, "Everybody hold it for a moment - I want to read disk!" Informal, but workable.

The situation is improving, and commercially available devices make this possible.

Spooling

It's almost possible to tell the printer to LISTEN and the disk to TALK and have information transferred directly between the two devices. Almost, but not quite. Commodore didn't quite get the logic precise enough to do the job.

It would be a tricky job in any case. PET would need to start things up and then stay away from the bus until the transfer was complete, at which time PET would jump back on and close down the devices. Even so, it would be nice to think that the capability was available if needed.

Summary

The IEEE 488 bus is really pretty good as it is. I can't help wishing, though, that a few extra features had been provided to allow development of the real potential that is there.

Maybe when the next round of Commodore devices is announced ...? ©

PET/CBM IEEE Bus Error

Gary R Huckell

There is an error in the PET/CBM IEEE Input/output routines that can cause an inactive device on the IEEE bus to randomly become enabled. The problem exists in both old and new ROM PETs, but has now been corrected in the new 80 column units (BASIC 4.0).

The problem normally occurs when reading from the disk and can result in the transfer of part of the file being read to a printer that is also on the bus, in a slowing of the transfer of data, or in an error that causes the transfer to be aborted. It appears that Commodore peripheral equipment is immune to the problem. Unfortunately, non-Commodore equipment can often cause trouble.

The error is caused by an improper control handshake when the PET/CBM signals a device to UNTALK. After the PET/CBM has made a device on the IEEE bus a talker, it then assumes the role of a LISTENER. While listening, it controls the reading of data by raising and lowering signals NRFD and NDAC. After reading a character (GET#) or a string of characters (INPUT#) these control signals are both low to indicate that no more data is requested. The PET/CBM must now switch its role to that of a controller and UNTALK the device. The proper procedure in switching to the controller role is to first drive the signal ATN low, and then raise signals NRFD and NDAC. The PET/CBM first raises signals NRFD and NDAC, and then drives ATN low. This causes two problems:

First, when the signal NRFD is raised, the device that is currently a TALKER assumes that the PET/CBM is ready for another character, so it places another byte of data on the bus and lowers DAV. The PET/CBM then raises signal NDAC which normally indicates that the data has been accepted. Many IEEE devices lose a character because of this error.

The second problem occurs after raising NRFD and NDAC, when the PET drives the ATN signal low. As soon as ATN is low, all devices on the bus prepare for a command from the controller by raising their NRFD signals. If the device to be UNTALKED is slow in responding to changes to the IEEE interface, as the CBM disk is, there is a short time period where the controller is driving ATN and the device to be UNTALKED has data on the bus. Other devices on the bus now see the ATN signal and the data, and may read the current data on the bus as a control word. If the data on the bus happens to be a LISTEN command, an unwanted device can become a listener.

Any time data is input from the disk there is the possibility that another device on the bus may become a LISTENER. If this device is not always ready to receive data, it can then slow or stop the transfer of data to the PET/CBM. LOADing a program from the disk never causes the problem since the disk is only UNTALKED once, at the end of the LOAD. For the same reason, using INPUT# causes less trouble than GET#. Even though the problem is very easy to demonstrate, many users never see the problem. TNW's interfaces work with the Commodore Word Processor without problems, probably because this program reads a disk file as one block.

Since the UNTALK error looks so much like a proper data read operation, it may be hard to modify an IEEE device that is subject to this problem. We have a fix for our modem (the TNW488/103) that seems to work. We know our serial interfaces have the problem but we have received only a couple of complaints, fewer then we would have expected. Though it is hard to correct the problem by modifying the hardware, it is easy to program the PET around the problem. Two software corrections are shown here, one to be used with a machine language program and the second that can be used in a BASIC program.

Machine Language Fix

OPEN I/O files in the usual way in BASIC. Then the following routines permit use of the PET/CBM's IEEE-488 bus from programs written in 6502 machine language:

FFC6: LDX with the file number of an open input file, JSR FFC6 makes the device a TALKER.

FFC9: LDX with the file number of an open output file, then JSR FFC9 makes the device a LISTENER.

FFCC: JSR FFCC issues UNLISTEN and UNTALK commands on the IEEE-488 bus, and restores I/O to the keyboard and screen. This is the routine that contains the error. It also "clobbers" the accumulator A.

FFD2: LDA with a byte, then JSR FFD2 to "PRINT#" the character to the current output file.

FFE4: JSR FFE4 "GET#s" a character from the current input file, leaves it in the accumulator A.

The only change required to correct the UNTALK error is to set the ATN signal before calling the UNTALK routine. Do not set ATN before calling the same routine to perform an UNLISTEN.

```
              ; UNTALK
LDA E840      ; READ IEEE SIGNALS
AND #$FB      ;SET ATN
STA E840      ;WRITE IEEE SIGNALS
JSR FFCC      ;UNTALK
```

BASIC Fix

The following routine will store a machine language program in the second cassette buffer that can then be called using the SYS statement to GET# a byte.

```
10 FOR I = 826TO847:READ D:POKE I,D:NEXT
20 DATA 162,0,32,198,255,32,228,255,133,1,173
30 DATA 64,232,41,251,141,64,232,32,204,255,96
```

OPEN I/O files in the usual way, then POKE 827 with the file number of the input device. Instead of using the GET# statement to read a byte of data, use SYS(826). The machine language program will read a single data byte and store it in memory cell 0001 where it can be read using PEEK(1). Output normally using the PRINT# statement. The following program will read a file from the disk and list it to the screen.

```
40 OPEN1,8,2,"FILE 1,SEQ,READ"
50 POKE 827,1
60 SYS 826:D = PEEK(1):IF ST = 0 THEN PRINT
    CHR$(D);:GOTO 60
70 CLOSE 1:END
```

Commodore has now fixed this problem in the new 80 column PETs, so the above software corrections are not necessary with these units, but they should not cause a problem. All software written by TNW Corporation that uses the CBM disk incorporates the above software fix. We would like to hear from other PET/CBM users that have encountered this problem and hope that this fix is of value. ©

The Single-Board 6502

Eric Rehnke

High-Speed Data Transfer

Necessity is INDEED the mother of invention.

For quite some time I've thought about how neat it would be to have some way of transferring data at high speed between two computers. But, as usual, there was always something "more important" to do.

Recently, the need arose to have such a high-speed data transfer system.

As newsletter editor for INTERACTIVE (a newsletter published by Rockwell for the AIM 65), I frequently need to print AIM 65 program listings.

Now the AIM is a great little machine, and the on-board thermal printer is very convenient but a 20 column wide assembly language or BASIC listing just doesn't cut it for publication.

Hooking my Decwriter up to the AIM wouldn't solve the problem because AIM's ROM assembler still formats the outut for a 20 column wide printout.

Clearly, the only practical solution was to somehow move the source code over to my KIM system and assemble it with the HDE assembler.

Fortunately, except for the fact that AIM 65 text editor doesn't use line numbers, the source code is completely compatible between the two machines. (That's because both assemblers have the same origin.)

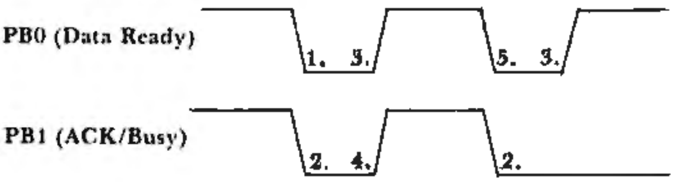
The software I'm presenting is a version which dumps object code from either the AIM, SYM or 6522 equipped APPLE to my KIM.

I'm not providing the source file transfer program because I've still got some bugs in it. (Maybe I'll print that routine some other time.)

One of the fastest and, perhaps, even the simplest method of transferring data from one computer to another is to do it in parallel. Each computer needs an 8-bit I/O port and several "handshaking" lines for signaling "data sent" and "data received". All of my systems have a user accessible I/O port (I recently installed a 6522 VIA in my Apple II) so all that I needed to do was hook up the lines and write the software. (It always turns out to be "easier said than done", however.)

The first problem turned out to be figuring out the proper "handshaking" sequence. I first looked at the popular "Centronics" style handshaking sequence but decided to simplify it down to two lines (instead of three).

Handshaking Sequence



XMTR starts first

1. XMTR initializes 'Data Ready' low and waits for the RCVR line 'Acknowledge/Busy' to go low.

1. RCVR initializes 'ACK/Busy' low and waits for the 'Data Ready' line to go high indicating that there is a BYTE available on the lines.

3. XMTR puts a data BYTE on the lines, sets the 'Data Ready' line high and waits for the RCVR 'ACK/Busy' line to go high signifying that the data has been received.

4. RCVR accepts a data BYTE and sets the 'ACK/Busy' high

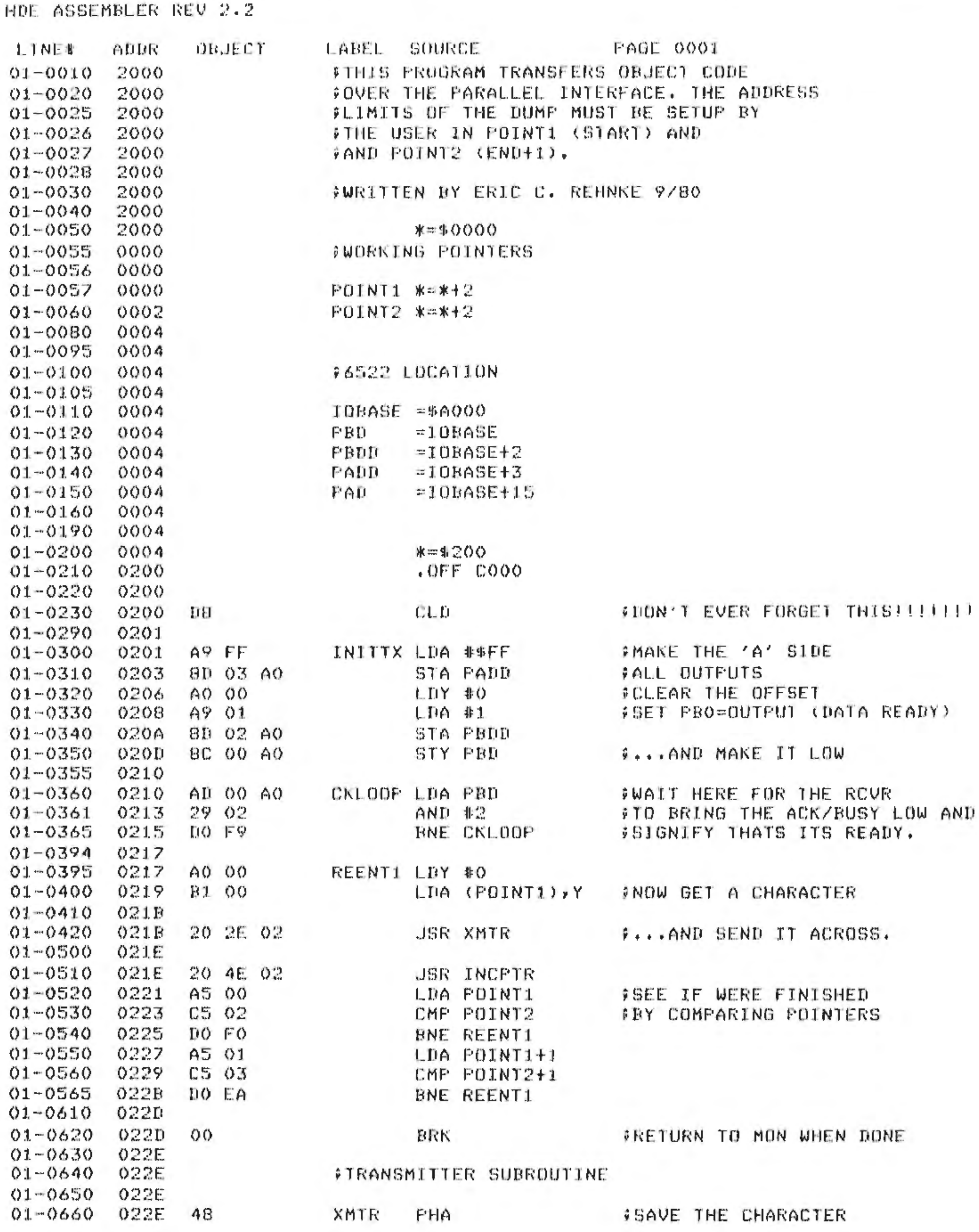
5. XMTR sets 'Data Ready' low after 'Ack/Busy' goes high

If I had to do it all over, I would have added a third line to indicate that the byte on the lines was the last byte to be transferred. This would be better for transferring binary dumps since, in that mode, with only two handshake lines, the receiver has no way of knowing when the data transfer in completed and must be RESET to get it out of an infinite loop.

The neat handshaking modes available in the 6522 on the AIM weren't used because I wanted to be able to use the same software for both the KIM and the AIM and those special I/O operating modes aren't available on KIM since it uses a 6530 for its user I/O. (Although the example software is only used to send data one way-- from AIM to KIM, it has been used to send data the other way also).

As far as the hardware connection goes--simply hook PA0-PA7 on the KIM to PA0-PA7 on the AIM (PA0 to PA0, PA1 to PA1 etc), PB0-PB1 on the KIM to PB0-PB1 on the AIM, and then tie the system grounds together. That's not too difficult, is it?

IMPORTANT NOTE: *Both systems must be reset* to put the I/O lines in a known state (all lines go "high" after a system reset). The order in which the programs are started is also important. The transmit program must be started first, then the receive program.



```
HDE ASSEMBLER REV 2.2

 LINE#    ADDR   OBJECT      LABEL  SOURCE             PAGE 0001
01-0010  2000               ;THIS PROGRAM TRANSFERS OBJECT CODE
01-0020  2000               ;OVER THE PARALLEL INTERFACE. THE ADDRESS
01-0025  2000               ;LIMITS OF THE DUMP MUST BE SETUP BY
01-0026  2000               ;THE USER IN POINT1 (START) AND
01-0027  2000               ;AND POINT2 (END+1).
01-0028  2000
01-0030  2000               ;WRITTEN BY ERIC C. REHNKE 9/80
01-0040  2000
01-0050  2000                      *=$0000
01-0055  0000               ;WORKING POINTERS
01-0056  0000
01-0057  0000               POINT1 *=*+2
01-0060  0002               POINT2 *=*+2
01-0080  0004
01-0095  0004
01-0100  0004               ;6522 LOCATION
01-0105  0004
01-0110  0004               IOBASE =$A000
01-0120  0004               PBD    =IOBASE
01-0130  0004               PBDD   =IOBASE+2
01-0140  0004               PADD   =IOBASE+3
01-0150  0004               PAD    =IOBASE+15
01-0160  0004
01-0190  0004
01-0200  0004                      *=$200
01-0210  0200                      .OFF C000
01-0220  0200
01-0230  0200  D8                  CLD                ;DON'T EVER FORGET THIS!!!!!!!
01-0290  0201
01-0300  0201  A9 FF        INITTX LDA #$FF           ;MAKE THE 'A' SIDE
01-0310  0203  8D 03 A0            STA PADD           ;ALL OUTPUTS
01-0320  0206  A0 00               LDY #0             ;CLEAR THE OFFSET
01-0330  0208  A9 01               LDA #1             ;SET PB0=OUTPUT (DATA READY)
01-0340  020A  8D 02 A0            STA PBDD
01-0350  020D  8C 00 A0            STY PBD            ;...AND MAKE IT LOW
01-0355  0210
01-0360  0210  AD 00 A0     CKLOOP LDA PBD            ;WAIT HERE FOR THE RCVR
01-0361  0213  29 02               AND #2             ;TO BRING THE ACK/BUSY LOW AND
01-0365  0215  D0 F9               BNE CKLOOP         ;SIGNIFY THATS ITS READY.
01-0394  0217
01-0395  0217  A0 00        REENT1 LDY #0
01-0400  0219  B1 00               LDA (POINT1),Y     ;NOW GET A CHARACTER
01-0410  021B
01-0420  021B  20 2E 02            JSR XMTR           ;...AND SEND IT ACROSS.
01-0500  021E
01-0510  021E  20 4E 02            JSR INCPTR
01-0520  0221  A5 00               LDA POINT1         ;SEE IF WERE FINISHED
01-0530  0223  C5 02               CMP POINT2         ;BY COMPARING POINTERS
01-0540  0225  D0 F0               BNE REENT1
01-0550  0227  A5 01               LDA POINT1+1
01-0560  0229  C5 03               CMP POINT2+1
01-0565  022B  D0 EA               BNE REENT1
01-0610  022D
01-0620  022D  00                  BRK                ;RETURN TO MON WHEN DONE
01-0630  022E
01-0640  022E               ;TRANSMITTER SUBROUTINE
01-0650  022E
01-0660  022E  48           XMTR   PHA                ;SAVE THE CHARACTER
```

```
01-0670  022F  48              PHA          ;TWICE
01-0680  0230  AD 00 A0   ACKLP1 LDA PBD    ;WAIT TIL 'ACK/BUSY' IS LOW
01-0690  0233  29 02             AND #2
01-0700  0235  D0 F9             BNE ACKLP1
01-0710  0237
01-0720  0237  68                PLA        ;RECOVER DATA
01-0730  0238  8D 0F A0          STA PAD
01-0740  023B  A9 01             LDA #1     ;RAISE 'DATA READY' HIGH
01-0750  023D  8D 00 A0          STA PBD
01-0760  0240
01-0770  0240  AD 00 A0   ACKLP2 LDA PBD    ;WAIT TIL 'ACK/BUSY' IS HIGH
01-0780  0243  29 02             AND #2
01-0790  0245  F0 F9             BEQ ACKLP2
01-0800  0247
01-0810  0247  A9 00             LDA #0     ;NOW DROP THE 'DATA READY' LINE
01-0820  0249  8D 00 A0          STA PBD
01-0830  024C  68                PLA        ;RECOVER CHAR FOR CR TEST
01-0840  024D  60                RTS
01-0850  024E
01-0860  024E             ;HERE WE INCREMENT POINT1
01-0870  024E
01-0880  024E  E6 00      INCPTR INC POINT1
01-0890  0250  D0 02             BNE EXIT
01-0900  0252  E6 01             INC POINT1+1
01-0910  0254  60         EXIT   RTS
01-0920  0255
01-0940  0255
01-0950  0255
01-0975  0255                    .END
```

```
HDE ASSEMBLER REV 2.2

 LINE#   ADDR   OBJECT    LABEL  SOURCE           PAGE 0001

01-0010  2000             ;THIS PROGRAM RECEIVES OBJECT CODE FILES
01-0020  2000             ;OVER THE PARALLEL INTERFACE AND STORES
01-0030  2000             ;THE DATA STARTING AT THE LOCATION
01-0040  2000             ;INDICATED BY THE POINTER AT $0000.
01-0050  2000             ;THIS POINTER MUST BE INITIALIZED BY THE USER.
01-0055  2000
01-0060  2000             ;WRITTEN BY ERIC C. REHNKE 9/80
01-0070  2000
01-0080  2000                    *=$0000
01-0090  0000             POINT1 *=*+2
01-0100  0002
01-0110  0002             ;6530 LOCATION
01-0115  0002
01-0120  0002             IOBASE =$1700
01-0130  0002             PBD    =IOBASE+2
01-0140  0002             PBDD   =IOBASE+3
01-0150  0002             PADD   =IOBASE+1
01-0160  0002             PAD    =IOBASE
01-0170  0002
01-0190  0002
01-0200  0002                    *=$2000
01-0210  2000
01-0220  2000
01-0230  2000
01-0240  2000
01-0250  2000
01-0251  2000  D8                CLD        ;DON'T EVER FORGET THIS!!!!!
01-0260  2001  A9 00      INITRX LDA #0     ;MAKE THE 'A' SIDE ALL INPUTS
01-0270  2003  8D 01 17          STA PADD
01-0280  2006  A0 00             LDY #0     ;CLEAR THE OFFSET
01-0290  2008  A9 02             LDA #2
01-0300  200A  8D 03 17          STA PBDD   ;SET PB1=OUTPUT (ACK/BUSY)
```

HDE inc.

BOX 120
ALLAMUCHY, N.J. 07820
201-362-6574

HUDSON DIGITAL ELECTRONICS INC.

THE TASK* MASTERS

HDE supports the *TIM, AIM, SYM and KIM (TASK) with a growing line of computer programs and peripheral components. All HDE component boards are state-of-the-art 4½" x 6½", with on board regulation of all required voltages, fully compatible with the KIM-4 bus.

OMNIDISK 65/8 and 65/5
Single and dual drive 8" and 5¼" disk systems. Complete, ready to plug in, bootstrap and run. Include HDE's proprietary operating system, FODS (File Oriented Disk System). From $795.00.

DM816-M8A
An 8K static RAM board tested for a minimum of 100 hours and warranted for a full 6 months. $195.00

DM816-UB1
A prototyping card with on-board 5V regulator and address selection. You add the application. $49.50

DM816-P8
A 4/8K EPROM card for 2708 or 2716 circuits. On board regulation of all required voltages. Supplied without EPROMS. $165.00

DM816-CC15
A 15 position motherboard mounted in a 19" RETMA standard card cage, with power supply. KIM, AIM and SYM versions. $545.00

DISK PROGRAM LIBRARY
Offers exchange of user contributed routines and programs for HDE Disk Systems. Contact Progressive Computer Software, Inc. for details.

HDE DISK BASIC
A full range disk BASIC for KIM based systems. Includes PRINT USING, IF . . . THEN . . . ELSE. Sequential and random file access and much more. $175.00

HDE ADVANCED INTERACTIVE DISASSEMBLER (AID)
Two pass disassembler assigns labels and constructs source files for any object program. Saves multiple files to disk. TIM, AIM, SYM, KIM versions. $95.00

HDE ASSEMBLER
Advanced, two pass assembler with standard mnemonics. KIM, TIM, SYM and KIM cassette versions. $75.00 ($80.00 cassette)

HDE TEXT OUTPUT PROCESSING SYSTEM (TOPS)
A comprehensive text processor with over 30 commands to format and output letters, documents, manuscripts. KIM, TIM and KIM cassette versions. $135.00 ($142.50 cassette)

HDE DYNAMIC DEBUGGING TOOL (DDT)
Built in assembler/disassembler with program controlled single step and dynamic breakpoint entry/deletion. TIM, AIM, SYM, KIM AND KIM cassette versions. $65.00 ($68.50 cassette)

HUE COMPREHENSIVE MEMORY TEST (CMT)
Eight separate diagnostic routines for both static and dynamic memory. TIM, AIM, SYM, KIM and KIM cassette versions. $65.00 ($68.50 cassette)

ORDER FROM THESE FINE DEALERS:

Progressive Computer Software
405 Corbin Road
York, PA 17403
(717) 845-4954

Lux Associates
20 Sunland Drive
Chico, CA 95926
(916) 343-5033

Johnson computers
Box 523
Medina, Ohio 44256
(216) 725-4560

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

Falk-Baker Associates
382 Franklin Avenue
Nutley, NJ 07110
(201) 661-2430

Laboratory Microcomputer Consultants
P.O. Box 84
East Amherst, NY 14051
(716) 689-7344

Perry Peripherals
P.O. Box 924
Miller Place, NY 11764
(516) 744-6462

```
01-0310  200D  8D 02 17          STA PBD          ;AND MAKE IT HIGH
01-0360  2010
01-0370  2010  20 4D 20   CONT   JSR INCPTR       ;BUMP THE POINTER
01-0380  2013  20 1B 20          JSR RCVR         ;GET A DATA BYTE
01-0390  2016  91 00             STA (POINT1),Y   ;STORE IT
01-0400  2018  4C 10 20          JMP CONT         ;KEEP LOOKING FOR DATA
01-0430  201B
01-0440  201B  A9 00      RCVR   LDA #0           ;DROP THE 'ACK/BUSY' LINE
01-0450  201D  8D 02 17          STA PBD
01-0460  2020
01-0470  2020  AD 02 17   DRLP1  LDA PBD          ;WAIT FOR 'DATA READY'
01-0480  2023  29 01             AND #1           ;TO GO HIGH
01-0490  2025  F0 F9             BEQ DRLP1
01-0500  2027  20 54 20          JSR DELAY
01-0510  202A  AD 02 17          LDA PBD
01-0520  202D  29 01             AND #1
01-0530  202F  F0 EF             BEQ DRLP1
01-0540  2031
01-0550  2031  AD 00 17          LDA PAD          ;GET DATA
01-0560  2034  48                PHA              ;SAVE IT
01-0570  2035
01-0580  2035  A9 02             LDA #2           ;SET 'ACK/BUSY' HIGH TO
01-0590  2037  8D 02 17          STA PBD          ;SIGNAL 'DATA RECEIVED'
01-0600  203A
01-0610  203A  AD 02 17   DRLP2  LDA PBD          ;NOW WAIT FOR 'DATA READY'
01-0620  203D  29 01             AND #1           ;TO GO LOW
01-0630  203F  D0 F9             BNE DRLP2
01-0631  2041  20 54 20          JSR DELAY
01-0632  2044  AD 02 17          LDA PBD          ;...AND THEN HIGH.
01-0633  2047  29 01             AND #1           ;THIS SAYS "DATA READY !"
01-0634  2049  D0 EF             BNE DRLP2
01-0640  204B  68                PLA              ;RECOVER DATA
01-0650  204C  60                RTS              ;AND RETURN
01-0660  204D
01-0670  204D
01-0680  204D  E6 00      INCPTR INC POINT1
01-0690  204F  D0 02             BNE EXIT
01-0700  2051  E6 01             INC POINT1+1
01-0710  2053  60         EXIT   RTS
01-0720  2054
01-0750  2054             ;THIS IS A DUMMY DELAY ROUTINE
01-0760  2054             ;THAT WAS USED FOR TESTING PURPOSES.
01-0770  2054
01-0771  2054  60         DELAY  RTS
01-0775  2055
01-0780  2055                    .END
```

Multi-Computer/Multi-User Games

No, I'm not a computer game freak. But, I am excited about the fantasy role playing games that are becoming available for computers. The intriquing Dungeons and Dragons game really grabbed my interest. Almost from the time I first become aware of it, I was toying with ways to computerize certain aspects of it. Certainly, the dice throwing part could be computerized, as well as the bookkeeping aspects of the game--like keeping track of the character attributes and whether or not certain moves are legal as well as the relatively complicated procedure of deciding how much damage has been done by certain moves. Freeing the player from having to handle all the complex paperwork should make the game all that much more enjoyable. Any game freaks out there care to comment?

As I look around the field, I don't see too much being done in the area of multi-user/multi-computer games. Computer games have been in the man-against-computer mode for quite some time and have made computer hobbyists appear almost anti-social. It's time for a change.

A fellow at work and I are working out the details for a two-player/two-computer game which uses a couple of AIM 65 computers. The first game will be rather simplistic but it will serve to get things started. Anyone out there working along the same lines? Get in touch? Let's join fantasies.

I can picture a time when many computers are linked together playing a rather complex fantasy type game, or, perhaps a realistic simulation type game.

Software Review

How would you like to develop 1802 programs on your AIM 65? Or, how would you like to set up a library of MACROS which can be called from your assembly language programs?

If either, or both of these things interests you, then you'll be interested in a new software package for the AIM 65 called MACRO.

MACRO is actually a pre-processor that works in conjunction with the AIM 65 assembler. Its function is to accept a source file that contains macro calls, expand those macros by looking them up in a library file, and outputting a new source file with all the macros expanded so that the AIM 65 ROM assembler can assemble it.

The macro library file must be set up which defines all the macros which are to be used and must be memory resident at the time the input file is submitted for expantion: (makes AIM 65 sound like a large machine, doesn't it?)

Here's an example of what it looks like:

SAMPLE MACRO

```
INCD POINTER
```

SAMPLE MACRO DEFINITION

```
& INCD
INC!1
BNE* +4
INC !1 +1
&
```

SAMPLE MACRO OUTPUT

```
INC POINTR
BNE* +4
INC POINTR +1
```

(The '&' character is used both to start and terminate a macro definition)

Now that last little programming sequence (incrementing a double byte pointer) is something 6502 programmers do alot of.

The same technique can be used to set up a cross assembler for most any other CPU (6800, 1802, 8080 etc). Pretty excitin' stuff!!!

According to the documentation that accompanies MACRO, the minimum usable system is an AIM 65 with 2K of RAM, the assembler ROM, and remote control of least one cassette deck. The price is $15 which includes documentation and a cassette of the object code. The source code for MACRO is available either on cassette or as a listing (you must specify) for an additional $30. (This would enable you to adapt MACRO to your 6502 floppy system).

So far, I haven't found any bugs in the system (I'm good at finding bugs) and it worked right the first time I tried it.

It's available from: POLAR SOLUTIONS
Box 268
Kodiak, Al. 99615

"AID" From HDE

AID (Advanced Interactive Disassembler) is a disassembler in the truest sense of the word. AID takes a machine language program as input and creates an assembly language source file as output. (Just the opposite of an assembler).

The source file includes labels and even equates for externally referenced locations. The file can then be assembled like any other source file.

Think about it. Remember all the time you spent manually building an assembler source file from a machine language program?

I can sure remember wasting lots of time getting a conventional disassembly listing, writing in labels and then typing the whole thing into a text editor file just to be able to modify a piece of software.

Since AID lets the computer do this "dirty"work, the programmer is free to spend more time doing the work that needs a bit more intelligence.

The source files can be assembled with the assembler from HDE which is compatible with the MOS Technology Cross Assembler.

More information on this exciting new software product can be obtained from Hudson Digital Electronics, POB 120, Allamuchy, N.J. 07820, (201) 362-6574. AID costs $95 and works just great.

No, I haven't made a source file from Microsoft BASIC as of yet. But, I'm sure some of you have it in mind. ©

Caveat Interruptor or Placating a Rebellious KIM Without Sacrificing RAM

Philip K. Hooper

Summary: The dialog below presents a bizarre experiment that is easy to perform with a KIM-1, creating a runaway computer which no longer responds to 'reset'. Fortunately, the use of an unusual keypad sequence will restore normal monitor control. Moreover, KIM's peculiar behaviour is shown to be a perfectly proper response to an abnormal situation.

O: If you have your interval timer set up to generate a non-maskable interrupt, i.e. a connection between PB7 and NMI (Figure 1), then you can try a strange little experiment. Game?

I: Sure, I'll give it a try. What do I do?

O: Turn on the power and set your NMI vector (17FA,B) so it points to some very long program-like cassette read (1872) without input or the 4C4C4C loop at 4C4C. You can even try just leaving it as it 'comes up', and still the experiment will probably work.

I: That's nothing new. Sometimes I DO point NMI to 1873, so I can load in consecutive files by just pressing 'ST' between loads. And sometimes I don't even bother setting it. What next?

O: Take a look into location 0170.

I: So? Looks pretty unremarkable - just another meaningless byte of hex garbage.

O: That's right. Whatever your KIM happens to drop into that location at 'power up'. Now, pretend you are reaching for the 'AD' key but miss it and press 'C' instead.

I: Come on. I thought you said we were doing some sort of wierd experiement, but this is just stu . . . Hey! What's happening? The display is gone, and I can't seem to bring it back using the keypad. Even 'ST' doesn't help. But reset will always . . . What the devil IS this, anyway? How come all I get when I press the 'RS' key is a brief flash and then nothing? Have you tricked me into ruining my computer or something? What can I do? Isn't reset always supposed to bring KIM back in a known starting state?

O: It is - and it does. Of course, you could switch the power off and back on. That would probably put things right. But suppose you had just keyed in two full pages of code and were about to save them on tape when this pathological behaviour started. Surely you don't want to lose all that code and have to key it in again! Can you regain control without losing your RAM?

I: Well I don't see how. The only control I have is from the keypad, and not a single one of the keys does me any good!

O: True. Not a *single* key will help - but *three* will.

I: Eh?

O: Hold down 'ST', momentarily press 'RS' and then ' + ', and then release 'ST'.

I: Say that again.

O: No! You just go back and read it again, from two lines above this and then do it.

I: How about that. It worked! And without losing all that imaginary RAM. You know, this could have accidentally happened to me, and with you not around I would have had to turn it off to fix it. Say, what happened, anyway?

O: Well, when you pressed the 'wrong key', you inadvertantly addressed the interval timer, at 170C*, and it responded by generating an interrupt, i.e. a signal to follow the NMI vector 'somewhere'. Naturally, unless this 'somewhere' included a routine to sample and respond to the keypad, no keys other than 'ST' or 'RS' could possibly have had any effect. However, pressing 'ST' generates another NMI. Instead of helping get KIM back, it just sends it off to wherever it went before, again. 'RS' *does* bring it back, but only long enough to summon *another interrupt.* You see, when it returns control to the monitor, the monitor immediately accesses 170C again, unless the address stored in the pointer 00FA,B has been altered meanwhile.

However, holding 'ST' down will prevent recognition of this interrupt (the one invoked by the monitor after 'RS'), while pressing ' + ' will alter the pointer** so that the monitor no longer interrogates 170C. Then, since no further interrupt is being generated, releasing 'ST' restores normal operation at this point. Now, aren't you glad you asked?

I: Well, I . . . .

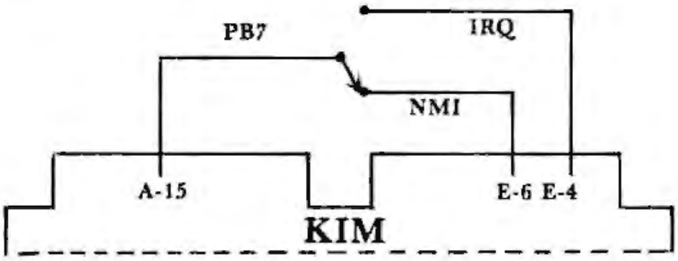
O: Say, give that ' + ' key another quick press, will ya;?

Author's Notes:

*several other addresses produce the same effect as 170C

**the hex keys, '0' - 'F', and also 'PC', alter the pointer as well and may be used in place of the ' + ' key

Figure 1 - Enabling the Timer Interrupt



Although a single wire between A-15 and E-6 is sufficient for the experiment explained above, a SPDT switch permitting the selection of either NMI or IRQ provides for more varied use of the timer interrupt. ©

COMMUNICATION!

Dann McCreary

While both the 1802 and the 6502 can handle quite a bit on their own, each has features which suit it to certain functions. Though the 1802 is not particularly fast, it has the advantages of low power consumption and low parts count needed to make a compact, portable system. On the other hand, the 6502 has the speed and software support for use as a powerful general purpose computer. Let's take a look at some ways to start a dialogue between an 1802 and a 6502.

Consider with me a few possible uses and layouts of COSMAC systems in communication with a central 6502 processor. One situation is the use of an 1802 to gather data from a remote location. The data would typically be transmitted to the main computer over a serial data link. This could take the form of a twisted wire pair, a radio transmission, a modulated light beam, telephone lines or even an intermediary like magnetic tape.

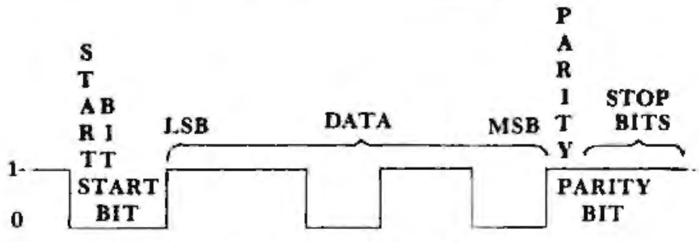
Another possibility is parallel communication. This would be used at closer range to achieve higher data rates. A parallel interface transfers an entire byte of data at a time. Some form of handshake is employed to coordinate the transfer timing. A portable 1802 unit might be brought and plugged into a central computer for a rapid transfer of data.

Perhaps the fastest and most direct communication between 6502 and 1802 could be obtained by combining the two processors as co-processors with common access to at least some memory regions. This would make possible the sharing of some tasks between the two processors. By setting or clearing specified bytes of shared memory, data might be passed from processor to processor and the activities of both coordinated.

Let's look at some serial data formats and the software considerations for producing them. The basic principal behind serial communication is to take a signal capable of presenting two states, 1 or 0, high or low, and to vary that signal in a specific time dependent pattern. This can be done readily by incorporating a UART such as the 1854 in your 1802 circuit. The 1854 is a CMOS UART (Universal Asynchronous Receiver / Transmitter). It has all the necessary circuitry on one chip for generating and interpreting serial data streams on a character by character basis. When connected to an 1802, the 1854 makes sending serial data as easy as outputting a byte of data to a selected port.

In the interests of keeping our 1802 system small and simple. However, let's do the following: we'll look at a way to use the Q line of our 1802 as a serial data output, and one of the External Flag lines as a serial data input. This eliminates the need for a UART, but it shifts the burden over to software.

What are the elements of serial data transmission that we must create by programming? Look at the illustration of an 8 bit data word in serial format. At the beginning of the word, the serial line is in a high (1) state. This high state is of an indefinite period of time. Transmission of the word is begun by bringing the line low for one bit-time. This is called the start bit. It is in effect saying, "Get ready guys-here comes the data!". The bit time is based on the desired data transfer rate, or "baudrate".

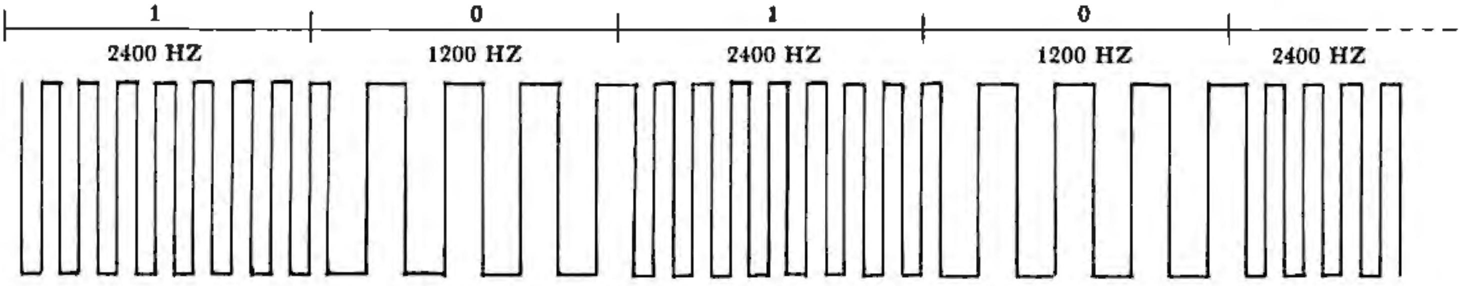


SERIAL REPRESENTATION OF 37_{16}

Following the start bit are 8 data bits, each using one bit-time. The first bit transmitted is the least significant bit of the data word. After the data bits comes a final parity bit. Finishing the transmission of the word are the stop bits. Stop bits are always 1 (high). For best reliability, 2 stop bits are recommended. This gives a receiver a fighting chance to synchronize itself with a continuous stream of data words. If a data word is not sent immediately, the line just remains high until the start bit of the next data word is sent.

For a variation on the theme, what if we wish to store the data on audio magnetic tape? We can use a very similar serial data format by superimposing audio tones onto our high and low segments of the signal. That is, let a high frequency tone represent a "1", and a low frequency tone a "0". The "Kansas City Standard" cassette format does in fact use this method. It differs from the above format only in that it does not use a parity bit. Each "0" consists of 4 cycles of 1200 HZ and each "1" consists of 8 cycles of 2400 HZ (see illustration).

Let's write a routine for generating either a straight serial data format or an audio-modulated cassette format. We'll set it up as a subroutine which, when called, will transmit the data in the "D" register in a serial format via the "Q" output flip-flop. We'll design our subroutine to allow for variation in the number of data bits. Parity will be settable as odd, even or completely off. The subroutine will also



PORTION OF A SERIAL BYTE IN AUDIO FORMAT

allow for either straightforward serial format or else audio-modulated serial format for use with a magnetic tape or telephone line transmission. In our next 1802 column, we'll examine some COSMAC code which will accomplish all this for us. ©

Book Review:"Son of Cheap Video"
Author:Don Lancaster
Publisher:Howard W. Sams, 1980
Price:$8.95

Reviewed by: Harvey B. Herman

To quote the author, "This is a you-build-it hardware book for hardware freaks...If you are not one of us, go away". I will assume that if you are still reading this review after seeing that quote that you will enjoy this book. It is intended for "poor folks" who like to tinker and construct useful things from a few chips and not much more. Specifically, it allows you to add a complete video display to a KIM-1 or the like for only $7 using five (count them) integrated circuits. Amazement is too mild a word for my reaction to that statement; flabbergasted is more like it.

The book is intended as a sequel to the author's earlier volume, "The Cheap Video Cookbook". Many references in the text to the earlier book suggest that it would be a good idea to have it close by to fully appreciate this effort. A legitimate criticism of the first circuitry concerned the amount of memory space used (28K bytes). What he now calls "scungy" video (I like the man's style) takes up 1K bytes for a 12x80 display - an impressive reduction in memory overhead.

A succession of projects is described in the book beyond scungy video. Lancaster shows how to combine cheap video with a "snuffler" coil on the outside of your TV set to free up processor time for normal comuting. This method locks the program and the display so picture jitter can be reduced with considerably less display program overhead. He includes a circuit for an EPROM programmer and describes how to use it in an extended music display example. Because the book leaves several projects as exercises (e.g. EPROM burning software) the book could be used as part of a course on microcomputers. Some of the construction hardware can be purchased from PAIA electronics (Oklahoma City, OK 73116) and could be conveniently provided to the students taking such a course.

I have not meant to leave the impression that the book is only for the KIM-1. Any of the enhanced-KIM clones (SYS or AIM) could benefit from the ideas in "Son of Cheap Video". Lancaster also includes chapters on 8080/Z80 systems, Heathkit H8, and Apple II (lower case display project). However, the book it not for every microcomputer owner as the initial quote suggested. Nevertheless it is well written, even entertaining in spots, can teach most of us a few things and save us money to boot. I recommend it highly. ©

Review:

Disk Operating System for KIM

Wilserv Industries $100
P.O. Box 115
Haddonfiled, NJ 08033

Reviewed by Harvey B. Herman

This is a short review of a disk operating system which has enhanced a KIM beyond my wildest dreams. I started with only a KIM - 1 but my system began to grow bigger and bigger almost immediately. Memory was added periodically, finally enough to use BASIC, using a KIMSI mother board. However, the weak link was the cassette operating system and the time it took to load programs. Switching to the Butterfield hypertape program helped but the delay (and occasionally bad loads) were irritating. I felt I really needed a better way to load and save programs. Wilserv had the answer.

Several years ago I purchased an Innovex 8'' disk and parts for a disk power supply. These sat around unused because an interface/controller to a minicomputer was never finished. To get the disk working on KIM, I needed a controller board (SDS Versafloppy I), a cable (made locally) and the software provided a Wilserv (Willi Kusche). To make a long story short, it works and I am very happy.

The KIM disk operating system is very convenient to use. It provides a link with BASIC and the same commands as the PET version. It provides a cheaper alternative for people like me who have most of the components already in hand. The only real disadvantages are the lack of random access files in the current version, and the element of do-it-yourself which does not appeal to everyone. Otherwise I recommend this software highly. ©

AIM 65 BY ROCKWELL INTERNATIONAL

AIM 65 is fully assembled, tested and warranted. With the addition of a low cost, readily available power supply, it's ready to start working for you.

AIM 65 features on-board thermal printer and alphanumeric display, and a terminal-style keyboard. It has an addressing capability up to 65K bytes, and comes with a user-dedicated 1K or 4K RAM. Two installed 4K ROMS hold a powerful Advanced Interface Monitor program, and three spare sockets are included to expand on-board ROM or PROM up to 20K bytes.

An Application Connector provides for attaching a TTY and one or two audio cassette recorders, and gives external access to the user-dedicated general purpose I/O lines.

Also included as standard are a comprehensive AIM 65 User's Manual, a handy pocket reference card, an R6500 Hardware Manual, an R6500 Programming Manual and an AIM 65 schematic.

AIM 65 is packaged on two compact modules. The circuit module is 12 inches wide and 10 inches long, the keyboard module is 12 inches wide and 4 inches long. They are connected by a detachable cable.

THERMAL PRINTER

Most desired feature on low-cost microcomputer systems . . .

- Wide 20-column printout
- Versatile 5 x 7 dot matrix format
- Complete 64-character ASCII alphanumeric format
- Fast 120 lines per minute
- Quite thermal operation
- Proven reliability

FULL-SIZE ALPHANUMERIC KEYBOARD

Provides compatibility with system terminals . . .

- Standard 54 key, terminal-style layout
- 26 alphabetic characters
- 10 numeric characters
- 22 special characters
- 9 control functions
- 3 user-defined functions

TRUE ALPHANUMERIC DISPLAY

Provides legible and lengthy display . . .

- 20 characters wide
- 16-segment characters
- High contrast monolithic characters
- Complete 64-character ASCII alphanumeric format

PROVEN R6500 MICROCOMPUTER SYSTEM DEVICES

Reliable, high performance NMOS technology . . .

- R6502 Central Processing Unit (CPU), operating at 1 MHz. Has 65K address capability, 13 addressing modes and true index capability. Simple but powerful 56 instructions.
- Read/Write Memory, using R2114 Static RAM devices. Available in 1K byte and 4K byte versions.
- 8K Monitor Program Memory, using R2332 Static ROM devices. Has sockets to accept additional 2332 ROM or 2532 PROM devices, to expand on-board Program memory up to 20K bytes.
- R6532 RAM-Input/Output-Timer (RIOT) combination device. Multipurpose circuit for AIM 65 Monitor functions.
- Two R6522 Versatile Interface Adapter (VIA) devices, which support AIM 65 and user functions. Each VIA has two parallel and one serial 8-bit, bidirectional I/O ports, two 2-bit peripheral handshake control lines and two fully-programmable 16-bit interval timer/event counters.

BUILT-IN EXPANSION CAPABILITY

- 44-Pin Application Connector for peripheral add-ons
- 44-Pin Expansion Connector has full system bus
- Both connectors are KIM-1 compatible

TTY AND AUDIO CASSETTE INTERFACES

Standard interface to low-cost peripherals . . .

- 20 ma. current loop TTY interface
- Interface for two audio cassette recorders
- Two audio cassette formats: ASCII KIM-1 compatible and binary, blocked file assembler compatible

ROM RESIDENT ADVANCED INTERACTIVE MONITOR

Advanced features found only on larger systems . . .

- Monitor-generated prompts
- Single keystroke commands
- Address independent data entry
- Debug aids
- Error messages
- Option and user interface linkage

ADVANCED INTERACTIVE MONITOR COMMANDS

- Major Function Entry
- Instruction Entry and Disassembly
- Display/Alter Registers and Memory
- Manipulate Breakpoints
- Control Instruction/Trace
- Control Peripheral Devices
- Call User-Defined Functions
- Comprehensive Text Editor

LOW COST PLUG-IN OPTIONS

• A65-010—4K Assembler—symbolic, two-pass	**$79.00**
• A65-020—8K BASIC Interpreter	**99.00**
• 3K RAM Expansion Kit	**50.00**

POWER SUPPLY SPECIFICATIONS

- +5 VDC ± 5% regulated @ 2.0 amps (max)
- +24 VDC ± 15% unregulated @ 2.5 amps (peak) 0.5 amps average

PRICE: $389.00(1K RAM)

Plus $4.00 UPS (shipped in U.S. must give ***street*** address), $10 parcel post to APO's, FPO's, Alaska, Hawaii. All international customers write for ordering information.

We manufacture a complete line of high quality expansion boards. Use reader service card to be added to our mailing list, or U.S. residents send $1.00 (International send $3.00 U.S.) for airmail delivery of our complete catalog.

RNB ENTERPRISES INCORPORATED

2951 W. Fairmount Avenue
Phoenix AZ 85017
(602)265-7564

master charge VISA

Expanding KIM-Style 6502 Single Board Computers

Part 1 of 3: Hal Chamberlin

Undoubtedly the most successful single board computer ever has been the KIM-1 made by MOS Technology (now Commodore). When introduced it apparently had just the right combination of features and price to attract tens of thousands of users. More recently of course the SYM-1 from Synertek and AIM-65 from Rockwell have incorporated numerous additional features into the same self-contained single board computer concept. Fortunately for users, all three of these machines are quite similar in their electrical characteristics.

Sooner or later however all computers need to be expanded and these single board machines are no exception. Although the SYM-1 and AIM-65 can be expanded somewhat merely by plugging in additional memory chips, the maximum limit is only 4K bytes of programmable memory. Thus additional boards are required for substantially increased RAM, ROM, or I/O capability. Recognizing this fact, the computer manufacturers as well as a number of independent accessory manufacturers have designed and brought to market a wide variety of expansion boards for the KIM, SYM, and AIM computers.

In most cases just having expansion boards available is not enough; there must also be a *motherboard* offered to plug them into since these computers have no on-board bus and slot sockets of their own. To date the computer manufacturers and independents have selected no fewer than four distinctly different ways to do this. First on the scene of course was MOS Technology who offered the KIM-4 expansion motherboard which mated with their KIM-2 and KIM-3 expansion memory boards. The bus presented by the KIM-4, which is called the "KIM-4 Bus", is in many ways similar to the bus presented by the computer itself as its own expansion edge connector. The primary difference is an altered pin assignment which is basically a one pin shift from the expansion connector assignments. This apparently was done to provide additional ground connections. Since then, independent manufacturers have also offered KIM-4 style expansion motherboards although there are important differences from the original KIM-4 (see Compute issue #3).

Shortly thereafter, as soon as the KIM's popularity became known, other independent manufacturers offered expansion motherboards which presented an S-100 style bus to the expansion boards. The primary advantage of this approach is that the user is not restricted to using expansion boards designed specifically for KIM-style machines but instead can choose from hundreds of S-100 compatible boards designed for 8080 based systems. Unfortunately many of the more sophisticated S-100 boards such as large dynamic memories, graphic display interfaces, and disk controllers could not be used because of substantial timing differences between 6502 and 8080 style microprocessors.

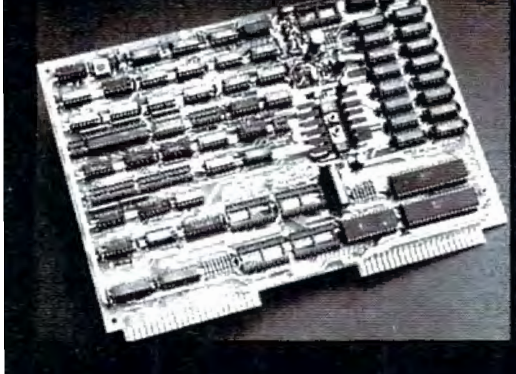
Late in 1977 Micro Technology Unlimited introduced a motherboard and card cage for the 6502 based single-board computers. The motherboard is little more than 5 edge connectors wired in parallel with one for the computer and the other 4 for expansion boards. The bus presented is the same pinout as that of the processor's expansion connector. The main advantage of this technique is the low cost and compact packaging afforded by the elimination of bus buffers. In addition, expansion boards compatible with this bus may be easily connected directly in parallel with the expansion connector if for some reason the motherboard is not desired. The main disadvantage is that the number of expansion boards is limited to four by the small drive capability of the computer's own bus.

Recently Rockwell has introduced its expansion motherboard which essentially presents an Exorcisor bus to the expansion boards. Motorola originated this bus for use in their Exorcisor microprocessor development systems. Rockwell also uses the Exorcisor bus in their system 65 development system. The advantage of this method, at least to Rockwell, is avoiding the need to develop new expansion boards just for the AIM-65. To users the biggest drawback of the Exorcisor bus probably is the lack of reasonably priced boards to plug into it.

All four of these techniques are quite viable methods for expanding KIM-1, SYM-1, and AIM-65 single board computers and each has a broad base of dedicated users.

Mechanics

All three of the single board computers are intended to simply rest flat on a tabletop using the several quarter-inch high rubber feet provided. Although not the most beautiful thing in the world, it works well in many cases and is certainly inexpensive. In situations where better appearance is desired or small children are present, there are vacuum-forced dress covers available that simply slip over the computer board hiding everything except the display and keyboard.



The KIM-4, S-100, and Exorcisor type expansion motherboards simply extend this board-on-the-table concept. Typically the expansion motherboard is roughly the same size as the computer board and plugs straight onto its expansion connector. On the motherboard are perhaps a dozen integrated circuits for address decoding, bus buffering, and voltage regulation. The majority of the space however is taken up by 4 to 8 edge connectors which form the "slots" of the expansion bus. When plugged into these slots, the expansion boards assume a vertical orientation.

A system expanded this way uses a large amount of additional table space, and in the case of th KIM, it is useful space to the left of the computer. The assembly of interconnected boards is also rather fragile and certainly not portable unless dismantled (most people would probably bolt the computer and motherboard to a sheet of plywood or plastic to avoid this). In particular a stray elbow can do considerable damage if a board is knocked out of its slot during operation. Unfortunately the available plastic dress covers do nothing to protect the added motherboard or expansion boards.

Another approach that has been slowly gaining acceptance is to place the expansion boards underneath and parallel to the computer board. Thus the expansion motherboard, which ties all of the boards together, is vertical. In order to hold this assembly of boards together, an aluminum frame with card guides is typically supplied and the motherboard is attached to an opening in the frame. Figures 1 and 2 show the KIM and AIM versions respectively of Micro Technology's implementation of this concept.

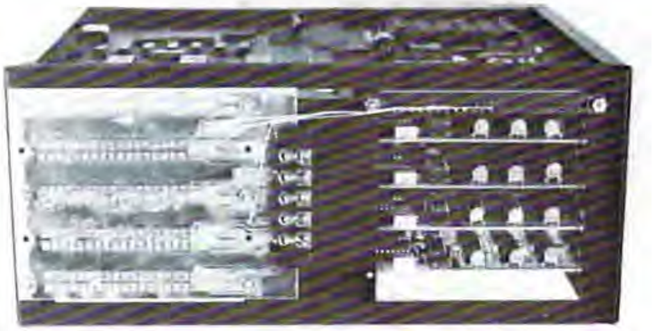


FIG. 1 KIM-1 INSTALLED IN A MICRO TECHNOLOGY UNLIMITED MOTHERBOARD/CARD FILE

The advantages of this configuration of course are reduced table space requirements and greatly increased protection for the expansion boards. The entire assembly of computer and boards is now one portable unit with only the power supply left over to worry about. The computer board is still exposed however. Probably the only potential disadvantage is that the computer's keyboard has been raised about 4 inches above the tabletop.

Electronics

There are electronic factors to consider as well when expanding a KIM, SYM, or AIM computer. In order to minimize cost, complexity, and power consumption, all three of these single board computers are designed without buffers between the microprocessor chip and the expansion edge connector. The KIM-1 went one step further and omitted part of the address decoding circuitry as well. The lack of buffers means that the expansion bus presented by these computers has a DC drive capability of only one standard TTL load, or equivalently, 5 low power Shottky loads. The AC drive capability depends on the desired signal rise-time. For bus operation at 1mHz, a total of approximately 25 "connections" at 6pF each can be driven. A connection here is defined as a gate input, disabled tri-state output, or MOS input (which does not contribute to DC loading).

Compared to other bus-oriented computers, such as S-100 machines, this does not sound like much of a bus at all since these machines typically have a drive capability of 30 standard TTL loads (74 series) or nearly 150 low power Shottky (74LS series) or over 200 low power TTL (74L series) loads. In fact, the original advertising for the MITS Altair computer boasted an expansion capacity of "over 200 boards". While this may have seemed necessary when using MITS's 1K memory and single port I/O boards, 10 slots is ample for even the largest S-100 setup when using today's dense memory and peripheral interface boards.

Over the years, experience has shown that several factors other than sheer driver power limit the number of boards that may be connected to a bus.

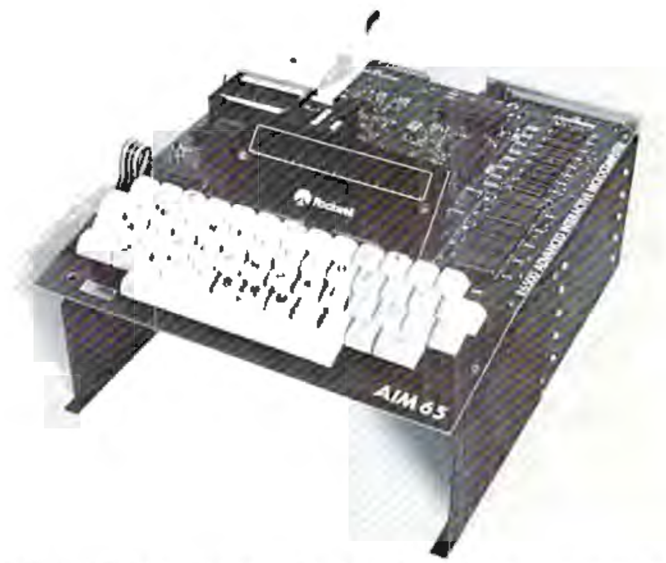


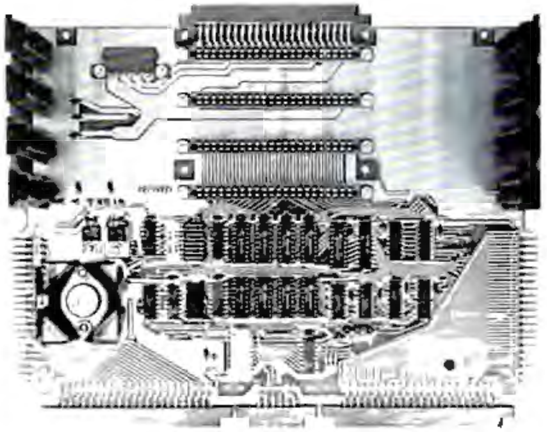
FIG. 2. AIM-65 INSTALLED IN A MICRO TECHNOLOGY UNLIMITED MOTHERBOARD/CARD FILE

The most serious of these is crosstalk noise between the bus address/data lines and the various bus control lines. This noise arises when large numbers of address and data lines change state simultaneously, which is a common occurrence. The fast voltage risetimes (around 5NS with the popular 8T97 drivers) and 50MA or greater surges of current along each changing line couple electrostatically and magnetically to other lines in the bus and on the expansion boards themselves. Longer busses and more boards plugged in gradually increase the crosstalk until noise on the control lines causes false triggering of memory and I/O boards and thus system failure. So severe is this problem that early S-100 systems would fail to operate even before the 16 board capacity of a single cabinet was reached.

A related, but much less severe problem, is signal reflection from the ends of the bus lines, which after all, act like transmission lines. This effect becomes significant when the signal transmission time exceeds about 1/2 of the signal risetime. At 1.5NS per foot with a 5NS risetime, the bus would have to be two feet long before termination was required. The apparent success of bus terminators sold for S-100 systems is probably due to their reduction of signal swings (the logic 1 level is limited to 3 volts and floating bus lines are pulled to 3 volts) which in turn reduces crosstalk noise.

From the author's experience in designing a large, fast bus oriented system (specifically the A. B. Dick Magna SL four terminal full-page word processing computer), there are three ways to solve bus noise problems. One is to thoroughly shield the bus with a full-width ground plane, or ideally, a three-layer motherboard with data/address on one side, ground in the middle, and control signals on the other side. This solves noise coupling on the bus but not on the expansion boards which in turn must be carefully designed to minimize their own crosstalk. This technique was used in the Magna SL machine because of speed requirements.

Another technique is to use filters and delays on the control signals obtained from the bus in order to reject narrow noise pulses. This technique can be extended to deal with any kind of noise problem at the expense of system speed and is the one typically used with minicomputers such as DEC PDP-11's and Data General NOVA's.



The third technique attacks the source of the noise, namely fast risetimes and large current surges, by using a low power bus. With liesurely risetimes of 50 to 100NS and drive capabilities of less than 10MA, such a bus is virtually noise-free and quite fast enough for normal microprocessor operation. This technique, coupled with some attention to groundplane shielding, is most applicable to unbuffered KIM/SYM/AIM expansion busses.

The foregoing is not meant to imply that all of the buffered expansion motherboards available for the KIM, SYM, and AIM computers are racked with noise. In fact, their bus length and number of slots is generally small enough to keep noise at tolerable levels. The major point is that high power drivers and indefinitely expandable busses do have drawbacks of their own.

The real question at this point then is: How many expansion boards can the unbuffered microprocessor bus drive before becoming overloaded? The 6502 microprocessor is rated to drive slightly more than 1 standard TTL load (equivalent to five low power shottky loads) on its address and data busses while most of the RAM's and ROM's tied to the data bus can drive two standard TTL loads. The 6520, 6522, and 6530 I/O chips have the same drive capability as the microprocessor. Thus in general the answer is at least four boards provided that the expansion boards themselves buffer the bus such that only one low power shottky load (.36MA in the zero state) is presented to the bus by the board. Many boards on the market and particularly those designed for an unbuffered bus do this. Actually, any well designed board would be expected to buffer the bus in order to provide clean signals for the remainder of the board logic. The reason that only four boards can be driven instead of five is that some of the address lines are loaded by a low power Shottky decoder IC on the computer board itself. ©

Next time: The Great Experiment

ISO-1

ISO-2

CAPUTE!

Wherein We Acknowledge Recent Goofs

The Group of Us

Please note that the article written by Neil Harris entitled "Times Square" in your November/December contains at least two errors. The program did not work until the following two changes were made:

lineno as written

```
410      FOR I = 0 to 9 STEP 3
```

correction

FOR I = 9 to 0 STEP -3 (this will put the bottom line where it belongs)

```
560      FOR L = 1 TO A
```

correction

FOR L = 1 to LEN(A$) (the original program causes line 730 to exceed the amt. of info available & print an error message)

The program still does not quite work properly in the sense that the "moving border" is not quite right. The stars are not surrounding the screen properly but I have not yet been able to cure this.

Leo Cerruti
Syosset, L.I., N.Y.

©

Nov. 30, 1980

Dear Sirs,

I would like to correct a misleading statement in my article in COMPUTE, issue 7, entitled VISIBLE MEMORY PET PRINTER DUMP. I implied that only a single byte needed to be changed to dump a different 8K block. (The LDA #VMORG on line 1590). While this is strictly speaking correct, it assumes the existence of RAM in the region $AFF6 -$AFFF. These are the last 9 bytes of a VM located at $9000. Since many VM users may not have RAM there, the simplest 'fix' is to go over the listing and change all the three byte commands ending in $AF to VMORG + $1F. For example, if your VM is located at $4000, change those $AF's to $5F's.

Also, in the data tables at the end of the listing, the assembler did not print out data bytes beyond three per line. The complete assembly listing should read -

```
2170: 0397 F0 C8 A0 78
2180: 039B 50 28 00
2190: 039E 00 00 00 00 00 00
2200: 03A4 80 06 35 1D
```

To err is human, myself and **COMPUTE!** being no exception. The captions on the upper-left and lower right photographs on p.19 should be interchanged. The youngster to the left of Jim Butterfield is my son, Philip.

Keep up the excellent work.

Frank Covitz
Deer Hill Rd.
Lebanon, NJ 08833

©

Keep Those Cards and Letters Coming

COMPUTE! Needs You!

Address articles, programming notes and comments to:
The Editor
COMPUTE!
P.O. Box 5406
Greensboro, NC 27403

Index to Advertisers

Program Listings for COMPUTE

Cursor control characters will appear in source listings as shown below:

```
h̲=HOME         , ĥ=CLEAR SCREEN
↓=DOWN CURSOR  , ↑=UP CURSOR
→=RIGHT CURSOR, ←=LEFT CURSOR
r̲=REVERSE      , r̂=REVERSE OFF
```

Graphics (i.e. shifted) characters will appear as the unshifted alphanumeric character with an underline. This does not apply to the cursor control characters. The Spinwriter thimble doesn't have a backarrow symbol, so a "~" is used instead.

The "⌐" is used to indicate the beginning of a continuation line. It is also used to indicate the end of a line which ends with a space. This prevents any spaces from being hidden. ©

The 6502 Resource Magazine

COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

My computer is: PET ____ **APPLE** ____ **ATARI** ____ **OSI** ____ **SYM** ____ **KIM** ____
AIM ____ **Other** ____ **(Specify)** ____________________

□ Please enter my 1 year (12 issue) subscription to COMPUTE
□ New subscription
□ Renewal subscription

______$16.00 (U.S. Mail Order Subscription)
______$18.00 U.S. Funds (Canadian Subscription)

My Name, Address: ____________________________
____
____
____

Charge my Visa __________ MC __________
Number ____________________ Expires _____/_____/_____

1 2 3 4 5 6 7 8 9 10 11 12

The 6502 Resource Magazine

COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

My computer is: PET ____ **APPLE** ____ **ATARI** ____ **OSI** ____ **SYM** ____ **KIM** ____
AIM ____ **Other** ____ **(Specify)** ____________________

□ Please enter my 1 year (12 issue) subscription to COMPUTE
□ New subscription
□ Renewal subscription

______$16.00 (U.S. Mail Order Subscription)
______$18.00 U.S. Funds (Canadian Subscription)

My Name, Address: ____________________________
____
____
____

Charge my Visa __________ MC __________
Number ____________________ Expires _____/_____/_____

1 2 3 4 5 6 7 8 9 10 11 12

The Editor's Feedback:

My computer is: PET ____ **APPLE** ____ **ATARI** ____ **OSI** ____ **SYM** ____ **KIM** ____
AIM ____ **Other** ____ **(Specify)** ____________________

My application is (check all that apply):
____ Home/Personal ____ Business ____ Educational
____ Industrial ____ Other

Comments.

Content
Best Article This Issue (page #, title)

____

Other suggestions

1 2 3 4 5 6 7 8 9 10 11 12

Place
Stamp
Here

COMPUTE./compute II
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

Dear Micro Technology Unlimited.

Please rush me the catalog of 6502 expansion and application products

I have the following computer(s)

Old PET____ New PET____ AIM-65____ KIM-1____ SYM-1____

Thank you for expediting

Sincerely,

____

NAME ____________________

STREET ____________________

CITY/STATE/ZIP ____________________

COMPUTE 6

PROMINICO · PROMINICO · PROMINICO · PROMINICO · PROMINICO · PROMINICO · PROMINICO · PROMINICO

XDOS ROM™ – $97.50 for the new CBM/PET computer and 2040 disk unit. Sophisticated disk and printer support package in ROM which greatly simplifies diskette management. Comprehensive instruction manual included.

SORT ROM™ – $97.50 for the new CBM/PET computer. SORT ROM is a versatile general purpose array-sorting utility giving extremely fast sorting rates.

Please send__________ XDOS ROM @ $97.50 + $2.50 shipping = $__________

__________ SORT ROM @ $97.50 + $2.50 shipping = $__________

TOTAL $__________

ROM SOCKET: □ Left ($B000) □ Centre ($A000) □ Right ($9000) (B.C. Residents add 4% tax.)

□ Cheque □ Money Order □ Visa/Chargex No.__________ Expiry Date__________

NAME:____________________

ADDRESS:____________________

CITY:__________ STATE:__________ ZIP:__________

For complete description see PROMINICO advertisement in this issue or phone (604) 738-7811

PROMINICO · PROMINICO · PROMINICO · PROMINICO · PROMINICO · PROMINICO · PROMINICO · PROMINICO

Skyles Electric Works

YES, I WANT MORE INFORMATION ABOUT:

PET PERIPHERALS

- □ 8, 16, 24 K bytes of PET Memory Expansion Systems
- □ Full sized PET Keyboard with numeric pad
- □ PAL-40 (40 column), PAL-80 (80 column) PET Printers

PET PROGRAMMING AIDS

- □ BASIC PROGRAMMERS TOOLKIT
- □ MacroTeA 6502 Software Development System

Name ____________________

Address ____________________

City/State/Zip____________________

□ *Please send name of local Skyles Electric Works dealer.*

Place Postage
Stamp Here

Micro Technology Unlimited
P.O. Box 12106
2806 Hillsborough St.
Raleigh, NC 27605

PLACE
STAMP
HERE

PROMINCO LTD.
1921 BURRARD STREET
VANCOUVER, B.C.
V6J 3H3

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit No. 503 Cupertino, CA

Postage will be paid by addressee:

Skyles Electric Works
10301 Stonydale Drive
P. O. Box 574
Cupertino, CA 95015

Big Savings On Atari & PET!

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

CALL NOW (717) 323-7921

Atari® 800™ 16K Personal Computer

List $1080

only $779

Atari® Peripherals:

400 8K	$ 444
410 Recorder	59
815 Disk	1139
810 Disk	519
822 Printer	349
825 Printer	749
830 Modem	149
850 Interface Module	169

Atari® Accessories

CX852 8K RAM	$ 89
CX853 RAM	144
CX70 Light Pen	59
CX30 Paddle	17
CX40 Joystick	17
CX8100 Blank Diskettes (5/box)	21

Atari® Software

Entertainment:

CXL4004 Basketball	$29
CX4105 Blackjack	12
CXL4009 Chess	29
CXL4011 Star Raiders™	44
CX4111 Space Invaders	17
CXL4006 Super Breakout™	29
CXL4010 3-D Tic-Tac-Toe	29
CXL4005 Video Easel™	29

Personal Interest & Developement

CXL4104 Mailing List	$16
CX4107 Biorhythm	12
CXL4007 Music Composer	44
CX4110 Typing	19
CX4101 An Invitation to Programming	16

Information & Communication:

CXL4015 TeleLink™	19

Programming Languages:

CXL4003 Assembler Editor	45
CXL4002 BASIC Computing Language	45

Program Cassettes:

CX4121 Energy Czar™	$12
CX4108 Hangman	12
CX4102 Kingdom™	12
CX4112 States & Capitals	12
CX4114 European Countries & Capitals	12

Education: (Talk & Teach Courseware)

CXL4001 Education System Master	$20
CX6001 U.S. History	22
CX6002 U.S. Government	22
CX6003 Supervisory Skills	22
CX6004 World History (Western)	22
CX6005 Basic Sociology	22
CX6006 Counseling Proceedures	22
CX6007 Principles of Accounting	22
CX6008 Physics	22
CX6009 Great Classics	22
CX6010 Business Communications	22
CX6011 Basic Psychology	22
CX6012 Effective Writing	22
CX6014 Principles of Economics	22
CX6015 Spelling	22
CX6016 Basic Electricity	22
CX6017 Basic Algebra	22

Professional Applications:

CX8102 Calculator	$22
CX4109 Graph It	16
CX4103 Statistics	16

Investment Analysis:

CX8106 Bond Analysis	$ 19
CX8107 Stock Analysis	19
CX8101 Stock Charting	19

commodore

Commodore Computers:

PET 2001 Professional Computer - 16N	$ 825
32N	1070
CBM 2001 Business Computer - 16B	825
32B	1070
CBM 8000 Business Computer - 8016	1240
8032	1480

Commodore Peripherals:

CBM 2022 Printer	$ 655
CBM 2023 Printer	575
CBM 2040 Dual Drive Floppy Disk	1070
CBM 8050 Dual Drive Floppy Disk	1395
CBM C2N Cassette Drive	85
CBM Modem	360
CBM Voice Synthesizer	360

Cables:

PET to IEEE Cable	$ 36
IEEE to IEEE Cable	45

Software

Professional Software Inc.

WordPro 2 (40 col.)	$ 80
WordPro 3 (40 col.)	160
WordPro 4 (80 col.)	245

Personal Software, Inc.

Visicalc - Apple	$122
Atari	163
PET	163
Desktop Plan - Apple	80
CCA Data - Apple	80

Texas Instruments:

Texas Instruments - RF Modulator	$ 64
Console	499

Call For Radio Shack Prices!

To Order:

Phone Orders invited. Or send cashiers check or money order. Equipment shipped UPS collect. Pennsylvania residents add 6% sales tax. Equipment is subject to price change and availability without notice.

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

The Great American Solution Machine.

Meet Commodore. The business computer that's providing solutions for more than 100,000 people all over the world. Built by one of the pioneers in office machines. With a reputation for quality that can only come from vertical integration of the total manufacturing process. Commodore builds, not assembles.

Compare Commodore's word and data processing capabilities with computers costing twice or even three times as much. You'll see why so many small businesses are turning to Commodore for solutions to problems as varied as these:

□ A car leasing company's customers were terminating too early for account profitability. Solution: A 16K Commodore. It analyzes cash flow on over 1200 accounts, identifies those for early penalties, and even writes up lease contracts. Commodore paid for itself within weeks.

□ A law firm needed a high quality, easy-to-use, affordable word processing system. Solution: Commodore plus its WordPro software package. At a $6,000 savings.

□ A gasoline retailer needed to inventory, order and set prices; determine Federal and state income taxes; and comply with Federal pricing and allocation regulations. All done daily, weekly, monthly and yearly. Solution: Commodore. It keeps his business on track—and Uncle Sam off his back.

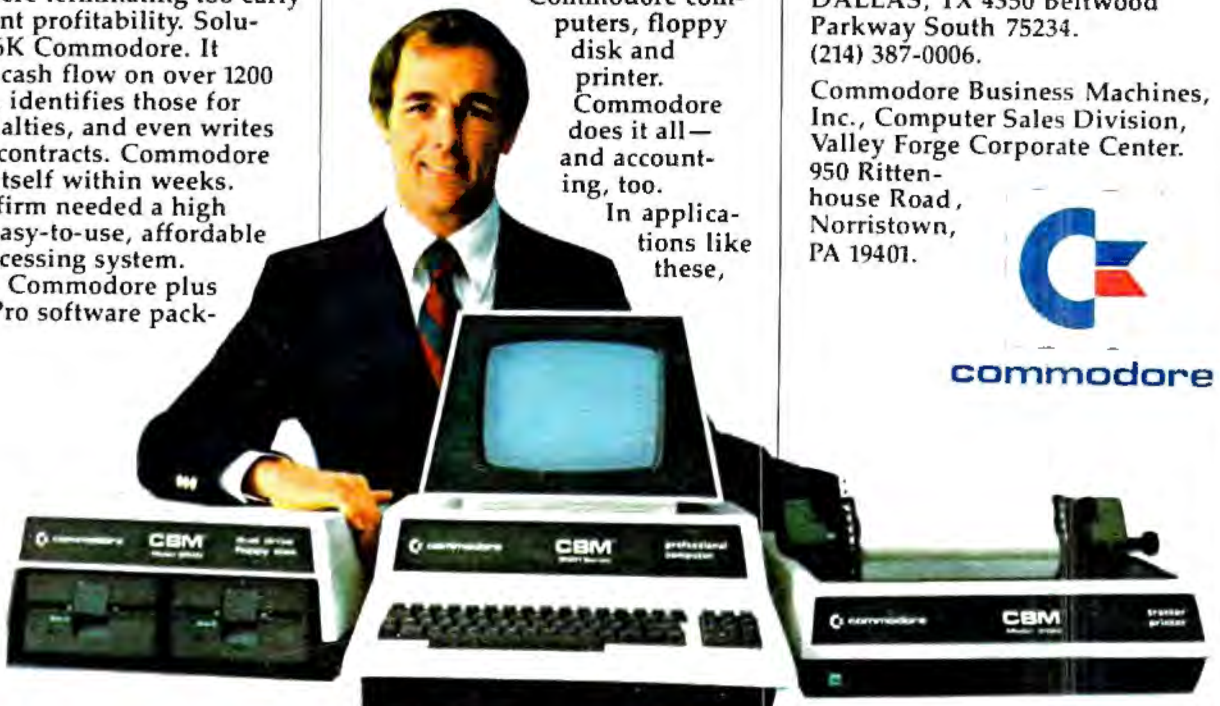
□ A paint and wallpaper store had to inventory over 600 expensive wallpaper lines for profitability, monitor distributor sales, set and track salesmen's goals, and help the customer select the right size, pattern and quantity. Solution: Two 32K Commodore computers, floppy disk and printer. Commodore does it all—and accounting, too.

In applications like these, and many more, Commodore solves the problems that stand in the way of increased profitability. Commodore can provide the solution in your Great American business, too. Find out more by calling or writing any of Commodore's District Sales Offices.

COSTA MESA, CA 2955 N. Airway Avenue 92626. (714) 979-6307.
SANTA CLARA, CA 3330 Scott Boulevard 95051. (408) 727-1130.
DECATUR, GA 5360 Snapfinger Woods Drive 30035. (404) 987-3311.
BENSENVILLE, IL 790 Maple Lane 60106. (312) 595-5990.
NORRISTOWN, PA
950 Rittenhouse Road 19401.
(215) 666-7950.
DALLAS, TX 4350 Beltwood Parkway South 75234.
(214) 387-0006.

Commodore Business Machines, Inc., Computer Sales Division, Valley Forge Corporate Center.
950 Rittenhouse Road,
Norristown,
PA 19401.

commodore



COMMODORE