Timekeeping For The VIC-20

The Resource Magazine For CBM/PET • Apple • Atari • OSI

Creating A Simple Word Processor

COMPUTE!

$2.50
February, 1982
Issue 20
Vol. 4, No. 2
63379

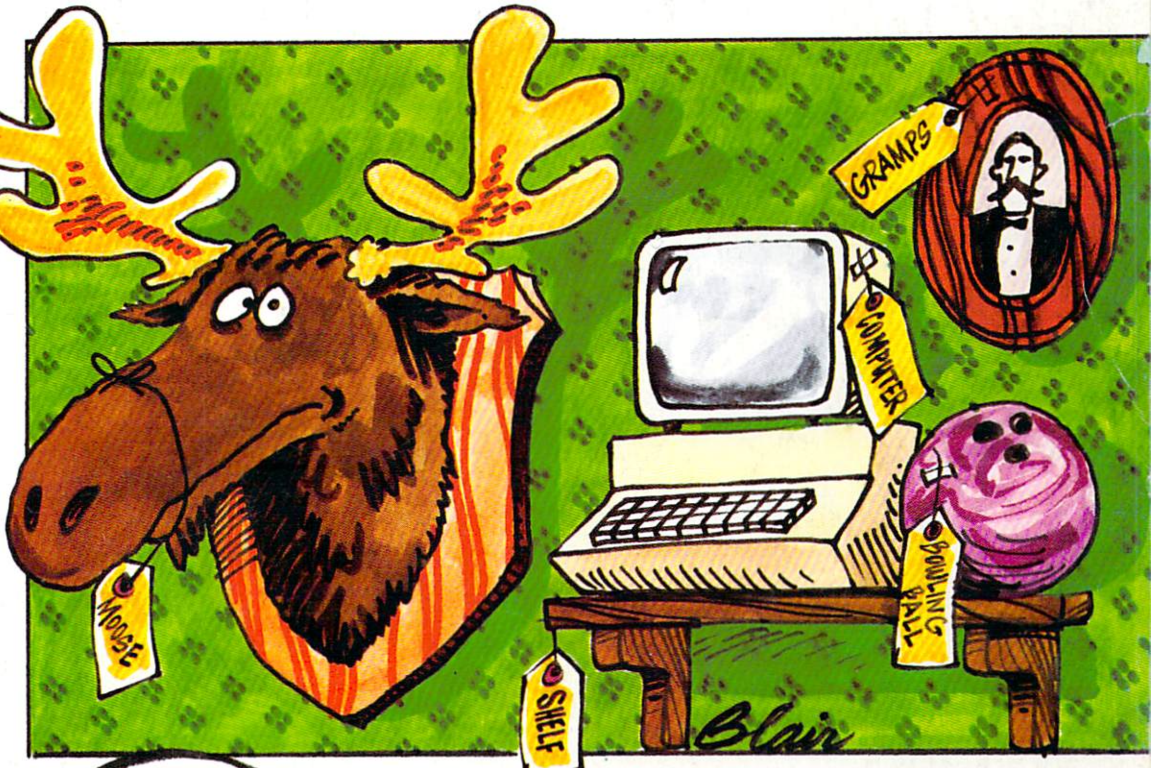
The Journal For Progressive Computing™

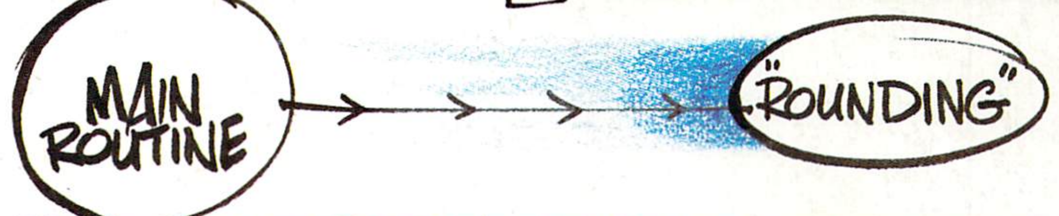
Programming Your Home Insurance Inventory

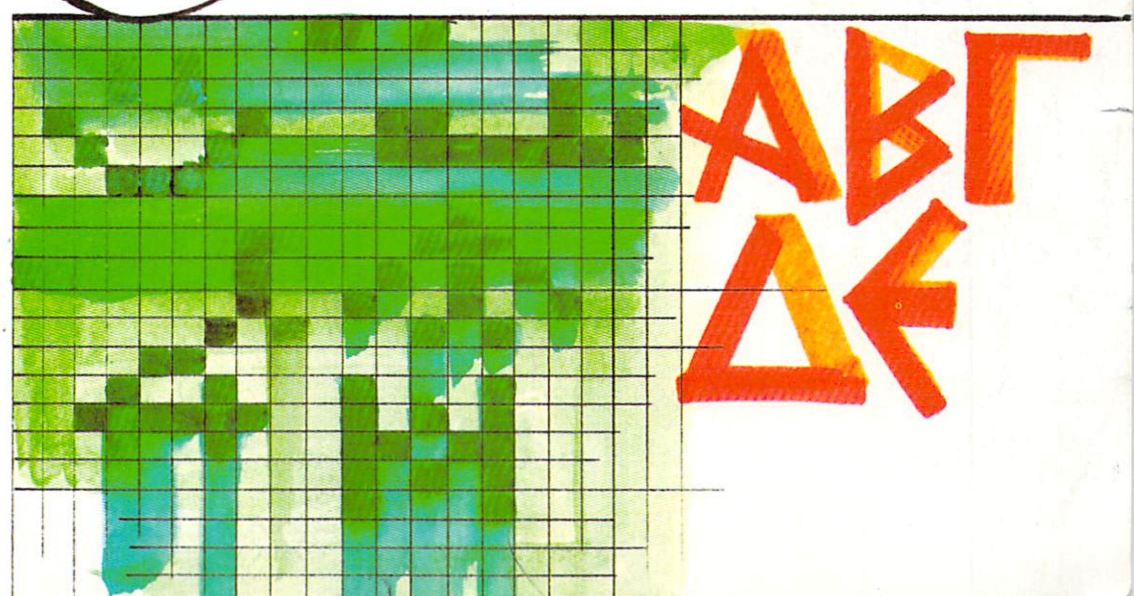
Named GOSUB With Variable Passing: Apple

Atari Player-Missile Graphics Made Easy

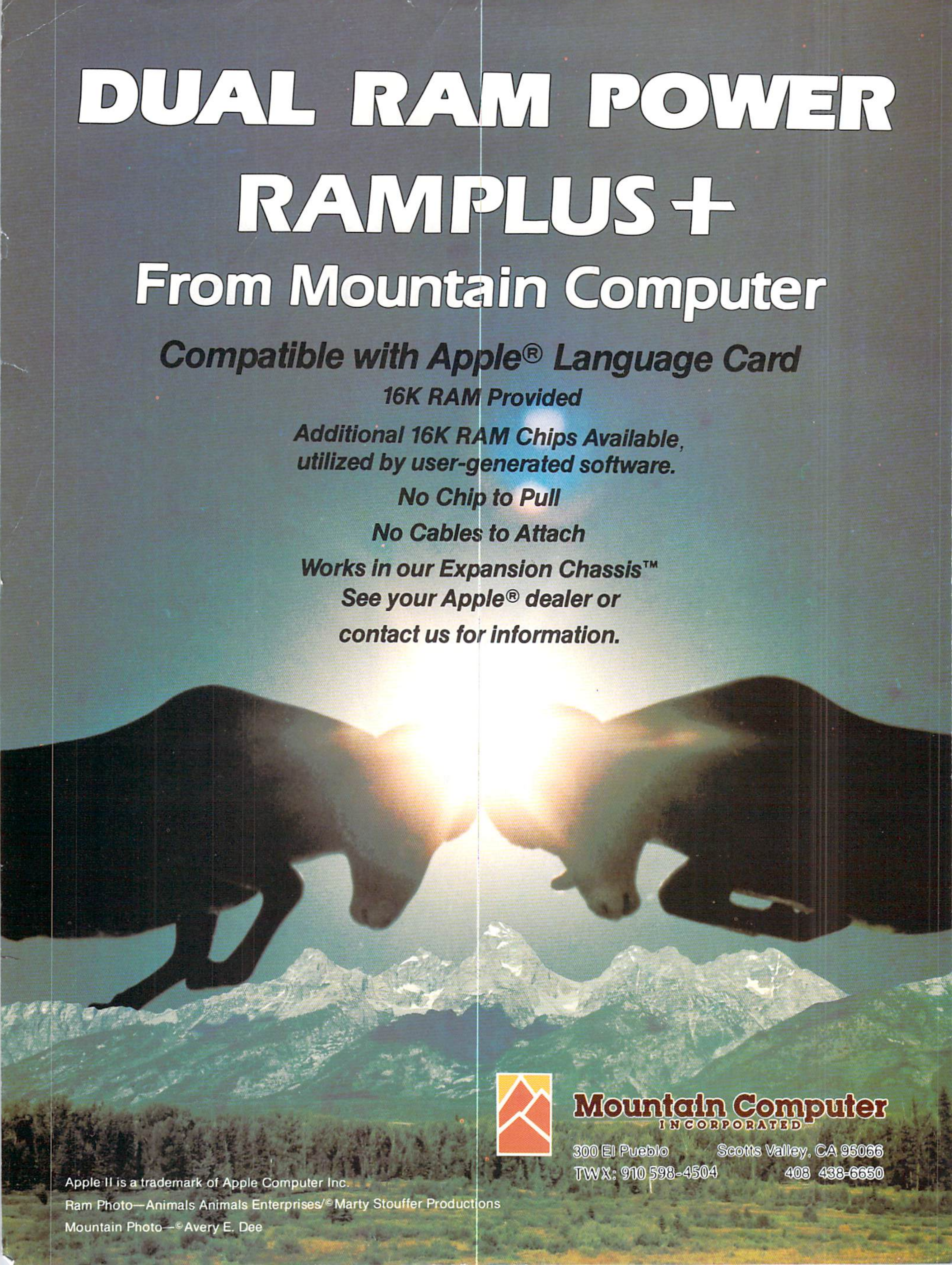
A User Defined Character Editor For PET/CBM







0 74470 63379 02


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

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

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

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

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

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

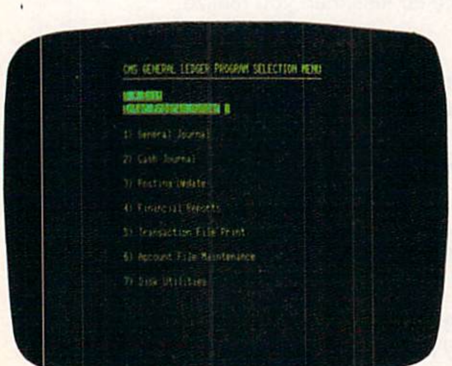
Invest in your office's future...
Invest in WordPro PLUS...
Call us today for the name of the WordPro PLUS dealer nearest you.

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

CMS Softwa
General Accou

For the Commodore 8000

Now there is an interactive General Accounting System designed especially for the first time user. The user is led through each function by highlighted prompts explaining the required input at each point. And all input requests are prompted with complete verification. Plus, a step-by-step instruction manual guides the user through the program. The user prompts and the detailed manual make it virtually impossible for the first time user to get lost or to accidentally crash.

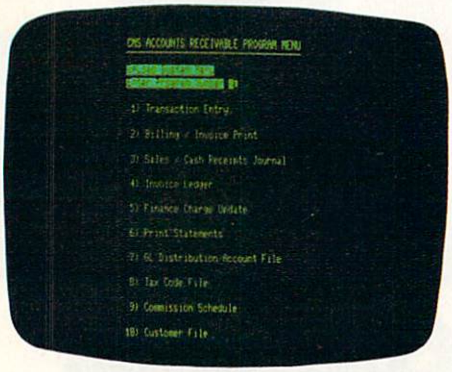


General Ledger

Cash Disbursements and Cash Receipts Journal as well as a General Journal. Maintains account balances for Present Month, Quarter to Date and Year to Date. Accepts postings from Accounts Receivable, Accounts Payable, Payroll and other programs. Also provides user customized financial statements. Up to 1000 accounts and up to 99 departments.

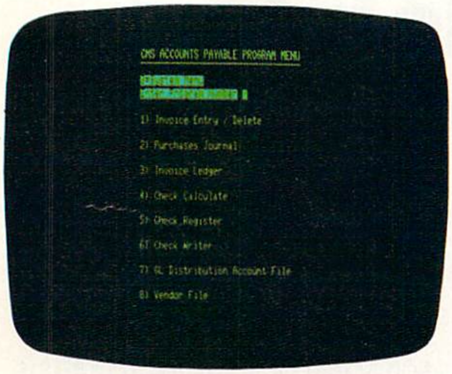
re Systems
nting System II

Series Computer System



Accounts Receivable

Invoices and Monthly Statements as well as Credit and Debit Memos. Accomodates full or partial payments. User sets finance charge rate and period. User also sets invoice aging reports and aging breaks. Invoices may be distributed among nine different General Ledger accounts with automatic updating to the General Ledger. During invoice data entry the invoice is displayed on the screen and typed exactly as if in a typewriter. File capacity is 2000 minus number of customers multiplied by 1.4. Five hundred customers allows room for 2100 invoices.



Accounts Payable

Accounts Payable checks with full voucher detail. Detailed check register. Automatic application of Credit Memos. User sets invoice aging reports and aging breaks. Invoices may be distributed among nine different General Ledger accounts with automatic updating to the General Ledger. File capacity is 2000 minus the number of vendors multiplied by two.

Payroll

Payroll checks with full deductions and pay detail. Pays regular, overtime, holiday and piece work hours. Accommodates Weekly, Bi-Weekly, Semi-Monthly and Monthly employees. Up to eight miscellaneous deductions or payments per employee. Prints Payroll Journal, Payroll Check Register and an Absentee Report as well as 941 information and W2 forms. Maintains Monthly, Quarterly and Yearly totals for up to 350 employees. Automatic General Ledger updating.

See your nearest Commodore dealer for a demonstration.

CMS Software Systems

2204 Camp David   Mesquite, TX. 75149   214 285-3581

TASC™. The Applesoft* Compiler. It turns your Apple into a power tool.

Step up to speed. TASC, the Applesoft Compiler, converts a standard Applesoft BASIC program into super-fast machine code. By increasing program execution speed up to 20 times, Microsoft gives you a power tool for Applesoft BASIC programming.

Highest capacity available. TASC will compile and run larger programs than any other Applesoft Compiler. As a disk-based system, it doesn't require the simultaneous presence of compiler and program in memory. The memory you save allows you to compile significantly bigger programs.

Power without bulk. Code expansion of up to 100% severely restricts other compilers. TASC's special code compression schemes typically limit code expansion to only 25%. You'll really appreciate that with complex programs or programs that utilize Apple's hi-res graphic pages.

More BASIC power. TASC's powerful new commands increase Applesoft BASIC programming capability. Chain with COMMON allows compiled programs to share variables, so a main menu supports several programs in a single runtime environment.

TASC's True Integer Arithmetic and Integer FOR...NEXT capabilities maximize the execution speed of compiled programs. TASC's near total compatibility with Applesoft speeds compilation of existing programs with little or no modification.

What about mistakes? You perfect your programs interactively with Applesoft. If something does slip by, TASC recovers from errors discovered in compilation and traps all runtime errors. It even permits graceful interruptions during compilation.

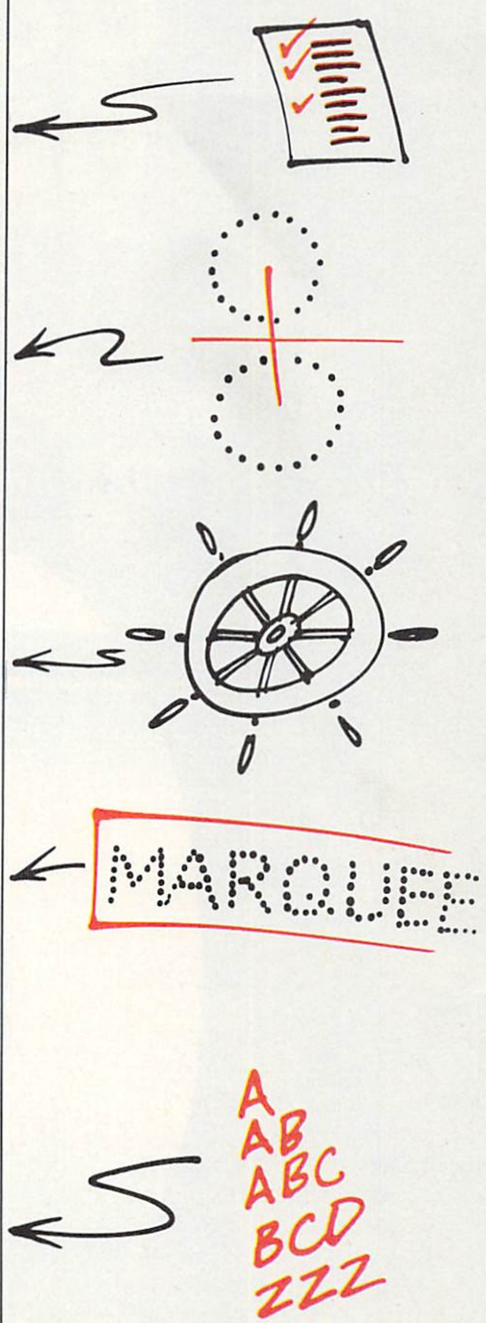
See for yourself. Ask for a demonstration of TASC at your Microsoft dealer. Discover the software package that turns your Apple into a power tool.

MICROSOFT
CONSUMER PRODUCTS
A Division of Microsoft Inc.
10700 Northup Way • Bellevue, WA 98004

*Applesoft is a trademark of Apple Computer, Inc.

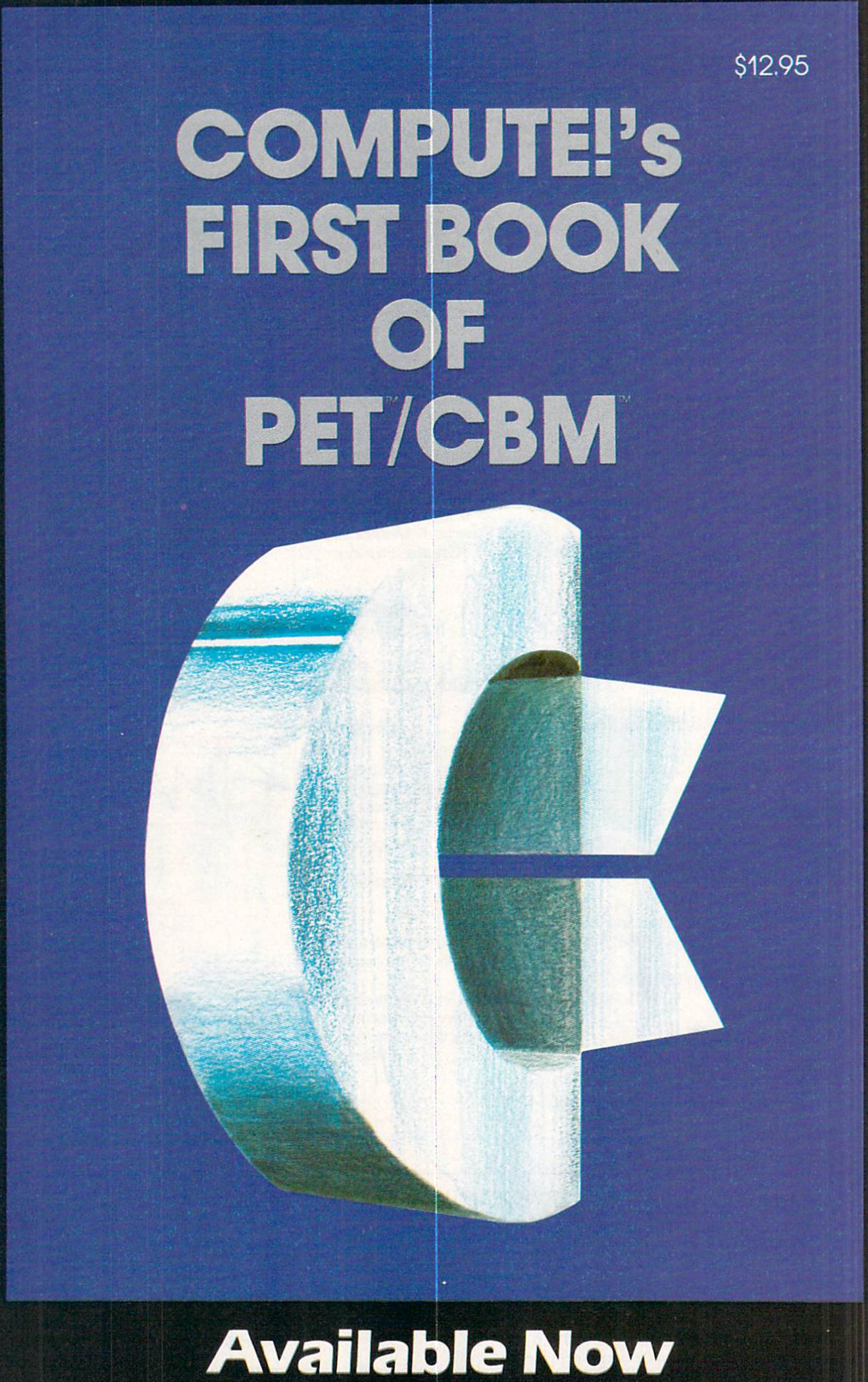
Table of Contents

February 1982, Vol. 4, No. 2



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

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


$12.95
COMPUTE!'s
FIRST BOOK
OF
PET™/CBM™

The Editor's notes...

Robert C. Lock
Publisher/Editor

A Major Format Change

Now that we've completed the merger of *Home and Educational COMPUTING!* and *Recreational Computing* into **COMPUTE!**, we're concentrating on revamping the organization of the magazine to better serve you. Beginning in March, **COMPUTE!** will have two distinct sections rather than the six it has now. The first section will be called "Home and Educational COMPUTING!," containing applications, tutorials, columns, and reviews. The second section of the magazine will become "The Journal," carrying a mix of articles for intermediate and advanced users.

We'll continue "New Products," *and* continue to provide the same excellent resource and applications articles. As we move into the new year with continuing explosive growth, we're sure you "old timers" will find the new format easier to use, and you beginners to the world of personal computing will find it much more convenient. Remember, this starts in March and, as always, we'd appreciate your feedback and comments.

The Hardware Wars: Late-breaking and Major News

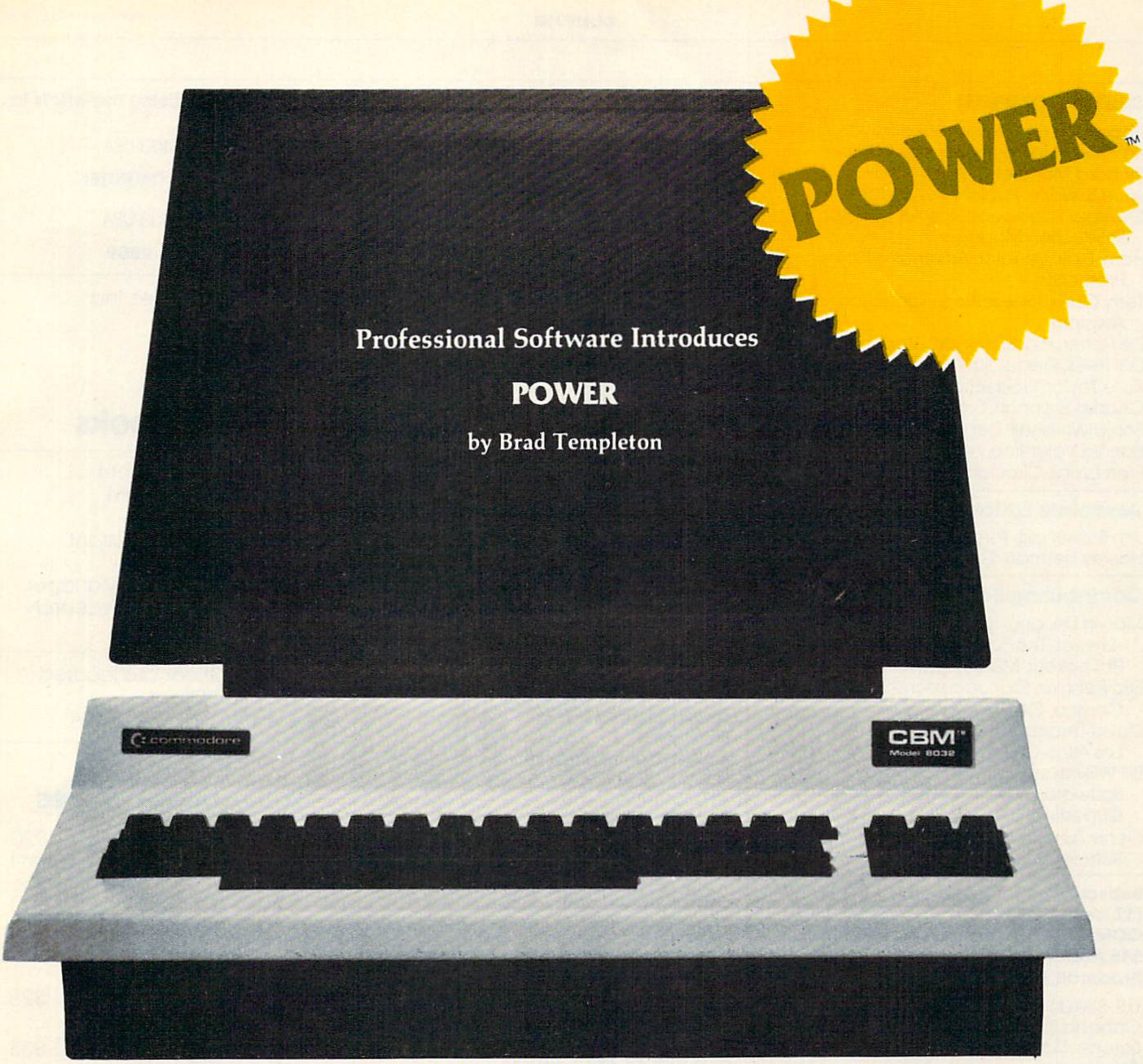
Atari, Inc. has just slashed the suggested retail price of the Atari 800 system from $1,080.00 to $899.00. Commodore is currently introducing two new machines that promise to be quite competitive in the personal market. Shown at the Consumer Electronics Show in Las Vegas the first week in January: a "game" computer with plug-in cartridges and a flat keyboard for around $150.00. And you can add a BASIC cartridge to learn programming. On the "high" end, as it were, and also from Commodore: a 64K color, graphics computer (also for TV connection) said to retail for less than $600.00. And that's with the 64K of memory. Look's like 1982 will surely be an interesting year!.

And As The Industry Grows

In recent editorials, we've commented on software protection, copyrights, the right of back-up, and more. We welcome your thoughts on these and other areas of interest to the personal computer consumer. A letter from a subscriber raised another question that we haven't considered and will put on the 1982 "comments coming" list. I'll raise some of his points here and solicit your input.

> ...I just resisted purchasing an expensive piece of computer software for which the warranty reads in part:
>
> > All...computer programs are distributed on an "as is" basis without warranty of any kind. The entire risk as to the quality and performance of such programs is with the purchaser. Should the programs prove defective...the purchaser and not the manufacturer...assumes the entire cost of all necessary servicing or repair. (The company) shall have no liability or responsibility to a purchaser.
>
> This is not mere legal jargon. It's the embodiment of a business philosophy which seriously harms all of us... To software companies I say: Accept responsibility for your products. Get the bugs out *before* you sell them. Don't try to sell a program debugged by your customers as a "revised" or "improved" product at additional cost.
>
> To software consumers I say: If possible, avoid products for which there is no warranty. Don't buy on faith. Complain loudly to software companies which provide no warranty...

Our reader makes a series of interesting points. While I'm no lawyer, I would wonder if the portions of the warranty shown above are realistic in enforceability. Would some of you lawyer/readers care to join this discussion with the rest of us? I'll look forward to your comments. ©



ADD POWER™ TO YOUR COMMODORE COMPUTER

$89.95

POWER produces a dramatic improvement in the ease of editing BASIC on Commodore's computers. POWER is a programmer's utility package (in a 4K ROM) that contains a series of new commands and utilities which are added to the Screen Editor and the BASIC Interpreter. Designed for the CBM BASIC user, POWER contains special editing, programming, and software debugging tools not found in any other microcomputer BASIC. POWER is easy to use and is sold complete with a full operator's manual written by Jim Butterfield.

POWER's special keyboard 'instant action' features and additional commands make up for, and go beyond the limitations of CBM BASIC. The added features include auto line numbering, tracing, single stepping through programs, line renumbering, and definition of keys as BASIC keywords. POWER even includes new "stick-on" keycap labels. The cursor movement keys are enhanced by the addition of auto-repeat and text searching functions are added to help ease program modification. Cursor UP and cursor DOWN produce **previous** and next lines of source code. COMPLETE BASIC program listings in memory can be displayed on the screen and scrolled in either direction. POWER is a must for every serious CBM user.

Call us today, for the name of the Professional Software dealer nearest you.

Professional Software Inc.
166 Crescent Road
Needham, MA 02194
Tel: (617) 444-5224   Telex #951579

TM POWER is a Registered Trademark of Professional Software, Inc.

Staff of COMPUTE!
Robert C. Lock, Publisher/Editor
Kathleen Martinek, Managing Editor
Richard Mansfield, Assistant Editor
Alice S. Wolfe, Publisher's Assistant
Georgia Papadopoulos, Art Director/ Production Manager
Harry Blair, Director, Advertising Sales/ Promotion
Terry Cash, Typesetting/Production Assistant
De Potter, Production Assistant
Dai Rees, Production Assistant
Kate Taylor, Production Assistant
Charles Brannon, Editorial Assistant
Sonja Whitesell, Dealer Coordinator
Bonnie Valentino, Accounting Coordinator
Fran Lyons, Circulation Coordinator

Associate Editors
Jim Butterfield, Toronto, Canada
Harvey Herman, Greensboro, NC

Contributing Editors
Marvin DeJong, Dept. of Mathematics -Physics, The School of the Ozarks Pt. Lookout, MD 65726
Eric Rehnke, 1067 Jadestone Lane, Corona, CA 91720
David Thronburg, P.O. Box 1317, Los Altos, CA 94022
Bill Wilkinson, Optimized Systems Software, 10379-C Lansdale, Ave. Cupertino, CA 95014
Gene Zumchak, 1700 Niagara St., Buffalo, NY 14207

Subscription Information (12 Issue Year): COMPUTE! Circulation Dept. 515 Abbott Drive Broomall, PA 19008 USA
U.S. $20.00
Canada $25.00 (U.S. funds)
Europe: Surface Subscription, $25.00 (U.S. funds)

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

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

Advertising Sales

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

Northern California, Pacific Northwest, Rocky Mountain States, Texas, Oklahoma
Phoebe Thompson and Associates
101 Church Street, Suite 13
Los Gatos, CA 95030
408-354-5553
PHOEBE THOMPSON

Southern California, Arizona, New Mexico, Nevada
Phoebe Thompson and Associates
P.O. Box 700-462
Redondo Beach, CA 90277
JOANN SULLIVAN
213-378-0143

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

New England, New York State:
The Gittelman Company
Statler Office Building
Suite 518
20 Providence Street
Boston, MA 02116
617-451-0822
JOAN DONAHUE

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

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

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

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

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

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

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

Telephone: 919-275-9809

Small System Services, Inc. publishes:

COMPUTE!
The Journal For Progressive Computing

COMPUTE! Books

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

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

COMPUTE! Subscription Rates

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

Coming In The March Issue of COMPUTE!:

Home Energy Workbook– Third In The Series

Atari Disk Data Structure

PET/CBM Disk Checkout

And Much, Much More...

Ask The Readers

Robert Lock, Richard Mansfield,
And Readers

Please address any questions or answers to: Ask The Readers, **COMPUTE!** *Magazine, P.O. Box 5406, Greensboro, NC 27403.*

Answers

"This is in response to Jerry Stern's question in the August 1981 issue, concerning the use of a keypad controller with the Atari 800. All of Atari's controllers are usable with all of their computers, video game included. And the keypad controller works quite well for repetetive numerical data. However, it is not limited to use with numbers. The computer's response to the keypad controller is defined by the program you write or load into it. Therefore you can use it to output any one of twelve symbols or execute any of twelve commands or any combination of both the above.

There is a program on page H-14 of the Atari Basic Reference Manual *that will get you started with using the controller. The* Operating System Manual *goes into detail on how it works.*

Point of Interest: In [**COMPUTE!** #14] *his excellent article, 'Atari Tape Techniques,' Richard Kruse mentions that the use of LIST "C"/ENTER "C" can reduce the size of your BASIC program, but that the reason for this is undocumented. The documentation is on pages 2 and 3 of the* Basic Reference Manual, *under the section titled 'Variable Name Limit.'"* Roberta L. Mevis

"I enjoyed the article by William Taylor in **COMPUTE!** *#17. Apparently, the Stringy Floppy was interfaced using the regular cassette SAVE and LOAD routines. Your readers might be interested to know that a full ROM-based operating system and plug-in hardware board exist for mating the Stringy Floppy with the OSI C1P. It operates at 7200 BAUD, is available at power-up, and resides in the otherwise-unused memory locations from $E800-$EFFF. Information on the availability of this device may be obtained by writing: MSB Electronics, Barre-Montpelier Road, Barre, Vermont 05641."*
Philip K. Hooper

[On PET/CBM disks] "The ID contained in a track and sector is written once only – when the disk is NEW/ HEADER-ed or created with a BACKUP/DUPLICATE. It can never be written again; disk writes don't touch this special area.

If you wanted to write a non-standard ID into a track-sector header, you'd have to do it by generating your own NEW formatting routine. This is not an easy trick, since it involves downloading a program into the inner processor of the disk. It would almost certainly involve destroying all information written on at least one track of the disk." Jim Butterfield

"In response to Mr. Keplinger's commentary on Computer Assisted Instruction – Worth The Effort?, I say a big positive yes. Going on four years without a promotion in the Air Force, I purchased the Atari 400 computer. I programmed all of my study material multiple choice questions and answers that I could find and had the computer drill me day after day. When it came to test for promotion, it seemed that I knew all the answers. A month later, I was notified of my promotion. Computer assisted instruction really works." Bob Holsti

"Re: Question from John Fry about files in OS65D 3.0 **COMPUTE!** #18.

It seems that although the program example is complete, there was no mention of the creation of a buffer. The program on my C4P MF was completely erased as I knew it would be when I typed it in to confirm my suspicions. Since OSI uses the beginning of the workspace for file buffers, the file is brought on top of the program or portions thereof when the open occurs.

There are many inordinate constructions in OS65D, one of which is the placement of the buffers. I have successfully edited the DOS to place the buffers at the top of memory. The advantage is that I can now easily write programs with sequential and/or random files and need not use the awkward CHANGE utility to create the buffers before I write the program. (This was an enormous help to me as virtually 80% to 90% of the programs I write use random files.)" Ross C. Votaw

Questions

I was delighted with the idea of 'The Unwedge – Tape Append and Renumber' by David Hook in the Sept. '81 issue, p. 103, but ran into problems when I tried it out:
1. The formula in line 26: QV$ = MID$ (STR(4 + 2 (QV = 1)),2). [What can the* QV = 1 *mean?]*
2. [What about the DATA statement in line 116?]..."
John Sweeney

Thanks for the kind words, John. The program was completely rechecked and, indeed, the final number in that DATA line should be H259, not


NASIR GEBELLI PRESENTS:
PATHFINDER
PATHFINDER
Never before released, a super arcade-adventure game for Atari . . . alone in a vast transparent maze inhabited by radioactive monsters and nuclear waste, you must blast the waste and free the maze of mutant monsters. By Solitaire Group, requires 24K Atari with disk drive.
MATCH RACER
MATCH RACER
Brand new for the Atari, this one or two player game tests your driving skills on a race track filled with treacherous obstacles. See how far you can get before your luck and skill run out! By Solitaire Group, requires 16K Atari with disk drive.
ANDROMEDA
ANDROMEDA
Introducing a spectacular, original arcade-adventure for the Atari . . . a battle in inner space . . . you must invade a living organism and destroy its tissues before its antibodies destroy you. By Solitaire Group, requires 16K Atari with disk drive.
HOLLYWOOD
GEBELLI
Gebelli Software Inc.
1771 Tribute Rd., Suite A, Sacramento, CA
Gebelli Software, Inc. Atari is a registered trademark of Atari, Inc.

H25. We make every effort to assure that typos do not get into **COMPUTE!**, and we feel that we have succeeded in eliminating most of the causes. However, publishing 20 to 30 programs each month results in an occasional error. We attempt to announce any corrections (or useful, optional program modifications) the following month in our CAPUTE section in the back of the magazine.

As to the meaning of QV = 1, programmers sometimes choose to use *relational expressions*. Try this in immediate mode: ? 5 = 5. Then try: ? 5 = 2. As you will see, if the proposed equality is true, the "value" of the equality is -1. If false, it's zero.

```
10 INPUT X,Q
20 Y=Y-(X=Q)
30 ?Y:GOTO 10
```

Since subtracting a negative from a positive is, in fact, "addition" – the program above will increase the value of Y whenever X and Q are equal. You could achieve the same result with: 20 IF X = Q THEN Y = Y + 1. Strings can also be used as expressions and evaluated in this way. Likewise, such statements as: IF NOT X THEN PRINT "-1" or IF X THEN PRINT "0" will trigger the THEN action on -1 and zero, respectively.

"I know that there are screendump programs which exist for Atari which will allow the contents of the screen to be put on a printer. What I need is something similar to this but allowing the screen to be saved to tape (or in DATA statements written by the program itself) so that that screen could later be recreated easily. I am currently trying to write a graphics type of Adventure game which uses a redefined character set and requires numerous POSITION and PRINT commands to use those special characters to draw some fairly complicated floorplans. I am nearly at the tearing-out-my-hair point from trying to code these floorplans. It would be much simpler if I could draw the room, using the edit and cursor control functions, and then save screen to be used in the playing of the game itself. Does such a pair of utilities exist or is anyone currently working on one?" Michael A. Ivins

---

"I am an electronics instructor at a technical school and I am interested in programming my own PROMs. We have the KIM-1 at our school and I understand that they can be adapted for this purpose. [Please advise where I can obtain] a schematic and/or instructions."
Mark Iskovitz

"In **COMPUTE!** #16 *you had an article for Applesoft on loading tape. Well, we have an Atari 800, and my husband is having trouble loading and unloading tape, he keeps losing his program... Could you publish something in regards to it?"* Mrs. W. Phipps

Here are some suggestions and precautions: 1. Use the more expensive, better quality tapes. 2. Remember to issue an LPRINT before any CSAVE. 3. Always have two copies (in case one goes bad). 4. Make sure that files are saved with the aid of the digital counter. Don't overlap. 5. When loading, try several times with the tape positioned slightly differently each time (via FFwrd). 6. Clean the tape heads and the rubber wheel that pulls the tape through with cleaning solution and demagnetize the heads with a demagnetizer (both items available at electronics stores). 7. Don't leave the Play or Play and Record buttons down for any long periods. 8. Experiment with alternative ways to SAVE programs: CSAVE/CLOAD, SAVE"C"/LOAD"C", or LIST"C"/ENTER"C". 9. If, after all this, you're still experiencing problems – chances are your heads are misaligned or something else is electronically wrong. Take the recorder in for professional adjustment.

"As owner of a CBM 8032, with 4040 disk drive and an Epson MX-80, I have been reading current and back issues of **COMPUTE!** *ever since my computer dealer introduced me to the magazine several months ago. While I have found much in the magazine of interest, that interest has bordered at times on desperation arising out of statements such as 'This program will not work with the new ROM or with the 80 column screen.'*

How about a program that will rewrite any other program from old or new ROM 40-column to 80-column format, including assembly-language programs? While you are at it, why not a program that will translate from CP/M programs, especially the hundreds of TRS-80 programs, to 'PET BASIC'. I am sure that many of your readers would 'rise up and call you blessed.' "
Dr. Harold Peters

Your suggestion is an excellent one, and we would welcome programs which stand between various machines and translate and harmonize. Unfortunately, writing a program which *emulates* another computer is not an easy task. Some work has been done in this direction, though. In **COMPUTE!** #6 is "Feed Your PET Some Applesoft." Going in the other direction, in **COMPUTE!** #8 is "Load PET Programs Into The Apple II." An extraordinary program which snaps the 80 column CBM screen into 40 columns appears in **COMPUTE!** #12: "Running 40 Column Programs On A CBM 8032." Also, for hand-translations, see **COMPUTE!** #16, "PET, Apple, Atari: On Speaking Terms." Time permitting, we translate individual programs, often presenting multiple versions of programs appearing in the PET Gazette. In addition, we print several programs each month (in the Applications section at the front of the magazine) with versions for both Atari and Microsoft BASICs. ©

Computers And Society

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

On Piracy ...

As the mist cleared we could see our goal before us. High in the foothills overlooking the San Francisco Bay, the opening to the cave was unmarked – except for the power lines which snaked their way into the cavern through a crack in the rock.

"You are on your own now," my guide said as he scurried down the hill. At last I had found the home of the famed software pirate, Long John Silicon.

As I entered the cave, I asked myself why an editor of a prestigious magazine would risk his life in pursuit of a story, but the recent lawsuit preventing Long John Silicon from selling his home video copy of Tooth Fairy was too exciting to ignore.

With a great heave, I opened the door and found Long John sitting at a keyboard, ready for our interview.

DT: Long John, you have a reputation as a vicious software pirate. Tell us – is it deserved?

LJS: Aye matey! I am the meanest software pirate to ply the 57 keys. Once I see a game I like, it is only a matter of time (usually months) before the game is up and running on the computer of your choice.

DT: Wait a minute. I'm not sure I understand what you mean. I thought software pirates just made carbon copies of other people's software.

LJS: Copy existing programs? Ha Ha! Oh matey, you must be kidding! My parrot wouldn't do something that easy. No, what I do is the true pirate's craft. I slink around the arcades looking for new games. When I first saw Tooth Fairy I knew that riches were at hand.

DT: Once you find a game you like, how do you go about copying it?

LJS: First, I spent many pieces of eight playing Tooth Fairy, gaining mastery in every aspect of the game. In the space of a few weeks I was playing the game in my sleep. Next I created a story board for the game.

DT: Excuse me, Long John, I'm not too versed in the pirate's craft. Would you tell our readers what a story board is?

LJS: Of course. A story board is a visual map showing the play of the game. It includes pictures of the screen and so on.

DT: That sounds like a lot of work to go through before writing any of the program.

LJS: Of course it is. Who said piracy is easy work? In any event, once the story board is finished the real work begins. One just doesn't sit down and copy a game without worrying about display resolution, color, machine speed, game controller options – ah, the stories I could tell...

DT: Yes, well I am sure our readers would be fascinated, but tell us more about the game. Is it an exact copy of the arcade version?

LJS: The same? How insulting! I've half a mind to slit you from your index register to your stack! No pirate would miss the chance to improve on a game. To start with, I spent about as much time copying Tooth Fairy as its creator's spent designing it in the first place. Why shouldn't I improve the game.

DT: Oh, I agree with that; but why is this piracy then. After all, people who write love stories aren't being sued by Shakespeare's estate. From what I can see, you might have created a new game.

LJS: No! A thousand times, no. If my version of Tooth Fairy was new, I wouldn't have been sued for infringement by Ajax Computer Company would I?

DT: I guess not. Say, your copy of Tooth Fairy runs on the Ajax computer doesn't it?

LJS: Yes, in fact it has helped sell their computers. Most of the local computer stores used to use my game to show off the Ajax's power.

DT: Then why are they suing you? They didn't write the original software.

LJS: Yes, that's true; but they did buy the home video rights to the game, so I guess I infringed on their copyright, even though they didn't write any of the original program.

DT: Well, I'll bet that their version of Tooth Fairy is a real knockout. Now that your program is illegal, I assume I can buy theirs.

LJS: Oh no! First of all, they have to go through all the work I did to get the game to run on their computer. Their version is at least six months away.

DT: What a shame! Why didn't they just license your copy?

LJS: License me! A pirate! Shiver me disks. Why would they do that? Of course I asked for a license, but they insisted on having the job done over.

DT: I'm not trying to downplay the devious immorality of your crime, but I am still having a hard time thinking of your work as piracy. Tell me, are you countersuing?

LJS: Aye, a countersuit is in progress, but I don't have the legal resources of a company like Giant Toys, Inc.

DT: Oh yes, Giant Toys sells a hand-held version of Tooth Fairy. Is Ajax suing them too?

LJS: Not yet. Ajax only has a few attorneys, and they can't sue everyone at once. I'm flattered that they picked me first, even if I have to give up the battle when my money runs out. After all, even pirates live in fear of their lawyers.

DT: What a shame. You mean that you might lose because you can't afford the fight?

LJS: Yes. After all, if Giant was sued, the case might drag on for years. My suit will probably be mercifully short. My days as a pirate are nearing an end.

DT: So you have given up on piracy forever then?

LJS: Not exactly. Just last night, for example, I used my video cassette recorder to make a copy of a movie that was broadcast past my bedtime. ©

The Beginner's Page

Translating Equations

Richard Mansfield
Assistant Editor

Computers are excellent teachers. They have infinite patience; provide instant pass-fail corrections of your efforts; permit you to work at your own speed on topics of your choice; and they don't (as yet) become sarcastic when you blunder.

Many people, myself included, decided long ago that math was not their forté. This decision is usually made at age fifteen or thereabout and follows a series of mishaps in the educational system. Algebra is often the final blow.

Computers cure this math phobia rather quickly. The machine does all of the tiresome calculations for you. You are free to float above and observe relationships, discover patterns, even construct visual analogs where you can watch the numbers transform on the screen.

Algebraic Equations, BASIC Assignments

After you get over the initial surprise that, in BASIC, A = A + 1 makes perfect sense – you will find that the meaning of *variable* becomes quite clear. A variable is simply a "name" written on a "box." You might have a box in your house marked "BILLS." Each month you pay all the bills and the box is empty (BILLS = 0). Then, when each bill comes in, you put it in the box (BILLS = BILLS + 1). This is not an algebraic equation, it is an *assignment* of a certain number (BILLS + 1) to the variable BILLS.

In algebra, an equation is expected to balance: whatever is on the left side of the equals sign is presumed to be equal to the right side. In BASIC, the variable on the left side is *being defined* by whatever is on the right side. In earlier versions of BASIC, you had to type: LET BILLS = BILLS + 1 to show that you were *assigning* a new value to BILLS, not stating an equality. One other thing: computers allow you to use meaningful, easily recognized variable names such as BILLS, or INTEREST, or DOLLARS. This, too, can be an advantage over the traditional single-letter variable names of algebra.

In any case, much useful math becomes clear after you work a while on your computer. For example, let's put this on our computer (to see how easy it really is):

$$F = D(1 + I/C)^{C*Y}$$

or

$$FINALAMOUNT = DOLLARS(1 + INTEREST/COMPOUNDING)^{COMPOUNDING * YEARS}$$

This formula will let you know how much money you'll end up with after making an investment. It can also tell you how much your house will be worth if it is going up in value a certain amount each year or show the effects of inflation. It's a handy formula, but to the "non-mathematical" it looks forbidding. On the computer, it's a snap. Just use INPUT statements to ask for each of the variables and then, (in line 100), duplicate the formula using BASIC symbols:

```
10 PRINT "WHAT IS THE ORIGINAL AMOUNT IN
     VESTED";
20 INPUT DOLLARS
30 PRINT "HOW MANY YEARS BEFORE YOU CASH
     IN THE INVESTMENT";
40 INPUT YEARS
50 PRINT "WHAT IS THE ANNUAL INTEREST RA
     TE";
60 INPUT ITEREST
70 PRINT "HOW MANY TIMES PER YEAR IS IT
     COMPOUNDED";
80 INPUT COMPOUNDING
90 ITEREST = ITEREST/COMPOUNDING/100:REM
     MAKE ITEREST INTO A DECIMAL FRA
     CTION
100 FINALAMOUNT = DOLLARS * (1 + ITEREST)
     ↑ (COMPOUNDING * YEARS)
110 PRINT "AT THE END OF "; YEARS; "YEARS
     YOU WILL HAVE $"; FINALAMOUNT
```

Notice that we spell it *iterest* to avoid using one of BASIC's special, reserved words *INT*. It is also necessary to enclose *compounding multiplied by years* in parentheses to show that this is to be calculated before the other part is raised to a power. The order in which calculations are performed is, of course, quite important and you should familiarize yourself with what your computer's manual instructs on this subject. When in doubt, use parentheses – they will always cause whatever is within them to be figured first.

The Universal Rounding Engine

Programs can often be refined, customized, and made to perform new functions with surprisingly little effort. This same program could include a function to round off the *finalamount* to the nearest penny by adding this line:

105 FINALAMOUNT = INT(FINALAMOUNT * 100 + .5)/100

What would this Universal Rounding Engine do if you changed the two 100's to 1000's...or 10's? We can also easily adjust the program to predict how much your house will be worth in ten years, given a rise in value of, say, six percent per year. The math stays the same, all we need to do is change the *prompts* (the questions the computer asks). Line 10 should read: "HOW MUCH IS YOUR HOUSE

WORTH NOW?" Line 30: "HOW MANY YEARS DO YOU WANT TO PROJECT?" Line 50: "HOW MUCH IS IT INCREASING IN VALUE EACH YEAR?" Line 70: COMPOUNDING = 1. In line 110, change "YOU WILL HAVE" to "YOUR HOUSE WILL BE WORTH."

To work with inflation projections, make the following replacements:

```
10  "WHAT IS THE COST OF THE ITEM TODAY";
50  "WHAT IS THE ANNUAL INFLATION RATE";
70  COMPOUNDING=1
110 [change YOU WILL HAVE to: IT WILL COST]
```

When creating such useful variations to simple programs, you are, at the same time, learning new things about mathematical relationships. It's fun and therefore painless. As an experiment with the Inflation version of this program, try adding: 5 THISYEAR = 1982 so the computer will know what year it is. Then, using the information gathered in line 30, have the computer give its answer (in line 110) in the form: BY THE YEAR 1985 IT WILL COST $(whatever).

All of this is worlds away from that algebra class where some of us mistakenly decided that mathematics, when it wasn't impossibly obscure, was tedious. By pushing and shaping programs, you can see and feel numbers, their interactions, their beauty. ©

Guest Commentary

Hidden Costs Of Computer Technology

Craig Brod
President, Technostress International, Inc.

During the 1970's most banks computerized their operations. At one California bank, a team was assigned to develop a program to pay savers their interest automatically on the first of each month. The task was completed to everyone's apparent satisfaction and most members of the team were reassigned to new projects. The day before the first automatic payments were to be disbursed, due to a fluke – a favored customer being handed his computerized check a day early – it was discovered that the bank had overpaid everybody as much as double the interest due them. At 6:00 P.M. the team's remaining analyst was called in. The project manager came. His manager came. The vice president came. An estimated eight to ten million dollars in bank funds were on the line, to be disbursed when the doors opened for business the next morning. Could the analyst find the flaw in his team's program? Could he develop an algorithm to withdraw the appropriate amount of overpayment from each of the savers? This analyst was a prime candidate for, if not the victim of, *technostress.*

Exactitude, Repeatability, Detail

Computer technology has become a fact of organizational life and has brought with it new values and new costs for those within the organization. Exactitude, repeatability, and close attention to detail are the hallmarks of everyday operations. Computer technology promotes formal relations between people, their machines, and their environment.

The new technology is qualitatively different from the old. Compared to a computer information system, a telephone or xerox machine were simple communication devices whose use required a minimum of quiet and concentration and whose users had a great deal of latitude. The computer, on the other hand, requires a specific response time from the user (turned *operator*).

The recording of information and the retrieval of data within the language of the program both place constraints on the operator, who has become machine-dependent and works in a captive environment. Control over sound, lighting, and work flow is important for maximum concentration and

"Computer technology often reverses the relationship between age, experience, and competence at work."

effective management of data. While assembly-line work, or even typing, are sometimes grueling types of work requiring attention to detail, the demands on the computer operator are unlike those heretofore known in the workplace.

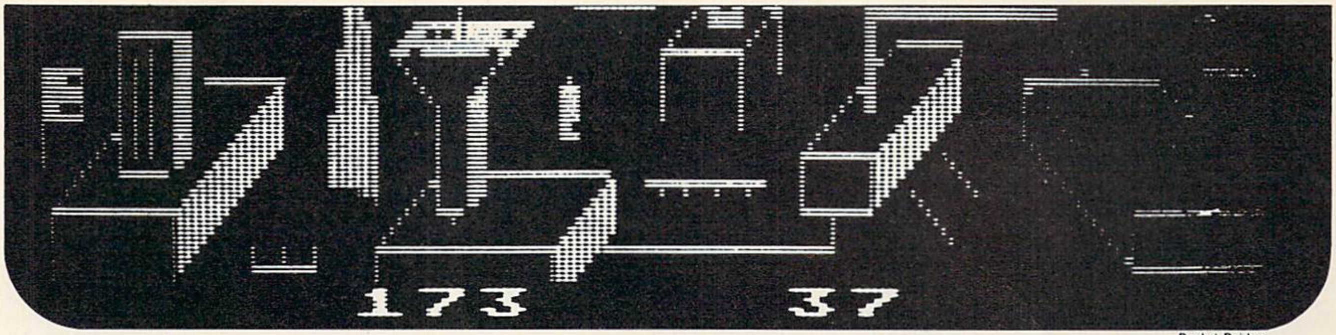
Captive environments and machine-dependent people are indications of new forms of organizational life, and one result is technostress: the condition resulting from the inability of a person or organization to cope with the demands created by the operation and maintenance of computer technology. It occurs where necessary technological stress (such as response to work changes) is translated into unnecessary human strain. There are examples of technostress at all levels.

Age, Experience And Competence: A Reversed Relationship

Computer technology often reverses the relationship between age, experience, and competence at work. Unlike managers of the past who passed tips on to new employees on how to "kick the ditto machine" to make it work, their years of experience have often merely accumulated outdated knowledge in today's managers. And they are usually at a disadvantage to young recruits who command a great deal of recent technical knowhow.

Today's project manager has no reliable way to measure productivity. The manager functions as a go-between, talking to the system user – say the department of a bank that wishes interest payments computerized – and then schedules it, deciding whether it should take six people three months, four people a year, or whatever.

To most programmers, such schedules are a joke: "You could throw darts at a board and do as well," is often heard. One told recently of a project scheduled for three months and which came in at two and a half, earning the group praise; yet it could have been done in three weeks and, that the group dallied, constituted a mini-revolt. No matter; as long as the projects come in ahead of arbitrary schedules, managers are happy. Able to measure only results and not understanding the work well enough to gauge productivity, many of today's managers who have not come up through the



Rocket Raiders

Our High Quality Software Is More Than A Stroke Of Genius... It's A Work Of Art.

□ **PM EDITOR:** by Dennis Zander (Atari, 16K)
Create your own fast action graphics game for the Atari 400 or 800 using its player missile graphics features. By using player data stored as strings, players can be moved or changed (for animation) at machine language speed. All this is done with string variables (POS(Y)=SHIP4). This program is designed to permit creation of up to 4 players on the screen, store them as string data and then immediately try them out in the demo game included in the program. Instructions for use in your own game are included. PM EDITOR was used to create the animated characters in **ARTWORX** RINGS OF THE EMPIRE and ENCOUNTER AT QUESTAR IV.
PRICE .... **$29.95** cassette **$33.95** diskette

□ **ROCKET RAIDERS** by Richard Petersen (Atari 24K)
Defend your asteroid base against pulsar bombs, rockets, lasers, and the dreaded "stealth saucer" as aliens attempt to penetrate your protective force field. Precise target sighting allows you to fire at the enemy using magnetic impulse missiles to help protect your colony and its vital structures.
PRICE .... **$19.95** cassette **$23.95** diskette

□ **INTRUDER ALERT!** by Dennis Zander (Atari, 16K)
This is a fast paced action game in which you must escape from the "Dreadstar" with the secret plans. The droids are after you and you must find and enter your ship in order to escape. If you fail, the rebellion is doomed. **PRICE** .... **$16.95** cassette **$20.95** diskette

□ **THE RINGS OF THE EMPIRE:** by Dennis Zander (Atari 16K)
The Empire has developed a series of battle stations protected by one or more rings of energy. You must destroy these weapons by attacking them in your Y-wing fighter armed with Zydon torpedoes. Each time you blast through the rings and destroy the station, the Empire develops a new station with more protective rings.
PRICE .... **$16.95** cassette **$20.95** diskette

□ **FOREST FIRE!:** by Richard Petersen (Atari, 24K)
Using excellent color graphics, your Atari is turned into a fire scanner to help you direct operations to contain a forest fire. You must compensate for changes in wind, weather and terrain. Not protecting valuable property can result in startling penalties. Life-like variables make FOREST FIRE a very suspenseful and challenging simulation. **PRICE** .... **$16.95** cassette **$20.95** diskette

□ **PILOT:** by Michael Piro (Atari, 16K)
Pilot your small airplane to a successful landing using both joysticks to control throttle and attack angle. PILOT produces a true perspective rendition of the runway, which is constantly changing. Select from two levels of pilot proficiency.
PRICE .... **$16.95** cassette **$20.95** diskette

□ **ALPHA FIGHTER:** by Douglas McFarland (Atari, 16K)
Consisting of two different programs, ALPHA FIGHTER requires you to destroy the alien starships. As you become more successful, the games get harder and harder. **PRICE** .... **$14.95** cassette **$18.95** diskette

□ **GIANT SLALOM:** by Dennis Zander (Atari, 16K)
Bring the Winter Olympics to your computer anytime of the year! Use the joystick to guide your skier's path down a giant slalom course consisting of open and closed gates. Choose from three levels of difficulty. Take practice runs or compete against from two to eight additional skiers.
PRICE .... **$15.95** cassette **$19.95** diskette

□ **HODGE PODGE:** by Marsha Meredith (Apple 48K, Applesoft or Integer BASIC)
This captivating program is a marvelous learning device for children from 18 months to 6 years. HODGE PODGE consists of many cartoons, animations and songs which appear when any key on the computer is depressed. A must for any family containing young children and an Apple.
PRICE .... **$19.95** diskette

□ **STUD POKER:** by Jerry White (Atari, 16K)
This is the classic gambler's card game. You will find the computer to be a worthy opponent who occasionally bluffs but never cheats! STUD POKER employs all of the Atari's sound, color and graphics capabilities.
PRICE .... **$14.95** Cassette **$18.95** diskette

TYPE-'N-TALK*

ARTWORX is offering the fantastic TYPE-'N-TALK™ from Vortrax*. This easy-to-use unit connects to your computer's serial port. Text is automatically translated into electronic speech enabling the TYPE-'N-TALK™ hobbyist to use and enjoy it immediately.
PRICE **$329.00**

The following **ARTWORX** programs are available for TYPE-'N-TALK:
STUD POKER (Atari, 24K) .... **$16.95** cassette **$20.95** diskette
TEACHER'S PET (Atari, 24K; North Star) **$16.95/ $20.95**
BRIDGE 2.0 (Atari, 24K; North Star) .. **$19.95 / $23.95**
NOMINOES JIGSAW (Atari, 24K) ..... **$17.95 / $21.95**
Please specify "TNT" version when ordering programs.

□ **CRANSTON MANOR ADVENTURE:** by Larry Ledden (Atari, North Star and CP/M)
You must enter mysterious Cranston Manor and attempt to collect its many treasures. This extemely challenging program will provide you with many hours (days?) of adventure. The program may be interrupted at will and your status saved onto the diskette.
PRICE .... **$21.95** diskette

□ **BLOCKADE:** by Edward Schneider (Atari, 16K)
Every games library needs Blockade program, and this is one of the best. Choose from three levels of difficulty and play against another person or by yourself against the clock.
PRICE .... **$14.95** cassette **$18.95** diskette

□ **TEACHER'S PET:** by Arthur Walsh (Atari, Apple, TRS-80, PET, North Star and CP/M (MBASIC) systems).
This is an introduction to computers as well as a learning tool for the young computerist (ages 3-7). The program provides counting practice, letter-word recognition and three levels of math skills.
PRICE .... **$14.95** cassette **$18.95** diskette

□ **FORM LETTER SYSTEM:** (Atari, North Star and Apple)
This is the ideal program for creating personalized form letters! FLS employs a simple-to-use text editor for producing fully justified letters. Addresses are stored in a separate file and are automatically inserted into your form letter along with a personalized salutation. Both letter files and address files are compatible with **ARTWORX** MAIL LIST 3.0 and TEXT EDITOR programs.
PRICE .... **$39.95** diskette

□ **TEXT EDITOR:** (Atari and North Star)
This program is very "user friendly" yet employs all essential features needed for serious text editing with minimal memory requirements. Features include common sense operation, two different justification techniques, automatic line centering and straightforward text merging and manipulation. TEXT EDITOR files are compatible with **ARTWORX** FORM LETTER SYSTEM.
PRICE .... **$39.95** diskette

□ **MAIL LIST 3.0:** (Atari, Apple and North Star)
The very popular MAIL LIST 2.2 has now been upgraded. Version 3.0 offers enhanced editing capabilities to complement the many other features which have made this program so popular. MAIL LIST is unique in its ability to store a maximum number of addresses on one diskette (typically between 1200 and 2500 names!). Entries can be retrieved by name, keyword(s) or by zip codes. They can be written to a printer or to another file for complete file management. The program produces 1, 2 or 3-up address labels and will sort by zip code (5 or 9 digits) or alphabetically (by last name). Files are easily merged and MAIL LIST will even find and delete duplicate entries! The address files created with MAIL LIST are completely compatible with **ARTWORX** FORM LETTER SYSTEM.
PRICE .... **$49.95** diskette

□ **THE VAULTS OF ZURICH:** by Felix and Ted Herlihy (Atari, 24K, PET)
Zurich is the banking capital of the world. The rich and powerful deposit their wealth in its famed impregnable vaults. But you, as a master thief, have dared to undertake the boldest heist of the century. You will journey down a maze of corridors and vaults, eluding the most sophisticated security system in the world. Your goal is to reach the Chairman's Chamber to steal the most treasured possession of all: THE OPEC OIL DEEDS!
PRICE .... **$21.95** cassette **$25.95** diskette

□ **BRIDGE 2.0** by Arthur Walsh (Atari (24K), Apple TRS-80, PET, North Star and CP/M (MBASIC) systems)
Rated #1 by Creative Computing, BRIDGE 2.0 is the only program that allows you to both bid for the contract and play out the hand (on defense or offense!). Interesting hands may be replayed using the "duplicate" bridge feature. This is certainly an ideal way to finally learn to play bridge or to get into a game when no other (human) players are available.
PRICE .... **$17.95** cassette **$21.95** diskette

□ **ENCOUNTER AT QUESTAR IV:** by Douglas McFarland (Atari, 24K)
As helmsman of Rikar starship, you must defend Questar Sector IV from the dreaded Zentarians. Using your plasma beam, hyperspace engines and wits to avoid Zentarian mines and death phasers, you struggle to stay alive. This BASIC/Assembly level program has super sound, full player missile graphics and real time action.
PRICE .... **$23.95** cassette **$27.95** diskette

□ **THE NOMINOES JIGSAW PUZZLE:**
by C. Minns/B. Brownlee (Atari, 24K, TRS-80, and Apple)
We quote . . . "A brainteaser supreme. . . the concept of NOMINOES JIGSAW is brillant. . . this video jigsaw game is so clever and completely original that only the most hardhearted puzzle hater could fail to be charmed."—ELECTRONIC GAMES MAGAZINE.
PRICE $17.95 cassette (also available for TRS-80 color computer) **$21.95** diskette.

Artworx
Highest Quality Software*, Guaranteed.
ARTWORX SOFTWARE COMPANY
150 North Main Street Fairport, NY 14450 (716) 425-2833

Call ARTWORX toll-free number to order direct:
800-828-6573 In New York, Alaska, Hawaii call: (716)-425-2833
All orders are processed and shipped within 48 hours.

Shipping and handling charges:
Within North America: Add $2.00
Outside North America: Add 10% (Air Mail)
New York State residents add 7% sales tax.
Quantity Discounts:
Deduct 10% when ordering 3 or more programs

Write for FREE Catalogue listing more information about these and other quality ARTWORX programs.

Ask for ARTWORX at your local computer store.

*ATARI, APPLE, TRS-80, PET, NORTH STAR, CP/M, and TYPE'N TALK are registed tradenames and/or trademarks.

COLLEGE BOARD SAT* PREP SERIES

TRS-80, APPLE, PET, OSI, ATARI, CP/M, PDP-11

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

S.A.T., P.S.A.T., N.M.S.Q.T. — Educator Edition set includes 25 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, and Mathematics. **Independent tests of S.A.T. series performance show a mean total increase of 70 points in students' scores.** Price **$229.95**

GRADUATE RECORD EXAM Series — Educator Edition includes 28 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, Mathematics, Analytical Reasoning and Logical Diagrams. Price **$289.95**

COMPETENCY PROFICIENCY EXAM PREP SERIES

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

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

Special editions available for California Proficiency Assessment Test and New York Regents Competency Tests. Call for Prices.

M.I.T. Logo for Apple $179.95

Odyssey In Time

This spectacular adventure game adds a new dimension of excitement and complexity to **Time Traveler.**

Odyssey In Time includes all the challenges of **Time Traveler** plus 10 additional eras. Each game is different and may be interrupted and saved at any point for later play. **$39.95**

Time Traveler

The best of the adventure games. Confronts the player with complex decision situations and the demand for real time action. Using the Time Machine, players face a challenging series of historical environments. To succeed you must build alliances and struggle with the ruling power. Each game is unique. **$24.95**

☆ NEW ☆
☆MICRO-DEUTSCH☆

Micro-Deutsch set includes 24 grammar lessons, covering all material of an introductory German course. Four test units also included. Grammar lessons use substitution transformation drills, item ordering, translations and verb drills. Drill vocabulary based on frequency lists. Suitable for use with any high school or college textbook. Extensively field tested at SUNY Stony Brook. Available for Apple II and PET/CBM. (PET version includes a special foreign language character chip.) Also available soon: MICRO-FRANCAIS, MICRO-ESPAÑOL, MICRO-IVRIT, MICRO-YIDDISH, MICRO-CHINESE, MICRO-JAPANESE. **$179.95**

Isaac Newton + F.G. Newton

Isaac Newton

Perhaps the most fascinating and valuable educational game ever devised - **ISAAC NEWTON** challenges the players (1-4) to assemble evidence and discern the underlying "Laws of Nature" that have produced this evidence. **ISAAC NEWTON** is an inductive game that allows players to intervene actively by proposing experiments to determine if new data conform to the "Laws of Nature" in question. Players may set the level of difficulty from simple to fiendishly complex.

F.G. Newton

Full Graphics Newton. This version of Isaac Newton presents all data in graphic form. Because data is graphic rather than symbolic, this game is suitable for very young children. Players may select difficulty levels challenging to the most skilled adults. **$49.95**

☆ NEW ☆
☆Pythagoras and The Dragon☆

Mathematics in a fantasy game context. Based on **The Sword of Zedek, Pythagoras and The Dragon** introduces Pythagoras as a mentor to the player. When called on for aid, Pythagoras poses math questions, and depending on the speed and accuracy of the player response, confers secret information. With Pythagoras as an ally, the quest to overthrow Ra, The Master of Evil, assumes a new dimension of complexity. Depending on the level chosen, problems range from arithmetic through plane geometry.

32K **$39.95**

Free Bonus with purchase of $300.00 or more: Applesoft Tutor Series

Krell Software Corp. has no official ties with the College Entrance Examination Board or the Educational Testing Service. Krell is, however, a supplier of products to the E.T.S.

PROGRAMS AVAILABLE FOR TRS-80, APPLE II, PET & ATARI

N.Y.S. residents add sales tax.

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

KRELL SOFTWARE CORP.

"The State of the Art in Educational Computing"

21 Millbrook Drive, Stony Brook, N.Y. 11790 (516) 751-5139

technical ranks lack the respect of their workers.

A Struggle Between Monotony And Perfectionism

To operate a computer is to live with stress, even for relatively low-level operators. Consider the operator whose job it is to process claims and transfer data from one source to another. The machine will have "peaks" and "valleys" – a job requiring thirty minutes at 7:00 A.M. may take several hours at mid-afternoon; and, every so often, due to overload, the whole system *crashes*.

These fluctuations fragment the worker's planning process, his ability to structure his workday. However, the machine makes no mistakes and turns out a uniform product. Given the repetitive nature of his task, the worker struggles between monotony and perfectionism. It is the machine that gets credit for a job well done, and there is no human feedback intrinsic to the system.

Analysts, those who write the programs, are familiar with the dreaded 2:00 A.M. phone call: "It blew up." And the challenge will be to fix it before 6:00 in the morning when perhaps thousands of other workers must depend on it, or, in the case above of the automatic interest payments, eight million dollars may ride on it. It is no surprise that many of them eat Maalox like candy.

An ace analyst is one who has few peers and earns little praise, due to the fact that so few understand what goes into his work. Knowledge builds with the number of systems upon which he has worked. This can have a snowball effect within the organization, always with increasing numbers of people asking "How does this program work?" about increasing numbers of systems. For some, it is a gradual process of becoming identified with the machine – more and more information demanded – with the effect that large chunks of self, time, and energy are drained. Tyrannized by their own expertise, the pressure ceases only when they change jobs. Then they are no longer responsible for every system upon which they have ever worked.

All of these problems are felt, but generally go unstated, surfacing as negative behavior – sabotage, absenteeism, last-minute sick calls, frequent job changes – or as direct problems with productivity. Employees and managers need more technical training, training on how to adapt to new technology, and better work designs.

The power of technology has silenced all but the gallows humor which has grown up around it: "If I hung myself, the machine would just keep plugging away." Technostress manifests itself in a variety of ways. ©

ROCKWELL Microcomputers from Excert, Inc.

THE AIM 65/40
Single Board or Smorgasbord

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

THE AIM 65
Take-Out Order

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

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

32K DRAM Board	ACIA Board	Prototype cards
CRT Controller	IEEE-488Board	Adaptor Buffer Modules
Floppy Disk Controller	CPU/SBC Board	General Purpose I/O Board
PROM Programmer	4-16 Slot Card Cages	PROM/ROM Board

NEW LOWER PRICES AND A CASH DISCOUNT* TO BOOT!

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

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

Mail Order to:

Educational Computer Division
EXCERT INCORPORATED

- SALES
- SERVICE
- INSTALLATION
- CONSULTING

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

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

Insurance Inventory

Robert W Baker
Atco, NJ

This program – versions for Microsoft and Atari BASICs – was written to provide an easy means of maintaining an inventory of personal possessions for insurance or other related purposes. Information is stored on cassette tape for later retrieval and easy, compact storage such as in a safety deposit box.

Running the program is quite simple: to create a new data file simply select that mode and answer the questions concerning the item description, make, model, serial number of other identifying markings, date acquired, and original value. Typing RETURN for any question will automatically enter a question mark for that entry. When all questions are entered, the entire entry will be displayed and you will be asked if it is correct before it is actually written in the data file.

Typing "D" (for DONE) for any entry will abort that entire item entry, close the output file, and return to the program command mode. Typing "E" (for ERROR) will abort the entire item entry and restart it with the first question. Be careful when entering new items into the data file, do not use commas "," or colons ":" to separate words within a description, etc. since BASIC thinks you may be entering more than one string. Use dashes or some other graphic character and play it safe. Avoid using quotes as well, for similar reasons.

A Full Update Capability

To read an already created data file, insert the tape and select that program mode. Three items will be displayed at a time, with all information. Hitting any key except "D" (or RUN/STOP) will display the next three entries. Typing "D" will terminate the read mode, close the input file, and return to the program command mode.

Other program modes are provided to copy or edit the data files produced by this program. The edit mode allows copying or deleting individual entries. You can insert new items at any point. Also, a search feature is included to copy all items until a specific item is found. However, with tape data files, two tape drives are required for these functions for obvious reasons.

All program modes provide file and/or drive selection for ease of use. A default file name of INVENTORY DATA will be generated unless you enter a specific filename. If you should have a large number of items to catalog, you may want to use separate data files for each room, for items acquired each year, specific collections, etc. Program use should be self-evident through prompting instructions displayed by the program. At present, the program does not provide a print option since it was designed for storage of large amounts of personal data. It should be rather easy to add a printing feature if you really think it's necessary.

Describing the actual program is rather difficult since portions of the program are used for every mode. The program flow changes depending on the selected mode and various control flags that are set dynamically during program execution. The program was originally developed on an 8K PET, so I tried to maximize memory usage, allowing more room for the data being generated and used.

Take my word for it, the program does work. If you want to avoid typing in the program, send $2 and an SASE and tape, for a copy on cassette tape. [This is for PET/CBM users only.] For anyone with a 2040 disk, I have another version of this program that uses sequential disk data files. This makes the EDIT mode much more useful. If you do send for a copy on tape, be sure to indicate which version you want. Also, please send all requests directly to me.

Robert Baker
15 Windsor Drive
Atco, NJ 08004

Program 1: Microsoft Version

```
80 POKE 59468,12 :PRINT"{CLEAR}     HOUS
    EHOLD INVENTORY PROGRAM" :GOSUB ~
    1340
90 PRINT"DESIRED PROGRAM MODE:
100 PRINT :PRINT" 0 = DONE
110 PRINT" 1 = READ DATA
120 PRINT" 2 = WRITE NEW DATA FILE"
130 PRINT" 3 = COPY DATA FILE @.
140 PRINT" 4 = EDIT DATA FILE @1@ REQ'S 2
     TAPES
150 GOSUB 1340 :PRINT :PRINT"MODE ?";
160 GOSUB 1440 :IF R$="0" THEN END
170 R=VAL(R$) :IF R<1 OR R>4 THEN 160
180 Z=R :ON R GOTO 400,190,390,390
190 GOSUB 1310 :IF Z>2 THEN T=2 :T$="2" :
    GOTO 220
200 PRINT :INPUT"OUTPUT TO TAPE DRIVE# (1
     OR 2)   2{03 LEFT}";T$
210 T=VAL(T$) :IF T<1 OR T>2 THEN 80
220 PRINT :PRINT"PUT OUTPUT TAPE IN DRIVE
     #";T$ :GOSUB 1390
230 IF F$<>"" THEN 260
240 F$="INVENTORY DATA" :IF Z>2 THEN F$=X
    $
250 PRINT :PRINT"DEFAULT FILENAME = ";F$
```

```
260 PRINT :PRINT"IS OUTPUT TAPE POSITIONE
    D"; :GOSUB 1360 :IF R$="Y" THEN ~
    340
270 PRINT :INPUT"NUMBER OF FILES TO SKIP ~
    "{03 LEFT}";R$:R=VAL(R$):IF R<
    1 THEN 80
280 GOSUB 1310 :PRINT :PRINT"POSITIONING ~
    OUTPUT TAPE
290 OPEN 2,T,Ø
300 GOSUB 1290 :IF C=Ø THEN 300
310 CLOSE 2 :IF C=2 THEN 580
320 R=R-1 :IF R>Ø THEN 290
330 PRINT :PRINT"OUTPUT TAPE IS NOW POSIT
    IONED" :GOSUB 1400 :PRINT
340 OPEN 2,T,1,F$ :IF Z=3 THEN 600
350 IF Z=4 THEN 650
360 GOSUB 940 :IF C>Ø THEN GOSUB 1180 :GO
    TO 360
370 CLOSE 2 :PRINT"{CLEAR}{REV}END OF MOD
    E #2{OFF}  DONE WRITING DATA FIL
    E
380 GOSUB 1340 :GOTO 590
390 GOSUB 1310 :T=1 :T$="1" :GOTO 430
400 PRINT"{CLEAR}{REV}MODE #1{OFF}  READ ~
    DATA FILE FROM TAPE" :GOSUB 1340
410 PRINT :INPUT"INPUT FROM TAPE DRIVE# (
    1 OR 2)   1{03 LEFT}";T$ :T=VAL(
    T$)
420 IF T<1 OR T>2 THEN 80
430 PRINT"{DOWN}PUT INPUT TAPE IN DRIVE #
    ";T$ :GOSUB 1390
440 IF F$=""" THEN F$="" :PRINT :PRINT"RE
    ADING NEXT FILE ON TAPE
450 OPEN 1,T,Ø,F$ :X$=""
460 IF Z>2 THEN 190
470 GOSUB 1220 :IF C>1 THEN 550
480 GOSUB 1140 :IF C>Ø THEN 560
490 GOSUB 1220 :IF C>1 THEN 560
500 GOSUB 1150 :IF C>Ø THEN 560
510 GOSUB 1220 :IF C>1 THEN 560
520 GOSUB 1150 :IF C>Ø THEN 560
530 GOSUB 1350
540 GOSUB 1460 :IF R$<>"D" THEN 470
550 PRINT"{CLEAR}{REV}END OF MODE #1{OFF}
     DONE READING DATA FILE" :PRINT
560 CLOSE 1 :GOSUB 1350
570 IF C=1 THEN PRINT"END OF DATA FILE!
580 IF C>1 THEN PRINT"TAPE READ ERROR ( S
    TATUS ="ST")
590 GOSUB 1400 :GOTO 80
600 I9$="" :GOSUB 1310 :PRINT"{REV}PLEASE
     WAIT{OFF}  ******  COPYING DATA
     FILE!
610 GOSUB 1220 :IF C>1 THEN 860
620 IF Z=4 THEN IF LEFT$(I$,LEN(I9$))=I9$
     THEN GOSUB 1310 :GOTO 660
630 GOSUB 1180 :IF C=1 THEN 860
640 IF Z=3 OR I9$<>"" THEN 610
650 GOSUB 1220 :IF C>1 THEN 860
660 GOSUB 1310 :GOSUB 1150 :GOSUB 1340 :P
    RINT"DESIRED ACTION:" :PRINT
670 PRINT"  1 = COPY THIS ITEM, NO CHANGE
680 PRINT"  2 = DELETE THIS ITEM
690 PRINT"  3 = INSERT ITEMS BEFORE THIS ~
    ONE
700 PRINT"  4 = SEARCH & COPY TILL ITEM F
    OUND" :PRINT
710 PRINT"ACTION ? ";
720 GOSUB 1440 :R=VAL(R$) :IF R<1 OR R>4 ~
    THEN 720
730 PRINT R$
740 PRINT"OK" :I9$="" :ON R GOTO 630,750,
    770,800
750 IF C=1 THEN 860
760 GOTO 650
770 I9$=I$ :W9$=W$ :M9$=M$ :S9$=S$ :D9$=D
    $ :V9$=V$ :C9=C
780 GOSUB 940 :IF C>Ø THEN GOSUB 1180 :GO
    TO 780
790 I$=I9$ :W$=W9$ :M$=M9$ :S$=S9$ :D$=D9
    $ :V$=V9$ :C=C9 :GOTO 660
800 GOSUB 1310 :PRINT"ALL ENTRIES WILL BE
     COPIED UNTILL
810 PRINT :PRINT"DESIRED ITEM IS FOUND;
820 PRINT"{02 DOWN}ENTER ITEM TO SEARCH F
    OR:
830 INPUT"    "{03 LEFT}";I9$
840 IF I9$=""" THEN I9$="" :PRINT"{03 DOW
    DOWN}SEARCH ABORTED" :GOTO 660
850 PRINT"{03 DOWN}SEARCHING" :GOTO 620
860 IF Z=3 THEN 570
870 GOSUB 1310 :IF C>1 THEN 580
880 PRINT"END OF INPUT FILE!
890 PRINT :PRINT"DO YOU WANT TO ADD ANY E
    NTRIES TO THE
900 PRINT :PRINT"END OF THE DATA FILE";
910 GOSUB 1360 :IF R$="N" THEN 590
920 GOSUB 940 :IF C>Ø THEN GOSUB 1180 :GO
    TO 920
930 GOSUB 1310 :GOTO 590
940 C=Ø :PRINT"{CLEAR}ENTER ITEM INFORMAT
    ION:" :PRINT
950 PRINT"D = DONE ENTERING DATA
960 PRINT"E = ERROR, RESTART ENTIRE ITEM
970 PRINT :PRINT"DO NOT USE ',' OR ':' WI
    THIN THE DATA
980 PRINT :PRINT"PRESS {REV}RETURN{OFF} A
    FTER EACH ENTRY
990 GOSUB 1340 :INPUT"{REV}ITEM{OFF}    ?{
    Ø4 LEFT}";I$ :IF I$="E" THEN 940
1000 IF I$="D" THEN RETURN
1010 INPUT"{REV}MAKE{OFF}    ?{Ø3 LEFT}";W$
     :IF W$="E" THEN 940
1020 IF W$="D" THEN RETURN
1030 INPUT"{REV}MODEL{OFF}    ?{Ø3 LEFT}";M
     $ :IF M$="E" THEN 940
1040 IF M$="D" THEN RETURN
1050 INPUT"{REV}SERIAL#/ID{OFF}    ?{Ø3 LEF
     LEFT}";S$ :IF S$="E" THEN 940
1060 IF S$="D THEN RETURN
1070 INPUT"{REV}DATE ACQ'D{OFF} (MONTH/DAY
     /YEAR)    ?{Ø3 LEFT}";D$ :IF D$="
     E" THEN 940
1080 D$=LEFT$(D$,8) :IF D$="D" THEN RETURN
1090 INPUT"{REV}$VALUE{OFF}    ?{Ø3 LEFT}";
     V$ :IF V$="E" THEN 940
1100 IF V$="D" THEN RETURN
```

```
1110 GOSUB 1140 :GOSUB 1340
1120 PRINT"IS THIS ENTRY CORRECT"; :GOSUB ~
     1360 :IF R$="N" THEN 940
1130 C=1 :RETURN
1140 PRINT"{CLEAR}";
1150 PRINT"{REV}ITEM:{OFF} ";I$ :PRINT"{RE
     REV}MAKE:{OFF} ";W$
1160 PRINT"{REV}MODEL:{OFF} ";M$ :PRINT"{R
     REV}SERIAL#/ID:{OFF} ";S$
1170 PRINT"{REV}DATE ACQ'D:{OFF} "D$;TAB(2
     2);"{REV}VALUE:{OFF} $";V$ :PRIN
     T :RETURN
1180 X$=I$ :GOSUB 1200 :X$=W$ :GOSUB 1200 ~
     :X$=M$ :GOSUB 1200
1190 X$=S$ :GOSUB 1200 :X$=D$ :GOSUB 1200 ~
     :X$=V$
1200 PRINT#2,X$
1210 RETURN
1220 GOSUB 1290 :I$=X$ :IF C>0 THEN RETURN

1230 GOSUB 1290 :W$=X$ :IF C>0 THEN RETURN

1240 GOSUB 1290 :M$=X$ :IF C>0 THEN RETURN

1250 GOSUB 1290 :S$=X$ :IF C>0 THEN RETURN

1260 GOSUB 1290 :D$=X$ :IF C>0 THEN RETURN

1270 GOSUB 1290 :V$=X$ :IF C=2 THEN C=1
1280 RETURN
1290 C=0 :INPUT#1,X$ :IF ST>0 THEN C=3 :IF
      ST=64 THEN C=2
1300 RETURN
1310 IF Z=2 THEN PRINT"{CLEAR}{REV}MODE #2
     {OFF}  WRITE NEW DATA FILE ON TA
     PE
1320 IF Z=3 THEN PRINT"{CLEAR}{REV}MODE #3
     {OFF}  COPY DATA FILE, REQ'S 2 T
     APES
1330 IF Z=4 THEN PRINT"{CLEAR}{REV}MODE #4
     {OFF}  EDIT DATA FILE, REQ'S 2 T
     APES
1340 PRINT
1350 PRINT"CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
     CCCCCCCC" :PRINT :RETURN
1360 PRINT" (Y/N) ? ";
1370 GOSUB 1440 :IF R$<>"Y" AND R$<>"N" TH
     EN 1370
1380 PRINT R$ :RETURN
1390 INPUT"{DOWN}FILENAME < {REV}RETURN{OF
     OFF} >    "{03 LEFT}";F$ :RETURN
1400 CLOSE 1 :CLOSE 2
1410 IF Z>2 THEN PRINT :PRINT"DEPRESS {REV
     REV}STOP{OFF} ON BOTH TAPE DRIVE
     S" :GOTO 1430
1420 PRINT :PRINT"DEPRESS {REV}STOP{OFF} O
     N TAPE #"T$
1430 PRINT"{DOWN}HIT ANY KEY WHEN READY TO
      CONTINUE"; :GOTO 1470
1440 GET R$ :IF R$="" THEN 1440
1450 RETURN
1460 PRINT :PRINT"HIT ANY KEY TO CONTINUE,
      D=DONE";
1470 GOSUB 1440 :PRINT :PRINT"OK" :RETURN
```

Program 2: Atari Version

```
10 REM INSURANCE INVENTORY
20 REM ROBERT W. BAKER ATCO, NJ
30 REM ATARI VERSION (C) 1981
40 REM SMALL SYSTEMS SERVICES, INC.
50 REM
60 OPEN #2,4,0,"K:"
70 DIM T$(1),X$(80),I$(80),W$(80),M$(80)
,S$(80),D$(8),V$(20)
80 GRAPHICS 0:? "     INSURANCE INVENTORY
 PROGRAM  "
90 PRINT "Desired Program mode:"
100 ? :? "0 = Done"
110 ? "1 = Read Data"
120 ? "2 = Write New Data File"
150 GOSUB 1340:? :? "Mode? ";
160 GOSUB 1440:IF R=48 THEN END
170 R=R-48:IF R<1 OR R>2 THEN 160
180 Z=R:ON R GOTO 400,190
190 ? "{CLEAR}| MODE #2 | WRITE NEW DATA
 FILE"
200 TRAP 200:? "{DOWN}Number of files to
 skip";:INPUT R:TRAP 40000:IF R<1 THEN 3
40
210 ? "{DOWN}Put tape in drive, press PL
AY"
220 ? "then press |RETURN|."
230 IF PEEK(764)=255 THEN 230
270 ? "{CLEAR}| MODE #2 | WRITE NEW DATA
 FILE"
280 ? "{DOWN}Positioning output tape..."

290 TRAP 310:OPEN #1,4,0,"C:"
300 GOSUB 1290:IF C=0 THEN 300
310 CLOSE #1:IF C=2 THEN 580
320 R=R-1:IF R>0 THEN 290
330 ? :? "Output tape is now positioned.
{DOWN}":GOSUB 1400:?
340 ? "Press PLAY & RECORD, press |RETUR
N|."
350 OPEN #1,8,0,"C:"
355 FOR I=1 TO 128:PUT #1,32:NEXT I
360 GOSUB 940:IF C>0 THEN GOSUB 1180:GOT
O 360
370 CLOSE #1:? "{CLEAR}|END OF MODE #2|
 Done writing data file"
380 GOSUB 1340:GOTO 590
400 ? "{CLEAR}|MODE #1|  READ DATA FILE
FROM TAPE":GOSUB 1340
430 ? "{DOWN}Put input tape in drive, pr
ess |PLAY|,  then press |RETURN|."
450 TRAP 560:OPEN #1,4,0,"C:":X$="":FOR
I=1 TO 128:GET #1,R:NEXT I:TRAP 40000
460 ? "{DOWN}Reading next file on tape."

470 GOSUB 1220:IF C>1 THEN 550
```

```
480 GOSUB 1140:IF C>0 THEN 560
490 GOSUB 1220:IF C>1 THEN 560
500 GOSUB 1150:IF C>0 THEN 560
510 GOSUB 1220:IF C>1 THEN 560
520 GOSUB 1150:IF C>0 THEN 560
530 GOSUB 1350
540 GOSUB 1460:IF R<>ASC("D") THEN 470
550 ? "{CLEAR}|END OF MODE #1| DONE READ
ING DATA FILE":?
560 CLOSE #1:GOSUB 1350
570 IF C=2 THEN ? "End of data file!":C=
1
580 IF C>1 THEN ? "TAPE READ ERROR-";PEE
K(195)
590 GOSUB 1400:GOTO 80
940 C=0:? "{CLEAR}Enter item information
:":?
950 ? "D = Done entering data"
960 ? "E = Error, restart entire item"
980 ? :? "Press |RETURN| after each entr
y."
990 GOSUB 1340:? "|ITEM|";:INPUT I$:IF I
$="E" THEN 940
1000 IF I$="D" THEN RETURN
1010 ? "|MAKE|";:INPUT W$:IF W$="E" THEN
 940
1020 IF W$="D" THEN RETURN
1030 ? "|MODEL|";:INPUT M$:IF M$="E" THE
N 940
1040 IF M$="D" THEN RETURN
1050 ? "|SERIAL#/ID|";:INPUT S$:IF S$="E
" THEN 940
1060 IF S$="D" THEN RETURN
1070 ? "|DATE ACQ'D| (MONTH/DAY/YEAR) ";
:INPUT D$:IF D$="E" THEN 940
1080 IF D$="D" THEN RETURN
1090 ? "|$VALUE| ";:INPUT V$:IF V$="E" T
HEN 940
1100 IF V$="D" THEN RETURN
1110 GOSUB 1140:GOSUB 1340
1120 ? "Is this entry correct?";:GOSUB 1
360:IF R=ASC("N") THEN 940
1130 C=1:RETURN
1140 ? "{CLEAR}";
1150 ? "|ITEM|: ";I$:? "|MAKE|: ";W$
1160 ? "|MODEL|: ";M$:? "|SERIAL#/ID|: "
;S$
1170 ? "|DATA ACQ'D|: ";D$,"|VALUE|: ";V
$:? :RETURN
1180 X$=I$:GOSUB 1200:X$=W$:GOSUB 1200:X
$=M$:GOSUB 1200
1190 X$=S$:GOSUB 1200:X$=D$:GOSUB 1200:X
$=V$
1200 PRINT #1;X$
1210 RETURN
1220 GOSUB 1290:I$=X$:IF C>0 THEN RETURN
1230 GOSUB 1290:W$=X$:IF C>0 THEN RETURN
1240 GOSUB 1290:M$=X$:IF C>0 THEN RETURN
1250 GOSUB 1290:S$=X$:IF C>0 THEN RETURN
1260 GOSUB 1290:D$=X$:IF C>0 THEN RETURN
1270 GOSUB 1290:V$=X$:IF C=2 THEN C=1:ST
OP
1280 RETURN
1290 TRAP 1300:C=0:INPUT #1,X$:TRAP 4000
0:RETURN
1300 C=3:IF PEEK(195)=136 THEN C=2:RETUR
N
1340 ?
1350 ? "{38 R}":RETURN
1360 ? " (Y/N) ? ";
1370 GOSUB 1440:IF R<>ASC("Y") AND R<>AS
C("N") THEN 1370
1380 ? CHR$(R):RETURN
1400 CLOSE #1:? "Depress |STOP| on tape
drive."
1410 ? :? "Press any key to continue..."
;
1440 GET #2,R:RETURN
1460 ? :? "HIT ANY KEY TO CONTINUE, D=DO
NE"
1470 GOSUB 1440:? "OK":RETURN
```

©

Creating A Simple Word Processor

Steve Gradijan
Carrollton, TX

Editor's Note: With minor adjustments, this simple (but effective) word processor will work on Apple, OSI, – any Microsoft BASIC. Described here for the PET/CBM, the author points out which lines to change for other machines. The program is well documented to permit easy adjustment to a variety of printer, disk/tape, etc. configurations. — RTM

Arnie Lee's LED a Line-oriented Text Editor described in **COMPUTE!** #9 can become a moderately sophisticated word processor with the addition of a few lines. Line 5 enables upper and lower case features of the PET; line 8075 disengages these features and returns the PET to normal mode. Minor modification to line 9050 eliminates the printing of the LED's line oriented identification numbers and allows Commodore printers to print both upper and lower case. The addition of the control character to the string referred to in this line, however, uses one position of the 80 character string limiting the text part of the string to 79 characters, including spaces. Thus, Mr. Lee's 80 character string is shortened in line 10055.

Adding Versatility

Additional program lines give the word processor more versatility. Tab functions or line indentation and simulated line feed are accomplished by lines 145, 146, and 10045. Lines 9025 and 9055 provide the option for double spaced print of copy to the printer.

Lines 235 and 21000 to 21100 set the margins. If the margins are not set at the beginning of a typing session, the program defaults to a 79 character line i.e. no margins. Lines 500, 520, 530, and 570 are modified to accomodate the additional command "s," set margins.

A "bell" is provided to prompt you when only five spaces are left in a line (lines 22000 to 22040). It makes use of the "CB-2 sound" provided at the user port and requires connection of a suitable amplifier/speaker to the PET.

Lines 5 and 8075 are not necessary with the Commodore CBM's. The upper-lower case and bell functions use POKE statements not compatible with other computers. However, all other modifications should be usable with machines using Microsoft BASIC.

Options And Commands

Additional commands now available include "s" which allows setting margins. "n" establishes a 79 character line and no margins; "s" creates five character wide left and right margins and a 69 character wide field; "m" increases the size of the margins to ten characters and the "l" to fifteen characters in width. "o" allows creation of your own margins, both left and right. You are asked to specify the length of both the left and right margins. If you forget to set the margins at the beginning of the program the margins will default leaving you with a 79 character line. Once set, margins may later be lengthened, but never shortened!

The sub-commands "@" and "@" + RETURN provide tab functions and line skipping. The sub-command "@" adds five spaces to the text string and is useful as a tab or an indent. It may be used at any time while in a)ppend, i)ndent, or r)eplace functions. To skip a line of text, type "@" followed by the RETURN key.

The option to print the text either single or double spaced is given after requesting the print function p)rint.

The PET keyboard will behave like a normal typewriter after modifying the LED program. The shift key will provide upper case. All punctuation and designators supplied by the PET character set are available except @, which is used for tab functions. Quotation marks are permitted, but they look a little strange when first seen on the CRT screen of the PET. Ignore the funny appearance of anything that you enclose in quotes and depend upon the "bell" to determine the end of the line. Everything within quotes will appear normal when later listed, printed, or saved!

Delete lines 9070 through 9090 from Arnie Lee's original program and you are ready to type a letter, an order, an article for **COMPUTE!** or whatever.

```
0:LIST
 0 REM LINE EDITOR (C)1980 ABACUS ~
    SOFTWARE
 3 REM   FROM   FEBRUARY 1981 COMP
    UTE
 4 REM   MODIFIED BY STEVE GRADIJA
    N, CARROLLTON, TEXAS
 5 POKE59468,14:REM  ENABLE UPPER ~
    & LOWER CASE
 10 DIMT$(500):REM BUFFER SPACE
 20 L$="":REM  CURENT LINE
 30 LL=1:REM LAST LINE #
```

JINSAM

Data Manager selected by NASA, Kennedy Space Center

SAVE TIME! SAVE MONEY!
JINSAM saves taxpayers 100's of thousands of $

OFFICIALLY APPROVED BY COMMODORE

"Much more powerful than you can imagine!"
Robert Baker
Kilobaud Microcomputing

JINSAM is an integrated system. It makes it easy to use your information to its fullest. No more will hundreds of valuable hours be spent searching or analyzing needed information nor re-entering information for various reports.

JINSAM transforms your desk-top computer into the "state of the art" data processing machine with features and accessories found nowhere, even at 10 times the price. NASA, Kennedy Space Center selected JINSAM 8.0 and saved approximately $95,000 over other software/hardware costs. Riley County, Kansas also selected JINSAM 8.0 and saved approximately $90,000 over other software/hardware costs.

JINSAM is designed for you. It is forgiving. It has help commands for every option, available at the touch of a button. The amount of information you store, its structure and/or your hardware can change but your data won't have to be re-entered. Recovery utilites are included even for catastrophes, security passwords are built in for privacy, simple editing and entry includes auto recall, and deleting records is easy and the space is reclaimed. JINSAM includes TWO FREE accessories for reports and labels. You have unlimited report formats with summing and lined up decimals and the label printer prints up to 5 across – any size and even allows single envelopes or index cards.

JINSAM 1.0 allows fast and easy file handling, manipulation and report generation for any CBM computer with CBM 2040 disk drive. It features a menu for ease, has encrypted passwords, 3 deep sorts, .5 to 3 second recall.

JINSAM 4.0 for CBM 4000 series adds JINSORT, user accessible machine sort of 1000 records in 15 seconds; compaction/expansion of information, automatic list maintainance, unlimited number of fields, unlimited record length and much more.

JINSAM 8.0 for CBM 8000 series has all 4.0 features plus unlimited sort, horizontal format, and search by key or record number.

JINSAM 8.2, NEW FOR '82 expands 8.0 capabilities by adding information search by word, key or record number and machine language print, format and manipulation routines.

- ★ CUSTOM DATA FILES
- ★ CUSTOM REPORTS/LABELS
- ★ KEYED RANDOM ACCESS
- ★ FAST/EASY/MENU DRIVEN
- ★ MULTIPLE SEARCH KEYS
- ★ PRIVACY ACCESS CODES
- ★ WILD CARD SEARCH

JINSAM EXECUTIVE version (soon to be released) is our most powerful professional system for the CBM 8000 and 9000 series. Executive will have 8.2 extended features plus allow multiple users with in-use lockout protection, executive command files, automatic math relations, join, merge or link files, greatly increased record capacity and machine information search by word, as well as by key or record number and many, many more features.

There are currently 7 more interfacing modules – and more under development, including independent interfaces between JINSAM and business packages for your G/L, A/R, A/P needs. We announce the availability of modules and enhancements in JINSAM's quarterly newsletter.

WORDPROPACK - Intelligent interface for WordPro 3, 3+, 4, 4+, creates lists of information from JINSAM files. It allows up to 10 conditions based on each item of information. Produce individualized letters, report cards, special reports, checks, invoices, etc.

MULTI-LABEL - Prints multiple labels per record with up to 2 lines for messages and consecutive numbering. Produce inventory, caution labels, bulk mail labels, etc.

MATHPACK - global calculator/editor +, −, *, /, by another field or constant; null (remove contents) of a field or replace contents of a field with any word, number or phrase. Sum multiple fields in each record or running sum of single field in all records. Extract information or effect permanent change. Replace in the same field or place in a waiting field.

DESCRIPTIVE STATPACK - Determine MEAN, MEDIAN, MODE, STANDARD DEVIATION, VARIANCE, RANGE. Generate HISTOGRAMS from 1 to 25 steps, and produce Z-SCORE reports.

ADVANCED STATPACK - (You must also acquire DESCRIPTIVE STATPACK) Generate one, two or three way CROSSTABS (number of occurance) CHI SQUARE, LINEAR REGRESSION, with graphic representation and prediction, LINEAR CORRELATION and SIMPLE ANALYSIS OF VARIANCE.

CALCPACK - 2 way interface to VisiCalc or any user program. It lets you use VisiCalc for complex manipulation, editing, placing results in JINSAM for sorting, storing or moving data to WordPro as well as giving the ability for exchange with your own applications.

INTERAC - Interface which can read VisiCalc files, WordPro files and almost any sequential files to build JINSAM databases automatically. For example: You could "download" information on holerith cards to sequential files and INTERAC would place them into JINSAM files.

All accessories are accessed thru the JINSAM menu and require security password to gain entrance.

JINSAM gives you FREEDOM OF CHOICE. Start with JINSAM 1.0 and upgrade hardware and data at any time. Choose from accessories at any time. The JINSAM Newsletter brings the latest updates, user input and uses and keeps an eye on the future.

JINSAM stands alone by placing "a lot of potential computing power in one integrated program package" (Fred Klein, Ferson div. of Bausch & Lomb). "The JINSAM package is justification for buying a system no matter what the hardware, be it Vector or Commodore or whatever the system" (Larry Colvin, Micro Computer Systems). It is a "grandfather" in this young field. JINSAM EXECUTIVE will be the third generation in development. All JINSAM systems are sophisticated and flexible yet easy to use. JINSAM is saving its users valuable time and money in government, educational and research institutions, business and industry worldwide.

JINSAM is a Commodore approved product. See your local dealer for a demonstration.

JINSAM is a trademark of JINI MICRO-SYSTEMS, Inc.
WordPro is a trademark of Professional Software, Inc.
VisiCalc is a trademark of Professional Software, Inc.
CBM is a trademark of Commodore Business Machines, Inc.

JINSAM Data Manager

.... Additional Information and nearest dealer
.... Newsletter Subscription ($5 US/$8 Worldwide)
.... JINSAM Demo Disk ($15, plus shipping & tax)
.... User's Guide ($40.00, plus shipping & tax)

Please send to:

Name ____________________
Position ____________________
Company ____________________
Address ____________________
City, State, Zip ____________________
Phone (   ) ____________________
Computer, Disk ____________________
Application ____________________

JINI MICRO-SYSTEMS, Inc.
Box 274C • Kingsbridge Station
Riverdale, New York 10463
(212) 796-6200

Dealer Inquiry Welcome

```
40 SP$=" ":DL$=CHR$(20)
45 M=79:REM DEFAULT TEXT STRING LE
   NGTH
50 EE=0:REM DISKERRROR CHANNEL CLO
   SED
60 PR=0
90 REM POKE144,49:REM DISABLE STOP
    KEY
100 PRINT"{CLEAR}      {REV}ABACUS S
    OFTWARE LINE EDITOR"
110 PRINT"{02 DOWN}    FUNCTIONS:"
130 PRINT
140 PRINTTAB(8);"A)PPEND-TO END OF ~
    TEXT"
145 PRINTTAB(10);"@ = 5 SPACE TAB"
146 PRINTTAB(10);"@+{REV}RETURN{OFF
    OFF} = SKIP LINE
150 PRINTTAB(8);"C)HANGE-STRING
160 PRINTTAB(8);"D)ELETE LINE(S)
170 PRINTTAB(8);"F)ILER COMMANDS
180 PRINTTAB(8);"I)NSERT BEFORE LIN
    E
190 PRINTTAB(8);"L)IST LINE(S)
200 PRINTTAB(8);"M)ENU DISPLAY
210 PRINTTAB(8);"P)RINT LINES(S)
220 PRINTTAB(8);"Q)UIT LINE EDITOR
230 PRINTTAB(8);"R)EPLACE LINE
235 PRINTTAB(8);"S)ET MARGINS
240 PRINT:PRINT"    ENTER SELECTION
    -> ";
250 GOTO510
500 PRINT:PRINT"{REV}ENTER{OFF} A,C
    ,D,F,I,L,P,Q,R,S,M)ENU->";
510 GETA$:IFA$=""THEN510
520 J=0:FORI=1TO11
530 IFA$=MID$("ACDFILRMQPS",I,1)THE
    NJ=I:I=11
540 NEXTI
550 PRINTA$
560 IFJ=0 THEN500
570 ONJGOTO1000,2000,3000,4000,5000
    ,6000,7000,100,8000,9000,2
    1000
1000 PRINT
1005 PRINT"{REV}APPEND{OFF} TO END O
    F TEXT"
1010 PRINT:PRINT LL">";
1020 GOSUB10000
1030 IFLEN(L$)=0THEN500
1040 T$(LL)=L$
1050 LL=LL+1
1060 GOTO1010
2000 REM CHANGE STRING
2010 PRINT:PRINT"{REV}CHANGE{OFF}";:
    GOSUB16000
2020 IFHI=0THEN500
2025 PRINT"{REV}CHANGE{OFF} STRING->
    ";:GOSUB10000
2030 L=LEN(L$)
2040 IFL=0THEN500
2050 IFL<4THEN2000
2060 DM$=LEFT$(L$,1):REM DELIMITER
2070 IFRIGHT$(L$,1)<>DM$THEN2000
2080 J=0:FORI=2TOL-1
2090 IFMID$(L$,I,1)=DM$THENJ=I
2100 NEXTI
2110 IFJ=0THEN2000
2120 IFJ=2THEN2000
2130 FR$=MID$(L$,2,J-2)
2140 IFJ+1=LTHENTS$="":GOTO2160
2150 TS$=MID$(L$,J+1,L-J-1)
2160 F=LEN(FR$)
2170 FORI=LOTOHI
2180 T=LEN(T$(I)):S=1:NL$=""
2190 FORJ=1TOT-F+1
2200 IFMID$(T$(I),J,F)<>FR$THEN2230
2210 NL$=NL$+MID$(T$(I),S,J-S)+TS$
2220 S=J+F:J=S-1
2230 NEXTJ
2240 IFS<>1THENNL$=NL$+RIGHT$(T$(I),
    T-S+1):T$(I)=NL$
2250 NEXTI
2260 GOTO500
3000 REM DELETE LINES
3005 PRINT:PRINT"{REV}DELETE{OFF} ";
    :GOSUB16000:REM GET RANGE
3010 IFNOTDFTHEN3015:REM NOT DEFAULT
     ON ENTIRE FILE
3011 PRINT"{REV}DELETE{OFF} ENTIRE F
    ILE?  ";
3012 GETA$:IFA$=""THEN3012
3013 PRINTA$:IFA$="N"THEN500
3014 IFA$<>"Y"THEN3011
3015 IFHI>LL-1THEN500
3020 IFHI=LL-1THENLL=LO:GOTO500
3030 J=HI-LO+1
3040 FORI=LOTOLL-J-1
3050 T$(I)=T$(I+J)
3060 NEXTI
3070 LL=LL-(HI-LO)-1
3080 GOTO500
4000 REM FILLER
4010 PRINT"{DOWN}{REV}FILER{OFF} ENT
    ER L)OAD OR S)AVE-> ;
4020 GETA$:IFA$=""THEN4020
4030 IFA$<>"L"ANDA$<>"S"THENPRINT:GO
    TO4000
4040 PRINTA$:M$=A$
4050 PRINT"{REV}ENTER{OFF} FILENAME-
    > ";
4070 GOSUB10000
4075 IFLEN(L$)=0THEN500
4076 IFLEN(L$)>12THEN4050
4080 FI$=L$
4090 PRINT"{REV}ENTER{OFF} D)ISK OR ~
    T)APE-> ";
4100 GETA$:IFA$=""THEN4100
4110 PRINTA$
4120 IFA$<>"D"ANDA$<>"T"THEN4090
4130 IFA$="D"THEN4160
4140 IFM$="L"THEN4400
4150 GOTO4200
4160 DR$="":IFLEFT$(FI$,2)<>"0:"ANDL
    EFFT$(FI$,2)<>"1:"THENDR$=
    "0:"
4170 GOTO4600
4200 REM TAPE SAVE
4210 IFLL=1THENPRINT"NO FILE TO SAVE
    ":GOTO500
4220 OPEN2,1,2,FI$+".SOURCE"
```

```
4230 FORI=1TOLL-1
4240 FORJ=1TOLEN(T$(I))
4250 PRINT#2,MID$(T$(I),J   ,1);
4260 NEXTJ
4270 PRINT#2,CHR$(255);
4280 NEXTI
4290 CLOSE2
4300 PRINTSPC(6);FI$;"  SAVED"
4310 GOTO500
4400 REMTAPE LOAD
4410 OPEN2,1,0,FI$+".SOURCE"
4430 LL=0:REMLINE COUNT
4440 LL=LL+1:T$(LL)=""
4450 GET#2,A$
4460 IFST=64THEN4500:REM END OF FILE
4465 IFST<>0THEN PRINT"*** LOAD ERRO
   R ***":GOTO500
4470 IFA$=CHR$(255)THEN4440:REM END ~
   OF LINE
4480 T$(LL)=T$(LL)+A$
4490 GOTO4450
4500 CLOSE2
4510 PRINTSPC(6);FI$;" LOADED"
4520 LL=LL+1
4530 GOTO500
4600 REM DISK SAVE
4610 IFM$="L"THEN4800
4620 IFLL=1THENPRINT"NO FILE TO SAVE
   ":GOTO500
4630 FL$="@0"+DR$+FI$+".SOURCE,S,W"
4640 OPEN2,8,2,FL$
4650 GOSUB20000:REM ERROR CHECK
4660 FORI=1TOLL-1
4670 FORJ=1TOLEN(T$(I))
4680 PRINT#2,MID$(T$(I),J,1);
4690 NEXTJ
4700 PRINT#2,CHR$(255)
4710 NEXTI
4720 CLOSE2
4730 PRINTSPC(6);FI$;" SAVED"
4740 GOTO500
4800 REM DISK LOAD
4810 FL$=DR$+FI$+".SOURCE,S,R"
4820 OPEN2,8,2,FL$
4830 GOSUB20000:REM ERROR CHECK
4835 IFE1<>0THEN500
4840 LL=0:REM LINE COUNT
4850 LL=LL+1:T$(LL)=""
4860 GET#2,A$
4870 IFST=64THEN4500:REM END OF FILE

4880 IFST<<>0THENGOSUB20000:GOTO500
4890 IFA$=CHR$(255)THEN4850:REM END ~
   OF LINE
4900 T$(LL)=T$(LL)+A$
4910 GOTO4860
4920 CLOSE2
4930 PRINTSPC(6);FI$;" LOADED"
4940 LL=LL+1
4950 GOTO500
5000 REM INSERT LINE
5010 PRINT:PRINT"{REV}INSERT{OFF} BE
   FORE ";:GOSUB17000:REM GET
    LINE #
5015 IFLO>LLORLO<1THEN5000
5020 PRINT:PRINTLO;">";
5030 GOSUB10000:REM READ LINE
5040 IFLEN(L$)=0THEN500
5050 LL=LL+1
5060 FORI=LLTOLOSTEP-1
5070 T$(I)=T$(I-1)
5080 NEXTI
5090 T$(LO)=L$
5100 LO=LO+1
5110 GOTO5020
6000 REM LIST LINES
6010 PRINT:PRINT"{REV}LIST{OFF} ";:G
   OSUB16000:REM GET RANGE
6020 IFHI=0THEN500
6030 SS$="N":PRINT:FORI=LOTOHI:REM P
   ERFORM LIST
6040 PRINTI;">";T$(I)
6050 GETA$:IFA$=CHR$(18)THENFORJ=1TO
   1024:NEXTJ
6060 IFA$<>CHR$(3)THEN6110
6070 SS$="Y"
6080 GETA$:IFA$=CHR$(13)THENSS$="N":
   GOTO6110
6090 IFA$<>CHR$(32)THEN6070
6100 GOTO6120
6110 IFSS$="Y"THEN6070
6120 NEXTI
6130 GOTO500
7000 REM REPLACELINE
7010 PRINT:PRINT"{REV}REPLACE{OFF} "
   ;:GOSUB17000:REM GET LINE#

7020 IFLO>=LLORLO<1THEN7000
7030 PRINT:PRINTLO,">";
7040 GOSUB10000:REM RED LINE
7050 IFLEN(L$)=0THEN500
7060 T$(LO)=L$
7070 GOTO500
8000 REM QUIT
8010 PRINT:PRINT"     {REV}LEAVE EDI
   TOR-ARE YOU SURE?{OFF} ";
8020 GETA$:IFA$=""THEN8020
8030 PRINTA$
8040 IFA$<>"Y"ANDA$<>"N"THEN8000
8050 IFA$="N"THEN500
8060 PRINT:PRINT"      {REV}** END L
   INE EDITOR **{OFF}"
8070 POKE144,46:REM ENABLE STOP KEY
8075 POKE59468,12:REM ENABLE UPPERCA
   SE AND GRAPHICS
8080 END
9000 REM PRINT LINE
9010 IFPR=0THENPR=4:OPENPR,PR
9020 PRINT:PRINT"{REV}PRINT{OFF} ";:
   GOSUB16000:REM GET RANGE
9025 INPUT"NUMBER OF SPACES BETWEEN ~
   LINES(1-2)";S1
9030 IF HI=0THEN500
9040 FORI=LOTOHI:REM PERFORM PRINT
9050 PRINT#PR,SPC(SP)"{DOWN}"+T$(I)
9055 IFS1=2THENPRINT#PR:REM DOUBLE S
   PACE
9060 NEXTI
9065 GOTO9100:REM  ***  TO GET RID O
   F JUNK ***
9070 PRINT#PR
9080 PRINT#PR,"***";LL-1;"LINES IN B
```

DYNACOMP

Quality software for:*

ATARI — TRS-80 (Level II)**

PET — NORTH STAR

APPLE II Plus — CP/M Disks/Diskettes

(see Availability box)

DYNACOMP OFFERS THE FOLLOWING

- Widest variety
- Guaranteed quality
- Fastest delivery
- Friendly customer service
- Free catalog
- 24 hour order phone

CARD GAMES

BACCARAT (Atari only) — **Price: $18.95 Cassette/$22.95 Diskette**
This is the European card game which is the favorite of the Monte Carlo jet set. Imagine yourself at the gaming table with 007 to your left and Goldfinger to your right. Learn and play BACCARAT at your leisure on the Atari. Contains full high resolution color graphics and matching sound. Runs in 16K. Requires one joystick.

GIN RUMMY (Apple only) — **Price: $18.95 Cassette/$22.95 Diskette**
This is the best micro computer implementation of GIN RUMMY existing. The computer plays exceptionally well, and the HIRES graphics are superb. What else can be said?

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

CRIBBAGE 2.0 (TRS-80 only) — **Price: $14.95 Cassette/$18.95 Diskette**
This is simply the best cribbage game available. It is an excellent program for the cribbage player in search of a worthy opponent as well as for the novice wishing to improve his game. The graphics are superb and assembly language routines provide rapid execution. See the software review in 80 Software Critique.

THOUGHT PROVOKERS

MANAGEMENT SIMULATOR (Atari, North Star and CP/M only) — **Price: $19.95 Cassette/$23.95 Diskette**
This program is both an excellent teaching tool as well as a stimulating intellectual game. Based upon similar games played at graduate business schools, each player or team controls a company which manufacturers three products. Each player attempts to outperform his competitors by setting selling prices, production volumes, marketing and design expenditures etc. The most successful firm is the one with the highest stock price when the simulation ends.

FLIGHT SIMULATOR (Available for all computers) — **Price: $17.95 Cassette/$21.95 Diskette**
A realistic and extensive mathematical simulation of take-off, flight and landing. The program utilizes aerodynamic equations and the characteristics of a real airfoil. You can practice instrument approaches and navigation using radials and compass headings. The more advanced flyer can also perform loops, half-rolls and similar aerobatic maneuvers. Although this program does not employ graphics, it is exciting and very addictive. See the software review in COMPUTRONICS. Runs in 16K Atari.

VALDEZ (Available for all computers) — **Price: $15.95 Cassette/$19.95 Diskette**
VALDEZ is a computer simulation of supertanker navigation in the Prince William Sound/Valdez Narrows region of Alaska. Included in this simulation is a realistic and extensive 256 × 256 element map, portions of which may be viewed using the ship's alphanumeric radar display. The motion of the ship itself is accurately modelled mathematically. The simulation also contains a model for the tidal patterns in the region, as well as other traffic (outgoing tankers and drifting icebergs). Chart your course from the Gulf of Alaska to Valdez Harbor! See the software reviews in 80 Software Critique and Personal Computing.

BACKGAMMON 2.0 (Atari, North Star and CP/M only) — **Price: $14.95 Cassette/$18.95 Diskette**
This program tests your backgammon skills and will also improve your game. A human can compete against a computer or against another human. The computer can even play against itself. Either the human or the computer can double or generate dice rolls. Board positions can be created or saved for replay. BACKGAMMON 2.0 plays in accordance with the official rules of backgammon and is sure to provide many fascinating sessions of backgammon play.

CHECKERS 3.0 (PET only) — **Price: $16.95 Cassette/$20.95 Diskette**
This is one of the most challenging checkers programs available. It has 10 levels of play and allows the user to change skill levels at any time. Although providing a very tough game at level 4-8, CHECKERS 3.0 is practically unbeatable at levels 9 and 10.

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

LEM LANDER (32K Apple Disk only) — **Price: $16.95 Diskette**
Pilot your LEM LANDER to a safe landing on any of nine different surfaces ranging from smooth to treacherous. The game paddles are used to control craft attitude and thrust. This is a real-time high res challenge!

FOREST FIRE! (Atari only) — **Price: $16.95 Cassette/$20.95 Diskette**
Using excellent graphics and sound effects, this simulation puts you in the middle of a forest fire. Your job is to direct operations to put out the fire while compensating for changes in wind, weather and terrain. Not protecting valuable structures can result in startling penalties. Life-like variables are provided to make FOREST FIRE! very suspenseful and challenging. No two games have the same setting and there are 3 levels of difficulty.

SPACE EVACUATION! (Apple, Atari and TRS-80 only) — **Price: $15.95 Cassette/$19.95 Diskette**
Can you colonize the galaxy and evacuate the Earth before the sun explodes? Your computer becomes the ship's computer as you explore the universe to relocate millions of people. This simulation is particularly interesting as it combines many of the exciting elements of classic space games with the mystery challenge of ADVENTURE.

MONARCH (Atari only) — **Price: $11.95 Cassette/$15.95 Diskette**
MONARCH is a fascinating economic simulation requiring you to survive an 8-year term as your nation's leader. You determine the amount of acreage devoted to industrial and agricultural use, how much food to distribute to the populace and how much should be spent on pollution control. You will find that all decisions involve a compromise and that it is not easy to make everyone happy. Runs in 16K Atari.

CHOMPELO (Atari only) — **Price: $11.95 Cassette/$15.95 Diskette**
CHOMPELO is really two challenging games in one. One is similar to NIM; you must bite off part of a cookie, but avoid taking the poisoned portion. The other game is the popular board game REVERSI. It fully uses the Atari's graphics capability, and is hard to beat. This package will run on a 16K system.

SPACE LANES (Available for all computers) — **Price: $10.95 Cassette/$14.95 Diskette**
SPACE LANES is a simple but exciting space transportation game which involves up to four players (including the computer). The object is to form and expand space transportation companies in a competitive environment. The goal is to amass more net worth than your opponent. The economics include stock purchases and company mergers. Watch your wealth grow!

AVAILABILITY

DYNACOMP software is supplied with complete documentation containing clear explanations and examples. Unless otherwise specified, all programs will run within 16K program memory space (ATARI requires 24K). Except where noted, programs are available on ATARI, PET, TRS-80 (Level II) and Apple (Applesoft) cassette and diskette as well as North Star single density (double density compatible) diskette. Additionally, most programs can be obtained on standard (IBM 3740 single density/double density compatible format) 8" CP/M floppy disks for systems running under MBASIC (for example, Altos, Xerox 820 and many others). 5¼" CP/M diskettes are available for the North Star and Osborne computer systems.

**ATARI, PET/CBM, NORTH STAR, CP/M, IBM, OSBORNE and XEROX are registered tradenames and/or trade marks.*

***Except where noted, all TRS-80 Model I software is available on cassette (only) for the TRS-80 Model III. Exceptions: VALDEZ, CRIBBAGE, GRAFIX, CHESSMASTER. TRS-80 diskettes are not supplied with either DOS or BASIC.*

AND MORE...

STARTREK 3.2 (Available for all computers) — **Price: $11.95 Cassette/$15.95 Diskette**
This is the classic Startrek simulation, but with several new features. For example, the Klingons now shoot at the Enterprise without warning while also attacking starbases in other quadrants. The Klingons also attack with both light and heavy cruisers and move when shot at! The situation is hectic when the Enterprise is besieged by three heavy cruisers and a starbase S.O.S. is received! The Klingons get even! See the software reviews in A.N.A.L.O.G., 80 Software Critique and Game Merchandising.

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

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

ESCAPE FROM VOLANTIUM (Atari only) — **Price: $15.95 Cassette/$19.95 Diskette**
Bring the action and excitement of an arcade into your home with ESCAPE FROM VOLANTIUM! To escape you must maneuver your space ship around obstacles and laser blast the dragon (without being eaten). If he is killed with a direct shot (not just a leg lopped off), a door opens to the outside. However, the door does not stay open indefinitely. If you fail to escape in time, the door closes and a new dragon appears. Sometimes you can smash through the door by repeatedly chipping away at it. Other times it is impervious. At the higher levels of play more obstacles and dragons appear, adding to the excitement. Uses high resolution graphics and sound. Runs in 16K.

ALPHA FIGHTER (Atari only) — **Price: $14.95 Cassette/$18.95 Diskette**
Two excellent graphics and action programs in one! ALPHA FIGHTER requires you to destroy the alien starships passing through your sector of the galaxy. ALPHA BASE is in the path of an alien UFO invasion; let five UFO's get by and the game ends. Both games require the joystick and get progressively more difficult the higher you score! ALPHA FIGHTER will run on 16K systems.

THE RINGS OF THE EMPIRE (Atari only) — **Price: $16.95 Cassette/$20.95 Diskette**
The empire has developed a new battle station protected by rotating rings of energy. Each time you blast through the rings and destroy the station, the empire develops a new station with more protective rings. This exciting game runs on 16K systems, employs extensive graphics and sound and can be played by one or two players.

INTRUDER ALERT (Atari only) — **Price: $16.95 Cassette/$20.95 Diskette**
This is a fast paced graphics game which places you in the middle of the "Dreadstar" having just stolen its plans. The droids have been alerted and are directed to destroy you at all costs. You must find and enter your ship to escape with the plans. Five levels of difficulty are provided. INTRUDER ALERT requires a joystick and will run on 16K systems.

MIDWAY (Atari only) — **Price: $14.95 Cassette/$18.95 Diskette**
MIDWAY is an exciting extension of the game of Battleship. It mixes the challenges of strategy and chance. Your opponent can be another human or the computer. Color graphics and sound are both included. Runs in 16K.

TRIPLE BLOCKADE (Atari only) — **Price: $14.95 Cassette/$18.95 Diskette**
TRIPLE BLOCKADE is a two-to-three player graphics and sound action game. It is based on the classic video arcade game which millions have enjoyed. Using the Atari joysticks, the object is to direct your blockading line around the screen without running into your opponent(s). Although the concept is simple, the combined graphics and sound effect lead to "high anxiety".

GAMES PACK I (Available for all computers) — **Price: $10.95 Cassette/$14.95 Diskette**
GAMES PACK I contains the classic computer games of BLACKJACK, LUNAR LANDER, CRAPS, HORSERACE, SWITCH and more. These games have been combined into one large program for ease in loading. They are individually accessed by a convenient menu. This collection is worth the price just for the DYNACOMP version of BLACKJACK.

GAMES PACK II (Available for all computers) — **Price: $10.95 Cassette/$14.95 Diskette**
GAMES PACK II includes the games CRAZY EIGHTS, JOTTO, ACEY-DUCEY, LIFE, WUMPUS and others. As with GAMES PACK I, all the games are loaded as one program and are called from a menu. You will particularly enjoy DYNACOMP's version of CRAZY EIGHTS.

Why pay $7.95 or more per program when you can buy a DYNACOMP collection for just $10.95?

MOON PROBE (Atari and North Star only) — **Price: $11.95 Cassette/$15.95 Diskette**
This is an extremely challenging "lunar lander" program. The user must drop from orbit to land at a predetermined target on the moon's surface. You control the thrust and orientation of your craft plus direct the rate of descent and approach angle. Runs in 16K Atari.

SPACE TRAP (Atari only, 16K) — **Price: $14.95 Cassette/$18.95 Diskette**
This galactic "shoot'em up" arcade game places you near a black hole. You control your spacecraft using the joystick and attempt to blast as many of the alien ships as possible before the black hole closes about you.

CHIRP INVADERS (PET/CBM only) — **Price: $14.95 Cassette/$18.95 Diskette**
CHIRP INVADERS is an addictive game using action graphics. A Federation space station must be reached before the Chirps conquer the Earth. Stationary obstacles, moving meteors, and the attacking Chirps must all be avoided for a successful journey. Good luck.

ADVENTURE

CRANSTON MANOR ADVENTURE (North Star and CP/M only) — **Price: $29.95 Diskette**
At last! A comprehensive Adventure game for North Star and CP/M systems. CRANSTON MANOR ADVENTURE takes you into mysterious CRANSTON MANOR where you attempt to gather fabulous treasures. Lurking in the manor are wild animals and robots who will not give up the treasures without a fight. The number of rooms is greater and the associated descriptions are much more elaborate than the current popular series of Adventure programs, making this game the top in its class. Play can be stopped at any time and the status stored on diskette. Not available in 5¼" CP/M format.

GUMBALL RALLY ADVENTURE (North Star only, 48K) — **Price: $21.95 Diskette**
Take part in this outlaw race from the east coast to the west coast. The goal is to find your way to the finish line while maintaining the highest possible speed. You may choose one of five cars available at the garage. The choice will affect your speed and range. Remember to take spare parts and don't get caught speeding!

UNCLE HARRY'S WILL (North Star only, 40K) — **Price: $24.95 Diskette**
Uncle Harry has died and has left you everything. However, he has neglected to mention where everything is! Instead, his will consists of a poem which contains clues. You will have to travel all over the United States both by car and on foot to solve the puzzle, and there are over 300 locations to probe. Be careful and watch out for red herrings!

SPEECH SYNTHESIS

DYNACOMP is now distributing the new and revolutionary TYPE-'N-TALK™ (TNT) speech synthesizer from Votrax. Simply connect TNT to your computer's serial interface, enter text from the keyboard and hear the words spoken. TNT is the easiest-to-program speech synthesizer on the market. It uses the least amount of memory and provides the most flexible vocabulary available anywhere!

List price $375. DYNACOMP'S price $329.95. Please add $5.00 for shipping and handling.

TALK TO ME (T'N'T Atari only, 24K) — **Price: $14.95 Cassette/$18.95 Diskette**
This program presents a superb tutorial on speech synthesis using the Atari 800 and TYPE 'N TALK™. TALK TO ME will illustrate normal word generation as well as phoneme generation. The documentation includes many helpful programming tips.

MISCELLANEOUS

CRYSTALS (Atari only) — **Price: $ 9.95 Cassette/$13.95 Diskette**
A unique algorithm randomly produces fascinating graphics displays accompanied with tones which vary as the patterns are built. No two patterns are the same, and the combined effect of the sound and graphics are mesmerizing. CRYSTALS has been used in local stores to demonstrate the sound and color features of the Atari. Runs in 16K Atari.

NORTH STAR SOFTWARE EXCHANGE (NSSE) LIBRARY
DYNACOMP now distributes the 23 volume NSSE library. These diskettes each contain many programs and offer an outstanding value for the purchase price. They should be part of every North Star user's collection. Call or write DYNACOMP for details regarding the contents of the NSSE collection.

Price: $9.95 each/$7.95 each (4 or more)
The complete collection may be purchased for $149.95

```
   UFFER ***"
9090 PRINT#PR
9100 GOTO500
10000 REM INPUT A LINE OF TEXT
10010 L$=""
10020 PRINT"$_{LEFT}";
10030 GETA$:IFA$=""THEN10030
10040 IFA$=CHR$(13)THENPRINT" ":RETUR
   N
10045 IFA$="@"THENA$="     ":REM TAB
10050 IFLEN(L$)=(M-5)THENGOSUB22000
10055 IFLEN(L$)>MTHENGOTO15000:REM SP
   ECIFIES MAX LENGTH OF STRI
   NG
10060 IFA$>=SP$ANDA$<=CHR$(95)THEN101
   ØØ
10065 IFA$>=CHR$(161)ANDA$<=CHR$(223)
   THEN10100
10070 IFA$<>DL$THENGOTO10030
10080 IFLEN(L$) >ØTHENPRINTA$;:L$=LEF
   T$(L$,LEN(L$)-1)
10090 GOTO10020
10100 L$=L$+A$:PRINTA$;:GOTO10020
15000 REM LINE INPUT ERROR
15010 PRINT:PRINT"{REV}ERROR{OFF} LIN
   E TRUNCATED"
15020 RETURN
16000 PRINT"RANGE(LOW,HIGH)-> ";
16010 GOSUB10000:REM INPUT RANGE
16020 LO=1:HI=LL-1:REM DEFAULT LIST A
   LL
16025 L=LEN(L$)
16030 DF=Ø:IFL=ØTHENDF=-1:GOTO16150
16040 J=Ø:FORI=1TOL
16050 A$=MID$(L$,I,1)
16060 IFA$>="Ø"ANDA$<="9"THEN16090
16070 IFA$="-"THENJ=I:GOTO16090
16080 J=99:I=99
16090 NEXTI
16100 IFJ=99THEN16000
16110 IFJ=ØTHENLO=VAL(L$):HI=LO:RETUR
   N
16120 IFJ>1THENLO=VAL(LEFT$(L$,J-1))
16130 IFJ<LTHENHI=VAL(RIGHT$(L$,L-J))
16140 IFLO>HITHEN16000
16150 RETURN
17000 PRINT"-LINE#->";
17010 GOSUB10000:REM INPUT LINE#
17020 L=LEN(L$)
17030 IFL=ØTHEN17000
17040 J=Ø
17050 FORI=1TOL
17060 A$=MID$(L$,I,1)
17070 IFA$>="Ø"ANDA$<="9"THEN17090
17080 J=99:I=L
17090 NEXTI
17100 IFJ=99THEN17000
17110 LO=VAL(L$)
17120 RETURN
20000 IFEE=ØTHENEE=15:OPENEE,8,EE
20010 INPUT#EE,E1,E2$,E3,E4
20020 IFE1=ØTHENRETURN
20030 PRINTE1;",";E2$;",";E;",";E4
20040 PRINT"*** DISK ERROR ***"
20050 CLOSE2
20060 RETURN
21000 REM SET MARGINS
21010 PRINT:PRINT"{REV}SET MARGIN{OFF
   OFF}"
21020 PRINTTAB(7)"MARGIN SIZE:{DOWN}N
   )ONE
21025 PRINTTAB(19)"S)MALL  (1/2INCH) ~
                          M)ED
   IUM (1.ØINCH)
21027 PRINTTAB(19)"L)LARGE (1.5INCH)
21030 PRINTTAB(19)"O)WN DESIGN"
21035 GETMS$:IFMS$=""THEN21035
21050 IFMS$="N"THENM=79:SP=Ø:GOTO500
21060 IFMS$="S"THENM=74:SP=5:GOTO500
21070 IFMS$="M"THENM=69:SP=10:GOTO500
21075 IFMS$="L"THENM=64:SP=15:GOTO500

21080 IFMS$="O"THENPRINT:PRINT
21085 INPUT"  INCHES FOR LEFT MARGIN ~
               (10 CHARAC
   TERS/INCH)=";SP
21090 INPUT"  INCHES FOR RIGHT MARGIN
               (10 CHARAC
   TERS/INCH)=";RM
21100 SP=INT(LM*100)/10:M=INT(79-RM*1
   Ø):GOTO500
22000 REM BELL
22020 FORII=1TO 5:POKE59467,16:POKE59
   466,85:POKE59464,115:NEXTI
   I
22030 POKE59467,Ø:POKE59466,255:POKE5
   9464,254
22040 RETURN
```

©

Transposition

Janet Whitehead
Saint John, N.B., Canada

Editor's Note: This transposition algorithm works on both Atari and PET/CBM. We hope to see some exciting harmony or other musical applications as suggested by Janet at the end of her article. — RTM

On first observation, the sound command SOUND V,N,T,L seemed to have a lack of pattern for the sequence of numbers representing the note *N*. I recalled a question in a high school mathematics book that stated that the frequency of A above middle C was 440 cycles per second. To obtain the next higher note on a musical scale multiply by $2^{1/12}$, for a lower note divide by $2^{1/12}$; thus one can find the frequency of each of the twelve notes in an octave. (I am considering each octave as containing twelve notes, the five black notes as well as the seven white ones on a piano).

This, I thought, must be the basis for the sequence of numbers used for notes in Atari BASIC. As the value of *N* is from 0 to 255 (one byte), the frequency was not used, but some multiple of it. As frequency increases the pitch increases, but the value of *N* decreases as the pitch increases. Therefore, to increase the pitch you divide *N* by $2^{1/12}$ instead of multiplying by $2^{1/12}$.

This property can be used to transpose music. To raise a composition by one-half tone, one only needs to divide the *N* value by $2^{1/12}$, for a full tone divide by $(2^{1/12})^2$, for a tone and one-half by $(2^{1/12})^3$ etc.

To illustrate these properties, here are two simple programs. Program 1 prints the sequence of number used for *N* in the sound command. To obtain the sequence in *Atari BASIC* by Albrecht et al. an original *N* value of 259 was used instead of 255. If you find that these give values for *N* which produce sharp or flat tones, just change the 259.

Program 1:

Line 40: T1 finds successive values of $(2^{1/12})^0$, $(2^{1/12})^1$, $(2^{1/12})^2$ etc.

Line 50: Successive one-half tones, *N* values, are calculated.

Line 60: The results are printed.

Notice that, for notes one octave apart, the ratio of the two *N* values is 2:1.

Program 2:

This plays a few bars of music to illustrate how a piece of music can be transposed through one octave.

Line 100 – Sets the voice to 0, the tone as 10, and the loudness at 10.

Line 110-120 – M is the value of $2^{1/12}$ and T1 is the number of half-tones to transpose the music.

Line 140 – M1 calculates the value of $(2^{1/12})^{T1}$ which is the factor by which each *N* value must be divided to raise a piece by T1 half tones.

Line 150 – *A* holds the original value of *N*, and *B* indicates the length of time it is to be played. *A* is then transposed the desired number of half tones. The note is then played.

Line 170 – As the program plays the few bars of music through each successive half tone for one octave, this line increases the amount *N* is to be transposed one half a tone. As the data must be read each time, it needs to be RESTOREd.

Caution: If you exceed an *N* value of 255 in your transposition, you will get a very high pitched note. Only one byte is used for *N*, so 257 would be 1.

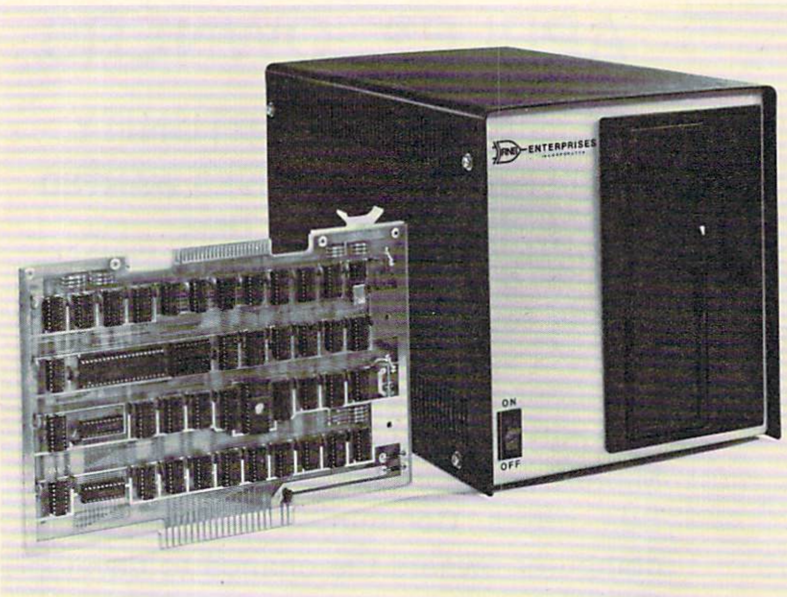
Perhaps some reader can expand on this to play chords or generate harmony. Knowing very little about music, I will have to leave that task to someone else.

Program 1.

```
10 M=2^(1/12)
20 T=0
30 FOR I=1 TO 40
40 T1=M^T
50 N=259/T1
60 PRINT I,INT(N+0.5)
70 T=T+1
80 NEXT I
90 END
```

VAK-7 8″ FLOPPY DISK SYSTEM

FOR AIM-65, SYM-1



The VAK-7 Disk System was specifically designed for use with AIM-65 and SYM-1 Microcomputer Systems. The VAK-7 is a complete full size (8″) Floppy Disk System. This system will Read, Write and Format IBM Single and Dual Density diskettes. The VAK-7 is available with both Single and Dual Sided Disk Drives. Each Disk Drive comes with its own cabinet and Power Supply. The VAK-7 can handle up to 4 disk drives, totaling more than 4.98 Megabytes of storage.

The VAK-7 Disk System incorporates both advanced hardware and innovative software designs. The addition of the VAK-7 produces a very powerful and useful computer system. Unlike most other disk systems, there is no requirement for the user to provide RAM to hold the Disk Operating System software. No valuable time is wasted loading in the DOS from cassette tape, because the VAK-7 DOS software is in onboard ROM. The VAK-7 is located above the 32K boundary (8000 HEX), leaving the user with a full 32K bytes of contiguous address space for his own use.

AIM-65—Allows the user to save and load object code thru the AIM Monitor; to load, save, and append Text thru the AIM Editor; to load, save, and append Basic Programs thru the BASIC INTERPRETER; to assemble directly from disk single or multiple file programs.

SYM-1—Allows the user to save and load Files for use with the SYM Monitor, SYM Basic, and RAE-1.

ADDITIONAL COMMANDS:

- ACTIVATE A DELETED FILE
- COMPRESS A DISK
- RENAME A DISK FILE
- COPY A DISK
- FORMAT A DISK
- DELETE A DISK FILE
- INITIALIZE A DISK
- LIST CATALOG

SPECIFICATIONS:

- Completely assembled, tested, and burned in.
- Occupies address 8000–8FFF for AIM-65, $9000–9FFF for SYM-1, or $E000–EFFF for KIM-1.
- IBM Format; Single Density (128 bytes/sector); Dual Density (256,512, or 1024 bytes/sector).
- All ICs are in sockets.
- Fully buffered address and data bus.
- Standard KIM-4* BUS (both electrical pin-out and card size).
- Designed for use with a regulated power supply, but has provisions for adding regulators for use with an unregulated power supply.
- Dimensions: Board—10″ wide x 7″ high (including card-edge). Cabinet—9.25″ wide x 10″ high x 16″ deep.
- Power Requirements: + 5v DC @ 2 Amps.
  117 AC 60Hz @ 2 Amps.
- Shipping Weight: 25 lbs.

KIM-4 is a product of MOS Technology/C.B.M.

PRICING:

VAK-7 **$1,299.00**
Controller and One Single-Sided Drive

VAK-7A **$599.00**
Additional Single-Sided Drive with Cabinet and Power Supply

VAK-7B **$1,599.00**
Controller and One Dual-Sided Drive

VAK-7C **$899.00**
Additional Dual-Sided Drive with Cabinet and Power Supply

CALL OR WRITE FOR FREIGHT CHARGES

RNB ENTERPRISES INCORPORATED

4030 N. 27th Avenue, Suite D
Phoenix, AZ 85017
(602) 265-7564

master charge VISA

Program 2: Atari Version

```
100 V=0:T=10:L=10
110 M=2^(1/12)
120 T1=0
130 ? :? "HAPPY BIRTHDAY"
140 M1=M^T1
150 FOR X=1 TO 26:READ A,B:A=INT(A/M1+0
5):SOUND V,A,T,L
160 FOR I=1 TO B:NEXT I:SOUND V,0,0,0:NE
XT X
170 T1=T1+1
180 IF T1<12 THEN RESTORE 190:GOTO 140
185 END
190 DATA 122,64,122,64,109,128,122,128,9
2,128,97,256
200 DATA 122,64,122,64,109,128,122,128,8
2,128,92,256
210 DATA 122,64,122,64,61,128,73,128
220 DATA 92,64,92,64,97,128,109,128
230 DATA 69,64,69,64,73,128,92,128,82,12
8,92,256
```

Program 3: CBM Version

```
100 POKE59467,16:POKE59466,15:POKE5
    9464,0:S=59464
110 M=2^(1/12)
120 T1=0
130 PRINT"HAPPY BIRTHDAY
140 M1=M^T1
150 FORX=1TO26:READA,B:A=INT(A/M1+.
    5):POKES,A
160 FORI=1TOB+B/2:NEXTI:POKES,0:NEX
    TX
165 REM VALUE OF B IS INCREASED HER
    E TO EQUALIZE THE DIFFEREN
    CES IN SPEED
168 REM BETWEEN THE CBM/PET AND THE
     ATARI
170 T1=T1+1
180 IFT1<12THENRESTORE:GOTO140
185 POKE59467,0:END
190 DATA122,64,122,64,109,128,122,1
    28,92,128,97,256
200 DATA122,64,122,64,109,128,122,1
    28,82,128,92,256
210 DATA122,64,122,64,61,128,73,128

220 DATA92,64,92,64,97,128,109,128
230 DATA69,64,69,64,73,128,92,128,8
    2,128,92,256
```

©

TELECOMMUNICATIONS

Why 300 Baud?

Michael E. Day
Chief Engineer, Edge Technology

Time plays a very important role in telecommunications. Every industry builds up its own group of words and terms to define those things it deals with that cannot be readily described with everyday language. The telecommunications field has more than its share of buzz words. Close analysis shows that many of these words deal with time and time-related things.

Time is important in all aspects of telecommunication, from the transmission of a single bit of information, to the overall system performance. Although there are no exact limits to the use of time as it relates to telecommunication, there are practical limits and deterioration effects which must be considered when designing a system.

There are many factors which must be taken into account when designing a telecommunications system. The purpose of the system generally determines what methods will be used. Of major importance is the amount of information flow that will occur. This will generally determine the minimum acceptable system communication speed.

Short Messages Permit Slow Speeds

If the message to be sent is relatively short, and little or no response is expected, then very slow system speeds can be acceptable. Western Union makes use of this on their TELEX network which operates at 6.7 characters per second. A slow system speed has the advantage of using only a small amount of the communications bandwidth. This means that more systems can be installed on the same communications link and can thus reduce the per-user cost of the communications system. The TELEX system serves users who must be sure that their messages get to the parties to which they were sent in a short period of time, but at low cost. (In some ways it can be cheaper to send a TELEX than it is to send a letter).

Another low cost system structure is the message forwarding system. Here higher communication speeds are allowed, but direct communication with the target party is not allowed. Instead the message is built up at a location close to the sending party and then transmitted to the target party at a time when the cost is at the lowest rate. In the case of *packet switchers*, it is sent when a communications link becomes available.

When a large amount of data is to be sent, the speed of the communications link becomes important. Transmission at higher speeds requires a larger communications bandwidth. Because of this, there is an associated increase in the cost of the communications link as well as the equipment required.

Between these extremes lie general information communications systems involving a low to medium amount of data transfer. Often this occurs in conjunction with user interaction with the data flow. This is the area where most computer use occurs.

Although the exact system configurations change as technology changes, the general structure of use remains relatively constant. Any change in one area of the telecommunications field tends to affect the other areas in some way.

The Three Common Computer Modems

There are three types of modems that the average computerist encounters: 1. The 103 style modem (the most commonly used). 2. The 202 style modem (far less common, but it has the advantage of higher speed at a reasonable cost, but with an increase in complexity of use). 3. Finally, the newer 212A style modem combines the higher speed of the 202 with the ease of use of the 103 (but costs more).

The 202 modem is sort of an ugly duckling in the telecommunications field. It has the advantage of higher speed than the 103, yet it can be built at a lower cost. The problem comes from its complexity of use.

The 202 is a half duplex modem. This means that it can only transmit in one direction at a time and, thus, requires some amount of computer control over its use. If the intended use requires high speed at lost cost, this is normally considered an acceptable trade off. If the computer control requirement is not acceptable, the higher cost 212A modem is used to achieve high speed without computer control. If cost is important, the 103 is used.

Although the 103 is more expensive to make on a price/performance basis, there is a demand for the low cost 103's. They are mass produced and mass distributed at low profit margins, and are often of lower quality.

Acoustic Versus Direct Connect

The modems come in two major classifications, the acoustic type and the direct connect type. Until recently, the acoustic was the most popular with the general user. This was due largely to regulations which inhibited the direct connection of modems to the phone lines. There are still many regulations which inhibit this, but they have been reduced to the point that the general user at least can consider

it as an option. It is actually possible to build a direct connect modem less expensively than the acoustic version, but, until the regulations are further reduced and demand increases, this will probably not happen.

Acoustic versions of the 202 modem are essentially nonexistent. The main reason for this is the telephone. The telephone uses a carbon microphone to convert sound into electricity. The carbon microphone is very rugged, cheap to build, has a high gain factor, and has natural filtering characteristics in the frequency range of the human voice. Unfortunately, it is also very noisy and disruptive to the signal that it converts.

The 202 cannot work within this environment. The 212A modem fares a little better because the part of the signal it uses for data transfer suffers a minimal amount of disruption from the carbon microphone. The 103, however, works reasonably well with the carbon microphone. This is due to the narrow bandwidth used for transmission, as well as the high redundancy in the signal used. Although some 103 modems are capable of operation at speeds of up to 600 baud, the disruption of the carbon microphone tends to limit the speed to 300 baud. If the phone system is in good workig order, it is possible to operate as high as 450 baud. Alternatively, if the system is in poor condition, it may require that operation be reduced to 200 baud, 150 baud, or even as low as 110 baud. Generally speeds below 110 baud do not actually achieve any improvement in reliability of operation on the 103.

If problems are encountered, or higher speeds are desired, the direct connect modems are generally required. If only a slight improvement is needed, some degree of improvement can sometimes be obtained by replacing the carbon microphone with a capacitor (or condenser) microphone (sometimes referred to as a *supermike*). The capacitor microphone is more expensive to make than the carbon, but does not disrupt the signal. There is one disadvantage to most capacitor microphones. It requires power to operate, and it gets this power from the phone line. If another phone comes on the line, it can steal the power away from the capacitor microphone and inhibit it from working. This is particularly true if the other phone has a carbon microphone in it which tends to drop the power well below the capacitor microphone's normal operating level.

The 103 type modem has become a standard for medium speed communications over the telephone network not because someone felt it should be, or because someone made it so. It became the standard because it was the optimal solution to the problem at hand. ©

Reading The Status Register

Bob Sullivan
Oak Park, IL

Here is a way to quickly analyze the flags in the 6502 Status Register. When you are working in machine language, the branch (BNE, BCC, etc.) instructions automatically check these flags for you and make their "decisions" based on the condition (or status) of a flag. However, there are times when *you* need to analyze the Status Register. When debugging, for example, you might place a BReaK instruction to stop the program and allow you to examine the condition of a flag.

Each flag is a bit within the Status Register byte. When you see that the SR has $F1 in it, how quickly can you determine that the Overflow flag is set? The table below does the job:

CARRY BIT-right digit
Set if odd: 1,3,5,7,9,B,D,F
Clear if even: 2,4,6,8,A,C,E

ZERO BIT-right digit
Set if: 2,3,6,7,A,B,E,F
Clear if: 0,1,4,5,8,9,C,D

INTERRUPT BIT-right digit
Set if: 4,5,6,7,C,D,E,F
Clear if: 0,1,2,3,8,9,A,B

DECIMAL MODE-right digit
Set if: 8,9,A,B,C,D,E,F
Clear if: 0,1,2,3,4,5,6,7

BREAK-left digit
Set if odd: 1,3,5,7,9,B,D,F
Clear if even: 0,2,4,6,8,A,C,E

OVERFLOW
Set if: $C0 to $FF
Clear if: $00 to BF

SIGN BIT
Positive: $00 to $7F
Negative: $80 to $FF

©

Book Review:

Some Common Basic Programs

Jim Butterfield
Toronto, Canada

This book, a collection of 76 short programs, is by Lon Poole and Mary Borchers. There are editions for several popular machines, with additional authorships: Carroll Donahue contributed to the PET/CBM edition, Karl Koessel to the TRS-80 edition, and David M. Castlewitz to the Apple II edition. There's an Atari edition, too. Cassette and diskette versions are available.

The book is well established. It may be a definitive answer to the question: What serious things can I do on my small computer without using data files?

Three Major Areas

The programs break into three major sections, with a few programs left over at the end. It might have been useful to have broken the material up into chapters. The first twenty programs are financial; the next twenty-seven, mathematical; and the third section contains twenty statistical programs. Nine miscellaneous programs are tacked on the end.

The financial programs are quite well commented, and often include optional coding for performing monthly (as opposed to yearly) calculations. Serious users will probably want to combine several short programs together; the book does not show how to do this, but it is not a difficult task. Users should not use these programs as the final word; financial methods differ in different organizations and the serious user will hopefully know what modifications will be needed for his circumstances.

The mathematical programs are somewhat more cryptic; it is expected that the user is quite familiar with the material. For example, three Integration programs are given; the user is expected to choose Simpson's Rule, Trapezoidal Rule, or Gaussian Quadrature according to his estimate of which will suit his needs. Fair enough; the book does not attempt to be a text, but just gives the relevant coding.

The statistical section is a mixed bag in terms of user levels. Some programs such as Average Growth Rate are easy for the naive user to understand; others such as F-Distribution require a comprehension of statistics.

Up Front

None of the programs is huge: all are easy to type in. Worked-through examples allow the user to check that his coding is, indeed, correct.

For the non-mathematical computer owner, the programs may seem to be rather obscure. In this case, the book may serve as a challenge and an indication of the resources he can tap if he wants. Such things as linear programming and regressions can be remarkably powerful tools to use in business ...if you know that they are there. It might be useful to see a companion guide to this book, explaining just how effective some of these mathematical techniques can be, even to the small user. The book doesn't try to do this: it just plunks down the coding.

There may be debate on whether the best mathematical and programming techniques are used in all cases; to me, this doesn't matter. The first thing to do is to find any way of approaching a problem. After you have one way, you can look for better ways; but finding that first one can be hard.

Son-Of-Some-Common...?

There are a couple of books that look like sequels to the well-established SCBP. *Practical Basic Programs,* edited by Lon Poole, gives more programs in a similar vein. *Science and Engineering Programs,* Apple II Edition edited by John Heilborn, delves more deeply into mathematics and statistics. The books are similar in organization to *Some Common Basic Programs,* but the programs are longer and there are fewer of them. Readers who found the first book useful will undoubtedly want to go after its successors.

Some Common Basic Programs is not a textbook. It doesn't teach you what to do with the programs. But it does give you working programs with documentation.

For those who know the methodologies, it will be a useful reference. For those who don't, it may open up new horizons: things that you didn't know a computer can do. In that case, you'll need to look elsewhere to learn the principles of the new technology. It's often an education to discover the existence of things you don't know. Or at least the start of an education.

Some Common Basic Programs. Lon Poole and Mary Borchers. Osborne/McGraw Hill. Berkeley. (1981)

©

NEW PRINTERS. NEW PERIPHERALS. SAME OLD RELIABLE QUALITY AND VALUE.

1982 will find more OEM's, businesses, dealers and personal computer users turning to MICROTEK than ever before.

TekWriter-1

80 Column Dot Matrix Printer
(Formerly BYTEWRITER-1)

The Tekwriter-1 printer is, dollar for dollar, the finest value in the industry. And we've proved it by comparing the Tekwriter-1 to the Epson MX-80. Our print speed is 14 lines per minute faster, our life expectancy is longer, the character sets are the same, and the interface, warranty and printhead replacement cost are all identical.* But the biggest difference is the price. The Tekwriter-1 is about $300 less.

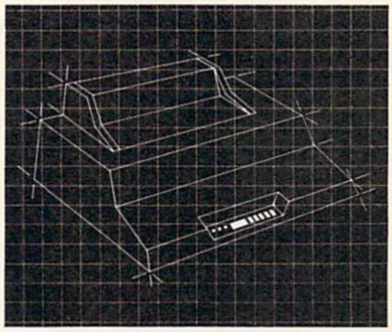
Our extensive testing has proved that the Tekwriter-1 interfaces problem-free to most parallel Centronics and serial (RS-232) computers.

The Tekwriter-1 is tough to beat for performance and quality.

*Data Source: Epson MX-80 Operation Manual

Parallel $349
Serial $389

TekWriter-2



NEW! 80/132 Column Dot Matrix Printer

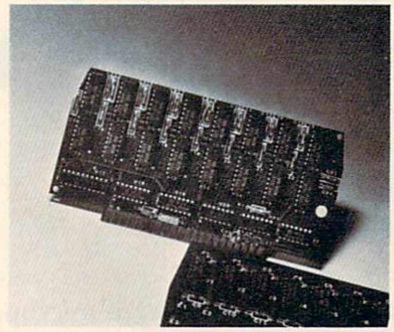
The Tekwriter-2 is perfectly suited to personal, business or OEM applications. Tekwriter-2 is designed to accept single sheet, roll or pin feed paper. It has a 9-wire dot matrix impact print head which produces crisp characters and has underlining capability. The printer is manufactured to run extremely quietly even while operating at peak output levels.

Tekwriter-2 is especially well suited to handle an abundance of text entry because of its data buffer expansion capability to 25K. This ability makes it an efficient graphics generator.

Parallel interface (Centronics type). Interfaces all models of TRS-80, Apple, and Atari 400/800, and most computers with Centronics printer interface.

$695

Peripherals



16K Apple Memory Board

Expands Apple II to 64K RAM Memory. Works with MICROSOFT Z-80 Softcard, Apple PASCAL and Visicalc software.

16K-32K Atari Memory Board

4116 RAM (200NS) Compatible with Atari 400/800

Parallel/Seriel Data Buffer Converter

Interfaces with most computers and printers on the market today. Switch selectable parallel or serial input/parallel or serial output. Data buffer 2K standard — expandable to 62K. Serial I/O BAUD rates switch independently selectable.

Atari Parallel or Serial Printer Cables

Pre-tested. Centronics or (RS-232) compatible.

APPLE PARALLEL INTERFACE CARD

Quantity and OEM discounts available.

Continuing our quest for excellence.

TRS-80 is a trademark of Radio Shack, Inc.
Apple II is a trademark of Apple Computer, Inc.
Atari 400/800 are trademarks of Atari, Inc.
Microsoft is a trademark of Microsoft Consumer Products, Inc.
Z-80 is a trademark of Zilog, Inc.
Visicalc is a trademark of Personal Software, Inc.

MICROTEK inc.

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

Multitask: A Realtime, Multitasking Operating System Emulator

Hal Bredbenner
Raleigh, NC

Most home applications of microprocessors are very basic and straightforward with the micros spending 99% of their existence in loops waiting for a key depression or an interrupt. The majority of systems with interrupts use them just for utilitarian functions such as screen refreshes, clock increments, and keyboard scans. If you have a PET, Apple, Atari, etc., you know how the machine really cannot do more than one operation at a time, and it also has a hard time responding quickly to outside inputs. The BASIC program described by this article emulates a way to allow many seemingly concurrent operations to occur with a fast system response time to the outside world. Keep in mind that this is an emulator. To actually be realized, the concept would have to be written in machine code form; however, the model shows the concept on the screen where it can be analyzed and easily understood.

There are two terms which should be understood before we continue. The first is realtime. An ideal realtime system is one that responds to a changed input immediately. This response will be a change in an output condition or an internal recognition of the input change. Ideally, this response is immediate. However, in reality, some time elapses before the realtime system can respond. The faster the response time, the more efficient the system.

The second term to be defined is multitasking. An ideal multitasking system is one that allows multiple operations to take place simultaneously in one system. Obviously, a single micro can do only one thing at a time, but through scheduling of desired tasks and assigning priorities to each operation, the appearance of more than one action at a time is accomplished.

For example, let's design a hypothetical simple realtime, multitasking system. The system will be a home security system that logs all its data and, upon command from a keyboard, produces a paper tape output of this data. It also has a CRT display and a battery back-up power system. The system must scan various inputs from the house and control outputs which would be interfaced to lights, alarms, and an electric fence. Since the electric fence poses a safety problem (for the good guys!), an Emergency Stop input would be needed. Internal DC power supplies should be monitored to detect system tampering and, if incoming AC power is out of tolerance (for example, a brownout), an alarm should be sounded. Prior to the back-up power system running out, the system should be shut down. When properly programmed, the system should monitor and accomplish all these tasks concurrently (if required) with a fast response time.

Logically breaking down the system software requirements, we can see that some tasks need to be done on a regular basis while others only need to be done upon command from the keyboard or other input. The regular tasks we will call Auto Rescheduling tasks. The tasks are called:

I0 DRIVER	**Reads inputs and writes outputs**
AC PWR CK	**Tests incoming AC power**
DC PWR CK	**Tests internal DC power**
READ KYBD	**Scans the system keyboard**
REFRESH	**Refreshes the CRT display**

These Auto Rescheduling tasks are regularly performed by the system and scheduled to be done again once they have been completed.

The remaining tasks are to be performed only when an outside input requires them. In our emulation, we can schedule one of these tasks by pressing its number on the numeric keys of the keyboard. In the hypothetical system, they would be initiated by the power fail detect circuitry, the system keyboard, or perhaps the Emergency Stop button. These tasks are called:

E-STOP	**Starts an emergency sequence**
PWR FAIL	**Initiates power down sequence**
MOVE FILE	**Transfers data to output buffer**
MEM TEST	**Exercise and test RAM memory**
PUNCH DATA	**Produces a paper tape**

The system would require at least these ten operations and, through the use of the emulator, we can see how the tasks are prioritized, scheduled, and executed.

In our system, we would require one master interrupt signal to drive the entire process. Each time this interrupt occurs, the operating system would perform the same actions. The first action is to read the status of all the system inputs and write, from a RAM buffer area, any new output data.

NEW COMMODORE ADD-ONS RAM-ROM: GETS RID OF SAFING ROM

***MX-910 CBM/PET RAM/ROM;**
Allows multi ROM protected programs using the same socket to be put onto diskette/cassette, no need to insert protect ROM in socket after initial load, eliminates need for ROM switch box, write protect in software, decoded for dual ROM socket usage, 4K expandable to 8K, easy internal CBM installation: $119.95

***MX-232 CBM/PET TO RS-232C INTERFACE:**
Low cost, bidirectional, 50 to 19,200 baud rate, full modem controls, parity allows for two RS-232C CBM ports, installs easily inside CBM: $199.95

SX-100 IEEE-488/PET MODEM SOFTWARE:
Best 810 modem software, by 8010 developer, works with Source/Micronet/CBM to disk/CBM to CBM: $79.95

***MX-200 IEEE-488/PET PARITY MODEM/SOFTWARE**
Talk to a host computer requiring parity, all features of SX232: $399.95

***MX-903 CBM/PET AUTORESET:**
Easily installed reset circuit: $19.95

ECX COMPUTER COMPANY
2678 NORTH MAIN ST.
WALNUT CREEK, CA 94596
(415)944-9277

CLEAR STAND

FOR
ATARI® 400/800*
APPLE®II**

- Beautifully polished high impact acrylic
- Holds up to 15" Dia. TV/monitor at eye level
- Open design allows maximum air flow
- Easy access to top sides & rear of processor

Dealer Inquiries Welcome

Send order with payment to:
BYTM Systems, Inc.
389 Fifth Ave. (Suite 400)
New York, N.Y. 10016

Allow 4-6 weeks delivery

Mail the coupon today

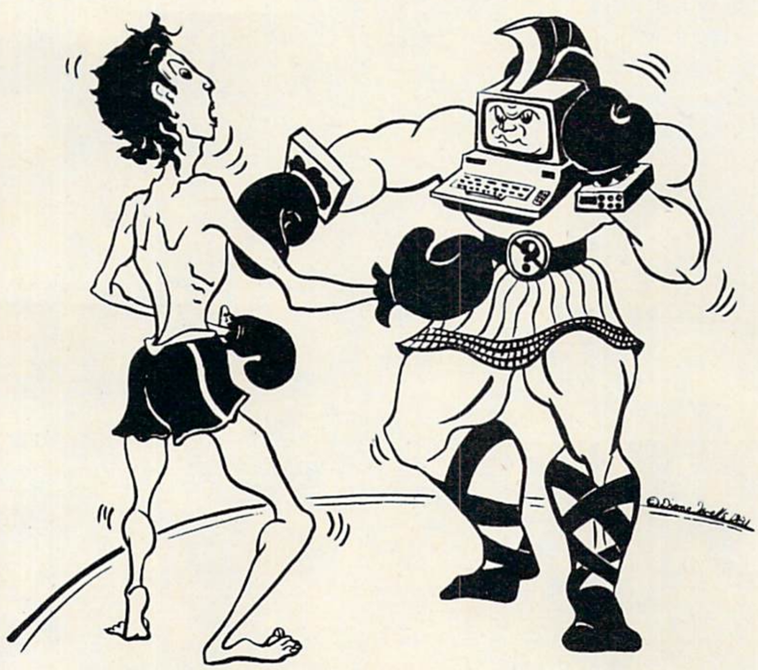
Send ______ Clear Stand @ $59.95
Add N.Y. Sales Tax for N.Y. Delivery

Ship to: ____________________
Address: ____________________
City: __________ State: ________ Zip: ________

*TM of Atari, Inc.
**TM of Apple Computers, Inc.

DON'T ASK PRESENTS:

ABUSE



For the APPLE II/II+ and the ATARI 400/800

Match wits with your computer in an insult war! With **ABUSE** your computer becomes a slightly demented, smart-aleck insult-exchanger.

- **Millions** of **different** insults.
- Understands and responds to user input.
- Hybrid BASIC/machine language program.
- Game feature: become a Master of **ABUSE.**
- **Many** surprises to discover.

Release your aggressions! Inflict **ABUSE** on anyone who's got it coming!

APPLE: REQUIRES 48K RAM AND APPLESOFT

ATARI: REQUIRES 40K RAM AND BASIC CARTRIDGE

At your computer store or send **$19.95** + $2.00 handling to:
DON'T ASK – 2265 Westwood Blvd. B-150 – Los Angeles, CA 90064 – (213) 397-8811

Calif. residents add 6% sales tax.

Dealer inquiries welcome

ATARI is a trademark of ATARI INC.
APPLE is a trademark of APPLE COMPUTER, INC.

CRYSTALWARE

ATARI 48K — w/disk

TRS-80 48 — w/disk

APPLE 48K — w/disk

THE FINEST IN FANTASY GAME SOFTWARE

"Having previewed over fifty of your competitors' games, I can assure you that your use of scrolling far exceeds anything I've seen for the Atari and, of course, for the Apple. I'm very impressed by the dedication and quality that your company exhibits by virtue of this demo." **David Sosna** — Associate Producer, Universal Pictures

Crystal has done its best to become the Porsche of the computer game industry. New scrolling techniques, video disk games, a real-life fantasyland — our mad programmers toil onward with little food or sleep to produce some incredible firsts in the microcomputer world. If you are an unappreciated genius and want to join our staff to help create the world of tomorrow today, give me a call. Our magazine *Crystal Vison* will within the next month have a circulation of 80,000 and we look forward very soon to producing our first full length motion picture. I'd like to thank my friends at Votrax and Axlon for giving us the tools (128K RAM for Atari and a vocal text synthesizer) to truly produce some programming miracles.

★ ★ ★ NEW RELEASES ★ ★ ★

THE CRYPT — One evening you awake at sunset to find yourself in what appears to be an endless cemetery. Although defenseless, you must somehow find your way out or perish from the hideous assaults of flesh eating zombies, rats, vampires, werewolves, and other repulsive monstrosities. To escape you may have to descend into the catacombs beneath the cemetery. This game is a little different from the others of our series because we use a lot of static graphics to set the mood. It is similar in some respects (without any copying intended) to those of our friends at On-Line who produce excellent static graphic adventures. You must use all your common sense and a great deal of courage to escape from this perilous adventure alive. We have made it so nearly impossible that the first player to do it successfully will receive a $200.00 prize. **$49.95 2 disks**

QUEST FOR POWER by Mark Benioff — An extraordinary game with the adventure and magic of Arthurian legend. Join Galahad as he leaves Camelot in search of the Scroll of Truth. Explore the treacherous depths of the Caves of Somerset, visit the medieval city of Essex. Along the way you will meet powerful wizards and great prophets. The villages of Sunderland and Leeds dot your path. Somewhere in an evil castle called Skenfirth, lurks the devil himself, while the Evil Giant Gogmogo, hungry for human prey, roams the forests. In Fantasyland tradition we include 64 full screens of hires scrolling and some sensational graphic and animation sequences. Well worth the **$39.95 1 disk**; enjoyable to all ages.

★ ★ ★ GALACTIC EXPEDITION ★ ★ ★

The year is 3021, almost 100 years since the expedition to the Sands of Mars has returned. The Starship Herman now rests quietly in the Zikon Museum in New Brisbane. It's nearly 80 years since World War III, the Ames Research Center celebrates its 150th anniversary, and you stand at the unveiling of a truly technological wonder — the first ion-propelled vessel, saucer shaped Lady Joanne, its viewport of pure diamond, its hull of synthetic emeralds. The Martian glyphs of the Meshim and those of Lemuria have now been deciphered and it appears that a much greater mystery is about to unravel. 7 planes and 7 doors — 7 guardians and 7 candles. 7 strange new worlds await the ultimate adventurer to unlock a timeless secret. The starship may seem strange and unfamiliar to our veteran adventurers, faced with its marvelous new technology; this craft must be flown by constant monitoring of ion stabilizers. During your galactic expedition you are surrounded by the flickering heavens, beset by meteor showers and time-warps. Each unique world holds one of the 7 keys to unlock the Great Mystery. The games all run off the Main Module which also is a game unto itself.

From Earth to Moon — On the Moon's dark side lie entrances to caverns extending to the moon's hollow core which contains a timeless secret. Here live a race of burrowing creatures, who have built vast earthen cities with storehouses full of precious stones. Gravity is extremely critical and you must use all your skills to manually land your craft. This first Master Disk contains the dos needed to run additional scenarios. Its price is **$39.95** and includes 64 screens of Hires graphics.

Mists of Venus — On Venus' ever hot surface are endless jungles and swamps. The air is unbreathable and spacesuits and oxygen must be carried. This world is especially treacherous with all sorts of loathsome creatures and hardly any place dry enough to land your ship. Beneath the green seas our adventurer may find the second key to solving the Mystery. **$29.95** (must have Master Disk to run)

Planet Herman — It is hard to tell where Herman's atmosphere ends and the surface begins. Much of this adventure will have the feeling of a starship submarine. Navigating around Herman is very dangerous but with a computer on board Lady Joanne it may be just possible. This senario costs **$29.95** and needs the Master to run.

The Asteroid Belt — Every play something oids. A combination of the best machine language sub-routines of our new Crystaloids with a fast moving adven ure game. Penal colonies, lurking pirates, and some unusual forms of scavenger life exist here. It's difficult to travel in the Asteroid Belt without getting blown up. Perhaps you should find some expert help by rescuing a pilot, who is also a sentenced thief or murderer, from one of the penal colonies. There are places for trading and you may wish to indulge yourself with a visit to the sensual Pleasure Planet. **$29.95** (needs Master Disk)

Uranus - World of Ice — A freezing place with nights of —200° F. Bring along Thermasuits, as well as some Laars with which to battle the Grungik, a 12 foot tall relative of Big Foot, fond of human flesh. Uranus also has a secret inner labyrinth with tropical flora and fauna. However, the King of the Ice Planet, Norion may have his own idea about your trespassing. Without proper clothing, weapons and supplies, your stay here may be very exciting and very short. **$29.95** (needs Master Disk to run)

Jupiter - World of Dwarfs — How would it feel to weigh 300 or so lbs.? A trip to Jupiter should fill you in fast. There is a particularly interesting red spot on Jupiter and a curious set of moons. Picking up some antigravs will help. Landing should really tax your energies. In the Jupiterian atmosphere, you fall fast! Be prepared to use 10 times the normal amount of fuel. Better find the 6th key quickly before your fuel and food are exhausted. **$29.95** (needs Master Disk)

The Crystal Planet - You will have to embark on this final portion of your expedition ignorant of what you may encounter here on this mysterious planet, excepting that the 7th world holds the ultimate key to winning the contest. **$29.95** (needs Master Disk)

The Contest — To the Winner with the highest score, who solves the mystery by November of 1982 will go $5000.00 in cash. Good Luck!

★ ★ ★ ★ ★ ★

GLAMIS CASTLE — According to ancient legend and records this castle is one of the most haunted sites in Great Britain. One Lady Glamis, known to be in league with the devil, liked to send out a destructive demon to harrass the townspeople. She finally was burnt at the stake on Castle Hill, cursing as she died all future generations of the Lyon family. Her demon still seems to haunt that spot, murdering the curious who stray up to Castle Hill after dark. The curse stipulated that each succeeding generation would have at least one child, often female, who would be a vampire. When an heir comes of age, there is a secret ceremony in which the heir, his father, and the steward take crowbars and chip away plaster concealing a hidden chamber, known only to them, that Earl Patie used when he gambled with the devil. Another tradition says that a creature, half-man, half-beast stalks the passages in the walls of Glamis to insure the fulfilling of the curse. The mystery, of course, is to determine the location of this secret chamber. Our game, occupying 2 disks, will have as exact a replica of the castle as possible. It's definitely one of a kind! And we will be offering a $500 prize to the first person daring enough to solve the centuries-old mystery of Glamis Castle. **$49.95 2 disks.**

Crystal has many other fine fantasy and space games. For a copy of CRYSTAL VISION which includes a complete catalog please send $3.00 to the address below.

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

(408) 778-2966 **CRYSTAL COMPUTER** **17120 Monterey Rd., Morgan Hill, CA 95037**

master charge · VISA

The next action of the operating system (OS) will be to determine if the inputs demand the scheduling of any tasks. For example, let's suppose the person using the system entered a request for a memory test from the keyboard by pressing an "M." The OS would recognize this as a MEM TEST request and read, from a lookup table, three things about the task: the task location, its execution time, and a basic priority value.

The three pieces of data about the task and the time of its request would then be entered into a queue table. This queue table is a list of all the tasks that are scheduled in the system at any one time. When all the required new tasks are entered into the queue, the OS then assigns a calculated priority to each task. This is done in various ways, some extremely complex; however, the most basic way is to make the calculated priority a function of both the task's basic priority and its elapsed time in the queue. This simple calculation is done in our emulator. After a new calculated priority value has been entered into the queue for each pending task, the OS then looks again at the calculated priorities and merely selects the most urgent task, the one with the highest priority, to be performed. The OS then passes control to that task. The selected task runs on the micro until it finishes or another interrupt occurs and, in either case, a return to the OS is made and a new task begun.

An OS like the one we have just described will respond quickly to any input and that is the main design goal for a realtime system.

Notice that if a task is running and another task is calculated to have a greater priority, the first task can be suspended while the more urgent task runs. This can happen if one task has been waiting in the queue for some time while another executing task has most of the processor time. Again, this determination of task suspension can be extremely complex, yet in our emulator it is made solely on the basis of the calculated priorities of the tasks. It also should be noted that because of memory resources, scheduling queues are limited in size. Because of this, any tasks that are requested when the queue is full are ignored and must be requested again later.

To be actually designed in a system, this type of OS requires careful planning. One consideration is the frequency of occurrence of the master interrupt signal. The maximum response time to any input would be one cycle of the interrupt and yet too fast an interrupt will tend to bind the processor down with OS tasks instead of real life tasks. Another consideration is that separate stack pointers should be kept for all active, scheduled, or suspended tasks so that, at any time, resumption of that task's execution will not be ruined by some other task's dealings. The use of a RAM buffer for output storage allows any programmed task to see what is in an output port and, if that RAM output buffer location is changed or modified during next interrupt cycle, the OS will automatically write that new data to the port. This is a good way of synchronizing output and also preventing interference between different program modules.

This emulator program is a model of the operating system required by the hypothetical home security system we talked of earlier. The emulator is written in BASIC and obviously is not as fast as the machine code OS would be. However, the basic design of the OS is graphically shown and is simple to understand. When run, the emulator displays the active task, its time of execution, the average time of response to those tasks in the queue, the entire scheduling queue, and a list of the available system tasks (See Figure 1). The Auto Rescheduling tasks are initially placed in the queue and the OS begins highlighting the active task as it "executes." At any time, by pressing one of the available task numbers (0-3 and 7), a new task will be added to the queue and serviced as the OS permits. The queue in the emulator will hold only 10 entries and then a "QUEUE FULL" response will be given to further inputs. Notice that the Auto Rescheduling tasks are added again to the queue as they are completed. Line 2335 of the program is the algorithm used to calculate the priority of the tasks in the queue. Experimentation with different prioritizing schemes will produce some very interesting results. Try your own algorithm and compare the average response times.

The realtime, multitasking Operating System Emulator given here requires less than 4K of memory and can be run on any Commodore system (most other systems would only require the modification of the cursor positioning characters). It is an excellent tutorial program that graphically shows how some of the most complex OS actually do what they do. Microcomputers can become as powerful as minis and mainframes with this kind of programming. I urge you to try to accomplish this type of OS in machine code. The resulting power of the microprocessor would be amazing.

Program 1.

```
110 REM  MULTITASKER EMULATOR
111 REM  WRITTEN BY HAL BREDBENNER
120 REM
130 FORTN=0TO9
140 READAR(TN),EX(TN),PR(TN),TN$(TN)
150 NEXTTN
160 DATA0,100,1,"E-STOP   ",0,100,1,"~
   ~PWR FAIL ",0,90,3,MOVE FILE
170 DATA0,50,8,"MEM TEST ",1,30,3,IO ~
   ~DRIVER,1,30,9,AC PWR CK
```

```
180 DATA1,30,9,DC PWR CK,0,90,7,PUNCH~
   ~DATA,1,30,2,READ KYBD
190 DATA1,30,3,"REFRESH  "
200 REM
210 FORQN=0TO4
220 READQT(QN),QT$(QN),QE(QN),QP(QN),~
   ~QA(QN),QW(QN)
230 PT(QN)=INT((QW(QN)/QP(QN))*100)
240 NEXTQN
250 DATA9,"REFRESH  ",30,2,1,0,8,READ~
   ~ KYBD,30,2,1,0
260 DATA6,DC PWR CK,30,9,1,0,5,AC PWR~
   ~ CK,30,9,1,0
270 DATA4,IO DRIVER,30,2,1,0
290 PRINT"{CLEAR}{REV}MULTI-TASKING O~
   ~PERATING SYSTEM EMULATOR"
300 PRINT"{03 DOWN}{REV}SCHEDULER{OFF~
   ~}
310 PRINT"{14 DOWN}{REV}AVAILABLE TAS~
   ~KS...{DOWN}
320 PRINT"0- E-STOP     1-PWR FAIL    ~
   ~2-MOVE FILE
330 PRINT"3-MEM TEST    4-IO DRIVER*  ~
   ~5-AC PWR CK*
340 PRINT"6-DC PWR CK* 7-PUNCH DATA  ~
   ~8-READ KYBD*
345 PRINT"9-REFRESH*     *-AUTO RESCHE~
   ~DULING{02 UP}
350 PRINT"{HOME}{05 DOWN}{REV}TASK NA~
   ~ME{OFF} {REV}TIME LEFT{OFF} {REV}~
   ~PRIORITY{OFF} {REV}TIME IN QUEUE{~
   ~OFF}
1000 REM-------------------
1010 REM REAL OPERATING SYSTEM AREA
1020 REM-------------------
1025 Q=0
1030 GOSUB2800:REM ADVANCE ACTIVE TAS~
   ~K
1040 GOSUB2500:GOSUB2700:REM ADVANCE ~
   ~QUEUE
1070 GOSUB2500:REM GET TASK AND ADD
1080 IFQE(Q)<10THEN1300
1085 GOSUB2400:REM PACK QUEUE TABLE
1090 GOSUB2500:GOSUB2300:REM DETERMIN~
   ~E HIGHEST
1100 IFPT(XP)>PT(Q)THENQ=XP
1110 GOSUB2500:GOSUB2000:GOTO1030
1300 REM-----------------
1310 REM DELETE FINISHED TASK
1320 REM-----------------
1322 CC=CC+1:TC=TC+QW(Q):AC=INT(TC/CC~
   ~*100)/100
1325 FT$=QT$(Q)+" "+STR$(AC)+"MSEC."
1330 QT$(Q)=""
1335 IFQA(Q)=1THENTN=QT(Q):GOSUB2570
1337 GOTO1085
2000 REM-----------------------
2010 REM DISPLAY QUEUE
```

```
2020 REM-------------------------
2025 GOSUB2100
2030 PRINT"{HOME}{05 DOWN}
2050 FORQN=0TO9
2060 IFQT$(QN)=""THENPRINT"          ~
~                              ":GOTO~
 ~2080
2065 PRINTTAB(10)"
 ~          {UP}"
2066 PT(QN)=INT((QW(QN)/QP(QN))*100)
2067 IFQN=QTHENX$="{REV}":GOTO2070
2068 X$="{OFF}"
2070 PRINTX$QT$(QN)"{OFF}"," "QE(QN),~
 ~" "PT(QN)," "QW(QN)
2080 NEXTQN
2085 PRINTQF$
2090 RETURN
2100 REM-------------------------
2110 REM DISPLAY ACTIVE TASK
2120 REM-------------------------
2130 PRINT"{HOME}{02 DOWN}{REV}ACTIVE~
 ~ TASK:{OFF} "QT$(Q)"  {REV}TIME L~
 ~EFT:{OFF}          {09 LEFT}"QE(Q)~
 ~
2135 PRINTTAB(10)"{REV}AVERAGE RESPON~
 ~SE:{OFF}          {07 LEFT}"AC
2140 RETURN
2200 REM-------------------------
2210 REM    DISPLAY TIME
2220 REM-------------------------
2230 PRINT"{HOME}{04 DOWN}"TAB(27)"{R~
 ~EV}TIME:{OFF}";
2240 PRINTLEFT$(TI$,2)+":"+MID$(TI$,3~
 ~,2)+":"+RIGHT$(TI$,2)
2250 RETURN
2300 REM-------------------------
2310 REM DETERMINE HIGHEST Q PRIORITY~
 ~
2320 REM-------------------------
2330 X=0:FORQN=0TO9
2333 IFQP(QN)=0THENGOTO2350
2335 PT(QN)=INT((QW(QN)/QP(QN))*100)
2340 IFPT(QN)>XTHENX=PT(QN):XP=QN
2350 NEXTQN
2360 REM EXIT WITH XP=HIGHEST PRIORIT~
 ~Y
2370 RETURN
2400 REM-------------------------
2410 REM  PACK QUEUE TABLE
2420 REM-------------------------
2430 FORQN=0TO9
2440 IFQT$(QN)=""THENQT(QN)=0:QE(QN)=~
 ~0:QP(QN)=0:QA(QN)=0:PT(QN)=0:QW(Q~
 ~N)=0
2450 NEXTQN
2460 FORQN=0TO9
2470 IFQT$(QN)<>""THENNEXTQN:GOTO2490~
 ~
2480 QT$(QN)=QT$(QN+1):QT(QN)=QT(QN+1~
 ~):QE(QN)=QE(QN+1):QP(QN)=QP(QN+1)~
 ~
2485 QA(QN)=QA(QN+1):QW(QN)=QW(QN+1)
2486 QT$(QN+1)=""
2487 NEXTQN
2490 REM TABLE IS NOW PACKED
2495 RETURN
2500 REM-------------------------
2510 REM  GET TASK AND ADD TO QUEUE
2520 REM-------------------------
2525 QF$="                   "
2530 GETX$:IFX$=""THENRETURN
2535 IFX$="4"ORX$="5"ORX$="6"ORX$="8"~
 ~ORX$="9"THENRETURN
2540 REM ADD TASK TO QUEUE
2550 IFVAL(X$)<0ORVAL(X$)>9THENRETURN~
 ~
2560 TN=VAL(X$)
2570 GOSUB2400:REM PACK QUEUE TABLE
2580 I=0
2590 IFI>9THENQF$="{REV}QUEUE FULL!{O~
 ~FF}":GOTO2620
2595 IFQT$(I)<>""THENI=I+1:GOTO2590
2600 QT(I)=TN:QT$(I)=TN$(TN):QE(I)=EX~
 ~(TN):QP(I)=PR(TN):QA(I)=AR(TN)
2610 QW(I)=0:PT(I)=INT((QW(I)/QP(I))*~
 ~100)
2620 GOSUB2000
2630 RETURN
2700 REM-------------------------
2710 REM INC QUEUE AND PRIORITIES
2720 REM-------------------------
2730 FORQN=0TO9
2740 IFQT$(QN)=""THENNEXTQN:GOTO2780
2750 QW(QN)=QW(QN)+1
2760 PT(QN)=INT((QW(QN)/QP(QN))*100)
2770 NEXTQN
2780 RETURN
2800 REM-------------------------
2810 REM ADVANCE ACTIVE TASK
2820 REM-------------------------
2830 QE(Q)=QE(Q)-10
2850 PT(Q)=INT((QW(Q)/QP(Q))*100)
2860 RETURN
READY.
```

Figure 1.

```
MULTI-TASKING OPERATING SYSTEM EMULATOR

ACTIVE TASK: MOVE FILE  TIME LEFT: 60

SCHEDULER AVERAGE RESPONSE: 12.57
TASK NAME  TIME LEFT  PRIORITY  TIME IN QUE

DC PWR CK   30        344        31
AC PWR CK   30        344        31
MEM TEST    50        262        21
MOVE FILE   60        666        20
PUNCHDATA   90        285        20
READ KYBD   30        450        9
REFRESH     30        200        6
IO DRIVER   30        100        3


AVAILABLE TASKS:

0- E-STOP    1-PWR FAIL    2-MOVE FILE
3-MEM TEST   4-IO DRIVER*  5-AC PWR CK*
6-DC PWR CK* 7-PUNCH DATA  8-READ KYBD*
9-REFRESH*   *-AUTO RESCHEDULING
```

©

THE apple® GAZETTE

Plotting Polar Graphs With The Apple II

Marvin L. De Jong
The School of the Ozarks
Pt. Lookout, MO 65726

You do not need long programs to make a computer perform a useful task in teaching mathematics. One of the more arduous tasks in trigonometry or analytic geometry is graphing functions in polar coordinates. For many polar curves, this task takes a lot of time, and not much learning takes place. On the other hand, it is an ideal task for the computer, the program to plot a polar graph is easily understood by the students, and it gives them a tool with which they can experiment with many graphs. Program 1 shows the simplest possible version. We shall discuss it shortly, but first here is a brief explanation of what we are trying to accomplish.

Suppose we have a relation between R, the distance from a point called the *pole,* and θ (Greek symbol theta), the angle measured counterclockwise from the *polar axis.* The pole is analogous to the *origin* in X-Y Cartesian coordinates, and the polar axis lies along the *X-axis.* The relation between R and θ is usually described by an equation of the form

R = F(θ).

The equation

R = 90*SIN(2*θ)

is one example. Refer to Figure 1 for an illustration of some of these concepts, including a graph of the equation R=90*SIN(2*θ), called a *four-leaved rose.*

The key to using a computer to graph polar coordinates is the transformation formulas

X = R*COS(θ)
Y = R*SIN(θ)

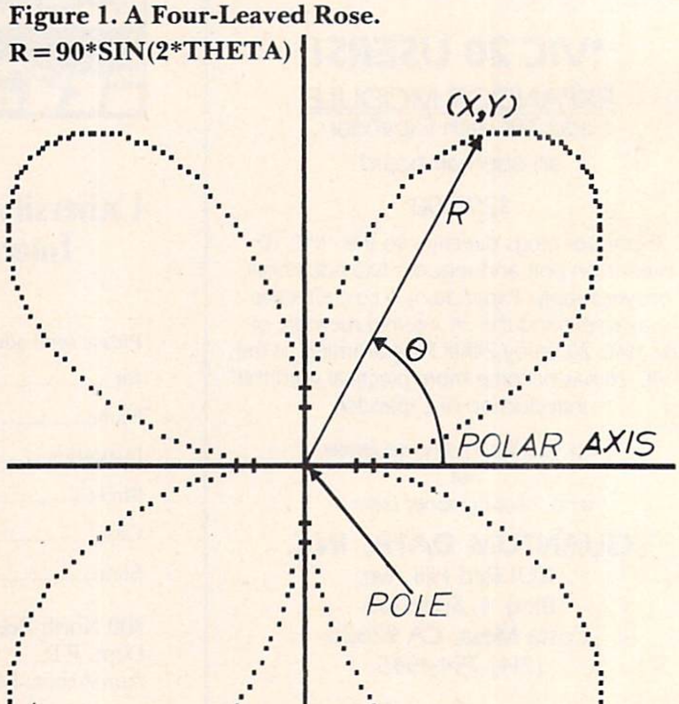
and, of course, the computer's ability to perform a PLOT X,Y instruction.

A "bare bones" approach to plotting polar graphs is given in Program 1. The student inputs the starting angle and the angle at which the graph is to end. These angles are in degrees. Line 30 initializes the HIRES mode with text on the lower part of the screen of the video monitor. Line 60 converts the angle to radians (pi radians = 180°).

Line 70 in Program 1 is the equation to be graphed. The entire program may be left unchanged while line 70 is modified to graph a large variety of polar functions.

Line 90 and 100 convert the polar coordinates (R,θ) to X-Y coordinates. Note that since the origin of the Apple II coordinate system is in the upper left-hand corner of the screen, we have *translated* it so the origin of our coordinate system is at (85,85). Furthermore, since Y is positive *downward* on the Apple, and we would prefer the more traditional "Y positive upward" convention, we use a negative sign in the Y-transformation equation. The results are plotted with the instruction on line 120. The instruction on line 130 increments the angle by one degree. Points will be continued to be plotted until

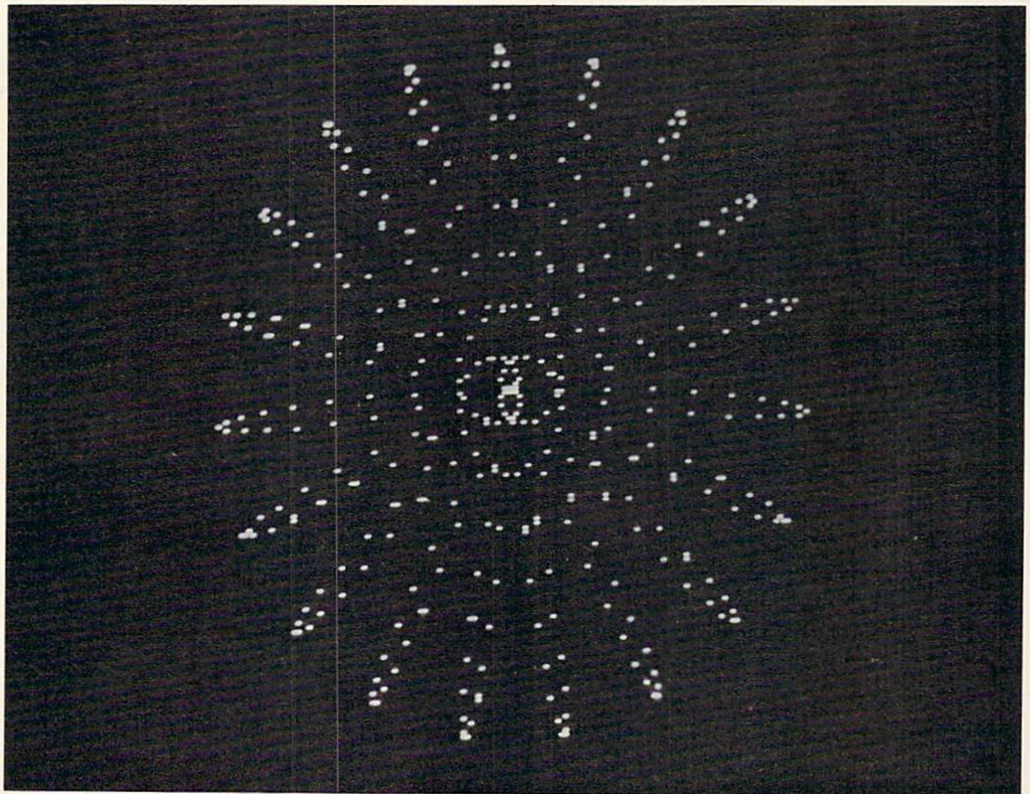
Figure 1. A Four-Leaved Rose.
R=90*SIN(2*THETA)



the angle exceeds the ending angle. The student can watch the points being plotted and see the corresponding R and θ values printed underneath the graph.

A photograph of the screen of the video monitor after the graph R = 85*SIN(19*THETA) was

Figure 2: A Nineteen-Leaved Rose



plotted is shown in Figure 2. Notice that different X and Y scale factors on the screen produce a slight distortion that is not important as far as the present application is concerned.

Of course, it is always possible to add a few bells and whistles. Program 2 represents a few non-essential, but nice, additions to the first program. The coordinate axes are drawn and the X and Y values are rounded to their nearest integer values before plotting. Also, we have made use of the entire screen with the HGR2 instruction on line 30. The scale of the graph was reduced so that we could plot the finished result on our little printer. If you are using a video monitor or a large printer, then you will want to keep the scale as large as possible (replace all the 80's with 90's).

Some of our results are given in the figures that follow. In Figure 3 we show a graph of R=80*SIN(3*THETA) a *three-leaved rose.* Figure 4 is a graph of a *13-leaved rose,* R=80*SIN(13*THETA). The *cardioid* R=40*(1+COS(THETA)) is illustrated in Figure 5. Figure 6 is the famous *Spiral of Archimedes,* R=6*THETA. Figure 7 is similar, but not identical to the *Limacon of Pascal.* We chose R=80*COS(THETA/3) for this figure. Figure 8 illustrates the *Litus* described by the equation R=25*(2+SIN(3*THETA)). Figure 9 has no name, but its equation is R=25*(2+SIN(3*THETA)).

Finding where two polar curves intersect is sometimes difficult. If you have a printer you can simply graph the polar curves, overlay their graphs, and find approximate points of intersection.

Students seem to enjoy working with these programs. They are simple enough so the students can modify the various parameters rather easily, giving them a chance to experiment freely. At the

Figure 3. A Three-Leaved Rose.

R=80*SIN(3*THETA)

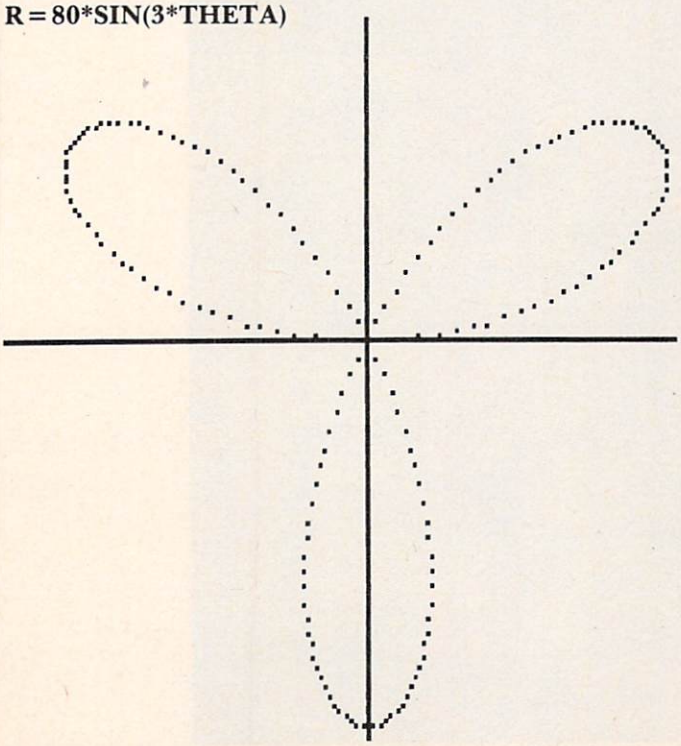


Figure 4. A Graph of R=80*SIN(13*THETA), A 13-Leaved Rose.

R=80*SIN(13*THETA)

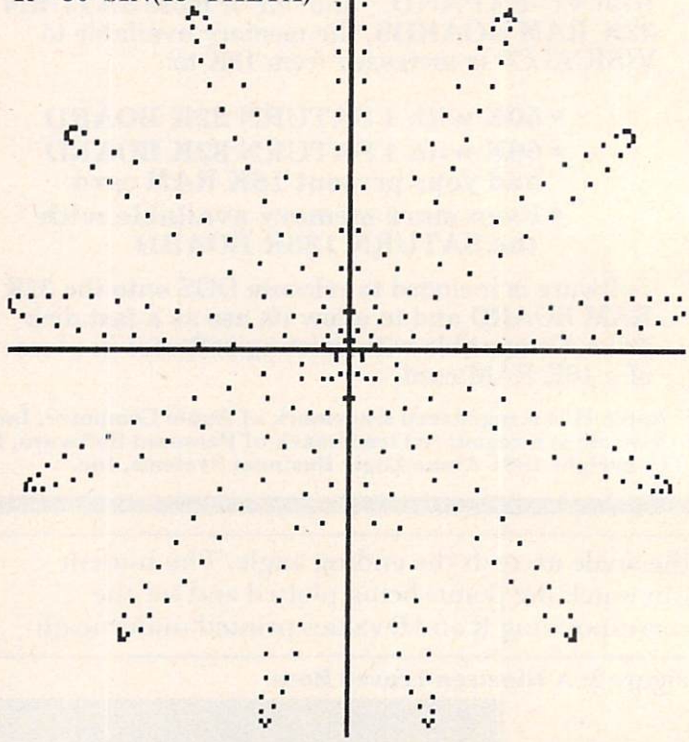
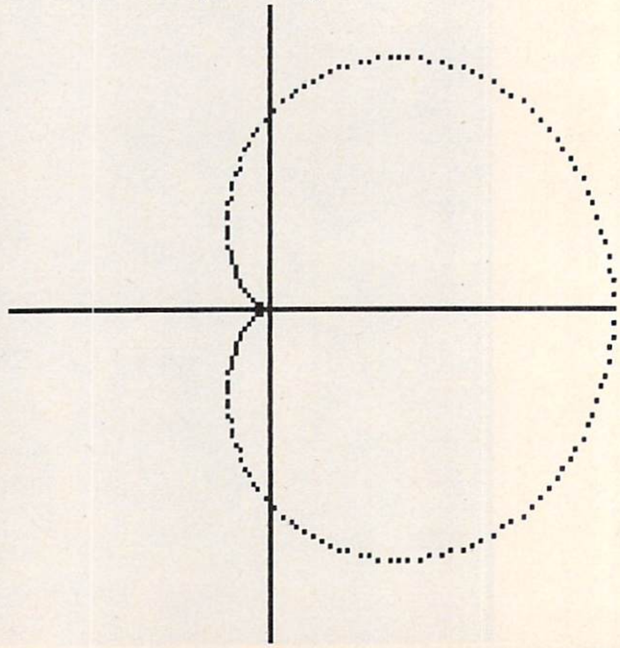


Figure 5. The Cardioid R=40*(1+COS(THETA)).

R=40*(1+COS(THETA))



very least, the programs release them from the drudgery of plotting points by hand.

Figure 6. Spiral of Archimedes with R=6*THETA.

R=6*THETA

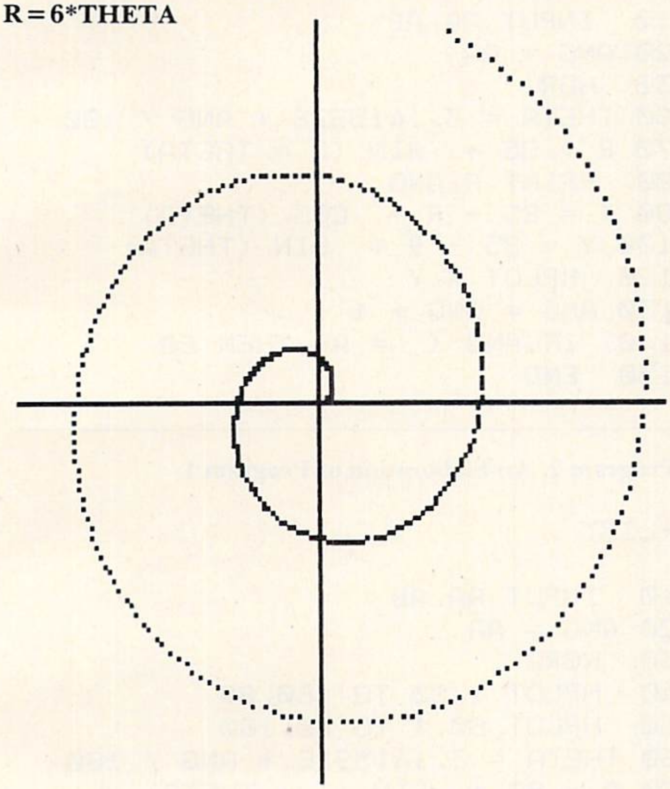


Figure 7. A Graph of R=80*COS(THETA/3).

R=80*COS(THETA/3)

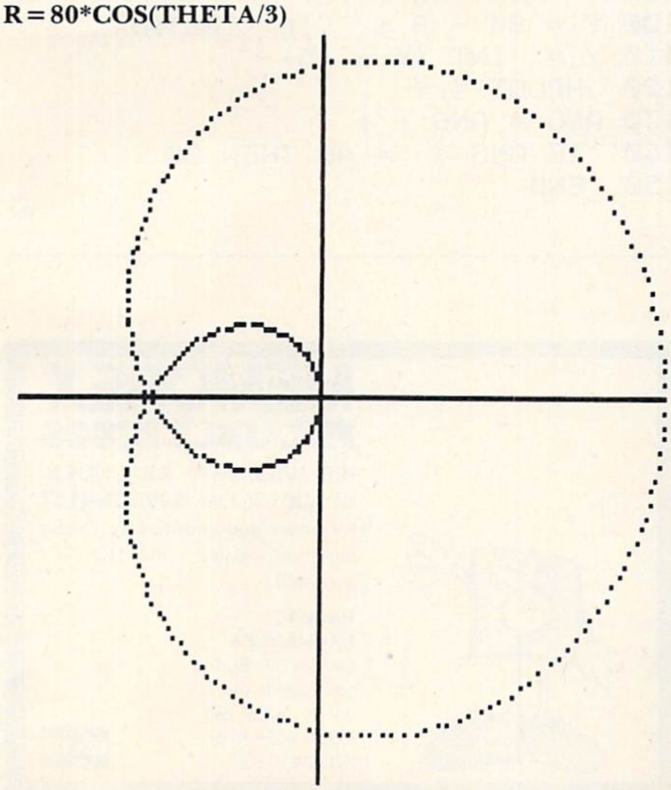


Figure 8. A Graph of R = SQR(3600/THETA).

R = SQR(3600/THETA)

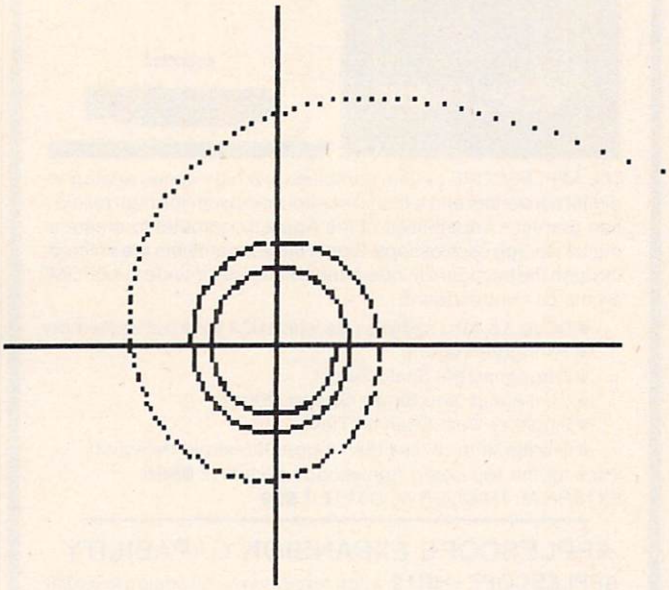
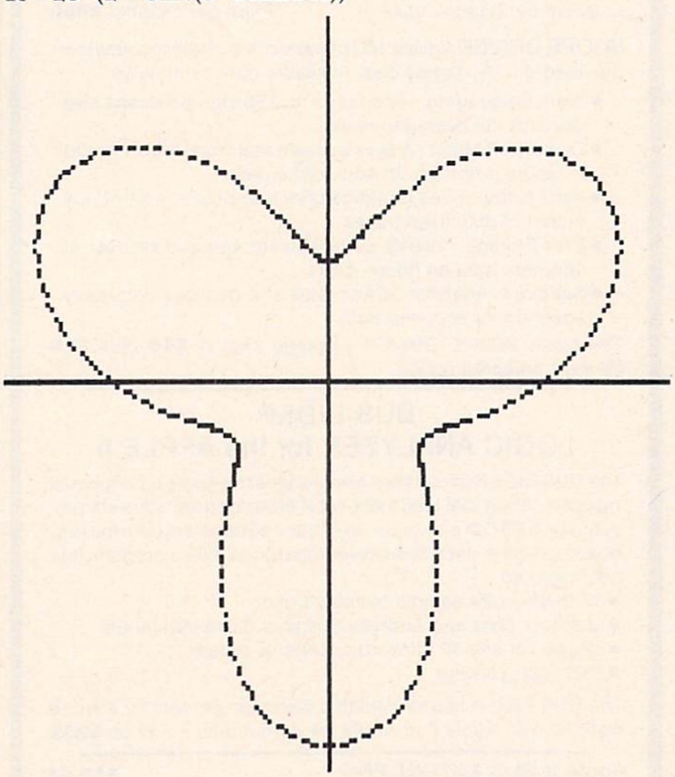


Figure 9.
Untitled Graph with R = 25*(2 + SIN(3*THETA)).

R = 25*(2 + SIN(3*THETA))



COMPUTE! The Resource.

Program 1. A Simple Program to Graph Polar Functions

```
←LIST

10  INPUT AA,AB
20 ANG = AA
30  HGR
60 THETA = 3.1415926 * ANG / 180
70 R = 85 *  SIN (2 * THETA)
80  PRINT R,ANG
90 X = 85 + R *  COS (THETA)
100 Y = 85 - R *  SIN (THETA)
120  HPLOT X,Y
130 ANG = ANG + 1
140  IF ANG <  = AB THEN 60
150  END
```

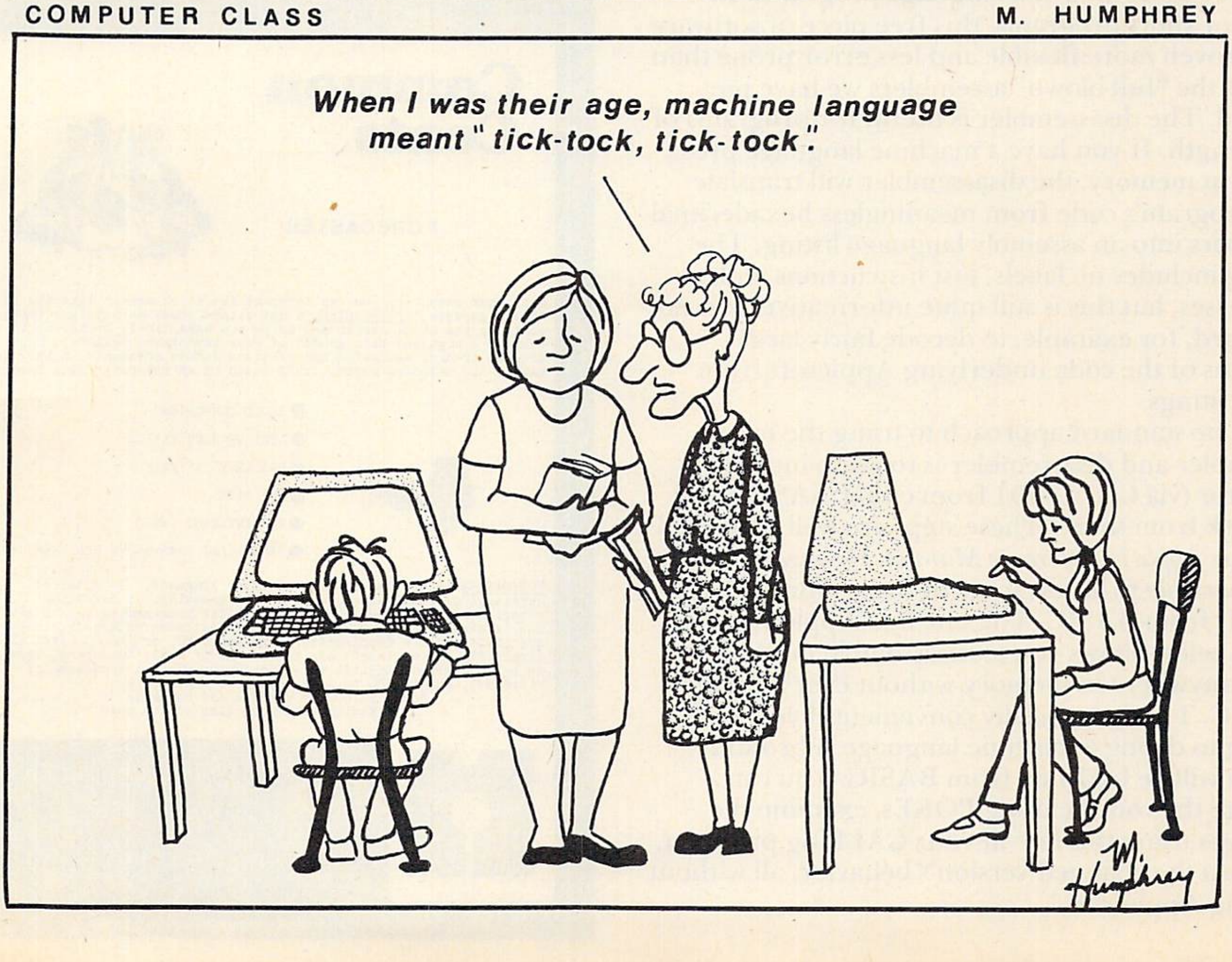
Program 2. An Elaboration of Program 1.

```
←LIST

10  INPUT AA,AB
20 ANG = AA
30  HGR2
40  HPLOT 1,80 TO 160,80
50  HPLOT 80,1 TO 80,160
60 THETA = 3.1415926 * ANG / 180
70 R = 80 *  SIN (3 * THETA)
80 X = 80 + R *  COS (THETA)
90 X =  INT (X + .5)
100 Y = 80 - R *  SIN (THETA)
110 Y =  INT (Y + .5)
120  HPLOT X,Y
130 ANG = ANG + 1
140  IF ANG <  = AB THEN 60
150  END
```

©

COMPUTER CLASS M. HUMPHREY



Disassembling Machine Language Programs Without Leaving BASIC

John R. Vokey and H. Cem Kaner
McMaster University
Hamilton, Canada

One of the nice features of the Apple computer is that it has a built in mini-assembler and disassembler. The mini-assembler is all you need for entry of short machine language programs. In fact, for short programs, this free piece of software has proven more flexible and less error prone than two of the "full blown" assemblers we have purchased. The disassembler is useful for programs of any length. If you have a machine language program in memory, the disassembler will translate the program's code from meaningless hexadecimal numbers into an assembly language listing. The listing includes no labels, just instructions and addresses, but this is still quite informative. It is not too hard, for example, to decode fairly large sections of the code underlying Applesoft from such listings.

The standard approach to using the mini-assembler and disassembler is to jump into the monitor (via CALL -151 from either BASIC) and to work from there. These steps are well described in your *Apple II Reference Manual.* However, it is also possible to access some of these monitor commands from BASIC. The one line Applesoft program below allows you to disassemble machine code anywhere in memory without ever leaving BASIC. This is especially convenient if you are trying to debug a machine language subroutine which will be CALLed from BASIC. You can change the routine using POKEs, examine the changes using this line in your CALLing program, and test the changed version's behavior, all without leaving Applesoft.

The program works by passing the user-specified START location of the code to be disassembled to the monitor program counter (labelled PC in the program). It then calls the monitor LIST subroutine which we label disassemble in the program. This routine disassembles the next 20 lines of machine code, incrementing the monitor program counter locations appropriately, and returns control to BASIC. The BASIC program then compares the value of the monitor program counter to the user specified value FINISH. If there is more to be done before location FINISH is reached, the program waits until you press any key, then continues the listing. Once FINISH is reached, the program ends.

As an example of the use of the program, if you set START to 65118 and set FINISH to 65140, you will disassemble the disassembler.

```
1000 DISASSEMBLE = 65121: PC = 58:
POKE PC, START - INT (START / 256)
* 256: POKE PC + 1,START / 256: FOR
I = 0 TO 1: HOME: CALL DISASSEMBLE:
PRINT: PRINT TAB (13);"<PRESS ANY
KEY>": GET Z$: I = ( PEEK (PC + 1)
* 256 + PEEK (PC)) > FINISH: NEXT I
```

©

Named GOSUB With Variable Passing

Mike Smith
Calgary, Canada

In **COMPUTE!** # 12, I described a machine language program which would allow subroutines to be called by name rather than by number. This article is an extension of that idea. It describes a machine language program which allows parameters to be passed in and out of subroutines.

One of the nicer features of FORTRAN and PASCAL is their ability to pass variables into a subroutine. This feature is very useful when you wish to do the same operation on a large number of variables. Passing parameters into subroutines is convenient since the variable names used outside the subroutine don't have to be the same as used for the calculation within the subroutine. This makes programming and documentation easier. In addition, subroutines of this type can be used as a sort of multi-line function.

A Brief Example Of Parameter Passing

Suppose that you wish to perform a complicated operation upon variables A, B and C and have the answer returned in D. Then you wish to have the same operation performed upon the variables A1, B1 and C1 and have that answer returned in D1.

In FORTRAN that program would look like this:

```
CALL COMPL(A,B,C,D)
     (call subroutine with first variable set)
CALL COMPL(A1,B1,C1,D1)
     (then with the second set)
.........
(Use D and D1 in calculations)
.........
SUBROUTINE COMPL(W,X,Y,Z)
     (use dummy variables with subroutine)
(Complicated calculation using W, X and Y)
.........
Z = .....
RETURN
```

In Applesoft BASIC things are a little more difficult. First, you must call the subroutine by a *number* rather than by a *name*. A second problem is that you can't pass the names of variables into the subroutine. Instead, you must move (reassign) the values into the variable names used in the subroutine. An equivalent Applesoft BASIC program would look something like:

```
10  W=A : X=B : Y=C
    (reassign first set of variables)
20 GOSUB 1000
30 D=Z
40 W=A1 : X=B1 : Y=C1
    (reassign second set)
50 GOSUB 1000
60 D1=Z
70 .........
    (Use D and D1 in calculations)
    .........

1000 (Complicated calculation using W, X and Y)
     .........
1100 Z= .....
1110 RETURN
```

Having to remember the subroutine number is no great problem if you are the person who did the programming, provided you only did the programming a week or so ago, and have not yet forgotten what subroutine number was needed for what. Having to reassign variables, as in statement 40, is no great problem either, provided you don't have a large number of different variables that need to be worked on. But why do something that the computer can make easier to understand and do?

The program described in this article uses the Applesoft BASIC *ampersand* command (&) to allow the naming of subroutines and the easy passing of numerical data. With the machine code routine installed in memory, the Applesoft program above becomes:

```
10  COMPL = 1000
    (establish the subroutines name)
20  & GOSUB COMPL !COMPL(0),A,B,C,D!
    (pass the parameters)
30  & GOSUB COMPL !COMPL(0),A1,B1,C1,D1!
40  .........
    (Use D and D1 in calculations)
    .........

1000  & GET !COMPL(0),W,X,Y,Z!
      (identify the dummy variables)
1010  (Complicated calculation using W, X and Y)
      ........
1100  Z = ....
1110  & RETURN !COMPL(0)!
```

In addition to passing parameters, Applesoft will now support GOTO and GOSUB statements that have names instead of numbers. For example

```
JUMP = 1000 : & GOTO JUMP or COMPL = 1000 :
    & GOSUB COMPL
FIRST = 1000 : DEUX = 2000 : ON X GOSUB FIRST,
    DEUX
```

I decided to develop this parameter passing routine because I am repeatedly asked to translate FORTRAN program with subroutines into Applesoft. Most of those subroutines pass variables. Making sure that I didn't duplicate names and that

I reassigned the right variable, was too much of a hassle. Hence this routine.

Loading The Program

The machine language program as described in this article is too long to put in a normally unused area of memory. The cassette buffer (at $300) will only accept around $CF locations before running into the DOS pointers at $3D0.

The program could be placed high in memory, just below the normal HIMEM. The HIMEM pointers must then be adjusted so that the program is not touched by Applesoft when strings are used. However, this means that people using 48K and 32K Apples, with or without the Program Line Editor at the top of memory, will all need different programs. The modifications are simple, if you know how. Therefore, I have adopted the technique of moving LOMEM up $200 bytes and storing the machine language code in the space created. Then everybody gets the same code.

Before entering the demonstration BASIC program, type:

POKE 104,10 : POKE 2560,0 : NEW

These three instructions adjust LOMEM and the various Applesoft RUN, LOAD or SAVE programs. The pointers can be shifted down to their normal place by typing FP.

After the BASIC program has been run, the machine code can be saved by the command BSAVE VARIABLE.PASS, A$803,L$181. The program will stay active, below your BASIC program, until you power down or do an FP.

To reload the ML program the next time you power up, type BRUN VARIABLE.PASS either from the keyboard or as part of your HELLO program. The LAST line of the HELLO program should be PRINT CHR$(4);"BRUN VARIABLE.PASS".

The first couple of statements of the hex code are the machine language equivalent of POKE 104,10 : POKE 2560,0 : NEW. That means that you only have to adjust the memory the first time you enter in the code. If you forget to adjust the memory before running the demonstration BASIC program, you will receive the message SYNTAX ERROR in 34057, a non-existent line. Simply type NEW : POKE 104,10 : POKE 2560,0 : NEW, reload the program from disk and RUN again. If you didn't adjust LOMEM, then, when the BASIC program stored the machine language program, it did so all over itself, causing a gigantic mess.

There is a sneaky reason for starting the machine language program at $803 (2051) rather than at $800, the start of the empty memory area. Suppose that, for some reason or another, you need to enter FP to recover from your program doing something strange. Typing FP causes 0's to be written at locations $800-$802 to indicate that there is no longer a program in the memory. This misses the ML program since it starts at $803. Thus, a quick CALL 2051 and ABRACADABRA, the pointers shift and the program is back in business.

The details of the demonstration and machine language programs are given after the description of the new SYNTAX of the instructions and limitations of the new commands.

Syntax For The New Commands

& GOSUB NAME !NAME(0),A,B,.....!

The name of the subroutine must be predefined before the subroutine is called (e.g. NAME = 1000).

The first parameter after the exclamation mark *must* be an array; otherwise, a BAD SUBSCRIPT ERROR occurs. It is suggested that the name of this array be the same as the name of the subroutine; for ease of remembering rather than necessity. If more than ten parameters are to be passed by the routine, the array must be DIMensioned to the number of parameters. No check is performed to see if the array is large enough for all the parameters used.

The other parameters must be numerical, either real variables (A, B etc.) or elements of a real array (A(1), B(1) etc.). The arrays don't have to be predimensioned unless their length is greater than ten. Errors will occur on attempting to pass a string (TYPE MISMATCH) or an integer (SYNTAX). It should be noted that it is the *value* of the array element that is moved and *not* the array itself. This means that you can't pass over the whole array by passing over the first element of an array. (c.f. In FORTRAN, it is the address which is passed and not the value of the array element. So, the whole array can be accessed from FORTRAN subroutine if you know the first address. In Applesoft, memory is continually being repositioned. The address of any variable is therefore continually changing, making any address stored very quickly invalid.)

The parameters do not need to have been defined before calling the subroutine. The machine language program makes use of Applesoft routines which automatically allocate space in the memory for new arrays and variables.

& GET !NAME(0),P,Q,.....!

This should be the first statement of the subroutine. The subroutine can't be recursive (it can't call itself).

This command does not extend an existing Applesoft command as did the & GOSUB, & RETURN and & GOTO commands. Therefore I had to use a different command. I decided to use GET, Since to me, this new command goes and *gets* the parameter values. If you would prefer a different command, such as LOAD, then the modification to allow this is simple. To have a different command, POKE its token into location 2600 ($828) before BSAVEing the program. For example, POKE 2600 ,167 will change this command to be & RECALL !......! rather than & GET !......!. (See page 121 of the *Applesoft Manual* for a list of the tokens).

The first parameter after the exclamation mark must be the same array used in the & GOSUB statement, otherwise unexpected values will be put into the parameters (P etc).

The other parameters must be real, otherwise a TYPE MISMATCH or SYNTAX ERROR will result. Either real variables (P) or elements of real arrays (P(1)) may be used. Again, the parameters don't have to be predefined before the subroutine call, unless they are arrays of length greater than ten. If the arrays need to be DIMensioned remember to do it *outside* the subroutine. Otherwise a REDIMENSIONED ARRAY ERROR will result on the second subroutine call.

The number of parameters in the & GET statement should be the same as the number of parameters in the & GOSUB statement. If this condition is not met, strange values could arrive in the parameters of the & GET statement.

& RETURN !NAME(0)!

The array used in the & RETURN statement should be the same array as used in the & GOSUB and & GET statements. As this array is used to temporarily store text pointers to the & GOSUB and & GET statements, strange results could result if the wrong array is used. However, it is probable that, instead of funny results, a SYNTAX ERROR will occur. The likelihood of the wrong array pointing to valid names in separate locations in memory is very small.

If the number of parameters in the & GET statement is not the same as the number of parameters in the & GOSUB statement, unpredictable values will be put into the parameters.

& GOTO NAME and & GOSUB NAME

The name of the subroutine must be established before it is called. If these commands are used, a normal RETURN is all that is needed. If & GET and & RETURN are used, a SYNTAX ERROR will occur.

& ON X GOSUB FNAME, SNAME and & ON X GOTO FNAME, SNAME

These ON X.... commands are supported, provided that no parameters are passed. That means that & ON X GOSUB FNAME, SNAME is permitted but & ON X GOSUB FNAME !FNAME(0),A,B,C,D!, SNAME !SNAME(0),A,B,C,D! is not. I felt that passing parameters in ON X... statements made the statements very unwieldy. The original idea behind introducing these new commands was to make the programs more readable rather than less. Multiline IF...THEN commands would do the same job, in a more readable fashion. For those people interested in implementing the unweildy ON... version, I have included the additional code needed (lines 176-189).

Warning

Warning on renumbering and crunching programs: Renumbering programs will not change the values of variables. Therefore, they will not change the pointers to the subroutines called by these new commands. This must be done by hand after the renumbering is complete. Utilities that crunch programs will not recognize the fact that the subroutines are being called and therefore will remove them as dead code. To overcome this removal problem, a dummy line that calls all subroutines, must be added to the program. After crunching, delete the dummy line. For example:

```
10 NAME=1000 : FIRST=2000
   (define the subroutines)
20 IF X=0 THEN GOSUB 1000 : GOSUB 2000 :
   GOTO 20 (dummy line to be removed after
   crunching)
```

Note that the dummy line is an IF..THEN statement that loops to itself. This means that a CRUNCHER, such as the one in DAKIN 5 PROGRAMMING AIDS 3.3, will leave that line alone, making it easy to remove.

BASIC Program Description

Line 180 – Establishes the machine language program.

Line 200 – Establishes the name of the subroutines to be called.

Line 220 – Demonstrates the command & GOSUB without passing any variables.

Line 250 – A loop is used to show that the stack is not corrupted by using these new commands. An OUT OF MEMORY ERROR will occur for 25 GOSUB calls without a proper return.

Line 260-280 – Establish random numbers for use in the variables.

Lines 290-320 – Demonstrates the & GOSUB command using both simple variables and arrays elements. The example subroutine adds together the first two numbers passed to it. The result is passed back in the third parameter.

Line 360 – Demonstrates that the subroutine call operated and that parameters were passed both ways.

Line 370 – Delay loop.

Line 1000 – Subroutine called without passing variables.

Line 2000 – New subroutine showing that variables were passed and used within the subroutine.

Line 5000-5070 – Machine language loading subroutine. It first checks that the DATA statements have been typed in correctly. Each DATA statement is the value of 16 locations plus the sum of the previous 16 locations used as a simple checksum. A typo error is indicated if the checksum is not the sum of the previous 16 locations.

Line 5080-5120 – Checks that POKEs have been performed.

Line 5130-5140 – POKEs the routine into memory.

Line 5150 – This establishes the AMPERSAND vector (&) pointers. This call is not necessary if the machine code is BRUN, but is necessary if the subroutine is BLOADed. Note that the CALL from BASIC is not the start of the ML program. If we did CALL the start of the program, an automatic NEW would occur, wiping out the demonstration program.

Machine Code Description

Briefly, the machine language program works as follows:

& GOSUB NAME!NAME(0),A,B...! The text pointers to the variable A are stored in the first two bytes of NAME(0). Then the value of A is moved into NAME(1), B into NAME(2) and so on.

& GET !NAME(0),W,X,...! The text pointers to the variable W are stored in the second two bytes of NAME(0). The value of NAME(1) is moved into W, NAME(2) into X and so on.

& RETURN !NAME(0)! The text pointer to W are recovered. The current values of W, X .. are moved into NAME(1), NAME(2) etc. Then the text pointer to A is recovered. The values in NAME(1), NAME(2) ... are moved into A, B....

The method of implementing the other commands is described in **COMPUTE!** #12.

Lines 15-31 – Zero page usage.

Lines 33-43 – Definition of tokens.

Lines 45-61 – Pointers to Applesoft routines. Internal Applesoft routines are used to cut down the amount of code required.

ADJMEM and AMPER. **Lines 65-77** – Do the machine language equivalent of POKE 104,0 : POKE 2560,0 : NEW. Then set the AMPERSAND vector.

ENTRY. **Lines 80-92** – Check on which of the new commands is required.

GOTO. **Lines 94-99** – Front end of the normal Applesoft GOTO routine moved and modified to allow variables and numbers to be used in the GOTO statement.

GOSUB. **Lines 101-134** – Handling of the & GOSUB command.

Line 101 – Front end of the normal Applesoft GOSUB routine moved and modified to allow variables and numbers in subroutine calls.

Line 121 – Is the first parameter an array? This array is used for the storage of the text pointers and the parameters. The stack would get too full if it were used.

Line 130: – Store text pointers.

Line 132 – Move the other parameters into the array for storage.

GET. **Lines 136-140** – Handling of the & GET command.

Line 136 – Locate the storage array.

Line 137 – Store the text pointers.

Line 139 – Move values stored in the array into the new parameters.

ARRay-GET. **Lines 142-149** – Gets and stores the location of the storage array after checking the leading exclamation mark.

ON. **Lines 151-189** – Handling of the ON X... command.

Line 152 – Get the value of X.

Line 154 – Determine if ON..GOTO or ON.. GOSUB.

Line 163 – Decrement X until find the subroutine requested.

Line 167 – Step over the values not being used.

Line 172 – Return to BASIC if subroutine not found.

Line 176-189 – Adding these instructions will allow the passing of parameters in ON X... commands.

RETURN. **Lines 191-213** – Handling of the & RETURN command.

Line 191 – Locate the storage array.

Line 198 – Store the current text pointers.

Line 199 – Recover the text pointers from the & GET statement.

Line 200 – Move the current values of the parameters in the & GET statement into storage.

Line 201 – Reset the storage array pointers.

Line 205 – Recover the text pointers from the & GOSUB statement.

Line 206 – Move the values in the storage array into the parameters used in the & GOSUB statement.

Line 207 – Recover the current text pointers and perform a normal RETURN.

CHecK-ARRay. **Lines 215-228** – Checks and adjusts the pointers to the storage array if new variables have been introduced during the commands & GOSUB and & GET.

Modifications to the next two subroutines, PARSTO and STOPAR will allow the passing of INTEGER parameters.

PARameters-to-STOrage. **Lines 230-243** – Moves the current values of the parameters in the & GOSUB and & GET commands into the storage array. Checks for integers and strings.

STOrage-to-PARameters. **Lines 244-257** – Moves the values in the storage array into the parameters in the & GOSUB and & GET commands.

STOre-TeXT-pointers. **Lines 259-264** – Stores the current text pointers into the zeroth element (NAME(0)) of the storage array. The Y register is preset.

GET-TeXT-pointers. **Lines 266-271** – Recovers the text pointers stored in the zeroth element of the storage array according to the value set in the Y register.

ADJust-PoinTers. **Lines 273-276** – Adjust the pointers to the storage array if they have shifted because a new variable has been made. Note that the pointers don't have to be adjusted if a new array has been made. All new arrays will be placed above the storage array in memory as the storage array is defined first.

COMmand-END. **Lines 278-283** – Looks for the final exclamation mark (!) of the command or other parameter. Pops the last subroutine address off of the stack allowing a quick return to BASIC if at the command's end.

References

"Applesoft Internal Entry Points" by Applesoft Computer Inc. in Apple Orchard March/April 1980, p. 12.

"Some Routines in Applesoft Basic" by J. Butterfield in COMPUTE!, September/October 1980, p. 68.

"Resolving Applesoft and Hires Graphics Memory Conflicts" by J. Schmoyer in COMPUTE!, April 1981, p. 76.

"Using Named GOSUB and GOTO Statements in Applesoft BASIC" by M. Smith in COMPUTE!, May 1981, p. 64.

```
100  ****************************
110  REM * MIKE SMITH              *
120  REM * 304, 86TH AVENUE SE     *
130  REM * CALGARY, ALBERTA        *
140  REM * CANADA  T2H 1N7         *
150  ****************************
160  REM
170  REM SET UP THE MACHINE CODE
180  GOSUB 5000
190  REM   SET UP THE SUBROUTINE NAM
     ES
200  DEMO =1000:ADDIT = 2000
210  REM  DEMONSTRATE NAMED GOSUB AN
     D GOTO
220  &  GOSUB DEMO:JUMP = 240: &  GO
     TO JUMP
230  REM DEMONSTRATE STACK OKAY
240  PRINT "HERE BY NAMED GOTO": PRI
     NT
250  FOR J = 1 TO 25
260  REM MAKE UP NUMBERS
270  K = INT (10 * RND (1)): = INT (
     10 *  RND (1))
280  P = INT ( 10 * RND (1)):Q(1) = ~
     INT (10 * RND (1))
290  REM
300  &  GOSUB ADDIT!ADDIT(0),K,L,M!
310  REM   DEMONSTRATE PASSING OF AR
     RAY ELEMENT
320  &  GOSUB ADDIT!ADDIT(0),P,Q(1),
     R!
330  REM
340  REM PRINT AND SHOW THAT HAVE US
     ED SUBROUTINE
350  REM
360  PRINT K;" + ";L;" = ";M: PRINT ~
     P;" + ";Q(1);" = ";R: PRIN
     T
370  FOR Z = 1 TO 500: NEXT Z
380  NEXT J: STOP
970  REM
980  REM  DEMONSTRATION SUBROUTINE
990  REM
1000  PRINT : PRINT "HERE BY THE GOSU
     B CALLED DEMO"
1010  PRINT : RETURN
1960  REM
1970  REM  SUBROUTINE ADDIT
1980  REM
1990  REM  DEMONSTRATE PASSING BACK O
     F ARRAY ELEMENT
2000  &  GET !ADDIT(0),T,U,V(4)!
2010  V(4) = T + U
2020  &  RETURN !ADDIT(0)!
4970  REM
4980  REM  MACHINE CODE ESTABLISHED
4990  REM
5000  BOT = 8 * 256 + 3:HIGH = 9 * 25
     6 + 10 * 16 + 2
5010  REM FLAG FOR CHECKSUM
5020  OK = 1:LINE = 6000
5030  FOR J = BOT TO HIGH STEP 16
5040  CHECK = 0: FOR K = J TO J + 15:
      READ IT:CHECK = CHECK + I
     T: NEXT K
5050  READ NUM: IF NUM < > CHECK THEN
      PRINT "TYPO IN LINE "LINE
     ::OK = 0
5060  LINE = LINE + 10: NEXT J
5070  IF OK = 0 THEN STOP
5080  PRINT : INPUT "DID YOU REMEMBE
     R THE POKES? ";A$
5090  IF LEFT$ (A$,1) = "Y" THEN 5130
5100  PRINT : PRINT "SAVE THIS PROGRA
     M AND THEN"
5110  PRINT : INVERSE : PRINT "NEW:PO
     KE104,10:POKE2560,0:NEW": ~
     NORMAL : PRINT
5120  PRINT "THEN RELOAD AND RUN.": S
     TOP
5130  RESTORE : FOR J = BOT TO HIGH S
```

```
     TEP 16
5140 FOR K = J TO J + 15: READ IT: P
     OKE (K),IT: NEXT K: READ I
     T: NEXT J
5150 PRINT : PRINT "BLOAD OKAY": CAL
     L BOT + 12: RETURN
5970 REM
5980 REM  MACHINE CODE DATA
5990 REM
6000 DATA 169,10,133,104,169,0,10,32
     ,75,214,169,76,141,245,168
     8
6010 DATA 3,169,31,141,246,3,169,8,1
     41,247,3,96,201,171,240,25
     ,1894
6020 DATA 201,176,240,36,201,190,240
     ,106,201,180,208,3,76,181,
     8,201,2448
6030 DATA 177,208,3,76,230,8,76,201,
     222,32,66,8,76,65,217,32,1
     697
6040 DATA 177,0,32,123,221,76,82,231
     ,169,3,32,214,211,165,185,
     72,1993
6050 DATA 165,184,72,165,118,72,165,
     117,72,169,176,72,32,66,8,
     32,1685
6060 DATA 183,0,201,0,240,38,201,58,
     240,34,201,44,240,30,32,16
     6,1908
6070 DATA 8,196,108,48,6,208,7,197,1
     07,16,3,76,150,225,32,249,
     1636
6080 DATA 234,32,106,221,160,0,32,11
     9,9,32,52,9,32,63,8,76,118
     5
6090 DATA 210,215,32,163,8,160,2,32,
     119,9,32,87,9,76,149,217,1
     520
6100 DATA 32,177,0,201,33,208,143,32
     ,177,0,32,227,223,133,0,13
     2,1750
6110 DATA 1,96,32,177,0,32,248,230,7
     2,201,176,240,13,201,171,2
     40,2130
6120 DATA 9,201,175,208,224,104,32,1
     77,0,72,198,161,208,4,104,
     76,1953
6130 DATA 31,8,32,177,0,32,227,223,3
     2,183,0,201,44,240,235,104
     ,1769
6140 DATA 104,104,96,32,163,8,141,16
     2,9,140,163,9,165,184,72,1
     65,1717
6150 DATA 185,72,160,2,32,129,9,32,5
     2,9,173,162,9,133,0,173,13
     32
6160 DATA 163,9,133,1,160,0,32,129,9
     ,32,87,9,104,133,185,104,1
     290
6170 DATA 133,184,32,177,0,76,107,21
     7,165,107,197,2,208,1,96,1
     33,1835
6180 DATA 2,169,7,208,2,169,5,24,101
     ,0,133,0,2,144,230,1,1197
6190 DATA 96,32,139,9,32,123,221,32,
     106,221,165,18,240,3,76,19
     8,1711
6200 DATA 8,32,27,9,166,0,164,1,32,4
     3,235,32,149,9,32,177,1116
6210 DATA 0,76,55,9,32,139,9,32,227,
     223,32,27,9,165,0,164,1199
6220 DATA 1,32,249,234,166,131,164,1
     32,32,43,235,32,149,9,32,1
     77,1818
6230 DATA 0,76,90,9,165,184,145,0,20
     0,165,185,145,0,96,177,0,1
     637
6240 DATA 133,184,200,177,0,133,185,
     96,165,107,133,2,32,40,9,7
     6,1672
6250 DATA 190,222,32,40,9,32,183,0,2
     01,33,208,2,104,104,96,0,1
     456
6260 DATA 104,104,96,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,304
```

©

THE ATARI® GAZETTE

INSIGHT: Atari

Bill Wilkinson
Optimized Systems Software
Cupertino, CA

This month marks the end of my series on Atari I/O. That certainly doesn't mean that we won't continue to discuss assembly language I/O of related topics; it simply means that I feel I have finished my formal presentation of the material. Again, I strongly urge you to purchase the *Atari Technical User's Notes* (available from Customer Service, 1340 Bordeaux Ave., Sunnyvale, CA 94086, for $30, including shipping). There is a lot of detail in those "notes," including much that I have glossed over. I hope that my presentation, though, has served as a usable introduction to the subject.

Also this month, I give you a method for creating relocatable assembly language programs (and a method to then load them). We use the loader to implement our "M:" driver from last month, completely via BASIC (thus making it usable for those of you not yet into assembly language...and it *is* usable).

Finally, we continue our discussion of how BASIC works. *De Re Atari*, and the serialized version thereof which appears in this month's *BYTE*, does a good job of discussing the *how* of BASIC's syntaxer; we will delve into the *why*.

Atari I/O, Part 4: GRAPHICS

Errata! Before we get started on this month's topic, I must report an error I made in **COMPUTE!** #18. On page 100, in Table 1, under the "Note" pertaining to ICBLL/ICBLH, I stated that the length is decremented by one for each byte transferred. Actually, Atari's OS is smarter than that: upon return from GET/PUT RECORD (text or binary) ICBLL/ICBLH contain a count of the number of bytes successfully transferred. This result is eminently usable (e.g., in copying records or even whole files), and perhaps we will have a program here soon that demonstrates its use.

On with the new: this whole series started as a result of a comment that I read which said something like "Atari graphics from assembly language are hard to do – you have to know about display lists, vertical blank interrupts, etc." Knowing how BASIC does graphics for its users I said, "Nonsense! It's easy! Someone should show *how* easy!" And Richard Mansfield, of **COMPUTE!**, said, "Gee, I wonder who we could get..." Ahem.

If what you are trying to do is write an improved version of Eastern Front or Pacman or some other such pioneering project, then you need to know everything ever published and then some. *But*, if what you want is simply a way to transfer what you have learned or written using BASIC into a reasonably simple set of assembly language routines, read on.

Remember, BASIC does *all* its graphics and I/O via Atari's OS. BASIC knows nothing of graphics modes, display lists, character sets, color registers, etc. (True, BASIC A+ does its own thing with Player/Missile Graphics, but that's only because Atari's OS doesn't know about PMG.) So, anything done with standard BASIC statements can be duplicated *easily* in assembly language. To demonstrate the truth of this, Figure 1 contains a list of the seven BASIC graphics statements together with a note on how each is accomplished.

Accompanying this article is a listing of my proposal for a set of standard routines to be used by assembly language programmers when interfacing to OS graphics. These routines duplicate, as far as practicable, the statements used to do BASIC graphics. The listing clearly calls out ENTRY and EXIT parameters for each routine (i.e., register usage), so study it carefully.

As a very simple example of the routines' usage, I offer a program fragment that is written in both BASIC and assembly language:

GRAPHICS 3	**LDA #3**
	JSR GRAPHICS
COLOR 3	**LDA #3**
	JSR COLOR
PLOT 10,10	**LDX #10**
	LDA #0
	LDY #10
	JSR PLOT
DRAWTO 25,15	**LDX #25**
	LDA #0
	LDY #15
	JSR DRAWTO
SETCOLOR 2,0,14	**LDX #2**
	LDA #0
	LDY #14
	JSR SETCOLOR

Before leaving this topic, some notes on the routines might be helpful: since the A-register will be zero upon entry to PLOT, DRAWTO, LOCATE, and POSITION for all graphics modes except GRAPHICS 8 (or 24), placing a LDA #0 in the beginning of POSITION would save code for anyone not using mode 8. Remember, Atari's "S:" driver can accomodate GRAPHICS 0 through 11 and 17 through 24. Adding 32 ($20) to any graphics mode (at the time of the call to GRAPHICS) will suppress the erasure of the screen. (I haven't figured out a use for this yet, but it's nice to know it's there.)

Obviously, one could save time (and sometimes space) by performing COLOR and SETCOLOR and POSITION via simple stores (e.g., STA), but there is a certain structuring and elegance that goes with the use of the routines. The graphics routines listed herein were assembled in the $600 page of memory, a much overworked location. I would hope that you would take the time to type them in to your assembler/editor and include them directly in future programs (EASMD users may .INCLUDE them indirectly). I really would appreciate hearing of your successes (or failures, if any) using these routines.

So far, no assembler available for the Atari produces relocatable, linkable object files (and, from what I have heard, neither will Atari's Macro Assembler). When we produced BASIC A+ and EASMD, we wanted them to move themselves to the top of memory, so we re-invented a scheme I have seen in several incarnations before: Assemble the program twice, setting the origin for any portion(s) to be relocated one page (256 bytes) higher for the second assembly, producing two object files. Write a program that compares the two objects and notes all locations that differ by one (differing by any other amount is an error). Produce a table (or bit map, or ...) of all these differences. At relocatable load time, read in the first object file (to where it is to be relocated) and use the table to change all the bytes which need to be relocated.

The system is a kludge, but a very effective one. It has a few limitations: you still don't have linkable object files, you must relocate in full page increments (i.e., multiples of 256 bytes), and you have to have some place safe to put the relocating loader. Are you willing to live with those limits? Then try this.

I present here three BASIC programs together with instructions for their use. The first program, MAKEREL (Program 1), seems to be to be perfectly adequate as is, written in BASIC. It's a little slow, but one only uses it when ready to create a new relocatable object file. The other two programs, LOADREL.A and LOADREL.B (Programs 2 and 3), could be advantageously rewritten in assembly language. They are presented here in BASIC because (1) this method fulfills the requirement for a "safe place" for the loader and (2) by presenting them in BASIC they can be used by those not yet ready to tackle assembly language and (3) it was easier for me.

The instructions below presume the use of the Atari Assembler/Editor or the OSS EASMD, but they can be easily adapted to most systems that produce Atari DOS-compatible object files.

How To Use The Relocator Programs

1) Write, assemble, and debug your code using some fixed address(es).

2) Ensure that your code is all in one piece (i.e., there is only one *=, at the beginning of the code segment).

3) Origin your code on an even page boundary (i.e., use ***=$hh00**, there 'hh' specifies any page from 02 through FE). Assemble the code into an object file on disk named "OBJECT1" (use **ASM ,,#D:OBJECT1**).

4) Change your origin to one page higher in memory (***=$nn00**, where 'nn' = 'hh'+1). Assemble the code to "OBJECT2" (**ASM ,,#D:OBJECT2**).

5) Run the MAKEREL program. It will produce the file "DATA.REL".

6) Adjust the value of the variable NUMBEROF-PAGES in both LOADREL.A and LOADREL.B (Programs 2 and 3) to reflect the number of 256-byte pages needed by your routine. **SAVE** the adjusted versions.

7) Anytime you want to load your routine, simply use **RUN "D:LOADREL.A"**.

Notes

A. Generally, it's a good idea to have your routine start execution at the origin (*=) point. Then you can invoke it from BASIC via **USR(PEEK(128) + 256 *(PEEK(129) - NUMBEROFPAGES))**

B. If you **RUN "D:LOADREL.A"** again without hitting RESET, it will load another copy above the first. Not too neat, *but* the advantages of being able to thus load several different modules should be obvious!

C. LOADREL.B performs an **ENTER "D: DATA.REL"**. Rather than waiting for the **ENTER** each time, you may **SAVE** the resultant program (after taking out the **ENTER** line) for a slightly faster load of a specific module.

Finally, we offer Program 4 which may be added to LOADREL.B to produce a relocatable load of last month's "M:" driver. (Again, be sure to delete the **ENTER** line from LOADREL.B.)

For once, I haven't forgotten you cassette

users. If you enter LOADREL.A (carefully, please!) and **CSAVE** it (or **SAVE"C:"**) on a blank tape you need only change the last line to read **RUN "C:"**. Then **NEW** and enter LOADREL.B, leaving out the **ENTER** line, but including the listing of Program 4. Use **SAVE"C:"** (do NOT use CSAVE...it won't work!) to place the resultant combination on the tape after LOADREL.A (and, of course, you could then follow on the same tape with a program of your own). You may now enjoy the "M:" driver via this tape by **CLOAD**ing and **RUN**ning the first program (or use **RUN"C:"** if you used **SAVE"C:"**, my own preference for all but the largest programs).

MAKEREL could also be adapted to cassette usage, though not without difficulty and/or a relatively large amount of memory. Obviously, these programs can be improved upon tremendously by simply adding, for example, flexibility of file name. But my intention was to present something as simple and straightforward as possible, in the hopes that everyone would find it readable and useful. Obviously, my techniques could be adapted to other machines (does the PET have a relocating assembler?), so adapt away (and be sure to send **COMPUTE!** the results to share with the rest of us). On to lighter subjects.

Inside Basic, Part 2: The Why Of Syntaxing

Last month I presented a program to print out the keywords of BASIC. If you took the time to enter and run that program, you saw some strange things in the printout of the operators. But there was a method to our madness, as you will see.

Let us examine the tokenized (internal) form of the following line:

1025 PRINT "HI THERE", THIS * (3 + IS(FUN)) : STOP

Assuming that we had just previously **NEW**ed, the tokenized form of that line is as follows (all numbers in decimal):

01 04 36 33 32 15 08 72 73 32 84 72 69 18 128 36 43 14 64 03 00 00 00 00 37 129 56 130 44 44 20 36 38 22

Now that isn't too terribly useful or readable, so let's examine the tokens one at a time:

01 04	This is the line number (4*256 + 1 = 1025) in standard 6502 form.
36	This is the line length, including the line number and this byte.
33	Statement length of the first statement. Actually, this is the displacement to the beginning of the next statement (from the beginning of the line).
32	The token for PRINT. Check the output of the keyword printing program from last month.
15	A special token that says a string constant follows.
08 72 73 32 74 72 69 82 69	The string constant consists of a byte that gives the length of the string followed by the characters of the string. Note that the quotes have disappeared.
18	The comma, tokenized.
128	Our first variable! Operator tokens over 127 are variables. The variable number (in the variable table) is 128 less than the token value. This variable is **THIS**.
36	The multiplication operator.
43	One variety of left parenthesis. This one is a normal or expression left parenthesis.
14	Another special token (actually, number 2

Figure 1.

BASIC Statement	Action performed
GRAPHICS g	If bit 4 ($10) of 'g' is on, this is the same as **OPEN #6, 12, g-16, "S:"** **If the bit is off, this is the same as OPEN #6, 16 + 12, g, "S:"** (Note: the fifth bit, $20, of 'g' should be copied into AUX1, the OPEN mode.)
COLOR c	Simply saves 'c' in a safe place.
POSITION h,v	Places 'h' in locations $55 and $56 (LSB,MSB) Places 'v' in location $54
PLOT h,v	Performs a **POSITION h,v** and then Performs a **PUT #6,c** (where 'c' is the color saved by **COLOR**)
LOCATE h,v,c	Performs a **POSITION h,v** and then Performs a **GET #6,c**
DRAWTO h,v	Performs a **POSITION h,v** and then Does a **POKE 763, c** ('c' is the COLOR saved, as above) and then Performs an **XIO 17, #6, 12, 0, "S:"**
SETCOLOR r,h,lu	Is equivalent to **POKE 708 + r, h*h16 + lu**

Note: FILL may be performed from assembly language by following exactly the same sequence specified in the *Basic Reference Manual,* using XIO 18, etc.

Program 1: MAKEREL

```
100 REM *** OPEN ALL 3 FILES ***
110 OPEN #1,4,0,"D:OBJECT1"
120 OPEN #2,4,0,"D:OBJECT2"
130 OPEN #3,8,0,"D:DATA.REL"
150 REM *** INITIALIZE VARIABLES ***
160 LINE=10000
```

	of 2), says a numeric constant follows.
64 03 00 00 00 00	The constant, in Atari BASIC internal floating point form. This is unique, as we shall see soon.
37	An addition operator.
129	The variable **IS** (already known to be an array, though it has not yet been DIMensioned).
56	Another left parenthesis. This one is called an "array left paren" in the BASIC source listing. We will later see why it is distinct.
130	Our last variable, **FUN**.
44 44	Two right parentheses. Strange, they are both the same.
20	Our End-Of-Statement token, otherwise known as a colon.
36	The statement end displacement for the second statement on this line.
38	The token for STOP. Again, refer to the keyword listing program.
22	An End-Of-Line token, otherwise known as a RETURN.

Wasn't that fun? For a masochist? Hopefully, you are asking questions that begin with "Why."

Why tokenize at all? For compactness: in our example we saved six bytes over a straight source line. For speed: it is much faster (at run-time) to discover that, for example, 32 means "PRINT" than it would be if we had to examine the letters "P", "R", "I", "N", "T" for a keyword match. Because tokenizing is almost an automatic byproduct of syntaxing.

Why syntax-check at entry? Because it is embarrasing to give a program to someone, have them run it, and get a SYNTAX ERROR message at line 23776 (the line that handles disk full conditions, which we never got to when we were testing). Because it makes program entry so much easier for be-

```
170 DCNT=0
200 REM *** STRIP HEADER ($FFFF) WORD ***
220 GET #1,FF:GET #1,FF
230 REM STRIP HEADER AND ADDRESSES FROM FILE2
240 GET #2,FF:GET #2,FF:REM HEADER
250 GET #2,FF:GET #2,FF:REM START ADDRESS
260 GET #2,FF:GET #2,FF:REM END ADDRESS
300 REM *** PROCESS ADDRESSES ***
310 GET #1,LOW:GET #1,FIRSTHIGH:FIRST=LOW+256*FIRSTHIGH
320 GET #1,LOW:GET #1,HIGH:LAST=LOW+256*HIGH
400 REM *** READY TO PRODUCE OUTPUT ***
410 FOR ADDR=FIRST TO LAST
420   IF DCNT=0 THEN PRINT #3;LINE;" DATA ";:LINE=LINE+10
430   GET #1,B1:GET #2,B2
440   IF B1=B2 THEN 480
450   IF B2<>B1+1 THEN PRINT "BAD RELOCATION":STOP
460   B1=B1-FIRSTHIGH:REM THE RELOCATION FACTOR
470   PRINT #3;"*";:REM AND FLAG THIS BYTE
480   PRINT #3;B1;
490   DCNT=DCNT+1
500   IF DCNT<=9 THEN PRINT #3;",";
510   IF DCNT>9 THEN DCNT=0:PRINT #3
520   NEXT ADDR
530 REM *** CLEAN UP ***
540 IF DCNT=0 THEN PRINT #3;LINE;" DATA ";
550 PRINT #3;"="
560 PRINT #3;"GOTO 500"
580 CLOSE #1:CLOSE #2:CLOSE #3
590 END
```

Program 2: LOADREL.A

```
10 REM *** THIS IS LOADREL.A ***
20 REM (THIS SIMPLY SETS UP MEMORY FOR LOADREL.B)
30 NUMBEROFPAGES=1:REM CHANGE THIS AS NEEDED
40 SIZE=256*NUMBEROFPAGES
100 REM *** SEE COMPUTE! #19 ***
110 LET LOMEM=743:MEMLOW=128
120 LADDR=PEEK(LOMEM):HADDR=PEEK(LOMEM+1)
129 REM -- LINE 130 ENSURES THAT 1K BYTES STARTS ON PAGE BOUNDARY --
130 IF LADDR<>0 THEN LADDR=0:HADDR=HADDR+1
140 ADDR=LADDR+256*HADDR
150 ADDR=ADDR+SIZE
160 HADDR=INT(ADDR/256):LADDR=ADDR-256*HADDR
170 POKE LOMEM,LADDR:POKE LOMEM+1,HADDR
180 POKE MEMLOW,LADDR:POKE MEMLOW+1,HADDR:RUN "D:LOADREL.B"
```

Program 3: LOADREL.B

```
100 REM *** THIS IS LOADREL.B ***
110 REM
120 REM THIS PROGRAM DOES THE ACTUAL RELOCATABLE LOAD
130 REM
140 DIM TEMP$(10)
150 NUMBEROFPAGES=1:REM ADJUST TO SAME AS LOADREL.A
200 REM AGAIN, SEE COMPUTE! #19
210 LET LOMEM=743:MEMLOW=128
220 POKE LOMEM,PEEK(MEMLOW):POKE LOMEM+1,PEEK(MEMLOW+1)
300 REM RPAGE IS THE MEMORY PAGE WHERE WE RELOCATE TO
310 RPAGE=PEEK(MEMLOW+1)-NUMBEROFPAGES
330 REM OBVIOUSLY, THIS VALUE SHOULD MATCH THE MEMORY
340 REM RESERVED IN 'LOADREL1.SAV'
350 ADDR=RPAGE*256:REM STARTING ADDR OF LOAD
```

ginners, particularly kids. Because I like it.

Why one-byte variable numbers? Again, for speed and compactness. Use variable names as long as you like: only the first usage eats up any more memory than a single-character, undecipherable variable name. There are disadvantages: a maximum of 128 different variables, a mispelled variable name can't be purged from the variable table without LISTing and reENTERing. On the whole, a very wise choice (*I* can say that, it's one part of Atari BASIC I *didn't* design into the specs).

Why internalized numeric constants? For speed. Period. Well, maybe for simplicity at run-time, but that's only a maybe. Did you know that numeric constants in Atari BASIC actually execute faster than variables? Write a timing loop and prove it to yourself.

Why line length bytes? Do you need them if you have statement length bytes? We don't *need* them, but they make line skipping (as when we are executing a GOTO) faster than it would be if we had to skip individual statements.

Why statement length bytes? Given that you have line length bytes? This one is harder to answer, because it has to do with how we execute GOSUB/RETURN, etc. I will leave that for a later article, but I will note that these bytes were extremely helpful when it came to implementing the **IF...ELSE...ENDIF** structure in BASIC A+.

Why decimal floating point? Because it is easier for beginners to understand (try **PRINT 123.123-123** using Applesoft) and is obviously preferable for money applications. Actually, our decimal add and subtract are faster than the corresponding binary routines. Admittedly, multiply suffers a little and divide suffers a lot.

Why different kinds of left parentheses? *Why* several kinds of equal sign? Because it's easy for the syntaxer to see the different

```
400 REM *****************************
410 REM *  GET THE RELOCATION DATA *
420 REM *****************************
450 ENTER "D:DATA.REL"
500 REM *** THE ENTER BRINGS US HERE ***
510 READ TEMP$
520 IF TEMP$(1,1)="=" THEN END
530 IF TEMP$(1,1)<>"*" THEN POKE ADDR,VAL(TEMP$):GOTO 550
540 POKE ADDR,VAL(TEMP$(2))+RPAGE:REM RELOCATION
550 ADDR=ADDR+1:GOTO 510
```

Program 4: DATA.REL

```
520 IF TEMP$(1,1)="=" THEN 1000
1000 REM LINE 1010 IS USED TO INITIALIZE THE M: DRIVER
1010 JUNK=USR(RPAGE*256+48)
1020 END
10000 DATA 162,0,189,26,3,240,10,201,77,240
10010 DATA 26,232,232,232,208,242,96,169,77,157
10020 DATA 26,3,169,59,157,27,3,169,*0,157
10030 DATA 28,3,169,0,157,29,3,169,0,141
10040 DATA 231,2,169,*1,141,232,2,96,104,240
10050 DATA 205,168,104,104,136,208,251,240,197,76
10060 DATA *0,111,*0,146,*0,133,*0,159,*0,73
10070 DATA *0,76,74,*0,160,1,96,189,74,3
10080 DATA 41,8,240,13,173,229,2,141,210,*0
10090 DATA 172,230,2,136,140,211,*0,173,210,*0
10100 DATA 141,206,*0,173,211,*0,141,207,*0,160
10110 DATA 1,96,189,74,3,41,8,240,12,173
10120 DATA 206,*0,141,208,*0,173,207,*0,141,209
10130 DATA *0,160,1,96,72,32,181,*0,104,160
10140 DATA 0,145,224,32,192,*0,96,32,160,*0
10150 DATA 176,7,160,0,177,224,32,192,*0,96
10160 DATA 32,181,*0,205,208,*0,208,9,204,209
10170 DATA *0,208,4,160,136,56,96,160,1,24
10180 DATA 96,173,206,*0,133,224,172,207,*0,132
10190 DATA 225,96,172,206,*0,208,3,206,207,*0
10200 DATA 206,206,*0,160,1,96,0,0,0,0
10210 DATA 0,0,=
```

Program 5: Graphics Routines, Equates

```
0000        1010          .PAGE "Equates, etc."
            1020 ;
            1030 ; CIO EQUATES
            1040 ;
E456        1050 CIO    =      $E456      ; Call OS thru here
0342        1060 ICCOM  =      $342       ; COMmand to CIO in IoCb
0344        1070 ICBADR =      $344       ; Buffer or filename ADdRess
0348        1080 ICBLEN =      $348       ; Buffer LENgth
034A        1090 ICAUX1 =      $34A       ; AUXilliary byte # 1
034B        1100 ICAUX2 =      $34B       ; AUXilliary byte # 2
            1110 ;
0003        1120 COPN   =      3          ; Command OPeN
000C        1130 CCLOSE =      12         ; Command CLOSE
0007        1140 CGBINR =      7          ; Command Get BINary Record
000B        1150 CPBINR =      11         ; Command Put BINary Record
0011        1160 CDRAW  =      17         ; Command DRAWto
0012        1170 CFILL  =      18         ; Command FILL (not used in this demo)
            1180 ;
0004        1190 OPIN   =      4          ; OPen for INput
0008        1200 OPOUT  =      8          ; OPen for OUTput
            1210 ;
            1220 ;
            1230 ; EQUATES used by the S: driver and
```

kinds of equal signs in, for example, **LET A = B = C + D$ = E$**. Sure, we could tell the difference at run time from context, but why should we when it's so easy to distinguish between a 45 and a 34 and a 52?

Why doesn't Atari BASIC have string arrays? I really didn't want to put this question in, but I wanted to save myself the letters and threatening phone calls. The best reason is that it was a choice of string arrays or syntax checking. (Obviously, I like the choice.) Other rationales include the fact that Atari was aiming for the educational market, where the HP2000 (with 72-character, Atari-style strings) was the *de facto* standard.

My personal favorite reasons are twofold: (1) anything you can do with string arrays you can also do with long strings (admittedly, sometimes with a little more difficulty) though the reverse is definitely not true; and (2) string arrays are unique to DEC/ Microsoft/??? BASIC and do not appear in that form in any other of the more popular languages (e.g., FORTRAN, COBOL, PASCAL, C, FORTH, etc.). Techniques learned with long strings are portable to these other languages: techniques involving string arrays are, at best, difficult to transfer. Finally, long strings as implemented on the Atari have some unique advantages not immediately obvious. I hope to explore some of these advantages in future columns.

```
               1240 ;   the VBLANK routines
               1250 ;
0055           1260 HORIZONTAL = $55
0054           1270 VERTICAL =   $54
02FB           1280 DRAWCOLOR =  $2FB
02C4           1290 COLOR0 =     $2C4
               1300 ;
               1310 ; miscellany
               1320 ;
00FF           1330 LOW     =     $FF
0100           1340 HIGH    =     $100
               1350 ;
Graphics routines for COMPUTE! #21
The actual routines
0000           1360            .PAGE "The actual routines"
               1370 ;
               1380 ; First, set the location and some miscellaneous
               1390 ;     RAM usage
               1400 ;
0000           1410            *=    $660
               1420 ;
0660 00        1430 SAVECOLOR .BYTE 0       ; where COLOR is saved
               1440 ;
0661 53        1450 SNAME   .BYTE "S:",0    ; the filename for open
0662 3A
0663 00
               1460 ;
               1470 ;
               1480 ; GRAPHICS g
               1490 ;
               1500 ; ENTRY: A-reg contains graphics mode 'g'
               1510 ;  EXIT: Y-reg has completion status
               1520 ;
               1530 GRAPHICS
0664 48        1540            PHA             ; save 'g'
0665 A260      1550            LDX  #6*$10     ; file 6
0667 A90C      1560            LDA  #CCLOSE
0669 9D4203    1570            STA  ICCOM,X
066C 2056E4    1580            JSR  CIO        ; First, we must close file #6
               1590 ; (we ignore any errors from the close)
               1600 ;
066F A260      1610            LDX  #6*$10     ; again, file 6
0671 A903      1620            LDA  #COPN      ; we will open this 'file'
0673 9D4203    1630            STA  ICCOM,X
0676 A961      1640            LDA  #SNAME&LOW
0678 9D4403    1650            STA  ICBADR,X   ; we use the file name "S:"
067B A906      1660            LDA  #SNAME/HIGH
067D 9D4503    1670            STA  ICBADR+1,X ; by pointing to it
               1680 ;
               1690 ; all is set up for OPEN, now
               1700 ; we tell CIO (and S:) what kind of open
               1710 ;
0680 68        1720            PLA             ; our saved 'g' graphics mode
0681 9D4B03    1730            STA  ICAUX2,X   ; is given to S:
               1740 ; (note that S: ignores the upper bits of AUX2)
0684 29F0      1750            AND  #$F0       ; now we get just the upper bits
0686 4910      1760            EOR  #$10       ; and flip bit 4
               1770 ; (Read the text.  S: expects this bit inverted
               1780 ;     from what normal BASIC usage is.)
0688 090C      1790            ORA  #$0C       ; allow read and write access (for CIO)
068A 9D4A03    1800            STA  ICAUX1,X   ; make CIO and S: happy
068D 2056E4    1810            JSR  CIO        ; and do the OPEN of S:
```

```
0690 60               RTS
             ;
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;
             ; COLOR c
             ;
             ;  ENTER: Color 'c' in A-register
             ;   EXIT: Unchanged
             ;
             COLOR
0691 8D6006           STA  SAVECOLOR
0694 60               RTS             ; exciting, wasn't it?
             ;
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;
             ; POSITION h,v
             ;
             ;  ENTER: h (horizontal) position in X,A
             ;               registers (LSB,MSB)
             ;         v (vertical) position in Y-register
             ;
             ;   EXIT: unchanged
             ;
             POSITION
0695 8655             STX  HORIZONTAL
0697 8556             STA  HORIZONTAL+1 ; read the text
0699 8454             STY  VERTICAL  ; too simple, right?
069B 60               RTS
             ;
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;
             ; PLOT h,v
             ;
             ;  ENTER: must have done a previous COLOR call
             ;         X,A,and Y registers set as in POSITION
             ;
             ;   EXIT: Y-register has completion status
             ;
             PLOT
069C 209506           JSR  POSITION
069F A260             LDX  #6*$10     ; file 6, again
06A1 A90B             LDA  #CPBINR    ; Command Put BINary Record
06A3 9D4203           STA  ICCOM,X
06A6 A900             LDA  #0
06A8 9D4803           STA  ICBLEN,X
06AB 9D4903           STA  ICBLEN+1,X ; if buffer length is zero...
06AE AD6006           LDA  SAVECOLOR ; then CPBINR puts one char from A-reg
06B1 2056E4           JSR  CIO        ; and this is how we PLOT
06B4 60               RTS
             ;
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;
             ; LOCATE h,v,c
             ;
             ;  ENTER: X,A,and Y registers set up as in POSITION
             ;   EXIT: A-register has the LOCATEd color
             ;         Y-register has the completion code
             ;
             LOCATE
06B5 209506           JSR  POSITION
06B8 A260             LDX  #6*$10     ; file 6
06BA A907             LDA  #CGBINR    ; Command Get BINary Record
06BC 9D4203           STA  ICCOM,X
```

```
06BF A900   2450        LDA  #0
06C1 9D4803 2460        STA  ICBLEN,X
06C4 9D4903 2470        STA  ICBLEN+1,X ; if Buffer LENgth is zero,
06C7 2056E4 2480        JSR  CIO        ; then the character is returned in A
06CA 60     2490        RTS
            2500 ;
            2510 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            2520 ;
            2530 ; DRAWTO h,v
            2540 ;
            2550 ;  ENTER: must have done a previous PLOT
            2560 ;         X.A,and Y registers as in POSITION
            2570 ;
            2580 ;   EXIT: Y-register has completion code
            2590 ;
            2600 DRAWTO
06CB 209506 2610        JSR  POSITION
06CE AD6006 2620        LDA  SAVECOLOR
06D1 8DFB02 2630        STA  DRAWCOLOR ; where DRAWTO expects its color
06D4 A260   2640        LDX  #6*$10    ; file 6...once more
06D6 A911   2650        LDA  #CDRAW    ; just a command to "S:"
06D8 9D4203 2660        STA  ICCOM,X
06DB A90C   2670        LDA  #$0C
06DD 9D4A03 2680        STA  ICAUX1,X  ; insurance
06E0 A900   2690        LDA  #0
06E2 9D4B03 2700        STA  ICAUX2,X  ; ...guaranteed to work
06E5 2056E4 2710        JSR  CIO       ; do the actual DRAWTO
06E8 60     2720        RTS
            2730 ;
            2740 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            2750 ;
            2760 ; SETCOLOR r,hue,lum
            2770 ;
            2780 ;  ENTER: X-register has color register 'r'
            2790 ;         A-register has hue
            2800 ;         Y-register has luminance
            2810 ;   EXIT: (undefined)
            2820 ;
            2830 SETCOLOR
06E9 0A     2840        ASL  A
06EA 0A     2850        ASL  A
06EB 0A     2860        ASL  A
06EC 0A     2870        ASL  A         ; we need hue * 16
06ED 9DC402 2880        STA  COLOR0,X  ; save it here for a nonce
06F0 98     2890        TYA
06F1 290E   2900        AND  #$0E      ; only luminance bits that matter
06F3 18     2910        CLC
06F4 7DC402 2920        ADC  COLOR0,X  ; end of the nonce
06F7 9DC402 2930        STA  COLOR0,X  ; and VBLANK will move this to hardware
06FA 60     2940        RTS
            2950 ;
06FB        2960        .END

Graphics routines for COMPUTE! #21
The actual routines

=E456 CIO          =0342 ICCOM       =0344 ICBADR          =0348 ICBLEN
=034A ICAUX1       =034B ICAUX2      =0003 COPN            =000C CCLOSE
=0007 CGBINR       =000B CPBINR      =0011 CDRAW           =0012 CFILL
=0004 OPIN         =0008 OPOUT       =0055 HORIZONTAL      =0054 VERTICAL
=02FB DRAWCOLOR    =02C4 COLOR0      =00FF LOW             =0100 HIGH
 0660 SAVECOLOR     0661 SNAME        0664 GRAPHICS         0691 COLOR
 0695 POSITION      069C PLOT         06B5 LOCATE           06CB DRAWTO
 06E9 SETCOLOR
```

©

P/M Graphics Made Easy

T. Sak, S. Meier
Baltimore, MD

Many people have called the Atari's graphics capabilities its best feature, especially the player-missile graphics. We won't argue, but how many of you have backed away because it looks too difficult to handle in BASIC or you simply are not satisified with the execution speeds which you are able to achieve?

Well, no more excuses! We've got a machine language subroutine that you can use with BASIC to achieve exciting graphics performance without a lot of muss and fuss. As a matter of fact, you make only one setup call to the subroutine and then forget it! And we promise you need know nothing about machine language. Just a few POKEs and you'll have your players dancing around the television screen.

You Don't Need To Know Machine Language

There have been a number of very helpful articles published describing the essential player-missile graphic information. Chris Crawford's description in **COMPUTE!** #8 is particularly noteworthy. We're going to assume that you are familiar with the fundamentals, but we'll review highlights as they're required.

A feature of the Atari with which you may not be familiar is its "interrupt" mechanism and how you can let it move your players for you at machine language speed — without the overhead of calling it from your BASIC program. Before we explore this useful feature, let's take a quick refresher course on interrupts.

As you know, the Atari keeps itself pretty busy doing its "housekeeping" chores even while it is interpreting your BASIC program. Among other things, the Atari must maintain the steady delivery of information to your television set, allowing it to paint a constantly up-to-date picture of the display data. Multiple, concurrent activities are performed by allowing one particular activity to periodically interrupt another.

The traditional analogy is that of a busy business executive who, while engaged in a meeting with an associate, is interrupted by a telephone call. The ringing phone signals the interrupt; the executive "checkpoints" his meeting and answers the phone. After disposing of the call, the executive resumes his meeting at the point of interruption.

A similar circumstance occurs each time a complete picture is painted by your television set. The television's electron beam paints the picture by sweeping horizontal rows across the picture tube beginning in the upper left hand corner and ending in the lower right. The beam is turned off when it reaches the lower right corner and is returned to its upper left starting position. This return trip is essentially a vertical positioning movement so this period when the beam is turned off is known as the *vertical blank time.*

Move During Vertical Blanks

The onset of the vertical blank cycle serves as an opportunity for the Atari's antic chip to signal an interrupt, the vertical blank or VBLANK interrupt. The operating system uses this occasion to perform some of its "housekeeping" duties. Fortunately, the operating system designers allow us to include a machine language subroutine which can be executed as one of these tasks.

The machine language vertical blank interrupt player movement subroutine described here is called VBLANK PM and it allows you to simply POKE the next x and y coordinate at which your player is to be displayed. There is no need to repeatedly call the subroutine from BASIC via the USR function. The subroutine will be automatically executed during the next vertical blank period. It is possible to move the players every time a new screen is painted on the television – and that's 60 times a second!

You may recall from other articles that an appropriate POKE to location 53248 (and the three memory locations following) permits you to position players zero through three horizontally along the x-axis. It's not quite as easy to position the players vertically along the y-axis. Not until now!

The VBLANK PM subroutine takes care to move the players in both directions. Movements along the vertical axis involve "erasing" and rewriting the player in the new position. VBLANK PM does this for you, automatically. There are a few things which you must do for VBLANK PM however.

First, you must get the VBLANK PM machine language subroutine into memory and notify the operating system that is to be included as one of the "housekeeping" tasks to be performed as a part of servicing the vertical blank interrupt. Next, it's up to you to draw your players and tell VBLANK PM how tall they are. After initialization, VBLANK PM looks after the positioning of your players until either a warm start (pressing SYSTEM RESET) or a cold start (power-off, power-on sequence) is performed.

Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky Tutorials

ATARI (tm) GRAPHICS AND SOUND MADE EASY!

SANTA CRUZ EDUCATIONAL SOFTWARE HAS WRITTEN A SERIES OF AFFORDABLE PROGRAMS THAT DEMONSTRATES MANY OF THE SPECIAL "TRICKS" THAT ONLY THE ATARI 400/800 COMPUTER IS CAPABLE OF DOING. WE OFFER EVERYTHING FROM A PROGRAM THAT DOES FANTASTIC HIGH RESOLUTION GRAPHICS TO ONE THAT ACTUALLY DIALS YOUR PHONE.....

****

MASTER MEMORY MAP(tm) — This is really the key to using the ATARI'S capabilities. We start out by explaining how to PEEK and POKE values into memory so that even new programmers can use this. Then: we give you over 15 pages of the memory locations that are the most useful. The information is condensed from both the ATARI'S Operating System Manual and various articles and programs. It is, of course, useful even for experienced programmers as a reference. Also, we highly suggest that dealers offer this Memory Map to customers who request to be told how to use the power of the machine. We guarantee it will answer many of the questions you have about the machine. **$5.95**

TRICKY TUTORIALS(tm)

#1: DISPLAY LISTS — This program teaches you how to alter the program in the ATARI that controls the format of the screen. For example: when you say graphics 8 the machine responds with a large graphics 8 area at the top of the screen and a small text area at the bottom. Now, you will be able to mix the various modes on the screen at the same time. Just think how nice your programs could look with a mix of large and small text, and both high and low resolution graphics. this program has many examples plus does all of the difficult caluculations! **$14.95**

#2: HORIZONTAL/VERITCAL SCROLLING — The information you put on the screen, either graphics or text, can be moved up, down or sideways. This can make some nice effects. You could move only the text on the bottom half of the screen or perhaps create a map and then move smoothly over it by using the joystick. **$14.95**

#3: PAGE FLIPPING — Normally you have to redraw the screen every time you change the picture or text. Now you can learn how to have the computer draw the next page you want to see while you are still looking at the previous page, then flip to it instantly. You won't see it being drawn, so a complicated picture can seem to just appear. Depending on your memory size and how complicated the picture, you could flip between many pages, thus allowing animation or other special effects with your text. **$14.95**

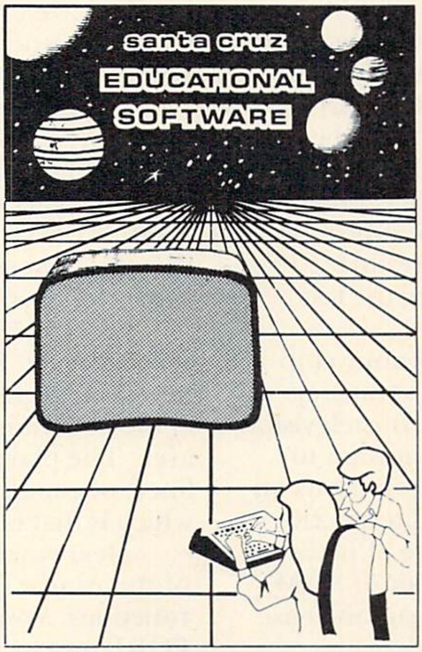
#4: BASICS OF ANIMATION — Shows you how to animate simple shapes using the PRINT and PLOT commands, and also has nice little PLAYER/MISSILE Graphics demo to learn. This would be an excellent way to start making your programs come alive on the screen. Recommended for new owners. **$14.95**

#5 PLAYER MISSILE GRAPHICS — This complex subject is demonstrated by starting with simple examples, and building up to a complete game and also an animated business chart on multiple pages! As always, the computer does most of the calculations. Requires 32K disk or tape and costs **$29.95**

#6: SOUND — From explaining how to create single notes, to demonstrating complex four channel sound effects, this newest tutorial is great. Even those experienced with ATARI's sound capabilities will find the menu of sound effects a needed reference that can be used whenever you are in the need of a special sound for your programs. Everyone will learn something new! Written by Jerry White. **$14.95**

Tricky Tutorials (except #5) require 16K memory for cassette orders and 24K for disk. The price is $14.95 each. You may order 1,2,3, & 4 for $49.95! All six in a colorful binder cost $89.95.

THE GRAPHICS MACHINE!! — Turn your computer into an incredible graphics tool with advanced commands like circle, box, fill, polygon, line, help, etc. 3 colors in graphics 8 with instant text!!! Create colorful business charts or beautiful drawings and then save or retrieve them from disk in 5 SECONDS. YES, it's that fast. Needs all 48K, disk, and costs **$19.95**



MINI-WORD PROCESSOR — This is for those of you who have a printer, but don't want to spend $100 or more for a fancy word processor. It is suitable for simple editing of text, accepts most control characters for your printer, and text is stored on disk for easy retrieval. Holds 2½ typed pages at a time. Requires 32K, disk or tape. **$14.95**

BOB'S BUSINESS — 14 small business type programs for home or office, all chosen from a nice menu. Supports printed output. 169 sectors of output require 16K tape, or 32K disk. **$14.95**

KID'S #1 — Includes the following: 1) TREASURE — search for the lost treasure while trying to keep from falling into the sea. Nice graphics if you find it!; 2)DIALOGUE — talk back to the computer about four subjects; 3) MATH QUIZ — Nice musical and graphical rewards for good scores. Parents input the level of difficulty.

KID'S #2 — A spelling quiz, a "scrabble" type game, and a version of Touch with the computer giving all the directions! Both Kid's programs require 16K tape or 24K disk and cost **$14.95** each.

MINI-DATABASE/DIALER — This unique new program stores and edits up to 8 lines of information such as name, address, and phone numbers, or messages, inventories or anything you want. It has the usual sort, search, and print options, but it also has an unusual feature. If your files include phone numbers and you have a touch-tone phone, the program will DIAL THE PHONE NUMBERS FOR YOU! This is perfect for those who make a lot of calls like salesmen, teens, or those trying to get through to busy numbers (acts as an auto-redialer). It is also a lot of fun to use. Requires 16K cassette or 24K disk and costs **$24.95**

FONETONE — For those who only want to store name and phone numbers and have the dialer feature as above, we offer this reduced version. Same memory requirements, but only costs **$14.95.** Don't forget you must have a touch-tone phone.

PLAYER PIANO — Turns your keyboard into a mini-piano and more. Multiple menu options provide the ability to create your own songs, save or load data files using cassette or diskette, fix or change any of up to 400 notes in memory, and play all or part of a song. The screen displays the keyboard and indicates each key as it is played from a data file or the notes you type. You don't have to be a musician to enjoy this educational and entertaining program. Requires 24K cassette or 32K disk. **$14.95**

BOWLERS DATABASE — Provides the league bowler with the ability to record and retrieve bowling scores providing permanent records. The data may then be analyzed by the program and displayed or printed in summary or detail form. Data may be stored on cassette or diskette and updated quickly and efficiently. The program proivdes such information as highest and lowest scores by individual game, (first, second, and third games throughout the season), high and low series, current average, and more. The program listing and documentation provided are a tutorial on ATARI basic and record keeeping. Requires 16K for cassette or 24K for disk. **$14.95**

By the time you read this all computers (400/800) being produced should have the fabled GTIA chips included. ATARI service may upgrade older computers... call and ask (it's easy to do yourself). We have one and the improvements that graphics modes 9,10, and 11 offer are great!! To help you figure out what to do with the new modes a new Tricky Tutorial will be offered in March on Modes 9 to 11. Either give us a call or write around that-time.

****

SEE YOUR LOCAL DEALER
OR
ORDER FROM:

SANTA CRUZ EDUCATIONAL SOFTWARE
5425 Jigger Dr., Soquel, CA 95073
C.O.D. Orders call (408) 476-4901
*Include $2.00 postage ($.75 for Memory Map)
*In California, include 6.5% tax

Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky

Program 1 is an example of the initialization and use of the VBLANK PM subroutine. This program causes VBLANK PM to be loaded and initialized and players zero and one to be drawn and then moved about the television screen in a random pattern. The players are male and female gender symbols which the program "dances" around the screen.

Lines 100 through 200 are the main program; we'll save an explanation of these lines until after you've gained some insight into the initialization subprogram contained in lines 1000 through 1110. The VBLANK PM machine language subroutine is expressed in the DATA statements numbered 2000 through 2100. Finally, lines 3000 through 3020 supply a description of the two players used in this example.

The first task is to load VBLANK PM into page six of memory. Page six is locations 1536 through 1791 (hexadecimal 600 through 6FF) and has been left available by Atari's software designers for applications such as this one. These 256 bytes of memory are not disturbed by BASIC or DOS; however, a cold start does cause page six to be cleared to zeroes. Line 1010 causes the VBLANK PM to be read and POKEd into memory. Line 1020 clears a few locations used by the subroutine; this statement can be omitted if you are sure that page six has not been altered since the last cold start.

We're going to employ the Atari's antic chip direct memory access (DMA) facility to transfer graphics information from memory to the television using single line resolution. (You might want to reread Chris Crawford's article or just "trust us on this one!") This means that we must allocate 2K (2048) bytes of memory for the storage of players. In line 1030 we obtain the page number of RAMTOP, deduct 16 pages, and call the result the base of the required 2K byte allocation.

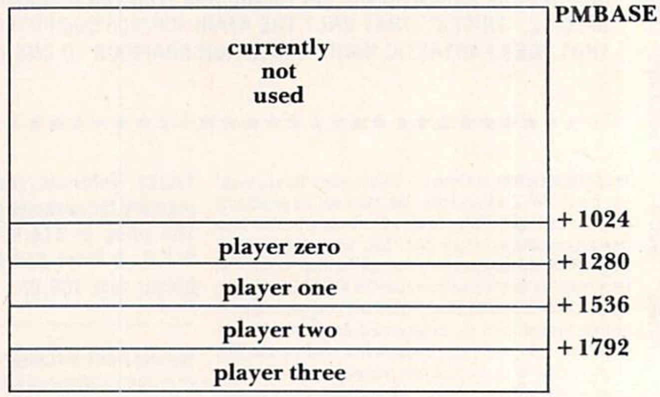
Memory Allocation

Why 16 pages? Well, first consider that 2K bytes are eight pages (a page contains 256 bytes) and that, depending on the graphics mode (i.e., GRAPHICS 0 through GRAPHICS 8), you must allow sufficient space at the top of RAM to contain the display list and screen data. Incidentally, the player-missile 2K byte allocation must begin at an address which is a multiple of 2048; we call this starting address PMBASE.

One more cautionary note: you will have to allow more than 16 pages between PMBASE and RAMTOP if you are using graphics modes six through eight. Fred Pinho's article in **COMPUTE!** #16 provides greater detail in this area.

Figure 1 depicts the 2K byte memory allocation.

Figure 1.



Remember, we didn't design this scheme, Atari did, and we're not sure why but there is a considerable amount of unused space involved. You can use the lower, unused bytes for your own purposes without disturbing anything, if you like. We're only going to use the upper 1K bytes.

Player zero occupies PMBASE + 1024 through PMBASE + 1279; player one is situated in locations PMBASE + 1280 through PMBASE + 1535, and so on for players two and three. Line 1040 clears any residual data – if you're in a hurry and are sure that this area is already clear (i.e., following a cold start), you won't need line 1040.

Lines 1050 and 1060 are used to draw players zero and one. VBLANK PM expects the players to be drawn such that their top line is initially placed at the beginning of the individual player's storage area. The player can be as tall as you like up to 255 lines; of course, you will never see all of a player which is that tall on the screen at the same time!

Next you can see that we've taken advantage of the Atari's special memory locations for some functions. You establish the players' colors with a POKE into locations 704 through 707 for players zero through three, respectively. Line 1070 is used to set the colors and assumes that you've set the variables PCOL0, PCOL1, PCOL2, and PCOL3 already.

Line 1080 establishes the positioning addresses which you will be using later to signal player movements using only POKEs. PLX and PLY are the locations POKEd to establish the next x and y position of player zero. A POKE into location PLX + 1 and PLY + 1 accomplishes the same thing for player one, and so forth for players two and three. PLL (and PLL + 1, PLL + 2, and PLL + 3) are POKEd to inform VBLANK PM of the length (or height) of each player.

Line 1090 initializes the remaining control parameters. A 62 is POKEd into location 559 to set the single line player-missile resolution graphics; a one placed into location 623 establishes the player/playfield priorities giving the players priority over the playfield. (You can change this to suit your purposes, if you wish.) Location 1788 is in VBLANK PM and is POKEd with the number of the first page containing player-missile data. Locations 53277 and 54279 are used to switch on the DMA graphics data transfer facility and to tell the ANTIC chip where in memory to find the player graphics data.

Wrapping Up The Loose Ends

You're almost ready to go! A subroutine call to VBLANK PM from line 1100 allows VBLANK PM to notify the operating system of both his presence and his desire to be automatically invoked as a part of the vertical blank interrupt process. This is the only time in which your BASIC program must explicitly call VBLANK PM.

Okay, to wrap up loose ends, let's take a quick look at the main program – lines 100 through 200. Line 100 turns off the cursor, clears the screen, and provides a black background so that we can readily see the players.

Line 110 sets the players' colors before the VBLANK PM initialization subprogram is executed. You know how to set the colors, right? Multiply the color number by 16 and add the desired intensity – the color and intensity numbers are the same as those used in the SETCOLOR command. Line 120 assures the VBLANK PM is launched.

Line 130 illustrates the manner in which you pass instructions to VBLANK PM. Here we are telling VBLANK PM that both players are eight lines tall. You can change this parameter at any time – we have a little surprise for you later about why you might want to change this parameter.

Lines 140 and 150 establish the initial television screen positions of players zero and one, respectively. A word about the available values for the x and y coordinates might be helpful as not all x and y values will result in the player being displayed. There are 255 x positions with only 160 of these appearing across the television screen beginning with an x value of 48.

Similarly, there are 255 y positions with 192 of these visible on the screen beginning with 32 at the top. (These x and y values may vary slightly depending on the adjustment of your television receiver.) VBLANK PM assumes that you are referring to the upper left hand corner of your player whenever you POKE new x and y coordinate values.

Lines 170 and 180 illustrate the use of the pseudo-random number function to determine the next set of x and y coordinates. Line 190 provides a small delay between player movements. Delete the FOR and NEXT statements if you want to see how fast – and easy – it is to move players.

Well who said player-missile graphics had to be anything but fun?! Give VBLANK PM a try in one of your current programs to add a little zip; or try it in your next graphics project.

Oh, we almost forgot that we promised you a surprise regarding why you might want to change the height of a player. VBLANK PM has a few more features which allow you to animate the movements of your players – but more about this next time!

```
100 POKE 752,1:PRINT CHR$(125):SETCOLOR 2,0,0
110 PCOL0=216:PCOL1=56:REM color of players
120 GOSUB 1000:REM initialize vb routine
130 POKE PLL,8:POKE PLL+1,8:REM player's height
140 POKE PLX,108:POKE PLY,102:REM player
 0's initial position
150 POKE PLX+1,108:POKE PLY+1,72:REM ditto player 1
160 REM let players dance!
170 POKE PLX,RND(0)*159+48:POKE PLY,RND(0)*191+32
180 POKE PLX+1,RND(0)*159+48:POKE PLY+1,
RND(0)*191+32
190 FOR I=1 TO 75:NEXT I:GOTO 170
200 END
1000 REM INITIALIZE VBLANK PM SUBR
1010 FOR I=1536 TO 1706:READ A:POKE I,A:NEXT I
1020 FOR I=1774 TO 1787:POKE I,0:NEXT I
1030 PM=PEEK(106)-16:PMBASE=256*PM
1040 FOR I=PMBASE+1023 TO PMBASE+2047:PO
KE I,0:NEXT I
1050 FOR I=PMBASE+1025 TO PMBASE+1032:RE
AD A:POKE I,A:NEXT I
1060 FOR I=PMBASE+1281 TO PMBASE+1288:RE
AD A:POKE I,A:NEXT I
1070 POKE 704,PCOL0:POKE 705,PCOL1:POKE
706,PCOL2:POKE 707,PCOL3
1080 PLX=53248:PLY=1780:PLL=1784
1090 POKE 559,62:POKE 623,1:POKE 1788,PM
+4:POKE 53277,3:POKE 54279,PM
1100 X=USR(1696)
1110 RETURN
2000 REM vblank interupt routine
2010 DATA 162,3,189,244,6,240,89,56,221,
240,6,240,83,141,254,6,106,141
2020 DATA 255,6,142,253,6,24,169,0,109,2
53,6,24,109,252,6,133,204,133
2030 DATA 206,189,240,6,133,203,173,254,
6,133,205,189,248,6,170,232,46,255
2040 DATA 6,144,16,168,177,203,145,205,1
69,0,145,203,136,202,208,244,76,87
2050 DATA 6,160,0,177,203,145,205,169,0,
145,203,200,202,208,244,174,253,6
2060 DATA 173,254,6,157,240,6,189,236,6,
240,48,133,203,24,138,141,253,6
2070 DATA 109,235,6,133,204,24,173,253,6
,109,252,6,133,206,189,240,6,133
2080 DATA 205,189,248,6,170,160,0,177,20
3,145,205,200,202,208,248,174,253,6
2090 DATA 169,0,157,236,6,202,48,3,76,2,
6,76,98,228,0,0,104,169
2100 DATA 7,162,6,160,0,32,92,228,96
3000 REM players 0 & 1
3010 DATA 6,6,8,126,195,195,195,126
3020 DATA 126,195,195,126,24,126,126,24
```

©

Review:

Eastern Front (1941)

Edward P. McMahon
Potomac, MD

Eastern Front (1941) by Chris Crawford of the Atari Staff is a paradigm for computer war games. Not a shoot-em-up type arcade game, it is a corps-level historical simulation. The subject of this excellent simulation is the first 41 weeks of Operation Barbarossa, Hitler's massive attack on Russia which began on June 22, 1941.

Eastern Front has many features of a well-done historical simulation wargame: simultaneous movement of both players, supply rules, reinforcements and resupply effects, and effects of terrain. There is some time pressure also, which is not usually found in simulation wargames of the board-and-counter variety. The computer (a worthy opponent playing the Russian side) thinks out its move during the vertical blank periods when you are planning your moves. The more time you take, the better will be the computer's move. More on this later.

The game starts immediately after booting in (it is an AUTORUN.SYS file on the disk version), but first-time players don't immediately respond. They are entranced by the graphics presentation. The playfield is 2⅓ screens horizontally and 4⅓ in the vertical dimension and is filled with excellent redefined character sets – mountains, rivers, forests, marshes, cities and coastal areas. As you move your hollow square cursor to any edge of the screen window, the map smoothly fine-scrolls to display the correct part of the playfield. The attention to detail is admirable. The trees in the forest areas are different sizes; the rivers and coasts are displayed to the highest possible resolution. The colors have been carefully chosen – I have not noticed any "bleeding" between adjacent colors – and dramatically indicate the change of seasons. The autumn season begins on October 5, 1941, when the green land changes to a purple-brown mud color. (Remember that date. If you haven't captured your objectives, destroyed most of the original Red forces, and established a strong defensive position by then, you are in trouble.) The ground changes again to white in winter, and the rivers and marshes freeze (blue to white) from north to south as the weeks progress. The process reverses in the spring. Another very nice detail.

A few words on the history (*History of the Second World War,* Sir Basil Liddell Hart (ed), Marshall Cavendish USA, Ltd., 1973-1974). Hitler began open plans to invade Russia with discussions in June, 1940. A late spring offensive was planned, and the first strategy (by Maj. Gen. Marcks) was

...it is a corps-level historical simulation.

two thrusts – the largest to Moscow through Smolensk, the second to Kiev. These would join in a pincer movement, trapping most of the Red Army. General Halder and the German High Command modified the Marcks plan by weakening the Kiev thrust to strengthen the push to Moscow, and added a third line of attack to Leningrad. Three Army Groups were defined: Army Group North (von Leeb), Army Group Center (von Beck), and Army Group South (von Rundstedt). Von Kleist's I Panzergruppe and Guderain's II Panzergruppe were aimed north and south of the Pripet marshes respectively. The General Staff and probably Army Group leaders played out major war games in late 1940, taking both sides of the campaign. But early in December, Hitler made what the German Army War Diary calls "a substantial alteration." Leningrad became the principal military target and Moscow was to be taken afterward.

The aim was still rapid advance and encirclement to prevent the Red Army from escaping into the interior, and the destruction of Russia's industrial power in the Ukraine, in Leningrad, and in Moscow. But Hitler's modification had the Army Group Center waiting until Army Group North achieved its more difficult, more distant objective before going on to Moscow. The High Command did not argue successfully with Hitler, and the directive for Operation Barbarossa was signed on December 18, 1940.

On June 22, the longest day, the largest invasion in the world began against an army which had suffered Stalin's 1937-1939 purges: three of five Marshals, 13 of 15 Army Commanders, 57 of 85 Corps Commanders and more had been shot or disappeared without a trace. The German attacks were devastating in the North and Center (Smolensk fell on July 15, but Kiev held out as a pocket of resistance until late September). Nearly two-thirds of the Red Army's strength at the outbreak of the war was destroyed. The Germans occupied Russia up to a line from Leningrad to the Crimea.

Estimated losses by the end of 1941 for the Red Army were 5-7 million killed or wounded, 3-5 million P.O.W., 21,000 tanks and 33,000 guns destroyed. Russia fought back with extraordinary national effort, calling on all its resources and extensive Allied help. The Germans achieved some additional victories, but the Blitzkrieg was blunted by the vastness of Russia, the mud and the cold.

Your only hope of winning the simulation is to follow the suggestions of the author, Chris Crawford, in the excellent user's manual which comes with the game: break through and use the mobility of the armored units to encircle the Russian corps from behind, and concentrate forces by pushing your infantry as fast as possible to attack and eliminate pockets of enemy units. These are the classical Blitzkrieg tactics. But, before the autumn mud stops your panzers, form a defensive line using terrain (rivers and cities) to your advantage. Fall back in order during the winter counterattacks.

The game is well documented, but the mathematical rules of combat and supply are not given. The user interface is well designed, so the game is almost entirely playable from the joystick alone (three keys are needed: START, OPTION, and SPACE BAR). The only feature which I find seriously lacking is the ability to save a game and restart it. The game takes two or three hours to play (more, if you want to keep a record of what you are doing) and I find it difficult to come up with an uninterrupted block of time like that. Moreover, I can't study different moves for a given situation, but perhaps that's good. The unknowns can't be resolved, so the game keeps my interest. There is randomness in the combat and supply rules (a good feature) so a tactic which works today may not work tomorrow. Another reason why replaying a tactic may not give meaningful results was mentioned earlier: the computer works on its move while you are entering yours. The computer selects a move for each of its units and, as time is available, iteratively improves each move. This is the feature of the game in which Crawford takes the most pride.

Crawford states that he uses only 75% of the Atari's graphics capabilities. He should know. He is one of Atari's most creative staff members, and certainly understands the machine. The way he uses that 75% makes *Eastern Front (1941)* a showpiece and a challenge to other program designers. (If he ever uses 100%, I think I'll sit and stare for a week or so.)

I am still experimenting with small, local tactics, as hopeless as that may be. If I ever get to March '42 again (before 2 a.m.). perhaps a late winter thrust to push some muster points farther west will add some victory points. Hmmm... ©

Odds & Ends

Clearing Memory

Charles Brannon
Editorial Assistant

Before using an area of memory for storage, it is often necessary to clear it out. For example, a GRAPHICS command clears the screen by writing zeros to all the screen memory. Since there are no BASIC statements that directly support player/missile graphics, the memory used by this facility has to be cleared by the programmer, usually with a FOR/NEXT loop.

ATARI BASIC does not clear out the old values of an array or string when a program is RUN, even if there is garbage in the memory used by these variables. This also necessitates some kind of loop to clear out this memory.

The problem with this is that an array of any substantial size requires a long time to clear out. For strings, there is a shortcut:

```
10 DIM A$(100)
20 A$(1)=" ":A$(100)=" ":A$(2)=A$
```

Line 20 will "instantly" fill A$ with spaces. The space in quotes can be changed in order to fill a string with a desired character.

There is a "quick and dirty" way to clear out memory. This relies on the previously mentioned GRAPHICS command. GRAPHICS 8+16:GRAPHICS 0 will clear out about 8K of high RAM. If executed before a DIM statement, this will usually suffice. Since most Player/Missile memory is in the top of memory, the GRAPHICS command is definitely satisfactory. If you don't have 8K of free memory, you'll get an ERROR-147 (Insufficient RAM for GRAPHICS mode), in which case you'll have to use GRAPHICS 7+16 (or lower), or resort to the BASIC clear loop. ©

COMPUTE! Overview:

Textwizard

Textwizard consists of a copy- and write-protected disk containing the program, an instruction manual, and a reference card which summarizes all commands. All of this is in a luxuriously padded binder.

Original purchasers may request a back-up disk when the warranty card is returned with a $5.00 check. After that, clobbered discs can be exchanged at the factory for a $30.00 fee.

Forewarned, we treated the disk with great respect. What we found was a carefully human-engineered word processing system for a personal computer. It will not turn your Atari into a $15,000 professional word processor, but many of the important differences between Textwizard and a professional system are a function of the 40-character display. Thus, it is not reasonable to consider constructing large organizational charts, flow charts, or even moderately complex graphs. Tables of data which span more than 40 columns are tortuous to create. Other features usually seen in professional systems, including automatic hyphenation, positioning of footnotes at the bottom of a page, tabular sorts, arithmetic functions and representation of mathematical symbols are not available on Textwizard.

The Manual

First impressions of the program come from the well made and finished looseleaf text which accompanies the program. Filled with 56 pages of instructions, including an index, the manual gives the feeling that the producers of this program care about quality.

The *Instruction Guide* is, however, for one panelist, the weakest element in the package. It was not carefully proofread nor carefully tested with naive users. There are simply too many instances where the user who methodically follows the step-by-step instructions is left hanging with a feeling of "What do I do now?"

In contrast to the above, one panelist was pleased with the manual: "The user's manual is a very good example of what in some educational circles is known as the "KISS" of knowledge (Keep It Short and Simple). The manual walks the user through each function in logical, explicit language, and is itself an excellent example of programmed learning."

"...the manual gives the feeling that the producers of this program care about quality."

Ease Of Use

The program uses the Atari DOS system and diskettes must be formatted before you can store data on them. Data saved to disk appears to be compacted prior to saving, thus increasing the amount of data which can be stored on each disk.

The program is written completely in machine language and is loaded without the left cartridge in place. This increases the amount of available memory for storage of the text. With 48K of memory installed in the computer, there is just over 30K left for text. Eighteen K Bytes of program seems somewhat large, considering the stated capabilities of the program; however, this size would certainly seem to take it out of the "kid's toy" category.

The program boots quickly and, unless the amount of text being held is substantial, no loss of speed is noticed. There is one quirk which appears when the amount of memory in use begins to exceed 5K. There is a delay in the text's appearance on the screen when the inserting command is being used. This can be somewhat disconcerting if the operator is a touch typist and is watching the screen. The nice part is that the letters are all picked up and, if the typing is done accurately, it will all eventually show up.

When first used, however, the program appears to lack features found in other word processors. The program is not menu driven. Consequently, a number of functions such as loading, saving, disk formatting, and drive # setting, require that the operator remember specific keyboard sequences. Some of these are quite lengthy and would be more easily used if they were contained on a menu. The Atari full cursor movement feature is well used; however, there are no provisions for fast single stroke movement from the center of a line to either end of the line. These are not essential, but

would speed editing. There is no scrolling ability which tends to limit the speed with which text can be edited. Offsetting this, to some degree, is the ability to move through the text in either direction using a combination of the OPTION key and the cursor arrow keys. The system is friendly. It may be conceptualized as two major programs: one is for creating and editing text, and one is for printing documents. With very few exceptions, commands to the editing component involve only two keystrokes.

Text formatting and printing functions utilize Atari's special keyboard controls (i.e., OPTION, SELECT, START) in conjunction with other keyboard characters which, when possible, relate to the particular function or mode required. For example, CTRL + T sends the cursor to the top of the page, CTRL + B sends the cursor to the bottom of the page; CTRL + M is used to Move text, CTRL + D to Duplicate text, CTRL + S for Search, etc. The use of characters such as BACKSP, INSERT and editing ARROWS will be familiar to Atari users and greatly facilitate operation – especially for the word processing beginner.

One particular feature proved extremely versatile. Margin control is set, not in inches or character widths, but in *dot* widths. Although this necessitates some computation by the user (150 dots = approx. 1 in.), it allows "fine tuning" when formatting a document; some very creative copy can be produced with such a tool. It would be a nice touch, perhaps, if Datasoft supplied a margin guage or ruler which translated dots into inches and/or character widths as a useful addition to the Command Reference Card – which, in itself, is very clear and precise.

One severe problem occurs with the buffer operation. There is no protection for information stored in the buffer. Thus, inadvertent loss of text can and does happen unless the operator takes careful note of buffer use. It seems best not to store text in the buffer for any length of time, but rather to use the buffer for simple movement operations only. The lack of a screen display for print formatting requires that the printer be used each time the operator wishes to see the actual formatting of the text. This can account for a significant loss of time and reams of paper being generated, when the actual formatting on the page must be seen. One panelist felt that this, combined with an inability to print single pages of a long text file, proved to be the most serious deficiency of the program.

Search, Merge, Disk Functions

The search feature does not work reliably, particularly when the string involves as few as three characters. For example, on two Model 800 systems a search for "he" (the string, h + e + space) not only correctly identified all occurrences of "he," but also incorrectly located embedded instances of "he" (as in "wherever") and, worse yet, totally inappropriate strings (e.g., "Ruth."). Search/replace operations were similarly plagued. Even when it behaves properly, global replacement is only semi-automatic: each "old phrase" must first be located; replacement with "new phrase" must then be manually verified.

Tab stops are preset to five spaces, and cannot be altered. This makes the construction of even 4- or 5-column tables of numbers overly cumbersome.

Generally Good Fatal Error Protection

It is generally very difficult for a user to make fatal errors. The exception to this rule is, however, important. File deletion is accomplished by simultaneously pressing OPTION D, followed by a file name. Even though the contents of the file still exist on the disk, they are thereafter forever inaccessible to the user. For an operation with such important consequences, it is reasonable to expect the system to help prevent the user from making devastating mistakes. Atari DOS requires an affirmative acknowledge prior to a file deletion. This feature probably should have been included.

Text requiring no special formatting features (pagination; underlining; centering; use of superscripts, subscripts, page numbers, boldface, etc.) may be printed to an Atari 825, Epson MX80, or Centronics 737 printers by issuing the command sequence OPTION P, followed by the filename. The print routine incorporates defaults for left, right, top, and bottom margins, proportional spacing, and right margin justification.

Overriding any of these defaults, or incorporating any of the many special features, requires the user to embed a command string within the text file. Features common to the entire file (e.g., placement of page numbers) are indicated on the first line in the text. Other commands (e.g., for centering) are embedded as they are needed. Unlike some other word processors, Textwizard does not permit underlining on the Epson MX-80. An approximation to boldface type may be made on the Epson, but not other printers; this restriction, too, is odd since Letter Perfect can produce boldface on all three printers. Generation of superscripts and subscripts is easy, but the instructions fail to mention that it cannot be accomplished on the Epson.

Pagination Is Especially Flexible

Arabic numerals may be placed automatically anywhere on the top or bottom lines of a manuscript. Since pagination may begin with any value, sections or chapters of a manuscript may be independently prepared.

A nice feature is the option to print text in double columns. It is a little tricky to set up because the columns will not align evenly *unless* care is taken when placing carriage returns. It was not bothersome, however, and, after some experimenting, the text printed very nicely, each column having margin requirements pre-selected by the user.

On page 43 the automatic page numbering function is presented as CTRL+@. This combination does not work. It took a few minutes to locate the correct sequence for page numbering; it is SHIFT +@.

Its few disadvantages considered, Textwizard has a number of features which do set it apart from a simple text editor and turn it into a functional word processor.

One reviewer argued that the best single feature of the program is the Insert Text function. Because of the ease with which this works, there is little need for a lot of text moving. During this mode of operation, the operator is able to make insertions anywhere in the text without concern for erasure of previously written text. In addition, the screen border changes color during this operation, providing a constant reminder of the mode of operation. Wrap-around is maintained during the insert phase and this is a definite asset during text editing. Indeed, the ability to wrap-around text, thus keeping the text on the screen readable, is one of the really fine, and well-executed, features of Textwizard.

The use of changed screen colors and borders is a feature of Textwizard, which truly takes advantage of the versatility of the Atari, and helps to bring to this product an overall feeling of polish, while giving the user a very clear indication of the current mode of operation.

Print commands and formatting ability with this program are superb, if the Atari 825 or Centronics 737 printer is used in conjunction with the program. The ability to do multiple column printing is a great asset and it is here that Textwizard demonstrates a clear superiority over other word processors for the Atari. This feature alone would make the program worthwhile for anyone who publishes a newsletter.

Printing copy on cut (vs. continuous) paper is facilitated by the page eject and wait commands. The former performs a form feed, seeking the top of the next page. The latter causes printing to be suspended until a new sheet of paper is loaded.

The chaining feature is a powerful means to overcome the limitations imposed on the size of text files by the amount of available memory. With a 32K system, no text file may be greater than the equivalent of about 6.5 single-spaced pages. Sooner or later, most users will confront this ceiling, and will despair unless they see one implication of CHAIN: manuscript components (ranging from single characters to the largest amount of text permitted by memory) may be strung together with a command string in the first text file, resulting in the sequential printing of the whole.

The Final Overview

- **Panelist #1:**

"Overall, Textwizard is a very clean, useful word processor, delivering all that Datasoft says it will. It is easy to use and requires very little effort on the user's part to get excellent performance. It is also fast. The editing and searching functions are extremely swift and accurate. The chain command works well and facilitates printing and editing large blocks of text efficiently. The only two enhancements it could use (but doesn't need) are graphic display of the formatted page, and perfect spelling."

- **Panelist #2:**

"In short, Textwizard is a generally well-conceived word processing system for the Atari. Sometime between conception and delivery to the user, however, various gnomes intruded and left indelible marks on the product. Textwizard is well-suited for preparing term papers, inter-office memos, and informal personal correspondence. It may even be appropriate for the Great American Novelist. Professional technical writers and business executives will be happier and more productive with the much more powerful – and costly – word processing products that are targeted to their more complex needs."

- **Panelist #3:**

"With over fifty commands available to aid in editing, formatting, storing and printing text, Textwizard certainly provides the user with serious word processing capabilities. The program is well thought out; the formatting commands are simple and easy to use. Although a touch typist will probably have some small difficulty learning to use the extra keys with finesse, this is certainly not a drawback of the program. While certain portions of the program are weak...lack of menu and scrolling, and a very time consuming search and replace function, these are more than offset by the speed and ease of use which other areas of the program deliver to the user. All things considered, Textwizard, at a list price of $99.95, is a good buy and one which could be recommended to all Atari 800 owners."

Textwizard. Datasoft Inc., 19519 Business Center Drive, Northridge, CA 91324. $99.95. 32K and one or more disk drives and compatible with Atari 825, Centronics 737, and Epson MX-80.

©

Put Graphics Modes 1 And 2 At The Bottom Of Your Screen

R. Alan Belke
DeKalb, IL

Most of you who are regular readers of **COMPUTE!** are familiar with the mixing of the graphics modes ("Mixing Atari Graphics Modes," **COMPUTE!** #6). The only problem is that you can't use a mode past its regular range. That is, if you wanted to use Mode 1 past line 20 or Mode 2 past line 10, you couldn't. So you were stuck putting text you wanted at the top of the screen or in the text window. Until now, that is!

What's The Display List?

First we'll look at the Display List to see what it is and what it does. Figure 1 shows the Display List for Mode 3. You can verify this by running Program 1. Locations 560,561 contain the starting address of the list.

Figure 1.

112,112,112,72,112,158,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,66,96,159,2,2,2,65,78,158

The purpose of the list is to tell the computer how to display the information stored in the screen and/or text memories. Let's see how it does this. The first three bytes (112) set up the margin at the top of the screen. How they do this, I don't know. Anyone out there know? Next comes what I call an address byte, (72). In this case, a Mode 3 address byte. (Figure 2 shows what the address bytes are for each of the modes.) This byte pulls double duty. First, it sets the first line to Mode 3. Then it tells the computer that the next two bytes contain the address of the screen memory.

Figure 2.

MODE	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
ADDRESS BYTE	**66**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**79**

The next 19 bytes (8) set one line each to Mode 3. I call these Mode 3 bytes. You get the value for these bytes by subtracting 64 from the address byte (72-64=8). From this, we can deduce that any byte with bit 6 on is an address byte. Also, notice that 19 Mode 3 bytes with the Mode 3 address byte give you 20 rows of Mode 3, which fills the screen up to the text window.

For whatever mode you are in, you will have 1 address byte and the number of rows, minus 1, regular bytes. For example, Mode 7 will have a Mode 7 address byte (72) and 79 regular Mode 7 bytes. Giving you 80 rows. To find out how many rows each mode has, check the "Table of Modes and Screen Formats." It's on the inside back cover of your *Basic Reference Manual.*

The Last Three Rows Of The Text Window

Now here's the important part. The next byte (66) is a Mode 0 address byte. But, instead of the next two lines containing the address of the Screen memory, they contain the address of the Text Editor memory. This is the start of the text window. Modes 1 through 8 use the Screen memory. Mode 0 uses the Text Editor memory. As you may have already guessed, the next 3 bytes (2) are Mode 0 bytes, giving us the last three rows of the text window. If we were in a full screen format, these last six bytes would not be here.

Now we are to the end of the list. This next byte (65) is also an address byte. But it has a special purpose. It tells the computer that it has reached the end of the list and that the next two bytes contain the starting address of the list. (The same as locations 560,561.)

Before we go on, let me say that the bytes that contain the addresses may vary, depending on the Mode you're in and on the amount of memory you have. All the other bytes will be the same.

So how do we get Modes 1 and 2 on the bottom of the screen? It's simple! Basically, all we do is change the Mode 0 bytes to Mode 1 or 2 bytes. Presto! The computer now displays the Text Editor memory in Modes 1 or 2.

Let's look at Program 2 to see how this is done:

Line 10: sets the margins to 40 characters per line and selects mode 3 with text window. Then it finds the address of the Display List.

Line 20: searches the list for the start of the text window.

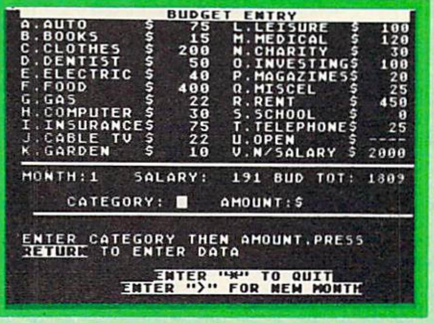
for Atari*

A Financial Wizard

The Ultimate Personal Finance Management System

This new, comprehensive system enables you to determine how much you can afford to spend and what can be saved or invested. It graphically illustrates the answers to: "Where does the money go?" It also does a great deal more . . . nothing else even comes close!

Budget Projections

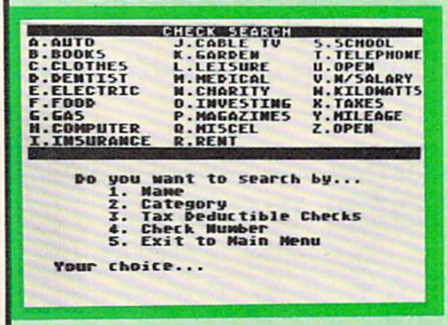


You are given 21 categories for which you enter a monthly plan and your anticipated income. The categories can be modified by name or amount at any time by you. The program prompts you every step of the way.

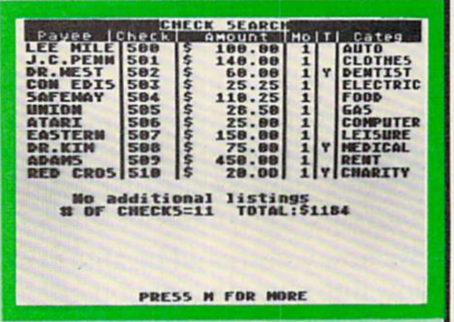
Check Entry

Enter data . . . month, check #, amount, y or n for tax deductibility, payee. Enter bank deposits and interest. Record up to ninety-six checks per month in amounts up to $99,999.00.

Check Search Menu

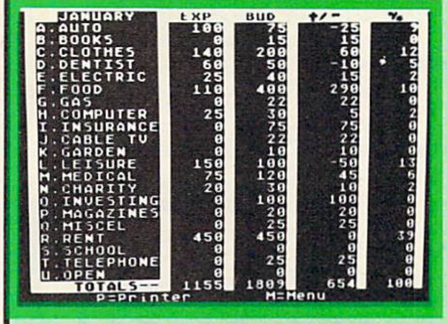


Check Search



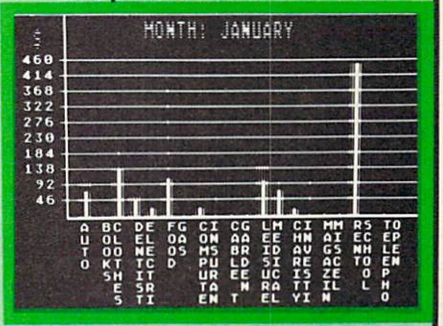
Your checks can be sorted and displayed in four different ways: Name, Category, Tax Deductibility, Number. This can be done on a monthly or annual basis. The checks, deposits and interest data are automatically filed for access by the Bank Statement Reconciliation Program.

Expense Comparisons



Detailed comparisons are displayed on a monthly or categorical basis which show ± and percentage relationships. You can then identify and analyze obvious and not-so-obvious relationships of the entered factors.

Graphic Presentations



Colored bar charts are generated rapidly via machine language comparing the amounts projected with actual expenditures. Chart displays are produced for both months and categories.

Also included in the system are:

- Printout options—all popular 40/80 col. printers.
- Multi-color graphics and sound enhancements.
- 4 record keeping categories for electric usage, gas consumption or whatever you wish to track.
- Check reconciliation-calculating totals and displaying # of checks and deposits outstanding.
- An audio/visual reward if your account balances.
- 8 utility programs.
- User-friendly operation and full documentation.

The system is designed for Atari* computers having a minimum of 24K and operating from a disk drive.

* trade mark of Atari, Inc.

OK Residents Add 4 % Tax

The price of the system is only $49.95 plus $2.00 for handling/postage.
If your local dealer does not have the Financial Wizard . . .
Telephone orders are accepted on Mastercharge or Visa credit cards.
Mail order must be accompanied by check or money-order or credit card #.
Dealer inquiries invited. (405) 751-2783 From Computari.

Exclusively thru ON LINE COMPUTER CENTERS of OKC™

**10944-A North May
Oklahoma City, OK 73120
(405) 751-2781**

☐ CHECK ENCLOSED ☐ VISA ☐ MASTERCHARGE

NAME (Print) ______________________

Address ______________________

City ____________ State ________ Zip ________

Card # ____________ Exp. ________ Sig. ____________

Line 30: changes the Mode 0 bytes to Mode 1 bytes.

Line 40-50: the periods denote spaces.

There are a few things of which to be aware. Even though you are using Modes 1 and 2, you're using the Text Editor memory; so the computer thinks in 40 column, not 20 column lines, which means two lines now equal one old line. Here is an example. Suppose we use an empty PRINT statement, planning to leave a blank line. Sorry, it won't work. We would have two blank lines. What we do is put 20 spaces in front of what we want printed on the second line. Also remember that we are using the Text Editor, so PRINT #6 will not work. Try some different things yourself.

What About Mode Two?

Well, that's almost as simple. Mode 2 lines are twice as wide as Mode 1 and 0; so there are only two combinations using Mode 2 possible: two rows of Mode 2 or one row Mode 2 with two rows of Mode 1. We can only use the same amount of room as was originally there. Program 3 uses the latter option from above:

Lines 10-20: same as Program 2.

Line 30: basically the same as in Program 2; only this time we make the second line Mode 2. And, since we use one less byte, we have to move the end of the list one location forward.

By now you should be able to change the text window into any combination of Modes 1 and 2 you want. If you have a program that would work better with the text at the bottom of the screen or the text window as Modes 1 or 2, get to work, experiment! Remember, you're the boss.

Program 1.

```
10 GRAPHICS 3:A=PEEK(560)+PEEK(561)*256
20 D=PEEK(A):? D;",";:IF D<>65 THEN A=A+
1:GOTO 20
30 ? PEEK(A+1);",";PEEK(A+2)
40 GOTO 40
```

Program 2.

```
10 POKE 82,0:GRAPHICS 3:A=PEEK(560)+PEEK
(561)*256
20 IF PEEK(A)<>66 THEN A=A+1:GOTO 20
30 POKE A,70:POKE A+3,6:POKE A+4,6:POKE
A+5,6
40 ? ".ATARI.AND.COMPUTE!.....AN.UNBEATA
BLE.."
50 ? "........TEAM..........FOUR.LINES.M
DE..1"
60 COLOR 2:SETCOLOR 1,10,6:PLOT 17,1:DRA
WTO 17,10:DRAWTO 9,18
70 PLOT 19,1:DRAWTO 19,18:PLOT 20,1:DRAW
TO 20,18
80 PLOT 22,1:DRAWTO 22,10:DRAWTO 30,18
90 GOTO 90
```

Program 3.

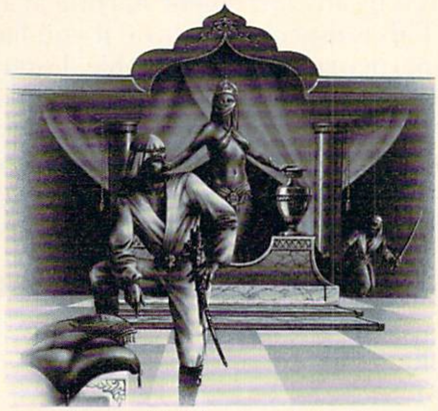
```
10 POKE 82,0:GRAPHICS 3:A=PEEK(560)+PEEK
(561)*256
20 IF PEEK(A)<>66 THEN A=A+1:GOTO 20
30 POKE A,70:POKE A+3,7:POKE A+4,6:POKE
A+5,65:POKE A+6,PEEK(A+7):POKE A+7,PEEK(
A+8)
40 ? ".ATARI.AND.COMPUTE!...1.LINE.OF.MO
DE.2."
50 ? "..2.LINES.OF.MODE.1"
60 COLOR 2:SETCOLOR 1,10,6:PLOT 17,1:DRA
WOO 17,10:DRAWTO 9,18
70 PLOT 19,1:DRAWTO 19,18:PLOT 20,1:DRAW
TO 20,18
80 PLOT 22,1:DRAWTO 22,10:DRAWTO 30,18
90 GOTO 90
```

©

Fantasy for your ATARI

Ali Baba and the forty thieves

By Stuart Smith



Guide your alter ego, Ali Baba, through the thieves mountain den in an attempt to rescue the beautiful princess. Treasure, magic, and great danger await you! One or more human players can guide up to seventeen friendly characters through the many rooms, halls, and caves. Some characters wander around randomly, making each adventure a little different.

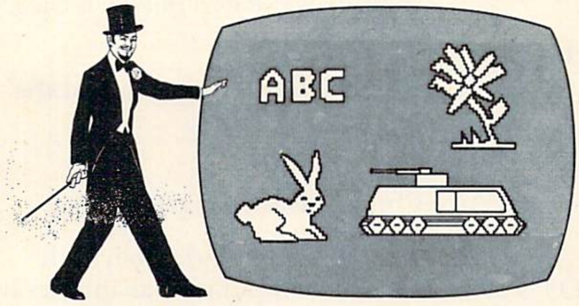
ALI BABA AND THE FORTY THIEVES is written in high resolution color graphics and includes music and sound effects. Adventures can be saved to disk and resumed at a later time. Requires 32K.

On Diskette Only — $32.95

Graphics for your ATARI

Character Magic

By Chris Hull



It's easy to create your own character sets and save them to diskette or cassette with CHARACTER MAGIC. But this is not just another character editor. CHARACTER MAGIC helps you use all the character types that the Atari is capable of, including descending characters (8 x 10 dots) and two types of five-color character graphics not supported by Atari's Operating System. Documentation includes examples of display lists that let you use these "secret" graphics modes. Requires 32K.

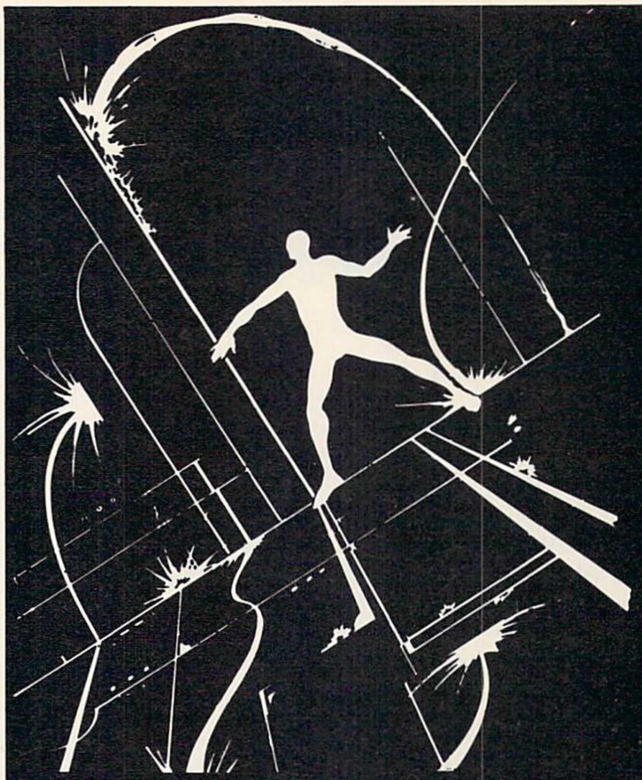
Cassette or Diskette - $29.95

FOR OUR COMPLETE LINE OF ATARI SOFTWARE PLEASE WRITE FOR OUR CATALOG

ASK FOR QUALITY SOFTWARE products at your favorite computer store. If necessary you may order directly from us. MasterCard and Visa cardholders may place orders by calling us at (213) 344-6599. Or mail your check or bankcard number to the address above. California residents add 6% sales tax. *Shipping Charges:* Within North America orders must include $1.50 for shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00. Pay in U.S. currency.

QS QUALITY SOFTWARE

6660 Reseda Blvd., Suite 105, Reseda, Ca. 91335 (213) 344-6599



MICROWORLD. an adventure within your computer is available on the ATARI and the TRS-80. You are transformed into an electroid and must explore the circuits of your computer. Over 80 locations and many original problems exist within the maze of transformers and transistors. We dare you to explore the maze of bit cells! Each version of Microworld explores the workings of its respective computer. Atari or TRS-80. Microworld comes with a booklet defining terms and describing the function of the mystifying inner workings of home computers. Come face to face with a staticon! Explore the Microworld!

SATISFACTION GUARANTEED!

If for any reason you are not satisfied with our products, return your order within 14 days for a prompt and cheerful refund.

ORDERING INFORMATION

Orders are processed within five working days. Shipping and handling charge of $1.00 will be added to all orders within the U.S. and Canada. Overseas orders please add $3.00 for air post.

Atari Microworld Atari 400 and 800
32K Cassette $19.95
Atari 400 and 800
32K Diskette $22.95
TRS-80 Microwrold TRS-80 Model I and Model III
Level II 16K Cassette $19.95
TRS-80 Model I and Model III
Level II 32K Diskette $22.95

■ MICROWORLD $
SHIPPING 1.00
TOTAL $

■ TRS-80 ■ ATARI
■ CASSETTE ■ DISKETTE
NAME
STREET
CITY STATE ZIP
■ CHECK ■ VISA ■ MASTERCARD
EXP. DATE

MED SYSTEMS SOFTWARE
P.O. Box 2674, Chapel Hill, NC 27514
1-800-334-5470

*Indicates trademarks of Atari

Atari PILOT At The Helm

Patricia Tubbs
Sunnyvale, CA 94086

As instructor for Gifted/Talented students in the Sunnyvale Elementary School District and educational computer consultant, I have had the opportunity of field testing Atari's PILOT language for the past year.

As a programmer in BASIC at an intermediate ability level, I've found Atari's PILOT extremely easy to use. Not only is text manipulation easily managed, but also both sound and high-resolution graphics are within the reach of beginners.

PILOT was originally developed by Dr. John Starkweather of the University of California Medical Center, San Francisco. It is a computer language which is word-oriented rather than number-oriented. People without any prior knowledge of computer programming find it easy and understandable to use in a very short time. The knowledge developed while using PILOT is a good foundation for moving on to other computer languages.

Curriculum-based Programming

ATARI PILOT makes preparing current curriculum-specific programs fairly easy. It has three modes of operation; they are: immediate mode, auto-number input mode, and run mode. When using the immediate mode your commands are executed immediately upon typing them and pushing the RETURN key. The auto-number input mode accepts PILOT statements, checks them for syntax errors and, if correct, assigns a number in sequence to each line and stores the statement in the program storage area. The run mode executes any program in the computer's memory.

With this language the programmer has the ability to control the appearance of words on the screen. In BASIC any print statement may appear to be spaced accurately and not divide words in inappropriate ways until that print statement is run. At that time the computer automatically divides any word at the end of a 40 space line. This is especially difficult if you wish to have a string variable (such as the student's name) inserted within that line. However, in PILOT, the computer will not break any words in a T: (type) statement, but will simply move them ahead to the next line. This feature is probably my favorite point for using PILOT when writing curriculum-based programs.

Another of the built-in features that is extremely helpful is the ability to renumber the program lines. The lines within the program may be renumbered by any increment starting at any number. This is especially helpful if you have created a particularly useful graphic design or musical piece and wish to use this module in some other program at another date. By renumbering the module to correspond to the new program, this module can become a part of the new program without the need of retyping it into the computer's memory.

The main text of your program is made with simple to use commands.

T: tells the computer to Type this on the screen.

A: the computer Accepts the user's input

M: matches the user's input with the programmer's expected answers.

C: Compute uses only integer arithmetic within the range of -32768 to 32767.

J:*LABEL allows the program to Jump to a module of the same name.

U:*LABEL allows Use of a module and then returns to the next statement following the U: statement.

***LABEL**
E: a module is created between these two commands.

GR: this command allows use of any of the various GRaphics capabilities.

SO: allows use of SOund.

PA: PAuses for a specific length of time.

Atari PILOT has been extended beyond PILOT's usual text and computational abilities to include Atari's capability for high-resolution graphics and sound. This graphics ability is called "turtle graphics," which comes to PILOT from the LOGO programming language, developed by Dr. Seymour Papert at MIT.

The programmer is able to control an imaginary robot called a "turtle" on the video screen. This turtle may be commanded to turn any number of degrees and to move forward any appropriate number of spaces. In doing so, it leaves a trace on the screen.

The full range of Atari colors is available for the turtle to use. However, you may use only three colors and the erase (or background color) at a time. After a figure has been drawn with the turtle, you may paint it by giving it a FILL command, at which time your figure is filled with color. (See

ALTERNATE REALITY SOFTWARE

Presents for the Atari

THE I CHING

THE ANCIENT CHINESE BOOK OF DIVINATION

- The complete text of the world's oldest book on disk
- 40K program
- 73 disk files (155,000 bytes)
- Occupying 1211 disk sectors
- High Resolution Graphics
- Animation
- Music and Sound Effects
- Instructional text material
- Calculates and Displays Hexagrams
- Displays Judgement, Image, Moving Lines for primary & secondary Hexagrams
- $44.95

order from:

Alternate Reality Software
2111 W. Arapahoe Drive
Littleton, Colorado 80120

Dealer inquiries invited
Atari is a T.M. of Atari, Inc.

ATARI® 400 AND 800™ COMPUTER OWNERS

AN OFFER YOU CAN'T REFUSE!

PDI produces some of the most innovative software for the Atari computer now available. In order to convince you, we have developed a sample cassette to show off our programming skills. The sample cassette has an animation-based educational program on side A, and a mixed graphics and voice demonstration on side B. Cost? Just $3.95. If you don't like the programs, simply erase and use the cassette as a blank! (The cassette is a superior quality, screw type C-20 with premium AGFA or BASF tape.)

If you accept our offer, we will credit $3.95 against any future order with our company. Just return the cassette with your order.

For the sample cassette send $3.95 to Program Design, Inc., 11 Idar Court, Greenwich, CT 06830. (Foreign orders include an extra $2.00 for postage.)

ATARI® is the registered trademark of ATARI, Inc.

SYNCRO, INC.

SOFTWARE DIVISION
31332 VIA COLINAS SUITE 107
WESTLAKE VILLAGE, CALIF. 91361
213-889-9508

SOFTWARE FOR ATARI T.M. 400/800

ALIEN HELL © 1982

ALL GRAPHIC ADVENTURE. CAN YOU FREE YOURSELF FROM ALIEN HELL? IN THE SPIRIT OF THE "MAZE OF DEATH".

JOYSTICK REQUIRED

16K Tape $14.95 — 24K Disk $19.95

MAR TESORO © 1981

A NEW GRAPHIC ADVENTURE, SIT BACK WITH YOUR JOYSTICK AND DIVE THE DEEP BLUE SEA. RENT RIG AND CREW-TO PLUNDER THE STRANGE TREASURES THAT AWAIT YOU.

JOYSTICK REQUIRED

24K Tape $16.95 — 32K Disk $21.95

MAZE OF DEATH © 1981

ARE YOU A SUPERAGENT? A FAST ACTION, ALL GRAPHIC GAME WITH PLENTY OF SPECIAL EFFECTS IN A MAZE LIKE UNDERGROUND COMPLEX. FIVE LEVELS OF PLAY IN REAL TIME.

JOYSTICK REQUIRED

16K Tape $12.95 — 24K Disk $17.95

VISA AND MASTERCARD ACCEPTED.
CALIF. RES. PLEASE ADD 6% SALES TAX
PLEASE INCLUDE $1.50 SHIPPING
DEALER INQUIRIES WELCOME

Sunrise Software®

BUDGETMASTER ($34.95)

- is a professional budgeting tool geared to develop and maintain even the most complex budget. At the same time, the easy to read manual and the friendly prompts allow even the beginning budgeter without computer experience to develop a budget to meet their needs.
- includes the option for hardcopy printout of any of the 39 different displays as well as an option to save your data to cassette or diskette.
- requires 32K RAM

CHECKBALANCER ($16.95)

- although the perfect companion program to the Budgetmaster, stands alone as a great tool to balance your checking account(s), with a monthly breakdown of your spending pattern.
- allows you to store your spending analysis to tape or diskette, or use **Checkbalancer** in conjunction with the **Budgemaster**, and automatically rewrite the expense data of the latter program process.
- requires 16K RAM

BUY BOTH PROGRAMS TOGETHER FOR $39.95 AND SAVE $11.95!!!

Send check or money order plus $1.50 for shipping and handling and save C.O.D. fee (Sorry, no credit cards)

Ohio Residents Add Sales Tax

(DEALER INQUIRIES WELCOME)

Sunrise Software® • P.O. Box 25621 • Garfield Hts., Ohio 44125
(216) 581-9359

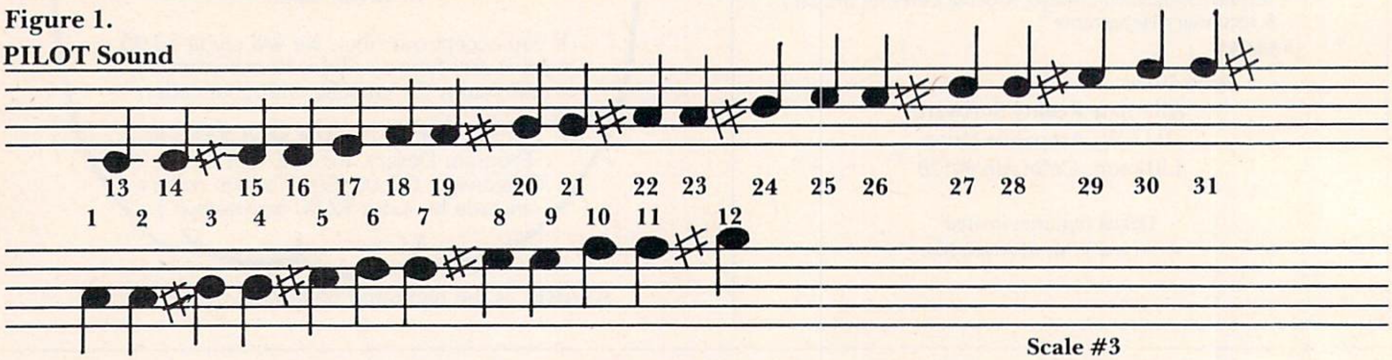
*Atari is a Trademark of Atari, Inc.

Program 1.)

When writing SOund components in your program you have the facility to command four voices (up to a four-note chord) between C below middle C to F# and C above middle C. This gives you a chromatic scale of 31 notes with C below middle C = 1 and F# above C above middle C = 31.

By including the PAuse command, one may hold each note or chord for a given length of time. The length of the pause is determined by the number of 1/60th of a second *ticks* selected (e.g. one tick equals 1/60th of a second, and 60 ticks equal one second). (See program 2 and Figure 1.)

The Atari joysticks and paddles can also be used within your program. You may use up to four joysticks in as many as 11 positions, with the joystick trigger having a two position value. Four pairs of paddle controllers with a range from 0 to 227 rotary positions may be used.

Figure 1.
PILOT Sound

Another useful feature is the synchronization of an audio tape to the computer. One could use the audio portion of a tape to give instructions for a given program or perhaps give a spelling word orally. I see this as a particularly convenient aspect when writing programs for beginning readers or children with learning disabilities. While the audio portion is running, the computer can display information on the monitor simultaneously.

PILOT is an easy to use programming language, one I am sure that most educators will find very useful for curriculum development for their specific classroom needs.

Beatrix Potter On The Computer

Program 3 was written to use in the study of the literature written by Beatrix Potter. This program was used as a biographical introduction and follow-up lesson. With each question, several possible answers were displayed. If students typed in the incorrect answer, they were given some further information and returned to the original question. In this particular program, even wrong answers produced further learning experiences.

This shows a color picture on the TV monitor when a correct answer is typed: a bunny and a duck. To draw these pictures using the turtle, I utilized the instructions which I found in a crafts book for sewing them in cross-stitch embroidery. These instructions come printed on a grid which is similar to the imaginary grid on the computer's video monitor. By placing the turtle at the middle of the embroidery grid, you can have it move the appropriate number of squares, turn, and move again where needed.

With the PILOT cartridge, Atari has provided an exceptionally beautiful teaching guide for children. It is an easy to follow as well as aesthetically attractive manual which all children will enjoy using. Also included is a general manual, as well as a documented demonstration tape.

My fifth-grade students who field-tested Atari PILOT had no difficulty learning quickly to draw pictures, manipulate text, and add sound effects to their programs. I see this language as the first language that beginning programmers should learn. It is motivating and a good basis for learning other computer programming languages.

Program 1.

```
200 *BLOCK
210 ,   R:THIS IS A SAMPLE PROGRAM OF
        BLOCKS STACKED ON ONE ANOTHER
220 ,       GR:GOTO-0,-30;CLEAR
300 ,       GR:PEN RED
310 ,       U:*SQUARE
350 ,       GR:GOTO -0,-14
360 ,       GR:PEN BLUE
370 ,       U:*SQUARE
380 ,       GR:GOTO-0,2
390 ,       GR:PEN YELLOW
400 ,       U:*SQUARE
405 ,       U:*SCALE
410 ,       E:
490 *SQUARE
500 ,       GR:4(DRAW16;TURN90)
550 ,       GR:FILL16
560 ,       E:
```

COMPUTE!'s Listing Conventions

Many programs which are listed in **COMPUTE!** use cursor control keys, color keys, and so forth. We have established a listing convention which we believe eases the task of typing programs in accurately.

Atari Conventions

For the Atari, all the editing and cursor-control characters are spelled out and surrounded by brackets: [CLEAR] for "clear screen." Other characters, such as CTRL-T (the "ball" character) will be listed as the "normal" character, but within brackets: [T]. A series of identical control characters will be indicated by a number within the brackets: [3 DOWN] means type the cursor-down key three times; [12 R] means type CTRL-R twelve times.

Two control characters, [=] and [–] should be shifted. Any reverse field text will be enclosed within vertical lines. (Press the Atari logo key [ꓘ] for each vertical line you see.)

Program 2

```
990 R: THIS IS THE ATARI MUSICAL SCALE
     WITH EACH NOTE HELD FOR 15/60TH
       SECOND
1000 %SCALE
1010 ,     C:#S=#S+1
1020 ,     SO:#S
1030 ,     PA:15
1040 ,     J(#S<31):%SCALE
1050 E:
```

Program 3.

```
10 T:

110 C:#H=1
120 %BORDER
130 C:#H=#H+1
140 T:

150 J(#H<09):%BORDER
160 T:              BEATRIX POTTER

170 T:

180 T:    ONCE upon a time there were fo
ur   little Rabbits, and their names
  \
190 T:  were _____
  \
200 T:                    Flopsy,
               Mopsy,
   \
210 T:          Cotton-tail,
        and Peter.
\
220 T:

230 T:

240 T:

250 PA:100
260 U:%MUSIC
270 R:THIS PROGRAM WAS WRITTEN TO BE USE
D AS AN AUTOBIOGRAPHICAL INTRODUCTION TO
 BEATRIX POTTER AND HER BOOKS.
290 R:  PATRICIA TUBBS, SUNNYVALE, CAIFO
RNIA
300 R: LINE NUMBER 970 SHOULD BE CHANGED
 YEARLY TO ALLOW FOR THE CORRECT    ANSW
ER.
310 GR:QUIT
320 T:What is your name?
330 A:$NAME
340 %FIRST
350 T:Who wrote THE TALE OF PETER RABBIT
?\
360 T: Beatrix Potter
   Mark Twain                          E
zra Keats
370 A:
380 GR:QUIT
390 M:BEATRIX, MARK,EZRA
400 JM:%BEATRIX %MARK %EZRA
410 %MARK
420 T:
430 T:Mark Twain wrote HUCKLEBERRY FINN
and other stories.  Please try again.\
440 J:%FIRST
450 %EZRA
460 T:
470 T:Ezra Keats wrote WHISTLE FOR WILLI
E and other  stories.  Please try again.\

480 J:%FIRST
490 %BEATRIX
500 T:
510 T:Yes, Beatrix Potter wrote THE TALE
 OF PETER RABBIT and 22 more books for c
hildren like you and your friends.\
520 A:
530 U:%BUNNY
540 %SECOND
550 T:Why did Miss Potter write THE TALE
 OF PETER RABBIT?\
560 T:
570 T: just for the fun of it
 for the money                         a
s a get well letter
580 A:
590 GR:QUIT
600 M:JUST,FOR,WELL
610 JM:%JUST %FOR %WELL
620 %JUST
630 T:Yes, Miss Potter did enjoy writing
 stories, but that was not her only reas
on.  Try again.\
640 T:
650 J:%SECOND
660 %FOR
670 T:Miss Potter did not need the money
 because she was the daughter of a wealt
hy lawyer   Please try again.\
680 T:
690 J:%SECOND
700 %WELL
710 T:Yes, she wrote this story as a get
```

```
 well letter for a young friend, 5 year
old Noel.  \
720 T:
730 A:
740 U:*BUNNY
750 *THIRD
760 T:Miss Potter lived in what country?
\
770 T:
780 T:     England
      Canada
   United States
790 A:
800 GR:QUIT
810 M:ENGLAND, CANADA, UNITED STATES
820 JM:*ENG    *CAN   *UNI
830 *CAN
840 T:
850 T:  Canada is an English speaking co
untry but not the home of Miss Potter.\
860 J:*THIRD
870 *UNI
880 T:
890 T: Both Ezra Jack Keats and Mark Twa
in are from the United States but Miss P
otter was not.\
900 J:*THIRD
910 *ENG
920 T:
930 T: Yes, she lived in England and THE
 TALE OF PETER RABBIT first appeared in
print in 1901.  \
940 *NUMBER
950 T:Can you tell your teacher and me h
ow long ago that was?  \
960 A:
970 M:80
980 TN: Try subtracting that one more ti
me.  Remember you put this year's number
 on the top with 1901 undernath.\
990 TY: My, that was a long time ago, wa
sn't it, $NAME ? \
1000 JN:*NUMBER
1010 A:
1020 U:*DUCK
1030 *FOURTH
1040 T:Why were the books published in t
he small size?       \
1050 T:
1060 T:  To fit small hands (or)
      The paper would cost less
1070 A:
1080 GR:QUIT
1090 M:HANDS,PAPER
1100 JM:*HANDS *PAPER
1110 *PAPER
1120 T: You're right, the paper would co
st less, but she had another reason for
wanting the books nice and small.\
1130 J:*FOURTH
1140 *HANDS
1150 T:Miss Potter told her printer that
 the books were to be small enough for l
ittle hands to hold, \
1160 T:and printed on 'stout' paper.  Th
e size she suggested was 5 inches by 3
3/4 inches. \
1170 A:
1180 U:*DUCK
1190 *FIFTH
1200 T:Was Peter Rabbit translated into
any other languages?
1210 A:
1220 GR:QUIT
1230 M:YES
1240 TY:Yes,Peter Rabbit has been printe
d in twelve languages including Afrikaan
s and Japanese as well as in\
1245 TY: Braille.
1260 TN:TRY AGAIN!
1270 JN:*FIFTH
1280 A:
1290 U:*BUNNY
1300 *SIXTH
1310 T:From where did the ideas for the
characters in Beatrix Potter's books com
e? \
1320 T:
1330 T:   other people
      pets and animals she watched
  children
1340 A:
1350 GR:QUIT
1360 M:OTHER,PETS,CHILD
1370 JM:*OTHER *PETS *CHILD
1380 *OTHER
1390 T:No not other people, try one more
 time, please, $NAME
1400 J:*SIXTH
1402 *CHILD
1404 T: Other children did often give he
r ideas; however, her main ideas came ot
her ways.
1406 J:*SIXTH
1410 *PETS
1420 T:Yes, as a child Beatrix owned man
y pets.  She took home wild, ill or hurt
 animals and nursed them back to\
1430 T: health.  Beatrix was the only da
ughter of wealthy parents who did not wa
nt her to go to public or private\
1440 T: school.  She had a tutor at home
```

```
, but no other children to play with, so
 she made friends with animals \
1450 T:  She spent much of her time sket
ching their pictures, which she used yea
rs later in her stories. \
1460 A:
1470 U:*DUCK
1480 U:*MUSIC
1490 U:*BUNNY
1495 T:Now it's time for you to read som
e of Beatrix Potter's books.  I hope you
 enjoy them, $NAME \
1500 E:
1510 *CHILD
1520 T:Sometimes children did give her i
deas.  She wrote many story letter to yo
ung friends including new sotries\
1530 T:expecially written for them.  But
, this was not wher most of her ideas ca
me.  Try again. \
1550 *BUNNY
1560 GR:DRAW8
1570 GR:TURN270;DRAW2
1580 GR:TURN90;DRAW3
1590 GR:TURN270;DRAW2
1600 GR:TURN90;DRAW2
1610 GR:TURN270;DRAW2
1620 GR:TURN90;DRAW3
1630 GR:TURN270;DRAW1
1640 GR:TURN90;DRAW3
1650 GR:TURN90;DRAW2
1660 GR:TURN270;DRAW3
1670 GR:TURN90;DRAW2
1680 GR:TURN 270;DRAW5
1690 GR:TURN270;DRAW2
1700 GR:TURN270;DRAW3
1710 GR:TURN90;DRAW2
1720 GR:TURN270;DRAW3
1730 GR:TURN90;DRAW2
1740 GR:TURN90;DRAW3
1750 GR:TURN270;DRAW2
1760 GR:TURN90;DRAW3
1770 GR:TURN270;DRAW2
1780 GR:TURN270;DRAW6
1790 GR:TURN90;DRAW2
1800 GR:TURN270;DRAW2
1810 GR:TURN45;DRAW3
1820 GR:TURN315;DRAW3
1830 GR:TURN270;DRAW3
1840 GR:TURN90;DRAW2
1850 GR:TURN270;DRAW2
1860 GR:TURN90;DRAW2
1870 GR:TURN90;DRAW2
1880 GR:TURN270;DRAW1
1890 GR:TURN90;DRAW2
1900 GR:TURN270;DRAW1
1910 GR:TURN270;DRAW2
1920 GR:GO2;DRAW2
1930 GR:TURN90;DRAW2
1940 GR:TURN90;DRAW3
1950 GR:TURN270;DRAW1
1960 GR:TURN90;DRAW1
1970 GR:TURN270;DRAW2
1980 GR:TURN270;DRAW1
1990 GR:TURN90;DRAW2
2000 GR:TURN270;DRAW1
2010 GR:TURN90;DRAW1
2020 GR:TURN270;DRAW3
2030 GR:TURN90;DRAW1
2040 GR:TURN270;DRAW9
2050 GR:TURN325;DRAW4
2060 GR:TURN270;DRAW7
2070 GR:PEN UP;DRAW-4
2080 GR:TURN135
2090 C:@710=12%16+3
2100 GR:PEN BLUE;FILL 50
2120 GR:PEN RED ;DRAW2
2130 T:Peter Rabbit really likes you,$NA
ME. \
2140 PA:400
2150 GR:QUIT
2160 E:
2170 *DUCK
2180 GR:PEN ERASE
2190 C:@710=15%16+4
2200 GR:PEN BLUE;TURN 55;DRAW8
2210 GR:TURN-55;DRAW6
2220 GR:TURN45;DRAW8
2230 GR:TURN45;DRAW2
2240 GR:TURN-90;DRAW2
2250 GR:TURN-90;DRAW4
2260 GR:TURN-45;DRAW10
2270 GR:TURN45;DRAW6
2280 GR:TURN-90;DRAW2
2290 GR:TURN90;DRAW5
2300 GR:TURN-90;DRAW2
2310 GR:TURN90;DRAW4
2320 GR:TURN90;DRAW2
2330 GR:TURN-90;DRAW2
2340 GR:TURN-90;DRAW2
2350 GR:TURN-90;DRAW2
2360 GR:TURN90;DRAW4
2370 GR:TURN-45;DRAW6
2380 GR:TURN-45;DRAW4
2390 GR:TURN-90;DRAW2
2400 GR:TURN90;DRAW6
2410 GR:GOTO15,16;PEN RED;DRAW4
2420 GR:TURN90;DRAW2
2430 GR:TURN90;DRAW4
2440 GR:TURN90;DRAW2;TURN180;FILL2
2450 GR:PEN BLUE
2460 GR:GOTO-10,1;TURN180;FILL5
```

```
2470 GR:GOTO-13,2;FILL 5
2480 GR:GOTO-10,-1;FILL5
2490 GR:GOTO-9,6;FILl 4
2500 GR:GOTO-3,8;FILL2
2510 GR:GOTO4,6;FILL2;GOTO6,8;FILL2
2520 GR:GOTO4,10;FILL2
2530 GR:GOTO6,12;FILL2
2540 GR:GOTO8,14;FILL2
2550 GR:GOTO10,16;FILL2
2560 GR:GOTO-15,7;FILL2
2570 GR:PEN RED;GOTO-17,-2;4(TURN90:DRAW
2);TURN180:FILl 2
2580 GR:GOTO-10,-4;4(TURN90;DRAW2);TURN9
0;FILL2
2590 GR:GOTO-15,-6;4(TURN90;DRAW2);TURN9
0;FILL2
2600 GR:GOTO-4,-2:4(TURN90:DRAW2);TURN18
0;FILL2
2610 GR:GOTO1,-6;4(TURN90:DRAW2);TURN90;
FILL2
2620 GR:GOTO2,-4;4(TURN90:DRAW2);TURN90;
FILL2
2630 GR:GOTO12,12;PEN YELLOW;DRAW6
2640 GR:TURN-90;DRAW4;TURN-45;DRAW4
2650 GR:TURN-115;DRAW6
2660 GR:GOTO0,-8;FILL4
2670 T:    Jemima Puddle-Duck knows you h
ave been workine hard, $NAME    \
2680 PA:400
2690 GR:QUIT
2700 E:
2710 *MUSIC
2730 C:#M=#M+1
2740 SO:20
2760 PA:60
2770 PA:60
2780 SO:13
2790 PA:60
2800 SO:17
2810 PA:60
2820 SO:6
2830 PA:60
2840 SO:15 5
2850 PA:60
2860 SO:13 3
2870 PA:60
2880 SO:17 1
2890 PA:120
2900 JK#M(2):*MUSIC
2910 SO:
2920 E:
```

©

Moire Magic

Leo Cerruti
Syosset, NY

This program produces such beautiful three-dimensional effects in any of four graphics modes that you will, in all likelihood, find yourself staring at the screen and shouting "Hey look at that one..." for hours on end.

This program is user controlled – you can request the density and type of patterns you like. Making use of the "attract mode" produces constant variations in color and intensity and you will never see the same pattern twice. There are two pattern options to choose from: center oriented and corner oriented. That is, lines will emit from the center of the screen or from the corners, as you wish.

Useful Subroutines

This program makes use of several subroutines which should prove useful in many other types of programs, both graphic and nongraphic. For instance; the subroutine from lines 1060 to 1075 causes the word START to flash within the sentence. Line 1070 demonstrates use of the START switch, line 1080 sets attract mode.

When the program is run, it will prompt you to press START to begin. After this it will ask you which graphics mode you wish, modes 3,5,7 or 8. Remember that you need at least 16K to operate in GR.8. The next prompt asks how much maximum spacing you want between the lines drawn on the screen. I suggest numbers between 25 and 40 for GR.8 and 15 to 25 for GR.7. The computer will randomly space the lines up to your maximum. It will never give a spacing of two or less, to prevent filling up the screen with lines or blanks.

You are then asked which pattern you want. Center patterns give a cartwheel moire effect while corner patterns give more of a three-dimensional effect similar to depth lines drawn on a flat plane to give the illusion of distance. Either choice has its own special effects which demonstrate the Atari's superior graphics and color capabilities. You are then asked if you want changing colors. If you type "Y," the "attract mode" is set. Then, happy viewing.

By the way, if you select attract mode do not touch any keys because this will set the "attract mode" counter back to default colors. After each complete cycle in both the center and corner patterns the drawing will pause, select a new line spacing, and then continue. The longer you let the drawing continue, the more complex it will become. You can press CTRL 1 to pause the drawing at anytime, and to continue, press CTRL 1 again. Pressing the BREAK key will end the drawing.

Program Modules

LINE #	DESCRIPTION
0-4	Displays title and jumps to line 1060 which instructs user to push START to begin.
6-9	Requests which graphics mode to use and jumps to lines 1000 to 1030 to set screen margins for the appropriate mode.
10	Requests maximum number of spaces to use between each line.
12-14	Requests corner or center effect moire patterns. Program jumps to line 16 for center pattern or line 600 for corner patterns.
1050	Will set attract mode if you wish.
1060-1090	Will display instruction to push START button and wait for you to do so. Type the word START and a space before and after an inverse video in line 1065. Use normal video for the word START and the spaces before and after in line 1075. This will flash the word START within the sentence.

```
0 ? "}":POSITION 9,8:? "***  MOIRE PATTE
RNS  ***"
1 POSITION 9,10:? "***  BY LEO CERRUTI
***"
2 ? :?
3 DIM C$(1)
4 GOSUB 1060
6 POKE 752,0:? "}":? "WHICH GRAPHICS MOD
```

```
E DO YOU WISH:"
7 ? ,"(1) MODE 3":? ,"(2) MODE 5":? ,"(3
) MODE 7":? ,"(4) MODE 8"
8 TRAP 8:INPUT GM:IF GM<1 OR GM>4 THEN 6
9 ON GM GOSUB 1000,1010,1020,1030
10 ? :? "HOW MUCH SPACING MAXIMUM ":INPU
T SPACE
12 ? :? "WHICH MOIRE PATTERN:":? ,"(1) F
ROM CENTER":? ,"(2) FROM CORNERS":INPUT
P
14 ON P GOTO 16,600
15 REM *** CENTER PATTERN ***
16 GOSUB 1050
20 GRAPHICS GR+16:COLOR 1
25 GOSUB 500
30 FOR A=0 TO Y STEP S
40 GOSUB 900:DRAWTO X,A
50 NEXT A
60 FOR B=X TO 0 STEP -S
70 GOSUB 900:DRAWTO B,Y
80 NEXT B
90 FOR C=Y TO 0 STEP -S
100 GOSUB 900:DRAWTO 0,C
110 NEXT C
120 FOR D=0 TO X STEP S
130 GOSUB 900:DRAWTO D,0
140 NEXT D
145 GOSUB 500
147 GOSUB 910
150 FOR E=0 TO Y STEP S
160 GOSUB 900:DRAWTO X,E:COLOR 0
170 NEXT E
180 FOR F=X TO 0 STEP -S
190 GOSUB 900:DRAWTO F,Y
200 NEXT F
210 FOR G=Y TO 0 STEP -S
220 GOSUB 900:DRAWTO 0,G
230 NEXT G
240 FOR H=0 TO X STEP S
250 GOSUB 900:DRAWTO H,0
260 NEXT H
265 GOSUB 910
270 COLOR 1:GOTO 25
299 REM
300 REM ***   RANDOM SPACING    ***
310 REM *** MAXIMUM DETERMINED ***
320 REM ***      BY USER        ***
321 REM
500 S=INT(RND(1)*SPACE)+2:RETURN
549 REM
550 REM *** CORNER PATTERNS ***
551 REM
600 GOSUB 1050
620 GRAPHICS GR+16:COLOR 1
625 GOSUB 500
630 FOR A=0 TO Y STEP S
640 PLOT 0,0:DRAWTO X,A
650 NEXT A
660 FOR B=X TO 0 STEP -S
670 PLOT 0,0:DRAWTO B,Y
680 NEXT B
685 GOSUB 500
690 FOR C=0 TO X STEP S
700 PLOT 0,Y:DRAWTO C,0:COLOR 0
710 NEXT C
720 FOR D=0 TO Y STEP S
730 PLOT O,Y:DRAWTO X,D
740 NEXT D
745 GOSUB 500
750 FOR E=Y TO 0 STEP -S
760 PLOT X,Y:DRAWTO 0,E:COLOR 1
770 NEXT E
780 FOR F=0 TO X STEP S
790 PLOT X,Y:DRAWTO F,0
800 NEXT F
805 GOSUB 500
810 FOR G=X TO 0 STEP -S
820 PLOT X,0:DRAWTO G,Y:COLOR 0
830 NEXT G
840 FOR H=Y TO 0 STEP -S
850 PLOT X,0:DRAWTO 0,H
860 NEXT H
870 COLOR 1:GOSUB 910:GOTO 625
900 PLOT C1,C2:RETURN
904 REM
905 REM *** PAUSE BETWEEN CYCLES ***
906 REM
910 FOR TIME=1 TO 400:NEXT TIME:RETURN
949 REM
950 REM *** GRAPHICS LIMITS ***
951 REM
1000 GR=3:X=39:Y=23:C1=19:C2=11:RETURN
1010 GR=5:X=79:Y=47:C1=39:C2=23:RETURN
1020 GR=7:X=159:Y=95:C1=79:C2=47:RETURN
1030 GR=8:X=319:Y=191:C1=159:C2=95:RETUR
N
1040 REM *** SET ATTRACT MODE ***
1041 REM ***      AND START       ***
1042 REM
1050 ? :? "DO YOU WANT CHANGING COLORS (
Y OR N)":INPUT C$
1055 ? :? "HIT [CTRL 1] TO FREEZE OR R
ELEASE  PATTERN  AT ANY TIME"
1060 C=PEEK(84)
1063 POKE 752,1
1065 FOR P=1 TO 50:NEXT P:POKE 84,C:? "
     HIT [START] TO BEGIN"
1070 IF PEEK(53279)=6 THEN 1080
1075 FOR P=1 TO 50:NEXT P:POKE 84,C:? "
     HIT  START  TO BEGIN":GOTO 1065
1080 IF C$="Y" THEN POKE 77,128
1090 RETURN
```

©

Put A Rainbow In Your Atari

Fred and Doug Tedsen
Sonoma, CA

You've probably seen programs that display 128 colors on the Atari. They are usually interesting to look at, but what do you do with them after you have run them two or three times? Well here is a program that displays a moving rainbow of all 128 colors, and the techniques could easily be used for dramatic title screens in your own programs.

The program begins by drawing the word COLOR in large block letters on the GRAPHICS 7 screen. This is performed by the subroutine at lines 1000 to 1200. While the letters are being drawn, the program is doing a graphics fill. The letter outline is drawn with color register 1 (controlled by SETCOLOR 1) and the inside area of the letters is filled with color register 0 (COLOR 1, SETCOLOR 0). Lines 2000 through 2500 contain the data points for drawing the letters.

The program now goes through a color changing sequence (lines 110-190). This section was included to demonstrate how colors can be controlled inside and outside of the fill areas and also to heighten anticipation for the part that follows. First we randomly change the colors inside the letters, leaving the background black. Next the background colors is changed while the letters remain black. And, finally, we change both the letter and background colors independently. Notice that the letter outlines remain white throughout.

Now The Fun Part

Now we come to the fun part. At the beginning of the program, the subroutine at lines 3000 to 3040 was run to load the machine language color rainbow generator into the strings CUP$ and CDOWN$. These are now used to produce the rainbow pattern. The pattern is first set moving up the screen within the letters, with a black background. The pattern is then put on the background with solid color letters. These steps are then repeated with movement down the screen, just to show that we can go both ways.

So how does it work? Briefly, the POTO register is read and the value obtained placed in one of the playfield color registers. Since the pots are continually counting down to zero, this value changes every scan line. A write to WSYNC makes the change occur at the end of a scan line, resulting in solid lines across the screen. Movement is accomplished by adding or subtracting the value of the 1/60th of a second frame counter to the POTO value before writing it to the color register. The write is directly to the registers in CTIA because the OS shadow registers are not copied until vertical blank and therefore would do nothing. The assembler source listing is included for reference. Notice that there is a direct correspondence between the source listing and the BASIC data statements at lines 3100 to 3280.

Modifications

There are several things which you can do with this routine to change the display:

1. Parameter two in the USR statement is the time in seconds that the routine is to be run. Thus X=USR(ADR(CUP$),4) will display the pattern moving up the screen for about four seconds.

2. You can affect any of the five playfield color registers. To do this you can change either the DATA statement at line 3180 or the machine language string. The values to use are 22, 23, 24, 25, and 26 for color registers 0, 1, 2, 3, and the background, respectively. For example, changing line 3180 to DATA 141,26,208 will affect the background. Line 240 demonstrates how the string may be changed to give the same result.

3. You can obtain a stationary rainbow pattern by changing line 3170 to DATA 234,234 (NOP's). Don't try to use CDOWN$ if you do this, however. A better way would be to change elements 12 and 13 of the string.

4. For those of you with Assembler, there is a myraid of patterns which can be generated by using AND and ORA before writing to the color register. You can create patterns with large bars of color, with small bars of different shades of the same color, or with some combinations – a rainbow of pastels.

5. While the machine language routine is running, your program can't do anything else. Though we haven't tried it yet, it should be possible to incorporate the logic in a display list interrupt routine. This would allow you to do things such as having the pattern roll down the screen with a curtain effect.

There are a couple of things to watch out for when running the program. A time value of zero will run the rainbow routine for about four minutes. If you accidently do this, you will have to press SYSTEM RESET to get out. Also, unplug your paddles from game port 1 to get the proper rainbow effect. You might want to plug a paddle in later to see the effect. It's kind of interesting.

```
1 REM COLOR RAINBOW
2 REM FRED AND DOUG TEDSEN, OCT 1981
10 DIM D$(3),CUP$(32),CDOWN$(32)
20 NS=4:NT=15
30 GOSUB 3000
40 GOSUB 1000
50 FOR I=1 TO 1000:NEXT I
100 N=0
110 SETCOLOR 0,INT(RND(0)*16),2*INT(RND(0)*8)
120 FOR I=1 TO 150:NEXT I:N=N+1:IF N<NT THEN GOTO 110
130 SETCOLOR 0,0,0:N=0
140 C=INT(RND(0)*16):I=INT(RND(0)*8)*2:SETCOLOR 2,C,I:SETCOLOR 4,C,I
150 FOR I=1 TO 150:NEXT I:N=N+1:IF N<NT THEN GOTO 140
160 N=0
170 C=INT(RND(0)*16):I=INT(RND(0)*8)*2:SETCOLOR 2,C,I:SETCOLOR 4,C,I:SETCOLOR 0,INT(RND(0)*16),2*INT(RND(0)*8)
180 FOR I=1 TO 120:NEXT I:N=N+1:IF N<NT THEN GOTO 170
190 SETCOLOR 2,0,0:SETCOLOR 4,0,0
210 SETCOLOR 0,12,6
220 CUP$(15,15)=CHR$(22)
230 X=USR(ADR(CUP$),NS)
240 CUP$(15,15)=CHR$(26)
250 X=USR(ADR(CUP$),NS)
260 SETCOLOR 0,3,4
270 CDOWN$(15,15)=CHR$(22)
280 X=USR(ADR(CDOWN$),NS)
290 CDOWN$(15,15)=CHR$(26)
300 X=USR(ADR(CDOWN$),NS)
310 SETCOLOR 0,7,2
400 FOR I=1 TO 1000:NEXT I:GOTO 40
1000 GRAPHICS 7+16
1010 SETCOLOR 0,0,0:SETCOLOR 1,0,14:SETCOLOR 2,0,0:SETCOLOR 4,0,0
1020 COLOR 2:FCOLOR=1
1030 RESTORE 2010
1100 READ D$:IF ASC(D$)<64 THEN GOTO 1180
1110 IF D$="P" THEN READ ROW,COLUMN:GOSUB 1200:PLOT COLUMN,ROW:GOTO 1100
1120 IF D$="O" THEN READ RORIGIN,CORIGIN:GOTO 1100
1130 IF D$="END" THEN RETURN
1140 IF D$<>"F" THEN GOTO 1100
1150 READ ROW,COLUMN:GOSUB 1200:POSITION COLUMN,ROW:POKE 765,FCOLOR
1160 XIO 18,#6,0,0,"S:":PLOT COLUMN,ROW:GOTO 1100
1180 ROW=VAL(D$):READ COLUMN:GOSUB 1200:DRAWTO COLUMN,ROW:GOTO 1100
1200 ROW=ROW+RORIGIN:COLUMN=COLUMN+CORIG
```

```
IN:RETURN
2000 REM "C"
2010 DATA 0,10,2
2020 DATA P,1,9,1,19,3,23,5,25,9,27,15,2
7,15,18,F,12,18,F,10,16,10,12,12,10,29,1
0,31,12
2030 DATA P,26,18,26,27,32,27,36,25,38,2
3,40,19,40,9,F,38,5,F,36,3,F,32,1,F,9,1
2040 DATA F,5,3,F,3,5,F,1,9,P,31,12,F,31
,16,F,29,18,F,26,18
2100 REM "O"
2110 DATA 0,18,32
2120 DATA P,1,9,1,19,3,23,5,25,9,27,32,2
7,36,25,38,23,40,19,40,9
2130 DATA P,10,12,F,10,16,F,12,18,F,29,1
8,F,31,16,31,12,29,10,12,10,10,12
2140 DATA P,40,9,F,38,5,F,36,3,F,32,1,F,
9,1,F,5,3,F,3,5,F,1,9
2200 REM "L"
2210 DATA 0,26,62
2220 DATA P,1,1,1,10,32,10,32,27,40,27,4
0,1,F,1,1
2300 REM "O"
2310 DATA 0,34,92
2320 DATA P,1,9,1,19,3,23,5,25,9,27,32,2
7,36,25,38,23,40,19,40,9
2330 DATA P,10,12,F,10,16,F,12,18,F,29,1
8,F,31,16,31,12,29,10,12,10,10,12
2340 DATA P,40,9,F,38,5,F,36,3,F,32,1,F,
9,1,F,5,3,F,3,5,F,1,9
2400 REM "R"
2410 DATA 0,42,122
2420 DATA P,1,1,1,19,3,23,5,25,9,27,15,2
7,19,25,21,23,22,20,40,27
2430 DATA 40,18,F,21,10,40,10,40,1,P,9,1
0,F,9,16,F,11,18,F,13,18,F,15,16,15,10,9
,10,P,40,1,F,1,1
2500 DATA END
3000 RESTORE 3100
3005 FOR I=1 TO 32
3010 READ C:CUP$(I)=CHR$(C)
3020 NEXT I
3030 CDOWN$=CUP$:CDOWN$(12,12)=CHR$(229)
3040 RETURN
3100 DATA 104
3110 DATA 104
3120 DATA 104
3130 DATA 72
3140 DATA 162,57
3150 DATA 160,0
3160 DATA 173,0,210
3170 DATA 101,20
```

```
3180 DATA 141,22,208
3190 DATA 141,10,212
3200 DATA 136
3210 DATA 208,242
3220 DATA 202
3230 DATA 208,237
3240 DATA 104
3250 DATA 56
3260 DATA 233,1
3270 DATA 208,228
3280 DATA 96
```

```
0100;  ;    RAINBOW COLOR GENERATOR
0110   ;    Fred and Doug Tedsen
0120 ;
0130   RTCLOK3  =     $14
0140   COLPF0   #     $D016
0150   POT0     =     $D200
0160   WSYNC    =     $D40A
0170   ;
0180            PLA              Throw out no. arguments
0190            PLA              Throw out high order byte
0200            PLA              Get no. seconds to run
0210   LOOPA    PHA              Push on stack
0220            LDX   #57        57x256 is about 1 second
0230   LOOPB    LDY   #0
0240   LOOPC    LDA   POT0       Read Pot 0,
0250            ADC   RTCLOK3    add value od 1/60 timer,
0260            STA   COLPF0     and put result in color register 0.
0270            STA   WSYNC      Wait for end of scan line.
0280            DEY
0290            BNE   LOOPC
0300            DEX
0310            BNE   LOOPB
0320            PLA              Get second counter from stack.
0330            SEC              Subtract 1 from counter
0340            SBC   #1         and branch until zero.
0350            BNE   LOOPA
0360            RTS
```

©

THE OSI® GAZETTE

Part II:

A Small Operating System: OS65D, The Disk Routines

T. R. Berger
Coon Rapids, MN

Editor's Note: Part I appeared last month. Here, the author presents a map of the disk routines. — RTM

Let's turn to track zero. Exactly one ms. after the index hole a two byte address is recorded on the disk in high byte-low order. This address is read by the ROM on boot. It is the start address for loading track zero into memory. Next comes the number of pages in track zero. Finally, that many pages of data are written on the track. There are no track start or stop markings. After track zero is loaded, the computer always jumps to $2200. Hopefully, track zero has been loaded in that vicinity. It would appear that OSI did not think the track zero format over very carefully.

Subroutine Descriptions

Most of the disk routines are self-explanatory. Because these routines are far more involved than those in the kernel, many more flow charts are needed. Let's run through the memory map in order, commenting on special properties of certain subroutines.

The timing routines at $2678, $267A, and $26A2 are independent of the system clock. The wait time in the routines at $2700, $289F, and $28A4 should be divided by *T* if the system clock is T MHZ.

OS65D does not use binary track numbers, but BASIC does. Thus BASIC uses $26A6, but OS65D enters this routine at $26BC with the BCD track number in the accumulator. With a binary track number in the accumulator, this routine may be entered at $26A9. It will move the disk head over the correct track after some error checking.

The sequence beginning at $2728 may be viewed as the standard startup to read or write a track or sector. It puts the head on the disk, finds the index hole, then initializes the disk data ACIA.

The EXAMINE command uses $2739 to load the entire contents of a track into memory without regard to error checking, track formatting, or sectoring. This type of command is only possible with the asynchronous data format used by OSI. If you crash a track, this command can prove invaluable in retrieving what may remain. I view this routine as a utility. It should reside on the disk and not in memory, unless needed. The initialize routine at $2768 used on a full disk falls in the same category. Such programs as these should be transient, i.e. only called when needed.

The major "Save a Sector" routine begins at $27D7. It uses the data in $265E-$2661. Most of OS65D's disk data is stored in page zero. Because Zpage is swapped out when BASIC comes in, the most important data is repeated in $265C-$2662. BASIC passes its values to these latter locations. LOAD and SAVE routines must then move this data to Zpage. Since OS65D can put information directly into Zpage, it puts the save vector into $FE, $FF directly, entering the Save routine at $27E1. Except when SAVE or CALL are used, all saving is done in Sector one for 12 ($OC) pages on 8" floppies and for eight pages on minifloppies. After a write, the sector is reread and compared with memory. If the comparison fails, the sector is reread again. This may occur up to four times. If comparison still fails, another attempt is made to write the sector. If comparison fails after four rereads again, the operation is aborted with Error #2. To my recollection, I've never seen Error #2 occur. It might happen on an old worn disk, on a midnight special, or with a very dirty head.

The major "Read a Sector" routine is $295D. It uses data in $265-E-$2662. Again OS65D may enter this routine at $2967 if the load vector at $FE, $FF has been set. This program tries to read a sector seven times. The only error check (other than sector seek errors which abort immediately) is a parity check for each byte. If, after seven tries, a read still fails, then the head is moved down then up one track. This whole process may be repeated up to four times before Error #1 is reported. This error also seems to be very rare.

Both read and save routines use the sector seeking routine at $28C4 which, in turn, calls $2998.

Further, they both use a dual purpose routine at $2905. If the accumulator is zero on entry, this routine reads to memory. If it is nonzero, then the routine compares with memory. The actual read and compare loops within this routine are separate. With 8" floppies and a 1 MHZ clock, the 6502 is not fast enough to get from one disk byte to the next if the read and compare loops are combined into one. As it stands, the compare loop just barely returns in time for the next comparison. With a 2 MHZ clock there is plenty of time.

I view the sector directory routines at $29F3 and $2A41 as utilities. They do not need to be resident in memory.

Machine language routines may access the disk directly. For example, to write a sector, locations $265E-$2662 should be assigned correct values. The following segment of code will write a sector to the disk.

```
10 JSR $26A6  ;Move head to track
20 JSR $2754  ;Engage head, find start of track
30 JSR $27D7  ;Write sector
40 JSR $2761  ;Disengage head
50 RTS
```

If the write address is already in $FE, $FF then $27D7 may be entered at $27E1. In this case, lines 20-40 may be replaced by JSR $2CA7, a kernel routine.

To read a sector, again assign correct values to $265E-$2662 then perform the following.

```
10 JSR $26A6  ;Move head to track
20 JSR $2754  ;Engage head, find start of track
30 JSR $295D  ;Read sector
40 JSR $2761  ;Disengage head
50 RTS
```

If the read address is already in $FE, $FF then $295D may be entered at $2967. In this case, lines 20-40 may be replaced by the kernel routine:

```
JSR $2B1A
```

When we discuss the I/O section of OS65D we will see additional ways to read from and write to the disk.

References:

1. Jefferson Harman, "IBM Compatible Disk Drives", Byte October 1979, p. 100
2. Ira Rampil, "A Floppy Disk Tutorial", Byte December 1977, p. 24
3. Les Solomon, "BASICS of Computer Disk Systems", Popular Electronics November 1980, p. 53

MAP – OS65D DISK HANDLER

DISK-MEMORY DATA

265C	DRIVE NUMBER
265D	CURRENT BCD TRACK NUMBER
265E	SECTOR NUMBER
265F	PAGE LENGTH OF SECTOR
2660	LOW BYTE LOAD/SAVE VECTOR
2661	HIGH BYTE LOAD/SAVE VECTOR
2662	BINARY TRACK NUMBER

DISK-Z PAGE

E5	LAST TRACK OF FILE BEING HANDLED
F6	NUMBER OF RETRIES ON WRITE
F7	NUMBER OF HEAD MOVE RETRIES ON READ
F8	NUMBER OR READ RETRIES BEFORE HEAD MOVE
F9	SECTOR COUNT
FA	TARGET TRACK NUMBER ON SEEK
FB	SECTOR NUMBER READ ON DISK
FC	STACK POINTER (IN $29F3)
FD	SECTOR PAGE COUNT (IN $27D7)
FE	SYSTEM POINTER. USED AS
FF	LOAD AND SAVE VECTOR BY DISK

Subroutines – OS65D Disk Handler

2663	Home the Disk. Move the disk head to track 0.
2678	Wait 12 ms.
267A	Wait X ms.
2683	Step up one track toward track 76.
268A	Step down one track toward track 0.
26A2	Wait 8 ms.
26A6	Fetch binary track number from 2662 then:
26A9	Convert track number to BCD then:
26BC	Check for track 0-76 BCD, check for drive ready, move disk head to track, adjust head current, and if an error occurs, abort and send an error message via 2A4B.
2700	Wait 20Y + 7 microseconds (1 MHZ clock).
2708	Adjust head current.
271D	Find trailing edge of index hole.
2728	Engage head then:
272B	Find index hole then:
272E	Initialize disk ACIA.
2739	Engage head, read from index hole full around to index hole, then quit.
2754	Head down.
2761	Head up.
2768	Initialize full disk.
277D	Initialize one track.
27C2	Send a byte to the disk.
27CD	Fetch a byte from the disk.
27D7	Fetch sector save vectors then:
27E1	Save a sector.
289F	Wait 800($FA) microseconds.
28A4	Wait 100Y microseconds.
28B0	Fetch a byte from the disk. Abort with an error message if over the index hole.
28C4	Find the end of the sector preceding the one in 265E.
2905	Read a sector to or compare a sector with memory.
295D	Fetch disk read vector then:
2967	Read and reread a sector to memory, quit if successful or the full number of retries are exhausted.
2998	Find the end of the present sector.
29C6	Select the drive in 265C then:
29DA	Check if the drive is ready.
29EB	8 drive select data bytes.
29F3	Output a sector directory.
2A41	Output subroutine for 29F3.

A Correction For Progressive Computing Chess 1.9

Dave Leskin
Calgary, Canada

Progressive Computing, based in Windsor Ontario, is an excellent source of OSI software with prompt and courteous service; however, there is a major error in their tape version of "Chess 1.9". This error is found in the opening tables. If you try the following sequence of moves you can determine if your copy of "Chess 1.9" has this error too. Note that the last move by the computer is illegal. Microchess notation in brackets.

	Computer (White)	Human (Black)
1	P-K4 (13-33)	P-K4 (63-43)
2	N-KB3 (01-22)	N-QB3 (76-55)
3	B-QN5 (02-46)	N-KB3 (71-52)
4	B-KN5 (05-41) ???	

As you see the "B-KN5" jumps right over the Queen Pawn at 14. To solve this problem I changed the program so that the Queen Knight was moved from 06 to 25 instead. This results in a "Four Knights Game" which is a common opening used by many players. Just follow the steps listed below to effect the change.

The Changes

1. Load "Chess 1.9"

2. Press "D" to enter monitor once the board appears.

3. Press ".0B34"

4. Press "/" to enter the data mode

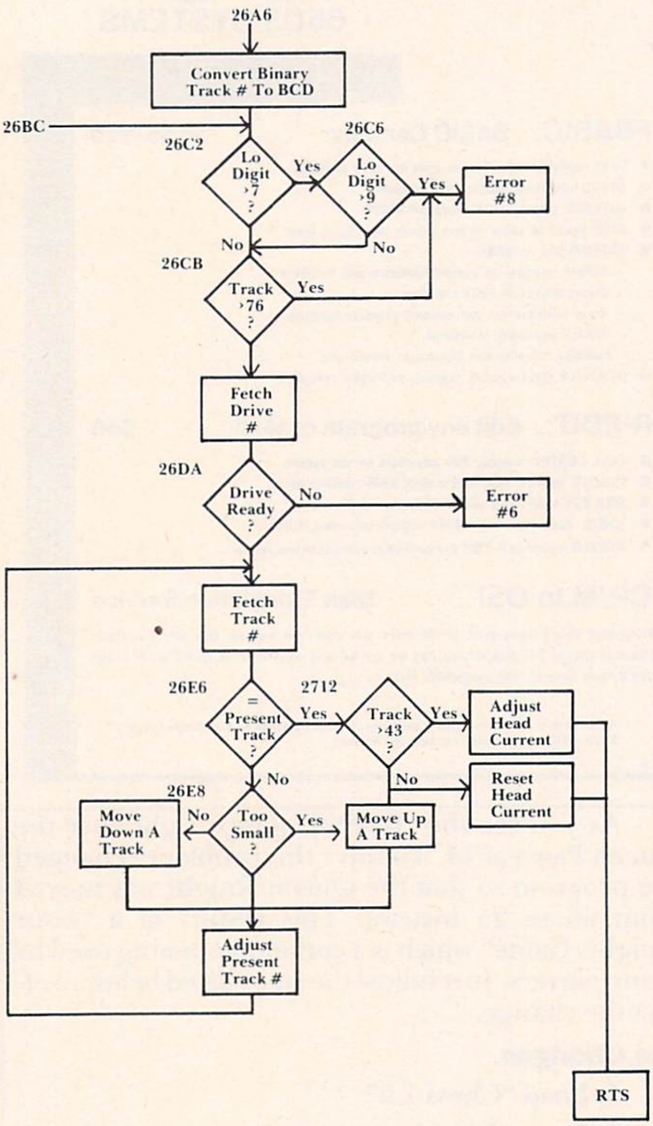
5. Press "25" which refers to the square that the piece will be moved to

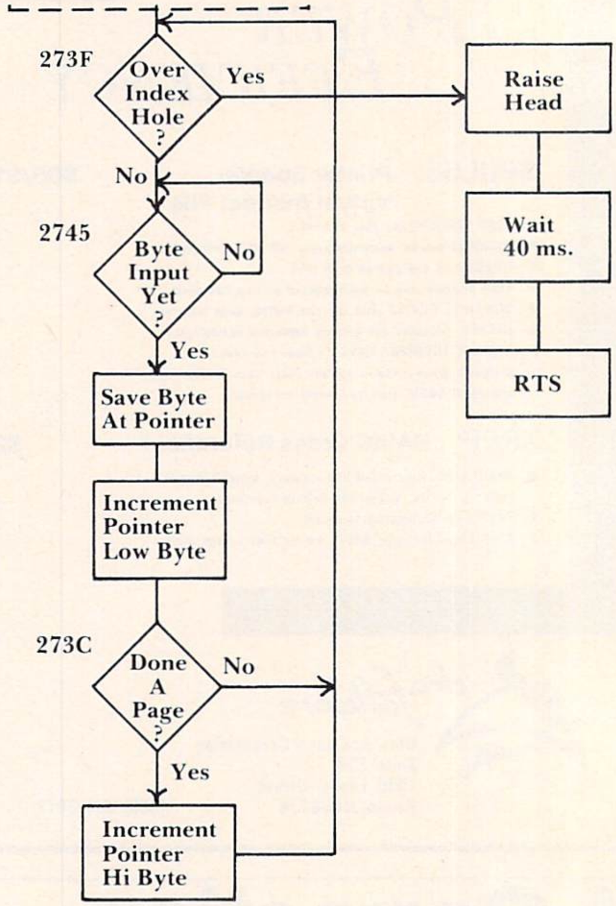
6. Press the "RETURN" key

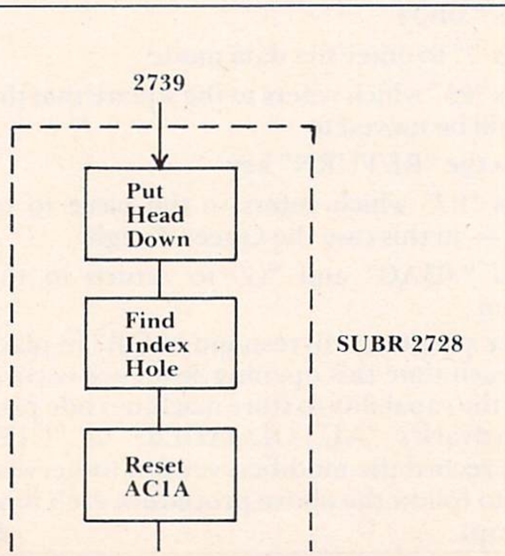
7. Press "07" which refers to the piece to be moved — in this case the Queen Knight

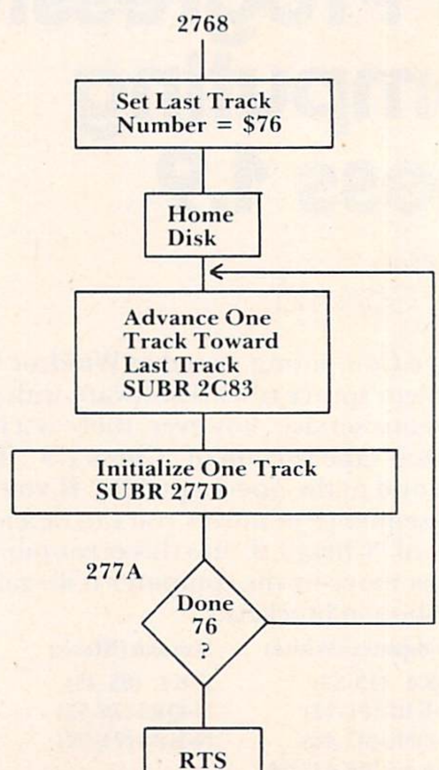
8. Press ".03AC" and "G" to return to the program

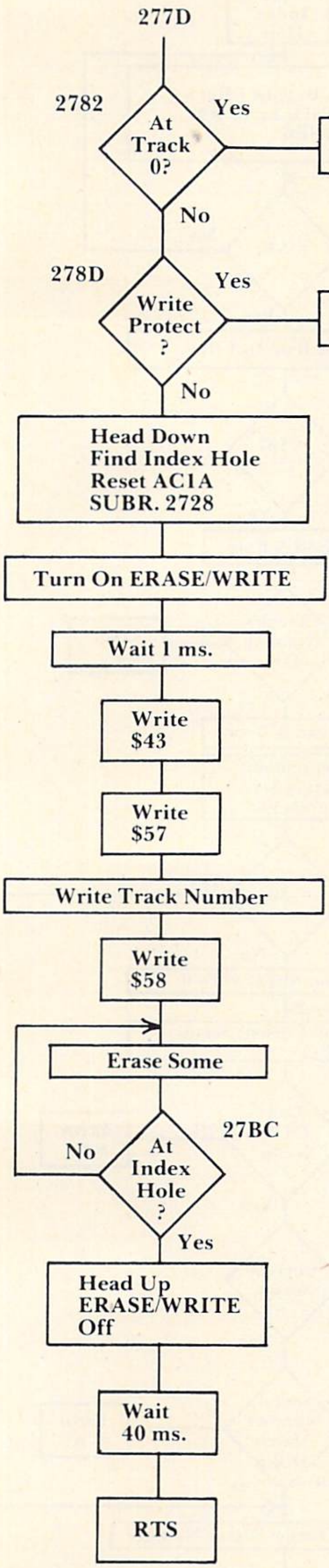
Now the program will respond N-QB3 in place of B-KN5 each time this opening sequence occurs. If you have the capability to store machine code programs (Aardvark's "AUTOLOADER" or "C1E" ROM) then record the modified version (otherwise you'll have to follow the above procedure each time you power up). ©

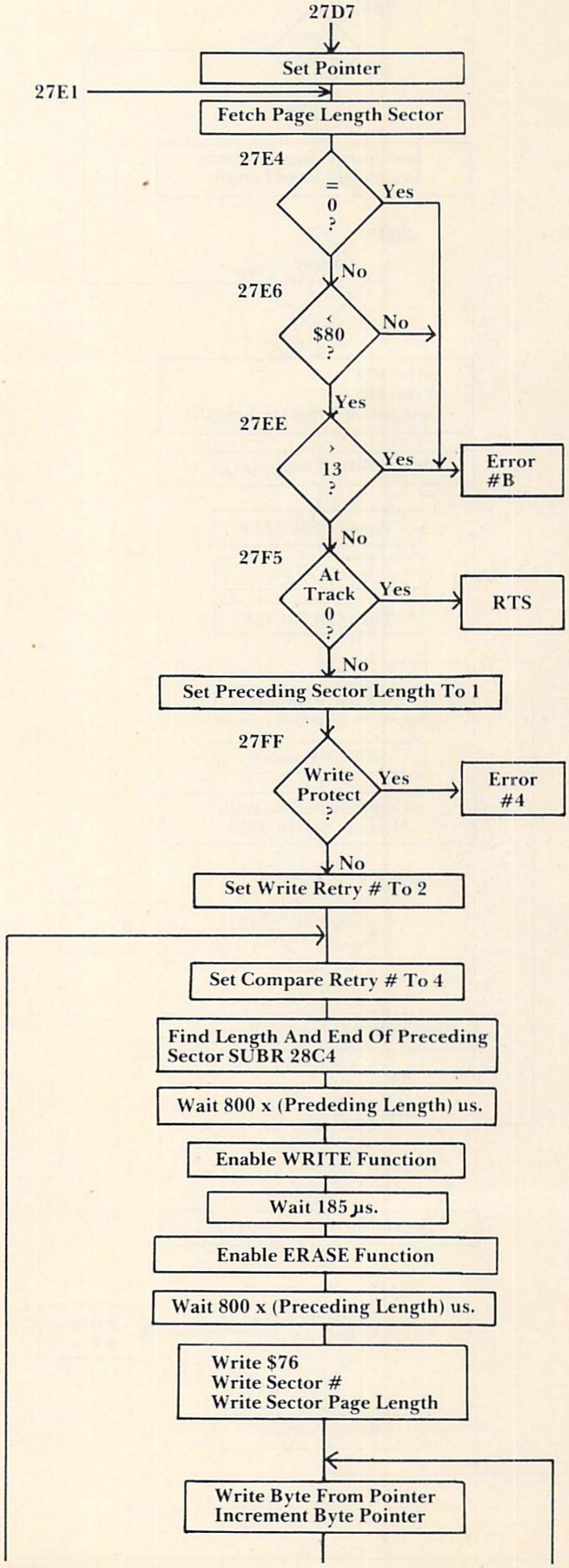

26A6
Convert Binary Track # To BCD
26BC
26C2
Lo Digit >7 ?
Yes
26C6
Lo Digit >9 ?
Yes
Error #8
No
No
26CB
Track >76 ?
Yes
Fetch Drive #
26DA
Drive Ready ?
No
Error #6
Fetch Track #
26E6
= Present Track ?
Yes
2712
Track >43 ?
Yes
Adjust Head Current
No
Reset Head Current
No
26E8
No
Too Small ?
Yes
Move Down A Track
Move Up A Track
Adjust Present Track #
RTS

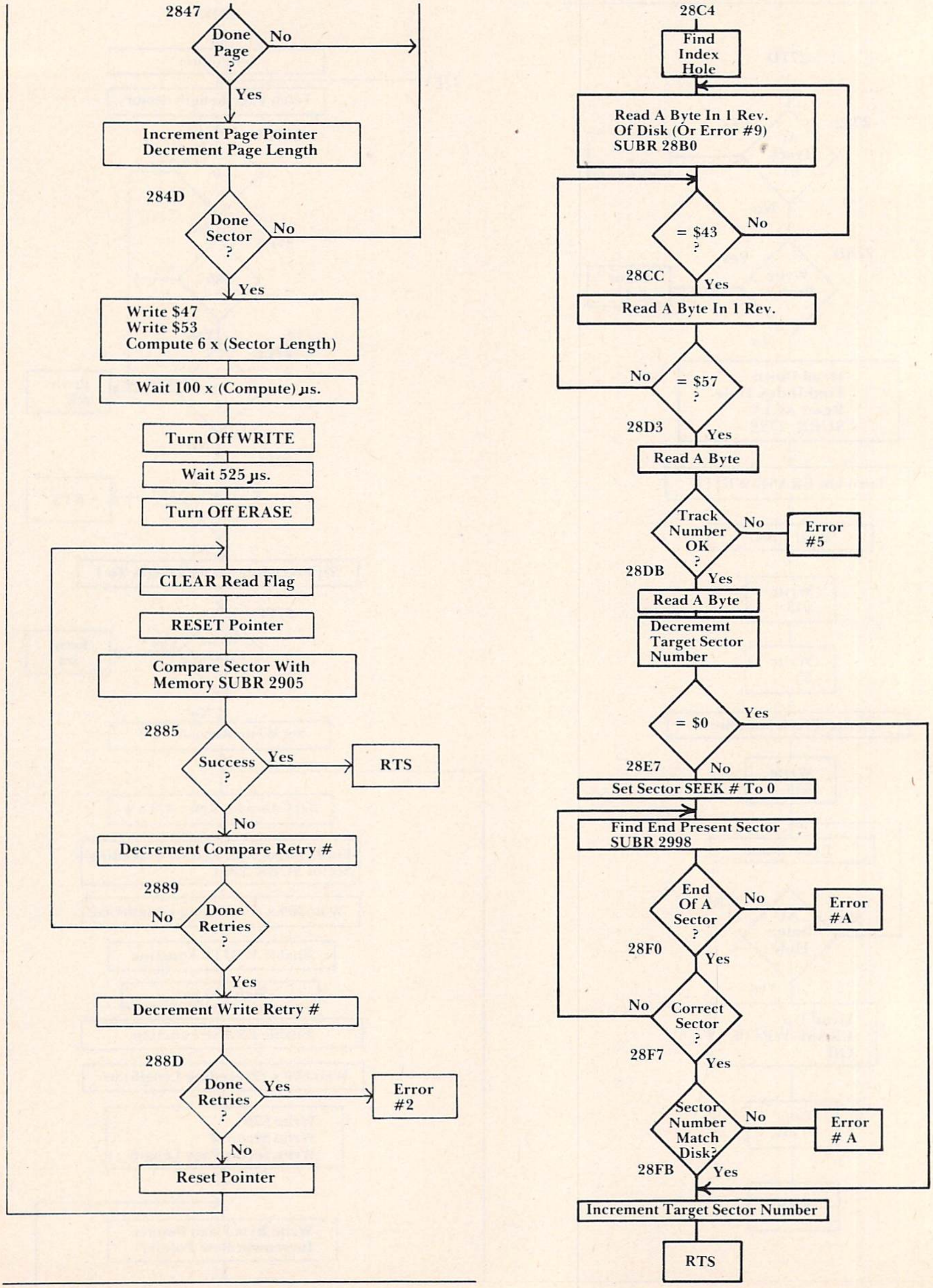

273F
Over Index Hole ?
Yes
Raise Head
Wait 40 ms.
RTS
No
2745
Byte Input Yet ?
No
Yes
Save Byte At Pointer
Increment Pointer Low Byte
273C
Done A Page ?
No
Yes
Increment Pointer Hi Byte

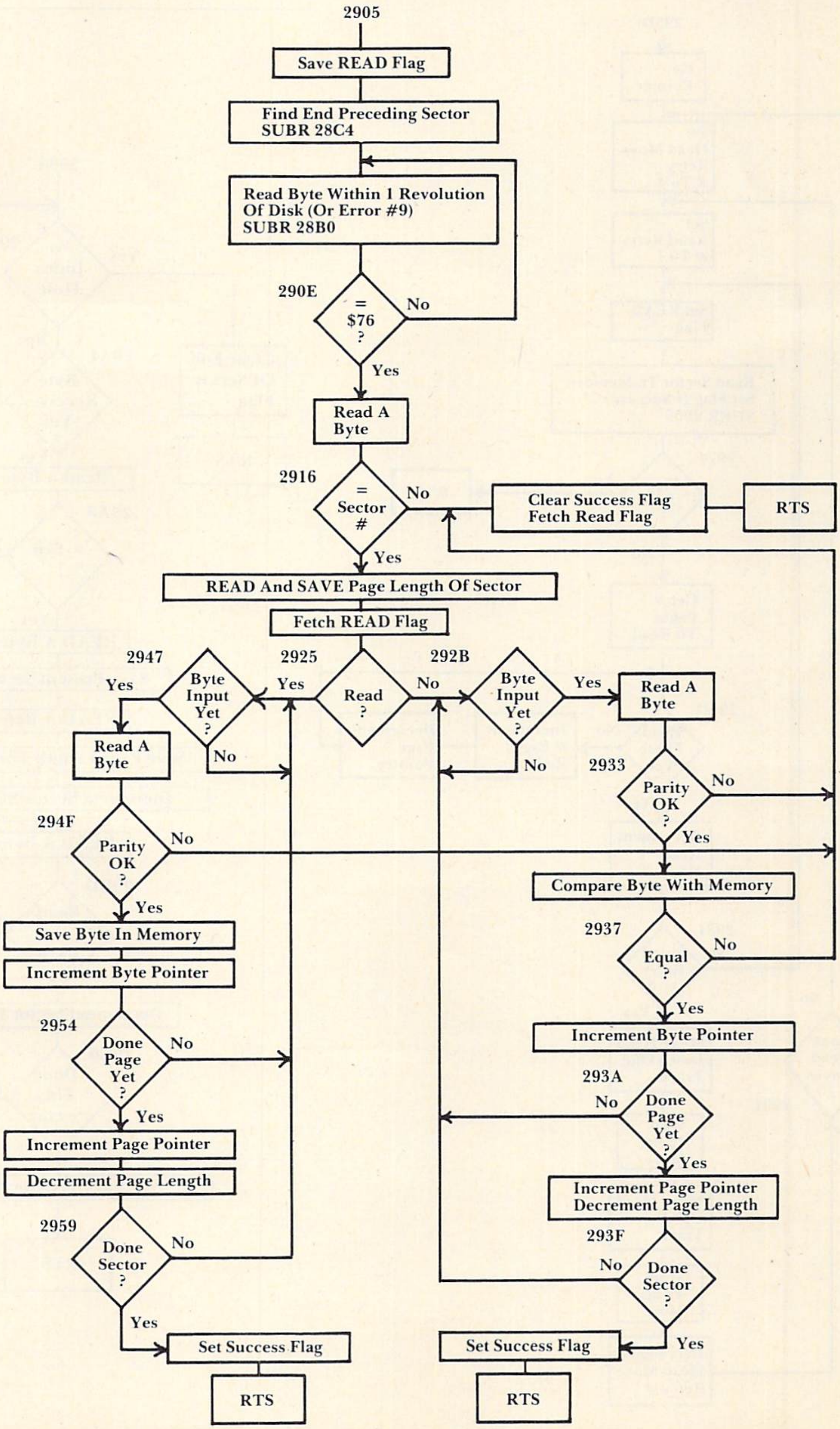

2739
Put Head Down
Find Index Hole
SUBR 2728
Reset ACIA

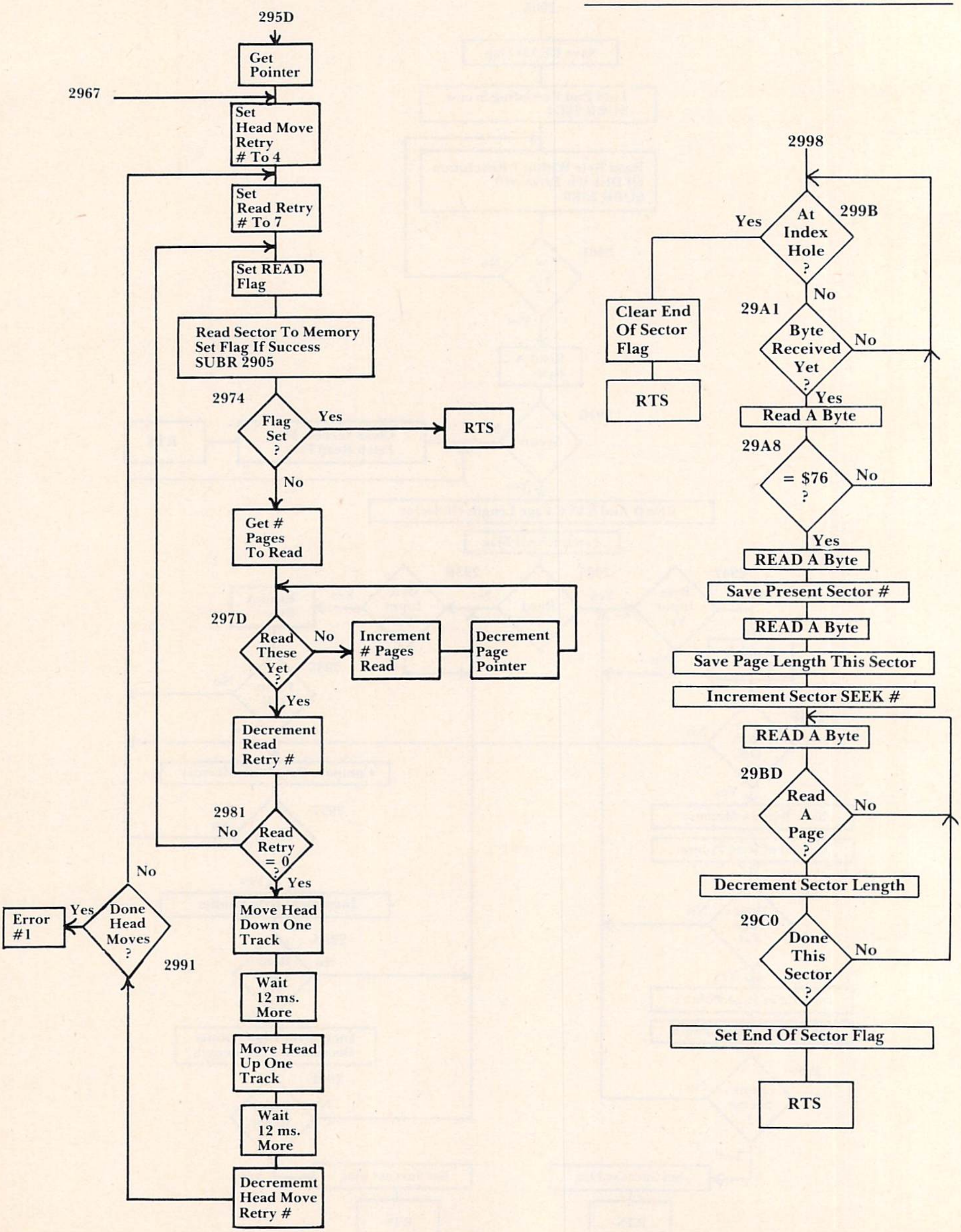

2768
Set Last Track Number = $76
Home Disk
Advance One Track Toward Last Track SUBR 2C83
Initialize One Track SUBR 277D
277A
Done 76 ?
RTS

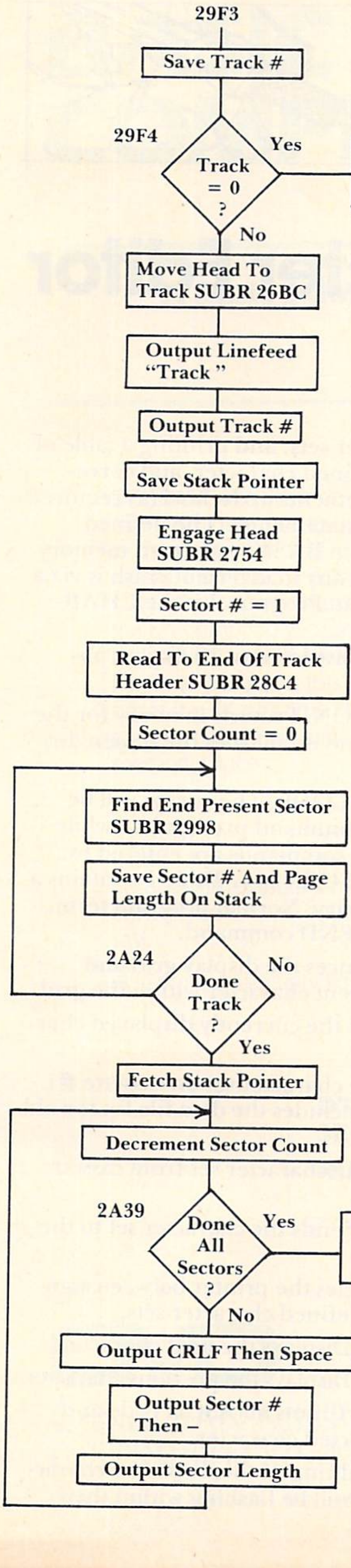

277D
2782
At Track 0?
Yes
Error # 3
No
278D
Write Protect ?
Yes
Error # 4
No
Head Down
Find Index Hole
Reset ACIA
SUBR. 2728
Turn On ERASE/WRITE
Wait 1 ms.
Write $43
Write $57
Write Track Number
Write $58
Erase Some
27BC
No
At Index Hole ?
Yes
Head Up
ERASE/WRITE
Off
Wait 40 ms.
RTS

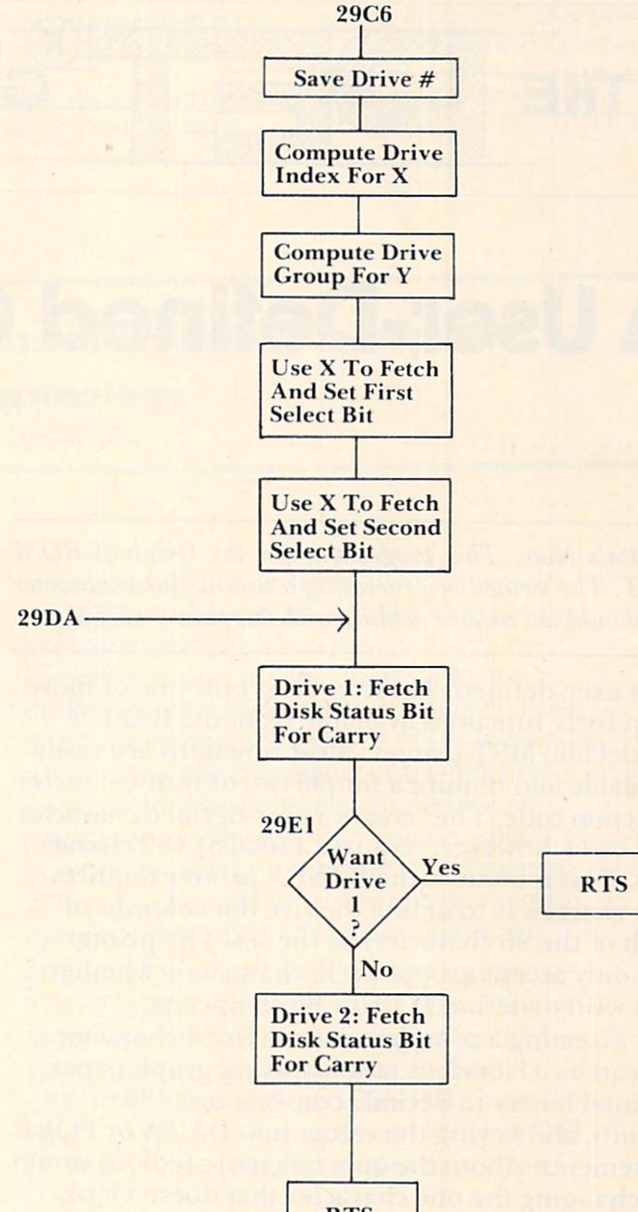

27D7
Set Pointer
27E1
Fetch Page Length Sector
27E4
= 0 ?
Yes
No
27E6
< $80 ?
No
Yes
27EE
> 13 ?
Yes
Error #B
No
27F5
At Track 0 ?
Yes
RTS
No
Set Preceding Sector Length To 1
27FF
Write Protect ?
Yes
Error #4
No
Set Write Retry # To 2
Set Compare Retry # To 4
Find Length And End Of Preceding
Sector SUBR 28C4
Wait 800 x (Prededing Length) us.
Enable WRITE Function
Wait 185 µs.
Enable ERASE Function
Wait 800 x (Preceding Length) us.
Write $76
Write Sector #
Write Sector Page Length
Write Byte From Pointer
Increment Byte Pointer


2847
Done Page ?
No
Yes
Increment Page Pointer
Decrement Page Length
284D
Done Sector ?
No
Yes
Write $47
Write $53
Compute 6 x (Sector Length)
Wait 100 x (Compute) µs.
Turn Off WRITE
Wait 525 µs.
Turn Off ERASE
CLEAR Read Flag
RESET Pointer
Compare Sector With Memory SUBR 2905
2885
Success ?
Yes
RTS
No
Decrement Compare Retry #
2889
Done Retries ?
No
Yes
Decrement Write Retry #
288D
Done Retries ?
Yes
Error #2
No
Reset Pointer
28C4
Find Index Hole
Read A Byte In 1 Rev. Of Disk (Or Error #9) SUBR 28B0
= $43 ?
No
28CC
Yes
Read A Byte In 1 Rev.
= $57 ?
No
28D3
Yes
Read A Byte
Track Number OK ?
No
Error #5
28DB
Yes
Read A Byte
Decrememt Target Sector Number
= $0
Yes
28E7
No
Set Sector SEEK # To 0
Find End Present Sector SUBR 2998
End Of A Sector ?
No
Error #A
28F0
Yes
Correct Sector ?
No
28F7
Yes
Sector Number Match Disk?
No
Error # A
28FB
Yes
Increment Target Sector Number
RTS


2905
Save READ Flag
Find End Preceding Sector
SUBR 28C4
Read Byte Within 1 Revolution
Of Disk (Or Error #9)
SUBR 28B0
290E
= $76 ?
No
Yes
Read A Byte
2916
= Sector # ?
No
Clear Success Flag
Fetch Read Flag
RTS
Yes
READ And SAVE Page Length Of Sector
Fetch READ Flag
2947
Byte Input Yet ?
Yes
No
2925
Read ?
Yes
No
292B
Byte Input Yet ?
Yes
No
Read A Byte
2933
Parity OK ?
No
Yes
Compare Byte With Memory
2937
Equal ?
No
Yes
Increment Byte Pointer
293A
Done Page Yet ?
No
Yes
Increment Page Pointer
Decrement Page Length
293F
Done Sector ?
No
Yes
Set Success Flag
RTS
Read A Byte
294F
Parity OK ?
No
Yes
Save Byte In Memory
Increment Byte Pointer
2954
Done Page Yet ?
No
Yes
Increment Page Pointer
Decrement Page Length
2959
Done Sector ?
No
Yes
Set Success Flag
RTS


295D
Get Pointer
2967
Set Head Move Retry # To 4
Set Read Retry # To 7
Set READ Flag
Read Sector To Memory Set Flag If Success SUBR 2905
2974
Flag Set ?
Yes
RTS
No
Get # Pages To Read
297D
Read These Yet ?
No
Increment # Pages Read
Decrement Page Pointer
Yes
Decrement Read Retry #
2981
No
Read Retry = 0 ?
Yes
Move Head Down One Track
Wait 12 ms. More
Move Head Up One Track
Wait 12 ms. More
Decrememt Head Move Retry #
No
Error #1
Yes
Done Head Moves ?
2991
2998
Yes
At Index Hole ?
299B
Clear End Of Sector Flag
RTS
No
29A1
Byte Received Yet ?
No
Yes
Read A Byte
29A8
= $76 ?
No
Yes
READ A Byte
Save Present Sector #
READ A Byte
Save Page Length This Sector
Increment Sector SEEK #
READ A Byte
29BD
Read A Page ?
No
Decrement Sector Length
29C0
Done This Sector ?
No
Set End Of Sector Flag
RTS


29F3
Save Track #
29F4
Track = 0 ?
Yes
Clear Carry
RTS
No
Move Head To Track SUBR 26BC
Output Linefeed "Track"
Output Track #
Save Stack Pointer
Engage Head SUBR 2754
Sector # = 1
Read To End Of Track Header SUBR 28C4
Sector Count = 0
Find End Present Sector SUBR 2998
Save Sector # And Page Length On Stack
2A24
Done Track ?
No
Yes
Fetch Stack Pointer
Decrement Sector Count
2A39
Done All Sectors ?
Yes
Reset Stack Pointer
RTS
No
Output CRLF Then Space
Output Sector # Then –
Output Sector Length


29C6
Save Drive #
Compute Drive Index For X
Compute Drive Group For Y
Use X To Fetch And Set First Select Bit
Use X To Fetch And Set Second Select Bit
29DA
Drive 1: Fetch Disk Status Bit For Carry
29E1
Want Drive 1 ?
Yes
RTS
No
Drive 2: Fetch Disk Status Bit For Carry
RTS

©

character grid while in this mode. Table 2 contains the subcommands of the EDIT mode. A raster (*) is the only exit from this mode.

GENERAL INPUT. The CURSOR input routine. It is called by COMMAND DECODE and INPUT CODE.

TEST. Prints the entire character set on the printer.

TABLE. Prints a table in the format of Table 3.

END. Closes all devices and clears the screen. Can be called at any time without disturbing the new character set.

It is possible to create a reasonable facsimile of the PET's shifted keyboard graphics with the 5 by 7 BASE 2 print matrix. (The PET screen matrix is 8 by 8.) DATA statements for this pseudo-PET character set are included in Program 2. Printing all characters displayable on the PET screen would require 3 user defined character sets on the BASE 2. An easier way of implementing "full" pseudo-PET graphics is via the graphics function of the BASE 2.

Standard and user-defined characters can be displayed on the same line, but doing so requires two passes of the print head; i.e., normal print followed by carriage return without linefeed and user-defined print. This is not difficult to arrange during formatted printing from a program. Listing a program is more complicated. One approach is to list to the screen, sort and count characters, and then use one of the many screen printing routines previously published in **COMPUTE!** and other magazines.

While on the subject of listings, it should be mentioned that the normal list sequence of:

OPEN5,4:CMD5:LIST

will not work for long listings, probably due to some bug in the timing when CMD is invoked. The following sequence, using the terminal buffer feature of the BASE 2 (run in either immediate or program mode) will work:

OPEN5,4
PRINT#5,CHR$(27);CHR$(82);CHR$(20);CHR$(80);
PRINT#5,CHR$(27);CHR$(54);
CMD5:LIST (or LIST XXX- in program mode)
PRINT#5,CHR$(27);CHR$(83):CLOSE5

Whether your application is mathematics, foreign languages, APL, or whatever, design your own character set with ease using CHARACTER EDITOR.

Table 1.

COMMANDS (all followed by RETURN)

I(RETURN)##	**Input code and display character**
+	**Display next character**
–	**Display previous character**
E	**Edit. (See Table 2)**
L	**Load character set from tape**
P	**Download Character Set To Printer**
S	**Save Character Set to Tape**
T	**Toggle printer**
TABLE	**Print table in format of Table 3**
TEST	**Test print entire character set**
END	**Terminate program**

Table 2.

EDIT MODE SUBCOMMANDS

›	**Move cursor to next grid point**
‹	**Move cursor to previous grid point**
↑	**Move cursor up one row**
←	**Move cursor down one row**
SPACE key	**Erase matrix point and move to next grid point**
shifted &	**Insert matrix point at grid point**
C	**Clear character grid**
*****	**Store displayed character in memory and return to COMMAND DECODE mode**

Table 3.

```
CHR$( 32)   " "   0    0    0    0    0
CHR$( 33)   "▌"   127  127  127  0    0
CHR$( 34)   "▄"   120  120  120  120  120
CHR$( 35)   "‾"   1    1    1    1    1
CHR$( 36)   "_"   64   64   64   64   64
CHR$( 37)   "| "  127  0    0    0    0
CHR$( 38)   "▒"   85   42   85   42   85
CHR$( 39)   " |"  0    0    0    0    127
CHR$( 40)   "░"   80   40   80   40   80
CHR$( 41)   "◤"   127  63   15   3    1
CHR$( 42)   "▐"   0    0    0    127  127
```

Program 1.

```
100 REM PET/BASE TWO PROGRAMMABLE ¬
    ¬CHARACTER EDITOR
105 REM
110 REM BY P. J. ROVERO
115 REM    NOCC COMNAVMAR BOX 2
120 REM    FPO S. F.        96630
125 REM
130 REM THIS PROGRAM ENABLES THE USER ¬
    ¬TO EASILY BUILD, STORE, AND
135 REM EDIT CHARACTERS IN THE FORMAT ¬
    ¬USED BY THE BASE TWO MODEL
140 REM 800 MST PRINTER.
145 REM
150 REM THE VARIABLE BA SHOULD BE ¬
    ¬CHANGED TO SUIT THE SYSTEM.
155 REM BA= START ADDRESS OF 482 BYTES ¬
    ¬OF BASIC-PROTECTED
160 REM MEMORY REQUIRED FOR USER ¬
    ¬CHARACTER TABLE.
165 REM
170 REM COMMAND SUMMARY:
175 REM
180 REM S  SAVE CHARACTER SET TO TAPE#1
185 REM L  LOAD CHARACTER SET FM TAPE#1
190 REM P  DOWNLOAD CHARACTER SET TO ¬
    ¬PRINTER MEMORY
195 REM T  TOGGLE PRINTER BETWEEN ¬
```

```
    ¬CHARACTER SETS
200 REM TEST  PRINT USER DEFINED ¬
    ¬CHARACTER SET
205 REM TABLE  PRINT A TABLE IN FORMAT ¬
    ¬CHR$(X)  "CHARACTER"  DATA STREAM
210 REM +  DISPLAY NEXT CHARACTER
215 REM -  DISPLAY PREVIOUS CHARACTER
220 REM I  INPUT CHARACTER CODE AND ¬
    ¬DISPLAY CHARACTER
225 REM E  EDIT CHARACTER DISPLAYED ON ¬
    ¬SCREEN.  SUBCOMMANDS IN THIS
230 REM    MODE ARE >,<,^,^ FOR CURSOR  ¬
    ¬CONTROL AND * TO ENTER
235 REM    DISPLAYED CHARACTER INTO ¬
    ¬MEMORY.
240 REM END  CLOSE ALL DEVICES AND END
245 REM******
250 REM        INITIALIZATION ROUTINE
255 REM******
260 POKE134,ØØ:POKE135,6Ø:REM PROTECT ¬
    ¬TOP 1K OF MEMORY
265 PRINT"ĥINITIALIZING VARIABLES"
270 TG=77:CU=32849:BA=15361
275 DIM SL%(7),SH%(7),CC%(5),LC%(7)
280 CR$=CHR$(13)
285 FORI=ØTO4:CC%(I)=1+I*3:NEXT
290 FORI=ØTO6:LC%(I)=1+I*3:NEXT
295 FORI=ØTO6:READ SL%(I):NEXT
300 DATA8Ø,2ØØ,64,184,48,168,Ø32
305 FORI=ØTO6:READ SH%(I):NEXT
310 DATA128,128,129,129,130,130,131
315 CO%=32:POKEBA,27:POKE(BA+1),75
320 REM******
325 REM         COMMAND DECODE ROUTINE
330 REM******
335 PRINT"ĥ"
340 GOSUB415:REM INITIAL DISPLAY
345 GOSUB1165:REM  INPUT
350 IFIN$="S"THENGOTO65Ø
355 IFIN$="L"THENGOTO765
360 IFIN$="P"THENGOSUB835
365 IFIN$="T"THENGOSUB89Ø
370 IFIN$="TEST"THENGOSUB126Ø
375 IFIN$="+"THENGOSUB93Ø
380 IFIN$="-"THENGOSUB965
385 IFIN$="I"THENGOSUB1ØØØ
390 IFIN$="E"THENGOSUB1Ø6Ø
395 IFIN$="TABLE"THENGOSUB1315
400 IFIN$="END"THENGOTO13Ø5
405 GOTO345
410 REM******
415 REM        INITIAL DISPLAY
420 REM******
425 A$="O##":B$="O#P":C$="L$$":D$="L$:"
430 E$="%  ":F$="% '"
435 A1$=A$+A$+A$+A$+B$
440 A2$=E$+E$+E$+E$+F$
445 A3$=C$+C$+C$+C$+D$
450 PRINT"ĥ"
455 FORI=ØTO6
460 PRINTA1$:PRINTA2$:PRINTA2$
465 NEXT
470 PRINTA3$
475 GOTO345
480 REM******
485 REM        DISPLAY SR
490 REM******
495 BT=BA+2+((CO%-32)*5)
500 FORM=BTTO(BT+4)
505 C2=CU+((M-BT)*3)
510 CH%=PEEK(M)
515 FORN=ØTO6
520 C3=C2+(N*12Ø)
525 POKEC3,42
530 IFCH%AND(2^N)THENPOKEC3,1Ø2:GOTO54Ø
535 POKEC3,32
540 NEXTN
545 NEXTM
550 POKE224,184:POKE225,129:POKE226,19:
    ¬POKE245,11:POKE5,19
555 PRINT"            ←←←←←←←←←←←←";"CHR$(";
    ¬CO%;")";
560 RETURN
565 REM******
570 REM         RASTER SR
575 REM******
580 FORJ=ØTO4
585 CH%=Ø:SPOT=CU+(J*3)
590 FORK=ØTO6
595 IFPEEK(SPOT)=102ORPEEK(SPOT)=23ØTHEN
    ¬CH%=CH%+(2^K)
600 SPØT=SPOT+12Ø
605 NEXTK
610 BT=BA+2+((CO%-32)*5)+J
615 POKEBT,CH%
620 NEXTJ
625 GOSUB485
630 RETURN
635 REM******
640 REM         SAVE SR
645 REM******
650 PRINT"ĥ"
655 INPUT"CHARACTER FILENAME";A$
660 POKE243,122:POKE244,2
665 OPEN1,1,2,A$
670 M=Ø
675 FORN=(BA)TO(BA+481)
680 CI%=PEEK(N)
685 PRINT#1,CI%;
690 PRINTCI%;
695 M=M+1:IFM=191THENGOSUB725
700 PRINT#1,CHR$(13);
705 M=M+1:IFM=191THENGOSUB725
710 NEXT
715 PRINT"ĥ":CLOSE1:GOTO415
720 REM******
725 REM        FORCE INTER-RECORD GAP
730 REM******
735 POKE59411,53
740 T1=TI
745 IF(TI-T1)<2ØGOTO745
750 POKE59411,61:M=Ø
755 RETURN
760 REM******
765 REM         LOAD SR
770 REM******
775 PRINT"ĥ"
780 INPUT"CHARACTER FILENAME";A$
785 OPEN1,1,Ø,A$
790 FORN=ØTO481
795 INPUT#1,CI%
```

```
800 IFST=-128GOTO82Ø
805 POKE(BA+N),CI%
81Ø PRINTCI%;
815 NEXT
82Ø CLOSE1
825 PRINT"ĥ":GOTO415
83Ø REM******
835 REM       DOWNLOAD CHARSET
84Ø REM******
845 OPEN5,4
85Ø FORM=BATO(BA+481)
855 CH%=PEEK(M)
86Ø PRINT#5,CHR$(CH%);
865 NEXTM
87Ø PRINT#5:PRINT#5,CHR$(27);CHR$(76)
875 CLOSE5
88Ø RETURN
885 REM******
89Ø REM       TOGGLE SR
895 REM******
9ØØ IFTG=77THENTG=76:GOTO91Ø
9Ø5 IFTG=76THENTG=77
91Ø OPEN5,4:PRINT#5,CHR$(27);CHR$(TG)
915 CLOSE5
92Ø RETURN
925 REM******
93Ø REM       INCREMENT SR
935 REM******
94Ø CO%=CO%+1
945 IFCO%>127THENCO%=32
95Ø GOSUB485
955 RETURN
96Ø REM******
965 REM       DECREMENT SR
97Ø REM******
975 CO%=CO%-1
98Ø IFCO%<32THENCO%=127
985 GOSUB485
99Ø RETURN
995 REM******
1ØØØ REM       INPUT CO% SR
1ØØ5 REM******
1Ø1Ø GOSUB1165
1Ø15 IN=VAL(IN$)
1Ø2Ø IFIN<32THENIN=32
1Ø25 IFIN>127THENIN=127
1Ø3Ø CO%=INT(IN)
1Ø35 GOSUB485
1Ø4Ø RETURN
1Ø45 REM******
1Ø5Ø REM       EDIT SR
1Ø55 REM******
1Ø6Ø POKE224,Ø8Ø:POKE225,128:POKE226,1:
     ¬POKE245,1
1Ø65 N=Ø:A$=""
1Ø7Ø POKE548,Ø
1Ø75 GETA$:IFA$=""GOTO1Ø75
1Ø8Ø IFA$=">"THENN=N+1:GOTO112Ø
1Ø85 IFA$="<"THENN=N-1:GOTO112Ø
1Ø9Ø IFA$="^"THENN=N-5:GOTO112Ø
1Ø95 IFA$="^"THENN=N+5:GOTO112Ø
11ØØ IFA$="&"THENPRINT"&<";:N=N+1:
     ¬GOTO112Ø
11Ø5 IFA$=" "THENPRINT" <";:N=N+1:
     ¬GOTO112Ø
111Ø IFA$="*"THENPOKE548,1:GOSUB57Ø:
     ¬GOTO1155
1115 IFA$="C"THENPOKE548,1:GOSUB12Ø5:
     ¬GOSUB485:GOTO1Ø6Ø
112Ø POKE514,Ø:WAIT514,6
1125 IFN<ØORN>34THENN=Ø
113Ø NC=N:NL=INT((N/5))
1135 IFNC>4THENNC=NC-5:GOTO1135
114Ø POKE224,(SL%(NL)):POKE225,(SH%(NL))
     ¬:POKE226,(CC%(NC)):POKE245,
     ¬(LC%(NL))
1145 SPOT=256*SH%(NL)+SL%(NL)+CC%(NC)
115Ø GOTO1Ø7Ø
1155 RETURN
116Ø REM******
1165 REM       GENERAL INPUT SR
117Ø REM******
1175 POKE224,24:POKE225,129:POKE226,19:
     ¬POKE245,7:POKEØØ5,19
118Ø PRINT"          <<<<<<<<<<CMD?";
1185 IN$=" ":ZT=TI:ZC=2:ZD$=CHR$(2Ø)
119Ø GETZ$:IFZ$<>""THENGOTO122Ø
1195 IFZT<TITHENPRINTMID$(" &",ZC,
     ¬1);"<";:ZC=3-ZC:ZT=TI+1Ø
12ØØ GOTO119Ø
12Ø5 BT=BA+2+(5*(CO%-32))
121Ø FORI=BTTO(BT+4):POKEI,Ø:NEXTI
1215 RETURN
122Ø Z=ASC(Z$):ZL=LEN(IN$):IF(ZAND127)<3
     ¬2THENPRINT"<";:GOTO1235
1225 IFZL>254THENGOTO119Ø
123Ø IN$=IN$+Z$:PRINTZ$;ZD$;Z$;
1235 IFZ=13THENIN$=MID$(IN$,2):PRINTCR$;
     ¬:RETURN
124Ø IFZ=2ØANDZL>1THENIN$=LEFT$(IN$,
     ¬ZL-1):PRINT"<";:GOTO119Ø
1245 IFZ=141THENZ$=CHR$(-2Ø*(ZL-1)):
     ¬FORZ=2TOZL:PRINTZ$;:NEXTZ:GOTO1165
125Ø GOTO119Ø
1255 REM******
126Ø REM       TEST SR
1265 REM******
127Ø OPEN5,4:PRINT#5,CHR$(27);CHR$(5Ø)
1275 FORN=32TO127:PRINT#5,CHR$(N);:NEXT
128Ø PRINT#5:CLOSE5
1285 RETURN
129Ø REM******
1295 REM       END
13ØØ REM******
13Ø5 PRINT"ĥ":CLOSE1:CLOSE5:END
131Ø REM******
1315 REM       TABLE SR
132Ø REM******
1325 OPEN5,4
1327 PRINT#5,CHR$(27);CHR$(1Ø6);
133Ø FORI=ØTO95
1335 PRINT#5,CHR$(27);CHR$(77);
134Ø PRINT#5,"CHR$(";I+32;")     ";CHR$(34
     ¬);CHR$(13);
1345 PRINT#5,CHR$(27);CHR$(76);"          ¬
     ¬        ";CHR$(I+32);CHR$(13);
135Ø PRINT#5,CHR$(27);CHR$(77);"          ¬
     ¬        ";CHR$(34);"     ";
1355 FORK=ØTO4
136Ø J=BA+2+(I*5)+K:PRINT#5,PEEK(J);"     ¬
     ¬";
1365 NEXTK
```

```
1370 PRINT#5
1375 NEXTI
1380 CLOSE5:RETURN
READY.
```

Program 2. PSEUDO-PET CHARACTER SET

```
115 DATA27,75,0,0,0,0,0,127,127,127,0,0,120,120,120,120,120,1,1,1,1,1
120 DATA64,64,64,64,64,127,0,0,0,0,85,42,85,42,85,0,0,0,0,127,80,40,80,40,80
125 DATA127,63,15,3,1,0,0,0,127,127,0,0,127,8,8,0,0,120,120,120,0,0,15,8,8
130 DATA8,8,120,0,0,96,96,96,96,96,0,0,120,8,8,8,8,15,8,8,8,8,120,8,8
135 DATA8,8,127,0,0,127,127,0,0,0,127,127,127,0,0,0,0,127,127,127
140 DATA3,3,3,3,3,7,7,7,7,7,112,112,112,112,112,64,64,64,64,127
145 DATA120,120,120,0,0,0,0,15,15,15,8,8,15,0,0,15,15,15,0,0,15,15,127,120,120
150 DATA8,8,8,8,8,28,14,127,14,28,0,127,127,0,0,24,24,24,24,24,12,12,12,12,12
155 DATA2,2,2,2,2,48,48,48,48,48,0,127,127,0,0,0,127,127,0,8,8,112,0,0,0,0,7
160 DATA8,8,8,8,7,0,0,127,64,64,64,64,3,4,8,16,96,96,16,8,4,3,127,1,1,1,1
165 DATA1,1,1,1,127,62,127,127,127,62,32,32,32,32,32,12,30,60,30,12
170 DATA0,127,0,0,0,0,0,112,8,8,99,20,8,20,99,62,65,65,65,62,28,10,127,10,28
175 DATA0,0,0,127,0,12,30,63,30,12,8,8,127,8,8,85,42,85,0,0,0,0,127,127,0
180 DATA4,126,2,126,3,1,7,15,63,127
185 DATA0,0,0,0,127,127,127,0,0,120,120,120,120,120,1,1,1,1,1
190 DATA64,64,64,64,64,127,0,0,0,0,85,42,85,42,85,0,0,0,0,127,80,40,80,40,80
195 DATA127,63,15,3,1,0,0,0,127,127,0,0,127,8,8,0,0,120,120,120,0,0,15,8,8
200 DATA8,8,120,0,0,96,96,96,96,96,0,0,120,8,8,8,8,15,8,8,8,8,120,8,8
205 DATA8,8,127,0,0,127,127,0,0,0,127,127,127,0,0,0,0,127,127,127
210 DATA3,3,3,3,3,7,7,7,7,7,112,112,112,112,112,64,64,64,64,127
215 DATA120,120,120,0,0,0,0,15,15,15,8,8,15,0,0,15,15,15,0,0,15,15,127,120,120
```

©

Marquee

Mark Bernstein
Department of Chemistry
Harvard University

Editor's Note: Although Mr. Bernstein's annotated source code starts at address $7000 (28672 decimal), we have included a BASIC loader (Program 1) which places the routine at $0360 (864 decimal) for those who have less RAM memory or prefer the convenience of storing machine language routines in the second cassette buffer. The screen size is set for 40 and the speed is 5. For an 80 column screen, POKE 864,80 and to change the speed, POKE 866,X. To test the routine, at this location, you would type SYS 1008. — RTM

The video display is a programmer's canvas. In this small space the programmer must communicate, inform, and perhaps entertain and enthrall. But, like all artists, programmers must work within the confines of their frame and the limits of their medium; all too often, the TV screen seems cramped and small.

All programmers must adjust to and accomodate the limitations of their computer's display. When using machine language, though, programmers must often work with awkward and clumsy tools. BASIC, PASCAL, FORTH and the like provide simple amenities like carriage returns, automatic spacing and tabs, while machine language leaves programmers to do all the work themselves. High level languages let programmers think in terms of character strings and display lines; assembly language programmers must think of individual symbols and screen locations.

In simulation and game programming, screen design can become a contest between graphics and text. An abundance of information, some vital, some merely interesting, competes for space within the screen's limited frame. Intricate graphics and display modes can compress lots of information into a small space — a picture is worth a thousand words — but usually demand intricate and time-consuming programming. Often the special programming is simply not worth the effort, and so the display has to be pruned. Information that won't fit on the screen remains forever hidden inside the computer.

Scrolling Text

The programmer's art ought not to be limited by the confines of the machine, only by skill and imagination. One useful solution to this conflict between the information and display space is the *marquee*, a small area of the screen across which text scrolls from right to left. The whole message doesn't have to be displayed at one time, so less space needs to be reserved for text and more area can be used for graphics. Long and short messages can be displayed with equal ease. And users, trained by long years of watching scoreboards, advertising displays and theatre marquees, find scrolling displays easy to understand and to use.

Using Interrupts

The computer takes only a few milliseconds to write a conventional message on the screen. Normally, writing occupies the computer's complete attention, and everything else must wait until the whole message has been displayed. But, since computers can write very quickly and people read comparatively slowly, most of the computer's time remains free for data processing.

Marquee displays, on the other hand, are intimately tied to human reading speed. The computer needs very little time to update the marquee, and could add a new letter a thousand times a second. If the computer wrote at full speed, the message would whiz across the screen, an illegible blur. To be useful, the marquee must move slowly.

Long marquee displays require many seconds, even minutes. This delay would be unacceptable if the computer were continuously occupied while displaying the message. The computer should not have to wait for the slow human reader. Instead, useful work can be accomplished in the long intervals between marquee updates.

We use a programmable timer to *interrupt* the computer periodically. A few times each second this interrupt instructs the computer to advance the marquee one step. The computer spends the rest of the time running its program normally and returns, after each marquee update, to the task that was interrupted.

An important benefit of this *interrupt-driven* strategy is *transparency*. Conventional, all-at-once output is simple and modular. The user's program calls an output routine, the output routine writes the specified letters on the screen, and then control returns to the user's program. To make marquees easy for the programmer, they should seem (to the programmer) to work just like normal output routines.

Interrupts make marquee displays as easy to use as normal output routines. Conventional routines do the writing immediately; the marquee controller arranges for the periodic interrupts which, without further intervention from the program, will draw the message on the marquee.

The Marquee Generator

Figure 1 shows the structure of a user program which invokes the marquee generator. The user program can activate the marquee by calling STOP. When the marquee is active, periodic interrupts divert the computer's attention from the user's program (left column) to the interrupt service routines (right column).

Figure 1.

The user doesn't have to control the marquee directly. Instead, periodic interrupts invoke IRQSRV, which decides whether the marquee should be updated. Updates are handled by invoking SCROLL.

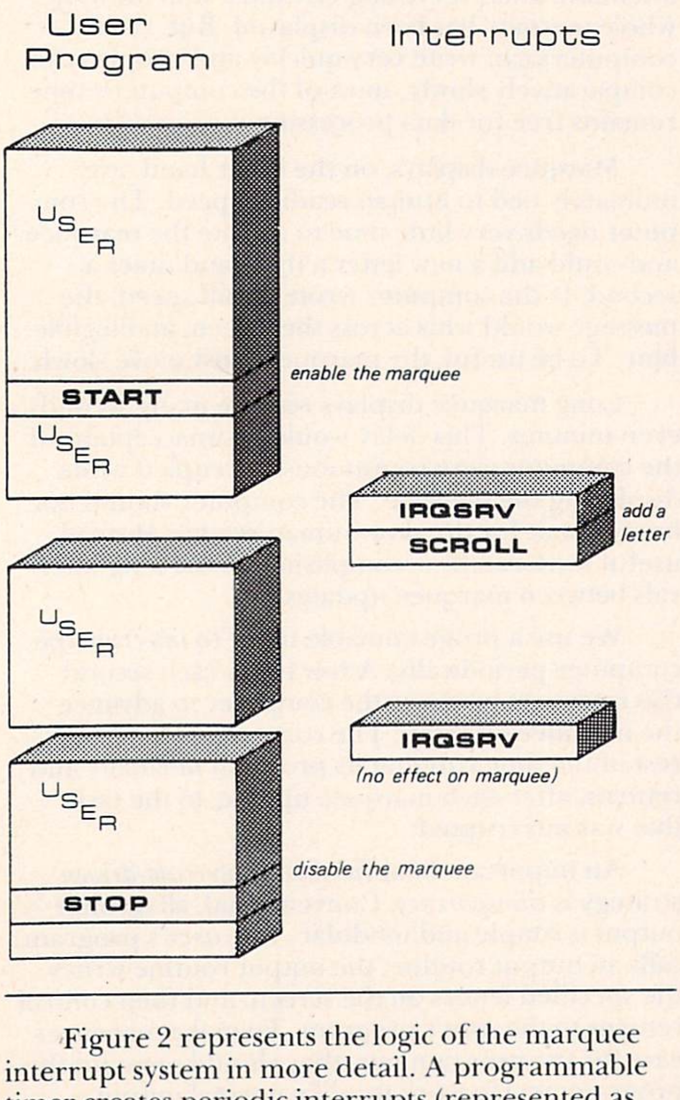


Figure 2 represents the logic of the marquee interrupt system in more detail. A programmable timer creates periodic interrupts (represented as marbles) at regular intervals. When the marquee is inactive, control falls directly into the computer's "normal interrupt handler" which ultimately returns control to the user program.

When the marquee is active, interrupt processing is diverted through IRQSRV, which decides whether or not to update the marquee. If no update is necessary, control passes directly to the normal interrupt handler. If the marquee is to be updated, IRQSRV, invokes SCROLL before allowing control to revert to the normal path.

Figure 2:

In this drawing, interrupts are represented by marbles rolling downhill through troughs. When the marquee is not active, IRQSRV is disconnected from the interrupt system and control passes directly to the normal interrupt handler. Activating the marquee inserts IRQSRV and SCROLL into the interrupt path.

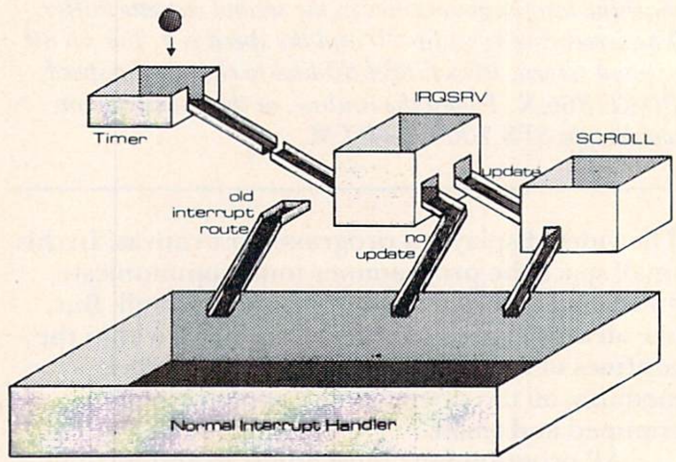
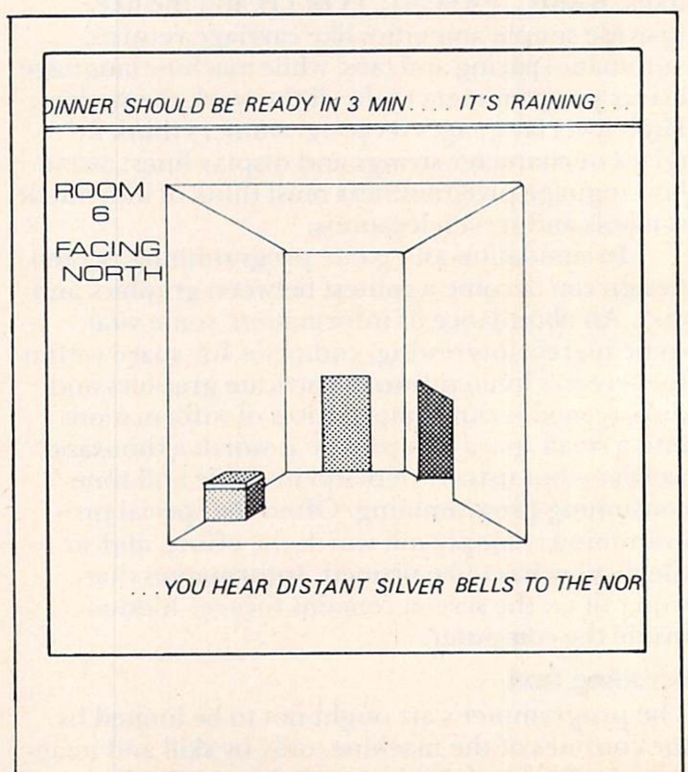


Figure 3.



chips...chips...chips...chips...chips...chips...

Limited Time: FREE Disk-O-Pro.

Special: You can get this Disk-O-Pro **absolutely free**... when you order the new Commodore 2031 single disk drive from Skyles. (See page 8.)

Command-O or Command-O-Pro?

It's called the Command-O-Pro in Europe, Command-O in the U.S.

But whatever you call it, this 4K byte ROM will provide your CBM BASIC 4.0 (4016, 4032) and 8032 computers with 20 additional commands including 10 Toolkit program editing and debugging commands and 10 additional commands for screening, formatting and disk file manipulating. *(And our technical writer dug up 39 additional commands in the course of doing a 76-page manual!)*

Extends Commodore's 8032 advanced screen editing features to the ultimate. **SCROLL** up and down, insert or delete entire lines, delete the characters to the left or right of the cursor, select **TEXT** or **GRAPHICS** mode or ring the PET's bell. Redefine the window to adjust it by size and position on your screen. And define any key to equal a sequence of up to 90 key strokes.

Resides in hexadecimal address **$9000**, the rightmost empty socket in 4016 and 4032 or the rearmost in 8032. If space conflicts, Socket-2-ME available at a very special price.

Command-O from Skyles Electric Works $75.00
Complete with **Socket-2-ME** 95.00

Disk-O-Pro ROM:

25 new commands at $75.00, that's only $3.00 a command!

The powerful chip for all "classic" PETs. In any PET with Version III (BASIC 2.0) ROMs (### **COMMODORE BASIC** ###) Disk-O-Pro will give 19 software compatible disk instructions*: 15 identical with the new BASIC 4.0 (or with 8032 ROMs) compatible with both old and new DOS. Plus 4 additional disk commands...including program appending (**MERGE**), program overlaying (**MERGE #**) and **PRINT USING**, allowing formatting output of strings and numbers on the PET screen or on any printer.

***NOTE:** *Old DOS doesn't recognize 3 commands.*

Plus softtouch key (**SET**) which allows you to define a key to equal a sequence of up to 80 keystrokes; **SCROLL** whereby all keys repeat as well as slow scrolling and extra editing features; **BEEP** which allows you to play music on your PET.

Completely compatible with the BASIC Programmer's Toolkit. *(The chip resides in the socket at hexadecimal address* **$9000**, *the rightmost empty socket in most PETS.)* And for the owners of "classic" (or old) PETs, we do have interface boards.

(For those owning a BASIC 4.0 or 8032, even though the Disk-O-Pro may not be suitable, the Command-O is. We have never abandoned a PET owner.)

Complete with 84-page manual written by Greg Yob.

Disk-O-Pro from Skyles Electric Works $75.00
Complete with **Interface Board** (for "Classic" PETs) 95.00

"The PET with the new PicChip...is like home movies."

The new ROM that took Europe by storm is available only from Skyles Electric Works in the U.S. and Canada.

PicChip, a ROM extension of the BASIC Version III (BASIC 2), BASIC 4.0 or BASIC 8032 interpreter. Over 40 commands that allow you to create programs with dynamic graphics displays: plots, bar graphs, pictures; and rolling, scrolling, shifting and inverting. All instantly and easily added to your BASIC program.

The address for the 2000/3000 (which would require PicChip module PC2), for the 4000 (PC4) and for the 8000 (PC8) is $A000...unless you have a Mikro, Word Pro 3 or 4, or Jinsam, which occupy that same address. In those cases, you will need the PicChip on an interface board... for the 2000/3000 series, PCB2, for the 4000, PCA4, and for the 8000, PCA8. In all cases, the Mikro or WordPro would be switchable manually using the Skyles Socket-2-ME...available at a very special price.

PicChip from Skyles Electric Works $60.00
(Please indicate PC2, PC4, PC8)
Complete with Socket-2-ME 80.00
(Please indicate PCB2, PCA4, PCA8)

The New Mikro:

Now you can program in machine language as easily as writing BASIC.

4K machine language assembler ROM plugs into your main board. Does all the machine language work for you; all you have to do is start laying down the code.

Retains all the great screen editing features of the PET... even all the Command-O or Toolkit commands. Write your own machine language subroutine. The program you write is the source code you can save. Machine language monitor saves the object code. Not as professional as the Skyles MacroTeA...not as expensive, either.

A great learning experience for those new to machine language programming but who want to master it easily. Superb 50-page manual—including brief introduction to machine language programming—included. But we also recommend several of the books in the Skyles Book Barn.

Skyles Mikro Machine Language Assembler
MK 2/3 for 2000 or 3000 Series PETs $80.00
MK 4/8 for 4000 or 8000 Series PETs and CBMs
Complete with **Socket-2-ME** 95.00

Skyles Catalogue Page 1

For PET and CBM owners only:

This is just 1 of 20 pages of the newest and biggest Skyles catalog, hot off the press.

We know you'll want this page, in its full 8½ x 10 splendor, and another 19 pages of peripherals, software and books that will make your PET or CBM computer even nicer to live with. So, if we missed sending you your very own copy within the last few weeks, call us at **(800) 227-9998** (unless you live in California, in which case call **(415) 965-1735.**

From Skyles Electric Works, the oldest and largest professional specialists in the business.

Skyles Electric Works
231-E S. Whisman Road
Mountain View, California 94041

Program 1 presents an implementation of the marquee system for the Upgrade ROM PET. The user's program calls START when it wants to put a message on the marquee. START initializes several variables and constants and, most importantly, routes all further interrupt requests *via* the marquee update controller IRQSRV.

Program 1.

```
800 FOR ADRES=864TO1023:READ DATTA:
    POKE ADRES,DATTA:NEXT ADRE
    S
864 DATA 40, 0, 5, 85, 228, 7
870 DATA 5, 40, 160, 1, 185, 0
876 DATA 128, 153, 255, 127, 200, 2
    04
882 DATA 96, 3, 208, 244, 32, 161
888 DATA 3, 205, 97, 3, 240, 15
894 DATA 192, 255, 240, 11, 200, 14
    0
900 DATA 101, 3, 172, 96, 3, 153
906 DATA 255, 127, 96, 172, 96, 3
912 DATA 169, 32, 153, 255, 127, 23
    8
918 DATA 103, 3, 173, 103, 3, 205
924 DATA 96, 3, 176, 48, 96, 172
930 DATA 101, 3, 177, 0, 41, 191
936 DATA 96, 141, 0, 0, 142, 1
942 DATA 0, 169, 0, 141, 103, 3
948 DATA 141, 101, 3, 173, 144, 0
954 DATA 141, 99, 3, 173, 145, 0
960 DATA 141, 100, 3, 120, 169, 223

966 DATA 141, 144, 0, 169, 3, 141
972 DATA 145, 0, 88, 96, 120, 173
978 DATA 99, 3, 141, 144, 0, 173
984 DATA 100, 3, 141, 145, 0, 88
990 DATA 96, 206, 102, 3, 16, 9
996 DATA 32, 104, 3, 173, 98, 3
1002 DATA 141, 102, 3, 108, 99, 3
1008 DATA 162, 3, 169, 248, 32, 169
1014 DATA 3, 96, 77, 65, 82, 81
1020 DATA 85, 69, 69, 0, 0, 70
```

The PET's 6522 timer generates interrupts 60 times per second. While the marquee is active, these interrupt requests invoke IRQSRV. This routine decides whether or not it's time to update the marquee; the speed of the marquee display is determined by the variable RATE, which specifies the number of interrupts which will occur between marquee updates. By adjusting RATE, the marquee's progress may be speeded up or slowed down.

If IRQSRV decides not to update the display, it jumps to the computer's normal interrupt handler, whose address is stored in OLDIRQ. If IRQSRV decides to update the display, it calls SCROLL before returning control to the machine's normal procedures.

SCROLL alone actually writes and updates the marquee. SCROLL first moves each character on the marquee line one space to the left. Next, SCROLL calls GETCHAR, which locates the next character in the message. The new character is tacked onto the right-hand edge of the message before SCROLL returns to IRQSRV.

A special character, END (usually 00, the ASCII NUL character), marks the end of each marquee message. When SCROLL encounters the end of a message, it starts tacking blanks onto the end of the marquee line. Eventually all the text will travel off the left edge of the screen, leaving the marquee blank; at this point, SCROLL automatically invokes STOP to disable future marquee updates.

For Other Computers

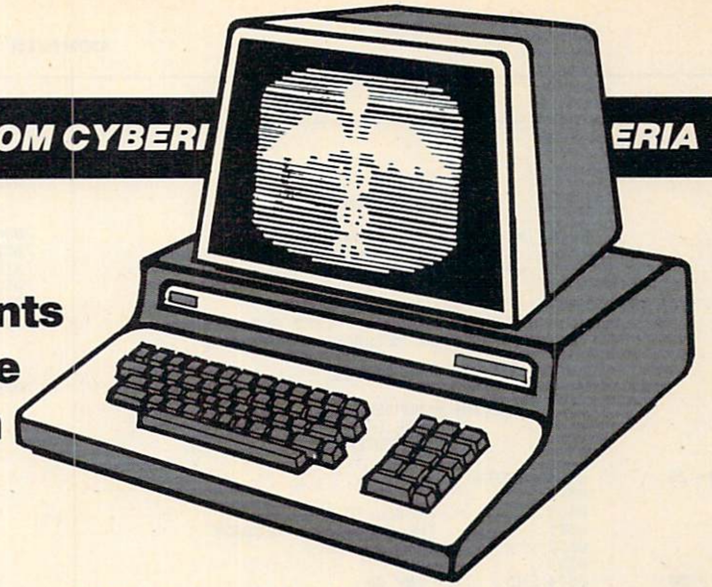
The marquee routines described here can be used on many 6502 systems with little or no change.

Different model PET's are easily accomodated. The only ROM-dependent instruction is the address IRQVEC, the page-zero location through which the PET vectors its interrupts.(It's the same, $90, 81 in 4.0 BASIC. For Original PETs, use $0219, 021A] 80-column computers, of course, can have 80-character marquees; simply change the value in LENGTH to 80.

Other computers should also be able to use this marquee system. The basic requirements are a memory-mapped display and a source of periodic interrupts. Many single-board computers, for example, use the 6522 VIA/timer which does this job admirably. Apple users will need to add an expansion board if one of their current accessories won't do the job. Several Apple parallel port I/O boards include the 6522; additionally, some time-of-day clock boards can generate periodic interrupts to drive the marquee.

Finally, note that marquees might be used in several different ways. They need not occupy an entire line; to use only a part of a line, simply change LINE (the address of the left end of the marquee) and LENGTH (the length of the marquee). The marquee may appear anywhere on the screen, although the top (used here) and bottom lines are likely to be most popular. Several marquees might appear on the same screen! Finally, note that marquees may move very rapidly (for speed reading practice?), and are not limited to text, suggesting several interesting possibilities for unusual graphics.

NEW FROM CYBERIA ★ NEW FROM CYBERIA ★ NEW FROM CYBERIA



Mari

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

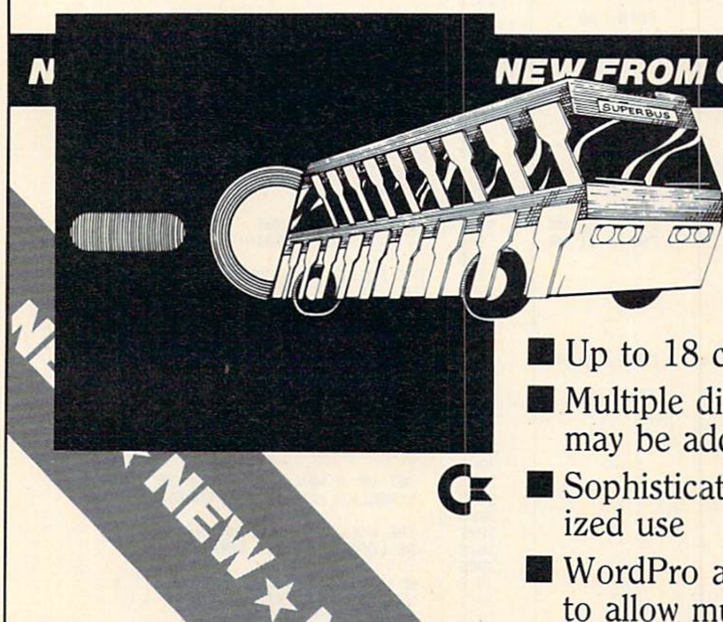
DESIGNED FOR COMMODORE 8000 SERIES COMPUTERS AND DISK DRIVES

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

Includes these standard reports:

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

NEW FROM CYBERIA ★ NEW FROM CYBERIA



SUPERBUS 4.0
Commodore Computer Networking System

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

*WordPro is a trademark of Professional Software, Inc.

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

Program 2.

```
                0010           .BA $7000
                0020           .OS
                0500 ;================================
                0501 ;   GLOBAL VARIABLES
                0502 ;================================
                0503 ;
                0504 ;
                0510 ; LINE = ADDRESS OF LEFT END OF
                0511 ;        THE MESSAGE DISPLAY LINE
                0512 ;
                0513 LINE      .DE $8000
                0514 ;
                0520 ; LENGTH = NUMBER OF CHARACTERS
                0521 ;          PER LINE FOR THIS
                0522 ;          COMPUTER'S VIDEO DISPLAY
                0523 ;
7000- 28        0524 LENGTH    .BY 40
                0525 ;
                0530 ; END  =  SYMBOL TO INDICATE
                0531 ;         THE END OF THE MESSAGE
                0532 ;         ( ASCII NUL )
                0533 ;
7001- 00        0534 END       .BY 00
                0535 ;
                0540 ; RATE =  NUMBER OF 1/60'THS SECONDS
                0541 ;         TO ELAPSE BETWEEN DIS-
                0542 ;         PLAY UPDATES.
                0543 ;
7002- 05        0544 RATE      .BY 5
                0545 ;
                0550 ; IRQVEC = ADDRESS WHERE THE
                0551 ;          COMPUTER STORES THE
                0552 ;          ADDRESS OF ITS INTERRUPT
                0553 ;          SERVICE ROUTINE.
                0554 ;
                0556 IRQVEC    .DE $90
                0557 ;
                0560 ; OLDIRQ = BUFFER FOR STORING THE
                0561 ;          COMPUTER'S NORMAL
                0562 ;          NORMAL INTERRUPT SERVICE
                0563 ;          ROUTINE.
                0564 ;
7003- 00 00     0565 OLDIRQ    .BY 0 0
                0566 ;
                0570 ; POINTR = PAGE ZERO POINTER
                0571 ;          TO THE START OF THE
                0572 ;          STRING WHICH THE
                0573 ;          COMPUTER WILL DISPLAY.
                0574 ;
                0575 POINTR    .DE 0
                0576 ;
                0580 ; CHAR = COUNT OF CHARACTERS
                0581 ;        WRITTEN TO THE MARQUEE
                0582 ;        TO DATE.  RESET BY START,
                0583 ;        UPDATED BY SCROLL, USED
                0584 ;        BY GETCHAR.
                0585 ;
7005- 00        0586 CHAR      .BY 0
                0587 ;
                0590 ; IRQCNT = COUNT OF INTERRUPT
                0591 ;          REQUESTS TO BE SKIPPED
                0592 ;          BEFORE ADVANCING THE
                0593 ;          MARQUEE.  USED ONLY BY
                0594 ;          IRQSRV.
                0596 ;
7006- 00        0597 IRQCNT    .BY 0
                0598 ;
                0599 ;==================================
                0600 ; AFTER = COUNT OF CALLS TO 'DONE'
                0601 ;         'DONE' APPENDS BLANKS TO
                0602 ;         THE MARQUEE UNTIL THE
                0603 ;         ENTIRE MARQUEE IS BLANK.
                0604 ;         'DONE' THEN DISABLES THE
                0605 ;         MARQUEE BE INVOKING
                0606 ;         'STOP'.
                0607 ;
7007- 00        0608 AFTER     .BY 0
                0609 ;
                0999 ;--------------------------------
                1000 ; SCROLL
                1001 ;
                1002 ; DISPLAY A 'TIMES-SQUARE' STYLE
                1003 ;   MESSAGE LINE.
                1004 ; 'SCROLL' IS CALLED PERIODICALLY
                1005 ;  BY AN INTERRUPT SERVICE
                1006 ;  ROUTINE, AND ADVANCES THE
                1007 ;  DISPLAY ONE NOTCH PER CALL.
                1008 ;
                1009 ;
                1010 ; SCROLL IS NOT USUALLY CALLED
                1011 ;  BY THE USER.
                1012 ;
                1013 ; INSTEAD, THE DISPLAY IS TUR'ED ON
                1014 ;  BY CALLING 'START'.
                1015 ;
                1016 ; AFTER THE ENTIRE MESSAGE IS
                1017 ;  DISPLAYED, THE DISPLAY ROUTINE
                1018 ;  TURNS ITSELF OFF.  IT CAN BE
                1019 ;  DEACTIVATED AT ANY TIME BY
                1020 ;  CALLING 'OFF'.
                1038 ;--------------------------------
                1039 ;
                1140 SCROLL
                1150 ;
7008- A0 01     1160           LDY #1        ; LEFT-MOST CHARACTER
                1170 ;
                1180 SCROLL1
700A- B9 00 80  1190           LDA LINE,Y
700D- 99 FF 7F  1210           STA LINE-1,Y
7010- C8        1220           INY
7011- CC 00 70  1240           CPY LENGTH    ; MOVED ENTIRE LINE ?
7014- D0 F4     1250           BNE SCROLL1   ; REPEAT 'TIL DONE
                1260 ;
                1270 ; GET THE NEXT CHARACTER
                1271 ;   TO BE DISPLAYED
                1272 ;
7016- 20 41 70  1280           JSR GETCHAR
                1290 ;
                1291 ; GETCHAR RETURNS THE NEXT CHARAC-
                1292 ;  TER IN A AND THE TOTAL NUMBER
                1293 ;  OF CHARACTER DISPLAYED SO FAR
                1294 ;  IN Y.
                1295 ;
                1296 ; NOW WE CHECK FOR THE END OF
                1297 ;  THE MESSAGE, WHICH HAPPENS
                1298 ;  AFTER THEN 'END' CHARACTER
                1299 ;  OR AFTER 256 CHARACTER HAVE
                1300 ;  BEEN DISPLAYED.
                1301 ;
7019- CD 01 70  1304           CMP END
701C- F0 0F     1310           BEQ DONE
                1312 ;
701E- C0 FF     1320           CPY #$FF
7020- F0 0B     1322           BEQ DONE
                1324 ;
                1330 ; UPDATE AND SAVE CHARACTER POINTER
7022- C8        1340           INY
7023- 8C 05 70  1350           STY CHAR
                1360 ;
                1370 ; PUT THE NEXT CHARACTER ON SCREEN
7026- AC 00 70  1380           LDY LENGTH
7029- 99 FF 7F  1400           STA LINE-1,Y
                1410 ;
702C- 60        1470           RTS
                1500 ;-----------------
                1501 ; DONE
                1502 ;-----------------
                1503 ;
                1520 DONE
702D- AC 00 70  1530           LDY LENGTH    ; END OF LINE
7030- A9 20     1540           LDA #$20      ; BLANK
7032- 99 FF 7F  1550           STA LINE-1,Y
                1560 ;
                1561 ; INCREMENT AFTER
                1562 ; QUIT WHEN AFTER>LENGTH
                1563 ;
7035- EE 07 70  1570           INC AFTER
7038- AD 07 70  1580           LDA AFTER
703B- CD 00 70  1590           CMP LENGTH
703E- B0 30     1600           BCS STOP
                1610 ;
7040- 60        1620           RTS
                1800 ;-------------------------
                1801 ; GETCHAR
                1802 ;-------------------------
                1803 ;
                1804 ; GET NEXT CHARACTER FROM
                1805 ; MESSAGE STRING
                1806 ;
                1807 ; RETURNS THE CHARACTER IN A
                1808 ;  AND THE TOTAL CHARACTER COUNT
                1809 ;  IN Y.
                1810 ;
7041- AC 05 70  1830 GETCHAR   LDY CHAR
7044- B1 00     1840           LDA (POINTR),Y
                1850 ;
                1851 ; THE FOLLOWING CONVERSION APPLIES
                1852 ; ONLY TO THE COMMODORE PET.
                1853 ; IT TRANSLATES ASCII STRINGS INTO
                1854 ; THE PET'S "SCREEN CODE"
                1856 ;
7046- 29 BF     1860           AND #$BF
                1870 ;
7048- 60        1880           RTS
                3000 ;------------
                3001 ;  START
                3002 ;------------
                3003 ;
                3004 ;  SET UP A NEW MESSAGE TO BE
                3005 ;  SCROLLED ACROSS THE SCREEN.
                3006 ;
                3007 ;  THE FOLLOWING REGISTERS MUST
                3008 ;  BE LOADED BEFORE CALLING START
                3009 ;
                3010 ;  A : LSB OF STRING ADDRESS
                3011 ;  X : MSB OF STRING ADDRESS
                3012 ;
                3013 ;
                3014 ;  THE DISPLAY WILL PROCEED
                3015 ;  AUTOMATICALLY UNTIL THE
                3016 ;  END-OF-MESSAGE CHARACTER
                3017 ;  (STORED IN 'END') IS FOUND.
                3018 ;  AFTER THE COMPLETE MESSAGE HAS
                3019 ;  BEEN DISPLAYED; THE DISPLAY
                3020 ;  WILL TURN ITSELF OFF.
                3021 ;  THE DISPLAY CAN BE DISABLED
```

```
                3022 ;  BY CALLING 'STOP' AT ANY TIME.
                3023 ;
                3024 ;----------------------------------
                3025 ;
                3026 START
                3027 ; STORE THE STRING'S ADDRESS
                3028 ;
7049- 8D 00 00  3030           STA POINTR
704C- 8E 01 00  3040           STX POINTR+1
                3050 ;
                3051 ; RESET 'AFTER' AND 'CHAR'
                3052 ;
704F- A9 00     3060           LDA #0
7051- 8D 07 70  3070           STA AFTER
7054- 8D 05 70  3072           STA CHAR
                3080 ;
                3081 ; SAVE OLD IRQ SERVICE VECTOR
                3082 ;
7057- AD 90 00  3090           LDA IRQVEC
705A- 8D 03 70  3100           STA OLDIRQ
705D- AD 91 00  3110           LDA IRQVEC+1
7060- 8D 04 70  3120           STA OLDIRQ+1
                3130 ;
                3131 ; SET UP NEW IRQ VECTOR
                3132 ;
7063- 78        3138           SEI
7064- A9 7F     3140           LDA #L,IRQSRV
7066- 8D 90 00  3150           STA IRQVEC
7069- A9 70     3160           LDA #H,IRQSRV
706B- 8D 91 00  3170           STA IRQVEC+1
706E- 58        3180           CLI
                3190 ;
706F- 60        3200           RTS
                3500 ;--------------------
                3501 ; STOP
                3502 ;--------------------
                3503 ;
                3504 ; DISABLE THE AUTOMATIC MESSAGE
                3505 ;  DISPLAY.
                3506 ;
                3507 ; STOP CAN BE CALLED DIRECTLY BY
                3508 ; THE USER.  IT IS ALSO INVOKED
                3509 ; BY 'DONE' WHEN THE COMPLETE
                3510 ; MESSAGE HAS BEEN DISPLAYED.
                3511 ;
                3520 STOP
                3521 ;
                3522 ; RESTORE THE ORIGINAL IRQ VECTOR
                3523 ;
7070- 78        3530           SEI
7071- AD 03 70  3540           LDA OLDIRQ
7074- 8D 90 00  3542           STA IRQVEC
7077- AD 04 70  3550           LDA OLDIRQ+1
707A- 8D 91 00  3552           STA IRQVEC+1
707D- 58        3560           CLI
                3570 ;
707E- 60        3580           RTS
                4000 ;------------------
                4001 ; IRQSRV
                4002 ;------------------
                4003 ;
                4004 ; THIS ROUTINE IS CALLED WHENEVER
                4005 ;
                4006 ;    A) THE PET TIMER CREATES AN
                4007 ;       INTERRUPT REQUEST
                4008 ;
                4009 ;    B)  THE MESSAGE ROUTINE HAS
                4010 ;        BEEN ENABLED BY CALLING
                4011 ;        'START',  AND HAS NOT
                4012 ;        YET BEEN DISABLED BY
                4013 ;        CALLING 'STOP'.
                4014 ;
                4015 ; THE PET TIMER REQUESTS AN
                4016 ; INTERRUPT 60 TIMES PER SECOND.
                4017 ;
                4022 ;
                4029 ;----------------------------------
                4030 ;
                4031 IRQSRV
707F- CE 06 70  4040           DEC IRQCNT
7082- 10 09     4050           BPL NORMAL
                4060 ;
                4061 ; CALL DISPLAY UPDATE
                4062 ;
7084- 20 08 70  4070           JSR SCROLL
                4080 ;
                4081 ; RESET IRQ COUNTER
                4082 ;
7087- AD 02 70  4090           LDA RATE
708A- 8D 06 70  4100           STA IRQCNT
                4110 ;
                4120 ; EXIT THROUGH THE STANDARD
                4121 ;  INTERRUPT SERVICE ROUTINE, WHOSE
                4122 ;  ADDRESS IS STORED IN 'OLDIRQ'.
                4123 ;
                4130 ;
708D- 6C 03 70  4140 NORMAL    JMP (OLDIRQ)
                5000 ;-------------
                5001 ; SAMPLE PROGRAM
                5002 ;-------------
                5003 ;
                5005 ;
                5006 TEST
                5007 ; LOAD THE ADDRESS OF THE MESSAGE
                5008 ; TO BE DISPLAYED INTO THE X AND
                5009 ; A REGISTER.
                5010 ;
7090- A2 70     5012           LDX #H,STRING
7092- A9 98     5020           LDA #L,STRING
                5022 ;
                5030 ;
                5031 ; CALL 'START' TO BEGIN DISPLAY
                5032 ;
7094- 20 49 70  5040           JSR START
                5050 ;
7097- 60        5110           RTS
                9000 ;-------------------------------
                9001 ;
                9002 ; STRING = SAMPLE STRING FOR
                9003 ;          TEST PROGRAM.
                9004 ;
                9005 ;-------------------------------
                9006 ;
7098- 4D 41 52  9010 STRING    .BY 'MARQUEE DISPLAYS ARE '
709B- 51 55 45
709E- 45 20 44
70A1- 49 53 50
70A4- 4C 41 59
70A7- 53 20 41
70AA- 52 45 20
70AD- 45 41 53  9011           .BY 'EASY TO PROGRAM AND '
70B0- 59 20 54
70B3- 4F 20 50
70B6- 52 4F 47
70B9- 52 41 4D
70BC- 20 41 4E
70BF- 44 20
70C1- 43 4F 4E  9012           .BY 'CONVENIENT FOR THE USER.
70C4- 56 45 4E
70C7- 49 45 4E
70CA- 54 20 46
70CD- 4F 52 20
70D0- 54 48 45
70D3- 20 55 53
70D6- 45 52 2E
70D9- 20 20
70DB- 54 48 45  9013           .BY 'THEY ALLOW LOTS OF INFOR'
70DE- 59 20 41
70E1- 4C 4C 4F
70E4- 57 20 4C
70E7- 4F 54 53
70EA- 20 4F 46
70ED- 20 49 4E
70F0- 46 4F 52
70F3- 4D 41 54  9014           .BY 'MATION TO BE DISPLAYED IN
70F6- 49 4F 4E
70F9- 20 54 4F
70FC- 20 42 45
70FF- 20 44 49
7102- 53 50 4C
7105- 41 59 45
7108- 44 20 49
710B- 4E 20
710D- 41 20 53  9015           .BY 'A SMALL SCREEN AREA.' 0
7110- 4D 41 4C
7113- 4C 20 53
7116- 43 52 45
7119- 45 4E 20
711C- 41 52 45
711F- 41 2E 00
                9999           .EN

LABEL FILE:  [ / = EXTERNAL ]

/LINE=8000          LENGTH=7000         END=7001
RATE=7002           /IRQVEC=0090        OLDIRQ=7003
/POINTR=0000        CHAR=7005           IRQCNT=7006
AFTER=7007          SCROLL=7008         SCROLL1=700A
DONE=702D           GETCHAR=7041        START=7049
STOP=7070           IRQSRV=707F         NORMAL=708D
TEST=7090           STRING=7098
//0000,7122,7122
```

©

Disk Disassembler

George H. Watson, Jr.
Physics Dept.
University of Delaware
Newark, DE

Editor's Note: This program works on either BASIC 3.0 or 4.0 and any 2040 DOS. It uses a printer. On some systems, the question "SKIP BASIC?" should be answered "NO" even though the program under disassembly is entirely in machine language.

There are several fine disassemblers available (in BASIC and in machine language) which disassemble programs while they reside in PET memory. Problems arise though when the program to be disassembled normally resides in the same memory space allocated to the disassembler. By relocating the disassembler (moving it to different memory space) it may still be used, although with a bit more difficulty. This problem may be circumvented by using a disassembler which does not require the program to be in PET memory. Instead, the program can be disassembled directly from the diskette on which it is stored by transferring the machine code byte-by-byte (reading the program) and translating into mnemonics, but not storing the bytes in memory.

A computer program is a set of instructions which are stored in the computer's memory in the form of bytes (8-bit words). A machine language program is a set of bytes which the microprocessor in your computer understands directly. On the other hand, a BASIC program consists of bytes which represent the various BASIC statements. When you RUN a BASIC program, each byte is interpreted and the microprocessor acts according to machine language subroutines which exist in the computer's ROMs. When you LIST a BASIC program, the operating system of your computer translates the bytes into BASIC statements, which are then displayed on the screen. Unfortunately no such LIST command is available for machine language programs on the PET microcomputer. But something *is* available which will translate the bytes into a form more understandable to a human. By allowing a disassembler to operate on the machine code, the program will be "LISTED" as 6502 microprocessor mnemonics, the heart of every PET.

DISK DISASSEMBLER opens a file to be read (the program to be disassembled) in the disk drive. The first two bytes which are read will contain the address at which the file is normally loaded into PET memory. The remaining bytes to be read comprise the program. All bytes will be translated into mnemonics until an end-of-file marker is detected (through the error word, ST), at which point the disassembly is finished.

Many programs which you may be interested in disassembling will be a combination of BASIC and machine code. DISK DISASSEMBLER handles the case where the machine code follows the BASIC program. All bytes are skipped over until three consecutive zeroes are detected which indicates the end of the BASIC program. All subsequent bytes will be disassembled.

As much as possible, I have attempted to make the output resemble the source code used by assemblers. (Source code for an assembler consists of the mnemonics for the microprocessor which the assembler converts into machine code.) One major benefit of an assembler is its ability to represent addresses with labels. Thus the machine language programmer is not required to calculate relative addresses needed for conditional branches — a tedious chore. DISK DISASSEMBLER does not provide the option of inputting labels (too time-consuming) but relative branches ARE converted to absolute branches, which makes understanding the disassembly easier.

DIS TEST is a compilation of all legal opcodes (instructions) available to the 6502 microprocessor. When disassembled, an alphabetical listing of the mnemonics along with their addressing modes will be printed out. If there are errors in the mnemonics or addressing modes, carefully check the DATA statements in lines 9000-9155. If the relative branches are wrong, check lines 670-675. Check all lines containing the address counter, AD, if the memory locations in the first column are incorrect.

Try DISK DISASSEMBLER on your favorite game or utility. You can learn much about machine language programming by studying the tricks used by others. You may also be able to learn more about the routines available in the PET's ROMs by examining how other programmers use them.

One option available in DISK DISASSEMBLER is the ability to change a legal opcode to an illegal opcode. Why do this? Some programs which you may disassemble use a legal opcode (unused otherwise) as filler between subroutines. I suppose this is to thwart disassembly since a simple NOP would also do the job. You may overcome this limitation by making the opcode illegal. How? Find the mnemonic in the DATA statement; make sure you find the one with the correct addressing mode. Now simply replace the number immediately following the mnemonic with a zero.

DISK DISASSEMBLER was written on a 32K PET (3.0) with 2040 disk drive. The program as written is slightly less than 7K in length, while variables, arrays, and strings require slightly less than 8K, so the program will run on a 16K PET; remove the REM statements if there is a problem. DISK DISASSEMBLER will also run on 4.0 PETs and with the new disk drive ROMs. For readers not inclined to type in long programs, contact me at the above address and I will provide tape copies at $3 each. (Include SASE, mailer, and tape.) Happy disassembling!

Speeding up BASIC

Some notes on DISK DISASSEMBLER:

1. Most frequently-used subroutines and the working part of the program should be placed at the beginning of the program (lower line numbers). When a GOSUB or GOTO is executed, BASIC begins at the first line of the program and compares each following line number until a match is obtained with the desired line number. Thus fewer line numbers need to be scanned for subroutines which are placed at the beginning. Disadvantage: a program may seem less structured.

2. Variables should be dimensioned as in lines 2000–2020 and the most-used variables should be initialized first. Similar to 1), when a variable is encountered, BASIC begins at the first variable in the table of variables and compares each following variable with the desired variable until a match is made. Dummy variables (constantly changing value and heavily used in subroutines) are good candidates for the first positions in the table. The variables should then be used as often as permitted.

3. When possible, use arrays of constants in place of conversions made with time-consuming subroutines. The biggest timesaving in DISK DISASSEMBLER was made by using an array of 256 hex characters, HG$( ), in place of a subroutine which converted the decimal value of a byte to the hex value. Disadvantage: more memory consumed.

4. Use IF FG THEN ... rather than IF FG<>0 THEN ... and IF ST-64 THEN ... rather than IF ST<>64 THEN ... The branch will be made if the argument of the IF .. THEN .. is nonzero.

5. Replace numbers with defined variables. In lines 300 and 400, B = 256. Time is saved since the conversion of the number 256 into the representation used by BASIC need not be done over and over; it was done once at initialization. Disadvantage: larger variable table.

I would also like to mention two shorthand tricks which are available.

6. Since any statement following a GOTO or RETURN on the same line is never executed, a remark may be placed there with no time lost and with no REM statement. See lines 10 and 100.

7. When DATA statements are read, if all that is seen is another comma (no data), then a variable is read to be zero and a string is read to be null.

Program 1.

```
                                                          ******* COMMENTS *******
10 GOTO1000:
100 IFST-64THENRETURN:                                    *CHECK FOR END-OF-FILE*
110 FG=1:RETURN
200 GET#5,D$:GOSUB100:IFD$=""THEND=0:D$="00":RETURN:*BYTE GET & CONVERSION*
210 D=ASC(D$):D$=H$(D):RETURN
300 A%=AD/B:AD$=H$(A%)+H$(AD-A%*B):RETURN:                *ADDRESS CONVERSION*
400 A%=D/B:C$=H$(A%)+H$(D-A%*B):RETURN:                   *DECIMAL -> 4-DIGIT HEX*
490 TI$="000000":REM                                      *BEGIN DISASSEMBLY*
500 IFFGTHENRETURN:                                        CHECK END-OF-FILE FLAG
510 GOSUB200:ONB%(D)GOTO540,600,700:REM                    GET 1ST BYTE & BRANCH
520 REM                                                   *1-BYTE INSTRUCTION*
530 D$=D$+"*":M$="":GOTO550:                               -ILLEGAL OPCODE
540 M$=M$(D):REM                                           -ACCUMULATOR,IMPLIED
550 PRINT#4,AD$"  "D$"      ",M$
560 AD=AD+1:GOSUB300:GOTO500
590 REM                                                   *2-BYTE INSTRUCTION*
600 B1=D:B1$=D$:M$=M$(D)+" ":GOSUB200:REM                  GET 2ND BYTE
605 ONA%(B1)GOTO610,620,630,640,650,660,670:               ADDRESSING MODE
610 M$=M$+"# $"+D$:GOTO680:                                -IMMEDIATE
620 M$=M$+"* $"+D$:GOTO680:                                -ZERO PAGE
630 M$=M$+"($"+D$+",X)":GOTO680:                           -INDEXED INDIRECT
```

NEW PRODUCT

SWARM-100

For 24 Pin ROM Machines Only.

JUST PLUG IT IN

• No soldering • No messy wires

SOFTWARE SELECTABLE

1. Software select one of two operating systems. (BASIC 2.0 / BASIC 4.0)
2. Software select utility ROMs at conflicting addresses.

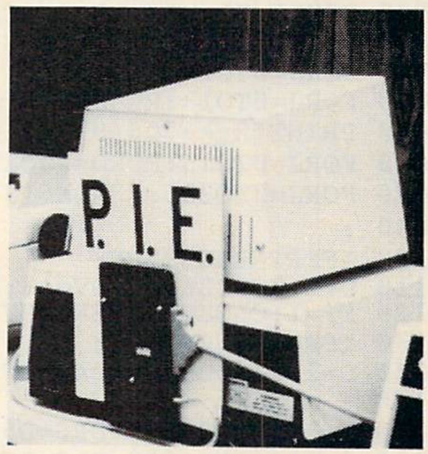
$150.00

Master Charge and VISA accepted.

BATTERIES INCLUDED

Village by the Grange
71 McCaul Street
Toronto, Ontario
Canada M5T 2X1
(416) 596-1405

PIE-C



PET/CBM* IEEE-488 TO PARALLEL PRINTERS By LemData Products

P.I.E.-C MEANS—Professional design, Indispensible features, Excellent quality and Cost effectiveness. You can't buy a better parallel interface for your PET/CBM.

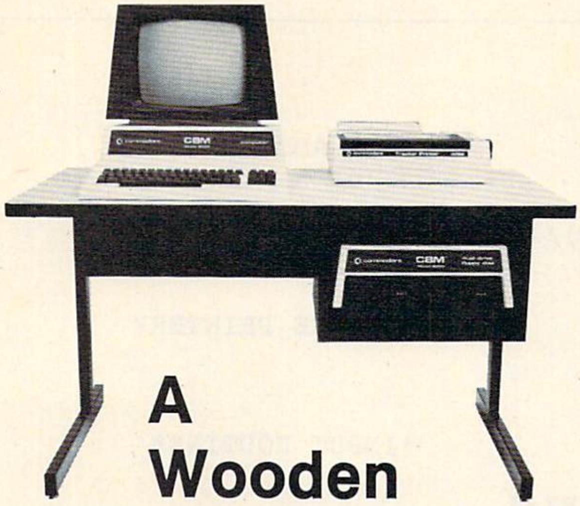
Our P.I.E.-C will interface your PET/CBM through the IEEE-488 bus to . . . . . .

the NEC Spinwriter, the C. Itoh Starwriter, printers by Centronics, Epson, Anadex, Escon Products, the Paper Tigers by IDS, the MIPLOT by Watanabe, the DIP printers, the AJ-841, the OKIDATA printers, plus ALL OTHER parallel ASCII printers.

Assembled with custom case, CBM-TO-ASCII code converter and appropriate cable, the **P.I.E.-C** is only $129.95 (+ $5 S&H). Md. Res. +5% tax. Specify printer and CBM models.

LemData Products, P.O. Box 1080, Columbia, Md. 21044 Phone (301) 730-3257
*PET/CBM are trademarks of Commodore Business Machines
(Dealer Inquiries Invited)

```
640 M$=M$+"($"+D$+"),Y":GOTO680:                  -INDIRECT INDEXED
650 M$=M$+"* $"+D$+",X":GOTO680:                  -ZERO PAGE INDEXED BY X
660 M$=M$+"* $"+D$+",Y":GOTO680:                  -ZERO PAGE INDEXED BY Y
670 IFD<128THEND=AD+D+2:GOTO675:                  -RELATIVE -> ABSOLUTE
672 D=AD+D-254
675 GOSUB400:M$=M$+"TO $"+C$
680 PRINT#4,AD$"  "B1$" "D$"    ",M$
690 AD=AD+2:GOSUB300:GOTO500
695 REM                                           *3-BYTE INSTRUCTION*
700 B1=D:B1$=D$:GOSUB200:B2$=D$:GOSUB200:REM      GET 2ND & 3RD BYTES
710 M$=M$(B1)+" $"+D$+B2$
720 ONA%(B1)GOTO760,730,740,750:                  ADDRESSING MODE
730 M$=M$+",X":GOTO760:                           -ABSOLUTE INDEXED BY X
740 M$=M$+",Y":GOTO760:                           -ABSOLUTE INDEXED BY Y
750 M$=LEFT$(M$,4)+"($"+D$+B2$+")":REM            -INDIRECT
760 PRINT#4,AD$"  "B1$" "B2$" "D$,M$:REM          -ABSOLUTE
770 AD=AD+3:GOSUB300:GOTO500
780 :
1000 GOSUB2000:REM INITIALIZE                     *BEGIN EXECUTION*
1100 GOSUB2100:REM SELECT PRINTER
1200 GOSUB2200:REM SELECT FILE
1300 GOSUB2300:REM OPEN FILE
1350 IFFETHEN1200: DISK ERROR
1400 GOSUB2400:REM GET LOAD ADDRESS
1500 GOSUB2500:REM SKIP BASIC
1600 GOSUB490: REM DISASSEMBLE FILE
1700 GOSUB2600:REM STOP?
1800 GOTO1100: REM REPEAT
1900 :
2000 DIMD,D$,AD,A%,B,B1,FG,C$,J:B=256:REM         *INITIALIZATION*
2010 DIMAD$,M$,B1$,B2$,DR$,FL$,FY,FP
2020 DIMDV,FD,DA$,FE,EN,EN$,EM$,ES$,ET$
2030 DIMD$(15),H$(255),M$(255),B%(255),A%(255),C$(13)
2040 FORJ=0TO15:READD$(J):NEXT:REM                FILL ARRAYS
2050 FORJ=0TO13:READC$(J):NEXT
2060 PRINT"ĥr>>"C$(0)C$(10):PRINT"↓"C$(11):PRINT"↓"C$(12)
2070 FORJ=0TO15:FORD=0TO15:H$(J*16+D)=D$(J)+D$(D):NEXT:NEXT
2080 FORJ=0TO255:READM$(J),B%(J),A%(J):NEXT:RETURN
2090 :
2100 IFFPTHENRETURN:                              *OPEN PRINTER*
2110 D=1:GOSUB6000:DV=A%:IFDV<3ORDV>30THEN2110
2120 FP=1:CLOSE4:OPEN4,DV:IFDV-3THENPRINT#4
2130 IFFDTHENRETURN:                              ENTER DATE
2140 D=2:GOSUB6000:DA$=D$:FD=1:RETURN
2150 :
2200 D=3:GOSUB6000:DR$=D$:IFA%ANDA%-1THEN2200:REM  *SELECT FILE*
2210 D=4:GOSUB6000:FL$=D$:IFDV=4ORDV=3THENPRINT#4,"ĥ
2220 PRINT#4,FL$,,DA$:PRINT#4:PRINT#4:RETURN
2230 :
2300 CLOSE15:CLOSE5:REM                           *INITIALIZE DISK DRIVE*
2310 OPEN15,8,15,"I"+DR$:GOSUB7000:IFFETHENRETURN: OPEN COMMAND CHANNEL
2320 OPEN5,8,5,DR$+":"+FL$+",P,R":GOSUB7000:REM    OPEN FILE FOR READ
2330 RETURN
2340 :
2400 GOSUB200:AD=D:AD$=D$:REM                     *GET LOAD ADDRESS*
2410 GOSUB200:AD=AD+D*B:AD$=D$+AD$:RETURN
2420 :
2500 PRINT"ĥ":D=5:GOSUB6000:IFFY=0THEN2570:       *SKIP BASIC*
2510 IFFY-1THEN2500
2520 PRINTC$(0)C$(6):J=0:REM                      CHECK FOR 3 ZEROES
2530 GET#5,D$:IFD$THENJ=J+1:GOTO2530
```



A Wooden Computer?

Not from Commodore!

So why should the desk look like wood? A pleasant cream and charcoal trimmed desk looks so much better with Commodore systems. One look and you'll see. Interlink desks are right. By design.

The specifications only confirm the obvious:

- •Cream and charcoal color beautifully matches the Commodore hardware and blends with your decor.
- •An ideal 710 mm (28") keyboard height yet no bumping knees because a clever cutout recesses the computer into the desk-top.
- •High pressure laminate on both sides of a solid core for lasting beauty and strength.
- •Electrostatically applied baked enamel finish on welded steel legs—no cheap lacquer job here.
- •T-molding and rounded corners make a handsome finish on a durable edge that won't chip.
- •Knocked down for safe, inexpensive shipment.
- •Patented slip joints for quick easy assembly.
- •Leveling glides for uneven floors.
- •Room enough for a Commodore printer on the desk, yet fits into nearly any den or office niche—H: 660 mm (26") W: 1170 mm (46") D: 660 mm (26").
- •Matching printer stand available with slot for bottom feeding.

Price: $299

In short, as Commodore dealers, we won't settle for anything that looks good only in the catalog! Our customers won't let us. They don't buy pictures. And neither should you. This is why **we will let you use one of our desks for a week** and then decide. If for any reason you don't like it, just return it in good condition for a cheerful refund.

If your Commodore dealer doesn't carry our desks yet, send a check for $299 and we will ship your desk freight paid!

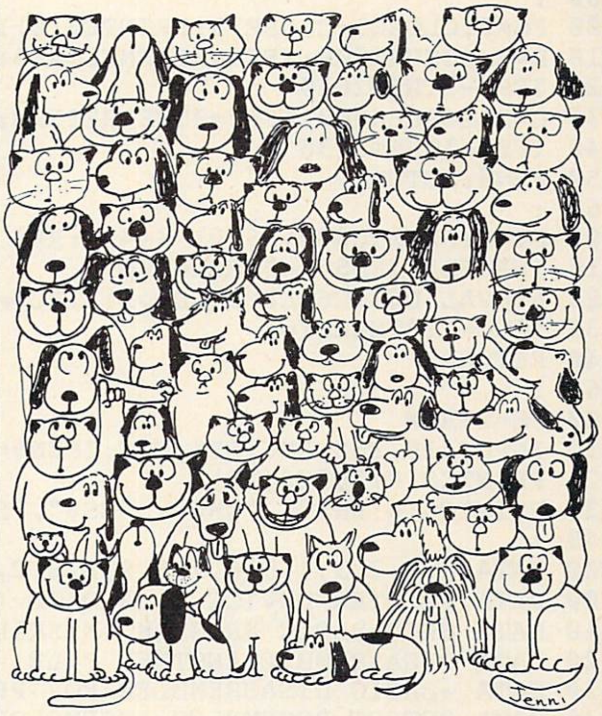
Name____________________

Address____________________

City__________ **St**_____ **Zip**_____

Interlink, Inc., Box 134, Berrien Springs, MI 49103
Master Charge and Visa welcome. Call our order line:
616-473-3103

Every PET Needs a Friend.



CURSOR is the best friend your Commodore PET will ever have. Since July, 1978 we have published 150 of the most user-friendly programs for the PET available anywhere. When we write or edit a program, we spend lots of time fussing about how it will treat you. We pay attention to lots of little things that help make using a computer a pleasure instead of a pain.

Naturally, **CURSOR** programs are technically excellent. Each program that we purchase is extensively edited or re-written by a professional programmer. But *imagination* is just as important as being user-friendly and technically good! We delight in bringing you off-beat, unusual programs that "show off" the abilities of your PET or CBM.

CURSOR is user-friendly, technically great and full of imaginative programs. And every issue of **CURSOR** is still available! We continue to upgrade previously published programs so that they'll work on the three varieties of Commodore ROM's (Old, New, and 4.0). New issues also work on the 80 column CBM.

For only $4.95 you can buy a sample issue and judge for yourself. Or send $18 for a four-issue subscription. Each **CURSOR** comes to you as a C-30 cassette with five programs and a graphic Front Cover, ready to LOAD and RUN on your PET.

Who knows? After your PET meets **CURSOR**, things may never be the same!

AUTHORIZED DISTRIBUTORS:

Great Britain
AUDIOGENIC, Ltd.
P.O. Box 88
Reading, Berkshire

Holland
COPYTRONICS
Bergemeester
Van Suchtelenstraat 46
7413 XP Deventer

Japan
SYSTEMS FORMULATE CORP.
Shin-Makicho Bldg. 1-8-17
Yaesu, Chuo-ku, Tokyo 103

Australasia
MICROCOMPUTER HOUSE, LTD.
133 Regent Street
Chippendale, Sydney
N.S.W. Australia 2008

THE CODE WORKS
Box 550
Goleta, CA 93116
805-683-1585

```
2540 GET#5,D$:IFD$THENJ=J+2:GOTO253Ø
2550 GET#5,D$:IFD$THENJ=J+3:GOTO253Ø
2560 AD=AD+J+3:GOSUB3ØØ:REM                    FIX ADDRESS
2570 PRINT"ĥ"C$(Ø)C$(7)"↓↓":RETURN
2580 :
2600 FG=Ø:CLOSE5:CLOSE15:PRINTC$(Ø)INT(TI/6)/1Ø"SEC
2610 D=8:GOSUB6ØØØ:IFFY=ØTHENCLOSE4:END:REM    *DO ANOTHER FILE?*
2620 IFFY-1THEN261Ø
2630 D=9:GOSUB6ØØØ:IFFY=ØTHENRETURN:REM        CHANGE PRINTER?
2640 IFFY-1THEN263Ø
2650 FP=Ø:RETURN
2660 :
6000 FY=2:PRINTC$(Ø)C$(D)"←←←";:REM            *INPUT ROUTINE*
6010 INPUTD$:IFD$="-"THEN6ØØØ
6020 A%=VAL(D$):C$=LEFT$(D$,1):IFC$="N"THENFY=Ø
6030 IFC$="Y"THENFY=1
6040 RETURN
6060 :
7000 FE=Ø:REM                                  *CHECK FOR DISK ERROR*
7010 INPUT#15,EN$,EM$,ET$,ES$:IFEN$="ØØ"THENRETURN
7020 PRINTC$(Ø)"r̲"C$(13)
7030 PRINTC$(Ø)EN$", "EM$", "ES$","ET$:FE=1:RETURN
7040 :
8000 DATA Ø,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,"↓↓↓→→→→→→→→→"
8500 DATA PRINTER DEVICE #  3,DATE (MO/DA/YR)  -,DRIVE #  Ø,FILENAME  -
8510 DATA SKIP BASIC PROGRAM  N,SKIPPING BASIC ....,DISASSEMBLING ....
8520 DATA DISASSEMBLE ANOTHER FILE  N,DIFFERENT PRINTER  N,DISK DISASSEMBLER
8530 DATA -BASIC DISASSEMBLER FOR PET DISK FILES-
8540 DATA OUTPUT RESEMBLES ASSEMBLER SOURCE CODE.,DISK ERROR
9000 DATA BRK,1,,ORA,2,3,,,,,,,,,ORA,2,2,ASL,2,2,,,
9005 DATA PHP,1,,ORA,2,1,ASL A,1,,,,,,,ORA,3,1,ASL,3,1,,,
9010 DATA BPL,2,7,ORA,2,4,,,,,,,,,ORA,2,5,ASL,2,5,,,
9015 DATA CLC,1,,ORA,3,3,,,,,,,,,ORA,3,2,ASL,3,2,,,
9020 DATA JSR,3,1,AND,2,3,,,,,,BIT,2,2,AND,2,2,ROL,2,2,,,
9025 DATA PLP,1,,AND,2,1,ROL A,1,,,,,BIT,3,1,AND,3,1,ROL,3,1,,,
9030 DATA BMI,2,7,AND,2,4,,,,,,,,,AND,2,5,ROL,2,5,,,
9035 DATA SEC,1,,AND,3,3,,,,,,,,,AND,3,2,ROL,3,2,,,
9040 DATA RTI,1,,EOR,2,3,,,,,,,,,EOR,2,2,LSR,2,2,,,
9045 DATA PHA,1,,EOR,2,1,LSR A,1,,,,,JMP,3,1,EOR,3,1,LSR,3,1,,,
9050 DATA BVC,2,7,EOR,2,4,,,,,,,,,EOR,2,5,LSR,2,5,,,
9055 DATA CLI,1,,EOR,3,3,,,,,,,,,EOR,3,2,LSR,3,2,,,
9060 DATA RTS,1,,ADC,2,3,,,,,,,,,ADC,2,2,ROR,2,2,,,
9065 DATA PLA,1,,ADC,2,1,ROR A,1,,,,,JMP,3,4,ADC,3,1,ROR,3,1,,,
9070 DATA BVS,2,7,ADC,2,4,,,,,,,,,ADC,2,5,ROR,2,5,,,
9075 DATA SEI,1,,ADC,3,3,,,,,,,,,ADC,3,2,ROR,3,2,,,
9080 DATA ,,,STA,2,3,,,,,,STY,2,2,STA,2,2,STX,2,2,,,
9085 DATA DEY,1,,,,TXA,1,,,,,STY,3,1,STA,3,1,STX,3,1,,,
9090 DATA BCC,2,7,STA,2,4,,,,,,STY,2,5,STA,2,5,STX,2,6,,,
9095 DATA TYA,1,,STA,3,3,TXS,1,,,,,,,,STA,3,2,,4,,,,
9100 DATA LDY,2,1,LDA,2,3,LDX,2,1,,,,LDY,2,2,LDA,2,2,LDX,2,2,,,
9105 DATA TAY,1,,LDA,2,1,TAX,1,,,,,LDY,3,1,LDA,3,1,LDX,3,1,,,
9110 DATA BCS,2,7,LDA,2,4,,,,,,LDY,2,5,LDA,2,5,LDX,2,6,,,
9115 DATA CLV,1,,LDA,3,3,TSX,1,,,,,LDY,3,2,LDA,3,2,LDX,3,3,,,
9120 DATA CPY,2,1,CMP,2,3,,,,,,CPY,2,2,CMP,2,2,DEC,2,2,,,
9125 DATA INY,1,,CMP,2,1,DEX,1,,,,,CPY,3,1,CMP,3,1,DEC,3,1,,,
9130 DATA BNE,2,7,CMP,2,4,,,,,,,,,CMP,2,5,DEC,2,5,,,
9135 DATA CLD,1,,CMP,3,3,,,,,,,,,CMP,3,2,DEC,3,2,,,
9140 DATA CPX,2,1,SBC,2,3,,,,,,CPX,2,2,SBC,2,2,INC,2,2,,,
9145 DATA INX,1,,SBC,2,1,NOP,1,,,,,CPX,3,1,SBC,3,1,INC,3,1,,,
9150 DATA BEQ,2,7,SBC,2,4,,,,,,,,,SBC,2,5,INC,2,5,,,
```

SM-KIT

for Commodore Computers

A Programming Productivity Tool

ONLY $40

A 4K ROM with both programming and disk handling aids.

The SM-KIT is a collection of machine language firmware programming and test aids for BASIC programmers. SM-KIT is a 4K ROM (twice the normal capacity) which you simply insert in a single ROM socket on any BASIC 4 CBM/PET–either 80 column or 40 column. Includes both programming aids and disk handling commands.

ERROR DETECTION: the SM-KIT automatically indicates the erroneous line and statement for any BASIC program error.
LINE NUMBERING: the SM-KIT automatically numbers BASIC statements until you turn the function off.
SCREEN OUTPUT: the commands FIND, DUMP, TRACE and DIRECTORY display on the CRT while you hold the RETURN key (display pauses when the key is released). Continuous output is selected with shift-lock.
OUTPUT CONTROL to DISK or PRINTER: in addition to displaying on the CRT, you can direct output to either disk or printer.
HARDCOPY: allows screen displays to be either printed or stored on disk.
FIND: searches all or any part of a program for text or command strings or variable names. Either exact search or wild card search supported.
RENUMBER: the SM-KIT can renumber all or any part of a program. The selective renumbering allows you to move blocks of code within your program.
VARIABLE DUMP: displays the contents of floating point, integer, and string variables (both simple and array). Can display all variables or any selected variables.
TRACE: SM-KIT can trace program execution either continuously or step by step starting with any line number. Selected program variables can be displayed while tracing.
DISK COMMANDS: as in DOS Support (Universal Wedge), the "shorthand" versions of disk commands may be used for displaying disk directory, initializing, copying, scratching files, load and run, etc.
LOAD: SM-KIT can load all or part of BASIC or machine language programs. It can append to a program in memory, overwrite any part of a program, load starting with any absolute memory location, and load without changing variable pointers.
MERGE: allows merging all or any part of a program on disk with a program in memory.
SAVE and VERIFY: SM-KIT provides one step program save and verification. It also allows you to save any part of a program, or any address range.

Developed by (and available in Europe from) SM Softwareverbund-Microcomputer GmbH, Scherbaumstrasse 29, 8000 Munchen 83, Germany

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

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

QUALITY ACCOUNTING SOFTWARE for the CBM COMPUTER

THE GENERAL LEDGER SYSTEM $150.00

All entries are made via formatted, fill in the blanks, screens. There is a separate check stub format disbursements entry screen and eight digit account numbers to allow sub coding as required. Up to fifty user designated journals are available. All data is verified on input with balance enforced. All journals are available for print at any point in the accounting cycle. Any printout may be printed by department. The general ledger prints: balance forward, full detail of each transaction, total credits, total debits, and end balance for each account. Available reports include: journals, disbursements register, current trial balance, audit trial balance, budget trial balance, income statement, balance sheet, cash flow analysis, and comparison of budget vs. actual amounts for year to date, or the current period.

FUND ACCOUNTING SYSTEM $200.00

The perfect accounting system for the municipal utility district, and the small city or school district. The system includes all features of the general ledger system with the added ability of printing all reports, and the general ledger, by fund as well as by department.

CLIENT ACCOUNTING SYSTEM $200.00

The accounting tool kit for the public accountant. This system includes all features of the general ledger system with the addition of a payroll check stub formatted screen for payroll check input. Also included are: a payroll disbursement register, a 1099 register, 941 reports, and W-2's.

INTERLOCKING MODULES

The following modules are available for any of the general ledger based systems:

Accounts Payable	$ 75.00
Accounts Receivable	$ 75.00
Payroll	$ 75.00
Job Cost Payroll	$100.00
Utility Billing System	$100.00

Accounting software may be reviewed at your dealer, or via mail. Full catalog, demo disc, sample operations manual, (please specify which system), and a $20. credit coupon — only $20.00

DEALERS! Please Write for Dealers Pack

Expanded Software Catalog Free on Request

BASIC SOFTWARE SERVICE
P.O. BOX 181, LA PORTE, TX., 77571
Phone Area Code 713/ 470-1857

```
9155 DATA SED,1,,SBC,3,3,,,,,,,,,,SBC,3,2,INC,3,2,,,
10000 *********************
10010 *                   *
10020 * DISK DISASSEMBLER *
10030 *                   *
10040 * G.H.WATSON   3/81 *
10050 *                   *
10060 *********************
11000 ----- VARIABLE TABLE ------
11010 J,D      DUMMY INDEX/VARIABLE
11020 A%       DUMMY INTEGER
11030 D$,C$    DUMMY STRINGS
11040 FG       END-OF-FILE FLAG
11050 B1       OPCODE (DEC)
11060 AD,AD$   ADDRESS (DEC,HEX)
11070 M$       INSTRUCTION
11080 B1$,B2$ 1ST & 2ND BYTES OF CODE
11090 DR$,FL$ DRIVE #,FILENAME
11100 DA$      DATE
11110 D$()     HEX NUMERALS
11120 H$()     HEX FOR BYTES
11130 M$()     6502 MNEMONICS
11140 B%()     # BYTES IN INSTRUCTION
11150 A%()     ADDRESSING MODE
11160 C$()     PRINT STRINGS
11170 EN,EN$   ERROR #
11180 EM$      ERROR MESSAGE
11190 ET$,ES$ ERROR TRACK/SECTOR
11200 FE       DISK ERROR FLAG
11210 FP,FD    PRINTER/DATE FLAG
11220 FY       FLAG FOR YES/NO
11230 DV       PRINTER DEVICE #
11240 B        CONSTANT = 256
READY.
```

Program 2.

```
5000 69 00 6D 00 00 65 00 61
5008 00 71 00 75 00 7D 00 00
5010 79 00 00 29 00 2D 00 00
5018 25 00 21 00 31 00 35 00
5020 3D 00 00 39 00 00 0E 00
5028 00 06 00 0A 16 00 1E 00
5030 00 90 00 B0 01 F0 7F 2C
5038 00 00 24 00 30 80 D0 FE
5040 10 FF 00 50 00 70 00 18
5048 D8 58 B8 C9 00 CD 00 00
5050 C5 00 C1 00 D1 00 D5 00
5058 DD 00 00 D9 00 00 E0 00
5060 EC 00 00 E4 00 C0 00 CC
5068 00 00 C4 00 CE 00 00 C6
5070 00 D6 00 DE 00 00 CA 88
5078 49 00 4D 00 00 45 00 41
5080 00 51 00 55 00 5D 00 00
5088 59 00 00 EE 00 00 E6 00
5090 F6 00 FE 00 00 E8 C8 4C
5098 00 00 6C 00 00 20 00 00
50A0 A9 00 AD 00 00 A5 00 A1
50A8 00 B1 00 B5 00 BD 00 00
50B0 B9 00 00 A2 00 AE 00 00
50B8 A6 00 BE 00 00 B6 00 A0
50C0 00 AC 00 00 A4 00 B4 00
50C8 BC 00 00 4E 00 00 46 00
50D0 4A 56 00 5E 00 00 EA 09
50D8 00 0D 00 00 05 00 01 00
50E0 11 00 15 00 1D 00 00 19
50E8 00 00 48 08 68 28 2E 00
50F0 00 26 00 2A 36 00 3E 00
50F8 00 6E 00 00 66 00 6A 76
5100 00 7E 00 00 40 60 E9 00
5108 ED 00 00 E5 00 E1 00 F1
5110 00 F5 00 FD 00 00 F9 00
5118 00 38 F8 78 8D 00 00 85
5120 00 81 00 91 00 95 00 9D
5128 00 00 99 00 00 8E 00 00
5130 86 00 96 00 8C 00 00 84
5138 00 94 00 AA A8 BA 8A 9A
5140 98 AA AA AA AA AA AA AA

S "DIS TEST",08,5000,5141
```

©

Line Input For The PET

Robert Lando
Toronto, Canada

Unfortunately, as many users will agree, the INPUT command on the Commodore Pet contains several undesirable features. First of all, if the RETURN key is pressed before any data is entered, the program will abruptly end and the user will be left with a READY. message. Although there are several "tricks" that the programmer can use to prevent this from happening, they do not alleviate another major problem.

No matter how an INPUT is programmed, when it is encountered the computer waits for a key to be typed, echoes it back to the screen, and waits for another until the RETURN key is pressed. The problem is that if the user enters a cursor movement key, its function will be echoed back to the screen. The user could, for example, clear the screen, and have no way of recovering the lost information.

Some computers offer a command that will accept one line of input from the user. The only acceptable keys are un-shifted letters from A-Z, digits from 0-9, the space bar, the delete key, and the return key. All other keys are ignored. Pressing RETURN, DELete, or SPACE before something else is typed will have no effect, and trailing spaces are ignored. This command, usually called INLINE (INput LINE), or LINPUT (Line INPUT), can be used in place of INPUT.

Below is a program written in machine language, that when called with a SYS command to its starting location, will do a LINPUT on the Commodore Pet with "upgrade" ROMs. Whatever the user types will be returned in the basic variable IN$. The program is completely relocatable, and occupies 305 bytes. The program is presented in assembler, and as a Basic loader. You may locate the program anywhere in memory, or have the loader program POKE it into the end of memory, and adjust the necessary pointers to protect it from being erased by string storage.

Program 1.

```
                0010           .OS
                0020           .BA $027A
                0030 LEN       .DE $B1      ;LENGTH OF STRING
                0040 BUL       .DE $B3      ;POINTER TO BUFFER
                0050 CUR       .DE 167      ;CURSOR STATUS
                0060 MAX       .DE 37       ;MAX # OF CHAR
                0070 BOS       .DE $30      ;BOTTOM OF STRINGS PTR
                0080 GCOLL     .DE $D400    ;GARBAGE COLLECT
                0090 CHRREC    .DE $B5      ;LAST KEY RECEIVED
                0100 TESTDIR   .DE 53888    ;TEST DIRECT MODE
                0110 VAR       .DE $2A      ;START OF VARIABLES PTR
                0120 SEARCH    .DE $B7      ;START OF SEARCH PTR
                0130 ENDV      .DE $2C      ;END OF VARIABLES PTR
                0140 ENDS      .DE $B9      ;END OF SEARCH PTR
                0150 SLEN      .DE $BB      ;FINAL STRING LENGTH
                0160 SNAME     .DE $42      ;VARIABLE NAME
                0170 ADDSTR    .DE $D001    ;ADD A VARIABLE
027A- 20 80 D2  0180           JSR TESTDIR  ;EXIT IF DIRECT MODE
027D- A9 02     0190           LDA #$2      ;BUFFER POINTER = $0200
027F- 85 B4     0200           STA *BUL+1
0281- A9 00     0210           LDA #00
0283- 85 B3     0220           STA *BUL
0285- A9 8D     0230           LDA #141
0287- 20 D2 FF  0240           JSR $FFD2    ;DO CRLF
028A- A9 3E     0250           LDA #62
028C- 20 D2 FF  0260           JSR $FFD2    ;PRINT PROMPT
028F- A9 00     0270           LDA #0
0291- 85 B1     0280           STA *LEN     ;SET LENGTH TO ZERO
```

80 COLUMN GRAPHICS

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

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

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

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

Mastercharge and Visa accepted

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

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

NOW 80 COLUMN PETS CAN HAVE MTU HIGH RESOLUTION GRAPHICS

```
0293- 85 A7              STA *CUR      ;TURN ON CURSOR
0295- 20 E4 FF   GET     JSR $FFE4
0298- F0 FB              BEQ GET       ;WAIT FOR A KEY
029A- 85 B5              STA *CHRREC   ;SAVE KEY PRESSED
029C- C9 0D              CMP #13       ;RETURN KEY?
029E- D0 19              BNE SKIP1     ;BRANCH IF NOT
02A0- A4 B1              LDY *LEN      ;IS LENGTH ZERO?
02A2- F0 15              BEQ SKIP1     ;BRANCH IF YES
02A4- A9 01              LDA #$01
02A6- 85 A7              STA *CUR      ;TURN OFF CURSOR
02A8- A9 92              LDA #146
02AA- 20 D2 FF           JSR $FFD2     ;PRINT RVS/OFF
02AD- A9 20              LDA #32
02AF- 20 D2 FF           JSR $FFD2     ;PRINT SPACE
02B2- A9 8D              LDA #141
02B4- 20 D2 FF           JSR $FFD2     ;DO CRLF
02B7- D0 4B              BNE ADDVAR    ;BRANCH ALWAYS
02B9- A5 B1      SKIP1   LDA *LEN      ;CHECK STRING LENGTH
02BB- C9 25              CMP #MAX      ;MAXIMUM LENGTH?
02BD- D0 06              BNE SKIP2     ;NO
02BF- A5 B5              LDA *CHRREC   ;YES, CHECK CHAR
02C1- C9 14              CMP #20       ;DELETE?
02C3- D0 D0              BNE GET       ;NO, NOT DELETE
02C5- 18         SKIP2   CLC
02C6- A5 B5              LDA *CHRREC
02C8- C9 41              CMP #65       ;IS CHAR 65 OR MORE
02CA- F0 04              BEQ OK1       ;YES
02CC- B0 02              BCS OK1
02CE- D0 12              BNE SKIP3     ;IT WAS LESS THAN 65

02D0- 18         OK1     CLC
02D1- C9 5B              CMP #91       ;IS IT LESS THAN 91?
02D3- B0 0D              BCS SKIP3     ;NO
02D5- A4 B1      OK2     LDY *LEN
02D7- A5 B5              LDA *CHRREC
02D9- 91 B3              STA (BUL),Y   ;STORE CHAR IN BUFFER
02DB- E6 B1              INC *LEN      ;INCREMENT LENGTH
02DD- 20 D2 FF   DEL     JSR $FFD2     ;PRINT CHARACTER
02E0- D0 B3              BNE GET       ;BRANCH ALWAYS
02E2- A5 B1      SKIP3   LDA *LEN      ;CHECK THE LENGTH
02E4- F0 06              BEQ SKIP4     ;BRANCH IF LENGTH 0
02E6- A5 B5              LDA *CHRREC   ;IS IT A SPACE?
02E8- C9 20              CMP #32
02EA- F0 E9              BEQ OK2       ;YES, THE SPACE IS OK
02EC- A5 B5      SKIP4   LDA *CHRREC   ;IS IT A DIGIT?
02EE- C9 30              CMP #48
02F0- 90 04              BCC SKIP5
02F2- C9 3A              CMP #58
02F4- 90 DF              BCC OK2       ;YES, IT'S A DIGIT
02F6- C9 14      SKIP5   CMP #20       ;IS IT A DELETE?
02F8- D0 9B              BNE GET       ;BRANCH IF IT ISN'T
02FA- A5 B1              LDA *LEN      ;CHECK THE LENGTH
02FC- F0 97              BEQ GET       ;BRANCH IF IT'S ZERO
02FE- C6 B1              DEC *LEN      ;DECREMENT LENGTH
0300- A9 14              LDA #20       ;LOAD A WITH DELETE
0302- D0 D9              BNE DEL       ;BRANCH ALWAYS
0304- 20 00 D4   ADDVAR  JSR GCOLL     ;GARBAGE COLLECT
```

```
0307- C6 B1     0850           DEC *LEN
0309- A4 B1     0860 CHECKSP   LDY *LEN
030B- B1 B3     0870           LDA (BUL),Y
030D- C9 20     0880           CMP #$20      ;PEEL OFF TRAILING SPACES
030F- D0 04     0890           BNE MOVEIT
0311- C6 B1     0900           DEC *LEN      ;DECREMENT LENGTH COUNTER
0313- D0 F4     0910           BNE CHECKSP   ;BRANCH ALWAYS
0315- A4 B1     0920 MOVEIT    LDY *LEN
0317- C8        0930           INY
0318- 84 BB     0940           STY *SLEN     ;SAVE CURRENT LENGTH
031A- C6 30     0950 MOVE1     DEC *BOS      ;DEC STRING PTR
031C- A5 30     0960           LDA *BOS
031E- C9 FF     0970           CMP #255
0320- D0 02     0980           BNE POS
0322- C6 31     0990           DEC *BOS+1
0324- A4 B1     1000 POS       LDY *LEN
0326- B1 B3     1010           LDA (BUL),Y
0328- A0 00     1020           LDY #$00
032A- 91 30     1030           STA (BOS),Y   ;MOVE FROM INPUT
032C- C6 B1     1040           DEC *LEN      ; BUFFER TO BOTTOM OF
032E- A5 B1     1050           LDA *LEN      ; STRINGS
0330- C9 FF     1060           CMP #255
0332- D0 E6     1070           BNE MOVE1
0334- A9 01     1080           LDA #$01
0336- 85 A7     1090           STA *CUR      ;MAKE SURE CURSOR OFF
0338- A5 2A     1100 LINKTAB   LDA *VAR      ;SET SEARCH PTR
033A- 85 B7     1110           STA *SEARCH   ;TO LOCATIONS POINTED
033C- A5 2B     1120           LDA *VAR+1    ;TO BY START OF
```

```
033E- 85 B8     1130            STA *SEARCH+1        ;VARIABLES POINTER
0340- A5 2C     1140            LDA *ENDV   ;SET END-OF-SEARCH
0342- 85 B9     1150            STA *ENDS   ;POINTER TO LOCATIONS
0344- A5 2D     1160            LDA *ENDV+1 ;POINTED TO BY END-OF-
0346- 85 BA     1170            STA *ENDS+1 ;VARIABLES POINTER
0348- A0 00     1180 HUNT       LDY #$00    ;HUNT FOR IN$ IN TABLE
034A- B1 B7     1190            LDA (SEARCH),Y
034C- C9 49     1200            CMP #$49    ;FOUND I?
034E- D0 18     1210            BNE INCSEVEN         ;NO
0350- C8        1220            INY
0351- B1 B7     1230            LDA (SEARCH),Y
0353- C9 CE     1240            CMP #$CE    ;FOUND N?
0355- D0 11     1250            BNE INCSEVEN         ;NO
0357- A5 BB     1260            LDA *SLEN
0359- C8        1270            INY
035A- 91 B7     1280            STA (SEARCH),Y       ;SET LENGTH OF IN$
035C- C8        1290            INY
035D- A5 30     1300            LDA *BOS
035F- 91 B7     1310            STA (SEARCH),Y       ;POINTER TO STRING
0361- C8        1320            INY
0362- A5 31     1330            LDA *BOS+1
0364- 91 B7     1340            STA (SEARCH),Y
0366- D0 28     1350            BNE RET     ;BRANCH ALWAYS
0368- A2 07     1360 INCSEVEN   LDX #$07    ;INCREMENT SEARCH
036A- E6 B7     1370 NBYTE      INC *SEARCH ; POINTER BY 7
036C- D0 02     1380            BNE SAMEPAGE
036E- E6 B8     1390            INC *SEARCH+1
0370- CA        1400 SAMEPAGE   DEX
0371- D0 F7     1410            BNE NBYTE
0373- A5 B8     1420 DONE?      LDA *SEARCH+1        ;CHECK FOR END
0375- C5 BA     1430            CMP *ENDS+1 ; OF VARIABLE TABLE
0377- 90 CF     1440            BCC HUNT
0379- D0 06     1450            BNE ADDIN
037B- A5 B7     1460            LDA *SEARCH
037D- C5 B9     1470            CMP *ENDS
037F- 90 C7     1480            BCC HUNT    ;MORE TO GO
0381- A9 49     1490 ADDIN      LDA #$49    ;ADD IN$ TO TABLE
0383- 85 42     1500            STA *SNAME  ;SET UP NAME OF STRING
0385- A9 CE     1510            LDA #$CE
0387- 85 43     1520            STA *SNAME+1
0389- 20 01 D0  1530            JSR ADDSTR
038C- F0 AA     1540            BEQ LINKTAB ;NOW GO LOOK FOR IT
038E- D0 A8     1550            BNE LINKTAB
0390- A5 2C     1560 RET        LDA *ENDV   ;POINTERS OK?
0392- C5 2A     1570            CMP *VAR
0394- D0 15     1580            BNE BASIC   ;YES
0396- A5 2D     1590            LDA *ENDV+1
0398- C5 2B     1600            CMP *VAR+1
039A- D0 0F     1610            BNE BASIC
039C- A2 07     1620            LDX #$07
039E- E6 2C     1630 ADJUST     INC *ENDV   ;NO, FIX POINTERS
03A0- E6 2E     1640            INC *ENDV+2
03A2- D0 04     1650            BNE SAPAGE
03A4- E6 2D     1660            INC *ENDV+1
03A6- E6 2F     1670            INC *ENDV+3
03A8- CA        1680 SAPAGE     DEX
```

```
03A9- D0 F3    1690           BNE ADJUST
03AB- 60       1700 BASIC     RTS  ;RETURN TO BASIC
               1710           .EN
```

Program 2.

```
100 REM    *** RELOCATABLE LINE INPUT
110 REM    *** FOR UPGRADE ROM PETS
120 REM
130 REM    *** BY ROBERT LANDO
140 REM       146 VAN HORNE AVENUE
150 REM       TORONTO, CANADA
160 REM
170 REM
180 PRINT"▯▩LINE INPUT■                      ROBERT LANDO"
190 PRINT"▩▩PLEASE ENTER THE DECIMAL LOCATION THAT"
200 PRINT"▩YOU WOULD LIKE THE LINE INPUT PROGRAM"
210 PRINT"▩TO START AT. IF YOU ENTER AN ASTERISK,"
220 PRINT"▩THE PROGRAM WILL BE PACKED INTO THE"
230 PRINT"▩END OF AVAILABLE MEMORY AND THE"
240 PRINT"▩NECESSARY POINTERS WILL BE ADJUSTED TO"
250 PRINT"▩PROTECT IT FROM BEING OVERWRITTEN BY"
260 PRINT"▩STRINGS.▩▩"
270 INPUT"   *■■■";S$
280 IFS$="*"THEN310
290 S=VAL(S$):IFS<512ORS>36559THENPRINT"▯";:GOTO270
300 E=S+305:GOTO330
310 M=PEEK(52)+PEEK(53)*256:E=M:S=M-305
320 M=S:POKE53,M/256:POKE52,M-256*PEEK(53)
330 FORX=STOE:READV:POKEX,V:NEXTX
340 PRINT"▯THE LINE INPUT PROGRAM IS NOW IN"
350 PRINT"▩MEMORY. WHEN YOU USE THE COMMAND"
360 PRINT"▩SYS (";S;") IN A PROGRAM, WHATEVER"
370 PRINT"▩THE USER ENTERS WILL BE TRANSFERRED"
380 PRINT"▩TO THE BASIC VARIABLE IN$"
390 END
400 DATA32,128,210,169,2,133,180,169,0,133,179,169,141,32,210,255,169
410 DATA62,32,210,255,169,0,133,177,133,167,32,228,255,240,251,133,181
420 DATA201,13,208,25,164,177,240,21,169,1,133,167,169,146,32,210,255,169
430 DATA32,32,210,255,169,141,32,210,255,208,75,165,177,201,37,208,6
440 DATA165,181,201,20,208,208,24,165,181,201,65,240,4,176,2,208,18,24
450 DATA201,91,176,13,164,177,165,181,145,179,230,177,32,210,255,208
460 DATA179,165,177,240,6,165,181,201,32,240,233,165,181,201,48,144,4,201
470 DATA58,144,223,201,20,208,155,165,177,240,151,198,177,169,20,208
480 DATA217,32,0,212,198,177,164,177,177,179,201,32,208,4,198,177,208
490 DATA244,164,177,200,132,187,198,48,165,48,201,255,208,2,198,49,164
500 DATA177,177,179,160,0,145,48,198,177,165,177,201,255,208,230,169,1
510 DATA133,167,165,42,133,183,165,43,133,184,165,44,133,185,165,45,133
520 DATA186,160,0,177,183,201,73,208,24,200,177,183,201,206,208,17,165
530 DATA187,200,145,183,200,165,48,145,183,200,165,49,145,183,208,40,162
540 DATA7,230,183,208,2,230,184,202,208,247,165,184,197,186,144,207,208
550 DATA6,165,183,197,185,144,199,169,73,133,66,169,206,133,67,32,1
560 DATA208,240,170,208,168,165,44,197,42,208,21,165,45,197,43,208,15
570 DATA162,7,230,44,230,46,208,4,230,45,230,47,202,208,243,96
READY.
```

©

Measure Time Intervals With The Pet Parallel User Port

Robert Macnaughton
Rexdale, Canada

This article describes a machine language program that can be used to measure seven successive small time intervals, using the CBM Parallel User Port (PUP), and eight phototransistors, to the nearest 1/10000s.

Since no page zero locations are used, this program should run on any PET (except 4.0, since it would need to be moved above 864 decimal for 4.0 BASIC).

The PUP, located at the back of the CBM, consists of 24 contacts to the main logic board, labelled as follows:

1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	H	J	K	L	M	N

Only the bottom row of contacts will be used. The top row of contacts are for use by CBM diagnostic routines during servicing.

On the bottom row of contacts, Pin M is the CB2 line, used in many programs for sound effects; contacts A and N are grounds, and contact B is the CA1 line.

We will use contacts C,D,E,F,H,J,K and L, known as PA0, PA1, PA2, PA3, PA4, PA5, PA6 and PA7, the programmable input/output lines, to receive information from eight phototransistors, the detectors of the position of some moving object.

The eight lines are treated by the PET as a single memory location, 59471 in decimal or $E84F in hexadecimal. It is known as the ORA, the output register for I/O Port A, without handshaking. At any time, a PEEK(59471) will indicate the condition of the ORA.

The DDR A, the data direction register for Port A, is used to designate which are the input and which are the output lines of the ORA. Its address is 59459 or $E843. A zero in bit three would make PA3 an input line and a one would make it an output line. If you POKE 59459,76 then PA2, PA3 and PA6 will be output lines and the rest input lines, since 76 in binary is 01001100.

In this timer, all eight lines are made inputs by POKE 59459,0. A PEEK(59459) when the CBM is first turned on will show that all the lines are initially inputs.

When running, the timer program looks at the contents of the ORA again and again. To understand the result, the contents of 59471 must be expressed as a binary number. Each of the eight I/O lines corresponds to one bit in this number. Any line grounded will be represented as a 0. If not grounded, it will be represented as a 1. More exactly, if a resistance of less than about 2000Ω is connected from a PA line to GND, the state of the line will be interpreted as a 0. If the resistance is more than 2000Ω, it will be interpreted as a 1.

If you PEEK(59471) with nothing connected to the PUP, you will get 255. If you short out all eight lines, you will get a 0. (First make sure that they are all input lines.)

	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
bit	7	6	5	4	3	2	1	0
value	128	64	32	16	8	4	2	1

59471								
255	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
4	0	0	0	0	0	1	0	0
8	0	0	0	0	1	0	0	0
16	0	0	0	1	0	0	0	0
32	0	0	1	0	0	0	0	0
64	0	1	0	0	0	0	0	0
128	1	0	0	0	0	0	0	0
214	1	1	0	1	0	1	1	0

The collectors of eight FPT100 phototransistor's are connected to the eight PA lines, and their emitters to ground at contact N. When enough light strikes a phototransistor such as the FPT100, its resistance falls to about 200Ω. This is interpreted as a 0 in the ORA. When the light is cut off, the resistance increases dramatically and is interpreted as a 1. As an object passes by a phototransistor, the state of that PA line will change from 0 to 1 and back to 0 as the light is temporarily interrupted.

I have placed the phototransistors in holes drilled in a meter stick 15 cm apart. The position of the first phototransistor must be adjustable to start the timer at the correct moment. Opposite each phototransistor is a small flashlight bulb attached to a second meter stick. The two meter sticks are placed on either side of a ramp. A large

PETTERM

INTELLIGENT 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.

PETTERM I	All features above	$75.00
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

forth

for PET/CBM

FORTH is a new concept in programming, with the speed of compilers and interactive ease of BASIC. Programs become a part of FORTH extending the power of FORTH and your PET.

8050,4040 disk, cassette all PET-CBMs 16k+

Starter	fig-FORTH w editor assembler	$35
Personal	floating point; strings; source	$75
Professional	turnkey development/data base	$259

F S S

software for small computers

1903 Rio Grande
Austin, Texas
78705

1-512-477-2207

P.O. Box 8403
Austin, Texas
78712

DEALERS INQUIRE

CBM/PET INTERFACES

RS-232 SERIAL PRINTER INTERFACE – addressable – baud rates to 9600 – switch selectable upper/lower, lower/upper case – works with WORDPRO, BASIC and other software – includes case and power supply.
MODEL – ADA1450 149.00

CENTRONICS/NEC PARALLEL INTERFACE - addressable – high speed – switch selectable upper/lower, lower/upper case – works with WORDPRO, BASIC and other software – has Centronics 36 pin ribbon connector at end of cable.
MODEL – ADA1600 129.00

CENTRONICS 730/737 PARALLEL INTERFACE – as above but with Centronics card edge connector at end of cable.
MODEL – ADA730 129.00

COMMUNICATIONS INTERFACE WITH SERIAL AND PARALLEL PORTS – addressable – software driven – true ASCII conversion – selectable reversal of upper-lower case – baud rates to 9600 – half or full duplex – X-ON, X-OFF – selectable carriage return delay – 32 character buffer – centronics compatible — much more.
MODEL – SADI 295.00

ANALOG TO DIGITAL CONVERTER – 16 channels – 0 to 5.12 volt input voltage range – resolution is 20 millivolts per count – conversion time is less than 100 microseconds per channel.
MODEL – PETSET1 295.00

REMOTE CONTROLLER WITH CLOCK/CALENDAR – controls up to 256 devices using the BSR X10 remote control receivers – 8 digital inputs, TTL levels or switch closure — 8 digital outputs, TTL levels.
MODEL – PETSET2 295.00

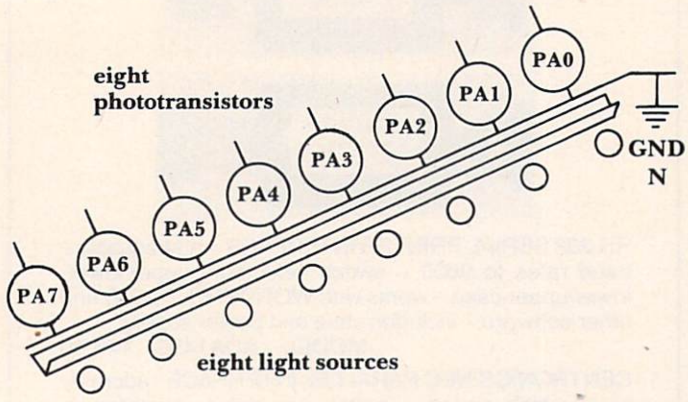
All prices are in US dollars for 120VAC.
Prices on 220 VAC slightly higher.
Allow $5.00 shipping & handling, foreign orders add 10% for AIR postage.
Connecticut residents add 7½% sales tax.
All prices and specifications subject to change without notice.
Our 30 day money back trial period applies.
MASTER CHARGE/VISA accepted.
MENTION THIS MAGAZINE WITH YOUR ORDER AND DEDUCT 5% FROM TOTAL.

IN CANADA order from: Batteries Included, Ltd., 71 McCaul Street, F6 Toronto, Canada M5T2X1, (416)596-1405.
IN THE USA order from your local dealer or direct: Connecticut microComputer, Inc., 34 Del Mar Drive, Brookfield, CT 06804, (203)775-4595.

Dealer inquiries invited.

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

ball bearing rolling down the ramp will be timed as it interrupts each light beam in turn.



If the times you wished to measure were long, you could write a BASIC program to measure these time intervals, using the internal "jiffy" clock of the PET. The light to each phototransistor would have to be cut off long enough that it would still be cut off when the program got around to checking the state of 59471.

To fully utilize the 1 megacycle clock in the CBM, a machine language program must be used.

The program begins by setting the interrupt flag. This will ensure that the timmig will not be interrupted by the CBM as it performs its normal interrupt every 1/60 s, to update its clock, flash the cursor if needed, etc.

It then goes into a loop to load all the various memory locations used to store the times, with zeros. At the same time it prints a ? at the top left of the screen. It then goes into a second loop to wait for PA? to become 1 when the ball is rolled into place at the top of the ramp. An R for READY now appears on the screen.

The following table shows how the ORA changes as the ball rolls down the ramp.

SCREEN	BINARY	DECIMAL	
?	00000000	0	ball not on ramp
R	00000001	1	ball in place at top of ramp
T	00000000	0	ball rolling
1	00000010	2	passes PA1
1	00000000	0	ball rolling
2	00000100	4	passes PA2
2	00000000	0	ball rolling
3	00001000	8	passes PA3
3	00000000	0	ball rolling
4	00010000	16	passes PA4
4	00000000	0	ball rolling
5	00100000	32	passes PA5
5	00000000	0	ball rolling
6	01000000	64	passes PA6
6	00000000	0	ball rolling
7	10000000	128	passes PA6

When 59471 becomes 0, the timer enters a timing loop. Each time through the loop it checks 59471 for a 0, then adds 1 to a counter. When 59471 has the next expected value, the contents of this counter are stored, and the timing resumes, continuing until all seven times have been measured. When the program returns to BASIC, the contents of the memory locations containing the count can be recalled and converted to seconds.

Since each timing loop takes 43 cycles of the CBM's internal 1 megacycle clock, each count represents 43 microseconds.

The count is contained in three locations. The first is incremented in each loop. The second is incremented only when the first passes 255 and becomes 0 again. The third is incremented only when the second passes 255 and becomes 0 again. The largest count possible is then (255x256x256) + (255x256) + 255 or 16777215. This is slightly more than 12 minutes.

I have included a second copy of the machine language program which shows the timing loop. Beside each step I have written the number of cycles of the PET's internal clock that are needed to complete each step. The total number of cycles is 43. Some extra time is used to store the count as each phototransistor is passed. If you wish, this could be calculated and added on to the total time as a correction.

I have also included a BASIC program to operate the clock in an organized fashion. It asks you how many runs you wish to make down the ramp, then stores the seven times for each run. Eventually, the average time for each part of the run is calculated. With a few minor changes, this program can be used in almost any situation where accurate timing is needed.

TIMER COMMENTS

1 Disable the interrupt flag
2 Load the accum with the code for ?
3 Store at top left corner of screen
4 Load the x-register with a 2
5 Store the 2 at 0336
6 Load the y-register with decimal 25
7 Load the accum with a 0
8 Store 0 at all locations from 03DF to 03DF + 25 by looping until y = 0
10 Compare y with zero
11 12 If y isn't zero then loop to step 7
12 Load accum with the contents of 59471
13 Check if PA0 is a 1 or a 0
14 If PA0 = 0 then loop and check again
15 Now PA0 is 1: R for Ready into accum
16 store R on the screen
17 This is a time delay while things
18 settle down. Load x and y with 255
19 and decrement them both to zero.
20 Each time x decrements from 255 to 0
21 y decrements by one. Finally both are
22 zero
23 Load accum with 59471 once more
24 Test to see if PA0 is still a 1
25 If so, loop back to 23 and try again
26 Now PA0 is a 0, the timing must start

27 Store a T on the screen
28 Begin timing loop by clearing the
29 carry flag, then load accum with 03E0
32 Add 1 to the contents of this
30 location and then store it back there
31 Add zero to the contents of 03E1
32 (and 1 if the carry flag was set by
33 the previous addition) and store it
34 Add zero(+ 1 if the carry flag is set)
35 to the contents of 03E2
36 03E0, 03E1, 03E2 contain the total time
37 Check 57471 to see if the next PA
38 line is a 1 or a 0 using 0336
39 0336 contains a 2: binary 00000010
40 If a 0, loop: if a 1, then arithmetic
41 shift left the value in 0336: see text
42 Store the three values representing
43 the elapsed time using the current
44 value of y (It is 0 to start with)
45
46
47
48 Clear the carry flag before addition
49 Transfer y, a counter, to the accum
50 Add 177 to it to make it the ASCII
51 code for y and store it on screen
52 Increment y (next time measurement)
53 Compare y with 7: If y is less than 7
54 then go back to start of timing cycle
55 If y is 7, the program is over: clear
56 interrupt flag and return to BASIC

Program 1.

```
TIMER                      $033A SYS 826

1 033A 78                  SEI
2 033B A9 BF               LDA #BF
3 033D 8D 00 80            STA  8000
4 0340 A2 02               LDX #02
5 0342 8E 36 03            STX  0336
6 0345 A0 19               LDY #19
7 0347 A9 00               LDA #00
8 0349 99 DF 03            STA  03DF,Y
9 034C 88                  DEY
10 034D C0 00              CPY #00
11 034F D0 F6              BNE  0347
12 0351 AD 4F E8           LDA  E84F
13 0354 29 01              AND #01
14 0356 F0 F9              BEQ  0351
15 0358 A9 92              LDA #92
16 035A 8D 00 80           STA  8000
17 035D A0 FF              LDY #FF
18 035F A2 FF              LDX #FF
19 0361 CA                 DEX
20 0362 D0 FD              BNE  0361
21 0364 88                 DEY
22 0365 D0 FA              BNE  0361
23 0367 AD 4F E8           LDA  E84F
24 036A 29 01              AND #01
25 036C D0 F9              BNE  0367
26 036E A9 94              LDA #94
27 0370 8D 00 80           STA  8000
28 0373 18                 CLC
29 0374 AD E0 03           LDA  03E0
30 0377 69 01              ADC #01
31 0379 8D E0 03           STA  03E0
32 037C AD E1 03           LDA  03E1
33 037F 69 00              ADC #00
34 0381 8D E1 03           STA  03E1
35 0384 AD E2 03           LDA  03E2
36 0387 69 00              ADC #00.
37 0389 8D E2 03           STA  03E2
38 038C AD 4F E8           LDA  E84F
39 038F 2D 36 03           AND  0336
40 0392 F0 DF              BEQ  0373
41 0394 0E 36 03           ASL  0336
42 0397 AD E0 03           LDA  03E0
43 039A 99 E3 03           STA  03E3,Y
44 039D AD E1 03           LDA  03E1
45 03A0 99 EA 03           STA  03EA,Y
46 03A3 AD E2 03           LDA  03E2
47 03A6 99 F1 03           STA  03F1,Y
48 03A9 18                 CLC
49 03AA 98                 TYA
50 03AB 69 B1              ADC #B1
51 03AD 8D 00 80           STA  8000
52 03B0 C8                 INY
53 03B1 C0 07              CPY #07
54 03B3 D0 BE              BNE  0373
55 03B5 58                 CLI
56 03B6 60                 RTS
```

Program 2.

```
READY.

 1Ø REM TIMER BASIC
 2Ø REM ROBERT MACNAUGHTON OCT 5/8Ø
 25 REM 2124 GREENHURST AVE
 3Ø REM MISSISSAUGA   L4X 1J6
 35 REM THE MACHINE LANGUAGE PROGRAM ¬
      ¬MEASURES 7 TIMES DURING A SINGLE ¬
      ¬TRIP
 4Ø REM UP TO 8 PHOTOTRANSISTORS ARE ¬
      ¬CONNECTED TO PAØ-7
 45 REM SYS 826 ACTIVATES THE TIMER AND ¬
      ¬? APPEARS
 5Ø REM WHEN PAØ IS BLOCKED OFF, R ¬
      ¬APPEARS AND THE TIMER IS READY TO ¬
      ¬START
 6Ø REM WHEN LIGHT AGAIN FALLS ON PAØ,
      ¬ THE TIMER STARTS AND T APPEARS
 7Ø REM AS EACH OF PA1-7 IS CUT OFF,
      ¬ THE TOTAL ELAPSED TIME IS STORED
 75 REM AS EACH MEASUREMENT IS MADE,
      ¬ ITS NUMBER APPEARS (1-7)
 8Ø REM UNUSED PA LINES SHOULD BE OPEN ¬
      ¬CIRCUITS
 2ØØ PRINT"ĥ"
```

```
205 INPUT"↓NUMBER OF RUNS";NR
21Ø FORJ=1TONR
215 SYS826
22Ø FOR I=ØTO6
225 REM THE NEXT STATEMENT CALCULATES ¬
    ¬THE TIMES
226 REM THE MEMORY LOCATIONS FOR THE ¬
    ¬TIMES ARE (995,1ØØ2,1ØØ9)(996,
    ¬1ØØ3,1Ø1Ø),
227 REM CONTINUING UP TO (1ØØ1,1ØØ8,
    ¬1Ø15)
228 REM EACH TIMING CYCLE TAKES 43 ¬
    ¬MACHINE LANGUAGE STEPS OR 43 ¬
    ¬MICROSECONDS
23Ø T(I,J)=43*(PEEK(995+I)+PEEK(1ØØ2+I)*
    ¬256+PEEK(1ØØ9+I)*256*256)/1ØØØØØØ
24Ø REM THE NEXT STATEMENT ROUNDS OFF ¬
    ¬THE TIMES TO 1/1ØØØØ S
25Ø T(I,J)=INT(T(I,J)*1ØØØØ)/1ØØØØ
26Ø PRINT T(I,J),
27Ø NEXT:PRINT:PRINT:NEXT
28Ø REM CALCULATE THE AVERAGE TIMES
29Ø PRINT"AVERAGE TIMES"
3ØØ FOR I=ØTO6:TM(I)=Ø:FOR J=1TONR
31Ø TM(I)=TM(I)+T(I,J)
32Ø AV(I)=TM(I)/NR
33Ø AV(I)=INT(AV(I)*1ØØØØ)/1ØØØØ
34Ø NEXT:NEXT
35Ø FOR I=ØTO6:PRINTAV(I),:NEXT:PRINT
4ØØ GOTO 2Ø5
```

©

COMPUTE!'s Listing Conventions

Many programs which are listed in **COMPUTE!** use cursor control keys, color keys, and so forth. We have established a listing convention which we believe eases the task of typing programs in accurately.

PET/CBM/VIC Conventions

Generally, PET/CBM/VIC programs will contain bracketed words for any special characters: [DOWN] means the cursor-down key; [3 DOWN] means type the cursor-down key three times.

If a program line runs over onto the next line down, the ~ symbol indicates where the line broke (in case the number of spaces is unclear between quotes). An underline means that that key is shifted.

8032/Fat 40 Conventions

SET WINDOW TOP	[SET TOP]
SET WINDOW BOTTOM	[SET BOT]
SCROLL UP	[SCR UP]
SCROLL DOWN	[SCR DOWN]
INSERT LINE	[INST LINE]
DELETE LINE	[DEL LINE]
ERASE TO BEGINNING	[ERASE BEG]
ERASE TO END	[ERASE END]
TOGGLE TAB	[TGL TAB]
TAB	[TAB]
ESCAPE KEY	[ESC]

All Commodore Machines

CLEAR SCREEN	[CLEAR]
HOME CURSOR	[HOME]
CURSOR UP	[UP]
CURSOR DOWN	[DOWN]
CURSOR RIGHT	[RIGHT]
CURSOR LEFT	[LEFT]
INSERT CHARACTER	[INST]
DELETE CHARACTER	[DEL]
REVERSE FIELD ON	[RVS]
REVERSE FIELD OFF	[OFF]

Review:

Screen Pro

Edward K. Crossman, Ph.D.
Logan, UT

As a behavioral scientist with both teaching and research responsibilities, I am always looking for ways to save time. Yet, so many of the applications touted by those in the computer field could be more quickly accomplished with a pencil and a stack of 3x5 cards. However, Screen Pro from Kansas City Computers, Inc. has saved me time, and, in this article, I will describe some of the features of this program, and how I applied it.

My home system consists of a CBM 8032 computer, a CBM 8050 disk drive, and an Epson MX-80 printer.

The data from my behavioral experiments (and many others) are expressed in a relative frequency distribution: 10% of the subjects exhibited behavior X, 20% exhibited behavior Y, etc. Typically, these data are represented in tabular form. When you have multiple experimental conditions and, thus, many tables, it is difficult to see trends or changes in the data. So, a bar graph, or histogram (see Figure 1) is often used instead. It is much easier to visually scan a series of histograms and detect changes in the data than with many tables. But how could I convert my frequency data into a histogram quickly and simply and have it printed out on my MX-80? That is where Screen Pro came to the rescue.

Screen Pro, written in BASIC and machine language, allows you to create text or graphics on the screen, and then prints out what was on the screen. Also, it can save the information as a screen file on disk; each screen file occupies eight blocks on disk. Once the screen file is on disk it can be recalled and edited as you see fit. I have found the editing functions adequate for my purposes. These functions for the CBM 8032 (in addition to the normal editing functions) include:

Set upper case
Set lower case
Scroll screen down
Expand Screen
Compress Screen
Insert a line
Delete a line
Erase to end of line
Erase from beginning of line
Enbable/disable quote mode (for graphics)
Send screen contents to printer (normal mode)
Send screen contents to printer (squeezed)
Abort current file on screen, retaining original if editing
Normal exit of editor

Screen Pro has some other nice features. When editing a screen file, it uses a temporary scratch space on disk, so you can either save or scrap the screen file you are currently editing, and at the same time keep a backup of the original screen file as it was before you started editing. Also the author of Screen Pro, Keith Peterson, has gone to some trouble to explain how, if you understand BASIC programming, to create your own programs using his machine language subroutine. Being an amateur programmer, I don't fully grasp his instructions, but perhaps you can. He has taken a very refreshing approach by not protecting any of the software, so you can examine it and change it to your specifications. Essentially, if you write your own program, you can create hundreds of screen files with the ability to branch from one file to many others in the series. Mr. Peterson has provided the would-be programmer with several demonstrations of how to do this. In essence, then, you could create a sort of information management program, although it would lack many features of a typical data base management program, such as Create-A-Base, or Commodore's Ozz.

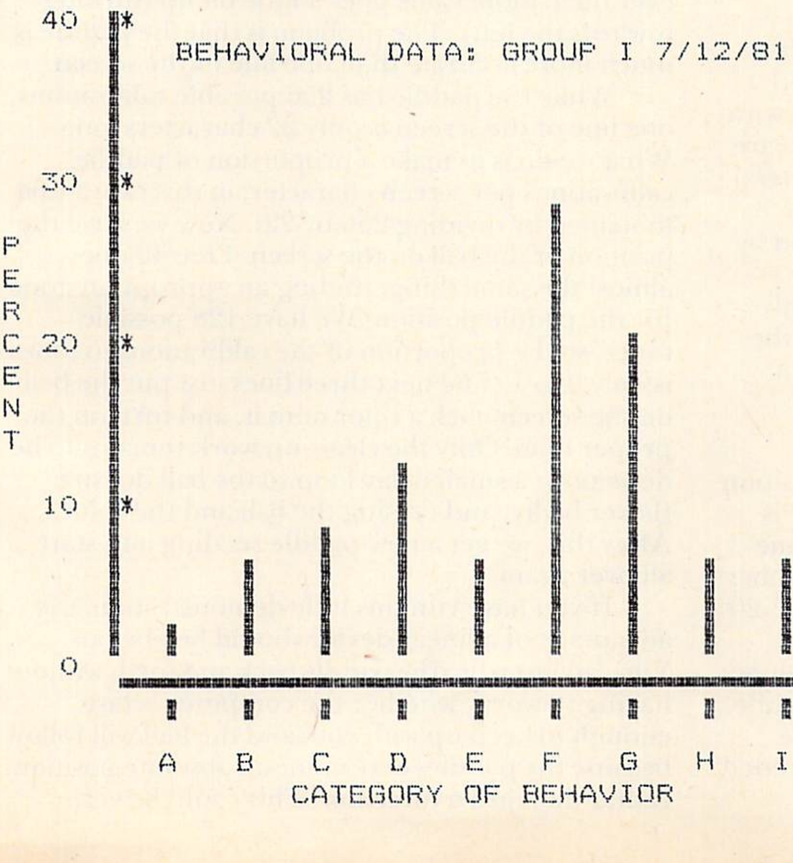
There are a couple of things I have not been happy with, however. First, the documentation, while better than some I have seen, is written by the programmer for people who already have some knowledge of programming and computers. My impression of many in the micro industry is that they consider quality documentation unimportant, yet there are people around who are trained to explain computerese in everyday language; usually, however, they are not programmers!

The second problem concerns the Epson printer. Screen Pro was designed only for the Commodore 2022 and 4022 printers, and the author states so and shows you the section of the program where you can write a routine for your own printer. Without such a routine, however, when text is placed on the screen the Epson prints upper case characters as Epson graphic characters; lower case characters are printed correctly, but capitalized. This is a serious problem for me, and I guess the only thing to do is to write a routine to handle the case conversion.

The Epson also presents a problem for graphics applications, such as mine, because it lacks the Commodore graphics characters. However, I did figure out a simple solution for my histogram application. When one of the Commodore graphics characters is placed on the screen and printed on the Epson, it comes out as one of the Epson block-type graphics characters. So I placed all of the

Commodore graphics characters, available from the 8032 keyboard, on the screen and then printed these on the Epson. From the mish-mash that resulted I was able to pick out suitable symbols to create the simple vertical and horizontal lines that are required in constructing a histogram. For example, the letter "J" on the 8032 keyboard produces a vertical bar on the Epson when the 8032 is in the graphics mode. This method produces funny pictures on the 8032 screen, but this has proved to be only a minor annoyance since there is enough resemblance to the final product produced by the Epson.

Figure 1. Histogram produced by Screen Pro



Overall, Screen Pro is a welcome addition to the software collection for the Commodore systems, and at $39.95 is very reasonably priced. It is fast, relatively easy to use, and fills a very specific need that I have had for some time. Now, if someone would just come up with a similar package which would allow the user to produce a graph rapidly, such as a sine wave, on the screen and have the image come out on an X-Y plotter, such as the MIPLOT! ©

VIC-20 Update

Extended VIC-20 Input Devices: Paddles And The Keyboard

Mike Bassman and Salomon Lederman
Woodside, NY

The VIC-20 has some remarkable capabilities not documented by the manual. Specifically, you can use game paddles with the VIC-20 as well as making better use of the keyboard.

The Paddles

Have you ever seen the little nine pin port right next to the power switch? This port can be used with paddles. To make life easy, it can be used with the widely available Atari game paddles (which are used with their video games and home computers). Just plug in a pair, and we'll be ready to begin. These paddles are *linear* devices. What is meant by this is that the paddle is a much more sensitive device than a directional joystick, which can only point in eight or so directions. You may think the paddle is not even as good, pointing only left or right. This is not true.

How It Works

What the paddle actually does is isolate one position out of the 256 possible ones. When the paddle is turned to the far right, this value is 0. Every time you turn the paddle in either direction the number is increased or decreased accordingly. The VIC-20 allows us to use up to two paddles. For each of them, we can obtain a position value. These values are in memory locations $9008 for the first paddle, and $9009 for the second. In decimal these are 36872 and 36873, respectively (A number preceded by a "$" signifies that it is hexadecimal).

How To Do It

Shown below is a quick one liner that prints out the values of both the paddle registers.

```
10 PRINT PEEK(36872);PEEK(36873): GOTO 10
```

Try typing and running this program now. You should see a continuous stream of two numbers flying by. Fiddle with the paddles. The numbers should change accordingly. The more you turn a paddle left, the higher the number goes (the opposite for right, of course).

Next, we'll try something a little more complicated and which might be more applicable. Program 1 will move a little ball across the screen according to your paddle position. It will also slide a musical tone up and down at the same time. Here are some notes which will explain some of the program. The first two lines are just set-up, setting volume for the tone generator and clearing the screen. Line 20 gets the initial paddle position. The next line, 30, determines the position of the ball on the screen. The ball can move from the far left edge of the screen (7900) to the far right (7921). Logically, the thing to do is to move the ball a little bit left whenever the paddle value goes a little bit up (turning towards the left). The problem is that the paddle is much more accurate than one line of the screen.

While the paddle has 256 possible calibrations, one line of the screen is only 22 characters long. What we do is to make a proportion of paddle calibrations per screen character, in this case 11.64 (obtained by dividing 256 by 22). Now we have the position of the ball on the screen. Line 40 does almost the same thing, finding an appropriate tone for the paddle position. We have 128 possible tones, so the proportion of the calibrations to tones is only 2 to 1. The next three lines just put the ball on the screen, tack a color onto it, and turn on the proper tone. Only the clean-up work remains to be done now: a small delay loop so the ball doesn't flicker badly, and erasing the ball and the color. After this, we get a new paddle reading and start all over again.

If you have run this little demonstration, the advantage of a linear device should be obvious. You can just whip the paddle back and forth without having to worry whether the computer is fast enough to keep up with you, and the ball will follow because the paddles determine an absolute position, rather than just a direction. This could be very

convenient in games where speed is at a premium. In the near future, I'm sure we'll see many clever and innovative ways to use the paddles.

The Keyboard

There are two types of keyboards: ASCII, or hardware keyboards, and *polled,* or software keyboards. The ASCII keyboard is a separate device from the computer which just sends out the ASCII value of the key being pressed. The polled keyboard is a little more subtle. A polled keyboard is split up into sections of eight keys, called rows. Generally, a polled keyboard has eight rows. The computer can test one row at a time, and detect which key along with row is being pressed, if any. The polled keyboard can also detect any number of key combinations along any particular row. Consequently, polled keyboards need a fair amount of system software to do what comes naturally to an ASCII keyboard.

Most microcomputers today, the VIC-20 included, use polled keyboards because of the added flexibility and lower price. Unfortunately, the VIC-20 does not let us normally get at some of those nice features. To us, from BASIC, it seems just like an ASCII keyboard. We can only obtain one character at a time using the GET command. If two keys are being pressed down at once, the GET command will almost randomly choose one of those two as the value that gets sent back to the user. If you wanted to do a two player game or a game requiring simultaneous depressing of more than one key, life would be very difficult. But here's how it can be done.

Polled Keyboard Encoding

The VIC-20 polled keyboard has eight rows of eight keys each. Each row can be selected by a particular value. The eight values for the eight rows are all shown in Table 1. These values are by no means arbitrary. If you examine the table, you can see that the values are given in binary, as well as decimal and hexadecimal. Row values were made by turning on all the bits in the byte, then turning off the bit which the row represents. For example, the first row has all the bits on (set to 1) except for the one on the far left, which is off (or 0). Then this binary number is simply used in its hexadecimal or decimal form to represent the row. Each key along the row is handled in exactly the same manner as the rows (for example, the value representing the first row would be the same as the one representing the first key in that row). This is a little confusing, but it works out well in the end. Table 2 is the keyboard encoding matrix. It shows all the row values going down, and all the keys along each row, and their value. For instance, the keys on row 223 are F3, =,:,K,H,F,S, and Commodore. The value of the Commodore key would be 254.

Implementing Keyboard Theory

Using an individual row on the keyboard is accomplished as follows. You select a row by POKEing its value into a memory location we'll call the row select register. Then you can get the information as to which key(s) is hit by PEEKing another location, the keyboard data register. The row select register is located at $9120 (37152), and the data register at $9121 (37153).

Things don't work out as easily as doing just one POKE, then another PEEK. The problem, in this case, is the RUN STOP ROUTINE. This part of BASIC is the one that checks if you hit this key during the execution of a program. If you have, the program stops. What the routine does is, after every command executed, it puts a 247 in the row select register (the row which has the RUN STOP key) and checks the data register for a value of 254 (eighth key over). If the data register is 254, then you have hit the RUN STOP key, and program execution terminates.

What this means for us is that, even after we have just chosen a row by POKEing a value into the select register, the RUN STOP routine will change it right back to a 247. Very bad news indeed, unless you only want to use row 247. Not only that, but you can't use the RUN STOP key for your own purposes. There is a way to disable the RUN STOP key. POKEing 808 with 114 turns off the RUN STOP key, and POKEing 808 with 112 turns it back on again. This does not solve our problem. Turning off the RUN STOP key will prevent it from ending program execution when that key is hit, but the routine still stores that 247 in the select register. However, when we clear up the major problem, turning off the RUN STOP key will allow us to use that key in our programs.

A Solution

The way to solve this problem is by noticing that this routine operates after every BASIC command. What must be done is to POKE in our select value, then PEEK the data register, all in the time of less than one BASIC command. Machine language is the answer. The VIC-20 can use machine language even though it has no direct facilities for entering or saving it. *[See Jim Butterfield's Tinymon in* **COMPUTE!** *#20, pg. 176 which provides a monitor for VIC – Ed.]* We are going to use a very short machine language routine that simply puts our row into the select register, looks at the data register, then puts the contents of the data register into a RAM location that the BASIC program can look into. Program 2 shows just such a machine language program. Not much to it at all, just five lines OF CODE. The first instruction loads in the row value, in this case a $7F (127). The second stores it in the select register. The next picks up a value from the data register. The last two just store that value in accessible RAM (at $1DFF, or 7679), then returns back to BASIC.

This routine will do the trick because it does what we want in less than one BASIC command. Even though the VIC-20 has no real method for entering machine code data, it can be done anyway. You just take the machine code values, convert them into decimal, and stick them into a BASIC DATA statement. Then just add a line of BASIC that reads the values and puts them in the correct place. In Program 3, we have a complete deomonstration. Lines 30 and 40 are the aforementioned DATA statement and reader/POKER. Line 5 turns off the RUN STOP.

Lines 10 and 20 need a little bit of explanation. We are going to put the machine language routine into the top of available memory. Unfortunately, BASIC also wants to use this space. These lines tell BASIC not to use the highest 21 bytes of RAM. Locations 51 and 52, as well as 55 and 56 contain the top of BASIC RAM in low, high format. Low, high format is when the low byte of an address

Table 1.

127 - 7F - 0111 1111
191 - BF - 1011 1111
223 - DF - 1101 1111
239 - EF - 1110 1111
247 - F7 - 1111 0111
251 - FB - 1111 1011
253 - FD - 1111 1101
254 - FE - 1111 1110

Table 2: Keyboard Matrix Table

Row (PEEK) → of Keys; Column (POKE) ↓

Column (POKE)	127	191	223	239	247	251	253	254
127	F7	Home	—	Ø	8	6	4	2
191	F5		@	O	U	T	E	Q
223	F3	=	:	K	H	F	S	COMMODORE
239	F1	RIGHT SHIFT	.	M	B	C	Z	SPACE
247	CURSOR	/	,	N	V	X	LEFT SHIFT	RUN STOP
251	CURSOR	;	L	J	G	D	A	CTRL
253	RETURN	*	P	I	Y	R	W	
254	DEL	£	+	9	7	5	3	1

precedes the high byte of it. To calculate an address from this format, just use this formula: (256*high byte + low byte = address). Normally the low and high byte for the top of BASIC are 00 and 30, respectively (yielding an address of 7680). These we change to 235 and 29, giving an address of 7659. Line 50 goes to our machine code routine, line 60 prints the result, and 70 repeats the process. Try it now. I'll wait. If you press one of the keys from the first row, the appropriate value will be printed. No key is indicated by its printing 255. As it is now, this program will print first row values. To change the row, just change the second item of data in line 30. I used this program, incidentally, to make the keyboard matrix chart.

All this may seem pretty useless to you at this point. Our next program will do something that cannot be done with regular old BASIC. Program 4 will play a tone of varying pitch depending on which of two keys you hit. Doesn't sound too exciting, but it will play the two tones one after the other even if both keys are pressed at the same time. This is the basis of two-player games, where the computer can fairly give one turn to each player. All the material in this program should be old hat to you now, so I won't bother to explain it.

Hopefully you've learned to use your paddles and keyboard now. Put them to good use!

Program 1.

```
1 REM Listing 1
5 POKE36878,3
10 PRINT"[Shift/Home]"
20 X=PEEK(36872)
30 L=7921-INT(X/11.64)
40 T=255-INT(X/2):IF T=255 THEN T=
   254
50 POKEL,81
55 POKEL+30720,2
60 POKE36874,T
70 FOR K=1 TO 10:NEXT
80 POKEL,32:POKEL+30720,1
90 GOTO20
OK
```

Program 2.

```
A9 7F    LDA #$7F
8D 20 91 STA $9120
AD 21 91 LDA $9121
8D FF 1D STA $1DFF
60       RTS
```

Program 3.

```
5 POKE 808,114
10 POKE 51,235:POKE 52,29
20 POKE 55,235:POKE 56,29
30 DATA 169,127,141,32,145,173,33,
   145,141,255,29,96
40 FOR K=1 TO 12:READ X:POKE 7659
   +K,X:NEXT K
50 SYS 7660
60 PRINT PEEK(7679);
70 GOTO50
OK
```

Program 4.

```
1 REM Listing 4
10 POKE808,114:POKE51,235:POKE52,29
   :POKE55,235:POKE56,29:POKE36878,
   3
20 DATA169,127,141,32,145,173,33,
   145,141,255,29,96
30 FOR K=1 TO 12:READ X:POKE 7659
   +K,X:NEXT K
40 POKE 7661,127:SYS 7660
50 IF PEEK(7679)=254THENPOKE36874,
   200:FORK=1TO500:NEXTK:POKE36874,
   0
60 POKE7661,191:SYS 7660
70 IF PEEK(7679)=127THENPOKE36875,
   200:FORK=1TO500:NEXTK:POKE36875,
   0
80 GOTO 40
OK
```

©

Timekeeping

Keith Schleiffer
Annandale, VA

The friendly computer guide that comes in the box with your VIC 20 mentions several interesting features that the casual reader can easily miss. In my most recent rereading, I discovered the timekeeping feature of the VIC. The computer can keep real clock time, count elapsed time, or time controlled pauses during program execution.

The clock is available as the reserved variables TI and TI$. TI actually counts time passing. TI$ is a string variable, which depicts this time count in HHMMSS format (hours, minutes, and seconds, without any punctuation) on a twenty-four hour clock.

How does the VIC do this timekeeping? When the computer is first turned on, the timekeeper initializes at 000000 (midnight). You can then set it to act as a clock by assigning to TI$ a string representing the correct time. For instance, if I initialize the timekeeper as a clock at 1:29:30 in the afternoon, I would enter the statement:

```
TI$ = "132930"
```

The VIC would convert this to 48570 seconds after midnight, multiply by sixty, and assign:

```
TI = 2914200
```

and continue counting from there. TI is counted in one-sixtieth second intervals; that is, when TI has increased by sixty, one second has passed. The time count is kept in memory locations 160, 161, and 162.

Once you have set the correct time, you can check it whenever you wish by entering:

```
PRINT TI$
```

and the VIC will display the time, again in HHMMSS format. I like to set TI$ to keep clock time, and check it occasionally, so my wife doesn't have to complain about getting less attention than the computer. The timekeeper can be used in programming to control operations at scheduled times during the day, such as periodic data-collection from an experiment, or to control your lights in a household security program.

To use the VIC to count elapsed time, you cannot start and stop the time counter. To get around this problem, you must run a second variable to count time in parallel with TI, then stop counting with that second variable when the timed period is over. The following program uses the "hit any key" concept to start and stop timing:

```
100 GET A$ : IF A$ = " " THEN 100
110 TS=TI
120 PRINT "TIMING"
130 TC=TI : GET A$ : IF A$ = " " THEN 130
140 TE=(INT ((TC-TS) / 6 + 0.5)) / 10
150 GOSUB 400 : PRINT T$
160 END
400 REM CONVERTS SECONDS TO HH:MM:SS.S
    FORMAT
410 H1=INT(((TE /60/60/24) - (INT(TE /60/60/24)))
    * 24)
420 B1=STR$(H1)
430 H$=MID$(B$,2,2) : IF H1<10 THEN H$="0"
    + MID$(B$,2,1)
440 T3=TE - (H1 * 60 * 24)
450 M1=INT(((T3 /60/60) - (INT(T3 /60/60))) * 60)
460 B$=STR$(M1)
470 M$=MID$(B$,2,2) : IF M1<10 THEN M$="0"
    + MID$(B$,2,1)
480 T2=T3 - (M1 * 60)
490 S1=INT(((T2 /60) - (INT(TE/60))) * 60)
500 B$=STR$(S1)
510 S$=MID$(B$,2,4) : IF S1<10 THEN S$="0"
    + MID$(B$,2,3)
520 T$=H$+":"+M$+":"+S$ : RETURN
```

Line 130 converts TE to the elapsed time in seconds and rounds off to the nearest tenth. The subroutine starting at line 400 will convert this to "clock" display, complete with colons in HH:MM:SS.S format, down to tenths of seconds. A simpler approach would use TI$ by assigning to it the elapsed time value and immediately printing it:

```
140 TE=TC-TS
150 TI=TE : PRINT TI$ : END
```

You won't want to use this method if you are using TI as a real clock, or if you're relying on the timekeeper to track more than one period at once.

You can use the timekeeper for the scoreboard in a game, either by displaying stopwatch time or TI$, to show time passing, or by calculating time remaining and displaying a countdown timer. The followinng program is a version of the countdown timer.

```
100 PL=5 : REM PERIOD LENGTH 5 MINUTES
110 PS=TI : REM PERIOD STARTS NOW
120 PF=PS+PL * 60 : REM PERIOD FINISH TIME
130 TR=PF-TI : REM TIME REMAINING
140 GOSUB 400
150 PRINT "[clr]" T$
160 IF TI<PF THEN 130
170 END
400 REM CONVERTS SECONDS TO MM:SS
    FORMAT
440 T3=INT (TR / 60 + 0.5)
450 M1=INT(((T3 /60/60) - (INT(T3 /60/60))) * 60)
460 B$=STR$(M1)
470 M$=MID$(B$,2,2) : IF M1<10 THEN M$="0"
    + MID$(B$,2,1)
480 T2=T3 - (M1 * 60)
```

```
490 S1=INT(((T2 /60) - (INT(TE /60))) * 60)
500 B$=STR$(S1)
510 S$=MID$(B$,2,2) : IF S1<10 THEN S$="0"
    + MID$(B$,2,1)
520 T$=M$+":"+S$ : RETURN
```

The most valuable feature of the timekeeper is the ability to control the length of pauses made during execution, independent of the program lines being executed. The friendly computer guide shows how to make delays by using a FOR ... NEXT loop with the statements:

```
FOR I=1 TO 100 : NEXT I
```

The major problem with this method is that it ties up the whole program while you pause. You can insert program lines for execution during the loop, but then some guesswork and experimenting will be necessary every time you program to obtain the desired pause. Frequently you will have to compromise between the statements you want to execute and the time you can allot to the pause. Finally, if the lines executed during the pause contain the decisions with varying amounts of program to be executed based on the decision, the length of the pause becomes unpredictable.

Getting Control Over Pause

The timekeeper counts independently, on a steady basis, and allows you to assume control of the length of a pause, while permitting other parts of the program to continue. To do this you simply note the time the pause begins and add the desired pause length, giving the time the pause will end. An IF decision watches for the clock to exceed that end time, and you can run other parts of the program while the pause is in progress. The decision watching for the end of the pause must be made with a reasonable frequency, so the number of statements you can execute between repetitions of the end-time decision will depend on how long the pause is and how exact you want the measurement of the pause to be.

As a very conservative rule-of-thumb, allow twenty eighty-character (multiple statement) program lines to reach the end-time decision at an interval of about ten percent of the total pause length. For example, if I pause for about ten seconds, I can allow up to one second, or about twenty program lines. Similarly, a two-second pause will allow up to four program lines between repetitions of the end-time decision. You can use a greater number of lines if they do not contain several statements each.

These time estimates are very rough: do some experimenting yourself to find how many statements you can squeeze in and still get accurate control of the pause length. Once you have established some rules for yourself, they should be useful in all your programming.

As an example of the pause, let's say that I'm writing a game program in which we explore a dungeon. If someone casts a magic spell of darkness, then I want to give no visual clues for the length of the spell – say twenty seconds – while the action of the program continues. The following segment of a program will provide that effect:

```
100 DEF FN PS(T2)=TI+(T2 * 60)
350 REM   THE SPELL IS CAST
360 GOSUB 900 : P1=FN PS(20)
370 REM P1=TIME TO END BLACKOUT
380 REM THE
390 REM  PROGRAM
400 REM   CONTINUES
410 REM    RUNNING
420 REM     WITH A
430 REM      BLACK
440 REM       SCREEN
490 REM  (UP TO FORTY PROGRAM LINES)
775 IF TI>P1 THEN GOSUB 902 : GOTO 800
780 GOTO 380
800 END
900 POKE 36879,8 : FOR I = 38400 TO 38906 :
    POKE I,0 : NEXT I
905 RETURN : REM BLACKOUT MAKER
920 POKE 36879,78 : RETURN : REM BLACKOUT
    LIFTER
```

This application uses the function PS to relate the desired pause length (T2) to a future time value (P1) which defines the end of the blackout.

Another application of the pause timer can limit how often I may perform an action. I'm writing a game in which the player fires a laser cannon that takes five seconds to recharge before it can be fired again. The line which times the firing interval looks like this:

```
350 IF PEEK(197)=35 AND TI>P1 THEN GOSUB
    800 : P1=TI+(5*60)
800 RETURN : REM VISUAL AND SOUND EFFECT
FOR LASER FIRING
```

Here there is no need to worry about running the end-time decision within a set interval – the next time I want to fire the cannon, the logical AND in the decision checks to see if it has recharged. This pause method can also be used in an education program, to limit how soon the student may answer after a question appears, or may try a second time after an incorrect first answer has been entered.

If you're interested in converting existing programs to timekeeper pauses, the statement:

```
FOR I = 1 TO 100 : NEXT I
```

is worth about eight counts on the timekeepeer, or 0.13 seconds. There will be some difference between this statement and a longer loop. For instance, modifying the statement to:

```
FOR I = 1 TO 1000 : NEXT I
```

This is worth 72 counts, or 1.2 seconds, not the eighty counts one might expect. This is because of the "overhead" time needed to establish the loop during execution. There may even be differences between machines. You can check your own timing with this simple program:

```
10 BT=TI
20 FOR J=1 TO 1000 : NEXT J
30 FT=TI
40 ET=FT-BT
50 PRINT ET, ET/60
```

This displays the time passed in both counts and seconds. Try varying the length of the loop in line 20 to get a general idea of what the "overhead" time is on your computer.

You need to do nothing to the timer to use it as a basis for pauses. However, if you have the VIC on for long periods, or if you set TI$ to keep clock time and run the program near midnight, be careful: if the pause starts before midnight and ends after, you may never reach the end of the pause, since the clock resets to 000000 at midnight. You can put in additional statements to watch for this problem and compensate for it; you can have the program reset the clock to 000000 before timing any pauses; or you can ignore the possibility and hope for the best. The third option, technically unsound as it is, requires the least effort and presents no great threat.

These pause techniques have two important features: controllable pause lengths and the ability to run other, unrelated parts of the program while the pause is in effect. When you develop a program, you can select a length of pause that will not change as you add, change, remove, or relocate program statements. The pause can also be lengthened or shortened to suit your needs, without major changes in the program itself. You have made your pause independent of the program that contains it. At the same time, you can execute lines of an unrelated portion of the program while the pause is in progress, making the program independent of the pause it executes. The timekeeper in the VIC gives the programmer much better control of realism in his game and simulation programs. ©

An Easy Way To Relocate VIC Programs On Other Commodore Computers

Greg Sherwood and Ross Sherwood
Manhattan, KS

BASIC programs that are generated on Commodore's new color computer, the VIC-20, start at memory location 4097 rather than at 1025 as in Commodore's other computers. Thus, if you wish to use features available for some of the other computers such as Toolkit, etc. to edit or modify a program written or stored from a VIC, you need to relocate the program so it starts at memory location 1025.

The following is a description of a quick and simple method of relocation of VIC programs, I will describe two versions, one using the built in monitor and the other done in direct mode.

To relocate a VIC program from the monitor, load the program from tape and then enter the monitor with SYS1024. Next, look at the first part of BASIC memory by typing M 0400 0400. Make the following changes to the displayed memory:

'M 0400 00 01 10 00 00 99 00 XX'
(XX means *doesn't matter.*)

Next exit the monitor by typing an X. Now type LIST and the VIC program should list out with an additional line (line 0) at the beginning: 0 PRINT Finally type "0" and RETURN and the VIC program is relocated and can be edited or modified at will.

To accomplish the same change in direct mode, the following six POKEs are entered:

POKE 1025,1:POKE 1026,16
change link pointers to VIC program
POKE 1027,0:POKE 1028,0
create line #0
POKE 1029,159
put PRINT on line 0
POKE 1030,0
end of line indicator

Now, as above, type LIST and the VIC program will list with the additional line 0 PRINT.

Last, type "0" and the line 0 will be eliminated so the VIC program can be edited.

This method works with both BASIC 3.0 and 4.0 Commodore computers and, through it hasn't been tested on other versions, it should work on those as well. It has been successfully used on both 40 and 80 column machines.

If you should wish to relocate several VIC programs in succession, the following assembly language subroutine can be used. It begins at location 926 in the second cassette buffer and can be called by SYS926. To load this program, enter the monitor and type M 039E 03C8 annd change the memory as follows:

```
039E A9 00 8D 03 04 8D 04 04
03A6 8D 06 04 A9 30 8D 6F 02
03AE A9 01 8D 01 04 A9 10 8D
03B6 02 04 A9 99 8D 05 04 A9
03BE 0D 8D 70 02 A9 02 85 9E
03C6 60 00 XX XX XX XX XX XX
```

This program can be saved on tape or disk by saving from 039E to 03C8 and then can be loaded in anytime and used to relocate VIC programs with a SYS926 command until the machine is turned off or the second cassette buffer is used for some other purpose. This subroutine is located high enough in the second cassette buffer that disk operations don't overwrite it.

This subroutine automatically erases line 0 and so that, when you return to BASIC, the VIC program is moved and ready to be edited, etc. without the necessity of removing line 0.

```
039E A9 00          LDA #$00
03A0 8D 03 04       STA $0403
03A3 8D 04 04       STA $0404
03A6 8D 06 04       STA $0406
03A9 A9 30          LDA #$30
03AB 8D 6F 02       STA $026F
03AE A9 01          LDA #$01
03B0 8D 01 04       STA $0401
03B3 A9 10          LDA #$10
03B5 8D 02 04       STA $0402
03B8 A9 99          LDA #$99
03BA 8D 05 04       STA $0405
03BD A9 0D          LDA #$0D
03BF 8D 70 02       STA $0270
03C2 A9 02          LDA #$02
03C4 85 9E          STA $9E
03C6 60             RTS
03C7 00             BRK
03C8 00             BRK
```

©

Review:

UMI Amok For VIC

Harvey B. Herman
Associate Editor

One of the reasons people buy personal computers is to play games. I confess that this reviewer is no exception. In contrast to all the serious applications of computers, it is still fun to relax and play an occasional interesting and challenging game. The Commodore VIC is particularly suited for game playing as it comes with an interface for a joystick and can play sound effects through the TV speaker. Unusual displays are also possible because one is not limited to a standard character set composed of letters, numbers, and graphics. A knowledgeable user can define a new set for special effects.

Until now, the VIC games I have previewed have, for the most part, been good, but nevertheless have not taken full advantage of all the VIC's capabilities. All of them have been written in BASIC which can be too slow for good animated displays. Machine language usually looks much more realistic. The AMOK program was a pleasant surprise. The author, Roger Merritt, seems to have done everything right. His machine language program uses the features of the VIC to advantage. My kids, on whom I rely to advise me on game programs, rated it a 9 (out of 10 possible). I did enjoy playing with it myself, but not as much as they did.

You Against The Angry Robots

The game works like this: you are in a partitioned room with angry robots. The robots, shown in various colors, are shooting at you and you, of course, are dodging and returning their fire. You get three chances before the game is over. You lose a chance whenever your character touches the walls, partitions, or robots. If the robots hit you with their fire, you also lose a chance. Your character is controlled with a joystick or the keyboard. Other rooms can be entered (the door closes behind you) where you encounter a new set of differently colored robots. The object of the game is to score the most points. The color of the robot determines its point score and there are bonus scores. The difficulty of the game is set at the beginning.

I think, in all fairness, our enthusiasm is partly due to some of the relatively pedestrian VIC programs which we have previously seen. Your character in this program is a sight to behold. I have not seen graphics this good on either the VIC or the PET and it approaches animations I've seen on dedicated video games. I recommend this program highly, particularly if you have game playing kids. I am told it is similar to the arcade game BEZERK. An adult may not stay interested for hours, but a kid will. Think of all the quarters you will save.

Hints by Herman – if you have added the 3K memory expansion, a special load sequence is necessary. Use:

```
LOAD " ",1,1
SYS 4110
```

This is not in the instructions.

United Microware Industries
3431 H Pomona Blvd.
Pomona, CA 91768
$18.95

©

Review:

UMI 3K VIC Memory Expansion

Harvey B. Herman
Associate Editor

This small circuit board is designed to plug into the 44 contact female edge connector inside the VIC. It adds 3K of RAM memory to the 5K normally present. When the VIC is powered up, the bytes free message should now total 6655 instead of 3583. This means that programs loaded from tape can be almost twice as long as before without running out of memory. The board also has two empty ROM sockets which allow up to 16K of programs on chips to reside permanently in memory. These programs do not disappear when power is turned off or lost. Initial startup with ROM-based software is much more convenient compared with a long program loading from tape. Many ROM games and other interesting ROM programs will be marketed by UMI and other companies. This board will allow you to use them without additional expense.

The circuit board is easily installed. The VIC case does not even have to be opened as the board fits through the opening in the rear. If you read and follow the quite explicit directions, I predict you won't have any difficulty.

It's Solidly Constructed

I have several positive comments. The board looks solidly constructed and seems to be well thought out. Each ROM can be placed in one of two areas of memory and three ROM sizes can be accommodated. The instructions even give technical hints to advanced hobbyists who intend to program their own ROMs. The price is competitive with similar boards I have seen advertised, but have not yet examined critically.

I have two minor negative comments. Contact fingers on the board are not gold plated so corrosion could be a problem under certain conditions. This will probably not be a concern in typical household use, however. The board sticks out a little from the back of the VIC and is unprotected. Users will have to take care that it does not get knocked about. Again, I do not see this as a serious problem.

As you may have inferred, I am quite happy with this memory expansion. I felt somewhat limited by the small amount of available RAM in the original VIC. Now I can run more ambitious programs. I am also looking forward to installing the better ROM-based programs when they become available.

United Microware Industries
3431 H Pomona Blvd.
Pomona, CA 91768
$79.95

©

Alphabetizer

Jim Wilcox
Vienna, WV

The following program will alphabetize letters or put numbers in order from lowest to highest. The first thing that will happen is the screen will clear and the message "HOW MANY VARIABLES?" will appear on the screen. You then type in the number of names you wish to sort. The variable "VAR" will take on the value typed in. The statement at line number 20 will set the amount of variables of the dimensioned variable "A$". If you are stuck on dimensioned variables, read on.

DIMensioned Variables

Dimensioned variables can be compared to houses on a street. Let's say the house numbers on this block start at one and end at ten. They all belong to the street named, say "Washington." To make things easier than naming each house after a different president, they are given numbers. There might be another house with the number two, but not on the same street. The name of the street is the variable, but there is more than one house on the street, which are variables too. To get a letter to house #2 on Washington Street, one would have to write the person's name, "Jones," who would reside at 2 Washington Street. In a computer program, one could set the variable WASHINGTON$(2) = "JONES". 1 Washington Street might have the "George's" living there so the variable would be WASHINGTON$(1) = "GEORGE". So a dimensioned variable is a variable that has other variables related to it, i.e. all the people on the block have in common the fact they live on Washington Street.

I recommend that you try a small list first, such as ten of the letters of the alphabet mixed up. This will not take long to put the characters in order and the programmer can tell whether the program was typed in properly. On longer lists it becomes tempting to hit the RUN/STOP key to see if the computer is stuck in an endless loop, but the longer the list, the longer it takes.

```
10 INPUT"{CLEAR}HOW MANY VARIABLES"
   ;VAR
20 DIMA$(VAR+22)
30 FORA=1TOVAR
40 PRINT"#"A;
50 INPUTA$(A)
60 NEXT A
70 PRINT"ALPHABETIZING"
80 FORA=1TOVAR-1
90 FORB=A+1TOVAR
100 IFA$(B)<=A$(
    A)THENSM$=A$
    (B):A$(B)=A$(
    A):A$(A)=SM$
110 NEXT B
120 NEXT A
130 PRINT"FINISH
    ED ALPHABETI
    ZING"
140 POKE36878,8
150 POKE36874,25
    0
160 FORA=1TO500
170 NEXT A
180 POKE36878,0
190 POKE36874,0
200 FORA=1TOVAR
    STEP22
210 FORB=ATOA+21
220 PRINTA$(B)
230 NEXT B
240 GETA$:IFA$="
    "THEN240
250 NEXT A
260 END
```

©

COMPUTE!'s Listing Conventions

Many programs which are listed in **COMPUTE!** use cursor control keys, color keys, and so forth. We have established a listing convention which we believe eases the task of typing programs in accurately.

PET/CBM/VIC Conventions

Generally, PET/CBM/VIC programs will contain bracketed words for any special characters: [DOWN] means the cursor-down key; [3 DOWN] means type the cursor-down key three times.

If a program line runs over into the next line down, the ~ symbol indicates where the line broke (in case the number of spaces is unclear between quotes). An underline means that that key is shifted.

VIC Conventions

SET COLOR TO BLACK	[BLK]
SET COLOR TO WHITE	[WHT]
SET COLOR TO RED	[RED]
SET COLOR TO CYAN	[CYN]
SET COLOR TO PURPLE	[PUR]
SET COLOR TO GREEN	[GRN]
SET COLOR TO BLUE	[BLU]
SET COLOR TO YELLOW	[YEL]
FUNCTION ONE	[F1]
FUNCTION TWO	[F2]
FUNCTION THREE	[F3]
FUNCTION FOUR	[F4]
FUNCTION FIVE	[F5]
FUNCTION SIX	[F6]
FUNCTION SEVEN	[F7]
FUNCTION EIGHT	[F8]
ANY NON-IMPLEMENTED FUNCTION	[NIM]

DISKSHARE 9000

Multi User System

- **Works with all combinations PET/CBM models.**
- **100% Hardware. No software required!**
- **No special commands, User TRANSPARENT!**
- **Up to 8 users can share a disk &/or printer.**
- **10, 15, 20, & 25 foot cables available.**
- **Connects in minutes, no tools necessary.**
- **Ideal for EDUCATIONAL & BUSINESS needs.**
- **Increases productivity – Reduces Costs!**

DISKSHARE 9000-3	3 USER SYSTEM WITH 6ft. CABLES	$ 749.00
DISKSHARE 9000-8	8 USER SYSTEM WITH: – THREE 6ft. CABLES – THREE 10ft. CABLES – TWO 15ft. CABLES	$1149.00
DISKSHARE 9001-6	ADD-ON CABLE (6 FOOT LENGTH)	$ 79.95
DISKSHARE 9001-10	ADD-ON CABLE (10 FOOT LENGTH)	$ 99.95
DISKSHARE 9001-15	ADD-ON CABLE (15 FOOT LENGTH)	$ 119.95
DISKSHARE 9001-20	ADD-ON CABLE (20 FOOT LENGTH)	$ 139.95

For More Information Call:

Questar International Inc.

Suite 102
7270 Woodbine Avenue,
Markham, Ontario Canada
(416) 490-8044

dealer and
distributor inquiries
invited

CAPUTE:

Corrections and Amplifications

— Issue #16, pg. 107, "The Unwedge": in line 116, the final datum should read H259, not H25.

— Issue #18, pg. 60, "Bits, Bytes and Basic Boole" the following lines should be changed in the program listing:

```
1060 FOR A1=0 TO 3
1070 A2=2↑A1
1090 SC(0,A3,A1) = SA(A3 OR A2)
```

— Issue #18, pg. 118, "Assembler Update" was missing the following program and was inadvertently in the Atari Gazette. Mr. Brandon's Assembler for the PET (which originally appeared in Issue #13, pg. 120) can be upgraded with the following modifications to permit LOAD/SAVEs of source code to disk:

Program 2.

```
300 PRINT"{DOWN}{REV}I{OFF}NPUT
    REV}D{OFF}ELETE I{REV}N{OF
    OFF}SERT {REV}L{OFF}IST {R
    REV}S{OFF}AVE L{REV}O{OFF}
    AD"
305 PRINT"{REV}A{OFF}SSEMBLE {REV}Q
    {OFF}UIT"
360 IF CM$="0"THEN11000
370 IF CM$="S"THEN12000
11000 INPUT"FILENAME ";FL$
11010 OPEN8,8,2,"0:"+FL$+",S,R"
11020 FORT=1TOMEM
11030 GET#8,IO$:IFIO$=CHR$(13)THEN110
    50
11040 A$(T)=A$(T)+IO$:GOTO11030
11050 NEXTT
11060 CLOSE8
11070 GOTO300
12000 INPUT"FILENAME ";FL$
12010 OPEN8,8,2,"0:"+FL$+",S,W"
12020 FORT=1TOMEM
12030 PRINT#8,A$(T);CHR$(13);
12040 NEXTT
12050 CLOSE8
12060 GOTO300
```

— Issue #18, pg. 148, "Inversion Partitioning" will run on the Original ROM PET with the following lines changed (our thanks to Lou Sander):

Program 3.

```
033A A2 00 E0 00 D0 1D A5 87

0362 A5 02 48 A5 66 48 A5 67

0372 02 A9 FE 85 66 A9 3F 85
037A 67 A1 01 48 A1 66 81 01
0382 68 81 66 E6 01 C6 66 E4
038A 01 D0 EE E6 02 C6 67 A5
0392 67 C9 21 D0 E4 68 85 67
039A 68 85 66 68 85 02 68 85
03A2 01 38 A9 FF E5 7C 48 A9
03AA 43 E5 7D 48 A9 FF E5 86
03B2 85 7C A9 43 E5 87 85 7D
03BA 68 85 87 68 85 86 EC 3D

03CA 8E 01 04 8E 02 04 85 6F
03D2 85 7D A9 01 85 6E A9 03
03DA 85 7C A5 7C 85 7E 85 80
03E2 A5 7D 85 7F 85 81 A5 86
03EA 85 82 A5 87 85 83 60 00
```

©

New Products

Board Offers 24K Additional Memory For VIC 20

Quantum Data Inc. of Costa Mesa, CA announced the first of its new products to expand the capabilities of the Commodore VIC 20 personal computer: A Board allowing up to 24K of additional user memory. Designated the QDI Expandor, it expands the memory of the VIC to a total of 29K bytes.

Designed for those users who need more than the 3K of user RAM available on the VIC, the QDI Expandor is available in several configurations from 0K to 24K of additional memory. The board uses state of the art memories which allow the board to consume less than 150 MA even when fully loaded with RAM. These memories are also pin compatable with popular EPROMs and ROMs and they can be mixed and matched in 8K blocks.

In its standard configuration, the QDI Expandor uses memory from HEX 2000 to 7FFF but it may also be jumpered to operate one of the 8K blocks in the A000 to BFFF range, "Where the ability of the board to handle ROMs is very convenient," explains Dick Edwards, QDI's president. "That's because the VIC starts looking at location A000 on power up to see if there is a machine language program present in ROM, and, if there is, it will run that program. We expect this feature to be of special interest to systems houses, who are going to use this computer where a resident machine language program is important."

Measuring 4.5 by 6 inches, the QDI Expandor can plug directly into the VIC memory expansion port or in the expansion chassis that QDI will soon be announcing. Priced at a suggested retail of $295.00, the board is available from stock.

For further information contact: Quantum Data, Inc., 3001 Redhill, Bldg. 4, Suite 105, Costa Mesa, CA 92626.

PDI Announces Publication Of Do-It-Yourself Spelling

Program Design, Inc., the Greenwich, Connecticut, firm that specializes in the design, development, and marketing of educational courseware for microcomputers, has just published a new spelling program entitled DO-IT-YOURSELF SPELLING. Unlike other spelling programs on the market, Do-It-Yourself Spelling allows the user to add voice to the program.

Do-It-Yourself Spelling allows teachers, parents, and other individuals to create their own spelling programs. Following simple instructions, the person enters a series of 10-word lists into the computer program. The word lists might consist of a child's vocabulary assignment. It might consist of science words or musical terms or even the names of baseball players.

Do-It-Yourself Spelling comes with a list of 1950 words that every child should recognize and be able to spell by the time he or she graduates from 6th grade. The words are organized by grade. There are 50 first-grade words, 300 words for each of the second, third, and fourth grades, and 500 words for each of the fifth and sixth grades.

Do-It-Yourself Spelling is available for use on Atari microcomputers with a memory of at least 8K. The program retails for $19.95.

For additional information, contact: Laurie Hall, Program Design, Inc., 11 Idar Ct., Greenwich, CT 06830 203-661-8799.

Mountain Computer Announces RAMPLUS + ™ For The Apple II Computer

Scotts Valley, CA — Mountain Computer has just released a new dual 16K RAM card for the Apple II®. Two banks of 16K selectable RAM expand the Apple to 80K of available memory. The second bank of 16K RAM is controlled by user-supplied software. Hardware and/or software selection of each 16K bank of RAM is controlled by the user. The card also provides its own refresh circuitry.

Card installation is simple, just install it in any I/O slot. When RAMPLUS+ is in Slot 0, it emulates an Apple Language Card®, and you can install multiple RAMPLUS+ cards into the same Apple II.

Status includes the area and bank of RAM selected, READ, READ or WRITE operation,

NOW TWO LOCATIONS

SAVE TIME • SAVE SHIPPING

ATARI **Computers for people.**™

800™ $699

410 Recorder	$59.00
810 Disc Drive	$444.00
822 Printer	$359.00
825 Printer	$629.00
830 Modem	$159.00
820 Printer	$269.00
850 Interface	$159.00
New DOS 2 System	$21.00
CX30 Paddle	$18.00
CX40 Joy Stick	$18.00
CX853 16K RAM	$89.00
Microtek 16K RAM	$75.00
Microtek 32K RAM	$159.00
One year extended warranty	**$50.00**

ATARI 400

16K.... $329
32K.... $478
48K.... $555

Intec 48K Board.... $249

ATARI SOFTWARE

CX404 Word Processor	$119.00
CX404 PILOT	$68.00
CX413 Microsoft Basic	$68.00
CX4101 Invitation To Programing I	$17.00
CX4102 Kingdom	$13.00
CX4103 Statistics	$17.00
CX4104 Mialing List	$17.00
CX4105 Blackjack	$13.00
CX4106 Invitation to Programing 2	$20.00
CX4107 Biorythm	$13.00
CX4108 Hangman	$13.00
CX4109 Graph It	$17.00
CX4110 Touch Typing	$20.00
CX4008 SPACE INVADERS	$32.00
CX4112 States & Capitals	$13.00
CX4114 European Countries & Capitals	$13.00
CX4115 Mortgage & Loan Analysis	$13.00
CX4116 Personal Fitness Program	$59.00
CX4117 Invitation To Programing 3	$20.00
CX4118-20 Conversational Languages (ea.)	$45.00
CX4121 Energy Czar	$13.00
CXL4001 Educational Master	$21.00
CX6001-17 Talk & Teach Series (ea.)	$23.00
CX8106 Bond Analysis	$20.00
CX8107 Stock Analysis	$20.00
CX8101 Stock Charting	$20.00
CXL4002 Basic Computing Language	$46.00
CXL4003 Assembler Editor	$46.00
CXL4004 Basketball	$24.00
CXL4005 Video Easel	$24.00
CXL4006 Super Breakout	$30.00
CXL4007 Music Composer	$45.00
CXL4009 Chess	$30.00
CXL4010 3-D Tic-Tac-Toe	$24.00
CLS4011 STAR RAIDERS	$39.00
CXL4012 MISSLE COMMAND	$32.00
CXL4013 ASTEROIDS	$32.00
CXL4015 TeleLink	$20.00
Visicalc	$149.00
Letter Perfect (Word Processor)	$109.00
Source	$89.00
CX481	$75.00
CX482	$109.00
CX483	$54.00
CX484	$319.00

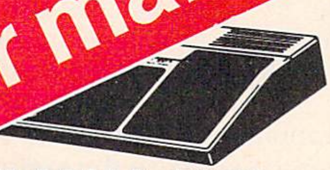
PRINTERS

Centronics 739-1	$649.00
Diablo 630 Special	$1799.00
Epson	
MX70	$359.00
MX80	$469.00
MX80FT	Call
MX100	Call
NEC	
8023	$639.00
7730	Call
7720	Call
7710	Call
Okidata	
82A	$499.00
83A	$769.00
84	$1129.00
Citoh Starwriter	
25 CPS-P	$1329.00
45 CPS-P	$1699.00
Paper Tiger	
445G	$699.00
460G	$899.00
560G	$1129.00
Talley	
8024-7	$1399.00
8024-L	$1629.00
Xerox 820	
System I 5½"	$2450.00
System II 8"	$2950.00
CPM 5¼	$169.00
Word Processing	$429.00
Super Calc	$269.00

computer mail order saves

RAM SALE!

Atari	
Microtek 16K	$64.00
32K	$129.00
Intec 32K	$139.00
40K	$219.00
Ramcrom 128K	$519.00
Apple	
Microtek 16K	$94.00
32K	$129.00
Commodore	
64K upgrade	$389.00
Hewlett Packard	
16K upgrade	$249



TI-99/4A $379

32K Expansion	$329.00
PHP Printer Solid State	$319.00
PHC 004 TI-99/4 Home Computer	$399.00
PHP 1600 Telephone Coupler	$169.00
PHP 1700 RS-232 Accessories Interface	$169.00
PHP 1800 Disk Drive Controller	$239.00
PHP 1850 Disk Memory Drive	$389.00
PHP 2200 Memory Expansion (32K RAM)	$319.00
PHA 2100 R.F. Modulator	$43.00
PHP 1100 Wired Remote Controllers(Pair)	$31.00

Texas Instruments

PHM 3006 Home Financial Decisions	$26.00
PHM 3013 Personal Record Keeping	$43.00
PHD 5001 Mailing List	$60.00
PHD 5021 Checkbook Manager	$18.00
PHM 3008 Video Chess	$60.00
PHM 3010 Physical Fitness	$26.00
PHM 3009 Football	$26.00
PHM 3018 Video Games I	$26.00
PHM 3024 Indoor Soccer	$26.00
PHM 3025 Mind Challengers	$22.00
PHM 3031 The Attack	$35.00
PHM 3032 Blasto	$22.00
PHM 3033 Blackjack and Poker	$22.00
PHM 3034 Hustle	$22.00
PHM 3036 Zero Zap	$18.00
PHM 3037 Hangman	$18.00
PHM 3038 Connect Four	$18.00
PHM 3039 Yahtzee	$22.00
PHM 3017 Terminal Emulator I	$39.00
PHM 3026 Extended Basic	$88.00
PHM 3035 Terminal Emulator II	$45.00

HOW TO ORDER: Phone orders invited or send check or money order and receive free shipping in the continental United States. PA residents add 6% sales tax.

computer mail order west

800-648-3351

IN NEVADA, CALL (702) 588-5654

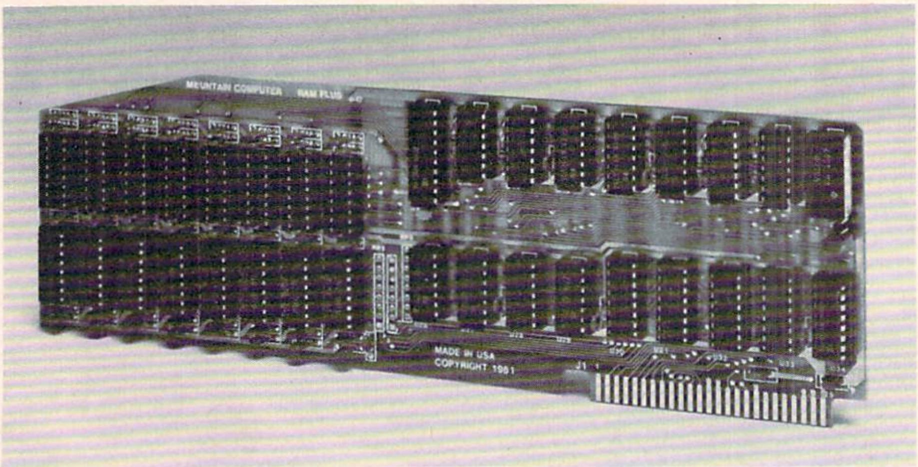
P.O. BOX 6689, STATE LINE, NEVADA 89449

and ROM Enable or Disable information.

The card is supplied with 16K of installed RAM for $189.00 The additional 16K of plug-in RAM costs only $24.95. RAM diagnostics have been developed (on diskette) and are supplied with the product.

For additional information, contact:

Mountain Computer Inc.
300 El Pueblo Rd.
Scotts Valley, CA 95066
(408)438-6650



Calendar/Clock System For Small Computers

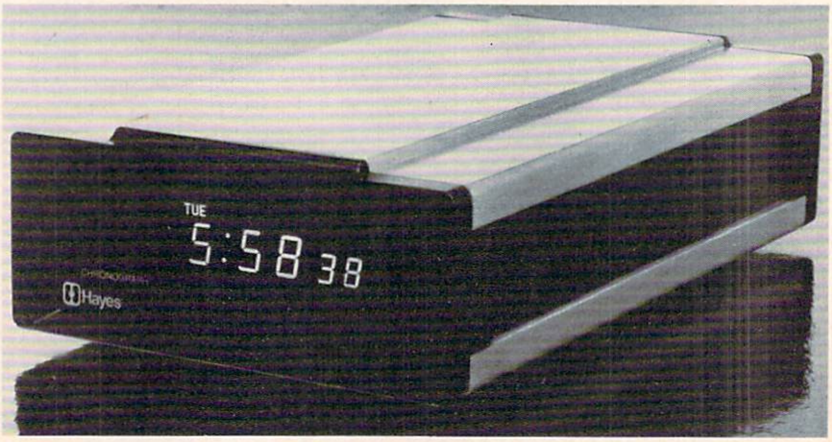
Norcross, GA — Hayes Microcomputer Products, Inc., announces the Hayes Stack™ Chronograph, an RS-232C compatible calendar/clock for small computers. The Chronograph is the latest in the Hayes Stack series of stackable microcomputer component systems.

Featuring quartz-crystal control, the Chronograph adds the dimension of precise timekeeping to computer systems. With the Chronograph and user-developed software, a computer can log programs and reports by day, date and time. Utilizing the computer alarm feature, the Chronograph can also provide a computer with information necessary to control lights, burglar alarms and sprinkler systems. To cut the cost of electronic mail, the user can combine the Chronograph with the Hayes Stack Smartmodem and a computer, then develop programs to batch messages during the day and send them at night when telephone rates are lowest.

The Chronograph is a stand-alone unit in an aluminum case with a large, easy-to-read display for time, date and weekday reporting. The display also features low battery, write-protect and alarm indicators.

The Chronograph reports the time in hours, minutes and seconds in 12- or 24-hour modes. The date is output in a year, month, day format with automatic leap year adjust, and the weekday is output as a single digit, 0-6.

Because it is powered independently, the Chronograph does not need to be reset when the computer is turned off. The Chronograph also features a battery backup to maintain time, date and weekday for up to a year when the power fails or is disconnected. For added protection, a write-protect switch on the rear panel of the Chronograph prevents accidental changing of the time and date.



The user controls the Chronograph through a command set of simple ASCII character strings. These commands allow the user to set, read and display calendar and clock data, control the computer alarm and select various options. Other features include 300 or 1200 baud operation and automatic baud rate, parity sense and word size detect.

The Chronograph system includes the Chronograph unit, power pack, 3 AA batteries and complete owner's manual. It is covered by Hayes two year limited warranty. The suggested retail price is $249.00. For further information contact Hayes Microcomputer Products, Inc., 5835 Peachtree Corners East, Norcross, GA 30092.

TYCOM Introduces Three Educational Packages

TYCOM Associates announces three new educational software packages for the Commodore PET/CBM computers, to compliment their existing educational software line. The programs are intended for drill and practice in conjunction with courses at the Junior High or High School level.

ALBEGRA *WORD PROBLEMS:* A CAI module intended to help teach algebra students to set up and solve word problems. A drill option supplies randomly chosen problems from several formats, with randomly generated values. The student is given the correct answer if a wrong answer is entered, and a score is given upon exiting the program.

SPANISH: Drill and practice programs intended as a noun and verb vocabulary building tool. User may choose passive or active mode of operation. Verb Conjugation drills are included.

GERMAN: Drill and practice programs intended as a vocabulary building tool. User may choose passive or active mode of operation. Noun drills include gender and plural forms.

The above programs are available on cassette tape only and run on all 40 or 80 column screen PET/CBM computers. Each sells for $19.95. A free list of all educational software offered by TYCOM Associates is available upon request.

Write to:

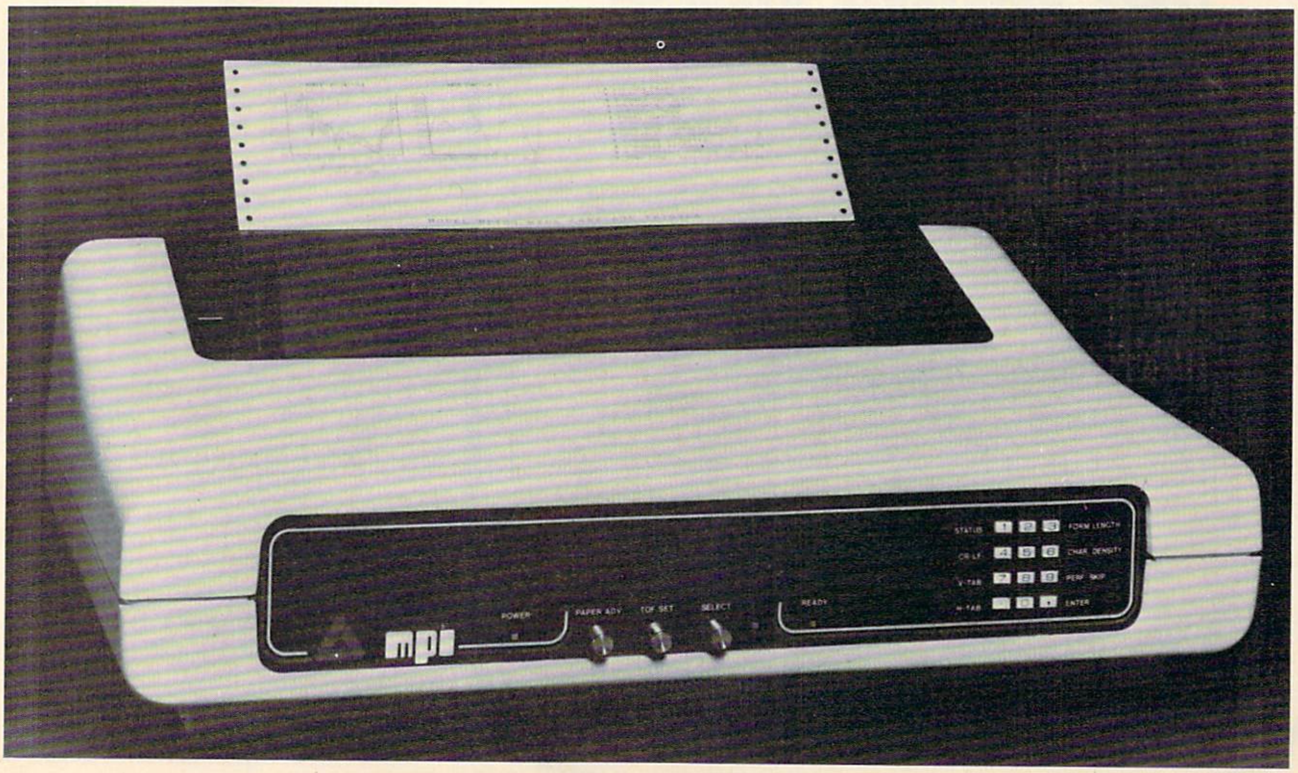
TYCOM Associates
68 Velma Avenue
Pittsfield, MA 01201

Model MP150 Wide Carriage Printer

The all new Model MP150 printer from MicroPeripherals, Inc. is the latest addition to their matrix printer line. It is the first of a series of wide carriage units designed specifically for mini and micro business systems.

The heavy duty printhead is rated for continuous duty and has an expected lifetime of over 100,000,000 characters. It forms characters bidirectionally in a logic seeking mode to optimize system thruput. Nine ballistically driven print wires form crisp, clear characters with true descenders and underlining capability. It can print a full 136 character line at 10 characters per inch or, by selecting either the 12 or 16.7 character per inch density, up to 226 columns may be printed. This allows full 136 column printouts to be condensed to fit on standard 8.5 inch wide paper. Double wide characters can be software selected in any of the character densities to give a total of six different CPI densities.

A 7x9 matrix font is used for high speed data printing while an 11x9 serif style matrix font is used for applications requiring a high quality correspondence printout. A standard 96 character USASCII set with four strap selectable foreign fonts are standard in each unit. Special fonts may be either down-line loaded into RAM memory, or permanently



located in ROM memory.

High resolution dot addressable graphics capability is included for those applications requiring plotting, printing of screen graphics, drawing of illustrations, and producing special characters or identification marks. Alphanumerics may be overprinted into the graphics area for labeling of graphs and illustrations.

Forms handling is implemented with a stepper motor drive tractor paper feed system which can be adjusted to accept forms ranging from 3 inches to 15 inches in width. Eight selectable forms lengths and a Skip-Over-Perforation feature provide for the precise paper handling required for business applications.

All printing and interface functions are placed under the direct control of a microprocessor array. The standard 1K buffer can be expanded to 8K for applications requiring additional character buffering. A Centronics type interface can accept parallel TTL level data at a transfer rate in excess of 1000 characters per second using either a Strobe/Ack or a Strobe/Busy handshake. An optional RS232C serial interface can be added and will accept data at any one of seven strapable baud rates up to 9600. Both X-ON/X-OFF and ETX/STX protocols are supported by the optional serial interface. The MP150 can also be interfaced to devices with an IEEE 488 Bus output through an optional IEEE-to-Centronics interface adapter card.

A long life mobius loop cartridge ribbon provides a minimum life of eight million characters.

An optional front console panel can be added to give greater flexibility in changing the print format parameters. It includes a non-volatile memory to store the format parameters for over three months without power.

The printer measures 23 inches wide by 16 inches deep by 7.5 inches high and allows for front, bottom or rear paper entry. It is designed to allow easy access to the electronic and mechanical components for simplicity of servicing.

The MP150 Printer, complete with graphics capability lists for $1095 with substantial discounts available for OEM quantities.

For additional information, contact:

Frank W. Irvin
Vice President, Marketing
MPI
4426 South Century Drive
Salt Lake City, UT 84107
(801)263-3081

Multi-purpose Interface For PET/CBM Computers

TEACHING TOOLS Microcomputer Services announces a new Multi-purpose Interface for PET/CBM Computers. This three-in-one interface provides the following:

1. Video monitor connector. Lets you show whatever is on the screen on a video monitor also. This is ideal for classrooms, and anywhere else a large display is needed. A high quality RF modulator (made by ATV Research) is also available, so you can use a TV in place of a video monitor. NOTE: The video adaptor is for PET/CBM computers with 9" screens only, *not* for 80 column CBMs or "Fat" 40 column PETs.

2. Sound adaptor with built in amplifier, speaker and volume control. Provides CB2 sound (the standard for PET/CBM computers). Takes its power from the PET – no batteries needed.

3. Audio tape recorder control.

Lets you add recorded messages to your programs – puts starting and stopping a cassette recorder under program control.

The multi-purpose interface easily plugs into the back of the PET/CBM. If desired, it can be clamped from inside the PET, so that it cannot be removed. Complete instructions are included for connecting the interface, and for using CB2 sound and tape recorded messages in your own programs.

Prices: Multi-purpose interface, (requires cable with RCA phono plug at one end, connector for your monitor on the other end) $109.95. Multi-purpose interface and RF modulator (includes all cables and batteries for modulator) $149.95. Sample copy of instructions (credited toward later purchase) $1.00.

Please add $3.00 for shipping and handling.

For prices in Canada, contact SES Computing, 465 King Street East, Suite 9, Toronto, Ontario M5A 1L6 (416-336-4242).

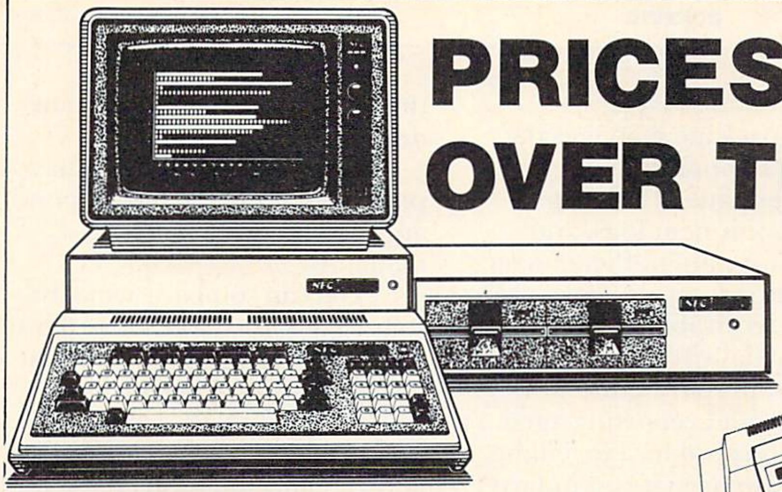
Axlon Personal Communication Terminal Announced

Sunnyvale, CA – November 19, 1981 – Axlon Incorporated of Sunnyvale, today announced the release of its portable personal communication terminal for the home and business market.

Called the Axlon HOTLINE™ Personal Communication Terminal, it can be used to transmit and retrieve information from data bases as varied as a personal telephone directory to the New York Stock Exchange.

The Axlon HOTLINE Personal Communication Terminal measures 1⅝" x 3⁹⁄₁₆" x 6¾" and weighs less than 11 ounces providing a truly portable means to communicate with host computer

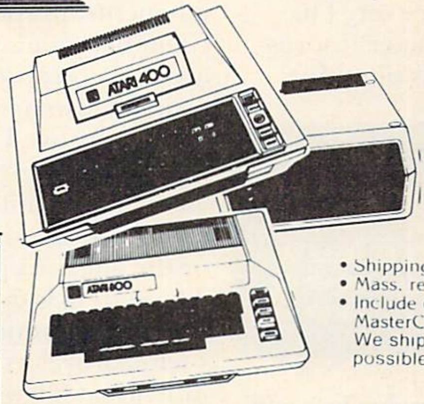
PRICES QUOTED OVER THE PHONE



Call Toll Free 800-343-5691

ATARI® Computers for people.™

NEC The end of the compromise.



PRICE EXAMPLES:

ATARI 800 with 16K memory	**$749**
ATARI 400 with 16K memory	**$329**
ATARI 810 Disk Drive	**$439**
ATARI Software 10% OFF	
NEW: Pro Football 16K (Cassette)	**$7.95**
Galactic Chase (Disc)	**$29.95**
Ghost Hunters (Disc) **$34.95** (Cass.)	**$29.95**
ATARI 410 Recorder	**$65**

- Shipping Address
- Mass. res. 5% sales tax
- Include check, money order, MasterCharge or VISA number

We ship by best means possible freight collect

Mass. res. (617) 843-1133

WARREN'S COMPUTER SYSTEMS
4 Elm St., Braintree, MA 02184
Please Dial 800-343-5691
SEND FOR CATALOG

OLYMPIC SALES COMPANY

Authorized Dealer
apple computer

NEW APPLE FAMILY SYSTEM

LARGE INVENTORY LOWER-THAN-EVER PRICES

- ★ 48K Apple II +
- ★ Disk w/controller
- ★ RF Modulator
- ★ 7 Personal & game disks

Retail: $2495.00 Call for BEST Price!

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

WE ALSO CARRY SOFTWARE!
Personal Software, Peach Tree Software, Microsoft, Micropro, Innovative American, System Plus and more.

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

Just some of the software in stock NOW—
Magic Window, Raster Blaster, Ultima, Home $ Minder, Tax Preparer, Pool 1.5, Typing Tutor, Olympic Decathalon, ABM, Robot War, Castle Wolfenstein, Cranston Manor, Expediter, OO-Topos, Cyborg, Gorgon, Falcons, Wizardry, Personal Filing System, PFS: Report, Flight Simulator. . .

Texas Instruments Home Computer

New—1982 Model with full typewriter-style keyboard, U/L case & more!
TI-99/4A New KEYBOARD **$369.95**

Retail $950.00 *Your Cost* **$369.95!**

New keyboard

	Retail	Your Cost
10" color monitor for 99/4	650.00	319.95
32K Exp. mem. module	399.95	314.95
Extended Basic, a MUST for 32K module	100.00	75.00
Speech synthesizer	150.00	129.95
Disk memory drive	500.00	397.95
RF modulator	69.50	49.50
Telephone coupler (modem)	225.00	189.95
Printer (solid state)	400.00	319.95

We carry a large inventory of software, & accessories

HEWLETT PACKARD

HP-11C Advanced Programmable Scientific LCD *Retail: $130.00* *Your Cost:* **$114.95**

HP-12C Advanced Programmable Financial LCD *Retail: $150.00* *Your Cost:* **$129.95**

	Retail	Your Cost
HP-125 New Microcomputer 64K CPU/Terminal/Keyboard	3750.00	3195.00
HP-85 Microcomputer—built-in printer/monitor	3250.00	2499.00
HP-83 Microcom—built-in mon	2250.00	1799.00
HP-2631B Printer, dot mtrx	3950.00	3199.00
HP82905A 80 col printer	945.00	755.00
HP-2601A Letter quality prter		3495.00
HP-82901M Dual master	2500.00	1999.00
Call us for the lowest prices on 7 disk drives		
Graphics plotter 7225	2450.00	2089.00
Personality module—a must for 7225	750.00	636.00
Graphics Tablet 9111A	2050.00	1699.00
HP-41CV	325.00	256.00
Card Reader for 41CV	215.00	171.00
Printer for 41CV	385.00	294.00
Quad Ram	95.00	84.95
Optical Wand for 41CV	125.00	99.95
HP-41C Calculator	250.00	188.95
Memory mod. for 41C		26.95
HP-97 Progrmble printer	750.00	579.95
HP-34C Progrmble scientific	150.00	117.95
HP-38C Progrmble business R/E	150.00	119.95
HP-32E Advanced scientific	55.00	48.95
HP-37E Business management	75.00	58.95

We carry a complete inventory of Libraries, accessories & supplies.

ATARI® COMPUTERS

Model		Retail	Your Cost
800	16K	1080.00	759.95
810	Disk Drive	600.00	469.95
Atari	VisiCalc	200.00	169.95
825	Impact Printer (Centronics 737)	1000.00	779.95

SUPER SPECIAL—ATARI 400 (16K)
Retail: $595.00 Your Cost: **$339.95**
Language Cart. not included—Opt'l at $54.95

Mattel 1670 Race Horse Computer
Ret. $125.00 Y/C: $24.95

ATARI® Programmable Color TV Game Video Console: *Retail:* $225.00 **$137.95**

MATTEL ELECTRONICS **INTELLIVISION** *Retail:* $325.00 *Your Cost:* **$234.95**

VIDEO TAPES

Min order: 3, mixed OK

SONY	L500	2 hr	$11.89
SONY	L750	3 hr	14.69
RCA	VK250	6 hr	13.95

TEXAS INSTRUMENTS

	Retail	You Pay
TI-59	295.00	179.95
TI-3011 LCD Stu Slide/NEW		18.95
TI-35SP LCD Sci		22.50
TI-40 LCD Sci/NEW		28.95
Bus. Anal. I		19.95
Bus. Anal. II		44.95
Bus. Card		39.95
MBA		54.95
Invest. Analyst		48.95
TI-54 Adv. LCS Sci/NEW		39.95
TI-5511 NEW		44.95
TI-57 Prog. Sci		39.95
TI-58C 480 Step. Prog.		89.95
PC-100C Print/Plot		149.95
LCD-Programmer/NEW		59.95

OHIO SCIENTIFIC

Professional Computers

	Retail	Your Cost
C8PDF—48K	3495.00	3195.00

- Dual 8" Drives
- 64 col x 32 line/color
- .7 MIPS *—FAST!*
- Many more stndard featres

Fortran & Pascal available

Many other OSI products available —at discounted prices, of course!

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

WE HONOR VISA and MASTERCHARGE

TELEX: 67 34 77

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

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

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

EPSON PRINTERS

	Retail:	Your Cost
MX80	645.00	499.00
Optional Graftrax Chip 80		95.00
MX80 FT	745.00	599.00
MX100	995.00	749.00

DIABLO 630 Letter Quality daisy wheel w/tractors
Retail: $3395.00
Y/C: $2595.00

PAPER TIGER PRINTERS

460	995.00	895.00
460G graphics	1094.00	969.95
560	1295.00	1099.00
560G graphics	1394.00	1195.00
445	795.00	695.00
445G	894.00	789.00

SANYO MONITORS High resolution

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

AMDEK (Leedex) High Quality Monitors

100	12" B/W, 12 MHz	179.00	139.95
100-G	12" Green, 12 MHz	199.00	174.95
300-G	12" Green, 18 MHz	249.00	199.95
Color I	13" Color, NTSC comp. input,	449.00	339.95

systems.

The terminal combines state-of-the-art features such as a built-in modem with more familiar features including an alphanumeric keyboard. The tactile keyboard consists of 43 functional keys arranged in typewriter sequence and provides a 64 upper case ASCII character set. The display is a 16 character fluorescent display which is tilted for

viewing. The terminal offers a 96 character display memory, which can be scrolled 16 characters at a time, and a 16 character display memory in the transmit mode.

The Axlon HOTLINE Personal Communication Terminal is designed for ease of use. The user dials the data base, waits for the connect signal, and then connects the modular headset jack to the terminal's modular telephone receptacle. The terminal also has a receptacle for connection of an ASCII serial printer should the user require hard copy of information retrieved.

For more information contact John Vurich or Robert Sultemeier, Axlon, Inc., 170 N. Wolfe Rd., Sunnyvale, CA 94086.

Medical Package For Apple III

Monument Computer Service has released a new software applications for the Medical Profession running on the Apple III Computer. The package, called the Medical Clinic, runs under the SOS operating system and is written in Business Basic. The package is designed for the multi-practitioner medical practice.

The package manages the physician's appointment schedules, does patient recall, prepares appointment logs, and provides for patient file management. The system also has a full accounts receivable system for managing daily transactions and payments, preparing monthly client bills, and reporting aged accounts receivable. The billing element also prepares standard AMA approved claim forms.

The system will handle a virtually unlimited patient base using either mini-floppy diskettes or the latest Apple hard disk. The system is designed to improve professional cash flow with such features as a superbill, individual bill preparation and cycle billing.

The package is available for $1,495.95 complete. A demonstration manual is also available for $50.00. Additional information is available from Monument Computer Service, Village Data Center, P.O. Box 603, Joshua Tree, CA. 92252. Technical questions and dealer inquiries should be directed to (714) 365-6668. Additional written information is available from the order center at (800) 854-0561 Ext. 802 (In California call 800-432-7257.)

Business Planner

Duosoft Corporation introduces BUSINESS PLANNER, a modeling package for entrepreneurs planning to start or expand a new business.

Designed to help develop viable business plans, the program groups labor, equipment and other costs into income-related projects.

Projects are combined into a model which generates graphical projections and estimated financial statements.

By exploring alternate models, you can evaluate critical timing and resource allocation decisions.

"What If" scenarios help you plan for the future and respond appropriately to changing demands.

You can compare actual results against the model in order to focus attention on problems that may harm future performance.

BUSINESS PLANNER is now available for the Apple II and III, and will soon be released for other machines.

For further information: Duosoft Corporation, Box 1827, Champaign, Illinois 61820.

Pascal Procedures For Business

Users Pascal Procedures Exchange Register (UPPER) has announced the release of "The Most Commonly Re-created Pascal Procedures for Business Application Programmers." This booklet contains UCSD p-System Pascal source code for user-friendly, bomb proof: screen input, access methods, printed report formatting, text formatting, data type conversions, and sample shell programs. These procedures can be incorporated into library units, segments, or used as in-line code.

Price: $19.95. Available from: Users Pascal Procedures Exchange Register, 1372 East 52nd Street, Chicago, IL 60615.

Financial Modeling Software Package

Osborne/McGraw-Hill has announced plans to distribute MicroFinesse, a financial modeling software package, with initial shipments to dealers commencing January, 1982. This move marks a major thrust by the McGraw-Hill Book Company into software distribution.

Written in Pascal, MicroFinesse runs on the Apple II microcomputer and will be distributed by Osborne/McGraw-Hill in the US and Canada only. The complete menu-driven package, with documentation, will sell for $495.00, available primarily through retail computer stores.

A financial modeling, forecasting and decision-making system, MicroFinesse was developed by the P-E Consulting Group, one of England's top management firms, with over 10 years experience in financial modeling.

According to Martin McNiff, Technical Group Manager at Osborne/McGraw-Hill, MicroFinesse is more than a spreadsheet package. "It offers planning capabilities seen before only on much larger systems, such as the ability to create investment and financial alternative models, as well as pro forma statements, sales productivity or profitability forecasts. Users can define target figures and use MicroFinesse to determine what must be done to meet those goals," says McNiff. He also points to the program's color graphics, model consolidations and report-generating features.

A significant commitment to after-sale support has also been announced by Osborne/McGraw-Hill, including a dedicated toll-free telephone "hot line" which will be in place at the time initial shipments commence.

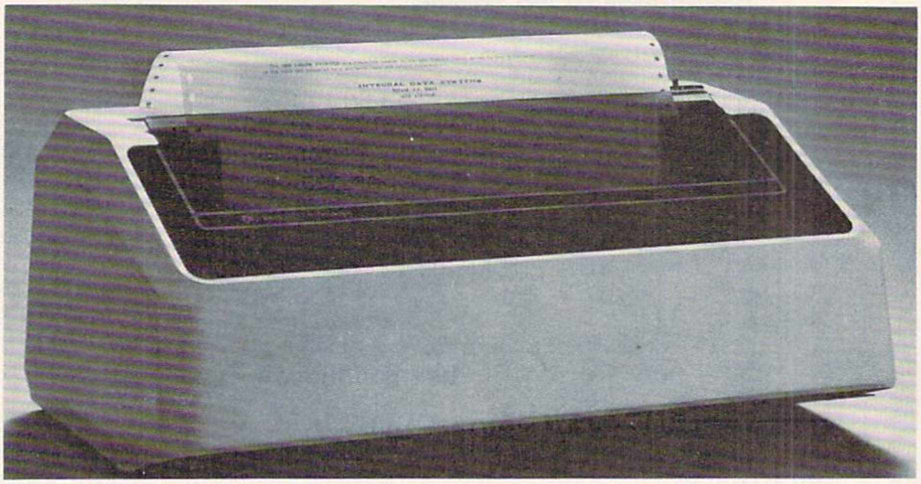
For more information, please contact: Chris Chambers, Sales and Marketing Director, Osborne/McGraw-Hill, (415) 548-2805.

Prism Printer From Integral Data Systems

A new color printer user-priced at $1,995 has been introduced by Integral Data Systems of Milford, New Hampshire. The Prism Printer™ is a low cost commercial color printer designed to compete with units costing three times as much.

The new 132-column dot matrix printer will produce eight colors using a four band ribbon which carries the process colors of cyan, magenta and yellow, as well as black.

"It's going to help define the expanding color graphics market," says Peter R. Eisenhauer, Integral Data Systems Vice President of Marketing. "There's a demand for color, primarily among business and professional users." Other immediate applications for the printer include the visual translation of scientific and medical data.



In addition to the color printer itself, Integral Data has plans to offer a number of collateral products which will facilitate the use of the Prism Printer in many key system environments. The first is expected to be an interface card for the Apple II and III which will have a graphics driver for the color printer resident on the card. Other products in the works include additional software drivers for Apple products as well as a color/graphics driver for the recently announced IBM Personal Computer.

The Prism Printer offers semi-automatic cut sheet feed, also a high-speed data mode. In the normal (correspondence) mode, the unit prints overlapping high density (24x9) matrix characters at up to 150 characters per second, bidirectionally. The high-speed data mode enables the user to select a standard density matrix and output large volumes of data at print speeds in excess of 200 cps.

Standard features include proportional spacing, enhanced (bold) text printing and standard print densities of 10, 12 or 16.7 characters per inch. The Prism Printer prints a full 132 characters per line at 10 pitch (characters per inch) with other pitches giving line lengths up to 220 columns on standard 15-inch-wide EDP paper.

Selectable features include automatic text justification, programmable horizontal and vertical tabbing, reverse paper feed, and "fine positioning" of characters of 1/120th of an inch. While the Prism Printer employs the standard ASCII upper- and lower-case 96-character set, up to four different 96-character sets can reside within the printer at the same time, for foreign language or custom character printing,

The Prism Printer is microprocessor controlled, with true "logic seeking" look-ahead capability and a high-speed slew for maximum output. It has a standard RS-232C serial interface as well as a Centronics-compatible

parallel interface. Serial transmission rates from 300 to 9,600 baud are switch selectable.

For more information on the Prism Printer and sample output, contact Integral Data Systems, Inc., Milford, New Hampshire 03055. (603) 673-9100; 800-258-1386; Telex: 953032.

New From Commodore

Valley Forge, PA, Nov. 12, 1981 – Commodore Business Machines has introduced its newest printer, the low-cost, high-speed CBM 8023P.

This latest addition to the growing line of CBM peripherals is a bidirectional, 136-column printer with both tractor and friction feed. The 8023P is dot-matrix, and prints 150 characters-per-second (CPS). It is available through Commodore dealers throughout the nation for $995.00.

The new CBM printer is designed to operate through software control, prints upper and lower case alphabetic characters, all graphic characters available with a Commodore computer, as well as user-defined characters.

The 8023P conforms to IEEE interface requirements and connects directly to a Commodore computer. It is designed to be used with the CBM floppy disk drives, and may be daisy-chained with other IEEE-488 devices.

Because the printer is an "intelligent" peripheral, it uses none of the computer's memory. In addition, the 8023P contains Random Access memory (RAM), which permits storage of formatting data.

****

A programmable character set and gamegraphics editor on cassette has been introduced by Commodore Business Machines, Inc., for users of its VIC 20™ personal computer.

Now available at authorized Commodore dealers throughout the nation for $14.95, the character set editor comes with a 16-page instruction manual and allows VIC users to create groups of 64, 128, or 192 programmable characters at a time and use them in BASIC programs. Each group of characters takes only one-half kilobyte (0.5K) of program space.

With the new character editor, Commodore VIC 20 users can create their own character set and easily modify letters, numbers, and graphics to include foreign language letters, mathematic and scientific symbols, or special "arcade" game graphics.

Commodore's new character set editor also allows VIC 20 users to save their newly-created character set on tape or disk for future use, and then easily insert the set in a BASIC program.

Along with the character set editor, also new from Commodore is the recently-introduced VIC 1515 low-cost dot-matrix printer. Available for $395, the VIC 1515 has a printing speed of 30 CPS, and prints any of the alphabetic, numeric, and graphic symbols common to the VIC.

From Krell Software Corp.

WAR OF THE SAMURAI is a game of combat and intrigue. Two to four players may compete in this original game that combines the strategic complexity of Go with the subtle dynamics of Chess. Detailed graphics.

Machines: Apple, PET, TRS-80, 16K, available on disk or cassette. $39.95

ALEXANDER THE GREAT is a vocabulary building game in a fantasy game context. Based on the Sword of Zedek, their best selling fantasy game. Alexander the Great introduces Aristotle as a mentor to the player. When called on, Aristotle poses vocabulary questions, and depending on the speed and accuracy of the player response, confers secret information. With Aristotle as an ally, the quest to overthrow Ra, The Master of Evil, assumes a new dimension of complexity. Players may select the level of vocabulary difficulty.

Machines: Apple, TRS-80, PET, 48K, available in two versions (K-8) & (9-College). $39.95.

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

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

Machines: Apple, PET, TRS-80, Atari, 16K, available on disk or cassette. $24.95

FIG NEWTON – full graphics Newton. This version of Isaac Newton presents all data in graphic form. Because data is graphic rather than symbolic, this game is suitable for very young children. Players may, however, select difficulty levels challenging to the most skilled adults.

Machines: Apple, PET, TRS-80, Atari, 16K, available on disk or cassette. $24.95

ODYSSEY IN TIME adventure game adds a new dimension of excitement and complexity to TIME TRAVELER. Players must now compete with the powerful and treacherous adversary in their exacting quest for victory.

To succeed they must vanquish this adversary in combat

that rages across 24 time periods.

ODYSSEY IN TIME includes all the challenges of TIME TRAVELER plus 10 additional eras, including those of Alexander the Great, Emperor Asoka of India, Attila the Hun, Genghis Khan. Each game is unique and may be interrupted and saved at any point for later play.

Machines: Apple, PET, TRS-80, Atari, 32K, available on disk or cassette. $39.95

Krell Software Corp.
21 Millbrook Drive
Stony Brook, NY 11790

From Strategic Simulations

Napoleon's Campaigns is a corps-level game simulating the last campaigns of Napoleon: Leipzig and Waterloo. It is an advanced-level, board-assisted computer game.

Each campaign is displayed on a colorful 18x21 hex grid map in Hi-Res Graphics. The game employs a unique system requiring orders to be sent and received through dispatches. The reports received vary in degree of accuracy based on a variety of historical factors. The computer acts as corps commander for each corps, interpreting the orders the corps receives and often acting on its own initiative. These features simulate the frustration experienced by commanders of the Napoleonic Era.

The game includes one diskette, rule book, player aid card, 2 two-sided map boards and 100 counters for $59.95.

Southern Command is a battalion-level simulation for the Israeli counterattack to cross the Suez Canal during the October War of 1973 against Egypt.

The Sinai battleground is displayed in Hi-Res Graphics on a 28x39 hex grid map which can be viewed on one screen or on twelve screens, using scrolling. More than ten unit types including tanks, halftracks, BDM's, infantry and Egyptian SAM sites (to combat Israeli airstrikes) are used in the two player and each of the four computer-as-opponent scenarios.

Modern warfare is accurately reflected in the ability of units to reorganize after they have been attacked and in the "Delayed Move" feature, allowing units to ambush moving enemy units. Each side also has the ability to sight hidden enemy units.

Southern Command is available with diskette, rule book, map and player aid card for $39.95.

Both games require a 48K Apple II with Applesoft in ROM and one disk drive.

From Strategic Simulations, 465 Fairchild Dr., Suite 108, Mountain View, CA 94043.

From Automated Simulations

Automated Simulations, Inc. has released a new MIND TOY, Ricochet, an original abstract strategy game designed exclusively for the home computer.

Ricochet is a game of subtle strategy combined with fast action and arcade-style graphics. The game can be played against any of four different computer opponents, or against another human.

The player maneuvers blocks to set up a shot at his opponent's goal and to protect his goal from attack. Each player has two launchers he can fire. His shots ricochet off the blocks, earning him points each time a block is hit, plus he gets bonus points for hitting his opponent's goal.

Before he can claim victory, the player must win two out of three (or three out of five) games. A match victory also boosts his personal Ricochet Player Rating, which measures his mastery of the game against other players.

Ricochet is available on cassette for the Atari 400/800 (16K with BASIC ROM cartridge) and TRS-80 (16K, Level II), or on disk for the Atari 400/800 (32K), TRS-80 (TRSDOS 32K) and APPLE (48K with Applesoft in ROM). $19.95

From Automated Simulations, P.O. Box 4247, Mountain View, CA 94040.

From Synergistic Software

Odyssey: The Compleat Adventure, is now available for the Apple II computer in the Applesoft BASIC language. This adventure game is expanded into three separate but interlocking programs. The programs have colorful high resolution animation as well as sound effects. Many different paths to the goal exist that will not trap the player or force repetition. Being a role playing game, player action determines alignment, charisma, wisdom, experience, etc. These features affect the outcome of friendship and battle encounterd during play.

The object of this game is to save a realm from an evil ruler. Starting alone on a large island you seek out gold, soldiers to join you, and useful tools while gaining experience. If you are successful and clever you can not only walk but also ride, fly, and sail.

With dozens of high resolution pictures and animation effects different each time you play, each game is unique. Careful planning and strategy are necessary to successfully complete this adventure. Requires 48K Apple II or Apple II Plus. Available in Integer or Applesoft for $30.00 from Synergistic Software, 5221 120th Avenue SE, Bellevue, WA 98006. (206)226-3216

LO•BALL COMPUTERS LO•BALL COMPUTE

ATARI
and more for less $
EPSON

ATARI

400 w/16K	$339.00
800 w/16K	$739.00
810 Disk Drive	$449.00
830 Modem	$159.00
850 Interface	$159.00
16K module	$85.00

EPSON

MX-70	CALL
MX-80	CALL
MX-80 F/T	CALL
MX-100	CALL
Graphics kit	$85.00

Interface cards and cables available for your machine!

We'll meet or beat any advertised price.

HEWLETT PACKARD

HP-85A w/16K	$2595.00
HP-125	$2995.00

TELEVIDEO

920C	$729.00
950	$939.00

ZENITH

Z-19	$725.00
Z-89	$2150.00

Call for more great prices on NEC, BMC, Northstar, Altos, Soroc, Amdek ... *and others.*

LO-BALL COMPUTERS
P.O. Box 1818
Beaverton, OR 97075
APO/FPO welcomed

CALL (503) 641-0211

Ordering Information: For fastest service, send money order, cashier's check or bank wire. Visa and MC orders, add 3%. Personal checks accepted (allow minimum 10 days to clear). Hours 9-5, M-F.

Call For Our Free Price List

LO•BALL COMPUTERS LO•BALL COMPUTE

ORDER TOLL-FREE

consumer computers Mail Order DISCOUNTS

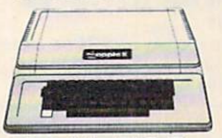
FREE SHIPPING*

NEC Microcomputer

CALL FOR BEST PRICE

apple computer

Authorized Dealer



CALL FOR BEST PRICE

DISK DRIVES AVAILABLE

NOW SELLING ZENITH XEROX — CALL FOR PRICES

COMING SOON! OSBORNE

MORE DISCOUNT PRICES

16K RamBoard by ConComp Industries	130	Joystick II by Keyboard Co	45	VisiTerm	129
Silentype Printer w/interface card	349	Music System (16 voices)	479	WordStar (Apple 80 col. version)	249
Hayes Micromodem II	299	CPS Multi-function card	189	Dow Jones Portfolio Evaluator	45
Novation Apple-Cat	339	Sup-R-Terminal 80 column cards by M&R	329	Apple Writer	65
Videx Videoterm 80 column card	269	We carry all California Computer		Apple DOS Toolkit	65
Z-80 Softcard by Microsoft	299	System Cards	CALL	Apple Plot	60
16K RamCard by Microsoft	169	VisiCalc version 3.3	159	Tax Preparer	99
Graphics Tablet	619	VisiFile (NEW data base manager)	199	Real Estate Analyzer	129
Thunderclock Plus clock/calendar	119	VisiTrend/VisiPlot	219	Personal Filing System (PSF)	85
23 Key Numeric Keypad by Keyboard Co.	120	VisiDex	159	Systems Plus Accting Software	CALL

PRINTERS

Anadex 9501 w/2K Buffer	1349
C. Itoh Starwriter 25 CPS daisywheel	1449
C. Itoh Starwriter 45 CPS daisywheel	1649
Epson MX-70	CALL
Epson MX-80 & MX-80 F/T	CALL
Epson MX-100	CALL
NEC 8023 Impact Dot Matrix	695
NEC Spinwriters (Latest models)	CALL
Paper Tiger IDS-445G w/graphics	699
Paper Tiger IDS-460G w/graphics	949
Paper Tiger IDS-560G w/graphics	1249
Silentype Printer w/Apple interface	349
Qume Sprint Daisywheels (Latest models)	CALL
Diablo 630 Daisywheel 40 CPS	1795

ORDER TOLL-FREE 800 854-6654

in California and outside U.S.
(714) 698-8088

MONITORS

Amdek/Leedex Video 100 12" B&W	155
Amdek/Leedex Video 100G 12" Green Phospher	179
Amdek (Hitachi) 13" Color w/audio output	179
NEC 12" Green Phospher Display JB-1201M	CALL
NEC 12" Lo-Res Color Display	CALL
NEC 12" Hi-Res RGB Color Display	CALL
Sanyo 9" B&W Display	185
Sanyo 9" Green Phospher Display	CALL
Sanyo 12" B&W Display	269
Sanyo 12" Green Phospher Display	285
Sanyo 13" Color Display	449
Zenith 12" Green Phospher Display ZVM-121	149

CALL FOR BEST PRICE

commodore CBM

Business computer

CALL FOR BEST PRICE

commodore PET

Personal computer

CALL FOR BEST PRICE

DISK DRIVES AVAILABLE SOON!

VIC-20 PERSONAL COMPUTER

$259

• color • sound • graphics

WE STOCK 1000S OF HARDWARE AND SOFTWARE ITEMS, PLEASE WRITE OR CALL FOR LIST

consumer computers

Mail Order
8314 PARKWAY DR.
LA MESA, CA 92041

Ordering information: Phone orders using VISA, MASTERCARD, AMERICAN EXPRESS, DINER'S CLUB, CARTE BLANCHE, bank wire transfer, cashier's or certified check, money order or personal check (allow ten days to clear). Unless prepaid with cash, please add 5% for shipping, handling and insurance. (minimum 5.00). California residents add 6% sales tax. Institutions and corporations please send for a written quotation. All equipment is subject to price change and availability without notice. All equipment is new and complete with manufacturer's warranty (usually 90 days). Showroom prices may differ from mail order prices.

SPECIALS on INTEGRATED CIRCUITS

6502	7.45	10/6.95	50/6.55	100/6.15
6502A/6512A	8.40	10/7.95	50/7.35	100/6.90
6520 PIA	5.15	10/4.90	50/4.45	100/4.15
6522 VIA	6.45	10/6.10	50/5.75	100/5.45
6532	7.90	10/7.40	50/7.00	100/6.60
2114-L200		3.75	25/3.50	100/3.25
2114-L300		3.15	25/2.90	100/2.65
2716 EPROM		7.00	5/6.45	10/5.90
2532 EPROM				14.50
6116 Hitachi 2K x 8 CMOS RAM				**14.50**
4116				**8 for 17**
Zero Insertion Force 24 pin Socket				2.00
6550 RAM (PET 8K)				12.70
S-100 Wire Wrap Socket				2.40

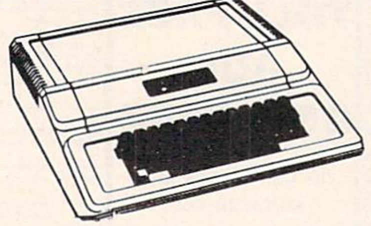
A P Products 15% OFF
A P Hobby-Blox 15% OFF

THE STAR MODEM

From Prentice/Livermore Data Systems

RS232 MODEM	**SALE**	**$128**
IEEE 488 MODEM	**SALE**	**$199**
RS232 CCITT		**$170**
IEEE 488 CCITT		**$270**

We carry Apple II+ from Bell & Howell



fullFORTH+ for Apple II **$75**
A full-featured FORTH with enhancements. Conforms to FORTH Interest Group standards.

VIC 3K RAM (with 2 ROM Sockets)	65
Write for VIC Software List	
REVERSAL (Spracklen) 32K Apple	28.00
Data Manager (Lutus) 24K Apple	40.00
Energy Miser - PET, Apple, Zenith	24.50
Histo-Graph (Boyd) 48K Apple	24.50
Data-Graph (Boyd) 48K Apple	40.00
Apple II User's Guide (Osborne)	12.00
Introduction to Pascal (Sybex)	10.30
Pascal Handbook (Sybex)	12.00
Musical Applications of Micros (Chamberlin)	20.00
Basic FORTRAN (Coan)	7.25
CP/M Handbook (with MP/M) Zaks	11.85
Programming the 6502 - Zaks	10.00
Microprocessor Interfacing Techniques	13.00
PET/CBM Personal Computer Guide (Osborne)	12.00
PET and the IEEE-488 Bus (GPIB)	12.25
6502 Assembly Language Prog. - Levanthal	13.50
Z-80 Assembly Language Prog. - Levanthal	12.75
8080A/8085 Assembly Language Programming	12.75
Z8000 Assembly Language Programming	16.00
6809 Assembly Language Programming	13.50
6800 Assembly Language Programming	12.75
The 68000 Microprocessor Handbook	5.75
BASIC and the Personal Computer (Dwyer)	11.00

commodore

CBM-PET SPECIALS

	list	SALE
8023 Printer - 136 col, 150 cps bi-directional	(995)	775
8300 (Diablo 630) Daisy Wheel - 40 cps bi-directional	(2250)	1725
8032 80 x 25 CRT, business keyboard	(1495)	1100
Super Pet	(1995)	1600
8096 Board (extra 64K RAM for 8032)	(500)	400
8050 Dual Disk Drive - 1 megabyte	(1795)	1345
8250 Dual Disk Drive - 2 megabyte	(2195)	1760
CBM IEEE Modem	(395)	199
4016 full size graphics keyboard	(995)	795
4032 full size graphics keyboard	(1295)	999
4040 Dual Disk Drive - 330,000 bytes	(1295)	999
2031 Single Disk Drive - 165,000 bytes	(695)	560
4022 Tractor Feed Printer	(795)	630
C2N External Cassette Deck	(75)	65
VIC 20 Color Computer	(299)	259
VIC 1515 Graphic Printer	(395)	325
Used CBM/PET Computers		CALL
8024-7 High Speed Printer	(1995)	1345

WE WILL MATCH ANY ADVERTISED PRICE

*** EDUCATIONAL DISCOUNTS ***

Buy 2 PET/CBM Computers, receive 1 FREE

WordPro 3 Plus - 32K CBM, disk, printer	200
WordPro 4 Plus - 8032, disk, printer	325
OZZ Data Base System for CBM 8032	335
VISICALC for PET, ATARI, or APPLE	155
SM-KIT - Super PET ROM Utilities	**40**
Programmers Toolkit - PET ROM Utilities	35
PET Spacemaker II ROM Switch	36
2 Meter PET to IEEE or IEEE to IEEE Cable	40
Dust Cover for PET	7
IEEE-Parallel Printer Interface for PET	110
IEEE-RS232 Printer Interface for PET	120
The PET Revealed	17
Library of PET Subroutines	17

4 PART HARMONY MUSIC SYSTEM for PET

The Visible Music Monitor, by Frank Levinson, allows you to easily enter, display, edit, and play 4 part harmony music. Includes whole notes thru 64ths (with dotted and triplets), tempo change, key signature, transpose, etc. The KL-4M unit includes D to A converter and amplifier ready to hook to your speaker.

KL-4M Music Board with VMM Program **$60**

Watanabe Intelligent Plotter

WATANABE WX4671 Plotter	**1195**
WATANABE WX4675 6-pen Plotter	**1445**

DISK SPECIALS

SCOTCH (3M) 5"	10/2.75	50/2.65	100/2.60
SCOTCH (3M) 8"	10/2.80	50/2.70	100/2.65
Verbatim 8" Double Dens.	10/3.45	50/3.35	100/3.20
Verbatim 5" Datalife	10/2.45	50/2.40	100/2.35
(add 1.00 for Verbatim 5" plastic storage box)			
BASF 5" soft	10/2.40	20/2.35	100/2.30
Wabash 5" in Plastic Box	10/2.70	50/2.60	100/2.50
Wabash 8" in Plastic Box	10/2.75	50/2.65	100/2.55

WE STOCK MAXELL DISKS

Diskette Storage Pages		10 for 3.95
Disk Library Cases	8" - 2.85	5" - 2.15
Disk Hub Rings	8" - 50 @ 7.50	5" - 50 @ 6.00

CASSETTES - AGFA PE-611 PREMIUM

High output, low noise, 5 screw housings.

C-10	10/.56	50/.50	100/.48
C-30	10/.73	50/.68	100/.66

All other lengths available. Write for price list.

SPECIALS

EPSON MX-80 Printer	
EPSON MX-80 F/T Printer	
EPSON MX-70 Printer	
EPSON MX-100 Printer	
Centronics 739 Printer with dot graphics	675
STARWRITER Daisy Wheel Printer	**1445**
Zenith ZVM-121 Green Phosphor Monitor	115
Amdek Color Monitor	355

ALL BOOK and SOFTWARE PRICES DISCOUNTED
OSBORNE/McGraw-Hill, HAYDEN, SYBEX, etc.

Synertek Systems

SYM-1 Microcomputer SALE	**199**
SYM BAS-1 BASIC or RAE 1/2 Assembler	85
KTM-2/80 Synertek Video and Keyboard	349
KTM-3/80 Synertek Tubeless Terminal	385

ZENITH data systems

Z90-80 64 K	2170
Z90-82 64K, 1 double dens. drive	2395
Z89-0 48K	1950
Z89-1 48K, 1 drive	2150
Z67 10 Megabyte + Floppy Drive	4495
Z37 1.3 Megabyte Dual Floppy	1495
Z25 High Speed Printer	1195
Z19 Video Terminal (VT-52 compatible)	670
ZVM-121 Green Phosphor Monitor	119
All Zenith Software discounted	

ATARI SPECIALS

800 Computer	779	410 Recorder	59
400 - 16K	339	Pilot	68
810 Disk Drive	449	Microsoft BASIC	68
825 Printer	629	Educ. Series	20% off
850 Interface	175	**MISSILE COMMAND**	**32**
822 Printer	359	**ASTEROIDS**	**32**
Paddle Pair	17	**STAR RAIDERS**	**32**
Joystick Pair	17	Space Invaders	32
16K RAM	85	Music Composer	45
Assembler/Editor	46	Chess	30
TeleLink	20	Super Breakout	30

Write for prices on other Atari items.

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

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

KMMM Pascal for PET/CBM $85

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

Requires 32K Please specify configuration.

EARL for PET (disk file based) $65

Editor, Assembler, Relocater, Linker

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

RAM/ROM for PET/CBM

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

RAM/ROM is compatible with any large keyboard machine. Plugs into one of the ROM sockets above screen memory to give you switch selected write protectable RAM.
Use RAM/ROM as a software development tool to store data or machine code beyond the normal BASIC range. Use RAM/ROM TO LOAD A ROM image where you have possible conflicts with more than one ROM requiring the same socket. Possible applications include machine language sort (such as SUPERSORT), universal wedge, Extramon, etc.

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

SUPERSORT by James Strasma $35

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

SuperGraphics

by John Fluharty **$30**

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

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

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

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

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

FLEX-FILE for PET/CBM Computers

Self Calculating
DATA BASE
REPORT WRITER
MAILING LIST

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

RANDOM ACCESS DATA BASE

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

MAILING LABELS

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

REPORT WRITER

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

FLEX-FILE by Michael Riley $60

Please specify equipment configuration when ordering.

Low Cost Disk Drive for PET/CBM

PEDISK II from cgrs Microtech is a new disk system ready to plug into your large keyboard PET/CBM.
PEDISK II offers speed, reliability, IBM compatibility.
Complete system prices with DOS and cable:

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

PROGRAM YOUR OWN EPROMS

Branding Iron for PET/CBM $79

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

CBM Software

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

FORTH for PET

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

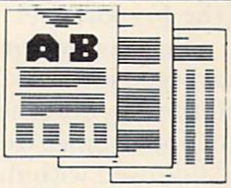
Features include:
- full FIG FORTH model.
- all FORTH 79 STANDARD extensions.
- structured 6502 Assembler with nested decision making macros.
- full screen editing (same as when programming in BASIC).
- auto repeat key.
- sample programs.
- standard size screens (16 lines by 64 characters).
- 150 screens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

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

Available soon:
Metacompiler for FORTH $30
simple metacompiler for creating compacted object code which can be executed independently (without the FORTH system).

PaperMate 60 COMMAND WORD PROCESSOR

by Michael Riley



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

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

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

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

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

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

To order Paper-Mate, please specify configuration.

Paper-Mate on disk or tape 40.00

BASIC INTERPRETER $200

Designed to support the CBM 8096 (8032 with add-on 64K board). A full interpreter implementation to automatically take advantage of the extra memory available to the 8032.

BPI General Ledger – 8032/8050	300
BPI Accounts Receivable – 8032/8050	300

Hayden Software

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

CASH MANAGEMENT SYSTEM $45

Easy to use. Keeps track of cash disbursements, cash receipts, cash transfers, expenses for up to 50 categories.

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

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

252 Bethlehem Pike
Colmar, PA 18915
215-822-7727

A B Computers

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

Sirius Software Announces Addition To Their Product Line

The Joyport for the Apple Computer provides expansion of the game paddle port to allow the use of four paddles (with all buttons functional) or to allow the use of one or two Atari™ type joysticks. Atari type joysticks are fast, simple and typically retail for less than $10.00 each. Some of the features of the Joyport are:

Two connectors to accept Atari type joysticks. All four axis and the fire buttons are fully functional. Since the analog inputs are not used with the Atari type joystick the response is very fast. No delays between reads is needed. The Atari joystick can be read just as fast as the keyboard.

Two 16 pin sockets to accept standard Apple style paddles, joysticks or other devices designed to operate from the Apple's game paddle port.

A switch to change from Atari to Apple controls. Both may remain plugged in. No need to unplug one control to use the other.

A switch to select either of the two Joyport game paddle sockets to be read as paddles 0 and 1 or even to allow your software to select which set of paddles will be read as paddles 0 and 1.

Total compatibility with existing software and hardware. Even the use of the Shift Key to TTL input #3 modification that many people use is compatible. No modification to the Apple is necessary and the Joyport does not take up any peripheral slots.

The Joyport sets next to the Apple with all sockets and switches easily accessible. The Atari type sockets are mounted on the front of the device and the Apple type sockets are inside the box. A convenient snap on cover provides a strain relief for the Apple type sockets and still allows changing them without getting inside the Apple.

A wide variety of software is being developed to take advantage of the added features of the Joyport. This includes most products that Sirius Software will be publishing as well as software from other vendors. New products available from Sirius Software that will be compatible include Foosball — a four player action game that is included FREE with each Joyport purchased prior to Christmas, 1981. Cops and Robbers, PGE — The Pascal Graphics Editor, and Outpost are compatible with the Joyport.

Existing software may be easily modified. The small (20 to 100 bytes) driver is relocatable. The presence or absence of the Joyport can be determined by your software and automatically enabled. The software can also allow simultaneous use of the keyboard and either the game paddles or the Atari type joysticks.

The custom PC board is machine wave-soldered, hand inspected and bench tested.

Fully documented with source code listings of sample driver routines.

Write for details to Sirius Software Inc., Joyport, 2001 Arden Way, #2, Sacramento, CA 95825.

Dial/Data Gives Micro Users Access To Financial Markets

A new software program that will give microcomputer users immediate access to financial markets has been developed by Remote Computing Corporation (RCC). The program is called "Dial/Data" since users can dial RCC's network over normal telephone lines, log on and automatically retrieve data on commodities, securities and options.

According to Alan J. Schnelwar, Vice President of Eastern Support and Development at RCC, Dial/Data will add a new dimension to the microcomputer.

"We've developed a complete, easy-to-use data service that will allow users to sit in their homes and offices and have all the daily and historical price information necessary to track any investment.

Schnelwar explained that micro users will receive the same data used in the Merlin service by investment professionals.

Dial/Data features the largest collection of time-shared data on commodity futures, including open, high, low, close, volume and open interest for all commodities on all major exchanges.

The program will also supply all New York and American Stock Exchange closing prices including open, high, low, close, volume and uptick volume, and all currently traded option prices including high, low, close, volume, open interest and the closing price of the underlying stock.

"For Apple II users, RCC provides programs to dial our Burroughs B-7700 main frame and automatically retrieve prices and to create and maintain data files on diskette.

"In addition," she said, "The Apple II data files are compatible with statistical software packages available from other sources such as Compu-trac and Orion."

An important feature of Dial/Data is its ability to handle dynamic symbol changing.

New Product releases are selected from submissions for reasons of timeliness, available space, and general interest to our readers. We regret that we are unable to select all new product submissions for publication. Readers should be aware that we present here some edited version of material submitted by vendors and are unable to vouch for its accuracy at time of publication. ©

COMPUTER CENTER

31 East 31st St. *(between Madison & Park Ave.)* New York
480 Lexington Ave. *(between 46th & 47th St.)* New York
21 West Street *(1 Block from Bklyn Batt. Tun.)* New York

presenting the LARGEST SELECTION OF SOFTWARE EVER ASSEMBLED...

for ATARI® • APPLE® • PET® • and other Microcomputers

at SUPER DISCOUNT PRICES!

ATARI

PHYSICS (AT)	24.50
GREAT CLASSICS (AT)	24.50
BASIC PHYSCOLOGY (AT)	24.50
PRINCIPLES OF ECONOMICS (AT)	24.50
SPELLING (AT)	25.50
BASIC ELECTRICITY (AT)	24.50
BASIC ALGEBRA (AT)	24.50
32 K RAMCRAM (AT)	180.00
KINGDOM (AT)	13.55
STATISTICS I (AT)	17.95
BLACKJACK (AT)	13.55
BIORYTHM (AT)	13.55
HANGMAN (AT)	13.55
SPACE INVADERS (AT)	35.95
EUROPEAN CAPITALS (AT)	13.55
MORTGAGE LOAN (AT)	13.55
STATES & CAPITALS (AT)	13.55
EDUCATION SYSTEM (AT)	22.50
ATARI BASIC (AT)	53.95
ASSEMBLER DEBUG (AT)	53.95
BASKETBALL (AT)	35.95
ASTEROIDS (AT)	35.95
SUPER BREAKOUT (AT)	35.95
MUSIC COMPOSER (AT)	53.95
COMPUTER CHESS (AT)	35.95
MISSILE COMMAND (AT)	35.95
STAR RAIDERS (AT)	44.95
TELELINK (AT)	22.50
PADDLES (AT)	17.95
JOYSTICKS (AT)	17.95
U.S. HISTORY (AT)	24.50
U.S. GOVERNMENT (AT)	24.50
SUPERVISORY SKILLS (AT)	24.50
WORLD HISTORY (AT)	24.50
BASIC SOCIOLOGY (AT)	24.50
PERSONAL FINANCE (AT) [D]	69.95

ADVENTURE INTERNATIONAL

ADVENTURE HINT SHEET	9.95
ADVENTURE (1,2,3) [D] (AP,AT)	35.95
ADVENTURE (4,5,6) [D] (AP,AT)	35.95
ADVENTURE (7,8,9) [D] (AP,AT)	35.95
ADVENTURE (10,11,12) [D] (AP,AT)	18.95
ADVENTURE (specify 1-12) (AP,AT)	17.95
PLANETOIDS [D] (AP)	17.95
PLANETOIDS (AP)	17.95
POKER (AP)	17.95
POKER (AP) [D]	18.95
KID-VENTURE #1 (AP)	17.95
STAR TREK 3.5 (AT)	17.95
MOUNTAIN SHOOT (AT)	13.45
SUNDAY GOLF (AT)	13.45
ANGLE WORMS (AT)	13.45
DEFECTIONS (AT)	13.45
GALACTIC EMPIRE (AT)	17.95

AVALON HILL

MIDWAY (AT,AP,P)	13.50
NUKE WAR (AT,AP,P)	13.50
PLANET MINERS (AT,AP,P)	13.50
CONVOY RAIDER (AT,AP,P)	13.50
B1 BOMBER (AT,AP,P)	13.50
LORDS OF KARMA (AT,AP,P)	18.00
CONFLICT 2500 (AT,AP,P)	13.50
COMPUTER ACQUIRE (AP,P)	18.00
TANK TICS (AP,AT)	21.50

AUTOMATED SIMULATION

TUESDAY QUARTERBACK [D] (AP)	26.95
STAR WARRIOR [C,D] (AP,AT)	35.95
THREE PACK [D] (AP,P)	45.00
STARFLEET ORION [C,D] (AP)	22.50
STARFLEET ORION [C] (AT,P)	22.50
INVASION ORION [C,D] (AP,AT)	22.50
INVASION ORION [C] (AT,P)	22.50
APSHAI [D] (AP)	35.95
APSHAI [C] (P)	35.95
CRUSH, CRUMBLE & CHOMP [D] (AP)	26.95

AUTOMATED SIMULATION

RYN [D,C] (AP,AT)	17.95
RYN [C] (P,AT)	17.95
MORLOC [C,D] (AP)	17.95
MORLOC [C] (P,AT)	17.95
RIGEL [C,D] (AP,AT))	26.95
RIGEL [C] (AT,P)	26.95
HELLFIRE [D] (AP)	35.95
HELLFIRE [C] (P)	35.95

QUALITY SOFTWARE

6502 DISASSEMBLER (AT)	10.55
ALI BABA (AT) [D]	28.95
ATARI ASSEMBLER (AT)	22.50
BABBLE (AP) [D]	17.95
BATTLESHIP COMMANDER [D] (AP)	17.95
FASTGAMMON [D] (AP)	22.50
FASTGAMMON (AP,AT)	17.95
FRACAS ADVENTURE [D] (AP)	22.50
TANK TRAP (AT)	10.55
TANK TRAP (AT) [D]	13.55
TARI TREK (AT)	10.55
TARI TREK (AT) [D]	13.55
FORTH (AT) [D]	72.00

PERSONAL SOFTWARE

DESK TOP PLAN II [D] (AP)	175.00
MONTY MONOPOLY [D] (AP)	31.00
VISICALC [D] (AT,P,AP)	170.00
ZORK (T) [D]	35.95
VISIDEX (AP) [D]	170.00
VISIPLOT (AP) [D]	162.00
VISITERM (AP) [D]	135.00
VISITREND (AP) [D]	210.00
VISIFILE (AP) [D]	220.00

CALIF. PACIFIC

3-D GRAPHICS (AP) [D]	35.95
AKALAPETH (AP) [D]	31.50
APPLE OIDS (AP) [D]	26.95
FENDER BENDER (AP) [D]	22.50
RASTER BLASTER (AP) [D]	26.95
BUDGE'S SPACE ALBUM (AP) [D]	35.95
BUDGE'S TRILOGY (AP) [D]	26.95

STRATEGIC SIMULATIONS

COMPUTER AMBUSH [D] (AP)	51.50
COMPUTER BISMARCK [D] (AP)	51.50
COMPUTER CONFLICT [D] (AP)	35.00
COMPUTER NAPOLEONICS [D] (AP)	51.50
COMPUTER QUARTERBACK [D] (AP)	35.00
AIR COMBAT [D] (AP)	51.50
WARP FACTOR [D] (AP)	35.00
CUTHROATS & CARTELS [D] (AP)	35.00
OPERATION APOCALYPSE [D] (AP)	51.50
TORPEDO FIRE [D] (AP)	51.50
PRESIDENT ELECT [D] (AP)	31.50
MAJOR LEAGUE BASEBALL [D] (AP)	31.50

SUB-LOGIC

3D GRAPHICS (AP)	45.00
3D GRAPHICS [D] (AP)	53.00
A-2-FS1 FLIGHT SIMULATOR (AP)	22.00
A-2-FS1 FLIGHT [D] (AP)	29.00

MICROSOFT SOFTWARE

ADVENTURE [D] (AP)	25.50
OLYMPIC DECATHALON [D] (AP)	25.00
TYPING TUTOR (AP) [D]	22.50
TYPING TUTOR (AP)	13.55
Z-80 SOFTCARD [D] (AP)	315.00
16k RAM BOARD (AP)	165.00

ARCADE PLUS

GHOST HUNTER (AT)	26.95
GHOST HUNTER (AT) [D]	31.50

ON LINE SYSTEMS

HI-RES ADVEN. #0 (AP) [D]	17.95
HI-RES ADVEN. #1 [D] (AP)	22.50
HI-RES ADVEN. #2 [D] (AP,AT)	29.00
HI-RES ADVEN. #3 [D] (AP)	31.00
HI-RES CRIBBAGE [D] (AP)	22.50
PADDLE GRAPHICS [D] (AP)	36.00
TABLET GRAPHICS [D] (AP)	44.95
MISSILE DEFENSE [D] (AP)	26.95
SUPERSCRIBE II [D] (AP)	115.00

SIRIUS

CYBER STRIKE [D] (AP)	36.00
STAR CRUISER [D] (AP)	22.50
GORGON [D] (AP)	33.00
PHANTOM FIVE [D] (AP)	36.00
SPACE EGGS [D] (AP)	26.95

SYNERGISTIC SOFTWARE

DUNGEON & WILDERNESS [D] (AP)	29.00
ARCTURUS [D] (AP)	31.50
DUNGEON [D] (AP)	15.75
ODYSSEY [D] (AP)	27.00
HIGHER GRAPHICS [D] (AP)	31.50
WILDERNESS (AP)	15.75
WILDERNESS [D] (AP)	18.00
PROGRAM LINE EDITOR [D] (AP)	36.00

BRODERBUND

GALACTIC EMPIRE (AP) [D]	22.50
ALIEN RAIN (AP) [D]	21.00
HYPER HEAD ON (AP) [D]	22.50
GALACTIC REVOLUTION (AP) [D]	22.50
APPLE PANIC (AP) [D]	26.95
TAWALA'S REDOUBT (AP) [D]	26.95
GALACTIC TRADER (AP) [D]	22.50
SNOOGLE (AP) [D]	32.95

MUSE COMPANY

ABM [D] (AP)	22.50
APPILOT II [D] (AP)	75.00
BEST OF MUSE [D] (AP)	35.95
GLOBAL WAR [D] (AP)	22.50
SUPER TEXT II [D] (AP)	135.00
THREE MILE ISLAND [D] (AP)	35.95
ROBOT WARS [D] (AP)	32.00
THE VOICE [D] (AP)	35.95

IRIDIS

IRIDIS 1 (AT)	8.95
IRIDIS 1 (AT) [D]	11.75
IRIDIS 2 (AT)	14.50
IRIDIS 2 (AT) [D]	16.95

SPECTRUM COMP.

GALACTIC CHASE AT [D]	26.95
GALACTIC CHASE AT	22.50

CRYSTAL COMP.

WORLD WAR III AP, AT [D]	26.95
GALACTIC QUEST AP, AT [D]	26.95
WATERLOO AP, AT [D]	44.95
HOUSE OF USHER AP, AT [D]	26.95
QUEST FOR POWER AP, AT [D]	35.95
FANTASYLAND 2041 AP, AT [D]	53.95
SANDS OF MARS AP, AT [D]	35.95
PROTECTOR AP, AT [D]	35.95
FORGOTTEN ISLAND AP, AT [D]	35.95
LASAR WARS AP, AT [D]	26.95
IMPERIAL WALKER AP, AT [D]	26.95
LITTLE CRYSTAL AP, AT [D]	35.95

HAYDEN

SARGON II (AP)	25.00
SARGON II (AP) [D]	30.00

K-BYTE

KRAZY SHOOTOUT (AT) CARTRIDGE	44.95

If you don't see it listed, write... we probably have it in stock!

Check program desired. Complete ordering information and mail entire ad. Immediate Shipments from stock.

Ship the above programs as checked to:

Mr./Mrs. ____________

Address ____________

City ____________

State ____________ Zip ________

I have a ____________ (Name of Computer)

with ____________ K memory

Number of Programs Ordered .... ________

Amount of order .... ________

N.Y. residents add Sales Tax .... ________

Add shipping anywhere in the U.S. 3.00

Total amount enclosed .... ________

Charge my: ☐ Master Charge ☐ Visa

Signature ____________

Card No. ________ Expires ________

COMPUTE/FEB. '82

Personal Checks please allow 3 weeks.

KEY:
AT-Atari
AP-Apple
P-Pet
D-on Disc.
C-Cassette

If not marked-Cassette

Mail to:

DIGIBYTE SYSTEMS CORP.
31 East 31st Street, New York, N.Y. 10016
OUTSIDE NEW YORK CALL TOLL FREE (800) 221-3144
IN NEW YORK CALL (212) 889-8130

Prices subject to change without notice.

Advertisers Index

COMPUTE! The 6502 Resource Magazine PET•ATARI•APPLE OSI•KIM•SYM•AIM

For Fastest Service, Call Our **Toll-Free** US Order Line
800-345-8112
In Pennsylvania call 800-662-2444

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**______
☐ **Don't yet have one...**

☐ Please enter my 1 year (12 issue) subscription to **COMPUTE!**
☐ New subscription ☐ Renewal subscription
☐ $20.00 US subscription
☐ $25.00 Canada and international surface mail. Payment in US funds, drawn on US bank or money order.
☐ Bill me (US only). $1.00 billing fee

Name, Address: ______

Charge my: ☐ Visa ☐ MC
Number______ Expires____/____/

2 3 4 5 6 7 8 9 10 11 12

COMPUTE! The 6502 Resource Magazine PET•ATARI•APPLE OSI•KIM•SYM•AIM

For Fastest Service, Call Our **Toll-Free** US Order Line
800-345-8112
In Pennsylvania call 800-662-2444

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**______
☐ **Don't yet have one...**

☐ Please enter my 1 year (12 issue) subscription to **COMPUTE!**
☐ New subscription ☐ Renewal subscription
☐ $20.00 US subscription
☐ $25.00 Canada and international surface mail. Payment in US funds, drawn on US bank or money order.
☐ Bill me (US only). $1.00 billing fee

Name, Address: ______

Charge my: ☐ Visa ☐ MC
Number______ Expires____/____/

2 3 4 5 6 7 8 9 10 11 12

COMPUTE! The 6502 Resource Magazine PET•ATARI•APPLE OSI•KIM•SYM•AIM

Foreign Readers

New subscription rates for **COMPUTE!** readers outside of the US.
☐ $25 Anywhere/Surface Mail (2-4 months delivery) FI=4,6,8
☐ $38 Europe/Air Delivery (7-10 days) FI=3
☐ $48 Middle East, North Africa, Central America/Air Mail FI=5
☐ $88 South America, South Africa, Far East, Australia/Air Mail FI=7

Name ______
Address ______

☐ Check here if renewal subscription.

Country ______

Payment must accompany this card.
Payment in US funds drawn on a US bank, International Money Order or charge card. VISA ☐ MasterCard ☐.
Account No. ______, Expires ____/____/

2 3 4 5 6 7 8 9 10 11 12

The Editor's Feedback:

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**______
☐ **Don't yet have one...**

During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

Content
Best Article This Issue (page **#**, title)

Other suggestions

2 3 4 5 6 7 8 9 10 11 12

Place
Stamp
Here

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008
USA

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 236 BROOMALL, PA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008

Place
Stamp
Here

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008
USA

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 236 BROOMALL, PA

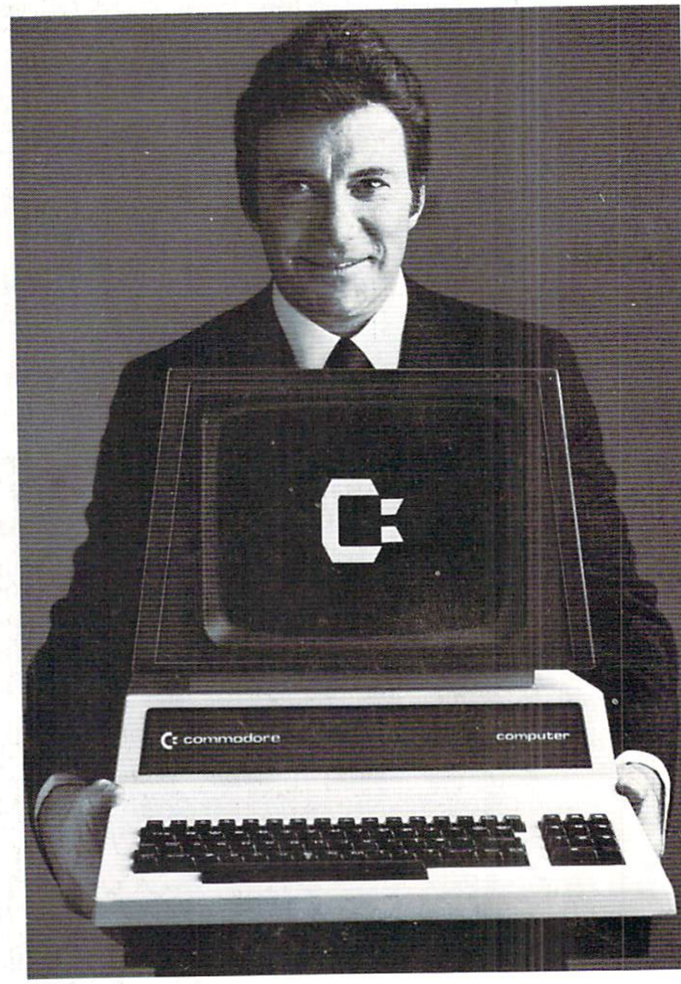
POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008

THE COMMODORE COMPUTERS

"FROM $300 TO $1995, THEY COST LESS AND GIVE YOU MORE FOR YOUR MONEY. READ OUR CHART."

—William Shatner



The idea of a computer in every office and home used to be science fiction. Now it's becoming a reality. The question is, with so many to choose from, which computer should you buy? When you consider the facts, the clear choice is Commodore.

COMPARE OUR $995 COMPUTER

FEATURES	COMMODORE 4016	APPLE II	IBM
Base Price	$995	$1,330	$1,565
12" Green Screen	Standard	299	345
IEEE Interface	Standard	300	NO
TOTAL	$995	$1,929	$1,910
Upper & Lower Case Letters	Standard	NO	Standard
Separate Numeric Key Pad	Standard	NO	Standard
Intelligent Peripherals	Standard	NO	NO
Real Time Clock	Standard	NO	NO
Maximum 5½" Disk Capacity per Drive	500K	143K	160K

Prices are as of the most recent published price lists, September, 1981 and approximate the capabilities of the (16K) PET® 4016. Disk Drives and Printers are not included in prices. Models shown vary in their degree of expandability.

Many experts rate Commodore Computers as the best desk-top computers in their class. They provide more storage power—up to 1,000,000 characters on 5¼" dual disks—than any systems in their price range. Most come with a built-in green display screen. With comparable systems, the screen is an added expense. Our systems are more affordable. One reason: we make our own microprocessors. Many competitors use ours. And the compatibility of peripherals and basic programs lets you easily expand your system as your requirements grow. Which helps explain why Commodore is already the No. 1 desk-top computer in Europe with more than a quarter of a million computers sold worldwide.

WE WROTE THE BOOK ON SOFTWARE.
The Commodore Software Encyclopedia is a comprehensive directory of over 500 programs for business, education, recreation and personal use. Pick up a copy at your local Commodore dealer.

FULL SERVICE, FULL SUPPORT.
Commodore dealers throughout the country offer you prompt local service. In addition, our new national service contract with TRW provides nationwide support. Visit your Commodore dealer today for a hands-on demonstration.

Commodore Computer Systems
681 Moore Road
King of Prussia, PA 19406

Canadian Residents:
Commodore Computer Systems
3370 Pharmacy Avenue
Agincourt, Ontario, Canada, M1W 2K4

Please send me more information. CBM-CO

Name ____________________

Company ____________ Title ____________

Address ____________________

City ____________ State ________ Zip ________

Telephone ____________________

Interest Area ____________________

☐ Business ☐ Education ☐ Personal

commodore
COMPUTER