Computer-Aided Instruction: Boon Or Bust

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

Running 40 Column Programs On A CBM 8032

COMPUTE!

$2.50
May, 1981
Issue 12
Vol. 3, No. 5
63379

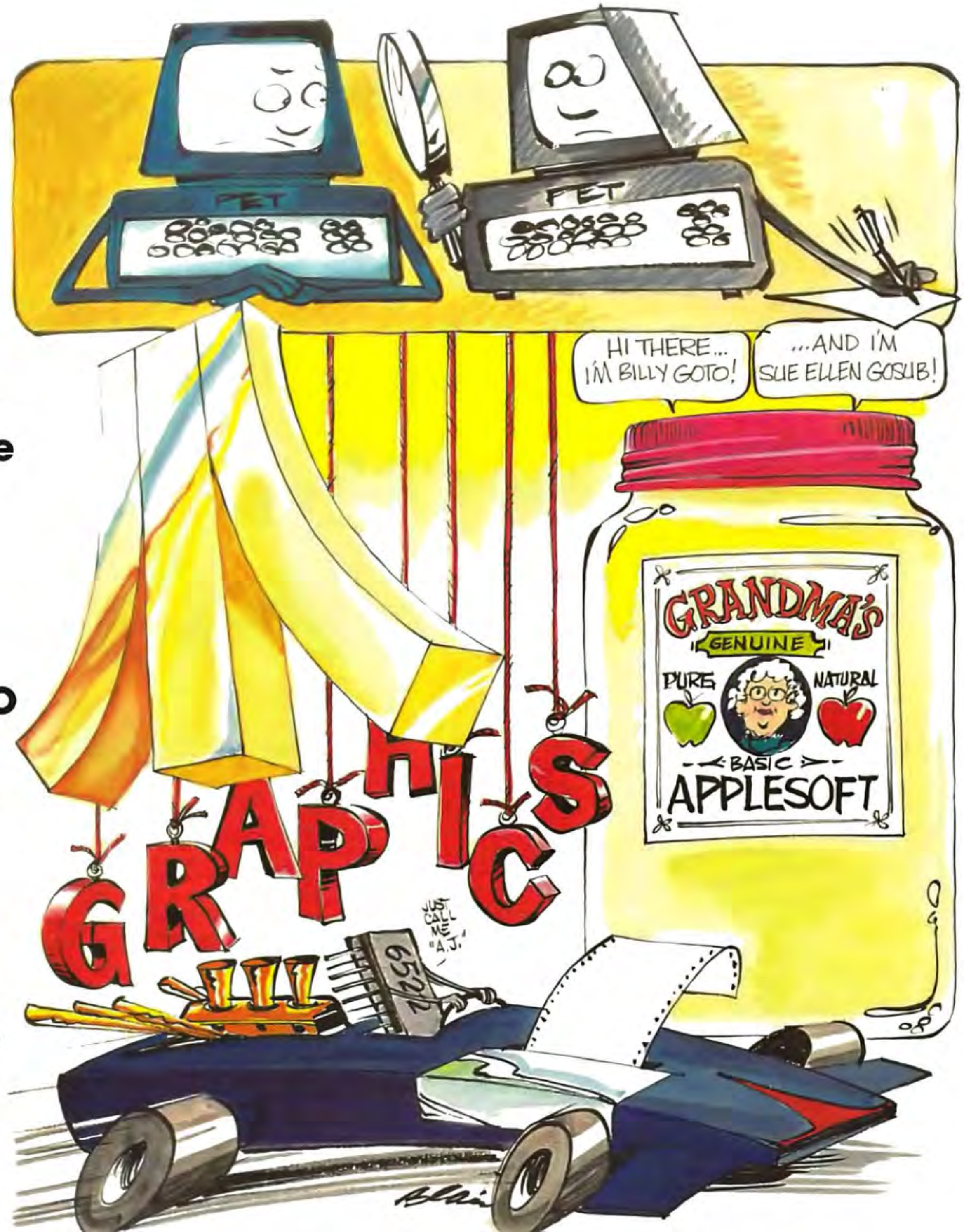
The Journal For Progressive Computing™

PET As An IEEE-488 Logic Analyzer

Using Strings For Graphics Storage On The Atari

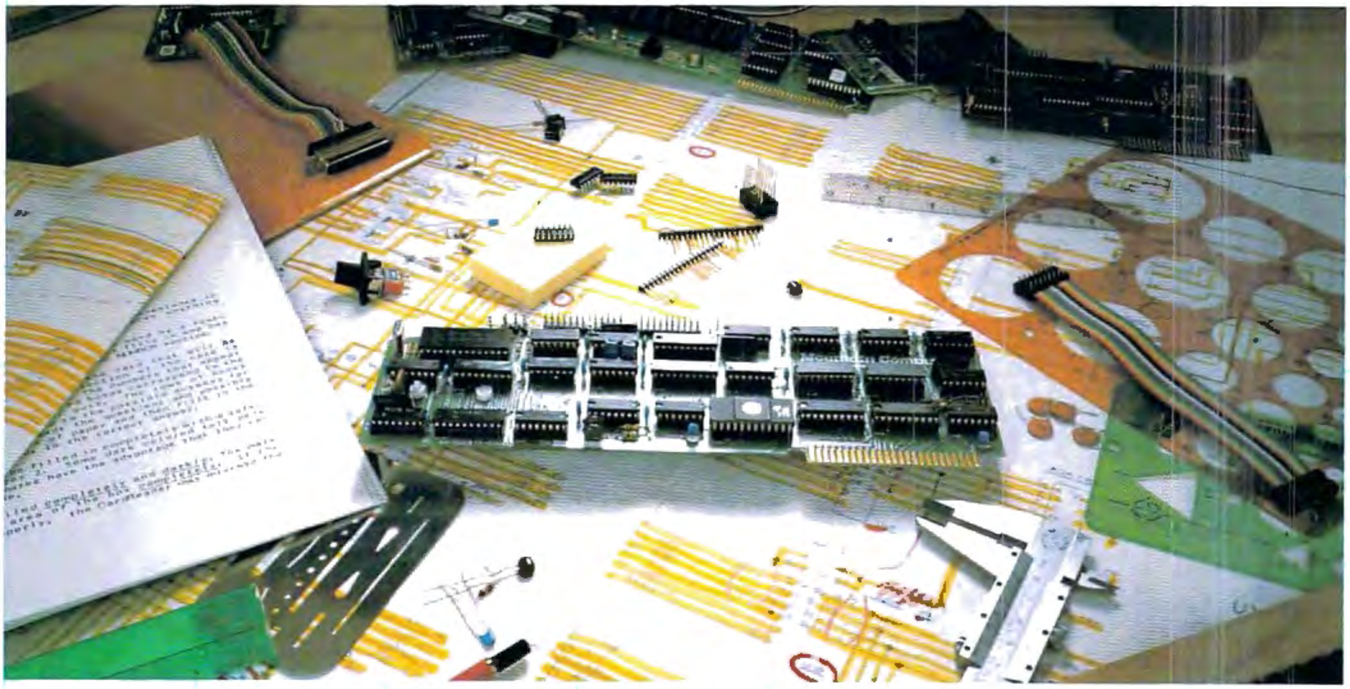
Using Named GOSUB And GOTO Statements In Applesoft BASIC

Using The 6522 To Drive A Printer



0 74470 63379 05

Mountain Computer put it all together for you.



The CPS MultiFunction Card

Three cards in one! The Mountain Computer CPS MultiFunction Card provides all the capabilities of a serial interface, parallel output interface and real-time clock/calendar—all on one card—occupying only one slot in your Apple II®. Serial and Parallel output may be used simultaneously from CPS. CPS is configured from a set-up program on diskette which sets the parameters (such as baud rate, etc.) for all functions contained on the card and is stored in CMOS RAM on the card. Once you have configured your card, you need never set it up again. You may also change parameters from the keyboard with control commands. All function set-ups stored on-board are battery powered for up to two years. "Phantom slot" capability permits assigning each of the functions of CPS to different slots in your Apple without the card actually being in those slots! For example, insert CPS in slot #4 and set it up so that is simulates a parallel interface in slot #1 and a clock in slot #7 and leave the serial port assigned to slot #4. CPS's on-board intelligence lets it function in a wide variety of configurations, thereby providing software compatibility with most existing programs. "We've put it all together for you"—for these reasons and many more! *Drop by your Apple dealer and see for yourself how our CPS MultiFunction Card can expand the capabilities of your Apple and save you a great deal of money as well!*

Calendar/Clock
- One second to 99 years
- Battery backed-up (2 years)
- Two AA standard alkaline batteries for back-up (provided)
- Compatible with MCI Apple Clock™ time access programs

Parallel Output
- Features auto-line feed, Apple tabbing, line length, delay after carriage return, lower to upper case conversion
- Centronics standard—reconfigurable to other standards
- Status bit handshaking

Serial Interface
- Features auto-line feed, transparent terminal mode, Apple tabbing, line length, delay after carriage return, local echo of output characters, simultaneous serial/parallel output, lower to upper case conversion, discarding of extraneous LFs from serial input
- Uses the powerful 2651 serial PCI chip
- 16 selectable internal baud rates—50 to 19.2Kbaud
- Half/Full duplex terminal operation
- I/O interface conforms to RS-232C
- Asynchronous/Synchronous operation

Mountain Computer
INCORPORATED

300 El Pueblo Scotts Valley, CA 95066
(408) 438-6650 TWX: 910 598-4504

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

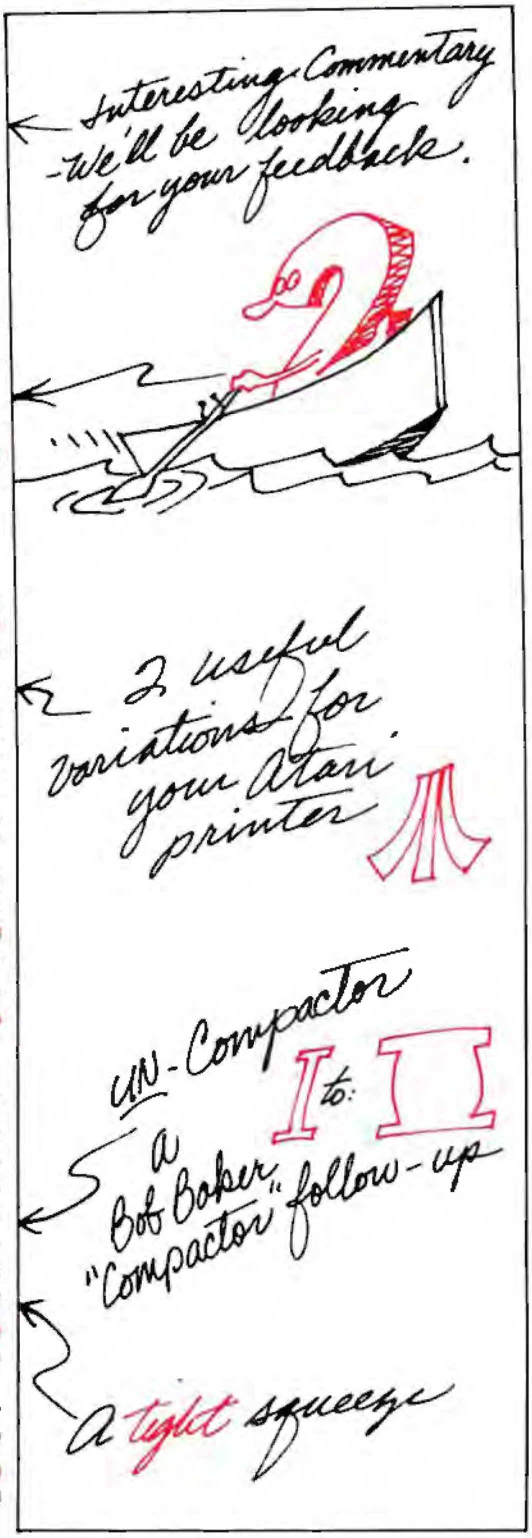
SoftCard is a trademark of Microsoft. Apple II and Apple II Plus are registered trademarks of Apple Computer. Z-80 is a registered trademark of Zilog, Inc. CP/M is a registered trademark of Digital Research, Inc.

MICROSOFT
CONSUMER PRODUCTS

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

Table of Contents

May, 1981, Vol. 3, No. 5



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

Domestic Subscriptions: 12 issues, $20.00. Send subscription orders or change of address (P.O. Form 3579) to Circulation Dept., **COMPUTE!** Magazine, P.O. Box 5406, Greensboro, NC 27403. Controlled circulation postage paid at Greensboro, NC 27403. Application to mail at controlled circulation rates pending at Hickory, NC 28601.
 ISSN 0194-357X.

A Small System Services, Inc. Publication

COMPUTE!

The Editor's notes...

Robert Lock,
Editor/Publisher

The West Coast Computer Faire was exceptional. A real joy. Do you realize how fast this industry of ours is growing? And I mean growing in terms of more people becoming interested in what we've been doing for the past few months or years, as well as growing in breadth. Here's a sample:

It Talks Back ... And Well

Votrax (500 Stephenson Highway, Troy, MI 48084, (313) 588-0341) showed off their "Type-'N-Talk™" a text to speech synthesizer that produces quite recognizable speech. You interface "Type-'N-Talk™" through an RS-232C interface, type English text with a talk command, and your computer talks back to you.

Now you should understand that this isn't a speech recognition device. It's a speech output device. It more than adequately constructs verbal strings of text from your keyed input in programs. It's just that you can't talk back to it. The company expects to have production quantities available in June. Suggested retail price is $345.00. Watch for a full review by Susan Semancik and our Delmarva Computer Club group in an upcoming *Micros With The Handicapped* column.

A second interesting product at the show was the Osborne 1, a (Z-80) based portable computer utilizing industry standard technology in a clever fashion. Designed as a portable, hand carriable unit, it meets its specs. Primary attractions, beyond that, are its price and some innovative software bundling. At a $1795 retail price, the Osborne 1 has these features:

- 64K, Z80A
- Standard Business Keyboard
- A 5" CRT with CLEAR resolution
- Serial and IEEE 488 interfaces
- Dual "100K" minifloppies
- Weatherproof carrying case

The interesting break is the software bundling — the $1795 price includes:

- Wordstar word processing with Mailmeye option
- The CP/M disk operating system
- CBASIC and MBASIC languages
- The Supercalc electronic calculator

Additional hardware options will be offered. I think if you're on the market for such a machine, this'll be a good place to start looking. As always, not the only place, but the concept of bundling of software is certainly attractive.

Introducing "Super-PET"

Commodore has made what appears to be a breakthrough of major significance for the industry. The machine's true name is unknown at press time. It has been variously called; the "Mini-Frame", the "Micro Mini-Frame", the "Mini Main-Frame", and the "Micro Main-Frame". *(We would have been happy to sponsor a "Name the Super-PET" contest.)*

We received much of this information in a March 3 interview, but held off because of on-going "delicate negotiations". These apparently over, "Super-PET" was introduced at the Hanover Faire in Germany during the first week of April.

How super is it? Here are the specifications:

- 134K Mixed RAM and ROM allocated as follows:
  - 18K ROM Operating System for the 6502 processor
  - 18K ROM Operating System for the 6809 processor
  - 2K Screen RAM
  - 32K "normal" CBM 8032 RAM
  - 64K Bank Switched RAM operating as virtual memory.

1 RS-232C fully programmable serial port
1 High-speed serial communications port for networking at 200KB

- Languages:
  - Waterloo Extended BASIC.

Some of the highlights of this BASIC include unlimited length strings, name called subroutines with parameter passing, local and global variables, program chaining, and total variable preservation. (Meaning you can correct some types of errors in a

Staff

Robert C. Lock, Publisher/Editor
Kathleen Martinek, Publication Assistant
Georgia Papadopoulos, Art Direction/Production Assistance
Joretta Klepfer, Manager, Dealer/Distributor Marketing
Carol Holmquist Lock, Circulation Manager
Dai Rees, Shipping Department
Charles Brannon, Software Lab Assistant
Terry Cash, Typesetting/Production
Harry Blair, Advertising/Promotion Manager

Associate Editors

Jim Butterfield, Toronto Canada
Harvey Herman, Greensboro, NC

Contributing Editors

Robert Baker, 15 Windsor Drive, Atco, New Jersey 08004
Gene Beals, 115 E. Stump Road, Montgomeryville, PA 18936
Len Lindsay, 5501 Groveland Terrace, Madison, WI 53716
Craig Patchett, 2 Swan Terrace, Greenwich, CT 06830

Subscription Information

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

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

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

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

Advertising Sales

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

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

Northern California, Pacific Northwest, Rocky Mountain States, Texas, Oklahoma
Jules E. Thompson, Inc.
1290 Howard Avenue, #303
Burlingame, CA 94010
In Texas or Oklahoma call our Houston number: 713 731-2605
Elsewhere: 408 354-5553 or 415-348-8222
Phoebe Thompson

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

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

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

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

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

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

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

Telephone: (919) 275-9809

running program, type continue and resume the program.)

- PASCAL The Jensen-Worth Standard implementation
- FORTRAN Waterloo Standard version
- APL
- COBOL (later in the spring)
- An Assembler that's supposed to be quite powerful

This entire package plugs into the standard CBM 8032. You plug it in and go with a switch on the side to select your processor mode.

The "delicate negotiations" were necessitated by the fact that all of this expansion power was developed outside of Commodore. Bill McLean and crew at BMB Compuscience in Canada were responsible for developing the hardware, and Waterloo University in Canada, developed the software. Commodore will be marketing the product worldwide. My thanks to Dr. Frank Winter at Sheraton College for his help in putting this all together.

The unit will be introduced in the US at the NCC beginning May 4. Given the configuration of hardware and software, it certainly looks as if we're looking at a potentially viable entry into the small business market of the Apple III and others. We have no confirmation of the upgrade price, but the reliable rumors suggest the expansion will cost much less than the current retail 8032 price of $1795.00.

Well Dr. Chip, it looks like **COMPUTE!** will be covering the 6809 before too long.

News From The Atari Front

Atari has announced a major software development and support project. See the new products section for more information. Axlon has announced a 256K memory system for the Atari 800. The unit provides eight expansion memory slots, allows bank selection of memory, and comes with memory management software. For more information, they're at 170 Wolfe Road, Sunnyvale, CA 94086. (408) 730-0216.

At the West Coast Faire, Atari interest was quite strong. Macrotronics, showing off their screen printer package. (Atari to Trendcom 200 or Paper Tiger) was quite busy. Atari corporate, though not exhibiting, had a private preview for user group officers. Among other things they showed off the new word processor and I heard excellent reports on it.

That's A Switch, PET

Data Equipment Supply was demonstrating a new ROM switching device at the show, and at least two companies (one, Canadian and one, English) have now announced versions of "soft" ROM — PET or CBM RAM expansion boards or chips that can retain information. In a future issue, we'll have some enlightenment on the situation, furnished by Jim Butterfield. ©

The Readers' Feedback

Robert Lock and Readers

It's nice to be back. First of all, we're hoping to have *Ask The Readers* up and running by next month. That's our new three-way column that serves as an interface between programmers with problems and readers with solutions. *The Beginner's Page* returns next month.

On this positive note, let's get started:

"Thanks for:

1. Putting the magazine into envelopes again.

2. Ask the Readers. I will answer.

3. A magazine that gets better each month."

Thanks for the boost. For you cynics saying it may be better but it also gets later, we know. We've expanded our production staff, and brought all typesetting and camera work in-house. Frankly, we've been growing so fast we had to do a little catch-up. This is our 12th issue, and we've almost tripled in circulation in the last 6 or 7 months of our 20 month history!

"Keep up the good work ... need some good small business programs for the Atari (Payroll, taxes, investments, etc.)."

"Article on stock market, financial news software. Is there software available that allows user to create own daily bar charts from Dow Jones News Service quotations?"

Okay, you reader/authors. Anyone willing to share their business investment programs in articles. We're all interested.

"The main reason I buy **COMPUTE!** *is the strength of its articles on ATARI. Better distributorship in the Orange County, CA area is needed."*

We're certainly interested in developing better distribution. If your local dealer sells out of **COMPUTE!** in two days (which many do), suggest they order more. If your local dealer doesn't yet carry **COMPUTE!**, ask them to give us a call.

Until next time ... keep those cards and letters coming.

Thanks,
Robert Lock

©

Computers And Society

David D. Thornburg
Los Altos, CA

Several years ago, when Betty Burr and I were conducting workshops for "computer-phobic" adults, we thought that someone should write a "computer demystification" book which would sell to general audiences. Since that time we have seen several such books come to market (some of which have been reviewed in this column).

I recently received another book on this topic which is certain to sell quite widely, both because it is handled by a well known publisher (Simon and Schuster) and because its principal author is the famous science fiction writer Frank Herbert. The book was written with the help of Max Barnard, the person who worked with Herbert in setting up his computer system.

The book's title, "Without Me, You're Nothing, is taken from the author's advice that when you first set up your computer you should stand in front of it and say:

> *"You stupid, inanimate chunk of hardware! Without me, you're nothing!"*

As you can see, this book is a bit theatrical. This sense of theatrics, more than anything else, becomes the basis of one of this book's greatest shortfalls. I share some of Frank Herbert's goals, *e.g.*, the demystification of computer technology for the general public; but my fear is that he has replaced one myth with another one.

Betty and I found that many adults feel that you have to be a technical wizard to use computers effectively. We feel that this is a most damaging myth since it serves to disenfranchise a large number of people who might otherwise find utility in this technology. Our position (as regular readers of this column might remember) is that computers are like automobiles in the following way. You do not *have* to know how to drive a car to survive in our society, but you do need to know enough about them to not walk out in the street in front of one. I think that "computer literacy" is important for much the same reason. Computers are becoming so commonplace that each of us should have enough awareness of their capabilities to decide for ourselves whether or not to gain access to this technology.

Frank Herbert has a different goal in mind. He places the potential computer user in an "us" vs. "them" context. For example:

> *Things are happening in our world that make a necessity of the skills we are about to share with you. Before long it will at least be a matter of self-defense for you to have your own computer and be able to use it. You are already being taken advantage of by people with computers. You will not be able to meet that challenge or keep up with other changes unless you acquire a computer yourself.*
>
> *... Please take our warning to heart. Very soon, if you don't have access to a computer, you're going to be racing in something equivalent to the Indianapolis 500—only you'll be on foot.*

...demystification of computer technology for the general public...

Hmmm. My fear is that Mr. Herbert's zeal will result in the replacement of one type of misconception with another one.

Fortunately there are delightful streams of insight in this book which tend to counter the mild spasms of hysteria sampled above. One of the most important points that Herbert makes is that the computer is a *tool*, not a "thinking machine". The computer can amplify creative imagination, but not be creative itself. As he says:

> *A pen is a tool. A typewriter is a more sophisticated pen. A library is a tool. A painter's easel is a tool. It is the creative mind behind the tool that is important.*

Later on he says:

> *Computers may be superb for logic and accuracy within described and describable limits, but don't ever depend on one for creative work. The machine will not go outside its limits. It has no imagination. In fact, people of limited imagination, people who don't understand what you mean by "creative brainstorming", tend to lead the argument for the "electronic brain" myth. They impose limits on themselves and they want to apply similar limits to the universe because that makes them feel safer.*

So much for philosophy. The book also promises to be "a practical, easy to understand guide to *using* your own personal computer system". The technical side of this book needs tremendous reworking. I am astounded that a publisher as large as Simon and Schuster would publish a book with so many basic errors in it. For example, I have *never* heard of a disk drive being refered to as a "disk driver", but that is what Herbert calls it throughout the entire book. In his quest to show

CRYSTALWARE

12215 MURPHY AVENUE,
SAN MARTIN, CALIFORNIA 95046

THE FINEST IN FANTASY GAME SOFTWARE

APPLE	ATARI	PET	TRS-80
48K — w/disk	48K — w/disk	32K — w/disk	48 — w/disk

SPACE GAMES — Climb aboard the Starship Herman

GALACTIC QUEST — Warp through the 64 solar systems in Hires graphics with Crystalsonics sound. Fire lasers and rifter pods as you battle the Vegans in real time simulation. In Galatic Quest we have attempted to combine the best elements of a Star Trek type game with one similar to Space Trader. What is Skishi and where is the Pleasure Planet? In some systems, lurking behind swarms of asteroids, are Space Pirates who will gut your ship and leave you a derelict in space. To win Galactic Quest you must be shrewd a business man as well as a top notch warrior. $29.95

LASAR WARS — Attacking aliens in 3D. Thousands of approach simulations and 10 levels of difficulty. Three kinds of invader craft scored by size and attack mode. An exciting Invaders type game with perspective and sound. For fantasy enthusiasts from 8 to 80. $29.95

SANDS OF MARS — You embark on the Maiden Voyage of the Starship Herman. This game takes up almost 200K and uses 2 disks. It includes more than 300 Hires screens animated scenarios, a 3 dimensional maze, and a Martian labyrinth drawn entirely with hexagons. The takeoff and landing sequences are paddle or joystick controlled for the Apple and Atari and the game has 5 full scenarios. Once you land on Mars, you wander through fields of Sasquati and Degwat. Beware of the Vishu and the Lizardmen of Meshim. Seek the wise Mudra and unlock secrets hidden for centuries. Glyphs written on the Martian Sand hold the key to the mystery for which we offer $100 to the first to solve it. To answer a question we have been asked many times — yes, Sands of Mars attempts to fully utilize the sound and graphics capabilities of the Apple and Atari to their max! $39.95

ADVENTURES — Cross the threshold to new worlds.

HOUSE OF USHER — Walk the dreaded corridors of the deadly House of Usher. Complete with 40 rooms and hundreds of aggressive monsters and unique treasures. This is not your typical text adventure game but goes far beyond that, with animated monsters and a visual display of each room. We believe this to be the first indoor-outdoor game ever written for a microcomputer game which includes graphics. You may choose to wander through Usher's scenic garden paths or brave the perils of the graveyard and descend into the crypt. Beneath the house there are labyrinths, shrinking rooms, and torture chambers with no doors or windows. Your character may pick up, drop, or use objects, fire arrows, or run frantically for the door when pursued by some loathsome creature. As the old grandfather clock ticks away, you will have until dawn to solve the Usher Mystery and win a real live $100 prize! $24.95

IMPERIAL WALKER (ATARI 16K) — For you Atari people out there, here's a fantastic game pack of 4 games with revolutionary new graphics routine and Crystalsonics. As Luke Skywalker you have taken control of an All Terrain Armored Transport and with your lasers are fending off the Imperial attack forces. Written especially for Atari by Mike Potter. The game pack also includes Laser Nim, Auto Race, and Gunfight. Will run on Atari 400 or 800 with disc and Joysticks. $29.95

ADVENTURELAND 2041 (Coming June 1) — Step with us through the Crystal door into the world of tomorrow's fantasy. In the great Outback of Australia in the year of 2041, will be built the greatest Adventureland of all time. Through your computer, you can experience six separate worlds of fantasy. These programs will take up more than 500,000 bytes of memory and fill 6 disks. Many of the options may be played by up to six players and in many cases against the computer itself. $59.95

WAR GAMES

WORLD WAR III — This is a three scenario war game in Hires graphics with sound. It is not merely the conversion of a board game to computer, nor are your pieces represented by lifeless text charcters. It may be played by two persons and takes about 8 hours to complete. The rules are simple enough that you won't have to spend several days reading your manual before you can play. It contains 2 world maps and a fairly detailed map of the Iran-Iraq battle field. All scoring, animation, and positions are handled by the computer — no separate tablets to fool with. Moves are input by both players in series of 3 and when the space bar is pressed the battle becomes animated. A must see to believe $29.95

WATERLOO (Coming July 1) — A war game with graphics very similar to World War III. We have attempted to make this as detailed as possible, down to what each individual is wearing, his line of sight, and the number of bullets he has fired. It will occupy two disks and may be saved over a period of weeks. We will be publishing more information on this in BYTE MAGAZINE in July. $49.95

Order line open 24 hrs / day (408) 683-0696

VISA — We accept VISA & Mastercharge — master charge how "simple" computers are, he says that a light switch is the simplest computer. This kind of misconception serves no one well. For one thing, computers *are* complex (just as automobiles are complex). The beauty of computers is that you don't have to understand how they work to use them. So why, for this audience, should an author fill the book with inaccurate simplifications which might make the reader feel like a fool when sharing this new found knowledge with more technical comrades?

The authors are strong proponents of top-down programming, and have developed a new flow chart system (called PROGRAMAP) for laying out programs. I found this concept to be poorly presented, but, like much else in this book, created with good intentions. As for languages, BASIC is king for Herbert. It isn't clear how well he grasps the language himself, though, as you can see from his definition for the BASIC keyword RETURN:

> *RETURN transfers the program back to the statement after GOSUB. It is the last statement of a subroutine. (Not to be confused with directions referring to the RETURN key on your keyboard. The RETURN we refer to here is a word in BASIC that performs in the computer in a way similar to that key. With this word, you build the key's function into the program.)*

COME *ON* FRANK! The RETURN key is built into a program by PRINTing CHR$(13). It is a line terminator, period. The keyword RETURN is completely unrelated to this function.

The author of a book with the circulation this one will have should be getting much better technical advice, and his agent and publisher must share the blame for mistakes of this sort. Now, if only Erma Bombeck would write the sequel. . .

A reader writes . . .

I received a letter a few weeks ago from **COMPUTE!** reader Bob Forman who is concerned that I might be paying too much attention to the futurists. Commenting on the January '81 Computers and Society column on communications, he says:

> *I'm a believer in the computer and its place in the family, in business and in many more places that it keeps falling into. BUT IT WILL NEVER REPLACE THE NEWSPAPER and the 10 o'clock news!*

As someone who works closely with the newspaper industry, Bob shared his experiences with the use of microfilm as an alternative to bound volumes of newspapers. He found that – whatever its efficiencies might be – the poor human factors aspect of microfilm prevented it from replacing bound files (as many thought it would). He says that the reasons for this are simple:

> *Why? Bound files are simple, easier to use. Try getting someone 70 years old to sit in front of a microfilm reader or a computer long enugh to read a whole newspaper. You can't sit back in your old lounge chair and read a film reader without some pretty expensive stands or cranes to manipulate the thing, so it's not a practical thing for every evening. The young bucks can stand to read a screen for a while but it's a more tiring process than reading a paper ... And, I haven't seen a high speed printer yet which will show a picture of a cabbage head accurately, or anything that approaches a good photograph.*

I think that reader Bob makes a good point – but only if one talks about one media format replacing another one. The telephone has not yet eliminated the mail and telegraph. The television has not yet eliminated the radio. I do not believe that any rational person thinks that the printed word will disappear when terminals appear in everyone's home. What I *do* believe is that a very large segment of the general population will start fitting the computer information utility into their mix of information sources, and that it will result in the kind of re-equilibration period we had when television started to compete with radio.

The most important advantage of computer based information utilities is their ability to access many diverse data bases, rather than forcing the user to listen to one person's view of the news.

As always, it is great to hear from readers. I look forward to your letters and messages (I can still be reached on the Source at TCE132). Till next month. . . . ©

You may write to David at Innovision,
P.O. Box 1317,
Los Altos, CA 94022.

10 DAY FREE RETURN

APPLE II
We carry the most complete inventory of Apple computers, peripherals, and software CALL!
Our Best Selling Apple System: Save over $250 on our most popular Apple System. System includes a 48K Apple II, Apple Disk, DOS3.3, & Controller, and a Sup R Mod RF Modulator
List $2209.00
COMPUMART Sale Price: $1895.00

SOFTWARE FROM APPLE
Apple Plot The perfect graphic complement for Visicalc $70.00
Dow Jones News & Quotes $95.00
Adventure (Uses 48K) $35.00
DOS Tool Kit $75.00
Apple Fortran $200.00
Tax Planner $120.00

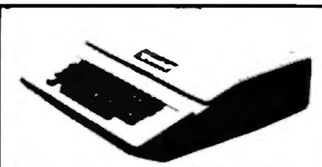
FROM PERSONAL software
Visicalc $149.00
Desk Top Plan $99.00

NEW FROM MUSE
The Voice $39.95
Super Text $150.00
Address Book $49.95

Miscellaneous Apple II Accessories.
Easy Writer (80 col. need a Videx) $249.00

NEW from Apple for the Apple II.
DOS 3.3 Convert disks to 16 sector format for 23% more storage and faster access $60.00

NEW for the Apple II
From MicroSoft
16K RAM Board $195.00
FORTRAN $175.00
COBOL $750.00
Card Reader from Mountain Hardware $1,195.00
Hi-Res Dump for 460 Printer from the Computer Station $49.95
Trend Spotter—Statistical graphics program. Works with Visicalc $275.00
Exclusive—Magic Wand for the Apple—Required Videx, Z-80 soft card **complete system** $925.00 (Requires 48K Apple and disk.)



HARDWARE ACCESSORIES FOR APPLE
Silentype Printer w/x face $635
Super Sound Generator (mono) $159 (stereo) $259
Light Pen $249
X-IO Controller (plugs into paddle port) $49

Mountain Computer—Expansion accessories for your Apple
Introl—IO System $289
Super Talker $299
The Music System $545
ROM plus board w/Keyboard filter $199
Clock Calendar $280
Apple Expansion Chassis $650
ROM Writer $175
16 Channel A to D Converter $350

From VIDEX!—Video Term.
80 Col. x 24 line, 7 x 9 Matrix plug in compatible board for the Apple II. Price **$325** without graphics EPROM. With graphics EPROM $350.
S.S.M. Serial & Parallel Apple Interface $225
ABT's Numeric Key Pad $110
California Microcomputer Keyboard $195

Save $200 on Apple Accessories with purchase of an Apple II. Order accessories for your Apple, and we will give you $200 off the purchase price of those accessories. **SAVE $200**

APPLE III IS IN STOCK—CALL!
APPLE III Information Analyst Package—128K Apple III, Black and White Monitor 12", and information analyst software **Price $4740.**

CompuMart carries the entire Commodore line of computers, peripherals, and accessories—call our sales force today for complete prices and specs.

The CompuMart/Commodore Word Processing System
Get crisp, letter quality output and ease of operation that's unsurpassed. This system includes a Commodore 8032 32K CPU, a 4040 Dual Disk, a C-Itoh printer and X/face, and Word Pro 4 Plus (all cables included). Call for details and low prices. List $5685. **COMPUMART $4995.**

A complete system including a Dual Disk Drive, Tractor Printer, and an 80 column 32K CPU for under $4,000. No interfaces needed. Cables included. List $3,985. **CompuMart $3635**

New accessories from Commodore for Commodore
Visicalc $199
Word Pro 4 $299
Ozz the Information Wizard $395
Wordcraft 80 $395
Tax Preparation Planner $495
Dow Jones Portfolio Mgmt System $149
Personal Tax Calculator $69
Assembler Development Package $99

SAVE $200 on Commodore Accessories with purchase of a 32K PET. When you buy a 16K PET, apply $100 toward PET accessories. **SAVE $200!**

Educators Take Note!! Commodore has extended its 3 for 2 deal until 6-30-81

PM65 Expansion Accessories for the Aim-65
Call for specs and prices.
Forth for Rockwell AIM-65 $195.00

ROCKWELL AIM 65
Our AIM system includes 4K AIM with BASIC interpretor assembler, Power Supply, Cassette recorder & Enclosure $799.00
4K AIM-65 $499.00
PL65 High Level Language $125.00
Paper for the Aim (roll) $2.50
Rockwell's 4-slot Motherboard (**sale**) $175.00
Fourth for Rockwell AIM-65 $175.00
850 Interface w/Cable $249.95

NEW SUPER SPECIALS—

HP83 Basic System—Includes HP-83, ROM Drawer, Mass Storage ROM, 5¼" Single Master Flexible Disc Drive, HP-IB Interface Module, 2 meter HP-IB cable. **$4249.**

HP-85 System—Save $400 on HP-85 Accessories with purchase of an HP-85. Order accessories with your HP-85 and we will save you $400 off the purchase price of those accessories. **SAVE $400!**

HP-85 ACCESSORIES—We carry HP Peripherals (Disk Drives to Graphics Plotters) Enhancements (BASIC Training, General Statistics, Financial Decision, Math, Linear Programming **$95 ea.**), HP-85 Accessories (Enhancement ROMs, ROM drawer, Overhead Transparency Kit); Supplies (Plotter Pens, Tape Cartridges); Interface Modules (HP-IB Interface, HP-IB Interconnect Cables, Serial (RS-232C) Interface Module). We can get your every HP peripheral made for the HP-85.

VISICALC PLUS FOR HP-85 Everything you ever wanted from Visicalc plus plotting **$200.**

Super Selling Terminals
We have the following **Lear Siegler** Terminals in stock at prices too low to print—Call for quotes.
ADM—3A Industries favorite dumb terminal for some very smart reasons.
ADM—3A +
NEW from Lear Siegler. CALL!
IT IS HERE!—It is the **new** Intermediate Terminal from Lear Siegler. Call for details.

HP-41C Calculators $199.00
Memory Modules. For storing programs of up to 2000 lines of program memory $29.95

"Extra Smart" Card Reader Records programs and data back onto blank magcards. $199.00

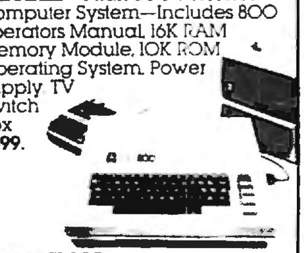
The Printer. Upper and Lower case. High resolution plotting. Portable Thermal operation $355.00
Application Modules $29.95

Visit our giant
ANN ARBOR STORE
1250 North Main Street
Ann Arbor, Michigan

VISIT US AT THE NCC '81 BOOTH 1623

SEND FOR OUR FREE CATALOG

COMPUMART exclusive ATARI SPECIALS—Atari 800 Personal Computer System—Includes 800 operators Manual, 16K RAM Memory Module, 10K ROM Operating System, Power Supply, TV Switch Box **$899.**

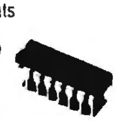


PERIPHERALS
Atari 410 Program Recorder **$89.95**
Atari 810 Disk Drive ($100 off with purchase) $699.95
NEW Dual Disk double density $1499.95
825 Printer (Centronics 737) $995.00
RS232 Interface w/Cable $249.95
NEW! Light Pens $74.95
NEW! Visicalc for Atari $199.00

Educators Take Note. Atari has extended its 3 for 2 deal until 6-30-81. Any educational institution that buys two Atari 800's will receive an Atari 400 computer absolutely **FREE.** Call our sales dept. for complete details.

Dysan Diskettes—Single side, Single density. Hard or Soft Sector **$5. ea.**
Memorex 3401's—5¼ discs **$3.25** with hub ring for Apple **$3.50.**

Memory Integrated Circuits—
Call for qty. discounts when ordering over 50 units
Motorola 4116 (200 Nanosecond, Plastic) **$4.50 ea.**
Fairchild 2114 (Standard Power, Plastic) **$4.50 ea.**



NEW SUPER 41-CV Systems with Quad RAMS built-in. (Maximum memory on-board leaves slots open for Application Pacs and peripherals.) The HP 41CV **$309.00**
HP 41CV Plus Card Reader **$474.00**
HP 41CV Plus Card Reader & Printer **$799.00**
Quad RAMS for the 41-C (Equivalent to four Memory Modules all packed into one.) **$95.00**

COMPUMART stocks the complete line of **MATROX PRODUCTS.** Call for specs.

COMPUMART now offers the ENTIRE **DEC LSI-11** PRODUCT LINE. Call for prices & delivery.

NOVATION CAT™ ACOUSTIC MODEM
Answer Originate 300 Baud, Bell 103 **$169.00**
NEW! D-CAT
Direct Connect Modem from Novation. **$199.**

IMPORTANT ORDERING INFORMATION All orders must include 4% **shipping and handling.** Mass. residents add 5% sales tax. Michigan residents 4% for sales tax. **Phones** open from 8:30 a.m. to 7:00 p.m., Mon.-Fri. 11:00 a.m. to 4:00 p.m. Sat. **P.O.'s** accepted from Dun & Bradstreet rated companies—shipment contingent upon receipt of signed purchase order. **Sale prices** valid for month of magazine date only—all prices subject to change without notice. Our Ann Arbor retail store is open 11:00 a.m. to 7:00 p.m. Tues-Fri. 10:00 a.m. to 5:00 p.m. Saturdays.

Editor's Note: From time to time we present what we choose to call "Guest Commentaries". These articles don't necessarily express the opinion of **COMPUTE!**, *but generally do raise questions we think should be discussed. . . RCL*

Computer Aided Instruction, Boon or Bust?

Alfred D'Attore
Phoenix, AZ 85021

Computer Aided Instruction, (CAI,) has been around for quite a while. Originally introduced into out public schools when "Time Share" became commonplace — about ten years ago — it has met with rather indifferent success. At its best, it appeared to offer no particular advantage over traditional teaching methods. At its worst — and that could be very bad indeed, with the frequent equipment "crashes" and student blunders — it was frustrating and ineffective. It was *always* crushingly expensive. School boards had horrible visions of endless banks of computer terminals with attending telephone connections, computer time costs, repair contracts — an endless cash flow.

The personal computer boom of recent years has eased expenses somewhat, but CAI is still not employed to any great extent in our public schools. Even when computer systems are purchased, they are rarely used for CAI. Rather, they are used to support a relatively minimal study of computer programming and the endless, ever-present games. Sometimes, they are not used at all. I know of one school in North Phoenix which recently purchased a disk-operated computer system complete with printer. Although access is provided, it lies virtually unused in an office, gathering dust.

The reason, of course, is the lack of suitable, appropriate software. Too few people are programming for our public schools. And when, occasionally, we do obtain CAI programs, they are most often tutorial in nature and therefore inappropriate for use in primary and secondary schools. Let me elaborate upon this point.

Any public school teacher can tell you that the normal learning process involves a very small amount of "teaching" and an immense amount of "doing." This is especially true when the subject areas are basic; for example: reading and arithmetic. In this circumstance, even the most skillful CAI, if it is basically tutorial, is a waste of time and good programming talent. It is simply too much work for too little return.

And this assumes the programming is successful. Often it is not. Often, the programming places too much burden upon the student with respect to display interpretation and console operation. Many programs have "bugs." Since with this type of programming, the student interfaces directly with the computer, the frustration level often runs very high.

But the most important reason for the general ineffectiveness of this type of programming in our public schools, lies in the very nature of our young students. The classroom teacher quickly learns that young people must establish an acceptable *personal* relationship with their instructor before meaningful learning can take place. An indifferent machine is at a big disadvantage there.

...let's allow the teacher to teach...

Certainly, if tutorial programs are prepared cleverly, students will be enthralled, initially. But that never lasts very long. In my classes, three weeks is about par, after which the system becomes just another classroom static fixture, like the countless desk calculators, visual aids and programmed instruction packages that remain largely unused in every classroom. Yet, if software is available at all for the first twelve grades, it is most often of this type.

Of even less use are the ancillary programs: the "curriculum guides to CAI," the "systems approaches-cum-administrative programming" packages and the various conceptual outlines. Teachers get "overviews" by the bucketfull. We treat them with the respect due most things that come in buckets. We need specifics, not generalities. I will be specific.

Let's allow the teacher to teach. Then we may use the computer to help him with his job.

The computer should be programmed to do that which it is uniquely qualified to do: create exercises. As I pointed out previously, individualized student work — exercises — represents the greater portion of the learning process. A computer, working in this fashion, will be helping the teacher do the greater part of his job. In skill-oriented subjects like arithmetic, for example, students are required to do exercises repeatedly, with graduated levels of difficulty. Students are drilled.

There, I've said it. That dirty word: drill. It has become anathema in recent years. It is supposed to turn students off. But realistically, there isn't anyother way to learn basic skills, especially basic

arithmetic skills. One must perform in a skill to make it one's own. And I don't mean just once. Indeed, I sometimes think, to the extent the learning process is difficult, to that same extent is the learning worthwhile.

But far from turning students off, my experience has taught me that young people become eager — even enthralled — when they begin to acquire measurable skills. And drill does it. Disciplined, repeated, old-fashioned drill. For drill, the computer is without parallel.

In my approach, a printer is required. Exercises must be printed out at all times, if the computer is to be used effectively. Exercises must be produced immediately, in unlimited numbers, tailored specifically to meet the particular need, and optimized for clarity, organization and student use.

Answers must be provided for all exercises. Where appropriate, they should be reduced to lowest terms. There should be no ambiguities. When dividing with decimals, for example, accuracy requirements should be ordered and neat. Since students work directly upon these exercise sheets, this will coerce them, gently, to be equally neat. This is most important, especially for students in remediation. Very often, their work is much too sloppy, and like other students, they tend to relate their teacher's requirement for neatness to "nit-picking," rather than to recognition of the fact that ours is a place-value number system. A digit's position in a numeral is quite as important as its value. Sloppiness confuses "place."

Spaces should be provided between digits in all those exercises where "carries" and like manipulations are required. Students should not be forced to crowd their work. Alternately, they cannot be permitted so much room as to encourage carelessness. "Neatness begets neatness. Order begets order." I don't know who said that first. Perhaps it's a paraphrase. But it is a dictum that should be kept foremost in mind when preparing computer aided instruction of this type.

To illustrate, a portion of an exercise sheet for integer addition is shown in figure 1. In this particular program, an ordered pair of numbers specifies the number of addends and the number of digits per addend. Note the "spacing" of digits. The number of problems and their spacing are set under program control. They vary automatically with the difficulty level of the problems.

```
LESSON NO. 1   Name ______________  Class ________
(01)   5 4 6 1    (02)   6 7 4 7    (03)   7 5 8 2
       5 4 6 5           2 2 7 2           7 7 6 7
       9 5 0 6           9 8 6 0           3 5 7 1
      ________          ________          ________
```

Figure 1

Of course, for basic skills instruction, programs running the gamut of arithmetic skills are required. I have used just such programming for the past five years. Permit me now to enumerate the advantages that have come to light in this period:

Programs are immediately adaptable to student competency levels. Through simple question and answer, an instructor may choose from a number of levels of difficulty.

Parents and family may enter into the training process. Since exercises are produced in moments and answers are provided in the appropriate formats, students may take any number of them home and be drilled by other family members.

...the computer becomes a valuable teacher's aid...it is not a surrogate teacher...

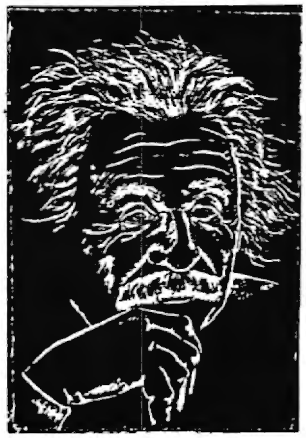
Individualized instruction — always desirable in the classroom situation — becomes less open-ended. The student runs little risk of drilling himself in incorrect procedures. With individualized instruction — for reasons of practicability — a student is often required to work for extended periods without direct supervision. With the answers before him, however, he cannot fail to be alerted to incorrect procedures.

The computer becomes a valuable teacher's aid. It is swift, versatile, flexible, indefatigable and inexhaustible. But it is an aid: no more. It is not a surrogate teacher. This approach is, therefore, non-threatening. Since computer aided instruction and its associated equipment must be sold — essentially — to teachers, this is a not-inconsiderable advantage.

Last, this approach is cost-effective. A computer system, used in this manner, is easily affordable. A 2,000 dollar system can serve a school. Such a system currently serves the school wherein I'm employed. Admittedly, this is a bare-bones approach, and I don't suggest for a moment that other schools should spend so little. In today's market, 5,000 dollars would purchase a disk-operated system with sufficient equipment backup to insure reliable operation for an indefinite period. *That* is the proper way to go.

In this article, I have dealt primarily with the mathematics in describing this "alternate approach" to CAI. But I have gone far enough afield in my programming efforts to have determined these methods are applicable in other teaching disciplines. With right programming, computers can be a boon indeed for our public schools. Without it, they are just expensive toys. So what shall it be? Boon or bust?

An Intelligent Alternative



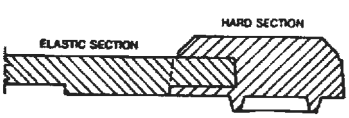
TYPRINTER 221

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

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

THE DAISY WHEEL

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



THE KEYBOARD

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

THE DISPLAY

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

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

PRINT MODE

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

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

MULTILINGUAL CAPABILITY

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

THE FEATURES

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

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

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

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

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

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

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

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

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

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

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

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

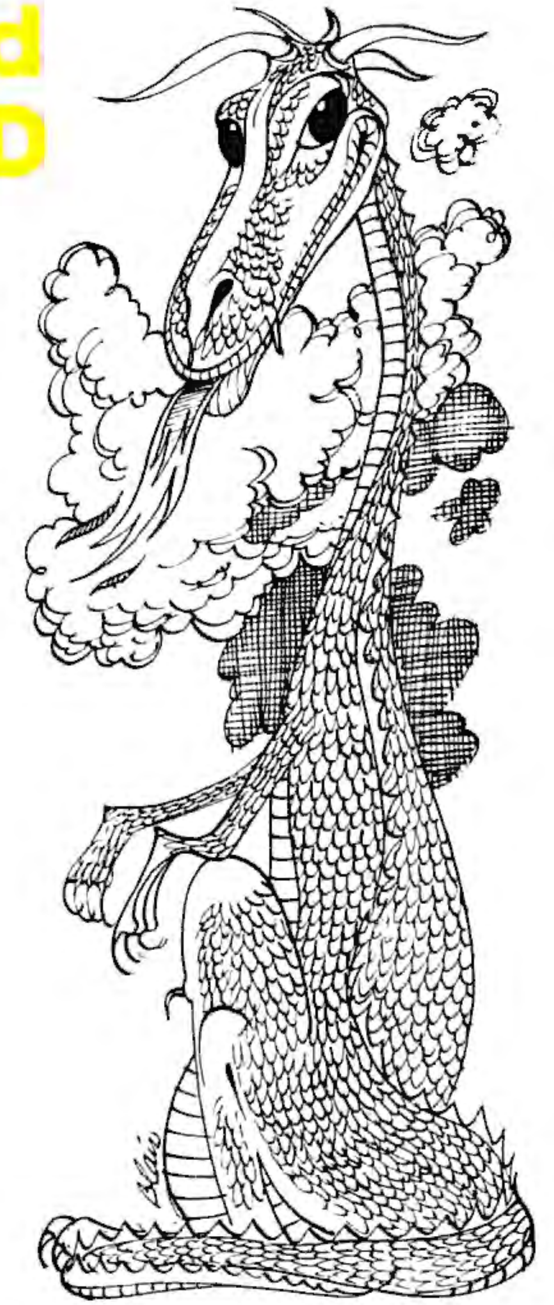
HOWARD INDUSTRIES
2051 E. CERRITOS AVE., 8C
ANAHEIM, CA 92806
714/778-3443

The Mysterious And Unpredictable RND

Bob Albrecht and
George Firedrake

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

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



> *Editor's Note:*
> *We conclude our presentation of The Mysterious And Unpredictable RND with this installment. We expect to make the series available to teachers, in booklet form, within the next few months.*
> *...RCL*

Solutions and Stuff

Here are our solutions. Yours may be different. That's OK, as long as they solve the problem. One really nice thing about computers: There are many ways to write a program that works!

Exercise 1.

(a) The smallest RND number is .0103099732 in the first sample.

(b) The largest RND number is .984101932 in second sample.

Exercise 2. Smallest RND Number In A Sample

```
100 REM::::SMALLEST RND NUMBER IN A SAMPLE

200 REM::::FIND OUT HOW BIG A SAMPLE
210 PRINT "[CLR]" ;
220 PRINT
230 INPUT "HOW MANY RND NUMBERS" ; N

300 REM::::SET SMALL EQUAL TO FIRST RND NUMBER
310 SMALL = RND(1)

400 REM::::DO REST OF SAMPLE. COMPARE EACH RND
410 REM::::NUMBER WITH SMALL. IF SMALLER, REPLACE.
420 FOR K = 1 TO N - 1
430   X = RND(1)
440   IF X < SMALL THEN SMALL = X
450 NEXT K

500 REM::::PRINT SMALL AND GO BACK FOR MORE
510 PRINT "LARGEST NUMBER IN SAMPLE IS" SMALL
520 GOTO 220

999 END
```

Exercise 3. The Small And Big

In this program, we first set *both* SMALL and BIG to the *same* first RND number (lines 310 and 320).

```
100 REM::::SMALLEST AND LARGEST RND NUMBER IN SAMPLE

200 REM::::FIND OUT HOW BIG A SAMPLE
210 PRINT "[CLR]" ;
220 PRINT
230 INPUT "HOW MANY RND NUMBERS" ; N

300 REM::::SET SMALL AND BIG EQUAL FIRST RND NUMBER
310 SMALL = RND(1)
320 BIG = SMALL

400 REM::::DO REST OF SAMPLE. COMPARE EACH RND
410 REM::::NUMBER WITH SMALL AND BIG.
420 FOR K = 1 TO N - 1
430   X = RND(1)
440   IF X < SMALL THEN SMALL = X
450   IF X > BIG THEN BIG = X
460 NEXT K

500 REM::::PRINT SMALL AND BIG, GO BACK FOR MORE
510 PRINT "SMALLEST NUMBER IN SAMPLE IS" SMALL
520 PRINT "LARGEST NUMBER IN SAMPLE IS" BIG
530 GOTO 220

999 END
```

The following method might not work. Why not?

310 SMALL = RND(1)
320 BIG = RND(1)

Exercise 4.

(a) 7 (b) 5 (c) 0

The integer part of .328904955 even though it isn't printed.

Exercise 5.

(a) 2 (b) 0 (c) 7

Exercise 6.

(a) 220 PRINT INT(2*RND(1)),
(b) 220 PRINT INT(6(RND(1)),
(c) 220 PRINT INT(100*RND(1)),

Exercise 7.

(a) 220 PRINT INT(2*RND(1))+1,
(b) 220 PRINT INT(8*RND(1))+1,
(c) 220 PRINT INT(100*RND(1)),+1,
(d) 220 PRINT INT(2*RND(1))+2,
(e) 220 PRINT INT(3*RND(1))+3,
(f) 5, 6, 7, or 8
(g) 2, 4, or 6

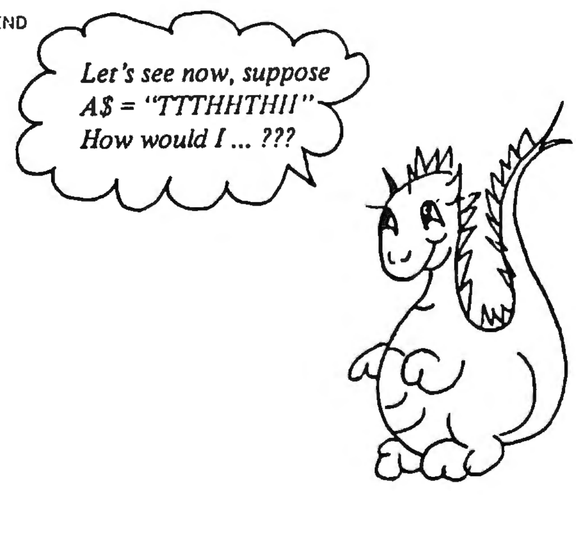
Exercise 8.

430 IF COIN=0 THEN T=T+1
440 IF COIN=1 THEN H=H+1

Exercise 9.

There are many ways to write this program. Here are two ways.

```
1ØØ REM***COIN FLIPPER #4

2ØØ REM***FIND OUT HOW MANY FLIPS
21Ø PRINT "[CLR]" ;
22Ø INPUT "HOW MANY COIN FLIPS" ; N

4ØØ REM***FLIP TWO COINS N TIMES
41Ø FOR K = 1 TO N

42Ø   C1 = INT(2*RND(1))
43Ø   C2 = INT(2*RND(1))

44Ø   IF C1 = 1 AND C2 = 1 THEN PRINT "HH" ,
45Ø   IF C1 = 1 AND C2 = Ø THEN PRINT "HT" ,
46Ø   IF C1 = Ø AND C2 = 1 THEN PRINT "TH" ,
47Ø   IF C1 = Ø AND C2 = Ø THEN PRINT "TT" ,

48Ø NEXT K
49Ø PRINT

999 END
```




```
1ØØ REM***COIN FLIPPER #4A
11Ø A$(Ø)="TT" : A$(1)="TH" ; A$(2)="HT" : A$(3)="HH"

2ØØ REM***FIND OUT HOW MANY FLIPS
21Ø PRINT "[CLR]" ;
22Ø INPUT "HOW MANY FLIPS" ; N

4ØØ REM***FLIP TWO COINS N TIMES
41Ø FOR K = 1 TO N

42Ø   C1 = INT(2*RND(1))
43Ø   C2 = INT(2*RND(1))

44Ø   PRINT A$(2*C1 + C2)

45Ø NEXT K
46Ø PRINT

999 END
```

Exercise 10.

We did it by modifying our first program of Exercise 9. Make these changes and additions to COIN FLIPPER 4.

```
1ØØ REM***COIN FLIPPER #5

3ØØ REM***SET FLIP COUNTERS TO ZERO
31Ø HH = Ø
32Ø HT = Ø
33Ø TH = Ø
34Ø TT = Ø

44Ø   IF C1 = 1 AND C2 = 1 THEN HH = HH + 1
45Ø   IF C1 = 1 AND C2 = Ø THEN HT = HT + 1
46Ø   IF C1 = Ø AND C2 = 1 THEN TH = TH + 1
47Ø   IF C1 = Ø AND C2 = Ø THEN TT = TT + 1

5ØØ REM***PRINT RESULTS OF N FLIPS
51Ø PRINT "OUTCOME", "NUMBER OF TIMES"
52Ø PRINT " HH    ", HH
53Ø PRINT " HT    ", HT
54Ø PRINT " TH    ", TH
55Ø PRINT " TT    ", TT
```

Exercise 11.

Program to roll two dice, N times.

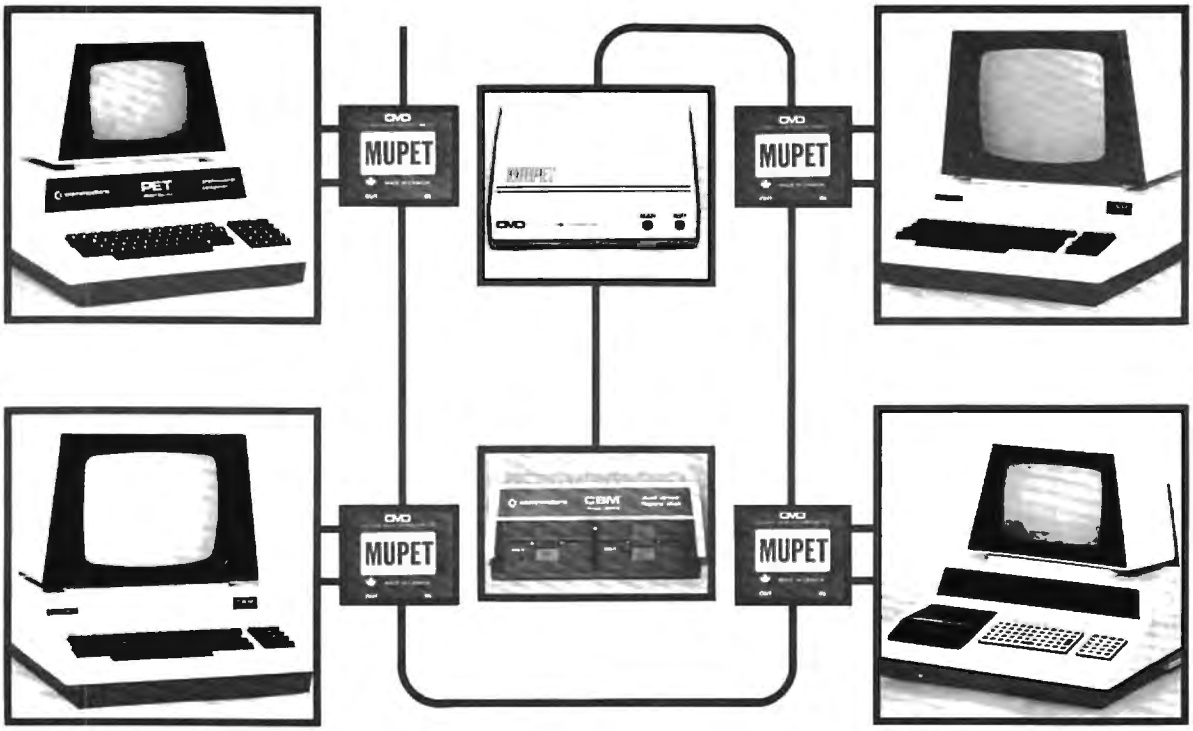
```
1ØØ REM***DICE ROLLER #2

2ØØ REM***FIND OUT HOW MANY ROLLS
21Ø PRINT "[CLR]" ;
22Ø INPUT "HOW MANY DICE ROLLS" ; N

4ØØ REM***ROLL TWO DICE N TIMES
41Ø FOR K = 1 TO N
42Ø   D1 = INT(6*RND(1)) + 1
43Ø   D2 = INT(6*RND(1)) + 1
44Ø   SUM = D1 + D2
45Ø   PRINT SUM,
46Ø NEXT K
47Ø PRINT

999 END
```

Exercises 12 and 13.

OUTCOME	NUMBER OF WAYS	PROPORTION
2	1	1/36 = .0278
3	2	2/36 = .0556
4	3	3/36 = .0833
5	4	4/36 = .1111
6	5	5/36 = .1389
7	6	6/36 = .1667
8	9	5/36 = .1389
9	4	4/36 = .1111
10	3	3/36 = .0833
11	2	2/36 = .0556
12	1	1/36 = .0278

Now The PET's Know How To Share

MUPET MULTIUSER SYSTEM



EDUCATION

* allows direct access to any programmes or data files
* reduces media cost - one set of disks needed for up to 8 users
* disk based languages such as Assembler, Pascal, Word Processing, available to all users

BUSINESS

* true multi-station wordprocessing with access to common data files
* cost per station drastically reduced
* printer spooling is supported

MANUFACTURING

* using optional DOUBLE MUPET, both local and common IEEE devices are supported
* test equipment can be controlled locally, while reports and programmes are managed universally

MUPET
USES
NO
NONE
NOT ANY
NOT ONE BYTE
OF
SOFTWARE

****

YOU CAN RUN
ANY COMBINATION
OF PROGRAMMES
OR LANGUAGES
AT THE SAME TIME

* word processing
* Pascal
* Assembler
* Future???

* MUPET supports any combination of CBM hardware — present or future — which supports the IEEE bus
* MUPET plugs in externally. No need to open the PET/CBM cabinet
* MUPET supports all existing and future CBM disk based software
* MUPET supports printers as well as disks
* MUPET supports **all** versions of Commodore Basic

PET/CBM Trademark of Commodore Business Machines

CMD

CANADIAN MICRO DISTRIBUTORS LTD.
365 Main Street, Milton, Ontario L9T 1P7 / 416 878-7277

Exercise 14.

In this program, we use FREQUENCY to mean NUMBER OF TIMES.

```
100 REM****DICE ROLLER #3

200 REM****FIND OUT HOW MANY FOLLS
210 PRINT " [CLR] " ;
220 INPUT "HOW MANY ROLLS" ; N

300 REM****SET OUTCOME COUNTS TO ZERO
310 DIM F(12)
320 FOR X = 2 TO 12
330   F(X) = 0          <----  F(X) will be the number of
340 NEXT X                     times outcome X occurred.

400 REM****ROLL DICE, COUNT OUTCOMES
410 FOR K = 1 TO N
420   D1 = INT(6*RND(1)) + 1 |
430   D2 = INT(6*RND(1)) + 1 | X is sum of two dice, D1 and D2
440   X = D1 + D2            |

450   F(X) = F(X) + 1 <----  Increase count for outcome X by 1

460 NEXT K

500 REM****PRINT COUNTS AND PROPORTIONS
510 PRINT
520 PRINT "OUTCOME, "FREQUENCY", "PROPORTION"
530 PRINT
540 FOR X = 2 TO 12
550   PRINT X, F(X), F(X)/N
560 NEXT X

999 END
```

Exercise 15.

Since we had to roll three dice six times, we used a *subroutine* to roll the dice.

```
100 REM****CREATE AN ADVENTURER
110 PRINT " [CLR] " ;

200 REM****ROLL = SUM OF THREE DICE
210 GOSUB 310 : PRINT "STR", ROLL
220 GOSUB 310 : PRINT "IQ", ROLL
230 GOSUB 310 : PRINT "LK", ROLL
240 GOSUB 310 : PRINT "CON", ROLL
250 GOSUB 310 : PRINT "DEX", ROLL
260 GOSUB 310 : PRINT "CHR", ROLL
270 STOP

300 REM****SUBROUTINE TO ROLL 3 DICE
310 D1 = INT(6*RND(1)) + 1
320 D2 = INT(6*RND(1)) + 1
330 D3 = INT(6*RND(1)) + 1
340 ROLL = D1 + D1 + D3
350 RETURN

999 END
```

Exercises 16 and 17.

We would like to see *your*solutions. Please send them to DragonQuest, P.O. Box 310, Menlo Park, CA 94025.

Land of the Lost

A Program For A Cassette Filing System

Steve Michel
Sterling, IL 61081

One day I pushed myself back from the green glow of the PET CRT and was struck by a fact that has been apparent to my family (translated here as wife) for quite some time. My office had become a jungle of little white plastic cases.

The major source of confusion was my cassette filing system. I HAD NONE!! There were some 200 plus programs strewn around on 100 plus C-10 cassettes. (I still drool over ads for floppy disk drives.) The disarray of cassettes was not so much a bother as was my MTBF. In most computer circles that stands for Mean Time Between Failure. In my case it stood for *Mean Time Between Finding.* It usually took me 2-3 times longer to find a particular program than it did to LOAD it. I decided it was a case of survival – find my way out now or be forever lost among those sequential magnetic I/O storage devices.

The ultimate solution was two pronged. The first step was to place each program into one of three categories:

1) EDUCATIONAL – I teach high school science.
2) UTILITIES – renumber, merger, business applications
3) GAMES – Need I say anything here?

These classifications covered the range of my programs fairly well.

The groups were then placed into appropriately labeled boxes. I have found that the boxes used to package those self-adhesive mailing labels that arrive on so much of our mail are an ideal size. They are exactly the right width and will hold about 15 cassettes. I get my boxes from a local industry that sends out mass mailings. The DP manager was more than happy to provide the empty boxes.

The last step in finding my way out of this "cassette block" was to devise a method for cataloging the programs, providing a short description of each, updating these as necessary and producing a final listing of the library contents. This effort resulted in the following program.

I tried to take an example from some of the larger computer systems and wrote a menu-driven program. This means that the operator is given a display on the screen which lists various options that can be selected by the pressing of a single key. After the option is complete, the user is then returned to the same or another menu to make another selection.

...It usually took me 2–3 times longer to find a particular program than it did to LOAD it...

The main advantage of this type of approach is that it allows people with little or no computer experience to feel comfortable and confident about running a particular job. It also cuts down on the chance of operator error because of the reduced input requirements.

PROGRAM DISSECTION:

Variables Used:

E$,U$,G$ ------ arrays that hold program names and the description of the programs
EX$ ------------- array used to LOAD and SAVE each of the individual categories
NM -------------- holds the total number of records LOADED or SAVED in each category
F ----------------- F 0 – return to SAVE MENU
F 1 – return to LOAD MENU
EN -------------- number of entry currently being edited
II ---------------- position in string that is beinng entered or edited
EE, EU, EG ---- number of titles *entered* from the keyboard for each category
LE, LU, LG ---- number of titles *loaded* from cassette file for each category
DN -------------- devise number on which final printed output will appear

Program Segments:

100 -------------- sets array sizes
105–1016 ------- **MAIN MENU**-listing of options
1. enter data from keyboard
2. save data file to cassette
3. load data file from cassette
4. print listing of titles
5. edit any previously entered data
6. end program
2000–2136 ----- **EXCHANGE ROUTINE**-this routine is used just prior to the SAVE routine which employs the general variable EX$-each category is transferred into EX$ before SAVE-ing.

3000–2136 ----- **SORT ROUTINE**-this is used to sort each category before it is saved to tape. It is a quick sort taken from **COMPUTE!**, issue 2, pg. 12.
4000–4391 ----- **EDIT ROUTINE**-this section allows any previously entered data to be reviewed or corrected.
It displays the entry and cursor by use of the cursor left and cursor right keys. Corrections are made by typing over the existing entry. No provisions were made for the insertion or deletion of characters.

MICRO MINI COMPUTER WORLD

74 Robinwood Ave.
Columbus, Ohio 43213
(614) 235-5813 or 235-6058

PRESENTS

BUSINESS ENHANCEMENTS
COMPUSERVICE BUSINESS SOFTWARE
for
COMMODORE AND APPLE COMPUTER SYSTEMS

B.E.C. #3000 Series

General Ledger	(Apple II & CBM 2001/8032/2040)	$150.00
Accounts Receivable	(Apple II & CBM 2001/8032/2040)	$150.00
Payroll	(Apple II & CBM 2001/8032/2040)	$150.00
Inventory	(Apple II & CBM 2001/8032/2040)	$150.00
Mail List	(Apple II & CBM 2001/8032/2040)	$150.00
Job Cost	(Apple II & CBM 2001/8032/2040)	$150.00
Accounts Receivables with Order Entry	(CBM 2001/8032/2040)	$220.00
ROM Control Chip required with CBM System		$ 70.00

B.E.C. #4000 Series NEW for 8032/8050

General Ledger	(CBM 8032/8050)	$175.00
Accounts Receivables with Order Entry	(CBM 8032/8050)	$220.00
Accounts Payable	(CBM 8032/8050)	$175.00
Payroll	(CBM 8032/8050)	$175.00
Inventory	(CBM 8032/8050)	$150.00
Mail List	(CBM 8032/8050)	$175.00
Job Costing	(CBM 8032/8050)	$150.00
ROM Control Chip required with CBM System		$ 70.00

Special Offer, buy any two (2) of the above packages and receive a 10% discount off suggested retail. Offer expires 1 July, 1981.

Send for documentation package for $20, apply this amount towards your first purchase or return documentation in re-saleable condition for complete refund.

CALL or WRITE for MMCWI's FREE CATALOG of computer products.

B.E.C. Software is Distributed in the East by MICRO MINI COMPUTER WORLD INC.

Dealer Inquiries Invited

Introducing
THE DOUGLAS POINT OF SALE SYSTEM

The DOUGLAS POINT OF SALE SYSTEM is a data processing system operating on the Commodore Business Machines 8032 or 2001 (with BASIC 4.0 ROM's installed), and the 8050 megabyte disk computing equipment. The standard business keyboard is required. The system integrates retail sales facilities and activities with the merchantile inventory to provide a complete "point-of-sale" data processing system for the retail trade establishment.

The DOUGLAS POINT OF SALE SYSTEM supports:

- Major requirements of the retail sale transaction:
  - Normal out-of-inventory sales.
  - Non-inventory sales such as installation fees.
  - Customer returns.
  - Down payments in cash or in used equipment.
  - Cash and credit transactions.
  - Provision for tax exempt customers.
  - Control of loaned inventory items.
  - Sales tax computations for sales and returns.
  - Printed receipts with company name and receipt number.
  - Capability to add personalized notes on each receipt.
- Integration of sales activity with inventory:
  - Automatic adjustment to inventory at time of sale.
  - Customer returns posted back to inventory at time of sale.
  - Stock replenishment determined at time of sale.
  - Complete sales history captured during each sale.
- Automated processing of:
  - Mail lists of customer and distributor addresses.
  - Major item customer inventory.
  - Trade-in acceptance data for used inventory report.
  - The inventory receiving process and associated reports.
  - Service and maintenance contracts.
  - Daily sales activity report.
  - File purge and system backup.
  - Password security for unattended computer.
  - Pricing and addressing labels.

DISTRIBUTED BY:

micro mini COMPUTER WORLD Inc.
TOWN & COUNTRY SHOPPING CENTER

SUGGESTED RETAIL $750.00

CALL OR SEND FOR MMCWI CATALOG

DEALER INQUIRIES INVITED

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

MICRO MINI COMPUTER WORLD

74 Robinwood Ave. Columbus, Ohio 43213
(614) 235-5813 or 235-6058

PRESENTS

THE INTEGRATED COMPUTER TECHNOLOGIES

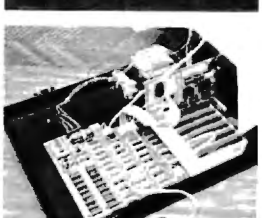
Prioress-44 Internal Motherboard:
The ICT P-44 is a 44 pin internal motherboard that facilitates expansion of your PET/CBM within the PET enclosure. The Prioress-44 is fully shielded on its underside by a massive ground plane. The connectors utilize any standard 44 pin edge card (many styles are available from Radio Shack). The following signals comprise the P-44 bus:

- +9v, −9v, +16v,GND,IRQ,RES,NMI,RDY,B02.
- BA0-BA15,BD0-BD7,BR/W,BW/R,SEL8,SEL9,SELA,SELB.
- DIAGNOSTIC SENSE,SYNC and 3 User definable.

The Prioress-44 is currently available for the new 2000 and 4000 series, and is under development for the 8000 series.

All ICT cards utilize the Prioress-44 bus.

Price: Prioress-44 with one connector . . $79.00
Prioress-44 with six connectors . . 95.00
Each additional connector 4 00
(specify when ordering)



The ICT Programmable Character Generator:
The ICT Programmable Character Generator is a 2K RAM replacement for the PET/CBM Character Generator ROM. The device allows the user to reprogram any or all of the 256 standard PET screen characters. The PCG also functions as 2K bytes of RAM in the $9000-$BFFF address range.

Uses of the ICT PCG:
a) Foreign character sets.
b) Math, Engineering and special notations.
c) Music notation.
d) Flow control and modeling.
e) Schematic and logic symbols.
f) Character oriented game symbols.
g) Architectural Drawings.
h) 320Hx200V BIT GRAPHICS
i) . . .many, many more.

The PCG has an empty socket for the original PET/CBM ROM. With the provided external switch, RAM or ROM may be selected.

ICT provides over 128K of software and data, allowing the user to immediately utilize the graphics system with extreme ease. Software is provided on 2040 format diskette and includes:

a) 7 complete 2K character sets (Russian, Katakana +).
b) Predefined graphics (including the Real-time rotating cube).
c) Development Tools including:
Charentry - used to program characters in an 8x8 matrix.
Draw - a program that allows drawing in a 320x200 area.
Plot - two versions, x,y plotting in the 320x200 matrix. A fast assembler version and a readable BASIC version.
Screen Dump - an assembler program to dump the EXACT screen contents to a Commodore 2022 printer.

Price: PCG with 2040 diskette and manual $240.00
Manual alone 7.50

The ICT HexROM:
A six socket programmable ROM board. Any three of the sockets may be programmed to become ROMs at $9000, $9800, $A000, $A800, $B000 and/or $B800. A simple BASIC POKE equates any socket to any of the above addresses.

Price: HexROM and manual $110.00
DumROM (6 sockets at fixed addresses) 69.00

The ICT EPROMer:
The EPROMer will READ/PROGRAM/VERIFY the following EPROMs:
2758, 2716, 2732 (24 pin EPROMs) and
2764, 27128 (28 pin EPROMs).
To a maximum of 36 pin I/O (5V).

The software (written in assembler) will support the above EPROM types and also allow the user to define any new EPROM configurations (5V Vcc, 25V Vpp).

Price: EPROMer, software and manual $180.00

ICT Products distributed by Micro Mini Computer world Inc.

***** Special introductory offer *****
10% off all suggested retail prices
(offer expires July 1, 1981) **DEALER INQUIRIES INVITED**

ICT Integrated Computer Technologies

5000–5331 ----- **INPUT ROUTINE**-this is the heart of the program. It provides both the enter data and edit data function. I first started with an INPUT statement here but it wasn't flexible enough (or I wasn't smart enough) to accomodate the edit function. The routine gets one character at a time, checks the value of the character, branches to appropriate routines for cursor control, or adds the character to the correct spot in the input string. The routine wraps around both forward and backward and allows up to 75 characters per entry. A more detailed explanation follows:

- **5000** ------ clears keyboard buffer of accidental keystrokes
- **5010** ------- prints flashing underline cursor
- **5015** ------ get keystroke
- **5020** ------ checks for cursor left key
- **5025** ------ increments position of cursor in string
- **5030** ------ checks for cursor right key
- **5040** ------ checks for wrap around in forward direction
- **5090-5100**- prints character on screen and adds it to the correct position in the string being entered or edited
- **5300-5330**- performs cursor left and reverse wrap around

10000-16031 -- **ENTER DATA MENU**

19000-20121 -- **SAVE ROUTINE**-saves the selected group from previously saved file. Also gives messages for tape handling.

30000-32041 -- **PRINT ROUTINE**-prints list of selected titles. User defines whether output is to screen or to printer in lines 42000-42060. Devise number 4 for printer, 3 for screen. The output file is then opened to the correct devise in line 40075, the file is printed and then closed in 41002. Figure 1 shows a sample of the output.

A few words about program modifications. The LOAD and SAVE routines should be easily modified to accomodate those lucky disk users out there. I strongly recommend that all REMs be omitted from the program when typing in because they take up an extra 2.5K of memory.

Well, that is the way it works. Some corollary, somewhere, must say, "It always looks easier after its done." At this point, all of my programs are neatly stacked in 5 well labeled boxes, every program has been backed up on a master tape, every program has its listing filed in an appropriate folder and I have an alphabetical list and description of every program in my library. It feels great to be back in civilization again.

```
5 REM    STEVE MICHEL
10 REM    STERLING HIGH SCHOOL
15 REM    STERLING IL   61081
20 REM
100 DIMG$(150),E$(150),U$(150),EX$(150)
105 PRINT"ĥ↓↓             ɾMAIN MENUf̂↓↓"
110 PRINT"ɾ1f̂.>ENTER DATA           ɾ2f̂. ¬
      ¬SAVE FILE"
112 PRINT"↓ɾ3f̂. LOAD FILE           ɾ4f̂. ¬
      ¬PRINT LIST"
113 PRINT"↓ɾ5f̂. EDIT TITLES         ɾ6f̂. ¬
      ¬END PROGRAM"
115 GETA$:IFA$=""THENGOSUB1000:GOTO115
116 A=VAL(A$)
120 ONAGOTO10000,19000,30000,40000,4000,
      ¬18000
125 GOTO115
1000 TT=TT+1
1002 IFTT/2=INT(TT/2)THENR$="ɾ":GOTO1010
1005 R$="f̂"
1010 PRINT"ĥ↓↓↓↓↓↓↓↓↓↓↓↓↓↓              ¬
      ¬";R$;"ENTER CHOICE"
1015 FORJ=1TO500:NEXTJ:RETURN
2000 A=VAL(A$):ONAGOTO2010,2020,2030
2010 FORJ=1TONM:EX$(J)=E$(J):NEXTJ:
      ¬GOSUB3000:FORJ=1TONM:E$(J)=EX$(J):
      ¬NEXTJ
2015 GOTO2100
2020 FORJ=1TONM:EX$(J)=U$(J):NEXTJ:
      ¬GOSUB3000:FORJ=1TONM:U$(J)=EX$(J):
      ¬NEXTJ
2025 GOTO2100
2030 FORJ=1TONM:EX$(J)=G$(J):NEXTJ:
      ¬GOSUB3000:FORJ=1TONM:G$(J)=EX$(J):
      ¬NEXTJ
2100 PRINT"ĥ↓↓  REWIND DATA TAPE ¬
      ¬ɾ";NM$"f̂."
2105 PRINT"↓↓  PRESS '*' TO RETURN TO ¬
      ¬MENU."
2110 PRINT"↓↓  PRESS ANY KEY WHEN DONE."
2120 GETA$:IFA$=""THEN2120
2130 IFA$="*"THENIFF=0THEN19000
2133 IFA$="*"THENIFF=1THENF=0:GOTO30000
2134 IFF=1THENF=0:GOTO31000
2135 GOTO20020
3000 PRINT"ĥ↓↓NOW SORTING ";NM$;" ¬
      ¬PROGRAMS."
3100 TP=1:LOWER(1)=1:UPPER(1)=NM
3120 IFTP<=0THENRETURN
3140 LB=LOWER(TP):UB=UPPER(TP):TP=TP-1
3160 IFUB<=LBTHEN3120
3180 I=LB:J=UB:TEMP$=EX$(I)
3200 IFJ<1THEN3260
3220 IFTEMP$>=EX$(J)THEN3260
3240 J=J-1:GOTO3200
3260 IFJ<=ITHENEX$(I)=TEMP$:GOTO3400
3280 EX$(I)=EX$(J):I=I+1
3300 IFI>NMTHEN3360
3320 IFEX$(I)>=TEMP$THEN3360
3340 I=I+1:GOTO3300
3360 IFJ>ITHENEX$(J)=EX$(I):J=J-1:
      ¬GOTO3220
3380 EX$(J)=TEMP$:I=J
3400 TP=TP+1
3420 IFI-LB<UB-ITHENLOWER(TP)=I+1:
      ¬UPPER(TP)=UB:UB=I-1:GOTO3160
3440 LOWER(TP)=LB:UPPER(TP)=I-1:LB=I+1
3460 GOTO3160
4000 PRINT"ĥ↓↓          ɾEDIT FILE MENUf̂"
4010 PRINT"↓↓EDIT WHICH SET OF PROGRAM ¬
      ¬TITLES?"
4020 PRINT"↓ɾ1f̂. EDUCATIONAL       ɾ2f̂. ¬
      ¬UTILITIES"
4030 PRINT"↓ɾ3f̂. GAMES             ɾ4f̂. ¬
      ¬MAIN MENU"
4050 GETA$:IFA$=""THENGOSUB1000:GOTO4050
4060 A=VAL(A$)
4070 ONAGOTO4100,4200,4300,105
4080 GOTO4050
4100 PRINT"ĥ↓↓WHICH EDUCATIONAL TITLE ¬
      ¬TO EDIT ?"
4110 INPUT"NUMBER";EN
4115 PRINT"ĥ    1    5    10    15    20    ¬
      ¬25    30    35"
```

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

High Fidelity Voice Response Industrial Quality Recognition

PET — AIM-65 — APPLE II

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

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

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

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

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

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

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

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

Dealer Inquiries invited.

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

WORDPROPACK

WORDPROPACK and JINSAM are trademarks of Jini Micro-Systems, Inc
WordPro s a trademark of Professional Software Inc

"This module is much more powerful than you can imagine."

Robert Baker,
February, '81 KILOBAUD

With WORDPROPACK, JINSAM's WORDPRO interface, you obtain the ultimate "state of the art" business tool. And, WORDPROPACK is just one of seven accessory modules and systems available with JINSAM Data Manager.

JINSAM is Commodore approved. JINSAM is available for all Commodore 32K microcomputers.

Send only $15 for your own 84K 5 program demonstration system or

SEE YOUR NEAREST COMMODORE DEALER FOR A DEMONSTRATION

JINI MICRO-SYSTEMS, INC.

P.O. Box 274 • Riverdale, N.Y. 10463
PHONE: (212) 796-6200

```
4120 PRINT"    ";LEFT$(E$(EN),35)
4125 LR=LEN(E$(EN)):IFLR>35THENPRINT"↓";
     ¬RIGHT$(E$(EN),LR-35)
4130 PRINT"h↓↓↓↓↓↓HERE IS THE LINE AS ¬
     ¬IT EXISTS NOW."
4140 PRINT"↓SIMPLY EDIT OVER THE ¬
     ¬MISTAKES AND"
4150 PRINT"PRESS RETURN WHEN DONE."
4160 PRINT"h":PRINT">>>";:B$=E$(EN):
     ¬II=0:GOSUB5007:T$=B$
4170 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓IT THAT CORRECT ¬
     ¬NOW r̲(Y OR N)?"
4175 GETA$:IFA$=""THEN4175
4180 IFA$="Y"THENE$(EN)=T$:GOTO4000
4185 IFA$="N"THEN4115
4190 GOTO4175
4200 PRINT"ĥ↓↓WHICH UTILITY TITLE TO ¬
     ¬EDIT ?"
4210 INPUT"NUMBER";EN
4215 PRINT"ĥ    1    5     10    15    20    ¬
     ¬25    30    35"
4220 PRINT"    ";LEFT$(U$(EN),35)
4225 LX=LEN(U$(EN)):IFLX>35THENPRINT"↓";
     ¬RIGHT$(U$(EN),LX-35)
4230 PRINT"h↓↓↓↓↓↓HERE IS THE LINE AS ¬
     ¬IT EXISTS NOW."
4240 PRINT"↓SIMPLY EDIT OVER THE ¬
     ¬MISTAKES AND"
4250 PRINT"PRESS RETURN WHEN DONE."
4260 PRINT"h":PRINT">>>";:B$=U$(EN):
     ¬II=0:GOSUB5007:T$=B$
4270 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓IT THAT CORRECT ¬
     ¬NOW r̲(Y OR N)?"
4275 GETA$:IFA$=""THEN4275
4280 IFA$="Y"THENU$(EN)=T$:GOTO4000
4285 IFA$="N"THEN4215
4290 GOTO4275
4300 PRINT"ĥ↓↓WHICH GAME TITLE TO EDIT ¬
     ¬?"
4310 INPUT"NUMBER";EN
4315 PRINT"ĥ    1    5     10    15    20    ¬
     ¬25    30    35"
4320 PRINT"    ";LEFT$(G$(EN),35)
4325 LR=LEN(G$(EN)):IFLR>35THENPRINT"↓";
     ¬RIGHT$(G$(EN),LR-35)
4330 PRINT"h↓↓↓↓↓↓HERE IS THE LINE AS ¬
     ¬IT EXISTS NOW."
4340 PRINT"↓SIMPLY EDIT OVER THE ¬
     ¬MISTAKES AND"
4350 PRINT"PRESS RETURN WHEN DONE."
4360 PRINT"h":PRINT">>>";:B$=G$(EN):
     ¬II=0:GOSUB5007:T$=B$
4370 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓IT THAT CORRECT ¬
     ¬NOW r̲(Y OR N)?"
4375 GETA$:IFA$=""THEN4375
4380 IFA$="Y"THENG$(EN)=T$:GOTO4000
4385 IFA$="N"THEN4315
4390 GOTO4375
5000 GETA$:IFA$<>""THEN5000
5005 II=0:B$=""
5007 PRINT">";
5010 PRINT"↓±↑";:FORI=1TO30:NEXTI:
     ¬PRINT"↓< <↑";
5015 GETA$:IFA$=""THEN5010
5020 A=ASC(A$):IFA=157THEN5300
5025 II=II+1
5030 IFA=29THENPRINT">";:GOTO5040
5035 GOTO5050
5040 IFII=35THENPRINT:PRINT
5045 GOTO5010
5050 IFA=13ORA=141THENPRINT:RETURN
5070 IFA=160THENA=32:A$=CHR$(A)
5080 IF(A>95ANDA<160)OR(A<32)OR(A=34)OR(
     ¬A=20)THENII=II-1:GOTO5010
5090 IFII=1THENPRINTA$;:B$=A$+MID$(B$,2,
     ¬LEN(B$)):GOTO5010
5095 IFII=75THENPRINTA$:B$=B$+A$:RETURN
5100 PRINTA$;:B$=LEFT$(B$,II-1)+A$+MID$(
     ¬B$,II+1,LEN(B$))
5110 IFII=35THENPRINT:PRINT
5120 GOTO5010
5300 IFII=0THEN5010
5310 II=II-1
5320 IFII=34THENPRINT"<↑";
5330 PRINT"<";:GOTO5010
10000 PRINT"ĥ↓↓           r̲ENTER DATA ¬
      ¬MENUf̂↓↓"
10010 PRINT"↓↓↓r̲1f̂. EDUCATIONAL        ¬
      ¬r̲2f̂. UTILITIES"
10020 PRINT"↓r̲3f̂. GAMES              r̲4f̂. ¬
      ¬MAIN MENU"
10030 GETA$:IFA$=""THENGOSUB1000:
      ¬GOTO10030
10035 A1=VAL(A$)
10040 ONA1GOTO10100,14000,16000,105
10045 GOTO10030
10100 PRINT"ĥ↓↓ENTER ED. PROGRAMS (0 ¬
      ¬WHEN DONE)"
10103 PRINT"h↓↓↓    1    5     10    15    ¬
      ¬20    25    30    35"
10105 EE=EE+1
10110 PRINTEE+LE;:GOSUB5000:E$=B$
10115 IFE$="0"THENEE=EE-1:GOTO105
10117 IFE$="*"THENEE=EE-1:GOTO10110
10120 E$(EE+LE)=E$
13000 GOTO10105
14000 PRINT"ĥ↓↓ENTER UTILITY PROGS. (0 ¬
      ¬WHEN DONE)"
14003 PRINT"ĥ    1    5     10    15    20    ¬
      ¬ 25    30    35"
14005 EU=EU+1
14010 PRINTEU+LU;:GOSUB5000:U$=B$
14015 IFU$="0"THENEU=EU-1:GOTO105
14017 IFU$="*"THENEU=EU-1:GOTO14010
14020 U$(EU+LU)=U$
14103 PRINT"h↓↓↓    1    5     10    15    ¬
      ¬20    25    30    35"
15000 GOTO14005
16000 PRINT"ĥ↓↓ENTER GAME PROGRAMS  (0 ¬
      ¬WHEN DONE)"
16003 PRINT"ĥ    1    5     10    15    20    ¬
      ¬ 25    30    35"
16005 EG=EG+1
16010 PRINTEG+LU;:GOSUB5000:G$=B$
16015 IFG$="0"THENEG=EG-1:GOTO105
16017 IFG$="*"THENEG=EG-1:GOTO16010
16020 G$(EG+LG)=G$
16030 GOTO16005
16103 PRINT"h↓↓↓    1    5     10    15    ¬
      ¬20    25    30    35"
18000 PRINT"ĥ↓↓↓WANT TO SAVE YOUR DATA ¬
      ¬FIRST(Y OR N)?"
18020 GETA$:IFA$=""THEN18020
18030 IFA$="Y"THEN19000
18040 IFA$="N"THENEND
18050 GOTO18020
19000 PRINT"ĥ↓↓           r̲SAVE FILE MENUf̂"
20000 PRINT"↓↓SAVE WHICH SET OF PROGRAM ¬
      ¬TITLES?"
20001 PRINT"↓r̲1f̂. EDUCATIONAL      r̲2f̂. ¬
      ¬UTILITIES"
```

```
20003 PRINT"↓ɼ3f̂. GAMES              ɼ4f̂. ¬
      ¬MAIN MENU"
20004 GETA$:IFA$=""THENGOSUB1000:
      ¬GOTO20004
20005 A=VAL(A$)
20006 ONAGOTO20010,20014,20017,105
20007 GOTO20004
20010 IFA$="1"THENNM$="EDUCATIONAL":
      ¬NM=EE+LE:GOTO2000
20014 IFA$="2"THENNM$="UTILITIES":
      ¬NM=EU+LU:GOTO2000
20017 IFA$="3"THENNM$="GAMES":NM=EG+LG:
      ¬GOTO2000
20019 GOTO20004
20020 OPEN1,1,1,NM$
20022 PRINT"↓↓NOW WRITING ";NM$;" FILE."
20025 PRINT#1,NM
20030 FORQ=1TONM
20050 PRINT#1,EX$(Q)
20060 NEXTQ
20070 CLOSE1
20080 PRINT"ĥ↓↓ A TOTAL OF";NM;NM$;" ¬
      ¬TITLES WERE"
20090 PRINT"↓SAVED."
20100 PRINT"↓↓PRESS ANY KEY TO RETURN ¬
      ¬TO MAIN MENU"
20110 GETA$:IFA$=""THEN20110
20120 GOTO105
30000 PRINT"ĥ↓↓          ɼLOAD FILE ¬
      ¬MENUf̂↓↓"
30010 PRINT"↓↓LOAD WHICH SET OF PROGRAM ¬
      ¬TITLES?"
30015 PRINT"↓ɼ1f̂. EDUCATIONAL       ɼ2f̂. ¬
      ¬UTILITIES"
30020 PRINT"↓ɼ3f̂. GAMES              ɼ4f̂. ¬
      ¬MAIN MENU"
30030 GETL$:IFL$=""THENGOSUB1000:
      ¬GOTO30030
30035 L=VAL(L$)
30036 ONLGOTO30040,30050,30060,105
30037 GOTO30030
30040 IFL$="1"THENNM$="EDUCATIONAL":
      ¬LE=0:F=1:GOTO2100
30050 IFL$="2"THENNM$="UTILITIES":LU=0:
      ¬F=1:GOTO2100
30060 IFL$="3"THENNM$="GAMES":LG=0:F=1:
      ¬GOTO2100
30070 GOTO30030
31000 OPEN1,1,0,NM$
31005 PRINT"↓↓FOUND ";NM$;".  NOW ¬
      ¬LOADING."
31010 INPUT#1,NM
31020 FORJ=1TONM
31030 INPUT#1,EX$(J)
31040 NEXTJ
31045 CLOSE1
31050 ONLGOTO31060,31070,31080
31060 LE=NM:FORJ=1TONM:E$(J+EE)=EX$(J):
      ¬NEXTJ:GOTO32000
31070 LU=NM:FORJ=1TONM:U$(J+EU)=EX$(J):
      ¬NEXTJ:GOTO32000
31080 LG=NM:FORJ=1TONM:G$(J+EG)=EX$(J):
      ¬NEXTJ
32000 PRINT"ĥ↓↓A TOTAL OF ";NM;NM$;" ¬
      ¬TITLES WERE"
32010 PRINT"↓LOADED."
32020 PRINT"↓↓    PRESS ANY KEY TO ¬
      ¬CONTINUE."
32030 GETA$:IFA$=""THEN32030
32040 GOTO105
40000 PRINT"ĥ↓↓HAVE YOU SAVED THE ¬
      ¬TITLES ON TAPE(YORN)?"
40010 GETA$:IFA$=""THEN40010
40020 IFA$="Y"THEN42000
40025 IFA$="N"THEN40030
40029 GOTO40010
40030 PRINT"↓↓ THEY NEED TO BE SAVED ¬
      ¬(SORTED) BEFORE"
40035 PRINT"↓THEY CAN BE PRINTED. SAVE ¬
      ¬THEM FIRST!!"
40040 PRINT"↓↓ PRESS ANY KEY TO GO TO ¬
      ¬SAVE FILE MENU."
40045 GETA$:IFA$=""THEN40045
40049 GOTO19000
40050 PRINT"ĥ↓↓          ɼPRINT LIST ¬
      ¬MENUf̂↓↓"
40060 PRINT"↓↓↓ɼ1f̂. EDUCATIONAL         ¬
      ¬ɼ2f̂. UTILITIES"
40070 PRINT"↓ɼ3f̂. GAMES   "
40075 OPEN1,DN,1
40080 GETA$:IFA$=""THENGOSUB1000:
      ¬GOTO40080
40085 A1=VAL(A$)
40090 ONA1GOTO40100,40200,40300
40095 GOTO40080
40100 PRINT#1,"ĥ         EDUCATIONAL ¬
      ¬TITLES"
40105 PRINT#1:PRINT#1
40110 FORJ=1TOEE+LE
40120 PRINT#1,MID$(STR$(J),2);". ¬
      ¬";E$(J):NEXTJ:GOTO41000
40200 PRINT#1,"ĥ         UTILITIES ¬
      ¬TITLES"
40205 PRINT#1:PRINT#1
40210 FORJ=1TOEU+LU
40220 PRINT#1,MID$(STR$(J),2);". ¬
      ¬";U$(J):NEXTJ:GOTO41000
40300 PRINT#1,"ĥ               GAMES ¬
      ¬TITLES"
40305 PRINT#1:PRINT#1
40310 FORJ=1TOEG+LG
40320 PRINT#1,MID$(STR$(J),2);". ¬
      ¬";G$(J):NEXTJ:GOTO41000
41000 PRINT#1:PRINT#1:PRINT#1,"SM = ¬
      ¬STEVE MICHEL      CC = CREATIVE ¬
      ¬COMPUTING"
41002 CLOSE1,DN,1
41005 PRINT"↓↓↓ PRESS ANY KEY TO RETURN ¬
      ¬TO MAIN MENU"
41010 GETA$:IFA$=""THEN41010
41020 GOTO105
42000 PRINT"ĥ↓↓↓↓SELECT OUTPUT DEVICE ¬
      ¬DESIRED"
42010 PRINT"↓↓    ɼPf̂RINTER"
42020 PRINT"↓↓    ɼSf̂CREEN"
42030 GETA$:IFA$=""THEN42030
42040 IFA$="P"THENDN=4:GOTO40050
42050 IFA$="S"THENDN=3:GOTO40050
42060 GOTO42030
```

©

Odds & Ends on the 2040 Disk

Jim Butterfield

WARNING: If you get an unclosed file — which shows up with an asterisk on the directory — do not scratch it; you may harm other files. Instead, do a Verify (called COLLECT on 4.0 systems). ©

Using The 6522 to drive a Printer

Edward H. Carlson
Okemos, MI

Low price compatible with good quality. If you are reaching the edge of your budget, the fifty dollars you can save by buying the parallel version of a printer may loom large. I wanted a printer for word processing and chose the Comprint 912P as suitable for rough draft printing. I was confident that the 6522 VIA on the CPU board of my Ohio Scientific C2-4P could handle the parallel interfacing. VIA stands for Versatile Interface Adaptor, and it can easily be configured to handle all the handshaking involved in the parallel transfer of data.

This article will describe how to wire the 6522 to the printer and will give a machine language program to drive it. The discussion is not at all restricted to OSI computers, nor even to the Comprint printer since the same principles apply to interfacing to other printers.

You may be interested in the features of the Comprint that appealed to me for word processing. It is fast, quiet and simple in design. The letter quality is high for a dot matrix printer as it has a 9x12 matrix. It is quiet because it is an electrostatic printer. This technology uses rolls of black paper which are coated with aluminum. The print head sparks holes through the aluminum to expose the black color below. The silvery paper is low in cost, thin and somewhat of a nuisance to handle. However, it Xeroxes very well. The 912 prints 3 lines a second of 80 characters each.

The Comprint has a variety of parallel options including the IEEE-488 convention and both wide and narrow strobe modes. I purchased the Comprint soon after it appeared on the market and made the modifications they suggested to operate with the Apple II Parallel Interface Card. (Since I also have an Apple, the same printer serves both computers.) The signal lines into the printer include seven parallel lines for the ASCII data and one line for DAV which is a narrow (one clock cycle is enough) strobe that tells the printer when valid data is on the 7 line bus. Signal lines from the Comprint include NDAC which goes low to acknowledge that the printer has accepted the character, and NRFD (not ready for data) which goes high when the printer's data buffer is full.

The 6522 VIA has two 8-bit ports, A and B, each with two control lines. The two ports are not identical and for no good reason I use the B port for the seven line ASCII bus. Since the eighth line is not needed for ASCII, I use it for the "busy" signal (NRFD). The B port control lines CB1 and CB2 are used for NDAC and DAV respectively.

The listing shows a subroutine, OUTCHR, that prints one character. Also included is a DRIVER that uses some subroutines in the OSI BASIC ROM's to read tape so its contents can be sent to the printer. Of course, this driver will need to be altered if your computer is not an OSI machine.

Implementing a 6522 can be a frustrating experience because of its many options. It has 16 registers of which we need 5. Three of the registers need be set only once. but we have plenty of time per character, and it is simpler to set these registers each time the subroutine is entered. Line 160 enables the B port by setting bit 1 in the Auxiliary Control Register. In line 170, the Data Direction Register for B port is loaded such that lines 0 to 6 are output (for the ASCII character) and line 7 as input (for the DAV signal). Finally, the Peripheral Control Register must be tickled so that CB1 and CB2 know what is expected of them. This is done in line 210. Bits 7, 6, 5 are set to 100 so that CB2 will pulse low when the CPU writes to the VIA, (the strobe). Setting bit 4 tells the VIA to raise a flag when CB1 makes a low to high transition (the acknowledgement).

...implementing a 6522 can be a frustrating experience because of its many options...

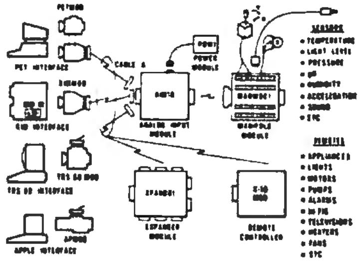
When the subroutine is entered, the accumulator A holds the character to be printed. It is saved by pushing it on the stack. Then the three registers mentioned above are configured. Next the VIA looks for the "busy" signal in lines 220 to 240. Upon finding a non-busy status, the character is pulled from the stack and sent to the B Output Register, and on to the printer. The last event is to detect the DAV acknowledgement. When it comes in on CB1, it sets a flag in the Interrupt Flag Register. Detecting this flag allows an exit from the loop of lines 300 to 330, and then exit from the subroutine.

There you have it. If you are interfacing to some other printer, the main thing to watch for is the polarity of the signal lines. Consult your 6522 data sheets for the code needed to reverse the polarity of the handshake signals. If by chance you have a Comprint 912P and have not configured it for Apple compatibility, I have written a program for that case too. An article describing it has been accepted for publication by BYTE. A copy of the program may be obtained by writing me at 3872 Raleigh Drive, Okemos, MI, 48864.

```
  1 0000           ;          ***   TAPE TO COMPRINT 912P   ***
  2 0000           ;
 10 C000            *         =$C000
 20 C000 2007BF    DRIVER JSR $BF07         GET CHAR. FROM TAPE PORT
 25 C003 8D00D2           STA $D200         STORE CHAR. ON SCREEN
 30 C006 200CC0           JSR OUTCHR        PRINT CHAR.
 40 C009 4C00C0           JMP DRIVER
 41 C00C           ;
 42 C00C           ;      MY ADDRESSES, SEE FOOTNOTE
 43 C00C           ;
 44 C00C           VIA   =$F700 ADDRESS OF 6522 IS $F7XX
 46 C00C           AUX   =$0E   AUXILIARY CTRL REGISTER
 48 C00C           BDD   =$08   B DATA DIRECTION REGISTER
 50 C00C           BPORT =$00   OUTPUT REGISTER FOR I/O PORT B
 52 C00C           PCTRL =$03   PERIPHERAL CONTROL REGISTER
 54 C00C           IFLAG =$07   INTERRUPT FLAG REGISTER
 60 C00C           ;
 61 C00C           ;      STANDARD ADDRESSES
 63 C00C           ;
 64 C00C           ; VIA     PER YOUR MACHINE
 66 C00C           ; AUX     =%1011
 68 C00C           ; BDD     =%0010
 70 C00C           ; BPORT   =%0000
 72 C00C           ; PCTRL   =%1100
 74 C00C           ; IFLAG   =%1101
134 C00C           ;
140 C00C 48        OUTCHR PHA               A CONTAINS CHARACTER
150 C00D A902             LDA #%00000010    ENABLE B PORT OF 6522
160 C00F 8D0EF7           STA VIA+AUX       AUX CTRL REGISTER
170 C012 A97F             LDA #%01111111    DATA DIRECTION
180 C014 8D08F7           STA VIA+BDD       B PORT DATA DIR REGISTER
190 C017 8D07F7           STA VIA+IFLAG     CLEAR INTERRUPT FLAGS
200 C01A A9B0             LDA #%10110000    PREPARE CB1 AND CB2
210 C01C 8D03F7           STA VIA+PCTRL     CB2 IS STROBE, PULSES LO
220 C01F AD00F7    BUSY   LDA VIA+BPORT     READ B PORT INPUT
230 C022 2980             AND #%10000000    BIT 7 IS NRFD OF COMPRINT
240 C024 30F9             BMI BUSY          BUSY IF BIT 7 IS HI
250 C026 68               PLA               LOAD CHAR. IN A
270 C027 8D00F7           STA VIA+BPORT     OUTPUT TO PRINTER
300 C02A AD07F7    ACK    LDA VIA+IFLAG     LOOK FOR NDAC ON CB1
310 C02D 2910             AND #%00010000    MASK OUT DESIRED FLAG
320 C02F C910             CMP #%00010000    NDAC IS ACKNOWLEDGE
330 C031 D0F7             BNE ACK           IF NOT FOUND, LOOK AGAIN
340 C033 60               RTS
350 C034           ;
400 C034           ; COMPRINT PARALLEL I/O BOARD (PBC 1184 Rev C)
405 C034           ;   HAS BEEN MODIFIED TO OPERATE WITH THE APPLE II
410 C034           ;   PARALLEL PRINTER INTERFACE CARD
415 C034           ;
420 C034           ; THE 6522 HAS ADDRESS LINES 0,1 CONNECTED TO
422 C034           ;   ADDRESSES 2,3 AND VICE VERSA
```

©

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



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

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

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

Connectors

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

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

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

Analog Input Module .

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

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

Power requirements are 12 volts DC at 60 ma.

POW1

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

TEMPSENS

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

Remote Controller-Clock and Calendar

AN INEXPENSIVE CONTROL SOLUTION FOR

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

SUPER X-10 MOD SPECS

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

XPANDR1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Printer And Communication Interfaces For The CBM/PET

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

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

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

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

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

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

ADA1600 • For Parallel NEC and Centronics Standard Printers

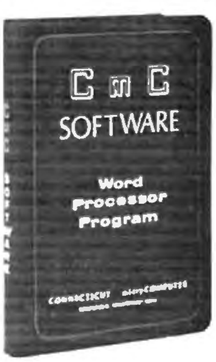
The ADA1600 is a low cost easy to use interface for the Commodore Computers. It allows the PET and CBM computers to use standard Centronics type printers (including the NEC 5530) for improved quality printing. The ADA1600 has a two foot cable which plugs into the PET IEEE port. Another IEEE card edge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

ADA1450 • Serial Printer Adapters

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

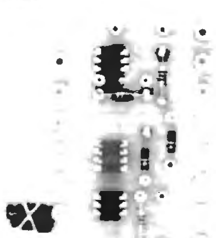
ADA730 Parallel • For the Centronics 730 and 737 Printers

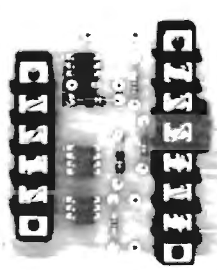
The ADA730 is a low cost easy to use interface for the Commodore Computers. It allows the PET and CBM computers to use Centronics type 730 and 737 printers. The ADA730 has a two foot cable which plugs into the PET IEEE port. Another IEEE card edge connector is provided for connecting disks and other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



Word Processor Program •

PET Word Processor. On tape - $39.50, On disk - 49.50
For 8K Pets 29.50
For 16K and 32K Pets 39.50
Compose and print letters, flyers, ads, manuscripts, etc. Uses disk or tape. 30 page manual included.





ADA400 RS-232 To Current Loop Adapter

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

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

QUANTITY	DESCRIPTION	PRICE	TOTAL

SUBTOTAL
Handling and shipping — add per order $4.00
Foreign orders add 10% for AIR postage
Conn. residents add 7% sales tax
TOTAL ENCLOSED

NAME ____________
COMPANY ____________
ADDRESS ____________
CITY ____________
STATE ____________ ZIP ____________
VISA O M/C O Expiration date ____________
Card number ____________

Mention this magazine and deduct 2% from the TOTAL

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

CmC

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

Using The Aim 65 As A Remote Terminal For An Apple

Tony Davis and Marvin L. De Jong
Department of Mathematics-Physics
The School of the Ozarks
Pt. Lookout, MO 65726

In the March issue of **COMPUTE!** (page 28 – Computer Communications Experiments) a circuit using the 6551 ACIA (Asynchronous Communications Interface Adapter) and a RS-232C interface to a modem were described. We have used this same interface to a NOVATION CAT modem on the AIM 65 to operate an Apple II over a telephone link. The Apple was equipped with a Hayes micromodem. The Apple was used to run BASIC programs, but its monitor can also be used to load machine language programs or data.

The circuit will not be repeated here, but we will provide the listing of the simple program that we used on the AIM 65. The Hayes Micromodem comes with its own firmware.

We operated the 6551 in the mode where a received character produces an interrupt. The interrupt routine simply prints the character on the display by jumping to an AIM 65 monitor subroutine. The program runs at 300 or 110 Baud. In Listing 1 we show the 6551 initialized to run at 300 Baud. Note that in either case the AIM 65 thermal printer was not used because its print time is so long that several characters are missed. To use it one would have to write a routine to buffer the incoming data. Our ultimate goal is to use the AIM 65 to access the college's big IBM mainframe. I am especially interested in being able to calculate my own salary and print my own paycheck at the end of each month.

Listing 1. Program to operate an Apple from an AIM 65 over a telephone line.

```
$0F00 58          START  CLI            Allow interrupts.
 0F01 D8                 CLD
 0F02 A9 09              LDA  #09       Set up the 6551
                                        command register.
 0F04 8D 02 94           STA  CMNDREG
 0F07 A9 16              LDA  #$13      Set up the control
                                        register for
 0F09 8D 03 94           STA  CNTREG    300 Baud.
 0F0C 20 3C E9    CHAR   JMP  READ      Get character from
                                        AIM keyboard.
 0F0F 8D 00 94           STA  DATA      Output data to the 6551.
 0F12 AD 01 94    CHECK  LDA  STATUS    Check the status register
 0F15 29 10              AND  #$10      Check bit four.
 0F17 FO F9              BEQ  CHECK     Wait for data to be
                                        transmitted.
 0F19 D0                 BNE  CHAR      Then get another
                                        character.
                         **********
Interrupt Routine
$0E00 48          IRQ    PHA            Save the accumulator.
 0E01 AD 00 94           LDA  DATA      Get character that
                                        was sent.
 0E04 20 7A E9           JMP  OUTPUT    Output character to
                                        display.
 0E07 AD 01 94           LDA  STATUS    Clear IRQ flag.
 0E0A 68                 PLA
 0E0B 40                 RTI
```

Be sure to load the interrupt vector $0E00.

THE STAR MODEM

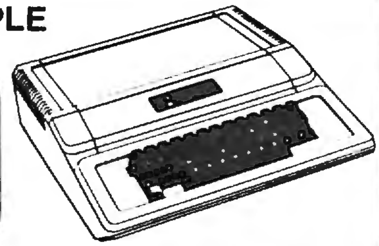
From Livermore Data Systems

RS232 MODEM	SALE $135
IEEE 488 MODEM	**SALE $199**
RS232 CCITT	$170
IEEE 488 CCITT	$280

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

WE CARRY THE BLACK APPLE

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



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

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

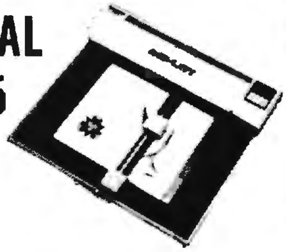
DISK MASTER by Robert Baker $12
Comprehensive diskette cataloging package to keep track of all programs and files on up to 140 diskettes. Specify 2040/4040/8050.

PROGRAM YOUR OWN EPROMS
Branding Iron for PET/CBM $79
EPROM Programmer with software for all ROM versions. Includes all necessary hardware and software to program or copy 2716 and 2532 EPROMS.

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

MIPLOT Intelligent Plotter
by Watanabe Instruments (Digiplot)

SPECIAL $1145



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

SPECIALS

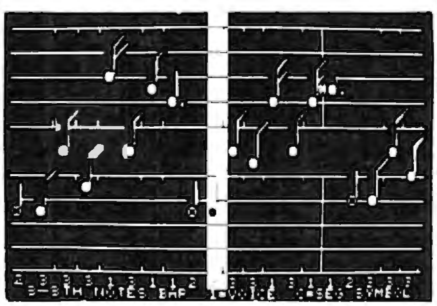
EPSON MX-80 Printer	
EPSON MX-70 Printer	
STARWRITER Daisy Wheel Printer	$1390
Centronics 737 Printer	725
NEC Spinwriter	2500
Diablo 630 Daisy Wheel	2150
Leedex Video 100 12" Monitor	135
ZENITH DATA SYSTEMS	
Z19 Video Terminal (factory assem.)	729
Z89 with 48K (factory assem.)	2150
SYM-1	209
SYM BAS-1 BASIC or RAE-1/2 Assembler	85
KTM-2/80 Synertek Video Board	349
Seawell Motherboard - 4K RAM	195
Seawell 16K Static RAM - KIM, SYM, AIM	320

6502	7.45	10/6.95	50/6.55	100/6.15
6502A	8.40	10/7.95	50/7.35	100/6.90
6520 PIA	5.15	10/4.90	50/4.45	100/4.15
6522 VIA	6.45	10/6.10	50/5.75	100/5.45
6532	7.90	10/7.40	50/7.00	100/6.60
2114-L200 ns RAM		3.75	25/3.50	100/3.25
2114-L300 ns RAM		3.15	25/2.90	100/2.65
2716 EPROM (5 volt)		9.90	5/9.45	10/8.90
TMS 2532 EPROM				23.50
4116-200 ns RAM				**8 for 29.00**
S-100 Wire Wrap				2.65

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

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

All other lengths available. Write for price list.



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

commodore

CBM-PET SPECIALS

FREE Up to $315 free merchandise with purchase of one of following CBM-PET items:

		FREE
8032 32K - 80 Column CRT	$1795	315
8050 Dual Disk Drive - 950,000 bytes	1795	315
4016 Full Size Graphics Keyboard	995	170
4032 Full Size Graphics Keyboard	1295	230
32B Business Keyboard -	SPECIAL	800
2040/4040 Dual Disk Drive - 340K	1295	230
4022 Tractor Feed Printer	795	140
CBM Voice Synthesizer	395	50
C2N External Cassette Deck	95	12
Used CBM/PET Computers		CALL
VIC Personal Computer	300	25

WRITE FOR SYSTEM PRICES

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

WordPro 3+ – 32K CBM, disk, printer	250
WordPro 4+ – 8032, disk, printer	385
OZZ Data Base System for CBM 8032	335
VISICALC for PET or ATARI	170
BPI General Ledger, A/P, A/R for PET/CBM	270
Programmers Toolkit – PET ROM Utilities	34.90
PET Spacemaker II ROM Switch	36.00
2 Meter PET to IEEE or IEEE to IEEE Cable	40.00
Dust Cover for PET	6.90
IEEE-Parallel Printer Interface for PET	110.00
IEEE-RS232 Printer Interface for PET	120.00
The PET Revealed	17.00
Library of PET Subroutines	17.00
Source Hookup over 1000 programs/services	88

DISK SPECIALS

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

WE STOCK MAXELL DISKS

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

ATARI 800 $745
All Atari Modules 20% OFF

ATARI EDUCATIONAL PLAN Write for details.

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

ALL BOOK and SOFTWARE PRICES DISCOUNTED

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

115 E. Stump Road
Montgomeryville, PA 18936 215-822-7727 **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.

EPIDEMIC

A Simulation Of An Epidemic In A Closed Community

Andy Gamble
Computer Science Instructor
Columbia College
Vancouver, BC V6J 2A2

It seems that the programs most readily accepted by students, with good reason, are the ones that involve an element of competition. How many times have you seen programs that, while advertised as simulations, are no more than textbooks written for the screen? The amount of student involvement is often limited to a 'Press any key to continue' or to a few multiple-choice questions.

EPIDEMIC operates in the realm of instruction and hypothesis-testing and, although the subject matter would seem to belie it, a certain amount of competitive fun. This competition, by the way, is on an individual basis, a kind of see-if-I-can-beat-the-machine, similar to the way HAMURABI works.

The program, through the use of PET graphics, illustrates how a disease could spread in a closed community. Given such a community, as for example an island with no physical connection to the world outside (are there any?), what are the parameters affecting the epidemic? The islanders move about randomly, infecting others if able to do so. The disease itself lasts for a specified amount of time, otherwise it is certain that all islanders will contract it. After this time, an infected person will become uncontagious, and also immune from further infection.

The RUN of the program prompts the user for such input as the number of inhabitants on the island, the number originally infected and the time for which the disease is contagious (lines 180-250).

Each person on the island is inspected to see if:

1) he is starting his period of infection. At this point a random move (or no move at all) is made (lines 500-520, 900-980 and 1000-1040). Note that no one is allowed to move off the island.

2) he is able to infect his immediate neighbors (lines 540 and 1060-1140)

3) he is infected by his neighbors (lines 560 and 1160-1240). This will only happen if he has not yet been infected.

4) his period of infection has finished (lines 580-620). He now passes into the immune category.

The program continues until there are no more infected people on the island (line 660). A bar chart is then presented which summarizes the history of the disease (lines 680-810).

The shape of the island is obtained from the DATA statements 1670-1710, and can easily be changed to suit your locale if you wish.

A further change could be a random element acting so that it is not absolutely certain that an islander will become infected if in contact with a diseased neighbor. Lines 1160-1230 would become

```
1160 IFPEEK(PP(I)-41)=COTHENIFRND-
  (1)*.5THENPOKEPP(I),CO etc.
```

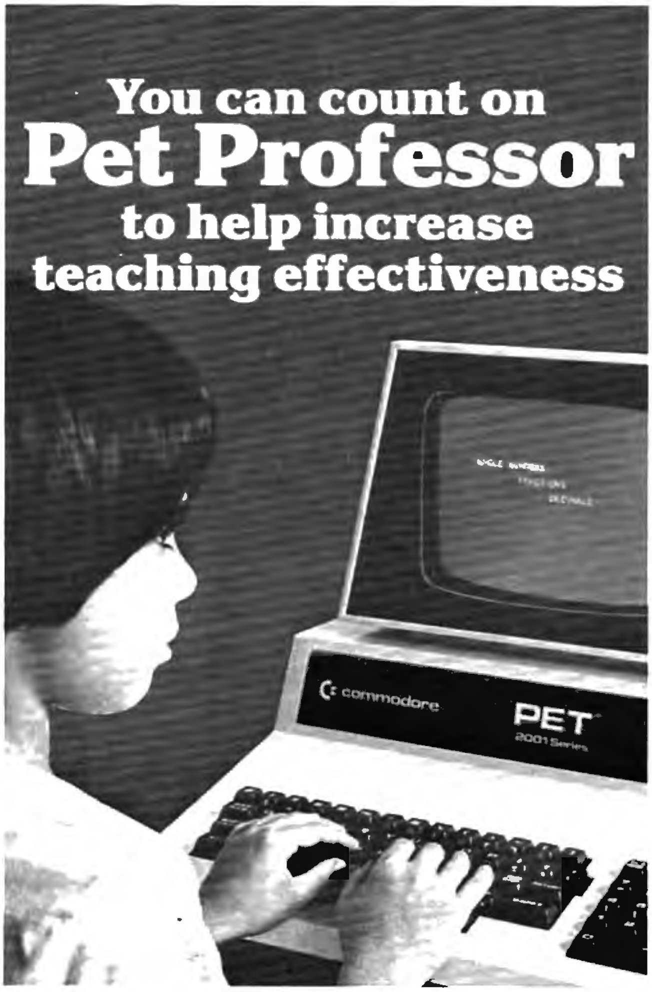
for a 50% chance of being infected.

Here is the challenge: given a constant number of inhabitants, what is the smallest number of infected people which insures that all will become infected? This makes a nice problem in statistics, if you want to go that far, but it is enjoyable to obtain experimental evidence from this program. That, after all, is what simulations are for.

The program will run on new and old roms, and uses less than 6K as given.

```
100 REM EPIDEMIC : ANDY GAMBLE, AUG 80
110 REM COLUMBIA COLLEGE,1619 W10 AVE   ¬
    ¬ VANCOUVER BC V6J 2A2
120 I=RND(-RND(0)):POKE59468,12
130 POKE59458,62:REM SPEED POKE
140 T$="h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓"
150 QP=515:QA=126:IFPEEK(50000)THENQP=15
    ¬1:QA=44
160 IG=160:VI=215:CO=209:GI=170
170 GOTO1360
180 PRINTCHR$(147)"HOW MANY ISLANDERS ¬
    ¬(<=100)?";:GOSUB1720:NP=VAL(Z1$)
190 IFNP=0THEN180
200 IFNP>100THENPRINT"↑↑";:GOTO180
210 PRINT"↓HOW MANY INFECTED AT ¬
    ¬START?";:GOSUB1720:NI=VAL(Z1$)
220 IFNI=0THENPRINT"↑↑";:GOTO210
230 IFNI>NPTHENPRINT"↑↑";:GOTO210
240 PRINT"↓CONTAGION TIME (DAYS)?";:
    ¬GOSUB1720:CT=VAL(Z1$)
250 IFCT=0THENPRINT"↑↑";:GOTO240
260 RESTORE
270 PRINTCHR$(147):FORI=1TO18:READIL,IR
280 FORJ=ILTOIR:POKEJ,IG:NEXT:NEXT
290 PRINT"hrWr̂=UNINFECTED  rQr̂=CONTAGIOU
    ¬S  r*r̂=IMMUNE"
300 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓CONTAGION"
310 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓TIME "CT;
320 IFCT>1THENPRINT" DAYS":GOTO340
330 PRINT" DAY"
340 REM GARBAGE COLLECTION FOR ARRAYS
350 POKEQA+2,PEEK(QA):POKEQA+3,PEEK(QA+1
    ¬):Z9=FRE(0)
360 DIMPP(NP),PG(NP),DC(NP),NI(50),
    ¬IM(50)
370 FORI=1TONP:PG(I)=VI:NEXT
380 FORI=1TONI:PG(I)=CO:NEXT
390 FORI=1TONP:DC(I)=0:NEXT
400 FORI=1TONI:DC(I)=CT:NEXT
410 FORI=1TONP
```

```
420 PP(I)=INT(690*RND(1)+32901):
      ¬IFPEEK(PP(I))<>IGTHEN420
430 POKEPP(I),PG(I):NEXT
440 ND=0
450 IM=0:PRINT"h↓"TAB(7)NP-NI-IM;TAB(20)
      ¬NI;TAB(30)IM
460 PRINT"h↓↓↓DAY";ND
470 ND=ND+1:NI=0:IM=0
480 FORI=1TONP
490 REM NEW CONTAGIOUS FROM LAST TIME
500 IF(PEEK(PP(I))=CO)AND(PG(I)=VI)THEND
      ¬C(I)=CT
510 PG(I)=PEEK(PP(I))
520 ONINT(9*RND(1)+1)GOSUB900,910,920,
      ¬930,940,950,960,970,980
530 REM INFECTING?
540 IFPG(I)=COTHENGOSUB1060
550 REM INFECTED?
560 IFPG(I)=VITHENGOSUB1160
570 REM ONE DAY LESS
580 IFPG(I)=COTHENDC(I)=DC(I)-1
590 REM END CONTAGION
600 IFDC(I)<0THENPG(I)=GI:POKEPP(I),GI
610 IFPEEK(PP(I))=COTHENNI=NI+1
620 IFPEEK(PP(I))=GITHENIM=IM+1
630 NEXT
640 IFND<=50THENNI(ND)=NI:IM(ND)=IM
650 PRINT"h↓"TAB(7)NP-NI-IM"<  ";TAB(20)
      ¬NI"<  ";TAB(30)IM"<  "
660 IFNI>0THEN460
670 PRINTT$;:GOSUB1250
680 PRINTCHR$(147)"DAY"TAB(5)"INFECTION"
      ¬:PRINT
690 FORND=1TO50
700 FORWT=1TO150:NEXT
710 PRINTND;TAB(4);
720 PRINTNP-NI(ND)-IM(ND);NI(ND);IM(ND):
      ¬PRINT
730 IFNP-NI(ND)-IM(ND)=0THEN750
740 FORI=1TONP-NI(ND)-IM(ND):PRINT"rW";:
      ¬NEXT
750 IFNI(ND)=0THEN770
760 FORI=1TONI(ND):PRINT"rQ";:NEXT
770 IFIM(ND)=0THEN790
780 FORI=1TOIM(ND):PRINT"r*";:NEXT
790 PRINT
800 IFNI(ND)=0THEN820
810 PRINT:NEXTND
820 PRINT"↓LIKE TO SEE THE CHART AGAIN ¬
      ¬(Y/N)?";:GOSUB1720:Q$=Z1$
830 IFQ$=""THENPRINT"↑↑";:GOTO820
840 IFLEFT$(Q$,1)="Y"THEN680
850 PRINTCHR$(147)LEFT$(T$,10)"WANT ¬
      ¬ANOTHER TRY (Y/N)?";:GOSUB1720:
      ¬Q$=Z1$
860 IFQ$=""THEN850
870 IFLEFT$(Q$,1)="Y"THEN180
880 END
890 REM MOVE S/R'S
900 MV=39:GOSUB1000:RETURN
910 MV=40:GOSUB1000:RETURN
920 MV=41:GOSUB1000:RETURN
930 MV=-1:GOSUB1000:RETURN
940 MV=0:GOSUB1000:RETURN
950 MV=1:GOSUB1000:RETURN
960 MV=-41:GOSUB1000:RETURN
970 MV=-40:GOSUB1000:RETURN
980 MV=-39:GOSUB1000:RETURN
990 REM MAKE MOVE
1000 IFPEEK(PP(I)+MV)<>IGTHENRETURN
1010 PG(I)=PEEK(PP(I))
1020 POKEPP(I),IG
1030 PP(I)=PP(I)+MV
1040 POKEPP(I),PG(I):RETURN
1050 REM INFECTING OTHERS
1060 IFPEEK(PP(I)-41)=VITHENPOKEPP(I)-41
      ¬,CO
1070 IFPEEK(PP(I)-40)=VITHENPOKEPP(I)-40
      ¬,CO
1080 IFPEEK(PP(I)-39)=VITHENPOKEPP(I)-39
      ¬,CO
1090 IFPEEK(PP(I)- 1)=VITHENPOKEPP(I)- ¬
      ¬1,CO
1100 IFPEEK(PP(I)+ 1)=VITHENPOKEPP(I)+ ¬
      ¬1,CO
1110 IFPEEK(PP(I)+39)=VITHENPOKEPP(I)+39
      ¬,CO
1120 IFPEEK(PP(I)+40)=VITHENPOKEPP(I)+40
      ¬,CO
1130 IFPEEK(PP(I)+41)=VITHENPOKEPP(I)+41
      ¬,CO
1140 RETURN
1150 REM INFECTION FROM OTHERS
1160 IFPEEK(PP(I)-41)=COTHENPOKEPP(I),CO
1170 IFPEEK(PP(I)-40)=COTHENPOKEPP(I),CO
1180 IFPEEK(PP(I)-39)=COTHENPOKEPP(I),CO
1190 IFPEEK(PP(I)- 1)=COTHENPOKEPP(I),CO
1200 IFPEEK(PP(I)+ 1)=COTHENPOKEPP(I),CO
1210 IFPEEK(PP(I)+39)=COTHENPOKEPP(I),CO
1220 IFPEEK(PP(I)+40)=COTHENPOKEPP(I),CO
1230 IFPEEK(PP(I)+41)=COTHENPOKEPP(I),CO
1240 RETURN
1250 PRINT"    rPRESS ANY KEY TO ¬
      ¬CONTINUE"
1260 GETQ$:IFQ$=""THEN1260
1270 RETURN
1280 REM*******VARIABLES*************
1290 REM NP=# OF PEOPLE,NI=#  INFECTED  ¬
      ¬           IM=# OF IMMUNE
1300 REM CT=DAYS FOR CONTAGIOUS,
      ¬IG=                ISLANDGRAPHIC,
      ¬PP=POS OF PEOPLE
1310 REM ND=# OF DAYS,VI=NOTYETINFECTED
1320 REM CO=CONTAGIOUS,PG=PEOPLEGRAPHIC
1330 REM GI=GRAPHIC IMMUNE,MV=MOVE,
      ¬DC=            DAYS OF CONTAGION ¬
      ¬LEFT
1340 REM*****************************
1350 REM TITLES
1360 PRINT"ħ":FORI=32768TO32807:
      ¬POKEI,224:POKEI+960,224:NEXT
1370 FORI=32808TO33688STEP40:POKEI,224:
      ¬POKEI+39,224:NEXT
1380 PRINT"h↓↓↓↓"
1390 PRINTTAB(9)"  Q# $'       L
1400 PRINTTAB(9)"  L$:::::::MNLLLLNM
1410 PRINT"h↓↓↓↓↓↓↓↓↓↓"TAB(14)"rEPIDEMIC
      ¬"
1420 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓"TAB(25)"ANDY ¬
      ¬GAMBLE"
1430 PRINTLEFT$(T$,20)TAB(5);"DO YOU ¬
      ¬NEED INSTRUCTIONS (Y/N)?";:
      ¬GOSUB1720:Q$=Z1$
1440 IFQ$=""THEN1430
1450 IFLEFT$(Q$,1)<>"Y"THEN180
1460 REM INSTRUCTIONS
1470 PRINTCHR$(147)"AN EPIDEMIC HAS ¬
      ¬BROKEN OUT ON A SMALL "
1480 PRINT"ISLAND. THE DISEASE IS NOT ¬
      ¬FATAL, AND"
```



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

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

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

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

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

Buy all 3 for $499.00. Save $86.00!

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

List of other educational programs available on request.

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

By Netronics

ASCII/BAUDOT, STAND ALONE Computer Terminal

COMPLETE FOR ONLY . . . $149.95

ASCII/BAUDOT
Computer Terminal Kit

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

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

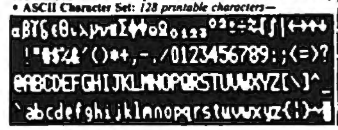
The Computer Terminal requires no I/O mapping and includes 1k of memory, character generator, 2 key rollover, processor controlled cursor control, parallel ASCII/BAUDOT to serial conversion and serial to video processing—fully crystal controlled for superb accuracy. PC boards are the highest quality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

VIDEO DISPLAY SPECIFICATIONS

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

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

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



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

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

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

Please send the items checked below—

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

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

NEW!!!-THE ELECTRIC MOUTH*

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

From $99.95 kit

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

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

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

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

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

Spoken Material Included

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

Also, the complete alphabet, spoken letter by letter.

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

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

Please send the items checked below:

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

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

Total Enclosed $______

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

```
1490 PRINT"ONCE CONTRACTED PROVIDES ¬
       ¬IMMUNITY FOR  LIFE."
1500 PRINT:PRINT"YOU ARE ALLOWED TO ¬
       ¬CHOOSE CERTAIN "
1510 PRINT"INITIAL CONDITIONS:":PRINT:
       ¬PRINT:PRINTTAB(5)"THE ISLAND ¬
       ¬POPULATION"
1520 PRINT:PRINTTAB(5)"THE NUMBER ¬
       ¬ORIGINALLY INFECTED"
1530 PRINT:PRINTTAB(5)"THE NUMBER OF ¬
       ¬DAYS FOR WHICH THE"
1540 PRINTTAB(5)"DISEASE IS CONTAGIOUS ¬
       ¬- THIS IS"
1550 PRINTTAB(5)"ALSO THE DURATION OF ¬
       ¬THE DISEASE."
1560 PRINT:PRINT:PRINT"THE ISLANDERS ¬
       ¬WILL MOVE ABOUT RANDOMLY,"
1570 PRINT"INFECTING OTHERS IF CONTAGIOU
       ¬S.";
1580 PRINT"THOSE WHOARE IMMUNE WILL NOT ¬
       ¬BE  INFECTED. "
1590 PRINT: GOSUB1250
1600 PRINTCHR$(147)"THIS WILL CONTINUE ¬
       ¬UNTIL THE DISEASE HASRUN ITS ¬
       ¬COURSE ";
1610 PRINT"(UNTIL THERE ARE NO MORE ¬
       ¬INFECTED PERSONS)."
1620 PRINT:PRINT"YOU WILL THEN BE GIVEN ¬
       ¬A DAY-BY-DAY BAR CHART OF THE ";
1630 PRINT"HISTORY OF THE EPIDEMIC, "
1640 PRINT"UP TO A MAXIMUM OF 50 DAYS."
1650 PRINT:GOSUB1250
1660 GOTO180
1670 DATA32902,32911,32940,32955
1680 DATA32978,32996,33015,33040,33050,
       ¬33083,33090,33123,33130,33162
1690 DATA33171,33203,33213,33244,33254,
       ¬33285,33297,33325
1700 DATA33337,33362,33376,33398,33417,
       ¬33437,33459,33475,33501,33515
1710 DATA33544,33556,33585,33592
1720 Z$="":Z1$=""
1730 PRINT"&<";:FORI=1TO50:NEXTI
1740 PRINT" <";:FORI=1TO50:NEXTI
1750 GETZ$:IFZ$=""THEN1730
1760 IFZ$<>CHR$(20)THEN1810
1770 IFZ$=""THEN1730
1780 ZZ=LEN(Z1$):IFZZ<1THEN1730
1790 Z1$=LEFT$(Z1$,ZZ-1):PRINT"<";
1800 GOTO1730
1810 IFZ$=CHR$(13)ORZ$=CHR$(141)THEN1850
1820 PRINTZ$;
1830 Z1$=Z1$+Z$
1840 GOTO1730
1850 FORI=1TO10:GETZ$:NEXTI
1860 PRINT
1870 RETURN
```

©

Odds & Ends on the 2040 Disk

Jim Butterfield

Verify works this way: all blocks on disk are freed. Then the processor works through the files in the directory, one by one, and re-allocates the blocks it finds in use there. You should not verify a disk that contains direct files unless special provision is made to allow this. Otherwise, the blocks will be freed and not re-allocated — which means trouble.

©

A Floating Point Multiplication Routine

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

Introduction

In two previous articles in **COMPUTE!** we have described:

1) A routine that inputs any signed number with magnitude between 1.70141183*E38 to 1.46936795*E-39 and converts it to a floating-point binary number.

2) A routine that outputs a signed floating-point binary number to an output device in BCD code.

In this article we add a floating-point multiplication routine to this set of routines that will eventually become a four-function floating-point package with nine digit accuracy.

The Floating-Point Multiplication Routine

A floating-point multiplication routine is given in Listing 1, and its flowchart is shown in Figure 1. The flowchart is essentially the same as that of B. Hashizume (**BYTE, V2**, Number 11, November 1977, p76). Studying the flowchart and the program comments should make the process understandable.

The multiplication routine uses three accumulators. Accumulator A occupies locations $0000 through $0003 with the most-significant byte in location $0000. Since the mantissa is normalized, there will always be a one in Bit 7 of location $0000, unless the mantissa is identical to zero. Location $0004 is used as a "guard" byte to do a 40-bit multiplication. The 40-bit result is rounded to 32 bits, giving approximately nine-digit decimal accuracy. Accumulator B occupies locations $0020 through $0023, with a guard byte in location $0024, an exponent (twos complement code) in location $0025, and a sign ($FF for minus, $00 for plus) in location $0027. The routine multiplies the contents of accumulator A with the contents of accumulator B. Intermediate results are stored in RES from $0010 to $0014.

The accumulator architecture just described proved to be very convenient for the multiplication routine. However, it differs slightly from the accumulator architecture used in the routines described in previous articles of this series. Rather than modify those two routines, which would not be difficult if you wish to try, we have included a little subroutine in Listing 2 that adjusts the accumulator used by the input routine to conform to the accumulator used in the multiply routine. Thus, after the BCD to Floating-Point Binary routine is called, the subroutine in Listing 2 must be called.

Once the accumulator is properly adjusted, it is moved to Accumulator B to await multiplication. The BCD to Floating-Point Binary routine is then called again to get the second number. Its accumulator is again adjusted to make it Accumulator A. Then the multiply routine is called, and finally the Floating-Point Binary to BCD routine is called to output the answer. This entire process is accomplished by the program in Listing 5, and this program can be used to test all three programs for proper operation.

One very important note. The BCD to Floating-Point Binary routine must be modified with the instruction listed in Listing 4 in order for it to work with the multiplication routine. The change is simple. Modify the byte at $0E02 from $20 to $1F. This prevents Accumulator B's most significant byte from being cleared whenever the BCD to Floating-Point Binary routine is called.

And a final note. If the combination of exponents to form the exponent of the result produces an overflow (exponent larger than 127 or exponent smaller than −128), the multiplication routine executes a BRK instruction. Normally this will send control back to the monitor, but one could write an interrupt routine to signal an overflow or an underflow.

Listing 1. The Floating-Point Multiplication Routine.

```
$0000 = ACCA; Most-significant byte of accumulator A.
$0005 = ACCX; Exponent for accumulator A.
$0007 = ACCS; Sign byte for accumulator A.
$0020 = ACCB; Most-significant byte of accumulator B.
$0025 = BCCX; Exponent for accumulator B.
$0027 = BCCS; Sign byte for accumulator B.
$0010 = RES; Most-significant byte of "result" accumulator.
$0014 = GRDR; "Guard" byte for "result" accumulator.
$0C28 A5 07   START    LDA ACCS       Determine the sign of the result.
 0C2A 45 27            EOR BCCS       Positive sign if signs are alike,
 0C2C 85 07            STA ACCS       negative otherwise.
 0C2E 18               CLC            To multiply, add exponents.
 0C2F A5 05            LDA ACCX
 0C31 65 25            ADC BCCX
 0C33 50 01            BVC ARND       Break to monitor if an exponent
 0C35 00               BRK            overflow (or underflow) results.
 0C36 85 05   ARND     STA ACCX       Store result into EXPONENT.
 0C38 A2 04            LDX #$04       Clear the locations that store
 0C3A A9 00            LDA #$00       the result for the mantissa.
 0C3C 05 10   HERE     STA RES,X
 0C3E CA               DEX
 0C3F 10 FB            BPL HERE
 0C41 A0 28            LDY #$28       Do a 40 ($28) bit multiplication
 0C43 A2 FB   BR2      LDX #$FB       starting here.
 0C45 18               CLC
 0C46 76 25   BACK     ROR ACCB+5,X   Rotate Multiplier right into carry.
 0C48 E8               INX
 0C49 D0 FB            BND BACK
 0C4B 90 0C            BCC PAST       No carry; don't add.
 0C4D A2 04            LDX #04        Add Multiplicand to Result.
 0C4F 18               CLC
 0C50 B5 00   MORE     LDA ACCA,X
 0C52 75 10            ADC RES,X
 0C54 95 10            STA RES,X
 0C56 CA               DEX
 0C57 10 F7            BPL MORE
 0C59 A2 FB   PAST     LDX #$FB       Shift Result right one bit.
 0C5B 76 15   BR1      ROR RES+5,X
 0C5D E8               INX
 0C5E D0 FB            BNE BR1
 0C60 88               DEY            Back for another bit in the
 0C61 D0 E0            BNE BR2        multiplier?
 0C63 A5 10   BR4      LDA RES        Check for zero result.
 0C65 F0 3F            BEQ OUT        If so, get out.
 0C67 30 14            BMI DETOUR     Check if mantissa is already
 0C69 A2 04            LDX #04        normalized.
 0C6B 18               CLC
 0C6C A5 05            LDA ACCX       For each shift left, decrement
 0C6E E9 00            SBC #00        exponent.
 0C70 50 01            BVC BR8        Overflow set?
 0C72 00               BRK            Yes, go to monitor.
 0C73 85 05   BR8      STA ACCX
 0C75 18               CLC
 0C76 36 10   BR3      ROL RES,X
 0C78 CA               DEX
 0C79 10 FB            BPL BR3
 0C7B 30 E6            BMI BR4
 0C7D A5 14   DETOUR   LDA GRDR       If most-significant bit of guard
```

```
0C7F 10  1C             BPL  BR5         byte is one, then round up.
0C81 38                 SEC
0C82 A2  03             LDX  #03
0C84 B5  10     BR6     LDA  RES,X
0C86 69  00             ADC  #00
0C88 95  10             STA  RES,X
0C9A CA                 DEX
0C9B 10  F7             BPL  BR6
0C8D 90  0E             BCC  BR5         Did rouding produce a carry from
0C8F A9  80             LDA  #$80        the mantissa?
0C91 85  10             STA  RES         Yes. Fix mantissa.
0C93 38                 SEC              And adjust exponent.
0C94 A5  05             LDA  ACCX
0C96 69  00             ADC  #00
0C98 50  01             BVC  BR9         Check for overflow.
0C9A 00                 BRK              Jump to monitor on overflow.
0C9B 85  05     BR9     STA  ACCX
0C9D A2  03     BR5     LDX  #03         Move result to accumulator for
0C9F B5  10     BR7     LDA  RES,X       the output (Binary to BCD) routine.
0CA1 95  01             STA  ACCA+1,X
0CA3 CA                 DEX
0CA4 10  F9             BPL  BR7
0CA6 60         OUT     RTS              Get out.
```

Listing 2. A Subroutine to Modify the Accumulator of the BCD-to-Binary Routine.

```
$0FB0 A0  08    SUB1    LDY  #08         Rotate the accumulator one byte
 0FB2 A2  04    B2      LDX  #04         (eight bits) left.
 0FB4 18                CLC
 0FB5 36  00    B1      ROL  ACCA,X
 0FB7 CA                DEX
 0FB8 10  FB            BPL  B1
 0FBA 88                DEY
 0FBB D0  F5            BNE  B2
 0FBD 60                RTS
```

Listing 3. A Subroutine to Transfer Accumulator A to Accumulator B.

```
$0FC0 A2  07    SUB2    LDX  #07         Move ACCA to ACCB.
 0FC2 B5  00    B3      LDA  ACCA,X
 0FC4 95  20            STA  ACCB,X
 0FC6 CA                DEX
 0FC7 10  F9            BPL  B3
 0FC9 60                RTS
```

Listing 4. An IMPORTANT Modification to the BCD-to-Binary Routine.

```
$0E01 A2  1F    MODIFY  LDX  #$1F        The multiply routine will not work
                                         without this modification.
```

Listing 5. An Input/Output/Multiply Calling Program.

```
$0050 20 00 0E          JSR  INPUT       Call the BCD to Floating-Point Binary Routine.
 0053 20 B0 0F          JSR  SUB1        Call the subroutine to modify the accumulator.
 0056 20 C0 0F          JSR  SUB2        Transfer ACCA to ACCB (it takes two to multiply),
 0059 20 00 0E          JSR  INPUT       and get the second number.
 005C 20 B0 0F          JSR  SUB1        Fix the accumulator again.
 005F 20 28 0C          JSR  MULTIPLY    Multiply the two numbers using
 0062 20 00 0B          JSR  OUTPUT      Listing 1 in this article. Then
                                         output the result using the Floating-Point
 0065 00                BRK              Binary to BCD Routine.
```

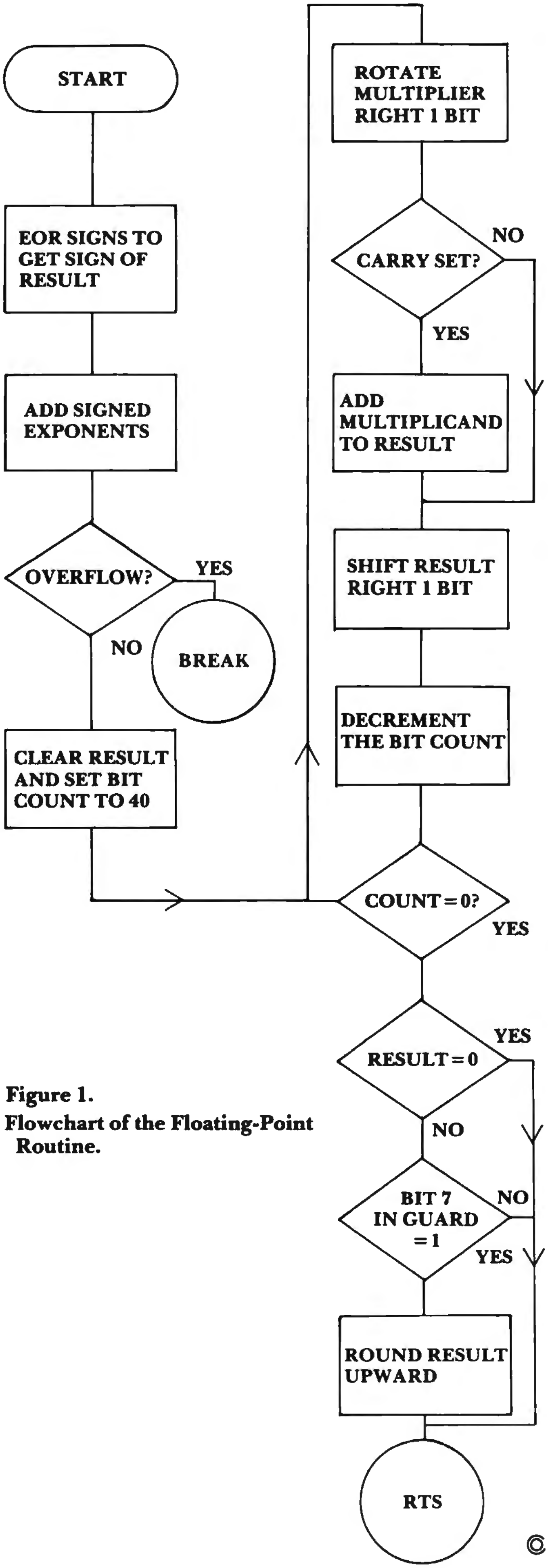
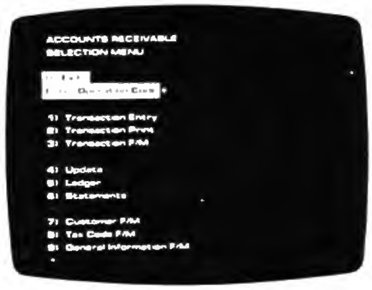


Figure 1.
Flowchart of the Floating-Point Routine.

©

Professional Business Software

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



General Ledger

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

Accounts Payable

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

Accounts Receivable

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

Payroll

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

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

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

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

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

CMS Software Systems

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

Naming Compounds

Tony A. Hartman
Texarkana, AR

Chemistry students seem to have less trouble 're-membering' names of elements and radicals when seated in front of a computer. The prefixes, suffixes and symbols used in nomenclature seem less confusing. Students seem to be able to calculate subscripts faster when challenged by the 'answer machine'. Students begin to rely less on lists of valences and sometimes need not even consult a periodic chart for the proper valences.

Try this program after you have 'hammered away' at valences and 'harped on' using the correct suffix in naming. In this program, answers are typed in exactly as they would be written on paper, except for the placement of subscripts on the screen (on the screen, SUBSCRIPTS are on the same line as the symbol). I think the program can best be utilized after practice and drill on naming compounds and writing formulas. I have found that students working in pairs, carefully selected, have shown the best response. The tendency to 'let the machine answer the hard ones' is lessened when working in pairs.

The following program was written on a PET computer for use in high school chemistry classes. As written, the program uses about 6K of memory. It will run as is on any model PET – original, upgrade, or 4.0 ROM. There are many statements which could be omitted or combined if you are interested in making it more compact.

The elements and radicals used in the compounds are some of the more commonly encountered ones. Students should be familiar with most of the symbols and valences. The names of elements and radicals used in the program can be changed easily as you will see later.

Well, enough of that. I am sure you will find an effective and practical way to use the program. Here is a summary of the program by line numbers:

30-130	Prints title, gives choice of writing names or formulas
140-170	Randomly chooses a name (called from line 880 & 990)
180-200	Delay a few seconds (used in the instructions)
210-250	Prints message and waits for space bar (called throughout)
260-310	Reads data statements
320-450	Compares valences and assigns subscripts
460-510	Displays 'correct' on the screen and increments correct answer counter
520-730	Instructions for writing formulas
740-860	Prints compound name on screen and asks for formula
870-990	Sets number of elements, calls subroutine to choose name and assign subscripts, sets the correct formula
1000-1280	Instructions for writing names of compounds
1290-1430	Uses subroutine 870 to randomly choose a compound
1440-1500	Prints student average and comment
1510-1580	Additional instructions
1590-1650	Comments on scores
1660-1760	Additional instructions
1770-1930	Data statements containing metal groups
1940-2060	Data statements containing nonmetal groups

The following is a summary of the variables used. Hopefully, this will help you to interpret and adapt the program a little easier if that is what you want to do.

c	number of correct answers
e$	name of element
s$	symbol of element
v%	valence of element
e1$	name of metal group
s1%	symbol of metal group
v1%	valence of metal group
e2$	name of nonmetal group
s2$	symbol of nonmetal group
v2%	valence of nonmetal group
n	number of metal/nonmetal ions listed in data statements
f$	formula of compound given by student input
f1$	correct formula of compound calculated by PET
n$	name of compound given by student input
n1$	correct name of compound
l$	line of graphic symbols printed on screen
s1%	subscript of metal group
so%	student score as a percent
t	try (student gets two tries to answer correctly)
x	random number
z%	number read to keep data statement pointer at the right spot
z$	strings read to keep data statement pointer at the right spot

What about personalizing the program? The statements which print the directions can be changed to 'your language'. You can change or take out the delay loop. Change the data statements to include more or different elements or radicals. If you change the number of elements, be sure to change the value of the variable n in line 880 to correspond to the number of metal groups and the value of n in line 990 to correspond to the number of nonmetal groups. Also, changing the comments to your own witty remarks will spark some interest.

One final note. I was reluctant to send an article to a nationally known magazine. I felt less competent than some because of a lack of formal computer training. But I am convinced that for educators to share their ideas on computers, programs and the use of these, we must all put aside our feelings of inadequacy and start sharing what we have. I look forward to seeing more science programs (or any programs for that matter) from you educators who have been holding back!

Editor's Note: Me too! RCL

VARIABLE REFERENCES

c	490	500	530	1450						
e$	280	760	890	1310						
e1$	760	890	1310							
f$	780	790	800							
f1$	800	830	950	960	970	980	1330			
i	40	50	60	70	190	270	300	460		
l$	20	530	610	630	710	730	770	820	840	1010
l$	1090	1100	1180	1190	1270	1320	1340	1390	1410	1460
l$	1470	1490	1510	1570	1670	1750				
n	150	160	290	300	880	900				
n$	1350	1360	1370							
n1$	1310	1370	1400							
q	130	720	760	850	1280	1330	1420			
q$	110	120	130	230	240					
s$	280	890	940	950	960	970	980			
s1	340	390	430	450	930	950	960	970	980	
s1$	890	930	950	960	970	980				
s2	340	390	430	450	940	950	960	970	980	
sc%	1450	1460	1600	1610	1620	1630	1640			
t	490	760	800	820	830	1330	1370	1390	1400	
ti	20									
v%	280	890	910							
v1%	330	360	370	390	410	430	450	890		
v2%	330	360	370	390	410	430	450	910		
x	20	150	160	270	290	300				
z$	300									
z%	300									

LINE REFERENCES

40	1500									
110	110	120								
140	880	900								
150	160									
180	640	650	660	670	680	690	700	1110	1120	1130
180	1140	1150	1160							
210	610	710	840	1090	1180	1270	1410	1500	1580	1750
230	230	240								
260	880	900								
310	290									
320	920									
350	400	440	450							
360	330									
390	370									
410	360									
430	410									
450	380	420								
460	810	1380								
510	490									
520	130									
730	850									
780	790	820								
820	800									
840	810									
870	750	1300								
1000	130									
1300	1420									
1350	1360	1390								
1390	1370									
1410	1380									
1440	860	1430								
1510	720	1280								
1590	1480									
1660	620									

```
10 POKE59468,14
20 L$="(((((((((((((((((((((((((((((((((
   ¬(((((((":X=RND(-TI)
30 REM PRINTS BOX CLOCKWISE
40 PRINT"ĥ";:FORI=1TO39:PRINT"&";:NEXT
50 FORI=1TO23:PRINT"&<↓";:NEXT
60 FORI=1TO39:PRINT"&<<";:NEXT
70 FORI=1TO24:PRINT"&↑<";:NEXT
80 PRINT"h↓↓↓↓"TAB(12)"NAMING COMPOUNDS"
90 PRINT"↓↓"TAB(6)"CHOOSE ONE OF THE ¬
   ¬FOLLOWING:"
100 PRINT"↓↓"TAB(12)"1. WRITE FORMULAS":
   ¬PRINT"↓↓"TAB(12)"2. WRITE NAMES"
110 GETQ$:IFQ$=""THEN110
120 IFQ$<>"1"ANDQ$<>"2"THEN110
130 Q=VAL(Q$):ONQGOTO520,1000
140 REM SUBRTN TO CHOOSE A NAME 0<X<18
150 X=INT(N*RND(1)+0.5)
160 IFX>NORX<=0THEN150
170 RETURN
180 REM SUBRTN TO DELAY A FEW SECONDS
190 FORI=1TO1000:NEXT
200 RETURN
210 REM SUBRTN TO PRINT MESSAGE AND WAIT
220 PRINT"↓PRESS THE rSPACEî BAR TO ¬
   ¬CONTINUE.
```

```
230 GETQ$:IFQ$=""THEN230
240 IFQ$<>CHR$(32)THEN230
250 RETURN
260 REM SUBRTN TO READ NAMES, FORMULAS
270 FORI=1TOX
280 READE$,S$,V%:NEXT
290 IFX=NTHEN310
300 FORI=X+1TON:READZ$,Z$,Z%:NEXT
310 RETURN
320 REM SUBRTN TO COMPARE VALENCE AND ¬
    ¬RETURN SUBSCRIPTS
330 IFV1%<>V2%THEN360
340 S1=1:S2=1
350 RETURN
360 IFV1%>V2%THEN410
370 IFV2%/V1%=2ORV2%/V1%=3ORV2%/V1%=4THE
    ¬N390
380 GOTO450
390 S1=V2%/V1%:S2=1
400 GOTO350
410 IFV1%/V2%=2ORV1%/V2%=3ORV1%/V2%=4THE
    ¬N430
420 GOTO450
430 S2=V1%/V2%:S1=1
440 GOTO350
450 S1=V2%:S2=V1%:GOTO350
460 FORI=1TO20:IFINT(I/2)=I/2THENPRINT"I
    ¬";
470 PRINT"*CORRECT*f
480 PRINT"↑↑":NEXT
490 IFT=1THENC=C+1:GOTO510
500 C=C+0.5
510 RETURN
520 REM WRITE FORMULAS WHEN GIVEN NAME
530 C=0:PRINT"ĥ↓"L$"↓"
540 PRINT"YOU WILL BE GIVEN THE NAME OF ¬
    ¬A COMPOUND
550 PRINT"AND ASKED TO WRITE THE ¬
    ¬FORMULA.  YOU
560 PRINT"↓MUST USE A SPECIFIC FORM IN ¬
    ¬ANSWERING
570 PRINT"↓THESE.  TO WRITE THE ¬
    ¬FORMULAS ON THE
580 PRINT"↓SCREEN, YOU CANNOT USE ¬
    ¬SUBSCRIPTS.
590 PRINT"↓YOU MUST TYPE IN THE ¬
    ¬SUBSCRIPT ON THE
600 PRINT"↓SAME LINE AS THE ELEMENT ¬
    ¬SYMBOL.↓        (INSTRUCTIONS ¬
    ¬CONTINUED)
610 PRINTL$:GOSUB210
620 GOSUB1660
630 PRINT"ĥ↓"L$"↓FOR EXAMPLE, TO WRITE ¬
    ¬THE FORMULA FOR
640 PRINT"↓THE COMPOUND CALCIUM ¬
    ¬CHLORIDE, YOU WOULD↓WRITE:↓":
    ¬GOSUB180:GOSUB180
650 PRINT"C";:GOSUB180:PRINT"A";:
    ¬GOSUB180:PRINT"C";:GOSUB180:
    ¬PRINT"L";:GOSUB180:PRINT"2":
    ¬GOSUB180
660 PRINT"↓FOR THE COMPOUND POTASSIUM ¬
    ¬SULFATE:↓":GOSUB180
670 PRINT"K";:GOSUB180:PRINT"2";:
    ¬GOSUB180:PRINT"S";:GOSUB180:
    ¬PRINT"O";:GOSUB180:PRINT"4":
    ¬GOSUB180
680 PRINT"↓FOR THE COMPOUND ZINC ¬
    ¬NITRATE:↓":GOSUB180
690 PRINT"Z";:GOSUB180:PRINT"N";:
```

```
     ¬GOSUB180:PRINT"(";:GOSUB180:
     ¬PRINT"N";:GOSUB180:PRINT"O";
700 GOSUB180:PRINT"3";:GOSUB180:
     ¬PRINT")";:GOSUB180:PRINT"2":
     ¬GOSUB180
710 PRINT"↓"L$:GOSUB210
720 Q=0:GOSUB1510
730 PRINT"ĥ↓"L$
740 REM SELECT NAME AND ASK FOR FORMULA
750 GOSUB870
760 Q=Q+1:PRINTQ". "E1$" "E$:T=0
770 PRINT"↓"L$"↓"
780 INPUT"FORMULA  _<<<";F$
790 IFF$="_"THENPRINT"↑↑":GOTO780
800 T=T+1:IFF$<>F1$THEN820
810 PRINT"↓":GOSUB460:GOTO840
820 IFT=1THENPRINT"↓WRONG!   TRY ¬
     ¬AGAIN.":PRINT"↓"L$:GOTO780
830 IFT=2THENPRINT"↓WRONG AGAIN!    ¬
     ¬FORMULA IS "F1$
840 PRINT"↓"L$:GOSUB210
850 IFQ<10THEN730
860 GOTO1440
870 REM SUBRTN TO CHOOSE NAME AND FORM
880 N=17:GOSUB140:GOSUB260
890 E1$=E$:S1$=S$:V1%=V%
900 N=13:GOSUB140:GOSUB260
910 RESTORE:V2%=ABS(V%)
920 GOSUB320
930 IFLEN(S1$)>2ANDS1>1THENS1$="("+S1$+"
     ¬)"
940 IF(LEN(S$)>2ANDS2>1)OR(S$="OH"ANDS2>
     ¬1)THENS$="("+S$+")"
950 IFS1<>1ANDS2<>1THENF1$=S1$+RIGHT$(ST
     ¬R$(S1),1)+S$+RIGHT$(STR$(S2),1)
960 IFS1<>1ANDS2=1THENF1$=S1$+RIGHT$(STR
     ¬$(S1),1)+S$
970 IFS1=1ANDS2<>1THENF1$=S1$+S$+RIGHT$(
     ¬STR$(S2),1)
980 IFS1=1ANDS2=1THENF1$=S1$+S$
990 RETURN
1000 REM WRITE NAMES WHEN GIVEN FORMULAS
1010 PRINT"ĥ↓"L$"↓YOU WILL BE GIVEN A ¬
     ¬FORMULA AND YOU
1020 PRINT"↓WILL BE ASKED TO WRITE THE ¬
     ¬NAME OF THE
1030 PRINT"↓COMPOUND.  SPELLING ¬
     ¬DEFINITELY COUNTS.
1040 PRINT"↓SO YOU WILL NEED TO BE ¬
     ¬CAREFUL WITH THE
1050 PRINT"↓ENDINGS SUCH AS 'ITE' AND ¬
     ¬'ATE' AND ALL
1060 PRINT"↓OTHER SPELLINGS AS WELL.  ¬
     ¬TYPE THE NAMES
1070 PRINT"WITHOUT USING CAPITAL ¬
     ¬LETTERS.
1080 PRINT"↓(INSTRUCTIONS CONTINUED)↓
1090 PRINTL$:GOSUB210
1100 PRINT"ĥ↓"L$"↓FOR EXAMPLE, TO WRITE ¬
     ¬THE NAME FOR KCL  ↓YOU WOULD ¬
     ¬WRITE:↓"
1110 GOSUB180
1120 GOSUB180:PRINT"POTASSIUM ";:
     ¬GOSUB180:PRINT"CHLORIDE↓":GOSUB180
1130 PRINT"FOR CU↓2↑O:↓":GOSUB180
1140 PRINT"↓COPPER";:GOSUB180:PRINT"(I) ¬
     ¬";:GOSUB180:PRINT"OXIDE↓":GOSUB180
1150 PRINT"FOR NA↓2↑SO↓4:":GOSUB180
1160 PRINT"↓SODIUM ";:GOSUB180:PRINT"SUL
     ¬FATE↓":GOSUB180
1170 PRINT"(INSTRUCTIONS CONTINUED)↓
1180 PRINTL$:GOSUB210
1190 PRINT"ĥ↓"L$"↓BE SURE TO INDICATE ¬
     ¬MULTIVALENT ELEMENTS
1200 PRINT"WITH THE ROMAN NUMERAL IN ¬
     ¬PARENTHESIS.
1210 PRINT"↓THE ROMAN NUMERAL MUST BE ¬
     ¬IN PARENTHESIS
1220 PRINT"NEXT TO THE METAL IT GOES ¬
     ¬WITH.  USE A
1230 PRINT"↓(I) FOR ONE, (II) FOR TWO,
     ¬ (III) FOR
1240 PRINT"↓THREE, (IV) FOR FOUR AND ¬
     ¬(V) FOR FIVE.
1250 PRINT"↓NOTE THAT THE ROMAN ¬
     ¬NUMERALS ARE CAPITAL
1260 PRINT"LETTERS.
1270 PRINTL$:GOSUB210
1280 Q=0:GOSUB1510
1290 REM SELECT NAME WRITE FORMULA
1300 GOSUB870
1310 N1$=E1$+" "+E$
1320 PRINT"ĥ↓"L$
1330 Q=Q+1:PRINTQ". "F1$:T=0
1340 PRINT"↓"L$"↓"
1350 INPUT"NAME  _<<<";N$
1360 IFN$="_"THENPRINT"↑↑":GOTO1350
1370 T=T+1:IFN$<>N1$THEN1390
1380 PRINT"↓":GOSUB460:GOTO1410
1390 IFT=1THENPRINT"↓WRONG!    TRY ¬
     ¬AGAIN.":PRINT"↓"L$:GOTO1350
1400 IFT=2THENPRINT"↓WRONG AGAIN!    ¬
     ¬NAME IS "N1$
1410 PRINT"↓"L$:GOSUB210
1420 IFQ<10THEN1300
1430 GOTO1440
1440 REM CALCULATE PERCENT & DISPLAY
1450 SC%=C/10*100
1460 PRINT"ĥ↓"L$"↓YOUR AVERAGE IS ¬
     ¬"SC%"%"
1470 PRINT"↓"L$"↓↓"
1480 GOSUB1590
1490 PRINT"↓"L$"↓"
1500 GOSUB210:GOTO40
1510 PRINT"ĥ↓"L$"↓YOU WILL BE GIVEN 10 ¬
     ¬PROBLEMS, ONE AT A
1520 PRINT"↓TIME.  YOU WILL HAVE TWO ¬
     ¬CHANCES TO
1530 PRINT"↓ANSWER CORRECTLY.  IF YOU ¬
     ¬ANSWER CORRECT
1540 PRINT"THE FIRST TIME, YOU GET 10 ¬
     ¬POINTS. IF
1550 PRINT"↓YOU ANSWER CORRECT ON THE ¬
     ¬SECOND TRY,
1560 PRINT"↓YOU GET 5 POINTS.
1570 PRINT"↓"L$
1580 GOSUB210:RETURN
1590 REM COMMENTS FOR SCORE
1600 IFSC%>=90THENPRINT"↓VERY GOOD! YOU ¬
     ¬MAY MAKE A CHEMIST!":RETURN
1610 IFSC%>=80THENPRINT"↓OK! ARE YOU IN ¬
     ¬ENRICHED CHEMISTRY??":RETURN
1620 IFSC%>=70THENPRINT"↓REALLY!! YOU ¬
     ¬CAN DO BETTER THAN THAT!":RETURN
1630 IFSC%>=60THENPRINT"↓COME ON!  DO ¬
     ¬HAVE A CHEMISTRY BOOK??":RETURN
1640 IFSC%>=50THENPRINT"↓YOU WERE ¬
     ¬READING THE QUESTIONS WEREN'T  ¬
     ¬YOU!!!":RETURN
```

```
1650 PRINT"↓DID YOU SIGN UP FOR THIS ¬
     ¬CLASS ALL BY   YOURSELF???":RETURN
1660 REM SUBRTN TO SUPPLEMENT INSTRUCTIO
     ¬NS ON WRITING FORMULAS
1670 PRINT"ħ↓"L$"↓THE FIRST LETTER OF ¬
     ¬THE SYMBOL MUST BE
1680 PRINT"↓CAPITALIZED AND THE SECOND ¬
     ¬LETTER LOWER-
1690 PRINT"CASE AS THEY ARE USUALLY ¬
     ¬WRITTEN.
1700 PRINT"↓WHEN A POLYATOMIC ION WHICH ¬
     ¬ALREADY CON-
1710 PRINT"TAINS A SUBSCRIPT IS TO BE ¬
     ¬SUBSCRIPTED,
1720 PRINT"↓THE ION MUST BE IN PARENTHES
     ¬IS WITH THE
1730 PRINT"↓SUBSCRIPT OUTSIDE.
1740 PRINT"↓(INSTRUCTIONS CONTINUED)
1750 PRINT"↓"L$"↓":GOSUB210
1760 RETURN
1770 DATA HYDROGEN,"H",1
1780 DATALITHIUM,"LI",1
1790 DATASODIUM,"NA",1
1800 DATAPOTASSIUM,"K",1
1810 DATABERYLLIUM,"BE",2
1820 DATACALCIUM,"CA",2
1830 DATAMAGNESIUM,"MG",2
1840 DATABARIUM,"BA",2
1850 DATAZINC,"ZN",2
1860 DATAALUMINUM,"AL",3
1870 DATA"COPPER(I)","CU",1
1880 DATA"COPPER(II)","CU",2
1890 DATA"IRON(II)","FE",2
1900 DATA"IRON(III)","FE",3
1910 DATA"LEAD(II)","PB",2
1920 DATA"LEAD(IV)","PB",4
1930 DATAAMMONIUM,"NH4",1
1940 DATAFLUORIDE,"F",-1
1950 DATACHLORIDE,"CL",-1
1960 DATABROMIDE,"BR",-1
1970 DATAIODIDE,"I",-1
1980 DATAOXIDE,"O",-2
1990 DATASULFIDE,"S",-2
2000 DATASULFATE,"SO4",-2
2010 DATASULFITE,"SO3",-2
2020 DATANITRATE,"NO3",-1
2030 DATANITRITE,"NO2",-1
2040 DATAHYDROXIDE,"OH",-1
2050 DATACARBONATE,"CO3",-2
2060 DATAPHOSPHATE,"PO4",-3
```

©

THE apple® GAZETTE

Using Named GOSUB And GOTO Statements In Applesoft Basic

M. R. Smith

Using subroutines greatly improves the readability of a program and makes it easier to debug. However remembering what a particular GOSUB does is often difficult. Was it GOSUB 1000 or GOSUB 2000 that was wanted?

One of the nice features of Integer Apple Basic is its ability to let you give a name as well as a number in GOSUB statements. The following Integer program demonstrates this:

```
10    GOSUB 100
20    SUB1 =100
30    GOSUB SUB1
40    STOP
100   PRINT "HERE" : RETURN
```

Typing this program whilst using Applesoft will lead to the error message "UNDEFINED STATEMENT IN 30".

The purpose of this program is to show how to use names GOSUB and GOTO statements within Applesoft. By loading the short machine language program described in this article, you are able to run the Applesoft program.

```
10    GOSUB 100
20    SUB1 = 100
30    & GOSUB SUB1
40    STOP
100   PRINT "HERE" : RETURN
```

For the murky details of how it works read the section "PROGRAM DESCRIPTION". Otherwise, type in the demonstration BASIC program and type RUN. The program includes a routine to check that the DATA statements have been entered correctly. Once the demo program has run correctly, the machine language program can be saved using BSAVE NAMED.GOSUB,A$300,L$43. To have the program ready for future sessions, simply type BRUN NAMES.GOSUB as the first part of your programming session. This will load and fix the code. It will remain ready but out of your way until you power down.

WARNING: If you use a RENUMBER program to reorder your program statements, you must remember that variables are NOT changed. Therefore your subroutine pointers will not be renumbered; you'll have to do that by hand.

WARNING: The instructions GOSUB and ON. . .GOSUB are entirely different. The machine code given here will not allow the statement ON X & GOSUB FNAME, SNAME.

Machine Language Program Description

The first statement (at $D93E) of the Applesoft Interpreter GOTO subroutine is the reason that Applesoft does not handle GOSUB's and GOTO's in the same manner as Integer Basic. This statement goes and gets an integer number for use within the GOTO. This means that the BASIC statement GOSUB 1000 is okay but N = 1000 : GOSUB N is not allowed as N as a variable.

Now changing these memory locations to cause the next EXPRESSION to be evaluated, rather than the next NUMBER, allows us to use named GOSUB's. To change these actual locations is impossible. Instead the GOSUB and GOTO routines must be relocated lower in memory at $300 (768) where they can be changed. The Apple's ampersand instruction (&) can then be used to make the new commands operate.

Lines 19–25. Set the ampersand vector (&) at $3F5.

Lines 27–32. Check for GOSUB or GOTO tokens after the &.

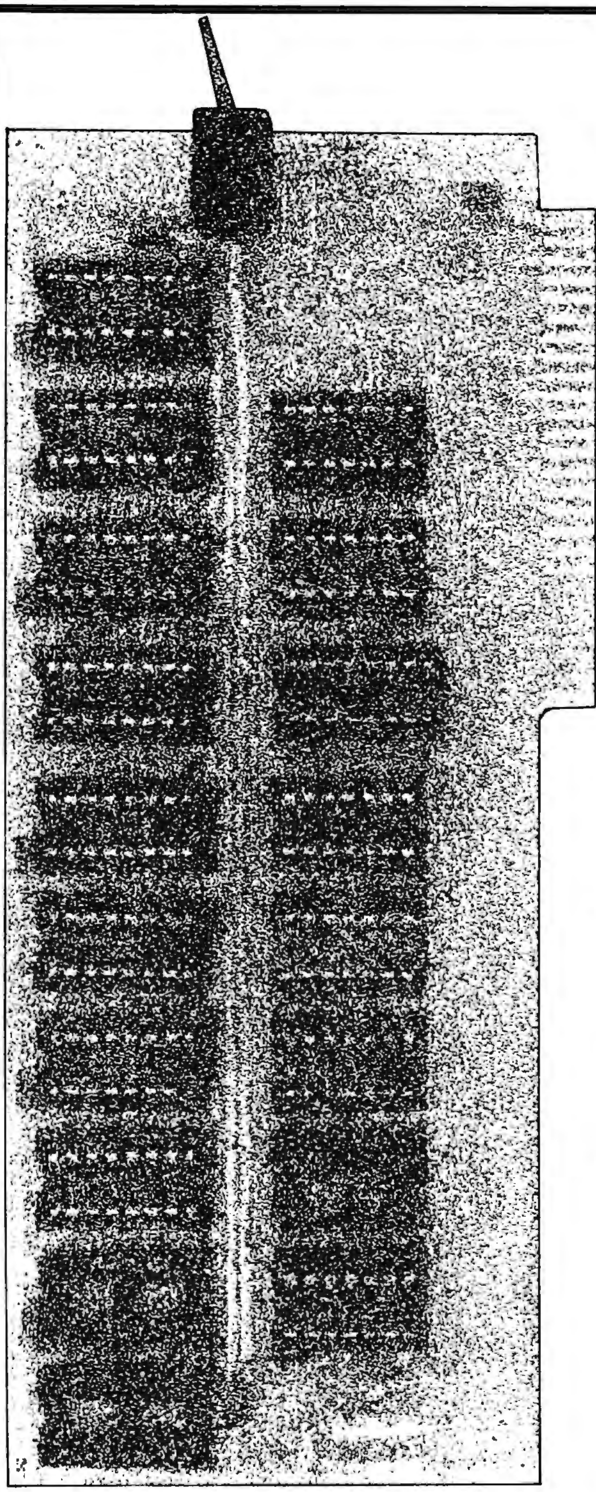
Lines 34–47. Relocated version of the monitor GOSUB routine. This now calls the new front end of the GOTO routine.

Lines 49–52. New front end to drive the monitor GOTO routine. It jumps into the middle of the old GOTO routine.

Lines 50 and **51** are the actual major changes.

BASIC Program Description.

Lines 20 and **5000–5200.** The program first checks that the DATA statements have been correctly entered. Each pair of DATA statements consists of 16 numbers and a checksum which is the previous 16 numbers added together. If this 17th number is not the actual sum of the previus 16 numbers, then an error is indicated. If all the statements are okay, then the code is loaded.



Line 40. Sets the ampersand vector. This is not necessary if the machine code is BRUN into memory but is necessary if the code is BLOADed.
Lines 60–80. These set the subroutine names.
Lines 100–140. Demonstrate the new instructions.
Lines 1000–3020. Demonstration Subroutines.

References

"AMPERSAND-INTERPRETER" by R. M. Mottala in **Nibble #6**, 1980, p27.

"APPLESOFT INTERNAL ENTRY POINTS" by Apple Computer Inc. in **Apple Orchard**, March/April 1980, p12.

"SOME ROUTINES IN APPLESOFT BASIC" by J. Butterfield in **COMPUTE!**, September/October 1980, p68.


```
]RUN
OKAY
OKAY
OKAY
OKAY
OKAY
BLOAD OKAY


THIS IS JOHN
HERE BY A NAMED GOSUB

THIS IS PETE
HERE BY A DIFFERENT NAMED GOSUB

THIS IS PHREDD
HERE BY A NAMED GOTO

BREAK IN 3020
]&FF
```

```
0300-  A9 4C 8D F5 03 A9 10 8D
0308-  F6 03 A9 03 8D F7 03 60
0310-  C9 B0 F0 09 C9 AB F0 1F
0318-  A2 10 4C 12 D4 A9 03 20
0320-  D6 D3 A5 B9 48 A5 B8 48
0328-  A5 76 48 A5 75 48 A9 B0
0330-  48 20 37 03 4C D2 D7 20
0338-  B1 00 20 7B DD 20 52 E7
0340-  4C 41 D9
*
 00 00 00 00 00
*
```

```
10  REM  LOAD THE ROUTINE -
    NORMAL GOSUB
20  GOSUB 5000
30  REM  ESTABLISH THE
    AMPERSAND VECTOR
40  CALL 768
50  REM ESTABLISH NAMES OF SUBROUTINES
60 JOHN = 1000
70 PETE = 2000
80 PHREDD = 3000
90  REM  CALL THE SUBROUTINES
100  &  GOSUB JOHN
110  &  GOSUB PETE
120  &  GOTO PHREDD
130  PRINT "DIDNOT WORK"
140  STOP
1000  PRINT "THIS IS JOHN"
1010  PRINT "HERE BY A NAMED GOSUB"
1020  PRINT : RETURN
2000  PRINT "THIS IS PETE"
2010  PRINT "HERE BY A DIFFERENT
      NAMED GOSUB"
2020  PRINT : RETURN
3000  PRINT "THIS IS PHREDD"
3010  PRINT "HERE BY A NAMED GOTO"
3020  STOP
4990  REM  LOAD IN ROUTINE
5000 LOW = 768:HIGH = 835
5010 OK = 1
5020  REM  LOAD IN GROUP OF SIXTEEN
5030  FOR J = LOW TO HIGH STEP 16
5040 CHECK = 0
5050  FOR K = J TO J + 15
5060  READ IT
5070 CHECK = CHECK + IT
5080  NEXT K
5090  REM  CHECK IF CHECK SUM OKAY
5100  READ SUM
5110 L$ = "OKAY": IF CHECK <  > SUM
     THEN L$ = "BAD":OK = 0
5120  PRINT L$
5130  NEXT J
5140  IF OK = 0 THEN  STOP
5150  REM  THINGS ARE OKAY - LOAD INTO MEMORY
5160  RESTORE : FOR J = LOW TO HIGH STEP 16
5170  FOR K = J TO J + 15: READ IT: POKE K,IT: NEXT K
5180  READ IT: NEXT J
5190  PRINT "BLOAD OKAY": PRINT : PRINT
5200  RETURN
6000  DATA  169,76,141,245,3,169,16,141,246
6010  DATA 3,169,3,141,247,3,96,1868
6020  DATA   201,176,240,9,201,171,240,31,162
6030  DATA 16,76,18,212,169,3,32,1957
6040  DATA 214,211,165,185,72,165,184,72,165
6050  DATA 118,72,165,117,72,169,176,2322
6060  DATA 72,32,55,3,76,210,215,32,177
6070  DATA 0,32,123,221,32,82,231,1593
6080  DATA  76,65,217,0,0,0,0,0,0
6090  DATA  0,0,0,0,0,0,0,358
```

```
SOURCE FILE: GOSUB1
------ NEXT OBJECT FILE NAME IS GOSUB1.OBJ0
0300:              1          ORG   $300
0300:              2 ;
0300:              3 ; M.R.SMITH  APRIL 1981
0300:              4 ;
0075:              5 CLINL    EQU   $75         ;CURRENT LINE NUMBER
0076:              6 CLINH    EQU   $76
00B1:              7 GETCHR   EQU   $B1         ;GET NEXT CHAR
00B8:              8 TXTPTL   EQU   $B8         ;TEXT POINTER
00B9:              9 TXTPTH   EQU   $B9
0300:             10 ;
D3D6:             11 STACK    EQU   $D3D6       ;CHECK ON STACK POINTER
D412:             12 WRONG    EQU   $D412       ;PRINT SYNTAX ERROR MESSAGE
D7D2:             13 NGOSUB   EQU   $D7D2       ;JUMP INTO NORMAL MONITOR GOSUB
D941:             14 NGOTO    EQU   $D941       ;JUMP INTO NORMAL MONITOR GOTO
DD7B:             15 FRMEVL   EQU   $DD7B       ;PUSH VALUE IN FAC
E752:             16 FIXGOTO  EQU   $E752       ;USE FAC AS GOTO POINTER
0300:             17 ;
0300:             18 ;FIX AMPERSAND VECTOR
0300:A9 4C        19 FIX      LDA   #$4C
0302:8D F5 03     20          STA   $3F5
0305:A9 10        21          LDA   #$10
0307:8D F6 03     22          STA   $3F6
030A:A9 03        23          LDA   #$3
030C:8D F7 03     24          STA   $3F7
030F:60           25          RTS
0310:             26 ;
0310:C9 B0        27 ENTRY    CMP   #$B0        ;IS IT GOSUB?
0312:F0 09        28          BEQ   GOSUB
0314:C9 AB        29          CMP   #$AB        ;IS IT GOTO?
0316:F0 1F        30          BEQ   GOTO
0318:A2 10        31          LDX   #$10        ;FORCE SYNTAX ERROR MESSAGE
031A:4C 12 D4     32          JMP   WRONG
031D:             33 ;
031D:A9 03        34 GOSUB    LDA   #$3         ;NORMAL GOSUB PROCEDURE
031F:20 D6 D3     35          JSR   STACK       ;RELOCATED FROM $D921
0322:A5 B9        36          LDA   TXTPTH      ;STORE CURRENT TEXT POINTERS
0324:48           37          PHA
0325:A5 B8        38          LDA   TXTPTL
0327:48           39          PHA
0328:A5 76        40          LDA   CLINH       ;STORE CURRENT LINE NUMBER
032A:48           41          PHA
032B:A5 75        42          LDA   CLINL
032D:48           43          PHA
032E:A9 B0        44          LDA   #$B0        ;IT NEEDS THIS
0330:48           45          PHA
0331:20 37 03     46          JSR   GOTO        ;DO A GOTO
0334:4C D2 D7     47          JMP   NGOSUB      ;CONTINUE NORMAL GOSUB
0337:             48 ;
0337:20 B1 00     49 GOTO     JSR   GETCHR      ;GET NEXT CHAR
033A:20 7B DD     50          JSR   FRMEVL      ;EVALUATE NEXT EXPRESSION
033D:20 52 E7     51          JSR   FIXGOTO     ;FIX GOTO LOCATION
0340:4C 41 D9     52          JMP   NGOTO       ;CONTINUE NORMAL GOTO ROUTINE
```

©

Commas, Colons And Quote Marks Too

Craig Peterson
Santa Monica, CA

Have you ever wanted to be able to input commas, colons or quotation marks as part of an input statement to one of your Applesoft programs? But, hard as you may try, Applesoft kept coming back with "EXTRA IGNORED." Contact 4 from Apple Computer, Inc., helped you by suggesting the use of the GET statement, but all that B$ = B$ + A$ stuff meant that you often had to endure string garbage cleanup delays. Then Contact 6 seemed to offer the ultimate solution, totally avoiding garbage collection. But was it? Besides requiring a small machine language program, there was a subtle problem you might not have been aware of. The input routine used to fill the input buffer made no allowance for the high bit of each character in the input line. The routine used to fill the input buffer left the high bit set, just as it comes from the keyboard. But Applesoft wants the high bit to be zero for its string characters. The line will print correctly and will look on the screen just like what you typed in, but if you ever try an IF IN$ = "Q", you'll never get a match. Or if you try to VAL (IN$), when IN$ was input as "1234", you'll get a value of 0.

The solution to this dilemma is in the program listed below. The subroutine shown in lines 1000 to 1020 (for Applesoft ROM Basic) will gather any input for you and place it into the variable IN$, even commas, colons and quote marks. The only exempt characters are the standard keyboard escape sequences. So, who is the little man at 54572? Well, he's the Applesoft equivalent of the monitor's keyboard input routine, with the difference being that he strips the high bit from all of the input characters. So line 1000 fills the input buffer with normal Applesoft string characters gathered from the keyboard. Line 1010 finds the length of the string, and line 1020 finds the IN$ variable and stuffs its pointers with the right info to point to the keyboard buffer. Then IN$ is relocated into RAM, away from the keyboard buffer. It is not necessary for IN$ to be the first variable used in the program. Lines 1000-1020 can be placed anywhere in your program. The pointers for IN$ are found through the magic of locations 131 and 132, which hold the address of the pointers for the last used variable. It's fast, it totally avoids string garbage build-up, and it's done in Basic. None of that nasty machine language stuff.

One additional note. Not only does this routine work slick for keyboard input, but it also performs the same super feat for disk input, which can be real handy. Commas, etc., in the middle of a name file cause no difficulty when read from the disk. Please note, however, that this routine limits the size of an input string to 239 characters just like the Applesoft INPUT statement does.

So if you need it, try it. It's an easy solution to a common problem.

```
10  HOME : VTAB 4: PRINT "INPUT A
     NYTHING THAT YOU WANT..": PRINT
     : GOSUB 1000: PRINT : PRINT
     "VOILA..": PRINT : PRINT IN$
     : END
20 :
30  REM   LINES 1000 TO 1020 ARE
     A SUBROUTINE THAT PUTS ANY
     INPUT INTO IN$
40 :
1000  CALL 54572
1010  FOR B = 512 TO 751: IF  PEEK
     (B) <  > 0 THEN  NEXT
1020 IN$ = "": POKE  PEEK (131) +
     256 *  PEEK (132) + 1,0: POKE
      PEEK (131) + 256 *  PEEK (1
     32) + 2,2: POKE  PEEK (131) +
     256 *  PEEK (132),B - 512:IN
     $ =  MID$ (IN$,1): RETURN
```

©

Generating Lower Case Text On The Apple II Plus Using The Paymar Chip

David Shapiro
Bloomington, IN

Introduction

The following program will allow lower case text to be displayed on an Apple II Plus which is equipped with a Paymar chip. The hardware requirements involve the "older" Apple with RAM configuration blocks (an "I.C. impersonator" which only contains jumper wires and is labeled with "16K"), and the PAYMAR lower case adapter, presently advertised as the "original LCA-1 (TM)". By appending this routine to a BASIC program, lower case characters can be embedded inside of quotation marks following a PRINT command by simply converting the corresponding upper case character in the given string. When the BASIC statement involving the PRINT command and the string are executed, the display of upper/lower case text is immediate. Lower case characters can also be converted back to upper case using this routine.

Sample Use Of The Lower Case Converter

Once this routine is appended to a BASIC program, it can then be used for converting between upper and lower case characters:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

A typical program statement may contain the string "POUR THE SOLUTIONS." and lower case conversion may be desired on all characters after the "P". The following brief example initially LISTs the statement containing this string, the lower case converter program (which starts at line number 63000) is then RUN, and finally the statement containing the now-converted text is reLISTed.

```
]LIST20
20 PRINT "POUR THE SOLUTIONS."
]RUN63000
WHAT LINE DO YOU WANT CONVERTED? 20
I HAVE FOUND THE LINE.
POUR THE SOLUTIONS.
DO YOU WANT TO CHANGE ANYTHING?
START WITH WHICH CHARACTER? 2
END WITH WHICH CHARACTER? 16
Pour the solutions.
DO YOU WANT TO CHANGE ANYTHING?
]LIST20
20 PRINT "Pour the solutions."
```

The program initially prompts the user for the line number of the BASIC statement to be converted. A search through the Apple's RAM continues until that line number is found, whereupon the characters within quotatin marks are then displayed (if no such line number exists, the program informs the user). A decision to change the string contents is then entered (Y in this case). Character limits for the conversion are individually entered, with only the characters from the upper/lower case sets (see above) sequentially counted (the spaces on either side of "THE" were ignored). The conversion will then start with "0" (the 2nd character) and terminate with the final "S" (the 16th character), with the resultant form displayed for more changes. No further changes were made (input of "N"), and the RESET key was pressed to terminate execution of this routine. This particular statement was then re-LISTed, displaying the quote-embedded lower case text.

More Lower Case Converter Details

The case conversion occurs between the user-defined limits in a continuous fashion. If there are two (or more) separated segments in the same string that are to be converted, then each segment conversion must be done individually. The string is re-displayed after each conversion for further changes if so desired. An individual character can also be converted if the lower and upper numerical limits are identical.

The first time "RUN 63000" is executed, the search for the input line number commences at the beginninng of the program. This search examines the appropriate locations in RAM which the program currently occupies, and with each new examination moves sequentially through the program (increasingly higher memory locations) in an attempt to find the line number. A variable (ML) contains the current RAM location when the line is eventually found. After making the necessary character changes in this statement as stipulated by the user, the search for the next line number will begin at this present memory location (ML). This optimizes the speed with which the program searches for the next line number. If the next line number is less than the last line number, or if it does not exist in the program, then the current RAM location variable ML is re-initialized to zero. The user is informed that the line can not be

found, and the next line number search must start at the beginning of the program. This unnecessarily increases the search time; therefore, for maximum speed-execution of the program, all entered line numbers must exist in the program, and they should be entered by increasing value.

The case conversion between upper/lower case in reciprocal; i.e., designated upper case characters will be converted to lower case, and lower case characters will be changed to upper case. Also, if the cursor is used to read a BASIC statement containing a string, any lower case characters will be converted back to upper case (an easy method for converting a mixed-case string to all upper case).

The line numbering of this routine begins at 63000 since lower line numbers should always be used when writing a BASIC program. It may be entered after the END command and accessed at the user's convenience. Typing "RUN 63000" from the keyboard RUNs the routine; pressing the RESET key will terminate its execution.

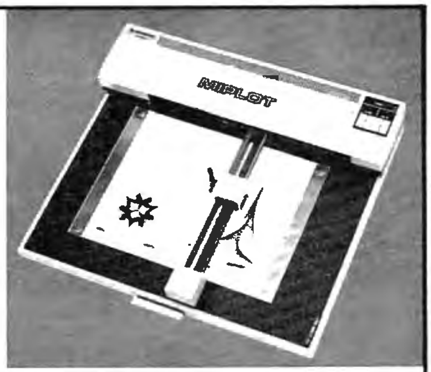
Program Listing And Explanation

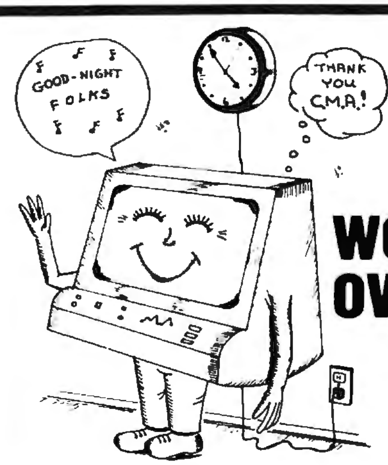
63000 Line number to be converted input as LN.
63010 Initialization of ML to start of BASIC on first RUN of program or when line number is not found; ML is the memory location currently being examined.
63020 NL equated to RAM location of start of next BASIC statement. TL is equated to the line number of BASIC currently being examined.
63030 Jump from search loop if line number is found.
63040 Jump from search loop if line number is not found.
63050 Equate ML to RAM location of the next BASIC statement.
63070 Loop to examine each character/token in the current BASIC statement. Check for quotation mark (ASCII code =34). MODE is a "toggle"; set to 0 when first quote is found.
63080 Printing of characters after 1st quote and up to 2nd quote.
63090 Close PRINT loop.
63100 If no changes ("N") execution transferred to 63000. All other input (including "Y") defaults to 63110.
63110-63120 Limits to define character conversion.
63130 Loop examination of each character/token in BASIC statement. When 1st quote is found, MODE is set to 0.
63140 If the character is between quotes and alphabetic, then counter PO is incremented. When the counter is between the stipulated character limits, the character is converted to upper case (add 32) or lower case (subtract 32) depending on the original value of Q.
63150 Close conversion loop. Control transferred to 63070 for any further changes.

```
63000  INPUT "WHAT LINE DO YOU WA
     NT CONVERTED? ";LN
63010  IF ML = 0 THEN ML = 256 *
      PEEK (104) +  PEEK (103)
63020 NL =  PEEK (ML) + 256 *  PEEK
     (ML + 1):TL =  PEEK (ML + 2)
      + 256 *  PEEK (ML + 3)
63030  IF TL = LN THEN 63060
63040  IF NL < ML OR TL > LN THEN
      PRINT "LINE NOT FOUND.":ML =
     0: PRINT : GOTO 63000
63050 ML = NL: GOTO 63020
63060  PRINT "I HAVE FOUND THE LI
     NE."
63070  PRINT :MODE = 1: FOR A = M
     L + 4 TO NL:Q =  PEEK (A): IF
     Q = 34 THEN MODE = 1 - MODE
63080  IF MODE = 0 AND Q <  > 34 THEN
      PRINT  CHR$ (Q);
63090  NEXT : PRINT
63100  PRINT : PRINT "DO YOU WANT
      TO CHANGE ANYTHING? ";: GET
     A$: PRINT : IF A$ = "N" THEN
     63000
63110  INPUT "START WITH WHICH CH
     ARACTER? ";S
63120  INPUT "END WITH WHICH CHAR
     ACTER? ";E:PO = 0
63130 MODE = 1: FOR A = ML + 4 TO
     NL:Q =  PEEK (A): IF Q = 34 THEN
     MODE = 1 - MODE
63140  IF MODE = 0 AND Q > 64 AND
     Q < 128 THEN PO = PO + 1: IF
     PO >  = S AND PO <  = E THEN
      POKE A,Q + 32: IF Q > 96 THEN
      POKE A,Q - 32
63150  NEXT : GOTO 63070
```
©

Apple Authors

COMPUTE! is looking for applications articles aimed at beginners and intermediate programmers. We're specifically interested in programming hints, tutorials, articles written to help users get more out of their machine.


MIPLOT


GOOD-NIGHT FOLKS
THANK YOU C.M.A.!

THE ATARI GAZETTE

A cure for Atari BASIC or Make Your Atari A Bit Wiser

Charles Brannon

As pointed out by Glen Fisher and Ron Jeffries in "The Ouch in Atari BASIC" (**COMPUTE!**, January/February 1980), the keywords AND and OR in Atari BASIC do not let you "get at" the individual bits of a number, as Microsoft BASIC does. Where PRINT 127 and 64 would give 64 in Microsoft BASIC, the Atari interprets the command as PRINT (not zero) AND (not zero) and returns "1". Although this is fine for logical comparisons (e.g. IF A = 12 and B = 22 THEN PRINT A$), it makes bit hakcers a little angry.

If you do not appreciate why, let me explain. Besides the logical uses of AND and OR, it is often advantageous to use these operands for bit manipulation. This is most important in preparing a byte for a POKE command, or interpreting one that was read with PEEK. Being able to process a number on the binary level gives more "bite" to a computer's number crunching abilities. For example, a major use of the AND operator is to *mask* a number, that is, zeroing out some of the bits in a number. The ASCII value of "3" is 51, or $33 hexadecimal. This looks like %00110011 in binary. If the leftmost four bits (the left nibble) could be cleared, we would have the numerical value of the character "3". The action takes place on the binary level.

	51	= 0011 0011	binary
if we AND with	15	0000 1111	
we get		0000 0011	= 3 in decimal

The AND is performed bit by bit. Refer to the **truth table** for AND. Therefore, the Microsoft BASIC command to mask the left four bits would be:

PRINT 51 and 15

The computer would respond with "3".

The OR operator is commonly used to force bits into a byte. For example: a reverse field character is specified by a one in bit seven (the leftmost one). To force a character to print in reverse field, we just OR its ASCII value with 128.

ASC("A") = 65	=	01000001	binary
if we OR with		1000000	(128)
we get		11000001	193 (reverse field "A")

Once again, refer to the truth table for OR for details.

One other very useful operand is EOR (Exclusive OR). Unfortunately, virtually no BASIC provides this function. It is used commonly to "flip a bit", that is, if a bit is exclusive OR'd with a one, then the opposite bit results. If a number is exclusive OR'd with all ones (255), then the complement is formed.

10101011	171	11000001	193 (reverse "A")
11111111	255	10000000	128
01010100	84	01000001	65 (normal "A")

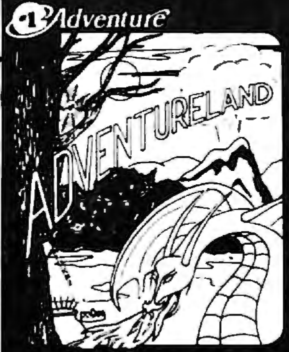
Perhaps now you can see why these operators are so useful. But why am I tormenting you? Didn't I say that Atari BASIC doesn't have this capability? Ah, too true, but once again – machine language comes to the rescue. Listing one is the assembly language program that will simulate the bitwise operators. (For 6502 programmers, notice the sequence CLC, BCC OUT. This will simulate an unconditional jump, yet the code remains relocatable.) Listing two is the BASIC program that will load the program into a protected area of memory. At least I think it is protected. **The Atari BASIC Reference Manual** claims that the area from $600 to $6FF is FREE RAM. If true, then this block of memory could be used like the "second cassette buffer" is used on the PET. When the machine language code is POKE'd here, it should remain there until the power is turned off. Listing three is an example program showing how to use the USR command to call the functions from your programs. It assumes that listing two has already been run. To use the operators in your program, first load the second program. If line 20 is changed to RETURN and the program is appropriately renumbered, then it could be called as a subroutine at the beginning of your program. The machine language program is called by the USR function. This is a truly remarkable command on the Atari, as it can have a variable length list of arguments for the machine language program to deal with. This machine language program uses three arguments. The format is:

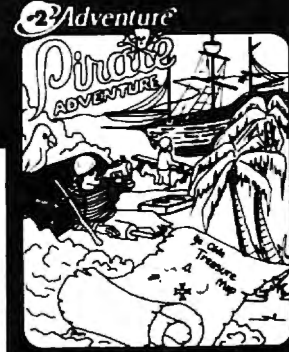
A = USR(ML,avar1,key,avar2)

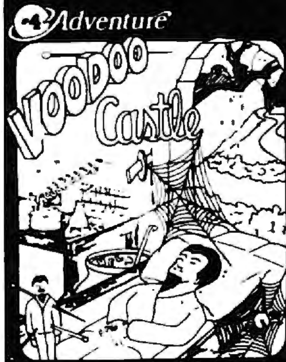
where **ML** is the starting location of the machine language program (1536), **avar1** is the first argument (value 0-255), **avar2** is the second argument

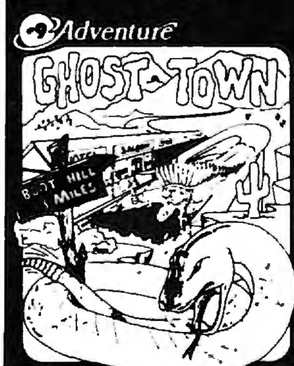
Adventure

BY Scott Adams









JUST RELEASED FOR ATARI

0.✓ **"SPECIAL SAMPLER"** - Never tried ADVENTURE? This special inexpensive sampler complete with 3 Treasures is a cut-down version of our large Adventureland. Guaranteed to supply hours of enjoyment: Try an ADVENTURE today!
ATARI - 24K TAPE to DISK* **051-0000 $ 6.95**

1.✓ **ADVENTURELAND** - You wander through an enchanted world trying to recover the 13 lost treasures. You'll encounter wild animals, magical beings, and many other perils and puzzles. Can you rescue the Blue Ox from the quicksand? Or find your way out of the maze of pits? Happy Adventuring....
ATARI - 24K TAPE to DISK* **051-0001 $14.95**

2.✓ **PIRATE'S ADVENTURE** - "Yo ho ho and a bottle of rum..." You'll meet up with the pirate and his daffy bird along with many strange sights as you attempt to go from your London flat to Treasure Island. Can you recover Long John Silver's lost treasures? Happy sailing, matey. . . .
ATARI - 24K TAPE to DISK* **051-0002 $14.95**

3. **MISSION IMPOSSIBLE ADVENTURE** - Good morning, your mission is to... and so it starts. Will you be able to complete your mission on time? Or is the world's first automated nuclear reactor doomed? This one's well named. It's hard, there is no magic, but plenty of suspense. Good luck. . . .
ATARI - 24K TAPE to DISK* **051-0003 $14.95**

4. **VOODOO CASTLE** - Count Cristo has had a fiendish curse put on him by his enemies. There he lies, with you his only hope. Will you be able to rescue him or is he forever doomed? Beware the Voodoo man. . . .
ATARI - 24K TAPE to DISK* **051-0004 $14.95**

5. **THE COUNT** - You wake up in a large brass bed in a castle somewhere in Transylvania. Who are you, what are you doing here, and WHY did the postman deliver a bottle of blood? You'll love this ADVENTURE, in fact, you might say it's Love at First Byte. . . .
ATARI - 24K TAPE to DISK* **051-0005 $14.95**

6. **STRANGE ODYSSEY** - Marooned at the edge of the galaxy, you've stumbled on the ruins of an ancient alien civilization complete with fabulous treasures and unearthly technologies. Can you collect the treasures and return or will you end up marooned forever?. . . .
ATARI - 24K TAPE to DISK* **051-0006 $14.95**

7. **MYSTERY FUN HOUSE** - Can you find your way completely through the strangest Fun House in existance, or will you always be kicked out when the park closes?. . . .
ATARI - 24K TAPE to DISK* **051-0007 $14.95**

8. **PYRAMID OF DOOM** - An Egyptian Treasue Hunt leads you into the dark recesses of a recently uncovered Pyramid. Will you recover all the treasures or more likely will you join its denizens for that long eternal sleep?. . . .
ATARI - 24K TAPE to DISK* **051-0008 $14.95**

9. **GHOST TOWN** - Explore a deserted western mining town in search of 13 treasures. From rattlesnakes to runaway horses, this ADVENTURE's got them all! Just remember, Pardner, they don't call them Ghost Towns for nothin'. (Also includes new bonus scoring system!)
ATARI - 24K TAPE to DISK* **051-0009 $14.95**

10. **SAVAGE ISLAND — PART 1** - WARNING FOR EXPERIENCED ADVENTURERS ONLY! A small island in a remote ocean holds and awesome secret. Will you be the first to uncover it? NOTE: This is the first of a larger multi-part Adventure;, it will be necessary to purchase additional packages to complete the entire Adventure.
ATARI - 24K TAPE to DISK* **051-0098 $14.95**

✓ RECOMMENDED FOR THE NOVICE ADVENTURER, with many built-in helps

*TAPE to DISK - a standard cassette tape that runs on a tape machine or can easily be transferred to disk.

ai Adventure INTERNATIONAL

A DIVISION OF SCOTT ADAMS, INC.

CATALOG & ORDERING TOLL FREE HOT LINE (800) 327-7172

BOX 3435, LONGWOOD, FL 32750 • (305) 862-6917

ADVENTURE Series also available for TRS-80 Models 1, 2 & 3, APPLE, Pet & Scorcerer....Call Today!

Listing 1

```
                  ;******  BOOLEAN FUNCTIONS
                  ;******    FOR THE ATARI
                  ;
                  WHICH      .DE $D0
                  ARG1       .DE $CB
                  ARG2       .DE $CF
                  RTN        .DE $D4
                  ;
                             .BA $0000
                             .OC
                  ;
0000- 68          INIT       PLA
0001- C9 03                  CMP #$03
0003- D0 2C                  BNE OUT
0005- 68                     PLA
0006- 68                     PLA
0007- 85 CB                  STA *ARG1
0009- 68                     PLA
000A- 68                     PLA
000B- 85 D0                  STA *WHICH
000D- 68                     PLA
000E- 68                     PLA
000F- 85 CF                  STA *ARG2
                  ;
0011- A5 D0                  LDA *WHICH
0013- C9 01       AND        CMP #$01
0015- D0 07                  BNE OR
0017- A5 CB                  LDA *ARG1
0019- 25 CF                  AND *ARG2
001B- 18                     CLC
001C- 90 13                  BCC OUT
                  ;
001E- C9 02       OR         CMP #$02
0020- D0 07                  BNE EOR
0022- A5 CB                  LDA *ARG1
0024- 05 CF                  ORA *ARG2
0026- 18                     CLC
0027- 90 08                  BCC OUT
                  ;
0029- C9 03       EOR        CMP #$03
002B- D0 04                  BNE OUT
002D- A5 CB                  LDA *ARG1
002F- 45 CF                  EOR *ARG2
                  ;
0031- 85 D4       OUT        STA *RTN
0033- A9 00                  LDA #$00
0035- 85 D5                  STA *RTN+1
0037- 60                     RTS
                             .EN

LABEL FILE:  [ / = EXTERNAL ]

/WHICH=00D0       /ARG1=00CB       /ARG2=00CF
/RTN=00D4         INIT=0000        AND=0013
OR=001E           EOR=0029         OUT=0031

//0000,0038,0038
```

Listing 3

```
100 REM SAMPLE PROGRAM
110 GRAPHICS 0:ML=1536
120 SCR=PEEK(560)+256*PEEK(561)+4
130 SCR=PEEK(SCR)+256*PEEK(SCR+1)
140 REM
150 REM DEMONSTRATE "EOR"
160 REM
170 FOR I=0 TO 199
180 A=USR(ML,PEEK(SCR+I),3,128)
190 POKE SCR+I,A
200 NEXT I
210 REM
220 REM DEMONSTRATE "AND" & "OR"
230 REM
240 OPEN #1,4,0,"K:"
250 GET#1,KEY
260 PRINT "NORMAL CHARACTER:";
270 A=USR(ML,KEY,1,127))
280 PRINT CHR$(A)
290 PRINT "REVERSED CHARACTER:";
300 PRINT CHR$(USR(ML,A,2,128)
310 REM
320 REM TEST EACH FUNCTION
330 REM
340 GRAPHICS 0
350 PRINT"(ENTER -1 TO STOP)"
360 PRINT "FIRST VALUE";
370 INPUT ARG1
380 IF ARG1=-1 THEN END
390 PRINT "ENTER FUNCTION:"
400 PRINT "1=AND, 2=OR, 3=EOR"
410 INPUT KEY
420 IF KEY<1 OR KEY>3 THEN 390
430 PRINT "SECOND VALUE";
440 INPUT ARG2
450 PRINT USR(ML,ARG1,KEY,ARG2)
460 PRINT:GOTO 350
READY.
```

Listing 2

```
10 ML=1536:FOR I=0 TO 55:READ X:POKE ML+I,X:NEXT I
20 NEW
30 DATA 104,201,3,208,44,104,104,133,203,104,104,133,208,104
40 DATA 104,133,207,165,208,201,1,208,7,165,203,37,207,24
50 DATA 144,19,201,2,208,7,165,203,5,207,24,144,8,201
60 DATA 3,208,4,165,203,69,207,133,212,169,0,133,213,96
READY.
```

of the function to be performed, and **key** is the code for which operator is being used.

1=AND
2=OR
3=EOR

The USR function MUST supply all four variables (ML,avar1,key,avar2) and in proper order or the Atari will "lock-up". It will not respond to the keyboard, necessitating a power off/on reset to regain control.

I have provided here a machine language program that extends Atari BASIC. It would be very useful if others could submit similar programming aids, particularly a graphics extension to use the player/missile graphics. Let's make the most of the USR function to extend Atari BASIC as far as possible.

Truth Tables

0 AND 0 = 0	0 OR 0 = 0	0 EOR 0 = 0
0 AND 1 = 0	0 OR 1 = 1	0 EOR 1 = 1
1 AND 0 = 1	1 OR 1 = 1	1 EOR 1 = 0

©

Odds And Ends

John Girard
Berkeley, CA

Here is an early routine I figured out for the ATARI that encourages people to play with the many sound possibilities.

HYPER DRIVE SIMULATOR

```
100 PRINT"TONE NUMBER";
      INPUT T                SEE BELOW
110 OPEN#1,4,0,"K:"
120 GET#1,K                  PRESS A KEY TO START
130 FOR I=200 TO 1 STEP-1
140 SOUND 0,I,T,8            SPACESHIP ACCELERATES
150 FOR J=1 TO 5:NEXT J
160 NEXT I
170 SOUND 0,0,0,0            KILL SOUND IN HYPERSPACE
180 GET#1,K                  PRESS A KEY TO FINISH
190 FOR I=1 TO 200
200 SOUND 0,I,T,8            SPACESHIP DECELERATES
210 NEXT I
220 SOUND 0,0,0,0            ENGINES OFF
230 GOTO 120
```

For even more realistic sounds, the volume can be made to rise and fall with the pitch of the engines:

```
140 SOUND 0,I,T,15-INT(I*.05)
200 SOUND 0,I,T,15-INT(I*.075)
delete line 220
```

Each run of the program requests T, a tone number. Giving T a value of 8 produces a satisfactory rushing noise for the engines. Other interesting values are:

10 — a pure tone
4 — damaged engine
12 — bizarre sounding engines

©

Editor's Note: Here are two variations of screen printers for your Atari. Enjoy them. RCL

Copy Your Atari Screen To Your Printer

Harry A. Straw
Wilmington, DE

Here's a handy routine for copying text from your ATARI screen (GRAPHICS 0 mode) to your printer. It is set up to use two GOSUB commands in your main program:

GOSUB 32010 to initialize.

GOSUB 32040 each time you want to line-print a page displayed on your screen.

The program is straightforward, but a few comments may help you to run it smoothly.

The main business of this program is the double FOR-NEXT loop in lines 32050–32110. With the POSITION command, these loops move the cursor over the entire screen, one position at a time. At each cursor position, line 32080 GETs the ASCII number for the character under the cursor, and line 32090 puts the corresponding character on the printer. Since I have an 80-column printer and the ATARI screen is only 40 characters wide, I need line 32105 to get printer carriage return at the proper place. You may be able to delete this line if you have a 40-column printer (or one that can be set to 40 columns).

Line 32040 (printer carriage return) makes sure that the printer head starts copying at its left-hand margin. Line 32120 "homes" the cursor at the end of the subroutine. This is not always necessary but, depending on the next line in your main program, it may prevent an ERROR - 141, "cursor out of range."

You must OPEN a port to GET from the screen. I use port no. 5, leaving ports 1-4 free for use in main programs. The initializing subroutine in lines 32010–32030 does this. It also expands the ATARI display to its full 40-character width and 24-line height to match the cursor movement controlled by lines 32050 and 32060. The OPEN command clears the screen, so you must OPEN before displaying the text you want to copy. Just be sure your main program says GOSUB 32010 *ahead* of the screen display to be printed.

If you have only a few lines to copy, no problem. Merely adjust line 32050 to cover the rows you want to scan. Otherwise, the printer will run for all 24 rows, printing a lot of blank spaces wherever nothing shows on the screen.

There is no CLOSE no. 5 statement in the listing. This leaves port no. 5 open so it is not necessary to repeat GOSUB 32010 for each page to be line-printed.

Take advantage of ATARI's ability to merge cassette-recorded programs with RAM-resident programs by recording this routine with the LIST"C command and reading the cassette with ENTER"C. CSAVE and CLOAD won't work this way. In fact, CLOAD erases programs in RAM! This routine starts with a high line number, 32000, so its line numbers won't conflict with those of a program already in RAM.

In a future note, we'll discuss copying graphics to a printer.

```
32000 REM - COPY SCREEN TO PRINTER.
32001 REM
32002 REM - "OPEN" CLEARS SCREEN.
32003 REM - DO THIS EARLY IN PROGRAM.
32004 REM - USE "GOSUB 32010" FOR THIS.
32005 REM
32010 POKE 82,0:POKE 83,39
32020 OPEN #5,4,0,"S:"
32030 RETURN
32031 REM
32032 REM - USE GOSUB 32040 TO LPRINT
32033 REM - TEXT FROM SCREEN.
32034 REM
32040 LPRINT CHR$(10)
32050 FOR Y=0 TO 23
32060 FOR X=0 TO 39
32070 POSITION X,Y
32080 GET #5,G
32090 LPRINT CHR$(G);
32100 NEXT X
32105 LPRINT CHR$(13)
32110 NEXT Y
32120 POSITION 0,0
32130 RETURN
```

©

Screen To Printer

Len Lindsay

Here is a simple program, completely in BASIC that will print what is on your screen to your printer. It is designed for the 40 column printer. Thus it can only print 39 characters per line, since printing the 40th character creates an extra line feed. To change to 40 characters per line you can change the 39 in line 32130 to 40.

The program is meant to be used as a subroutine. It depends on these two lines occuring at the beginning of the program first:

20 DIM XC$(39)

40 OPEN #3,4,0 "S:"

Note that the program is reading characters right off the screen. Screen input of this type can be used within other types of programs.

Finally, note that the ATARI printer will not print all the characters as on your screen. Often it will just print a blank space for a character it can't print.

Listing

```
0 REM PRINT SCREEN TO PRINTER
1 REM (C) 1980 LINDSAY
20 DIM XC$(39)
40 OPEN #3,4,0,"S:"
32100 XC$="
          ":REM PRINT SCREEN
32101 REM XC=CHARACTER READ FROM SCREEN
AS ASCII VALUE
32102 REM XLOOP=COL LOOP VARIABLE
32103 REM YLOOP=ROW LOOP VARIABLE
32104 REM XC$=LINE OF CHARACTERS FROM SC
REEN
32105 REM ** INCLUDE A DIM XC$(39)
32106 REM ** INCLUDE THESE AT START
32110 FOR YLOOP=0 TO 23
32120 POSITION 1,YLOOP
32130 FOR XLOOP=1 TO 39
32140 GET #3,XC
32150 XC$(XLOOP,XLOOP)=CHR$(XC)
32160 NEXT XLOOP
32170 LPRINT XC$
32180 NEXT YLOOP
32199 RETURN
```

Sample Output

```
FILENAME IS: DIRPRINT.1
     003 SECTORS
FILENAME IS: DIRPRINT.2
     005 SECTORS
FILENAME IS: DRFACTOR.
     068 SECTORS
FILENAME IS: PRINT.DRF
     009 SECTORS
FILENAME IS: TEST.HST
     001 SECTORS
FILENAME IS: DRFACTOR.HST
     001 SECTORS
FILENAME IS: MENU.
     023 SECTORS
FILENAME IS: PREVHIGH.
     001 SECTORS
FILENAME IS: LEN.HST
     001 SECTORS
FILENAME IS: SCREEN.PRT
     005 SECTORS
FILENAME IS: ROBERT.HST
     001 SECTORS
 SECTORS FREE =527
```

©

Hardware Information At Last!

Richard Bills
Lisle, IL

For those Atari owners who have been tormented by the inadequacy of information concerning the hardware and other technical aspects, relief is finally here. Three manuals are now available:

Atari 400/800 Technical User Notes (C016555)
Cost: $27.00 plus $3.00 shipping
Includes information on the OS and other hardware as well as schematic diagrams

Atari 400/800 Operating System (OS) (CA016557)
Cost: $17.00 plus $3.00 shipping
Includes machine language routines that reside in the OS

Atari 400/800 Disk Operating System (DOS) (C016558)
Cost: $4.00 plus $1.50 shipping
Includes machine language routines that are used in DOS

Both OS and DOS can be ordered for $24.00 (price includes shipping). To order a manual enclose a check and letter stating which manual you want to:

Atari Inc.
Customer Support
1346 Bordeaux Drive
Sunnyvale, CA 94086
Attn: Tom Harris

©

Using Strings For Graphics Storage

Michael Boom
Spokene, WA

If you've ever been frustrated attempting to PLOT and DRAWTO your way through a complex pattern or design in Atari Graphics, you might appreciate a method of graphics generation using text strings to store pixel data. While this string method is not simpler to use in all cases, its ease of data entry and manipulation possibilities make it a strong graphics tool.

Simple line drawings over large areas of the screen are best done using PLOT and DRAWTO commands, since this method uses less memory and generates images faster than the string method will. However, if you have a very complex pattern in a small area of the screen, the string method works well. The heart of string graphics lies in the fact that if you run a PRINT #6 statement followed by ASCII characters while in Graphics Modes 3-7, colored pixels will appear on the screen. Different letters and symbols will plot different colors, but for our purpose we will deal only with the letters A, B, C, and D. Each of these letters plots a different colored pixel in Graphics modes 3, 5, and 7:

A plots color 1 (color register #0)
B plots color 2 (color register #1
C plots color 3 (color register #2)
D plots color 0 (color register #4)

In Graphics modes 4 and 6, only the letters A and B need be used, A for the plotting color, B for the background color.

For a demonstration, if you type the command

GRAPHICS 3: PRINT #6; "ABCDA"

moves the pixel string down and to the right.

Creating A Graphics String:

We can now use the above methods to plot a pattern. First graph out the area needed for the pattern, then fill in the pattern using "A", "B", "C", and "D" to represent the colors wanted:

```
String 1   CDDDDAAAAA
String 2   DCDDDDDDAA
String 3   DDCDDDDADA
String 4   DDDCDDADDA
String 5   DDDDCADDDA
String 6   AAAAACDDDD
String 7   ABBBADCDDD
String 8   ABCBADDCDD
String 9   ABBBADDDCD
String 10  AAAAACCCCC
```

Now break down the graph as a series of strings, in this case 10 strings of 10 characters each:

String 1 is "CDDDDAAAAA"
String 2 in "DCDDDDDDAA"
etc.

Concatenate the 10 strings for more efficient data storage:

"CDDDDAAAAADCDDDDDDAADDCDDDDADADD DCDDADDADDDDCADDDAAAAAACDDDDABB BADCDDDABCBADDCDDABBBADDDCDAAAA ACCCCC"

We have now generated all the data necessary to plot our figure (a square with an arrow) in the graphics mode, and have stored it in one long string

Display

To plot the string on the screen, determine where you would like the upper left hand corner of the figure to be located, and enter it during the run of the following program after prompt "X,Y?"

```
10  GRAPHICS 5
20  DIM A$(100)
30  $="CDDDDAAAAADCDDDDDDAADDCDDDD
      ADADDDCDDADDADDDDCADDDAAAAAA
      CDDCDDDABCBADDCDDABBBADDDCDAA
      AAACCCCC"
40  PRINT "X,Y";:INPUT X,Y
80  FOR K= 1 TO 10
90  POSITION X,Y+K -1
100 PRINT #6; A$(K*10-9,K*10)
110 NEXT K
```

In this program, lines 20 and 30 set up our main pixel data string and line 40 establishes the upper left corner coordinates of the figure. Lines 80 and 110 set up a loop of 10 steps, to divide our main data string into 7 rows. Line 90 positions the cursor for each row, and line 100 prints 10 consecutive 10 character strings on the screen.

Obviously there are figures which require strings too long for direct entry in Atari Basic. In that case, divide the figure into several rectangular sections, each small enough for inclusion into one string (usually under 100 characters in length.) Then concatenate the string as explained in the Basic Reference Manual, p. 39.

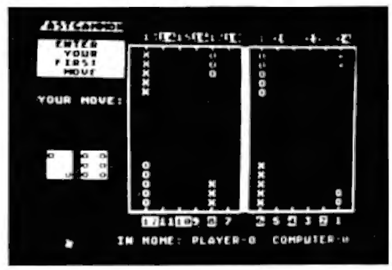
Figure Manipulation:

Plotting a figure using strinng graphics is fairly simple and straightforward. Its real strength lies in figure manipulation through string reading. Some easy manipulations are:

1. **Figure rotation (in 90° implements)**
2. **Figure inversion**
3. **Color changes**

For figure rotation, using the same example figure and data string, let's substitute and add to the previous program. For a 90 degree turn clockwise, add and substitute:

SOFTWARE FOR THE ATARI 800* AND THE ATARI 400*



TARI TREK™
By Fabio Ehrengruber

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

Cassette - $11.95   Diskette - $14.95

FASTGAMMON™
By Bob Christiansen

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

On cassette only - $19.95

TANK TRAP
By Don Ursem

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

Cassette - $11.95   Diskette - $14.95

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

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

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

On diskette only - $79.95

* * * * * * * * * *

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

On cassette only - $24.95

* * * * * * * * * *

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

Cassette - $11.95   Diskette - $14.95

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

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

*Indicates trademarks of Atari, Inc.

```
 20 DIM A$(100),B$(100)
 50 FOR K = 1 TO 10: FOR L = 1 TO 10
 60 B$(K*10–10+L,K*10–10+L)=A$((10–L)*10+K,
    (10–L) *10+K)
 70 NEXT L, NEXT K
100 PRINT #6;B$(K*10–9,K*10)
```

For a 270 degree clockwise rotation, substitute:

```
 60 B$(K*10–10+L,K*10–10+1)+A$(L*10+1–K,L*
    10+1–K
```

For a 180 degree clockwise rotation, substitute

```
 50 FOR K = 1 TO 100
 60 B$(K,K)=A$(101–K,101–K)
 70 NEXT K
```

To change color assignments, add and substitute to the *original* program:

```
 50 FOR K=1 to 100
 60 IF A$(K,K)="C" THEN A$(K,K)="A"
 70 NEXT K
```

To invert a figure, substitute to the original program:

```
100 PRINT #6; A$((11–K)*10–9,(11–K)*10)
```

To turn a figure left to right, substitute in the 180 degree rotation program:

```
100 PRINT #6; B$((11–K*10–9,(11–K)*10)
```

The string manipulations used to manipulate this 10x10 figure can easily be incorporated into subroutines for use in programs using repetitive figures in different positions. Further experimentation for more possibilities is definitely in order.

I hope that the method of string graphics is handy and useful for those of you interested in Atari graphics. Good luck with them. ©

Atari Machine I/O

Charles Brannon

There are three routines that will be of interest to ATARI machine language programmers.

Location $F6E2 waits for a key to be pressed, and will return its ASCII value in the accumulator. (Works like GET# in BASIC)

Location $F6A4 puts the character in the accumulator on the screen in the next print location. (Works like PUT#6) The X and Y registers are altered by this routine.

The INPUT routine at $F63E is a little trickier. It will input a line from the screen and keyboard, just like the INPUT statement does in BASIC. It does not store the line anywhere, however. To use it, do a JSR $F63E to get each character of the line. The character will be returned in the accumulator. Check for end of input by comparing the value to 155, the ATASCII value of the RETURN key. You must store the values in memory to save the input. Since the X and Y registers are altered by this routine, you have to save them if you are using them before you call the routine. The program at the end of this article demonstrates this.

Quick Reference

```
GETCHAR  $F6E2
OUTCHAR  $F6A4
INPUT    $F63E
```

Finally, I warn you that although these addresses work on my ATARI, they might be different on yours.

```
INPUT   LDX #0          ;initialize loop counter
NEXT    STX SAVEX       ;save it
        JSR $F63E       ;get a character
        LDX SAVEX       ;restore index
        STA STRING,X    ;save character
        INX             ;increment counter
        CMP #9B         ;is accumulator = 155
                          (RETURN)?
        BNE NEXT        ;if not, continue
        RTS             ;Finished
```

```
OUTPUT  LDX #0          ;initialize loop counter
NXT     STX SAVEX       ;save it
        LDA STRING,X    ;fetch a character from memory
        JSR $F6A4       ;print it
        LDX SAVEX       ;restore index
        INX             ;increment it
        CMP #$9B        ;accumulator = 155 (RETURN)?
        BNE NXT         ;if not, continue
        RTS             ;Finished
```

©

Disk Directory Printer

Len Lindsay

If you have an Atari disk, you know that you can see its directory by entering DOS and choosing option A. Well, here is a program I wrote completely in ATARI BASIC that will give you the same directory listing. Then a second program is listed that will give you an "expanded" directory.

The key to this program is being able to open the directory as a file for a READ. This is easily accomplished with the following statement.

100 OPEN #1,6,0,"D1:*.*"

Next you must know how the file name info is stored in the directory. The file info is stored as a string 17 characters long.

The first character tells if the file is locked or not. If it is "*" then it is locked. If it is " " (space) then it is not locked.

The file name comes next. Characters 3-10 are the file name. Characters 11-13 are the extension for the name. Any unused characters are stored as spaces. Note, however, that you can't imbed the spaces in your name when you access the file.

Characters 15-17 are the number of sectors used by the program.

With that info you can see how the second, expanded directory list, works. You now can read the directory within your programs by following the new simple methods shown.

Listing 1

```
0 REM PRINT DIRECTORY
1 REM *** (C) 1981
2 REM *** LEN LINDSAY
3 REM ***
4 REM *** SAME AS VIA DOS
10 GRAPHICS 0
20 DIM FILENAME$(20)
100 OPEN #1,6,0,"D1:*.*":REM OPEN DIRECT
ORY FOR A READ
110 TRAP 900:REM NO MORE FILES
200 INPUT #1;FILENAME$
300 PRINT FILENAME$
400 PRINT LEN(FILENAME$)
800 GOTO 200
900 END
```

Listing 2

```
0 REM PRINT DIRECTORY
1 REM *** (C) 1981
2 REM *** LEN LINDSAY
3 REM ***
4 REM *** EXPANDED DIRECTORY PRINT
10 GRAPHICS 0
20 DIM FILENAME$(20)
100 OPEN #1,6,0,"D1:*.*":REM OPEN DIRECT
ORY FOR A READ
110 TRAP 900:REM NO MORE FILES
200 INPUT #1;FILENAME$
300 IF LEN(FILENAME$)<5 THEN 900
400 PRINT "FILENAME IS: ";
410 FOR LOOP=3 TO 13
420 IF LOOP=11 THEN PRINT ".";
430 IF FILENAME$(LOOP,LOOP)<>" " THEN PR
INT FILENAME$(LOOP,LOOP);
440 NEXT LOOP
450 IF FILENAME$(1,1)="*" THEN PRINT "
 LOCKED ";
460 PRINT
500 PRINT "     ";FILENAME$(15,17);" SEC
TORS"
800 GOTO 200
900 PRINT " SECTORS FREE =";FILENAME$
```

©

Condensing Data Statements On The Atari

Craig Patchett

This article was originally written as an appendum to my article "Designing Your Own Atari Character Sets" (see the March 1981 issue of **COMPUTE!**). It then occured to me, however, that there are most likely many other applications where this simple technique might be useful, especially in the loading of machine language subroutines from BASIC DATA statements. In general, any program where a significant amount of numbers between 0 and 255 must be stored as data can be reduced in size using the technique.

An Atari memory location, as is true with most microcomputers, can only hold numbers in the range of 0 to 255. Not by coincidence, 0 to 255 is also the range of ATASCII values, each of which can be translated to an Atari character using the CHR$ function. On the same note, each Atari character can be translated to its ATASCII value using the ASC function. This means that one character can be used in place of from one to three digits. Since characters can be combined in character strings, one character can replace up to three digits *and* a comma when used in place of its corresponding value in DATA statements. Therefore, in programs that use a lot of numerical data in the 0 to 255 range, character strings can be utilized in the following way to cut down the program's memory requirements:

```
30000 REM /*Make sure we're not at the e
nd of the current string*/
30010 IF ME=LEN(DAT$) THEN ME=0:READ DAT
$
30020 REM /*Increment ME (pointer into D
AT$)*/
30030 ME=ME+1
30040 REM /*Convert next character to it
's ATASCII value*/
30050 VALUE=ASC(DAT$(ME,ME))
30060 REM /*All done*/
30070 RETURN
```

To use this subroutine, first DIMension DAT$ to the length of the longest data string you plan to use, and initialize ME to 0. Then, each time you would normally use a READ command, use a GOSUB 30000 instead and the data value will be returned in VALUE. Of course, you must first convert your data to the appropriate Atari characters. Appendix C: ATASCII Character Set, in the BASIC Reference Manual, can be used to aid in this task. Keep in mind that, for the most part, ATASCII values 128-255 are just the reverse of values 0-127 (in other words, use the reverse character key). The ‹ESC› key, in combination with other keys, can often be used to get the more evasive characters. To make life a little easier for you, I've included this short program that will print out the ATASCII values of any characters typed while it is running. Good luck!

```
10 OPEN #1,4,0,"K:"
20 GET #1,VALUE
30 PRINT VALUE
40 GOTO 20
```

(Note: to get the ATASCII value of a character such as ‹ECS›‹CTRL›+ using this program, just type ‹CTRL›+. Pressing the ‹ESC› key will give you the value of the ‹ESC›‹ESC› character.) ©

Real-Time Clock On The Atari

Richard Bills
Lisle, IL

As the popularity of the Atari Computer grows, more people are realizing that it offers more capabilities than other computers in the same price range. Many of its capabilities, however, are not advertised. For instance, I would not have known that it had real-time clock hardware if my dealer had not told me about it. I have since developed this flexible 3K program to utilize this hardware.

The program will first ask you if you want to set the alarm time. If you do, it will ask you to give the time in twenty-four hour format (for example, 15,30,20). Otherwise it will disable the alarm. Next, it will ask you if you would like to set the time. If you do, it will ask for the hours, minutes, and seconds, and will enter this time into the hardware registers. You may use twenty-four hour time if you wish. If you don't want to set the time, the time presently in the hardware registers will not be changed. After fulfilling these preliminaries, the clock time is then displayed in the center of the screen. The time is stored and kept in the hardware and should not be disturbed unless you hit SYSTEM RESET. You may have noticed that this program uses large line numbers (near the 32,767 limit). This enables you to attach this program to the end of another program (or several programs) as a subroutine. I suggest using LIST "C" to save the program and ENTER "C" to load the program. These commands allow you to enter the program without erasing the program that resides in memory. LIST "C", X,Y will list lines X through Y to the cassette, enabling you to save a certain routine without including the clock program. A line by line description of this program follows.

30000	Let's clear the screen and shut off the cursor.
30001–30002	OFF is a flag. When it equals 1 then the alarm will not go off.
30003	TOTAT is the total alarm time in sixtieths of a second
30010–30016	These lines input the time which is to be placed in the hardware registers.
30020–30021	Register 53279 is the register which indicates which console button(s) are pressed. It equals 6 when START is pressed.
30025	This line POKE's the clock hardware down to 0. The largest number a register can have is 255. Register 20 increments by 1 every sixtieth of a second and increments register 19 by 1 when it counts beyond 255 (back to 0 again). Register 18 increments by 1 when register 19 counts beyond 255.
30030–30049	Now we break the current time down into sixtieths of a second and store them in the hardware registers.
30100	This collects the time from the hardware registers in sixtieths of a second.
30150	If the time in the registers is greater than 24 hours, lines 31000–
31070	will be executed. They bring the time in the registers down to an equivalent time below the 24 hour level. This allows the time to continue to be kept in the hardware for an indefinite period of time by preventing all the registers from counting beyond the 255 level and going to 0 at the same time; this would cause the time to be lost.
30522	This line can be eliminated if 24 hour time is preferred.
30523	The time is obtained from the registers in order to compare it at line 30526 to the time the alarm was set to go off at. Since the program is too slow to be able to check the alarm time continuously, a tolerance (100) may be changed.
30524	This line may also be eliminated if 24 hour time is preferred.
30530	This is the line which produces the alarm sound. Use your imagination here!
30539–30700	The printing of the time is performed by these lines. They insure that the zeros will be correctly placed and that the length of the line will always be the same.

```
30000 PRINT "}":DIM X$(10):POKE 752,1
30001 ? "Do you want to initialize the a
larm";:INPUT X$:IF X$="YES" OR X$="yes"
THEN OFF=0:GOTO 30003
30002 OFF=1:GOTO 30004
30003 PRINT "Set alarm time [use 24 hour
 time in 0,0,0 format]":INPUT AH,AM,AS:T
OTAT=AH*60*60*60+AM*60*60+AS*60
30004 ? "Do you want to set the time";:I
NPUT X$
30005 IF X$="YES" OR X$="yes" THEN 30007

30006 GOTO 30099
30007 ? "}"
30010 PRINT "Hours";:INPUT H
30015 PRINT "Minutes";:INPUT M
30016 PRINT "Seconds";:INPUT S
30020 PRINT "Hit START to begin the time
"
30021 IF PEEK(53279)<>6 THEN 30021
30022 PRINT "}"
30023 REM ******* PUT CURRENT TIME IN HA
RDWARE REGISTERS*******
30025 POKE 18,0:POKE 19,0:POKE 20,0
30030 T=H*60^3+M*60^2+S*60
30040 POKE 18,INT(T/(256*256))
30043 T=T-(256*256)*(INT(T/(256*256)))
30045 POKE 19,INT(T/256)
30047 T=T-256*(INT(T/256))
30049 POKE 20,INT(T)
30099 ? "}"
30100 TIME=PEEK(20)+PEEK(19)*256+PEEK(18
)*256*256
30150 IF TIME>=5184000 THEN 31000
30200 TIME=INT(TIME/60+0.5)
30300 SEC=TIME-60*(INT(TIME/60))
```

```
30350 TIME=INT((TIME-SEC)/60)
30400 MIN=TIME-60*(INT(TIME/60))
30500 HOURS=INT((TIME-MIN)/60)
30505 IF SEC>=60 THEN 30510
30508 GOTO 30515
30510 MIN=INT(SEC/60)+MIN
30511 SEC=SEC-60*(INT(SEC/60))
30515 IF MIN>=60 THEN 30520
30518 GOTO 30522
30520 HOURS=INT(MIN/60)+HOURS
30521 MIN=MIN-60*(INT(MIN/60))
30522 IF HOURS=0 THEN HOURS=12
30523 ATCHECK=PEEK(18)*256*256+PEEK(19)*
256+PEEK(20)
30524 IF HOURS>12 THEN HOURS=HOURS-12
30525 SOUND 0,0,0,0
30526 IF ABS(ATCHECK-TOTAT)<100 AND OFF=
0 THEN 30530
30527 GOTO 30539
30530 ? " ":SOUND 0,50,10,10:FOR X=0 TO
1000:NEXT X:? "}}}}}}}"
30539 POSITION 15,10
30540 IF HOURS<10 THEN 30550
30542 IF MIN<10 THEN 30630
30544 IF SEC<10 THEN 30700
30545 PRINT INT(HOURS+0.5);":";INT(MIN+0
.5);":";INT(SEC+0.5):GOTO 30100
30550 IF MIN<10 THEN 30560
30551 GOTO 30600
30560 IF SEC<10 THEN PRINT "0",INT(HOURS
+0.5);":0";INT(MIN+0.5),":0";INT(SEC+0.5
):GOTO 30100
30561 PRINT "0";INT(HOURS+0.5),":0";INT(
MIN+0.5);":";INT(SEC+0.5):GOTO 30100
30600 IF SEC<10 THEN PRINT "0";INT(HOURS
+0.5);":",INT(MIN+0.5),":0";INT(SEC+0.5)
:GOTO 30100
30601 PRINT "0",INT(HOURS+0.5);":";INT(M
IN+0.5),":";INT(SEC+0.5):GOTO 30100
30630 IF SEC<10 THEN PRINT INT(HOURS+0.5
);":0";INT(MIN+0.5),":0";INT(SEC+0.5):GO
TO 30100
30631 PRINT INT(HOURS+0.5),":0",INT(MIN+
0.5);":";INT(SEC+0.5):GOTO 30100
30700 PRINT INT(HOURS+0.5);":",INT(MIN+0
.5);":0";INT(SEC+0.5):GOTO 30100
30900 REM The next lines will poke the h
ardware clock registers down 24 hours
31000 TIME=PEEK(18)*256*256+PEEK(19)*256
+PEEK(20)
31005 TIME=TIME-5184000*(INT(TIME/518400
0))
31020 POKE 18,INT(TIME/(256*256))
31030 TIME=TIME-(256*256)*INT(TIME/(256*
256))
31040 POKE 19,INT(TIME/256)
31050 TIME=TIME-256*(INT(TIME/256))
31060 POKE 20,INT(TIME)
31070 GOTO 30100
```

©

Review Stud Poker

Robert W. Baker
Atco, NJ

STUD POKER is an interesting card game program for the 16K Atari from Dynacomp, Inc., 6 Rippingale Road, Pittsford, NY 14534. ($11.95, cassette; $15.95, diskette) The program includes two separate menu selectable versions of familiar stud poker, each with simple graphics and some sound effects. The card displays are simply the card outline with the face value and suit, no fancy card displays are used. For sound, you get to hear the cards shuffled and dealt along with other appropriate "bells and whistles" at important times.

One of the games deals two cards to you and the Atari, with one card down for the Atari. You each bet on your hands, and bet again after each of the remaining three cards are dealt. At each betting interval you can call, bet/raise from $1 to $3, or fold. The current pot value and your current winnings or loses are always displayed. When the hand is over, the Atari's down card is turned over and the winner is declared.

The other game is even simpler, both you and the Atari are each dealt five cards. Two of the Atari's cards are face down and not displayed. You must bet on your hand ($10 to $100) and cannot fold. After betting, the Atari's down cards are turned over and the winner is declared. Again, your total winnings or loses are displayed.

The games are rather interesting and it would appear that the Atari's card playing skills are pretty good. However, the documentation supplied was rather confusing and did not match the program operation. The names of the two games as well as the betting limits were different in the manual from that used in the program. Also, a different method of indicating whether to continue or quit was used by each part of the program after each hand. One section wanted a "C" or "Q" while the other wanted a RETURN with a null or "Q" input. Totally confusing! With a little more consistency and clearer documentation this could be a very nice package. ©

THE OSI® GAZETTE

Through The Fill-The-Buffer Routine

With Gun And Camera

Kerry Lourash
Decatur, Illinois

This is an effort to shed some light on the Fill-the-Buffer routine (FTB) of OSI BASIC-in-ROM. Subroutines with FFXX addresses are for the C1P, but should be about the same for the C2P. Let me warn you - all numbers in this article are hexidecimal, unless stated otherwise! I will appreciate any corrections or additions readers may have.

What Is It?

The buffer mentioned is a section of zero-page memory (locations 13-5A). When you type in a line of BASIC or the tape recorder loads your favorite program the computer stores one BASIC line at a time in the buffer. Since the buffer is only 72 (decimal) bytes long, no BASIC line can be longer than 72 (dec.) characters. By the way, when you type a 4-digit line number, you have only 68 (dec.) characters left in the line. The FTB takes input from the keyboard or ACIA (Asynchronous Communication Interface Adapter), depending on the status of the SAVE and LOAD flags. After the line is stored in the buffer, other routines tokenize the line and store it in the BASIC workspace.

What Does It Do?

This is what the FTB does:

1. Filters input so no graphics or control characters except "BEL" (end of line) and NULL (zero) gets into the buffer.
2. Checks the "CTRL 0" (output) flag (loc. 64) to see if characters should be output to TV and ACIA.
3. Counts the number of characters input and gives an automatic carriage return/line feed (CR/LF) if the line length stored in loc. 0F is exceeded.
4. Outputs ten NULLS after a CR, and an additional number of NULLS equal to that stored in loc. 0D after a LF.
5. Implements control characters such as carriage return (0D), line feed (0A), "BEL" (07), backspace (5F), and line delete (@,40).
6. Puts a NULL in the buffer at the end of a line to mark the end of line for following routines. Sets the X and Y registers to the start of the buffer(-1).

Preparing For Our Journey

Machine language routines are murder to decipher, and the FTB is no exception. The code is compact in order to stuff BASIC into 8K of ROM, and uses nested subroutines extensively. In my chart, I've put the subs immediately after the point where they are called, instead of in numerical order. Also, subs are indented and bracketed, so the addresses at the far left are the main routine and the subs are at the right, in brackets. The format is somewhat like the outlines we did in school. I've tried to make the routine understandable to both machine language and BASIC oriented readers. The ML addresses have been kept so any part of the routine can be pinpointed and disassembled for additional info; BASIC readers can consider the addresses as line numbers. Most assembly language has been replaced by explanations of what is happening. I have used only a few mnemonics and have given their BASIC equivalents in the heading of the chart.

Into The Jungle

Now we're thru the preliminaries, on with the safari! Look for line A357 on the chart; this is our starting point. First, the X register is zeroed. The x-reg. counts characters as they are input into the buffer. Through a series of JSR's(JSR = GOSUB) and JMP(GOTO) thru RAM, we come to the input sub at FFBA. For those who have the Aardvark cursor program, this is where it steps in and does its stuff. Locations 218 and 219 are changed so that BASIC jumps to the Aardvark program instead of FFBA.

The Input Trek

The input sub looks at loc. 203, the LOAD flag. If the MSB (Most Significant Bit) of 203 is zero, the sub goes to FD00, the keyboard scan sub, which waits for an input from the keyboard, decodes it, puts it in the A register, and returns (RTS) to A389. On the other hand, if the MSB of loc. 203 is 1, the sub checks the LSB (Least Significant Bit) of F000, the ACIA's status register, and waits 'til it is zero, which means the ACIA has a byte ready in F001. This byte is stored in the A-reg. and the routine returns to A389, just like the keyboard routine does. Oh yes, one thing I forgot to mention: before F000 is

SOFTWARE FOR OSI

OSI OSI

A JOURNAL FOR OSI USERS!!

The Aardvark Journal is a bimonthly tutorial for OSI users. It features programs customized for OSI and has run articles like these:

1) Using String Variables.

2) High Speed Basic On An OSI.

3) Hooking a Cheap Printer To An OSI.

4) An OSI Disk Primer.

5) A Word Processor For Disk Or Tape Machines.

6) Moving The Disk Directory Off Track 12.

First year issues already available!
$9.00 per year (6 issues)

ADVENTURES

Adventures are interactive fantasies where you give the computer plain English commands (i.e. take the sword, look at the control panel.) as you explore alien cities, space ships, ancient pyramids and sunken subs. Average playing time is 30 to 40 hours in several sessions. There is literally nothing else like them — except being there yourself. We have six adventures available.

ESCAPE FROM MARS — Explore an ancient Martian city while you prepare for your escape.

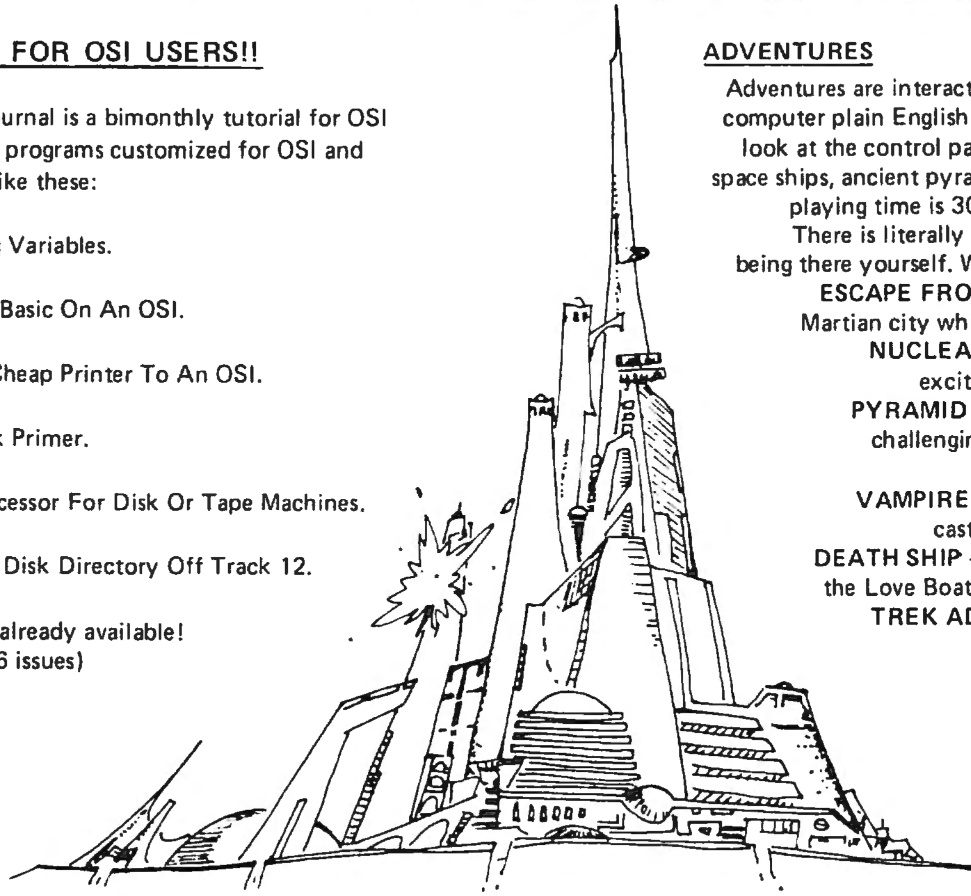
NUCLEAR SUBMARINE — Fast moving excitement at the bottom of the sea.

PYRAMID — Our most advanced and most challenging adventure. Takes place in our own special ancient pyramid.

VAMPIRE CASTLE — A day in old Drac's castle. But it's getting dark outside.

DEATH SHIP — It's a cruise ship — but it ain't the Love Boat and survival is far from certain.

TREK ADVENTURE — Takes place on a familiar starship. Almost as good as being there.



NEW SUPPORT ROMS FOR BASIC IN ROM MACHINES

C1S — for the C1P only, this ROM adds full screen edit functions (insert, delete, change characters in a basic line). Softwave selectable scroll windows, two instant screen clears (scroll window only and full screen), software chose of OSI or standard keyboard format, Bell support, 600 Baud cassette support, and a few other features. It plugs in in place of the OSI ROM. NOTE: this ROM also supports video conversions for 24, 32, 48 or 64 characters per line. Replaces video swap tape on C1P model 2. All that and it sells for a measly $39.95.

C1E/C2E for C1/C2/C4/C8 Basic in ROM machines. This ROM adds full screen editing, software selectable scroll windows, keyboard correction (software selectable), and contains an extended machine code monitor. It has breakpoint utilities, machine code load and save, block memory move and hex dump utilities. A must for the machine code programmer replaces OSI support ROM. Requires installation of additional chip when installed in a C2 or C4. C1 installation requires only a jumper move. Specify system $59.95.

DISK UTILITIES

SUPER COPY — Single Disk Copier
This copy program makes multiple copies, copies track zero, and copies all the tracks that your memory can hold at one time — up to 12 tracks at a pass. It's almost as fast as dual disk copying. — $15.95

MAXIPROSS (WORD PROCESSOR) — 65D polled keyboard only - has global and line edit, right and left margin justification, imbedded margin commands, choice of single, double or triple spacing, file access capabilities and all the features of a major word processor — and it's only $39.95.

P.C. BOARDS

MEMORY BOARDS!! — for the C1P. — and they contain parallel ports!
Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

PROM BURNER FOR THE C1P — Burns single supply 2716's. Bare board — $24.95.

MOTHER BOARD — Expand your expansion connector from one to five connectors or use it to adapt our C1P boards to your C4/8P. - $14.95.

ARCADE AND VIDEO GAMES

GALAXIA one of the fastest and finest arcade games ever written for the OSI, this one features rows of evasive, hardhitting, dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. — P.S. The price is a giveaway. SPECIFY SYSTEM!
Cassette $9.95 — Disk $12.95

TIME TREK (8K) — real time Startrek action. See your torpedoes move across the screen! Real graphics — no more scrolling displays. $9.95

INTERCEPTOR C1P ONLY! An all machine code program as fast and smooth as the arcades. You use your interceptor to protect your cities from hordes of enemy invaders. A pair of automatic cannons help out, but the action speeds up with each wave of incoming ships. The fastest and most exciting C1P game yet.
C1P Cassette $19.95

MINOS — A game with amazing 3D graphics. You see a maze from the top, the screen blanks, and then you are in the maze at ground level, finding your way through on foot. Realistic enough to cause claustrophobia. — $12.95

SINGLE STEPPER / MONITOR

This is probably the finest debugging tool for machine code ever offered for OSI systems. Its' trace function allows you to single step through a machine code program while it continuously displays the A, X, Y and status registers and the program and stack pointers. You can change any of the registers or pointers or any memory loca- at any time under program control. It takes well under 1k and can be relocated anywhere in free memory. It is a fine tool for all systems — and the best news of all is the extremely low price we put on it. — Tape $19.95 — Disk $24.95

FOR DISK SYSTEMS — (65D, polled keyboard and standard video only.)
SUPERDISK. Contains a basic text editor with functions similar to the above programs and also contains a renumberer, variable table maker, search and new BEXEC* programs The BEXEC* provides a directory, create, delete, and change utilities on one track and is worth having by itself. — $24.95 on 5" disk - $26.95 on 8".

AARDVARK IS NOW AN OSI DEALER!

Now you can buy from people who can support your machine.

— THIS MONTH'S SPECIALS —

Superboard II	$279
C1P Model II	429
C4P	749
8K 610 board for C1P	269
Epson MX-80 printer with RS232 installed	595

. . . and we'll include a free Text Editor Tape with each machine!

True 32X32 Video Mod Plans for C1P
(4 Chips $3.00 Crystal Required)
$7.95

VISA

This is only a partial listing of what we have to offer. We now offer over 100 programs, data sheets, ROMS, and boards for OSI systems. Our $1.00 catalog lists it all and contains free program listings and programming hints to boot.

master charge

OSI

Aardvark Technical Services • 1690 Bolton • Walled Lake, MI 48088
(313) 669-3110

OSI

checked, the keyboard is checked to see if the space bar has been hit. If so, the LOAD flag is turned off and we JMP to FD00 and then RTS to A389.

Now we have a byte, but we're not done processing it yet. At A389-A396 there is a section of code that tells the CPU to do nothing for a few microseconds. I'm not sure whether this is a time delay or just a spot where some code was deleted and the gap not closed up. Anyone know? After this lull, the MSB of the input byte is set to zero so we don't get any graphics characters and if the char. is a CTRL 0(0F) the output flag (loc. 64) is toggled. That means the output flag is changed to FF (all 1's) if it is zero, and vice versa. Finally, the input processing is completed and we RTS to the main routine at A35C.

Character Runs The Gauntlet

At A35C the character is tested to see if it is a "BEL". If it is, the X-register is checked to see if the buffer is full (more than 71 dec.). If there is room in the buffer, "BEL" is stored in the buffer and sent to the output sub A8E5 (more on this sub later). At A381 we are sent back to A359 to get another character. If the buffer is full, the "BEL" is output to the TV (or ACIA, if doing a SAVE) by A8E5, but "BEL" is not stored in the buffer. Now we are back at A359.

Let's temporarily bypass the test for carriage return (A360) and look at A364. This test blocks out control and graphics characters and sends us back to A359. That's why there's no way to stick a graphics char. directly into a line, even in a PRINT statement, without a CHR$ command. Look in your graphics manual and see what characters are legal (20-7D).

At A36C we test for @, the line erase character. We branch to A351 and JSR to A8E5 (outputs the @ character). Then a JSR to A86C, which sends a CR and a LF to A8E5, sending the cursor to "home". Now an RTS to A357 to zero the buffer counter, and we are back at A359, ready to start filling the buffer again. A370 tests for "SHIFT 0". Oddly enough, the ASCII of "SHIFT 0" happens to be 5F, which is also the cursor character. This time we branch back to A34B. A JSR to A8E5 outputs a cursor character. A34E decrements the buffer counter (X), and if we haven't erased backward beyond the start of the buffer, A34F sends us to ol' A359. If we have erased too far, a JSR to A86C homes the cursor, A357 zeroes the buffer counter, and we start filling the buffer at A359.

At A376 the buffer counter is checked. If the buffer is full, the input char. is changed to "BEL" (A37C) and output (A8E5) to tell you you're wasting your time. Nothing is stored in the buffer and we branch to A359 for another journey thru the FTB. Finally at A378, the character, if it has passed all the tests, is stored in the buffer. The contents of the buffer counter (X) are added to the number 13 (start of the buffer) and the character is stored at the resulting address. A37A increments the buffer counter, counter and A37E JSR's to A8E5, which prints the character.

The A8E5 Routine

Now for an explanation of the A8E5 sub. If the MSB of the output flag (loc. 64) is a 1, we RTS with no output to TV or ACIA.

If the MSB is zero, we check to see if the ASCII of the char. is less than 20 (BEL, CR, LF). If so, we skip the line length check and branch to A8FA. At A8FA we JSR to FFEE, which JMPs to the address found in 021A and 021B. This address is normally FF69, but you could cook up your own routine and put its starting address in 021A and 021B. From FF69, we JSR to BF2D, the video output sub, which I will explain in another article. To make a long story short, a "BEL" will be displayed as a graphics character, a CR will cause the cursor to be moved to the start of the line, and a LF will scroll the screen and "home" the cursor.

Now we RTS from the video sub and check the status of the SAVE flag (205). If 205 contains a zero, we RTS to A901. If the SAVE flag is non-zero the ACIA status register is monitored until its second bit is zero and then the character is sent to the ACIA (loc. F001). If the character is a CR then 10 (dec.) NULLs are also sent to the ACIA (this gives the computer time to process the line and scroll the screen when the program is LOADed from tape) and then we RTS to A901. A901 RTS's to A381 which brings us back to A359.

Back at A8EA, we assumed the input character would be less than 20. Let's see what happens if it's greater than 20. At A8EE addresses 0E and 0F are compared. 0E is the counter for the number of characters since the last CR. 0F contains the user-selectable line length (remember the "terminal width?" message at cold start?).

Don't confuse this line length with the maximum line length for the video stored at FFE1 or the cursor position counter at loc. 0200. If 0E and 0F are equal then there is a JSR to A86C. At A86C a CR and anLF are fed to the A8E5 sub for an automatic LF/CR. At A87A an additional number of NULLs equal to the number stored in loc. 0D are output. If 0E and 0F aren't equal there is a branch to A8F7 and 0E is incremented before the JSR to FFEE. The character is output to the TV and, if the SAVE flag is on, to the ACIA. Finally, we return to A359.

Last Leg of Our Journey!

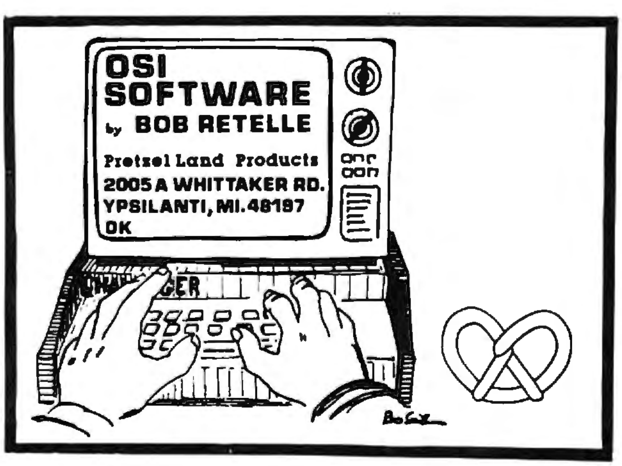
Have patience, our journey is almost at an end. We skipped over the CR test at A360, now let's go through that one. If the input is a CR, a branch is made to A86C which puts a NULL at the end of the line in the buffer, marking the end of the line. This done, we are at A86C, which starts the auto CR/LF and the extra NULLs from loc. 0D. When we reach the end of the sub at A88A we RTS not to the FTB but to the Tokenize-the-Buffer routine, which is another story.

I highly recommend both Carlson's *OSI Basic In ROM* and William's and Dorner's *First Book of OSI*. The information in their books was invaluable in writing this article. I would like to hear from other people interested in Basic-in-ROM.

Fill-The-Buffer Routine (A357)

```
             JSR – GOSUB
             RTS – RETURN
      BRANCH,JMP – GOTO
             INC – ADD 1 (TO)
             DEC – SUBTRACT 1 (FROM)
         /02180/ – CONTENTS OF (LOC. 0218)
            CHAR – ASCII CHARACTER
             MSB – MOST SIGNIFICANT BIT
             LSB LEAST SIGNIFICANT BIT
ALL NUMBERS IN HEX:
A34B JSR   A8E5
     A8E5  (SEE A8E5 BELOW)
     .....
     A901  RTS
A34E DEC   X-REG. (BUFFER COUNTER)
A34F IF X  GREATER THAN ZERO THEN A359
A351 JSR   A8E5
     A8E5  (SEE BELOW)
     .....
     A901  RTS
A354 JSR   A86C
     A86C  (SEE BELOW)
     .....
     A88A  RTS
A357 ZERO  X-REGISTER (BUFFER COUNTER = 0)
A359 JSR   A386
     A386  JSR FFEB
           FFEB JMP /218,219/ (NORMALLY FFBA)
           FFBA IF LOAD FLAG OFF, BRANCH TO FFD8
           FFBF IF SPACE BAR HIT, BRANCH TO FFD5
           FFCB IF ACIA NOT READY, BRANCH TO FFBF
           FFDI LOAD CHAR FROM ACIA AND RTS
           FFD5 TURN OFF LOAD FLAG
           FFD8 JMP TO FDOO (KEYBOARD SCAN SUB)
           FDOO (RETURNS WITH CHAR. IN A-REGISTER)
           FDCE RTS
     A389  TIME DELAY?
     A396
     A397  MASK MSB OF CHAR. TO ZERO
     A399  IF CHAR, IS "CNTRL 0" TOGGLE OUTPUT
           FLAG (0064)
     A3A5  RTS
A35C  IF CHAR. IS "BEL" (END OF LINE), BRANCH TO A376
A360  IF CHAR. IS CARRIAGE RETURN,    BRANCH TO A866
A866  PUT A NULL AT END OF LINE IN THE BUFFER (THIS SUB ALSO SETS X REGISTER & Y-REGISTER TO POINT
        AT BUFFER
      FOR GET-CHAR. SUB)
A86C  (SEE BELOW)
A88A RTS   GO TO TOKENIZE BUFFER ROUTINE-THE END.
A364  IF CHAR. IS LESS THAN 20 OR GREATER THAN 7D THEN A359
A36C  IF CHAR. IS @ (ERASE LINE) THEN A351
A370  IF CHAR. IS 5F (BACKSPACE, SHIFT 0) THEN A34B
A376  IF LINE LENGTH IS GREATER THAN 71(DEC.) THEN A37C
A378  STORE CHAR. IN BUFFER
A37A  INC X-REG. (BUFFER COUNTER) AND GOTO A37E
A37C  CHANGE A-REG. (CHAR. INPUT) TO "BEL"
A37E JSR   A8E5
     A8E5  IF OUTPUT FLAG(0064) IS ON, RTS (NO OUTPUT)
     A8EA  IF CHAR. IS LESS THAN 20(BEL, CR, LF)
           BRANCH TO A8F9
           CHARS ALLOWED PER LINE, JSR A86C
           A86C PUT CR IN A-REG. (TO BE OUTPUT)
           A86E PUT CR IN ACCRESS 0E
           A870 JSR A8E5
                A8E5
                .....
                A901 RTS
           A873 PUT LF IN A-REG.
           A875 JSR A8E5
                A8E5
                .....
                A901 RTS
           A87A OUTPUT NO. OF NULLS IN ADDRESS OD
           A886 ZERO ADDRESS OE (NO. OF CHARS. SINCE CR)
           A88A RTS
      A8F7 INC  0E
      A8FA JSR  FFEE
           FF69 JSR BF2D
                BF2D VIDEO OUTPUT ROUTINE
                .... (THIS WILL BE EXPLAINED
                NEXT INSTALLMENT.)
                BF72 RTS
           FF6D IF SAVE FLAG /0205/ IS OFF, RTS
           FF73 JSR FCB1
                FCB1 IF STATUS REG.(f000) OF ACIA
                NOT READY, THEN FCB1
                FCBA WRITE CHAR. TO ACIA (F001)
                FCBD RTS
           FF76 IF CHAR WAS NOT A CR, RTS
           FF7D WRITE 10(DEC.) NULLS TO ACIA
           FF8A RTS
      A901 RTS
   A381  BRANCH TO A359
```

©



FOOTU: FOO Revisited

A Game For The OSI C1P, or how we learned the true meaning of the oft used phrase "This program is easily adapted to..."

Charles M. and Michael J. De Santis

On p. 26:45 of the July 1980 issue of **MICRO**, the "small systems jornal" from Ohio Scientific listed a little race program called "FOO". It was stated that the program would run on disk based OSI machines but that "the program is easily adapted to" OSI basic-in-ROM machines. Well, maybe its easy if you're one of OSI's computer designers or software whizzes and know where all the goodies are tucked away inside all the OSI computers, but my son Mike and I had one devil of a weekend getting "FOO" to run on our diskless — C1P. However, I can't say it was a bad experience because we learned a lot about our little machine and have come up with a couple of things that should be of interest to other C1P owners as well.

A Carriage Control

For instance, did you know that SPC (0) when used in a PRINT statement causes about 15 line feeds to occur. We discovered this one while trying to figure out why the roadway on OSI's version of "FOO" would space out and break up occasionally (see their line 550).

Keyboard Control Routine

After that was corrected, our next problem was to get the vehicle in the game moving under keyboard control. We found that, for some reason that we didn't want to take a lot of time to discover, the subroutine starting at line 600 of the OSI version of the game wouldn't work on the C1P as the program was originally written.

To correct this problem, we just re-wrote the subroutine using the "more standard" format from the OSI graphics manual, i.e. POKE 57088, row #: IF PEEK (57088) = col. # THEN ...etc. However, our keyboard control software evolved into a form that we think is really useful for many other programs.

In the typical game program as in "FOO", numbers, i.e. number keys, are used to control the direction of an object on the video screen, e.g. "1" for movements to the left and "2" for movements to the right. A problem with this approach usually crops up at the end of a game if, for instance, an INPUT statement is used to query the user about continuing. If the player isn't fast enough (he's just been controlling a space ship and has crashed into a star at 30,000 mi/hr.) he enters a "1" or "2" where a "Y" or an "N" was expected, and he has to fuss around to correct the entry or restart the program if he's already hit the RETURN. The more insidious version of this problem arises when the "keyboard-control-during-program-execution" feature is turned off while you're still holding down the "1" or "2" key. This situation usually arises abrubtly because of a game rule violation of some sort. The game stops and control returns to BASIC. This happens so fast that you're still holding down one of the number control keys, and BASIC interprets this to be the entry of a program line number. If you type anything else and then hit the RETURN you've just added a new line to your program; and you won't know it until the next time you try to run it. My favorite error in this regard ends up with line 1 reading: 1 LIST. When the program is run, I get a listing.

Well here's how to fix things so that the problem never happens again. First of all, don't use numbers for control functions (obvious, right?); we've used the left and right shift keys for control for several reasons: (a) they're spaced a nice distance apart for hand control; and (b) they're both accessible using the same row number in the keyboard polling routine.

Secondly, and this is where the serendipity comes in, the SHIFT LOCK key must be released in order for the SHIFT keys to be activated since it is also accessed through the same row number. In our version of "FOO", after all of the game options are selected, we use instructions such as:

```
270 PRINT "TO START, RELEASE SHIFT LOCK"
271 POKE 57088, 254: IF PEEK (57088) = 254 THEN
    270
```

The "254" is the column number of the SHIFT LOCK key on the polled keyboard so that line 271 keeps getting repeated until the SHIFT LOCK is released. As soon as it is released, the game starts and the shift keys are active. If the game should end abruptly or unexpectedly, and keys that may have been pressed are not entered because the RETURN key is inactive while the SHIFT LOCK key is not depressed.

The SHIFT LOCK must be pressed in order for BASIC to respond. At the end of the game or at any intermediate INPUT statement, we print a reminder to "PUSH THE SHIFT LOCK" for the proper data entry or to restore normal operation. It's a great way to do it! Try it, you'll like it.

FOOTU — C1P Version

Listing 1 is our version of FOO modified to run on our C1P which has 4k of memory. Some of the scaling factors of the original program have been eliminated and the SHIFT and SHIFT LOCK features discussed in this article are employed. The display has been scaled to fit the C1P's capabilities. For other machines, lines 110, 230, 240, 290 and

520 may have to be modified. Also lines 600 – 660 will have to be modified for C2 and C4 computers. Just remember . . . . "This program is easily adapted to . . ." Good Luck!

```
FOOTU
 100 POKE 530, 1
 110 KY=57088:SM=2:MS=1:RN=0
 115 BS=54051
 117 ML=0:SN=255
 120 LP=5
 130 PL=2/LP
 155 KP=0
 160 IF A$="Y" THEN ME=EM:WI=WF:GU=UG:
     GOTO 270
 170 FOR I=1 TO 30:PRINT:NEXT I
 180 PRINT "FOOTU"
 190 PRINT:PRINT"RACEWAY"
 200 PRINT:PRINT"YOU RUN AT YOUR OWN
     RISK!"
 210 PRINT:PRINT"LEFT=LEFT SHIFT RIGHT=
     RT SHIFT"
 215 PRINT:PRINT"OVERDRIVE=CTRL"
 230 PRINT: INPUT"INITIAL WIDTH (1-20)";WI
 240 PRINT:INPUT"DELAY(1-15)";ME
 241 EM=ME
 245 PRINT
 250 GU=0:INPUT"PEDESTRIANS (Y/N)";X$:
     IF LEFT$(X$,1)="Y"THEN GU=.3
 260 KP=0: INPUT "KILLER FOO (Y/N)"; X$:
     IF LEFT$(X$,1)="Y" THEN PK=1
 270 PRINT: PRINT "TO START PRESS SHIFT
     LOCK"
 271 POKE KY, 254: IF PEEK (KY)=254 THEN 271
 280 FOR I=1 TO 30: PRINT:NEXT I
 290 WD=WI:WF=WI: WI=(12-WI) /2
 291 ME=54060-ME*32
 300 FOR M=1 TO LP; GOSUB 600: GOSUB 500:
     ML=ML+1: NEXT M
 350 WI=WI-1
 370 IF WI 4 THEN 300
 400 FOR M=1 to LP: GOSUB 600: GOSUB 500:
     ML=ML+1: NEXT M
 450 WI=WI+1
 470 IF WI<WD THEN 400
 490 GOTO 300
 500 RN=RN+SM*RND (1)–MS
 510 WT=WT+SGN(RN)
 520 IF WI+WT>20 THEN WT=WT-1: RN=0
 530 IF WT<0 OR WT=0 THEN WT=WT-1: RN=0
 540 IF WI<0 THEN WI=2
 545 IF WI>8 AND RND (1)<GU THEN POKE
       BS+WT+1+INT (WI*RND(1) ),  240
 550 PRINT SPC (WT); "XX"; SPC (WI); "XX"
 560 RETURN
 600 POKE Y, 254
 610 IF PEEK (KY) =251 THEN ME=ME–1KK=–1
 620 IF PEEK (KY) =253 THEN ME=ME+1:KK=1
 630 IF PEEK (KY) = 191 THEN ME=ME+KK
 640 IF PEEK (ME)<>32 THEN 700
 650 POKE ME, C
 660 RETURN
 700 IF PEEK (ME) =240 THEN GY=240
 705 IF GY=240 AND PK THEN KP=KP+1:
     GY=0: GOTO 650
 720 PRINT "YOU BLEW IT!"
 725 PRINT
 730 MI=ML*PL
 750 PRINT "AFTER"; MI;"MILES"
 755 IF PK THEN PRINT "AND"; KP; "KILLS"
 757 PRINT: PRINT "TOTAL POINTS:";
     INT(MI+4*(1–PK) *MI + 100*KP)
 760 GOSUB 1000
 810 GOTO 5000
1000 IF PK THEN WD=KP: GOTO 1030
1010 WD=MI/WF
1030 PRINT: PRINT "CONGRATULATIONS"
1040 PRINT "YOU MAY NOW CALL"
1045 PRINT "YOURSELF"
1050 PRINT: PRINT "      "
1060 IF WD<3 THEN PRINT "LITTLE"; GOTO 1200
1070 IF WD<5 THEN PRINT "TENDER";:
     GOTO 1200
1080 IF WD<12.5 THEN PRINT "MEDIOCRE";:
     GOTO 1200
1099 IF WD<25 THEN PRINT "BIG";: GOTO 1200
1100 IF WD<38 THEN PRINT "MASTER";:
     GOTO 1200
1110 IF WD<50 THEN PRINT "GRAND";:
     GOTO 1200
1120 PRINT "CHEATER"
1200 PRINT "FOO"
1210 IF GY=240 THEN PRINT "KILLER!"
1220 PRINT "!"
1230 RETURN
5000 PRINT: PRINT: PRINT "PRESS SHIFT LOCK"
5001 PRINT: INPUT "AGAIN"; A$: A$= LEFT$
     (A$,1)
5010 IF A$<>"Y" THEN 6000
5020 INPUT "SAME"; A$:A$= LEFT$ (A$,1)
5025 IF A$<>"Y" THEN CLEAR
5030 GOTO 100
6000 END
```

©

THE PET® GAZETTE

A Fast Visible Memory Dump

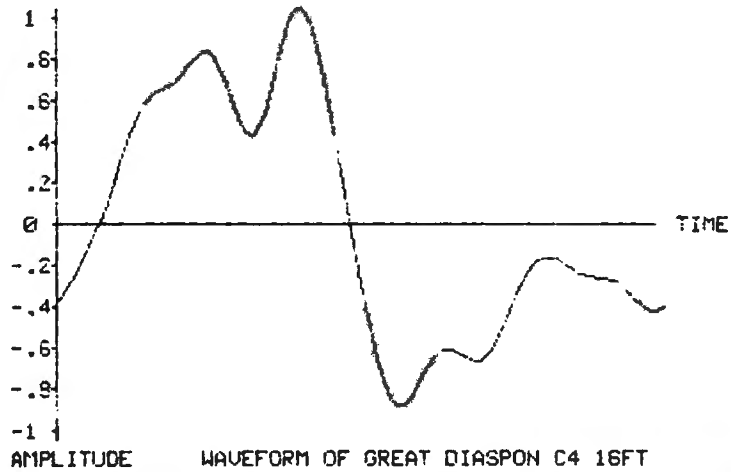
Martin J. Cohen, Ph.D.
Los Angeles, CA

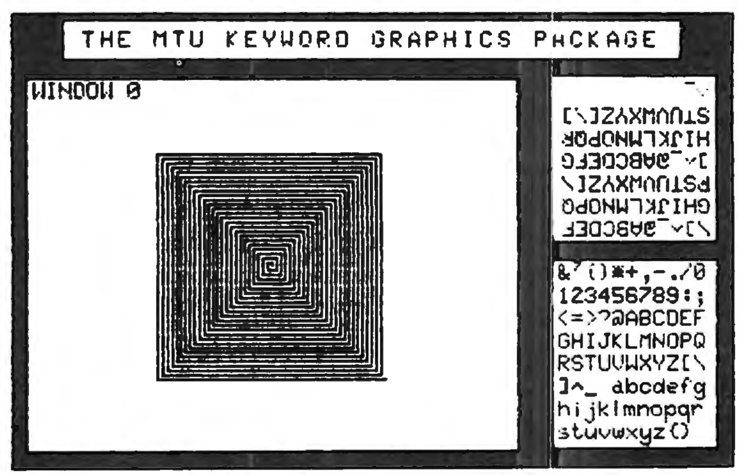
"The MTU Visible Memory is 8K bytes of dynamic RAM which, during refresh (transparent to the 6502), generates a video image of itself. The 320 (horizontal) by 200 (vertical) pixel display allows you to generate moderately high resolution graphics. (64,000 individual pixels can be set on or off — obviously a job for 6502 machine language or routines callable by BASIC.)"

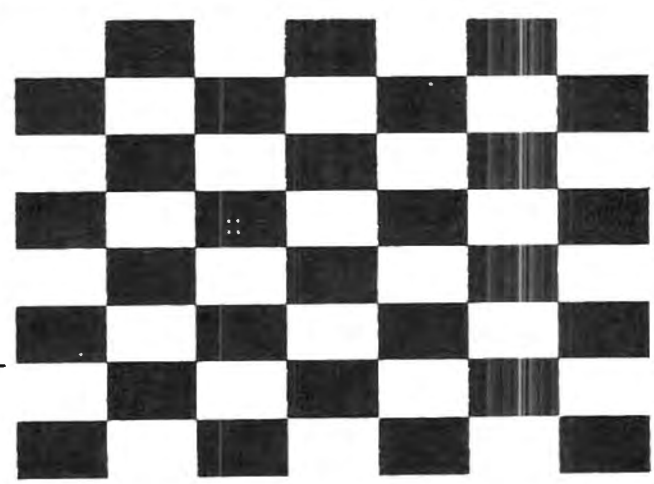
This description (on page 104 of **COMPUTE!**, issue 7, Nov./Dec., 1980) begins Dr. Frank Covitz's article in which he gives a truly ingenious method of using Commodore's 2022 tractor-feed printer to produce a hard copy of the MTU Visible memory. The primary disadvantage of this method is that, because the 2022 was not designed for graphics output, the process can take 10 to 30 minutes.

The 6502 machine language program described here, called SDUMP, produces a hard copy of the Visible Memory on Integral Data Systems' "Paper Tiger" printers with DotPlot graphics. Because these printers have graphics built in, the Visible Memory can be dumped in 90 seconds on any Paper Tiger and in only 45 seconds on the Paper Tiger 460 run at 9600 baud. These times apply to any contents of the Visible Memory, no matter how complicated or dense. The routine SDUMP does not even take advantage of clear areas of the Visible Memory, and could presumably be speeded up if this were done.

To see some of the capabilities of the Visible Memory/Paper Tiger combination, examine figures 1 through 3. Figure 1 shows four of the







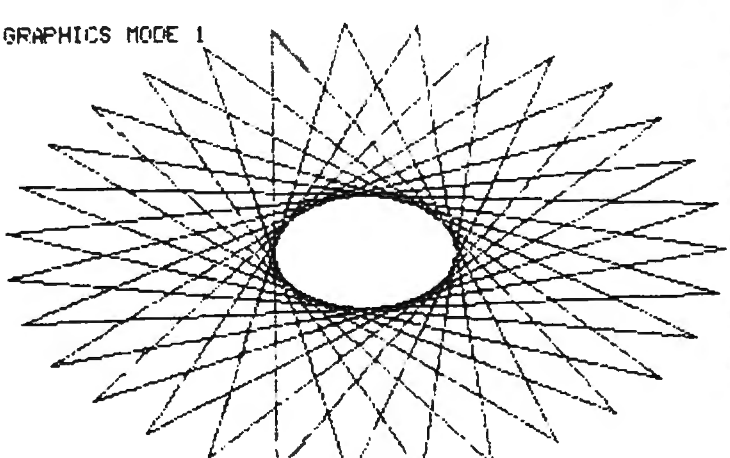


Figure 1

P1 (DENSITY) FOR K = 4, N = 2

Q	ALPHA = 0.	.125	.25	.5	1	2	4
	RHO = 0.	.5	1	2	4	8	16
.05	.285	.181686	.115778	.046960	.007693	.000203	0.
.1	.540000	.361678	.241868	.107708	.021050	.000768	0.
.15	.765000	.538112	.377244	.183775	.042380	.002078	.000004
.2	.960000	.708930	.520502	.276517	.074611	.004827	.000015
.25	1.125	.871864	.669801	.386991	.121355	.010224	.000052
.3	1.26	1.024422	.822792	.515782	.186954	.020314	.000166
.35	1.365	1.163877	.976548	.662790	.276438	.038453	.000491
.4	1.44	1.287245	1.127478	.826958	.395385	.069998	.001387
.45	1.485	1.391273	1.271233	1.005928	.549572	.123247	.003769
.5	1.5	1.472420	1.402602	1.195608	.744344	.210629	.009896
.55	1.485	1.526836	1.515399	1.389644	.983511	.350033	.025174
.6	1.44	1.550348	1.602333	1.578762	1.267587	.565844	.062113
.65	1.365	1.538434	1.654867	1.749954	1.591018	.888714	.148533
.7	1.26	1.486205	1.663062	1.885497	1.938004	1.351842	.343298
.75	1.125	1.388381	1.615403	1.961747	2.276254	1.979278	.762388
.8	.960000	1.239265	1.498604	1.947681	2.547874	2.757421	1.608574
.85	.765	1.032722	1.297404	1.803133	2.656193	3.572946	3.152575
.9	.54	.762148	.994325	1.476662	2.446936	4.086017	5.446845
.95	.285	.420448	.569423	.903005	1.681591	3.482044	7.005667

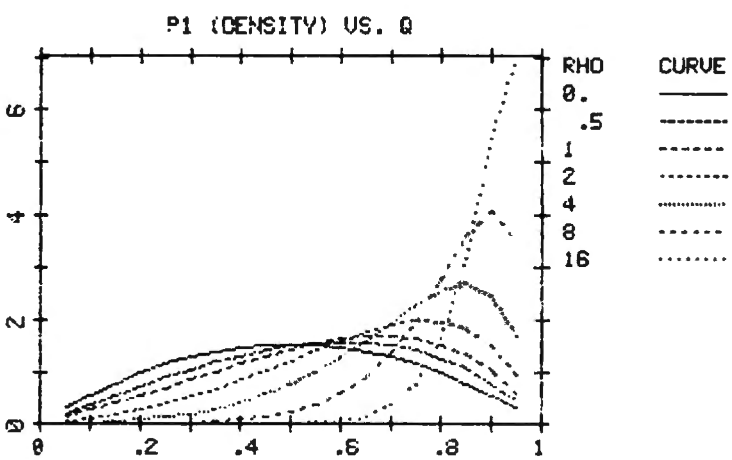


Figure 2

screens produced by the demonstration program supplied with the visible Memory. Figures 2 and 3 show some intermixed text and graphics produced using the MTU Keyword Graphics Package, of which I am the principal author. This package interfaces with BASIC to allow graphics commands to be entered as part of your BASIC program. Listing 1 shows the code used to produce the plots in figures 2 and 3.

The principal problem in dumping the Visible Memory to the Paper Tiger is that a byte of the Visible Memory is displayed as 8 pixels lined up horizontally, while a byte output to the Paper Tiger in graphics mode produces, depending on the model, 6 or 7 dots lined up vertically. The main task of SDUMP is therefore to take 6 or 7 bytes in the Visible Memory which are lined up vertically and convert them to 8 bytes of 6 or 7 bits which will then be output to the Paper Tiger.

My first attempt at this was done in BASIC, and is in listing 2. I knew it would execute extremely slowly, but it would be much easier to debug. Once the code was working, it was a fairly straightforward matter to translate the BASIC into assembly language — since I knew the logic was correct, I only had to make sure the translation was correct. Another advantage of this method is that if I want to program the routine in some other language, such as PASCAL, FORTH, or FORTRAN, it will be much easier to do it with BASIC as the basis instead of assembly language.

The current version of SDUMP is in listing 3. It is a modularized form of the BASIC code in listing 2, and is designed to be easily modifiable. It is assembled starting at $6000 (hex), so that it can reside in memory with the MTU Keyword Graphics Package, and be called with a SYS (96*256).

The initial part of SDUMP contains a transfer vector and a data area. The transfer vector has jumps to the three main routines in SDUMP: OUTVM, which dumps the whole Visible Memory; OUTROW, which dumps 6 or 7 rows of the Visible Memory starting at the location set in VM (at $6013); and OUTCOL, which outputs a column of

P1 (CUMULATIVE) FOR K = 4, N = 2

Q	ALPHA = 0.	.125	.25	.5	1	2	4
	RHO = 0.	.5	1	2	4	8	16
.05	.01425	.009084	.005788	.002348	.000384	.000010	0.
.1	.04125	.027168	.017882	.007733	.001437	.000048	0
.15	.0795	.054073	.036744	.016922	.003556	.000152	0.
.2	.1275	.089520	.062769	.030748	.007286	.000393	.000001
.25	.18375	.133113	.096259	.050097	.013354	.000905	.000003
.3	.24675	.184334	.137399	.075886	.022702	.001920	.000012
.35	.315	.242528	.186228	.109026	.036524	.003843	.000036
.4	.387	.306890	.242600	.150374	.056293	.007343	.000105
.45	.46125	.376454	.306162	.200670	.083772	.013505	.000294
.5	.53625	.450075	.376292	.260451	.120989	.024037	.000789
.55	.6105	.526417	.452062	.329933	.170164	.041539	.002047
.6	.6825	.603934	.532179	.408871	.233544	.069831	.005153
.65	.75075	.680856	.614922	.496369	.313095	.114267	.012580
.7	.81375	.755166	.698075	.590643	.409995	.181859	.029745
.75	.870000	.824586	.778845	.688731	.523808	.280823	.067864
.8	.918000	.886549	.853775	.786115	.651201	.418694	.148293
.85	.956250	.938185	.918646	.876272	.784011	.597341	.305922
.9	.983250	.976292	.968362	.950105	.906358	.801642	.578264
.95	.997500	.997315	.996833	.995255	.990437	.975744	.928547

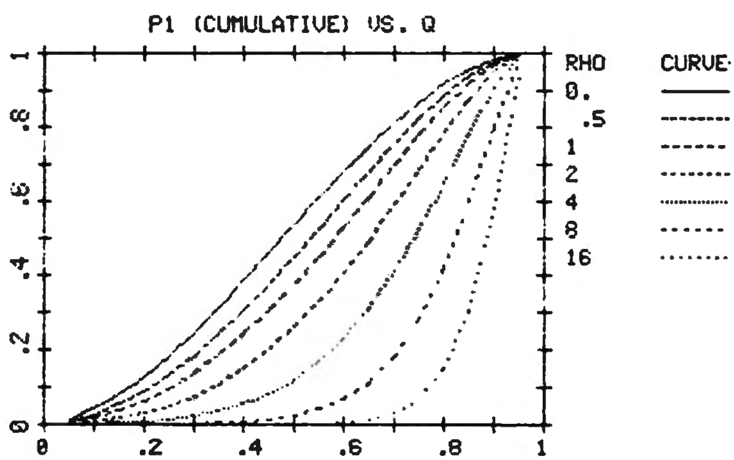


Figure 3

PE FOR K = 4, N = 2

M	ALPHA = 0.	.125	.25	.5	1	2	4
	RHO = 0.	.5	1	2	4	8	16
2	.509553	.454439	.408615	.338910	.259034	.218670	.288887
4	.758892	.708395	.661886	.582550	.469299	.362902	.359455

8 bytes, 6 or 7 bits high. These routines are made available in this manner in case you would like to mix text and graphics in a more sophisticated manner than a simple dump.

Following the transfer vector is the data area. The values here specify how the Visible Memory is to be dumped and where it is. SDUMP is assembled to work with the 460 Paper Tiger, but by making the changes described in lines 25-27, the code will work on the Paper Tiger 440. Presumably, with similarly minor changes, SDUMP will also work on the newest Paper Tiger, the 445.

The following should be noted about SDUMP and its use: The only code in SDUMP that is specific to a particular version of BASIC is that in OUTCH, lines 235-280. This code was given to me by Greg Yob — thanks Greg. It outputs the character in the ACC directly to the device whose number is in RDEV, at location $600E in the data area. Because this code bypasses the PET's file system and directly accesses the IEEE-488 routines, the device does not even have to be opened.

Each routine in SDUMP checks to see if the stop key is pressed, using the routine STOPTS at lines 281-292. If so, the routine quits and returns to the routine which called it. Because of the way the Paper Tigers enter and exit graphics mode, it is possible for them to be left in graphics mode when the stop key is pressed. If this happens, you will know it when it does, the easiest method of recovering is to turn the printer off, then on.

You should not have a CMD operation open

```
4000 REM PLOT X IN (XA,XB) NX WIDE, Y IN (YA,YB) NY HIGH, KEY PK
4005 IF PF=0 THEN RETURN
4010 CLEAR
4060 CX=NX/(XB-XA):REM CONVERSION CONSTANTS
4070 CY=NY/(YB-YA)
4090 FOR JR=1 TO NR:REM GET THE DATA POINTS
4100 DOTL DO(JR,1),DO(JR,2)
4110 FOR JQ=1 TO NQ
4120 X=QS(JQ)
4130 IX=INT(.5+(X-XA)*CX)
4140 IF IX<0 THEN IX=0:REM MAKE SURE X OK
4150 IF IX>NX THEN IX=NX
4160 IX=IX+OX
4210 IF PK=1 THEN Y=DD(JQ,JR):REM PK=1 FOR P1 DENS DIST
4220 IF PK=2 THEN Y=CD(JQ,JR):REM PK=2 FOR P1 CUM DIST
4300 REM CONVERT Y LIKE X
4310 IY=INT(.5+(Y-YA)*CY)
4320 IF IY<0 THEN IY=0:REM FORCE ON PLOT
4330 IF IY>NY THEN IY=NY
4335 IY=IY+OY
4340 IF JQ=1 THEN MOVE IX,IY
4350 DRAW IX,IY
4480 NEXT JQ
4490 NEXT JR
4495 DOTL 1,0
4500 REM PRODUCE THE PLOT
4510 MOVE OX,OY:REM BORDER
4520 DRAW OX+NX,OY:DRAW OX+NX,OY+NY
4530 DRAW OX,OY+NY:DRAW OX,OY
4590 MOVE OX+NX/2-3*(LEN(PL$)+6),OY+NY+10
4592 CHAR PL$;" VS. Q"
4594 PL$=""
4600 REM DISPLAY RHO AND DOTS
4610 IX=OX+NX+10.IY=OY+NY-7
4620 MOVE IX,IY:CHAR "RHO    CURVE"
4630 FOR I=1 TO NR
4640 V=RS(I):GOSUB2002
4650 IY=IY-12:MOVE IX,IY
4660 CHAR V$
4670 DOTL DO(I,1),DO(I,2):LINE IX+42.IY+3,319,IY+3
4680 NEXT I
4690 DOTL 1,0
4700 REM DRAW A GRID
4702 TL=3:REM TIC LENGTH
4705 DX=.1:REM X GRID SPACING (ALWAYS)
4710 DY=10:REM Y SPACING - HAVE TO SEARCH
4715 IF YB/5<DY THEN DY=DY/10:GOTO4715
4720 EX=INT(XB/DX+.01):EY=INT(YB/DY+.01)·REM POINTS ON GRID
4725 FX=1:IF EX>5 THEN FX=2:IF EX>10 THEN FX=5:IF EX>20 THEN FX=10
4730 FY=1:IF EY>5 THEN FY=2:IF EY>10 THEN FY=5:IF EY>20 THEN FY=10
4735 FOR I=0 TO EY:OZ=OY+I*DY*CY:LINE OX-TL,OZ,OX+TL,OZ:REM Y AXIS
4737 LINE OX+NX-TL,OZ,OX+NX+TL,OZ
4740 IF I=FY*INT(I/FY) THEN CHROT 1.MOVE OX-TL-5,OZ-3:CHAR MID$(STR$(I*DY),2)
4745 NEXT I
4750 FOR I=0 TO EX:OZ=OX+I*DX*CX:LINE OZ,OY-TL,OZ,OY+TL:REM X AXIS
4752 LINE OZ,OY+NY-TL,OZ,OY+NY+TL
4755 IF I=FX*INT(I/FX) THEN CHROT 0:MOVE OZ-3,OY-TL-10:CHAR MID$(STR$(I*DX),2)
4760 NEXT I
4900 REM PRODUCE THE PLOT
4910 PRINT:PRINT
4920 CMD3:REM REGULAR OUT TO THE SCREEN
4930 SYS(LP):REM THERE IT GOES
4940 CMD1:REM BACK TO THE PRINTER
4950 RETURN
```

Listing 1

to the Paper Tiger when SDUMP is called, because of the way this command is interpreted in the IEEE-488 system. To avoid this, open a unit to the screen (device 3) and switch to this unit before invoking SDUMP. For example:

```
OPEN 1,4:REM PRINTER FILE
OPEN 2,3:REM SCREEN FILE
CMD 1:REM OUTPUT TO PRINTER
.......
CMD 2:REM DIVERT OUTPUT
SYS(96*256):REM DUMP VISIBLE MEMORY
CMD 1:REM RESUME PRINTER OUTPUT
```

The byte in the data area called EORVAL (at $6011) is exclusive-ored with each Visible Memory byte when it is accessed for dumping. This gives a visible indication of the progress of the dump which I find entertaining. It is actually an instance

of Cohen's first law of interactive computing — "Always let the operator know that something is going on." However, this leaves the screen reversed when the dump finishes. If you do not like this, there are (at least) two possibilities: (1) Set EORVAL to zero ($00); the exclusive or will than not change anything. (2) If you are using the Keyword Graphics Package, follow the call to SDUMP with a 'SCFLIP 0,0,319,199'; this will reverse the whole screen, restoring its original condition.

To load SDUMP together with the MTU keyword Graphics Package, when reserving memory space, do a 'POKE 53,96' instead of 'POKE 53,98' for a 32K system, and similarly for smaller systems. This will reserve the two pages needed by SDUMP.

```
9100 REM MTU VISIBLE MEMORY TO IDS 460 PAPER TIGER SCREEN DUMP
9110 PRINT#U:REM SPACE
9120 PRINT#U,CHR$(3);:REM ENTER GRAPHICS MODE
9130 VM=256*PEEK(832):REM START OF VISIBLE MEMORY
9140 PVMEM
9142 GRSHRT
9145 S=7:REM ROWS PER GROUP
9150 FOR R0=0 TO 199 STEP S:REM S ROWS AT A TIME
9160 :R1=R0+S-1:REM END OF ROW GROUP
9170 :IF R1>199 THEN R1=199
9180 :FOR C=0 TO 39:REM A BYTE (8 BIT COLUMNS) AT A TIME
9190 ::FOR I=0 TO 7:REM CLEAR VALUES TO BE PRINTED
9200 :::P(I)=0
9210 ::NEXT I
9220 ::V=VM+C:REM LOC OF BYTE
9225 ::P2=1:REM POWER OF 2 TO ADD
9230 ::FOR R=R0 TO R1:REM SCAN THE ROWS
9235 :::PRINT C;R
9240 :::B=PEEK(V):REM GET THE BYTE (8 BITS)
9250 :::V=V+40:REM LOC OF BYTE BELOW
9260 .::IF B=0 THEN 9315.REM FASTER IF EMPTY
9270 :::M=1:REM MASK (2^(7-I))
9280 :::REM ACCUMULATE VALUES FOR PRINTING
9290 :::FOR I=7 TO 0 STEP -1
9295 ::::IF (B AND M)<>0 THEN P(I)=P(I)+P2
9300 ::::M=M+M
9310 :::NEXT I
9315 :::P2=P2+P2
9320 ::NEXT R:REM DO THE ROWS
9330 ::REM NOW, PRINT THE 8 COLUMNS OF ROWS
9340 ::FOR I=0 TO 7
9350 :::PRINT#U,CHR$(P(I));
9360 :::IF P(I)=3 THEN PRINT#U,CHR$(P(I));:REM 3 IS SPECIAL
9370 ::NEXT I
9390 :NEXT C:REM END OF COLUMN LOOP
9400 :PRINT#U,CHR$(3);CHR$(14);:REM GRAPHICS LINE FEED/RETURN
9410 :VM=VM+S*40:REM DOWN S ROWS
9420 NEXT R0:REM END OF ROW GROUP LOOP
9430 PRINT#U,CHR$(3);CHR$(2):REM LEAVE GRAPHICS MODE
9439 JX
9440 VISMEM
9450 RETURN:REM DONE
```

Listing 2

Listing 3

```
0000            . SDUMP.ASM - MTU TO IDS PAPER TIGER 460 (440) SCREEN DUMP
0000            ;
0000              BY MARTIN J. COHEN, DECEMBER 1980
0000            .
0000            , ANYONE WHO WANTS TO CAN USE THIS PROGRAM,
0000            . ALTHOUGH SOME ACKNOWLEDGEMENT WOULD BE APPRECIATED
0000            ;
0000            ;
0000            : APPROXIMATE TIME NEEDED TO DUMP VISIBLE MEMORY:
0000            ;
0000            . AT 1200 BAUD, 1 MIN, 30 SEC
0000            ; AT 9600 BAUD, 45 SEC (WITH 3 MS DELAY SET BY NMSDLY, BELOW)
0000            .
0000            , THE ACTUAL CPU TIME NEEDED IS ABOUT 3 SECONDS!!
0000            ;
0000            :
0000                    * = $6000        ; TWO PAGES BELOW KGP CODE
6000            ;
6000  4C 18 60          JMP OUTVM       , SKIP DATA AREA AND DUMP THE VIS MEM
6003  4C 8D 60          JMP OUTROW      ; OUTPUT ROW STARTING AT VM
6006  4C DF 60          JMP OUTCOL      ; OUTPUT A COLUMN OF 8 BYTES
6009            ,
6009            ; DATA AREA ...
6009            ,
6009            ; NOTE - TO RUN THIS ROUTINE ON A 440 INSTEAD OF A 460,
6009            , CHANGE THE FOLOWING VALUES AS INDICATED (VALUES IN DECIMAL):
6009            ; RPFXC=0, RREP=33, RVAL=6, REND=2, RXGR=11.
6009            ,
6009  14        RPFXC   .BYTE 20        , NUMBER OF BLANK PREFIX COLUMNS (440:0)
600A  02        RPFXR   .BYTE 2         , NUMBER OF BLANK PREFIX ROWS
600B  1C        RREP    .BYTE 28        , MAIN REPETITION COUNT (440:33)
600C  07        RVAL    .BYTE 7         . ROWS TO OUTPUT IN MAIN LOOP (440 6)
```

NEW LIFE for old Pets™?

Is it a dream ... is it fantasy?

Switch from old ROMs to new

- Not sure about the ROM Retrofit Kit from Commodore? Now you can use all three sets of Commodore ROMs and others as well.
- The Basic Switch allows switch selection of either ROM set (your original set or your retrofit set) from Commodore. Plus, Models 15-A and 15-B include an additional zero insertion force socket allowing easy use of ROMs like the BASIC Programmer's Toolkit ... concurrently.
- Models 15-A and 15-B The Basic Switch plus ... includes expanded cable assembly and zero insertion force socket. Your 15th ROM simple plugs in ... enabled while either ROM set is selected Socket 15 may be readdressed by the user for additional flexibility.
- The Basic Switch is sold in assembled form only. All models are designed for easy attachment to your PET with a convenient cable assembly. No soldering or drilling is required. The Basic Switch mates with a cable assembly at your primary board, and does not use the physical connectors of any PET ports.
- Price List:
  Old PETs $129.95*
  New PETs $139.95*
  *Case optional, $15 additional.
- Our prices and complete product specifications are available by contacting APPLIED MICRO SYSTEMS, Mishawaka, Indiana; or any Commodore Dealer.

Now available for Commodore® Basic 4.0

**Dealer Inquiries are encouraged.
Free MACHINE LANGUAGE MONITOR COMMANDS List will be included with pricing and product specification requests.**

APPLIED MICRO SYSTEMS

3502 Home Street, Mishawaka, Indiana 46544
1-219-259-3787 (Indiana) • 1-800-348-7208

COMPUTE!'s Book Corner

Back Issues:

COMPUTE! January, 1981, Issue 8 .... $2.50 ☐
COMPUTE! February, 1981, Issue 9 .... $2.50 ☐
COMPUTE! March, 1981, Issue 10 .... $2.50 ☐

For The Single Board Computer Owners: (KIM, SYM, AIM, OSI)

COMPUTE II, All Three Issues
(The only ones: **COMPUTE II** merged into **COMPUTE!**)
Issues 1, 2, and 3 .... $5.00 ☐

Individual Issues

COMPUTE II, #1 .... $2.00 ☐
COMPUTE II, #2 .... $2.00 ☐
COMPUTE II, #3 .... $2.00 ☐

Miscellaneous:

Best of the PET Gazette .... $10.00 ☐
(Note...an old anthology primarily for old ROM, 8K original PETs.)
Collected PET User Notes (7 issues) .... $10.00 ☐
(Above note applies)
Collected 6502 User Notes (17 issues) .... $20.00 ☐
(Specifically for Single-Board Computer Owners)

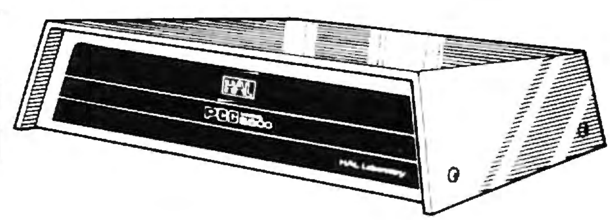
Ordering Information:

Address orders to:
Compute's Book Corner
P.O. Box 5406
Greensboro, NC 27403 USA

Payment, in US funds, required with order. *Master Charge/VISA accepted. All items subject to availability. Please add the following amount(s) for shipping/handling:
US $2.00 (UPS or mail) • Canada $2.00 (mail)
Foreign Air Mail $5.00

The $180 Programmable character generator that performs above and beyond its price.

It's the NEW HAL PCG 6500 for your PET. With HAL, you control each dot in the 8 x 8 matrix so you get sharp, clear graphics. HAL offers:



- Storage for 64 user-programmable characters
- Built-in CB2 sound amplifier
- PCG Manual and demo program
- Interface to PET/CBM with 24 pin character generator ROMS (If your PET uses 28 pin ROMS, Conversion Kits are available for $45. To simplify ordering, please indicate memory RAM #2114 or #6550.)

The HAL PCG 6500 is available exclusively at Systems Formulate...

Call today for more information or to place your phone order: (415) 326-9100 39 Town & Country Village, Palo Alto, CA 94301
We honor Master Charge, Visa, Check or Money Order (California residents add 6½% sales tax)

Dealer Inquiries Invited

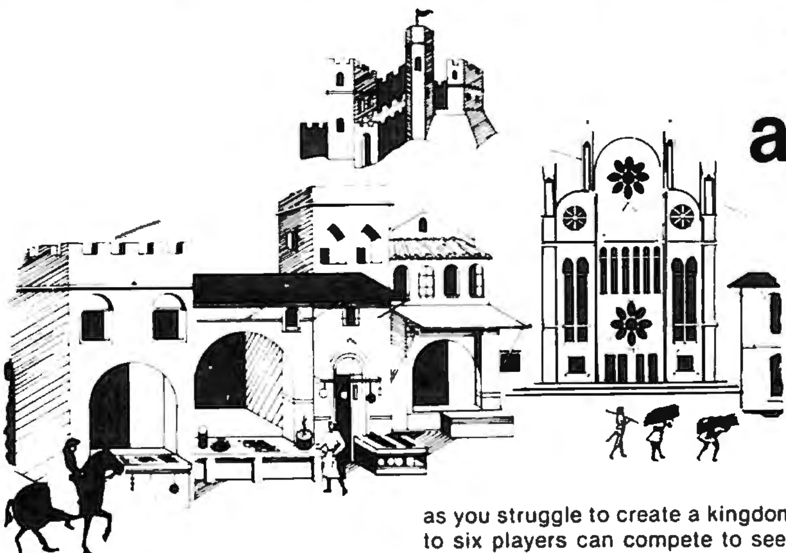
SYSTEMS FORMULATE CORPORATION
39 Town & Country Village • Palo Alto, California 94301

```
6 0 0 D   04          REND    .BYTE 4          , ROWS TO OUTPUT AT END (440:2)
6 0 0 E               ; THE TOTAL NUMBER OF ROWS OUTPUT = RREP*RVAL + REND = 200
6 0 0 E   04          RDEV    .BYTE 4          ; OUTPUT DEVICE
6 0 0 F   0E          RXGR    .BYTE 14         ; GRAPHICS RETURN (440·11)
6 0 1 0   03          NMSDLY  .BYTE 3          , MS TO DELAY AFTER EACH BYTE
6 0 1 1   FF          EORVAL  .BYTE $FF        , VALUE TO EOR WITH SCREEN LOC ($FF TO
FLIP, 0 TO SKIP)
6 0 1 2   90          VMPAGE  .BYTE $90        ; STARTING PAGE OF VISIBLE MEMORY
6 0 1 3               ;
6 0 1 3   00 00       VM      .WORD 0          ; LOCAL STORAGE - LOC OF A VIS MEM ROW
6 0 1 5   28          BYTEPL  .BYTE 40         ; BYTES PER VM LINE
6 0 1 6   00          RREPX   .BYTE 0          ; STORAGE FOR REP COUNT
6 0 1 7   00          RVALX   .BYTE 0          ; STORAGE FOR ROW COUNT
6 0 1 8               ,
6 0 1 8               ; OUTVM - OUTPUT THE WHOLE VISIBLE MEMORY
6 0 1 8               ;
6 0 1 8   AD 12 60    OUTVM   LDA VMPAGE       ; SET LOC OF VM
6 0 1 B   8D 14 60            STA VM+1
6 0 1 E   A9 00               LDA #0
6 0 2 0   8D 13 60            STA VM
6 0 2 3   AD 0B 60            LDA RREP         , SET MAJOR REP COUNT
6 0 2 6   8D 16 60            STA RREPX
6 0 2 9   20 88 60            JSR ENTRGR       ; ENTER GRAPHICS MODE
6 0 2 C   AE 0A 60            LDX RPFXR        ; SEE IF ANY PREFIX ROWS
6 0 2 F   F0 06               BEQ OUTVM1
6 0 3 1   20 C5 60    OUTVM0  JSR OUTRET       ; IF SO, OUTPUT THEM
6 0 3 4   CA                  DEX
6 0 3 5   D0 FA               BNE OUTVM0
6 0 3 7               ;
6 0 3 7               OUTVM1 = *
6 0 3 7   20 BA 61            JSR STOPTS       ; CHECK FOR STOP KEY
6 0 3 A   B0 3A               BCS OUTVMF
6 0 3 C   20 B1 60            JSR OUTPFX       ; OUTPUT A LINE P
6 0 3 F   AD 0C 60            LDA RVAL         ; SET ROW COUNT
6 0 4 2   8D 17 60            STA RVALX
6 0 4 5   20 8D 60            JSR OUTROW       , OUTPUT A ROW
6 0 4 8   20 C5 60            JSR OUTRET       ; OUTPUT A RETURN
6 0 4 B   AE 0C 60            LDX RVAL         ; SET VM = VM+RVAL*40
6 0 4 E   18          OUTVM2  CLC
6 0 4 F   AD 13 60            LDA VM
6 0 5 2   6D 15 60            ADC BYTEPL
6 0 5 5   8D 13 60            STA VM
6 0 5 8   AD 14 60            LDA VM+1
6 0 5 B   69 00               ADC #0
6 0 5 D   8D 14 60            STA VM+1
6 0 6 0   CA                  DEX
6 0 6 1   D0 EB               BNE OUTVM2
6 0 6 3               ;
6 0 6 3   CE 16 60            DEC RREPX        ; COUNT ROWS
6 0 6 6   D0 CF               BNE OUTVM1
6 0 6 8               ,
6 0 6 8   20 B1 60            JSR OUTPFX       ; START OF LAST ROW
6 0 6 B   AD 0D 60            LDA REND         ; NUMBER OF ROWS
6 0 6 E   F0 06               BEQ OUTVMF       , SKIP IF NONE
6 0 7 0   8D 17 60            STA RVALX
6 0 7 3   20 8D 60            JSR OUTROW       ; THERE IT GOES
6 0 7 6               OUTVMF = *
6 0 7 6   20 C5 60            JSR OUTRET
6 0 7 9   20 7D 60            JSR EXITGR       ; LEAVE GRAPHICS MODE
6 0 7 C   60                  RTS              ; DONE
6 0 7 D               ;
6 0 7 D               ; EXITGR - LEAVE GRAPHICS MODE
6 0 7 D               ;
6 0 7 D   A9 03       EXITGR  LDA #3
6 0 7 F   20 6D 61            JSR OUTCH
6 0 8 2   A9 02               LDA #2
6 0 8 4   20 6D 61            JSR OUTCH
6 0 8 7   60                  RTS
6 0 8 8               ;
6 0 8 8               , ENTRGR - ENTER GRAPHICS MODE
6 0 8 8               ;
6 0 8 8   A9 03       ENTRGR  LDA #3
6 0 8 A   4C 6D 61            JMP OUTCH
6 0 8 D               ,
6 0 8 D               ; OUTROW - OUTPUT THE ROW POINTED TO BY VM, RVALX DEEP
6 0 8 D               ;
6 0 8 D   AD 13 60    OUTROW  LDA VM           ; SET WHERE TO START
6 0 9 0   8D D1 60            STA V
```

```
6093  AD 14 60           LDA VM+1
6096  8D D2 60           STA V+1
6099  AE 15 60           LDX BYTEPL      . DO 40 COLUMNS
609C  AD 17 60   OUTR1   LDA RVALX        ; SET DEPTH COUNT
609F  8D D3 60           STA R
60A2  20 DF 60           JSR OUTCOL      ; OUTPUT THOSE 8
60A5  EE D1 60           INC V           ; BUMP LOC
60A8  D0 03              BNE OUTR2
60AA  EE D2 60           INC V+1
60AD  CA         OUTR2   DEX              ; COUNT
60AE  D0 EC              BNE OUTR1
60B0  60                 RTS             ; DONE
60B1             ,
60B1             ; OUTPFX - OUTPUT RPFXC SPACES TO START LINE
60B1             ;
60B1  AE 09 60   OUTPFX LDX RPFXC
60B4  F0 0E              BEQ OUTPF2      ; CHECK FOR NONE
60B6  20 7D 60           JSR EXITGR
60B9  A9 20              LDA #32         ; LOAD THE SPACE
60BB  20 6D 61   OUTPF1 JSR OUTCH         , OUTPUT IT
60BE  CA                 DEX             ; UNTIL DONE
60BF  D0 FA              BNE OUTPF1
60C1  20 88 60           JSR ENTRGR
60C4  60         OUTPF2 RTS               ; THAT'S ALL
60C5             ;
60C5             ; OUTRET - OUTPUT A GRAPHICS RETURN
60C5             ;
60C5  A9 03      OUTRET LDA #3
60C7  20 6D 61           JSR OUTCH
60CA  AD 0F 60           LDA RXGR
60CD  20 6D 61           JSR OUTCH
60D0  60                 RTS
60D1             ;
60D1             ; OUTCOL - OUTPUT 8 COLUMNS OF BITS
60D1             ;
60D1             ; PARAMETERS (BELOW) ARE V AND R
60D1             ;
60D1  00 00      V       .WORD 0          ; LOC IN VISIBLE MEMORY
60D3  00         R       .BYTE 0          ; NUMBER OF ROWS TO PROCESS
60D4             PO      * = *+8          ; RESULT TO OUTPUT
60DC             ;
60DC  00         P2      .BYTE 0          ; POWER OF 2 BIT
60DD  00         B       .BYTE 0          ; STORAGE FOR A BYTE
60DE  00         M       .BYTE 0          ; A MASK
60DF             ;
60DF             PGZ     = 1              . PAGE ZERO LOCATION TO USE
60DF             ;
60DF  48         OUTCOL PHA               ; SAVE REGS
60E0  8A                 TXA
60E1  48                 PHA
60E2  98                 TYA
60E3  48                 PHA
60E4  A5 01              LDA PGZ          ; SAVE PAGE ZERO AREA
60E6  48                 PHA
60E7  A5 02              LDA PGZ+1
60E9  48                 PHA
60EA  20 BA 61           JSR STOPTS       ; SEE IF STOP PRESSED
60ED  B0 72              BCS MOVEPF       ; IF SO, QUIT NOW
60EF  A9 00              LDA #0         ; ZERO P(0:7)
60F1  A2 07              LDX #7
60F3  9D D4 60   CLP2    STA PO,X
60F6  CA                 DEX
60F7  10 FA              BPL CLP2
60F9
60F9  A9 01              LDA #1          ; SET P2 TO 1
60FB  8D DC 60           STA P2
60FE  AD D1 60           LDA V           . STORE VM LOC
6101  85 01              STA PGZ
6103  AD D2 60           LDA V+1
6106  85 02              STA PGZ+1
6108  A0 00      RLOOP   LDY #0
610A  B1 01              LDA (PGZ)Y      ; GET VM BYTE
610C  8D DD 60           STA B           ; SAVE IT
610F  4D 11 60           EOR EORVAL       ; REVERSE IT FOR SHOW
6112  91 01              STA (PGZ)Y
6114  A5 01              LDA PGZ         ; POINT TO NEXT ROW
6116  18                 CLC
6117  6D 15 60           ADC BYTEPL
```

Captivate Yourself.

Santa Paravia and Fiumaccio



The most captivating and engrossing program ever made for the PET*

SANTA PARAVIA AND FIUMACCIO Become the ruler of a medieval city-state as you struggle to create a kingdom. Up to six players can compete to see who will become the King or Queen first. This program requires a PET 16K. **Order No. 0175P.$9.95.**

It is the dawn of the 15th Century; you rule a tiny Italian city-state. Your goal: The Crown!

Up to six players can compete as rulers of neighboring cities. You control the grain harvest, feed your serfs, set tax rates, dispense justice and invest in public works.

The future of your realm will depend on your decisions. If they are wise, your city-state will grow and you will acquire loftier titles. If your rule is incompetent, your people will starve and you may be invaded by your neighbors.

How will *you* rule your kingdom? Will you be an enlightened leader—or an unscrupulous despot? Only you can answer that question—with Santa Paravia and Fiumaccio.

CHIMERA

CHIMERA If you think the legendary Chimera was hard to handle, wait until you try the Chimera package. Included are:

- **Reflex**—Round and round the little white ball rolls. Only fast reflexes can guide it into the center of the maze.
- **Dragon**—You'll have to shoot down those pesky, fire-breathing dragons with your cannon. If you succeed your castle will be safe, if not it will mean a call to your fire insurance company. For one player.
- **Dungeon**—A very punctual guard comes down to the dungeon every day to torture you. This means that you have only thirty seconds to dig your way under the castle to freedom. For one player.
- **Dragon Hunt**—You must go forth and slay a fire-breathing dragon. The only thing that will protect you from the flames is your shield, if you know when to use it. For one player.
- **Dropoff**—You must make your opponent's men "dropoff" the board by moving and firing your own men. For one or two players. **Order No. 0110P. $9.95.**

PET DEMO I

PET DEMO I You can give yourself, your family, and your friends hours of fun and excitement with this gem of a package.

- **Slot Machine**—You won't be able to resist the enticing messages from this computerized one-armed bandit.
- **Chase**—You must find the black piece as you search through the ever-changing maze.
- **Flying Pheasant**—Try to shoot the flying pheasant on the wing.
- **Sitting Ducks**—Try to get your archer to shoot as many ducks as possible for a high score.
- **Craps**—It's Snake Eyes, Little Joe, or Boxcars as you roll the dice and try to make your point.
- **Gran Prix 2001**—Drivers with experience ranging from novice to professional will enjoy this multi-leveled race game.
- **Fox and Hounds**—It's you against the computer as your four hounds try to capture the computer's fox.

For true excitement, you'll need a PET 8K. **Order No. 0035P $7.95.**

DOW JONE$

DOW JONES Up to six players can enjoy this exciting stock market game. You can buy and sell stock in response to changing market conditions. Get a taste of what playing the market is all about. Requires a PET with 8K. **Order No. 0026P $7.95.**

We Guarantee It!

Instant Software Guarantee

OUR PROGRAMS ARE GUARANTEED TO BE QUALITY PRODUCTS. IF NOT COMPLETELY SATISFIED YOU MAY RETURN THE PROGRAM WITHIN 60 DAYS. A CREDIT OR REPLACEMENT WILL BE WILLINGLY GIVEN FOR ANY REASON.

Prices subject to change without notice.

TO ORDER

SEE YOUR LOCAL INSTANT SOFTWARE DEALER

OR

call now Toll-Free 1-800-258-5473

Instant Software™

*A trademark of Commodore Business Machines

PETERBOROUGH, N.H. 03458
603-924-7296

```
611A  85 01            STA PGZ
611C  A5 02            LDA PGZ+1
611E  69 00            ADC #0
6120  85 02            STA PGZ+1
6122           .
6122  A9 01            LDA #1          ; SET MASK TO 1
6124  8D DE 60         STA M
6127  A2 07            LDX #7           ; FOR I = 7 TO 0 STEP -1
6129  AD DD 60  ILOOP  LDA B            ; IF B AND M <> 0
612C  2D DE 60         AND M
612F  F0 09            BEQ ILOOP1
6131  BD D4 60         LDA PO,X        ; P(I)=P(I)+P2
6134  0D DC 60         ORA P2
6137  9D D4 60         STA PO,X
613A  0E DE 60  ILOOP1 ASL M            ; SHIFT MASK LEFT
613D  CA               DEX             ; SEE IF DONE
613E  10 E9            BPL ILOOP
6140           ;
6140  0E DC 60         ASL P2          , DOUBLE P2
6143  CE D3 60         DEC R           ; SEE IF OUTER LOOP DONE
6146  D0 C0            BNE RLOOP
6148           ;
6148           ; OUTPUT PO(0:7)
6148           ;
6148  A0 00            LDY #0
614A  20 BA 61  MOVEP  JSR STOPTS       ; SEE IF STOP PRESSEDD
614D  B0 12            BCS MOVEPF      ; IF SO, QUIT HERE
614F  B9 D4 60         LDA PO,Y
6152  20 6D 61         JSR OUTCH       ; OUTPUT A CHARACTER
6155  C9 03            CMP #3          ; SEE IF 3
6157  D0 03            BNE MOVEP1
6159  20 6D 61         JSR OUTCH       ; IF SO, DO IT AGAIN
615C  C8        MOVEP1 INY
615D  C0 08            CPY #8          ; ONLY DO 8
615F  D0 E9            BNE MOVEP
6161           ;
6161  68        MOVEPF PLA              ; RESTORE PAGE ZERO AREA
6162  85 02            STA PGZ+1
6164  68               PLA
6165  85 01            STA PGZ
6167  68               PLA              , RESTORE REGS
6168  A8               TAY
6169  68               PLA
616A  AA               TAX
616B  68               PLA
616C           ;
616C  60               RTS
616D           ;
616D           , OUTCH - OUTPUT A CHARACTER TO DEVICE RDEV
616D           ;
616D           ; THIS ROUTINE WAS SUPPLIED BY GREG YOB - THANKS MUCH
616D           ;
616D  8E B7 61  OUTCH  STX OUTCHX       , SAVE REGS
6170  8C B8 61         STY OUTCHY
6173  48               PHA
6174  A5 D4            LDA $D4          ; SAVE CURRENT DEVICE
6176  8D B9 61         STA TMPDEV
6179  AD 0E 60         LDA RDEV        ; SET MY DEVICE
617C  85 D4            STA $D4
617E  20 BA F0         JSR $F0BA       ; LISTEN
6181  20 2D F1         JSR $F12D       , ATTENTION
6184            OUTCH1 = *
6184  20 BA 61         JSR STOPTS       , SEE IF STOP PRESSED
6187  B0 11            BCS OUTCH2       ; IF SO, EXIT FROM HERE
6189  A9 00            LDA #0          ; CLEAR STATUS
618B  85 96            STA $96
618D  68               PLA             ; REGET CHAR
618E  48               PHA
618F  85 A5            STA $A5         , STORE WHERE IT SHOULD BE
6191  20 EE F0         JSR $F0EE       ; OUTPUT
6194  A5 96            LDA $96         ; SEE IF TIMED OUT
6196  25 01            AND 1
6198  D0 EA            BNE OUTCH1      ; IF SO, TRY AGAIN
619A            OUTCH2 = *
619A  20 83 F1         JSR $F183       ; UNLISTEN
619D  AD B9 61         LDA TMPDEV      ; RESTORE DEVICE
61A0  85 D4            STA $D4
61A2  AE 10 60         LDX NMSDLY       ; DELAY A FEW MS
```

```
00266  61A5   F0 08                 BEQ OUTCHF
00267  61A7   A0 C8        OUTCH3  LDY #200        ; 1 MS INNER LOOP (LOOP IS 5 CYCLES LONG)
00268  61A9   88           OUTCH4  DEY
00269  61AA   D0 FD                 BNE OUTCH4
00270  61AC   CA                    DEX
00271  61AD   D0 F8                 BNE OUTCH3
00272  61AF                OUTCHF  = *
00273  61AF   AE B7 61              LDX OUTCHX      ; AND REGS
00274  61B2   AC B8 61              LDY OUTCHY
00275  61B5   68                    PLA
00276  61B6   60                    RTS             . DONE
00277  61B7                ;
00278  61B7   00           OUTCHX  .BYTE 0
00279  61B8   00           OUTCHY  .BYTE 0
00280  61B9   00           TMPDEV  .BYTE 0
00281  61BA                ;
00282  61BA                ; STOPTS - SET CARRY IF STOP KEY PRESSED
00283  61BA                ;
00284  61BA                ; TAKEN FROM PAGE 5 OF MTU DOCUMENTATION FOR K-1002-6C
00285  61BA                ;
00286  61BA   AD 12 E8     STOPTS  LDA $E812       ; LOOK AT KEYBOARD
00287  61BD   C9 EF                 CMP #$EF        ; TEST FOR STOP KEY
00288  61BF   18                    CLC             ; CARRY CLEAR FOR NO STOP
00289  61C0   D0 01                 BNE STOPT1
00290  61C2   38                    SEC             , CARRY SET FOR YES STOP
00291  61C3   60           STOPT1  RTS
00292  61C4                ;
00293  61C4                        .END
ERRORS = 00000
SYMBOL TABLE
SYMBOL VALUE
 B        60DD    BYTEPL   6015    CLP2     60F3    ENTRGR   6088
 EORVAL   6011    EXITGR   607D    ILOOP    6129    ILOOP1   613A
 M        60DE    MOVEP    614A    MOVEP1   615C    MOVEPF   6161
 NMSDLY   6010    OUTCH    616D    OUTCH1   6184    OUTCH2   619A
 OUTCH3   61A7    OUTCH4   61A9    OUTCHF   61AF    OUTCHX   61B7
 OUTCHY   61B8    OUTCOL   60DF    OUTPF1   60BB    OUTPF2   60C4
 OUTPFX   60B1    OUTR1    609C    OUTR2    60AD    OUTRET   60C5
 OUTROW   608D    OUTVM    6018    OUTVM0   6031    OUTVM1   6037
 OUTVM2   604E    OUTVMF   6076    P2       60DC    PGZ      0001
 PO       60D4    R        60D3    RDEV     600E    REND     600D
 RLOOP    6108    RPFXC    6009    RPFXR    600A    RREP     600B
 RREPX    6016    RVAL     600C    RVALX    6017    RXGR     600F
 STOPT1   61C3    STOPTS   61BA    TMPDEV   61B9    V        60D1
 VM       6013    VMPAGE   6012
END OF ASSEMBLY
```

```
CROSS REFERENCE......PAGE    1

B       $60DD   152   182   196
BYTEPL  $6015    42    72   112   187
CLP2    $60F3   170   172
ENTRGR  $6088    54   103   132
EORVAL  $6011    38   183
EXITGR  $607D    90    95   127
ILOOP   $6129   196   204
ILOOP1  $613A   198   202
M       $60DE   153   194   197   202
MOVEP   $614A   213   222
MOVEP1  $615C   218   220
MOVEPF  $6161   167   214   224
NMSDLY  $6010    37   265
OUTCH   $616D    96    98   104   129
                138   140   216   219   240
OUTCH1  $6184   249   260
OUTCH2  $619A   251   261
OUTCH3  $61A7   267   271
OUTCH4  $61A9   268   269
OUTCHF  $61AF   266   272
OUTCHX  $61B7   240   273   278
OUTCHY  $61B8   241   274   279
OUTCOL  $60DF    21   115   157
OUTPF1  $60BB   129   131
OUTPF2  $60C4   126   133
OUTPFX  $60B1    64    83   125
OUTR1   $609C   113   120
OUTR2   $60AD   117   119
OUTRET  $60C5    57    68    89   137
OUTROW  $608D    20    67    87   108
OUTVM   $6018    19    48
OUTVM0  $6031    57    59
OUTVM1  $6037    56    61    81
OUTVM2  $604E    70    78
OUTVMF  $6076    63    85    88
P2      $60DC   151   175   200   206
PGZ     $0001   155   162   164   177   179
                181   184   185   188   189
                191   225   227
PO      $60D4   149   170   199   201   215
R       $60D3   114   148   207
RDEV    $600E    35   245
REND    $600D    33    84
RLOOP   $6108   180   208
RPFXC   $6009    29   125
RPFXR   $600A    30    55
RREP    $600B    31    52
RREPX   $6016    43    53    80
RVAL    $600C    32    65    69
RVALX   $6017    44    66    86   113
RXGR    $600F    36   139
STOPT1  $61C3   289   291
STOPTS  $61BA    62   166   213   250   286
TMPDEV  $61B9   244   263   280
V       $60D1   109   111   116   118   147
                176   178
VM      $6013    41    49    51    71    73
                 74    76   108   110
VMPAGE  $6012    39    48
```

©

Machine Language: Getting To The Machine Language Program

Jim Butterfield
Toronto, Canada

Your PET/CBM is a Basic machine. To run machine language you have to leave Basic – perhaps for a temporary period – and enter the machine language program. You'll often want Basic and Machine Language to work together. Where time is not critical, many things code easily into Basic. But where speed is important, or the job is beyond Basic's normal powers, you'll want to use machine language inserts. At that time, your computer will want to go into machine language.

There are four standard methods of doing this: some are more complex than others. Each has its own advantages and drawbacks.

The SYS command and the USR function call machine language whenever Basic desires to do so. This may be done with a direct command or from a program. The machine language program acts as a subroutine, and may return to the Basic calling point when it has done the job.

The more complex "wedge" method calls a machine language routine frequently whenever Basic is running. It doesn't wait for the Basic program to call it in; it seems to run simultaneously with Basic.

The interrupt method taps the PET's internal interrupt scheme. Every sixtieth of a second – whether Basic is running or not – PET's interrupt kicks in and does a number of quick jobs, such as checking the keyboard and flashing the cursor. Machine language programs which tap the interrupt seem to run continuously, even when Basic is not active.

The Machine Language Monitor has a Go (.G) command which allows you to start a machine language program directly. The program is not called as a subroutine, so it must find its own way back to the MLM when it is finished.

Each of the four methods will be discussed briefly here.

SYS And USR

SYS and USR create direct calls from Basic to a machine language program. This program runs only when called, and when it is finished it will hopefully return control to Basic and allow Basic execution to continue.

SYS is a command. You say SYS 7143, for example, as a direct command or within a program, and machine language at decimal address 7143 will start executing. SYS is quite convenient when you have several machine language programs to be run at different times: you just give the address of each one as you call it.

USR is a function, not a command. You cannot say USR(0) alone any more than you can say SQR(0): it must be part of a command. You might say any of: PRINT USR(0); X=USR(99); IF USR(7)=3 THEN ... or any similar syntax.

When Basic encounters the USR function within a Basic statement, it will start to execute machine language at a present address. Hopefully you will have set the address to point at the program you want to run; you do this by POKEing the desired address into locations 1 and 2. Once you've done this, USR will fire you into the desired machine code every time you use it.

The argument of the USR function — that's the value enclosed in brackets — is available to the machine language program if it wants to use it. This value may be found in the floating point accumulator, which is at hexadecimal B0 to B45 in original ROMs or at 5E to 63 in subsequent PET/CBM machines. It's store in floating point notation, which is devilishly hard to read if you don't know the system and not that easy if you do. When a simmple number like 5 comes up as hexadecimal 83 A0 00 00 00 20 you may be happy to reach for a built-in conversion routine that yields a much more readable fixed-point value of 00 05.

If you use the USR argument you may also leave a value in the same floating point accumulator just before you return to Basic. This value will be picked up by Basic as the value computed by the USR function.

To summarize: SYS lets you pick any of several machine language programs. USR takes you to a fixed location and allows you to pass a single value to and from machine language if you want. The SYS command seems simpler to the beginner, but USR is also straightforward once you get used to it.

The Wedge, Or Infiltrating Basic

This advanced technique gives the effect of a Basic "supervisor" which watches Basic run and occasionally kicks in with some of its own activities. It doesn't need to be called from Basic: once implanted, it will be there and active any time Basic is running.

It's a powerful method of extending Basic. Many systems use it: disk monitors, including the Commodore "wedge" DOS; Basic enhancers such as the Programmer's Toolkit or Basic Aid; and Brett Butler's TRACE as published in **COMPUTE!**,

issue 1.

How does it work? It's done by infiltrating a Basic subroutine called CHRGET which is located in page zero. This subroutine is called every time the Basic interpreter wants to get a character from your Basic program. By making very careful changes to this subroutine, you can force the Basic interpreter to do a little extra work for you.

It's not simple. But with a little persistence and a lot of bravery, you can train Basic to do some clever new tricks.

Interrupt

Sixty times a second, PET's normal activity freezes. An interrupt signal causes a completely independent program to run. When the interrupt program completes, the computer's normal programs unfreeze and continue exactly where they left off.

This powerful mechanism allows PET to do several important jobs. The jiffy clock is updated; the keyboard is checked for activity; the stop key is checked and its condition logged; the cursor is flashed when necessary; and the cassette motors are started or stopped. All of this is invisible to the main program, which clanks along happily without even noticing the interruptions.

The interrupt mechanism works all the time, even when Basic isn't running. If you add your own machine language program to the interrupt sequences, it too will work all the time — sixty times a second. It's ideal for watching special input/output ports, flashing parts of the screen, and similar jobs.

You can get at the interrupt routine quite easily. There is a memory location called the Hardware Interrupt Vector: in original ROMs, it's at hexadecimal 0219 and 021A; in new ROM systems it's at hex 0090 and 0091. In either case, the locations contain an address which points to the interrupt routine. If you change the address, the interrupt mechanism will go wherever you say, sixty times a second. At the end of your coding, don't forget to jump to the regular interrupt program so that the keyboard, clock, etc. still work properly.

Changing the address of the Hardware Interrupt vector has a small problem. Like all addresses, it comes in two chunks: a low order byte and a high order byte. If you have just changed the low order part and are about to change the second part when the interrupt strikes, you have a disaster on your hands. The address that the interrupt finds at that moment will be nonsense — part old address and part new.

Avoid this problem by making use of the SEI (Set Interrupt disable) instruction to lock out the interrupt while you are changing the vector. Don't forget to restore the interrupt with a CLI (Clear Interrupt disable) when you've finished putting the address in place.

It seems odd, but cassette tape can neither read or write after you have changed the interrupt vector from its usual address; and LOADs from disk may "hang" without saying READY. Be sure to make provision to restore the vector if you do much input or output.

Machine Language Monitor

In the Machine Language Monitor, you can type .G for Go and go directly to any machine language program you like. You will go with a direct jump (JMP) command, which means that the program is not treated as a subroutine. You can't get back with a return (RTS) instruction; instead, you will likely use a Break (BRK) command to reconnect with the monitor.

The Go command and associated BRK instructions are useful in debugging programs. After your program is written, replace several of the instructions in your program with Break commands. Try to scatter the Break commands evenly throughout your program, especially at the start of logical program "modules". Now perform Go to the start of your program. You should come back to the monitor almost instantly with the first Break point. If so, you've reached that program step safely; replace the Break instruction with the command that originally belonged there. Now you can Go to that address, and the program will resume and continue to the next Break. As you go through the program piece by piece, check that the registers contain the values you expect; if appropriate, check key memory locations, too

If the PET misbehaves or goes terribly quiet, at least you will have isolated the portion of the program that is doing it to you. On the next test, you can set your break points closer together in that area, and pin the problem down step by step.

Summary

There are several ways to link your PET to machine language programs. Beginners will want to stay with the SYS command and the USR function until they have gained confidence. They should learn the Machine Language Monitor (.G) and Break (BRK) functions as quickly as possible to help in checking out programs.

The advanced functions — wedge and interrupt — will be there when they are needed. ©

Odds & Ends on the 2040 Disk

Jim Butterfield

The disk has almost more brains than the PET. It contains two separate microprocessors, each of which has its own ROM program; the micros talk to each other via a shared block of memory.

* * * * * * * * * *

A Thirteen Line BASIC Delete

Arthur C. Hudson
Ottawa, Ontario

Here is a short program written entirely in BASIC, which allows you to delete any group of lines from an existing program. Typically the increment is 1, so that all lines in the group are deleted, but this is not necessary.

To use the Basic Delete, just screen merge it with your existing program. Hopefully no conflict of line numbers will occur, if there is conflict, then some renumbering will be required. After the merge, RUN7878, and as instructed, modify the listed line 7892 to define the start, the end and the increment. Then press return twice, and the delete process will begin. The line number being deleted is displayed and you may press BREAK (RUN/STOP) at any time.

As an example of using an increment other than unity, you could write all or part of a program using even numbers for the useful statements and odd numbers for the remarks. Save on tape or disk, and then automatically delete all the remarks and save again. Finally the original can be brought back, and all even numbered statements deleted. This gives a program consisting only of the remarks. Each of these three versions can have its uses.

Somewhat complicated programming techniques are used here, and the statements must be entered carefully. Note that after you have modified the automatically listed line 7896 and pressed RETURN, the cursor will rest on a direct statement, RUN7882. In this way, a second RETURN will initiate the delete process.

The program uses the dynamic keyboard feature of the PET. (See **COMPUTE!** Issue 4 page 58 and the earlier reference - Louder - cited therein). It uses bins 834 and 835 in the second cassette buffer, but this does not prevent use of the second cassette.

One of the more interesting problems in this type of program is that PET suffers from amnesia the moment it executes a delete (all variables set to 0). It is for this reason that parameters have to be embedded in a program statement, and also N, the number of the line currently being deleted, must be poked into memory before the deletion and retrieved after it.

Note that in line 7892 the word 'INCREMENT' is spelled incorrectly. Don't try to fix it, or PET will see thé word REM inside it and bomb out. Don't think you can get away with substituting 'step' for 'incrment', because PET will object to the use of ST, a reserved word. Finally don't try incr'ment, PET doesn't like this either, (not alphanumeric).

The first time that the Basic Delete is used, the asterisks in SN7896 will be replaced by numbers. There is of course no need to replace the asterisks when executing a SAVE.

My version of this program uses about 330 bytes. It is certainly possible to trim this down by about 50 bytes.

If You Have OLD ROM

Referring to statements numbered 7884 and 7886; for 623 and 624 substitute 527 and 528. For 158 substitute 525. These bins relate to the keyboard buffer. Note that Harvey Davis's article is written for old ROM, so the conversions given above apply in reverse, if you have new ROM.

Reference:

Algebraic Input for the PET, Harvey Davis, **COMPUTE!** Vol. 1, Issue 4, page 58.

```
 10 PRINT"ĥ↓A THIRTEEN LINE BASIC DELETE
 12 PRINT"↓ARTHUR C. HUDSON
 14 PRINT"↓11 AMBERLY PLACE
 16 PRINT"↓OTTAWA,ONT.
 18 PRINT"↓CANADA
 20 PRINT"↓K1J 7J9
 22 PRINT"↓PHONE (613) 749 5475
 30 PRINT"↓↓KEY IN CONT":STOP
 7878 PRINT"ĥ→MODIFY SN7896,THEN CR
 7880 PRINT"↓↓RUN7882↑↑↑":LIST7896
 7882 POKE835,0:POKE834,0:GOTO7894
 7884 POKE623,13:POKE624,13
 7886 POKE158,2:PRINT"ĥ↓↓↓GOTO7894
 7888 PRINT"h↓↓"N"↑↑↑";:N=N+IN
 7890 D=INT(N/256):POKE835,D
 7892 POKE834,N-D*256:END
 7894 N=256*PEEK(835)+PEEK(834)
 7896 FIRST=0000:LAST=0000:INCRMENT=01
 7898 IF N > LA THEN STOP
 7900 IF N < FI THEN N=FI
 7902 GOTO7884
READY.
```

©

Calculated Bar-graph Routines On The PET

Edward F. Heite
Camden-Wyoming, Delaware

To exploit the CBM printer's graphic potentials, programmers need a few routines that haven't been published yet. The "Keyprint" program (**COMPUTE!**, issue 7, page 84) is okay for dumping 40 columns to the printer, if you have the right ROM. But if you want to manipulate the full 80 columns and create complicated graphs, you need a way to calculate the length of the bar.

A calculated bar can be created as a string variable, by concatenating a graphic string to the desired length with a FOR ... NEXT loop. Listing 1 is a dummy program to demonstrate this process.

Line 1 sets B$ to an empty value. Line 2 defines C$ as a single graphic character. In line 3, the value of the bar is set at 20; in actual programs, this would be a calculated value. Line 4 sets the FOR ... NEXT loop to the value of J, and thus determines the length of the bar. Line 5 concatenates B$—C$ to create a new value for B$. Line 5 keeps adding symbols to B$ until the loop reaches the value of J. After the loop has cycled the required number of times, B$ is a bar of length J, which in this case is 20.

In normal program use, a series of these routines would create the bars. Then the printer routine would use them in a report; lines 7-9 are a typical printer routine.

LISTING 1

```
1 B$=""
2 C$="▒"
3 J=20
4 FOR X=1 TO J
5 B$=C$+B$
6 NEXT X
7 OPEN 1,4,0
8 PRINT#1, B$
9 CLOSE 1
```

©

The Revised Pet/CBM Personal Computer Guide

Jim Butterfield
Toronto, Canada

This article deals with the changes and new features of the well-known Osborne/McGraw-Hill guide. As such, it isn't a full scale review. Many PET/CBM users are familiar with the first edition; it was the first truly comprehensive user guide for their machine. As such, they were less likely to complain about its faults, which were few, and more likely to be thankful that such a book finally existed.

A Stronger Style

The new edition is a major revision. The previous casual, almost folksy style ("Assuming you have just brought your PET home in a box, you must unpack it") has been replaced by a much tougher down-to-business style. The name PET has been almost universally replaced by CBM. The new book socks it to you with a much more hard-hitting style.

The organization of the book is stronger. Chapters have been reorganized, and additional Basic programming material inserted. There's a stronger grouping of data with headings, subheadings and detail. The Preface suggests, "Even if you have never programmed a computer before, this book will teach you how to write your own Basic programs ... Chapters 4, 5 and 8 teach BASIC programming." That's 190 pages of Basic material, the last 50 of which are essentially reference. It may be rather too terse for many learners, but it's all there.

File Foulup?

The book covers the newer 4.0 ROM system. This is quite a feat considering how recent this system is. Unfortunately, some of this new material appears to have been prepared hastily.

The new Relative data files are discussed, but the book gets the whole thing wrong. It would be well for readers to stay away from this section entirely: relative files are easy to handle, but not in the manner the book suggests. It seems that the authors have confused the carriage return character with the IEEE-488 EOI line; somehow the comma gets dragged in as a field delimiter and we end up with a mess. Worse and worse: playing with the comma makes numeric file variables difficult to handle, and we end up with pages of explanation on how to cope with this. It would have been so easy if we'd started off on the right foot: for writing, one PRINT# statement writes one record; for reading, EOI (as detected in the ST value) signals the last field within a record. And no commas, please.

Appendices

The tables in Appendix A do a fair job of trying to sort out the various codes used by the PET. Between screen formats, PET ASCII, true ASCII and keywords, they take some unscrambling.

For a book which makes some effort to be up-to-date on such things as 4.0 machines, I was surprised to see the out-of-date list of CBM newsletters and references given in appendix D. The PET Gazette and PET User Notes were still listed, and there was no sign of **COMPUTE!** magazine. And I really thought that Commodore Canada's excellent Transactor should have been on the list.

Table F-3 near the end of the book is a curious piece of work. It seems that the authors got hold of the symbol table from Commodore/Microsoft's assembly and sorted and printed it for both Upgrade and 4.0 ROMs. It's fascinating: I suspect that it

PETTERM™

Terminal Program

Intelligent Communication

Uploads and Downloads

In Machine Language

Use your PET as an intelligent terminal. Access timeshare systems and networks such as the Source or Telenet. This is an interrupt driven system with buffered input/output. Unlike dumb terminals, the PET can do other tasks instead of having to wait for incoming data. This flexibility allows many advanced features.

These are complete assembled hardware and software packages. All include line editing/resend, auto-repeat, shift lock, output to IEEE printer and much more. Delivered on cassette or disk. Also, available for Commodore 8010 modem or Livermore Star 488 modem.

PETTERM I All features above . . . . . . **$ 80.00**

PETTERM II All features of I. plus local text editor with down-loading capability **90.00**

PETTERM III All features of II. plus 80 132 column scrolling window for viewing formatted outputs wider than 40 columns **100.00**

FORTH

A Programming Revolution

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

fig-FORTH. with editor. assembler **$50.00**
with floating point and strings **$60.00**
with floating point. strings. macro-assembler . **$70.00**
cassette and disk interface available.

DEALERS INQUIRE

F S S

software for small computers

1903 Rio Grande
Austin, Texas
78705

1-512-477-2207

P.O. Box 8403
Austin, Texas
78712

Source: TCE995

shows the original Commodore/Microsoft symbolic names for memory addresses: for example, the Floating Point Accumulator at hexadecimal 005E seems to be called FAC. But mixed in with these is a series of values which don't represent addresses at all. For example, hex 35 is the memory address of part of the top-of-Basic-memory pointer. But 35 is shown in the table as ERRFC, which happens to be the value loaded into the X register just before printing an ?ILLEGAL QUANTITY error message. Oddities in this computer-generated table: non-existent addresses are printed as 0000 rather than being left blank; and locations for which the authors apparently had no explanation are marked "X". It's a lovely table — I wish I could figure out why it's there.

Summing up.

Like its predecessor, the new book is a prodigious work. Its stronger style will improve its value as a reference, although some readers may miss the more casual approach of the first edition.

It's certainly the most comprehensive guide to using CMB/PET machines that is available today. The book is well organized and clearly written. It's generously fitted with examples, programs, diagrams and tables. Apart from the problems dealing with Relative files, the book is a sound approach to using the computer.

[**PET/CBM Personal Computer Guide**, Second Edition: by Adam Osborne and Carroll S. Donahue. Published by Osborne/McGraw-Hill, Berkeley, California.] ©

Un-Compactor

Robert W. Baker
Atco, NJ

Since my Compactor program was published in the Nov./Dec. '80 issue of **COMPUTE!**, I've had several requests for a companion program to un-Compact programs. The program shown here will do just that!

The program reads a BASIC program file from disk on drive 0 and creates a new copy on drive 1. The new program filename is the same as the original except for a "/U" suffix to indicate an un-compacted version. As with Compactor, load the newly created program file and enter a CLR command from the keyboard to correct the program links. Then save the program back to disk as usual. Un-compactor does not generate correct link values when writing the new program file, it merely writes a dummy value to reserve space for a link. This saves a fair amount of extra work not really needed in the program. The CLR command will force BASIC to correct the program links for you.

The program takes any multi-statement lines (statements separated by colons) and breaks them into separate program lines with new line numbers. The new line numbers are generated by adding one to the original line for each new line generated. This procedure is followed for however many statements exist in the line, as long as new line numbers can be generated without reaching the next line number in the original program. If that point is reached, the remainder of the original line is then copied as part of the last line generated with any appropriate separating colons.

The program must take into account certain BASIC tokens or keywords since they effect whether or not a particular line can be broken into separate lines. Thus, any data following a GOTO, END, RUN, IF, RETURN, REM, STOP, LIST, or CONT token is copied unchanged to the end of the current program line. Also, once a quote is detected, the line must be copied until another quote or end of the program line is reached.

Hope this proves to be of help, especially to those currently using Compactor. This program allows you to effectively re-create programs that were compacted. Now you can get a compacted program in Un-compactor to help speed up program execution. As usual, I'll supply copies of the program on cassette for $2 to cover costs.

```
 10 FOR X=1 TO 10
 11 PRINT X
 12 NEXT
 20 PRINT
 21 PRINT
 22 PRINT
 30 REM TEST FILE FOR UNCOMPACTOR
 40 A=1
 41 B=2
 42 C=3
 43 D=4
 44 E=5:F=6:G=7
 45 X=10
 46 Y=20
 47 Z=30
 100 END:THAT ALL!
READY.
```

- SAMPLE LISTING
OUTPUT FILE FROM UNCOMPACTOR

```
 10 FOR X=1 TO 10: PRINT X: NEXT
 20 PRINT:PRINT:PRINT
 30 REM TEST FILE FOR UNCOMPACTOR
 40 A=1:B=2:C=3:D=4:E=5:F=6:G=7
 45 X=10:Y=20:Z=30
 100 END:THAT ALL!
READY.
```

- SAMPLE LISTING
INPUT FILE TO UNCOMPACTOR

```
30 REM          UN - COMPACTOR
50 REM      BY: ROBERT W. BAKER
70 REM 15 WINDSOR DR., ATCO, NJ 08004
100 :
110 GOTO 270
120 :
130 REM >>>>>> SUBROUTINES <<<<<<<
140 :
150 GOSUB 160: V1=V
160 GET#5,C$: GOSUB 190
170 IF C$="" THEN V=0: RETURN
180 V=ASC(C$): RETURN
190 INPUT#15,EN,EM$,ET,ES
200 IF EN=0 THEN RETURN
210 PRINT "ĥr̲DISK ERROR":PRINT
220 PRINT EN;EM$;ET;ES
230 GOTO 1030
240 :
250 REM ***** INITIALIZATION *****
260 :
270 PRINT"ĥ";SPC(10);"r̲UN-COMPACTOR↓↓
280 PRINT" r̲INPUTf̂ FILE IN r̲DRIVE #0↓
290 PRINT"r̲OUTPUTf̂ FILE IN r̲DRIVE #1↓↓
300 INPUT"r̲INPUT FILENAMEf̂";FL$
310 DIM C(256)
320 OPEN 15,8,15
330 OPEN 5,8,5,"0:"+FL$+",P,R"
340 GOSUB 190
350 PRINT:PRINT"OK, WORKING ON LINE# ¬
     ¬.....↓
360 FO$=LEFT$(FL$,14)+"/U"
```

```
370 PRINT#15,"S1:"+FO$
380 OPEN 6,8,6,"1:"+FO$+",P,W"
390 GOSUB 190
400 GOSUB 150: PRINT#6,CHR$(V1);C$;
410 F=1: GOTO 580
420 :
430 REM ***** OUTPUT THIS LINE#
440 :
450 LN=NL: IF LK=0 THEN 1010
460 PRINT LN,
470 PRINT#6,CHR$(1);CHR$(1);
480 PRINT#6,CHR$(LL);CHR$(LH);
490 :
500 REM ***** READ THIS BASIC PGM LINE
510 :
520 X=1
530 GOSUB 160: C(X)=V
540 IF V>0 THEN X=X+1: GOTO 530
550 :
560 REM ***** GET NEXT LINK & LINE#
570 :
580 GOSUB 150: LK=V+V1: IF LK=0 THEN 600
590 GOSUB 150: NL=V1+(256*V): LL=V1:
     ¬ LH=V
600 IF F THEN F=0: GOTO 450
610 :
620 REM ***** BREAK UP LINE IF POSSIBLE
630 :
640 X=1
650 :
660 REM SKIP IF NOT COLON
670 :
680 IF C(X)<>58 THEN 810
690 IF X=1 THEN 950
700 LN=LN+1: IF LN>=NL THEN 950
710 PRINT#6,CHR$(0);CHR$(1);CHR$(1);
720 H=INT(LN/256): L=LN-(256*H)
730 PRINT#6,CHR$(L);CHR$(H);
740 X=X+1: IF C(X)=32 OR C(X)=58 THEN ¬
     ¬740
750 GOTO 680
760 :
770 REM COPY REST OF LINE IF ---
780 REM  GOTO, END, RUN, IF, RETURN
790 REM  REM, STOP, LIST, CONT
800 :
810 IF C(X)<128 OR C(X)>155 THEN 910
820 IF C(X)=128 OR C(X)>153 THEN 850
830 IF C(X)<137 OR C(X)>144 THEN 910
840 IF C(X)=140 OR C(X)=141 THEN 910
850 PRINT#6,CHR$(C(X));
860 IF C(X)>0 THEN X=X+1: GOTO 850
870 GOTO 450
880 :
890 REM SKIP IF NOT QUOTE
900 :
910 IF C(X)<>34 THEN 950
920 PRINT#6,CHR$(C(X)); : X=X+1
930 IF C(X)=34 OR C(X)=0 THEN 950
940 GOTO 920
950 PRINT#6,CHR$(C(X));
960 IF C(X)>0 THEN X=X+1: GOTO 680
970 GOTO 450
980 :
990 REM *** END OF BASIC PROGRAM
1000 :
1010 PRINT#6,CHR$(0);CHR$(0);
1020 PRINT"hrDONE":PRINT:PRINT
1030 CLOSE 5: CLOSE 6: CLOSE 15
READY.
```

©

Using the Hardware Interrupt Vector on the Pet

Eric Brandon

The operating system of the PET is divided into several distinct parts. Some of these get and process your BASIC statements; others deal with all Input/Output operations, and some update the clock, flash the cursor and take care of other sundry details every 60th of a second. This article will show you how to change the operation of the latter to suit your own needs.

Every 60th of a second the PET gets a signal on its IRQ interrupt. When this occurs, it saves all registers and goes to the memory locations specified in locations 537 and 538 (144 and 145 on new ROMs). It executes the machine language program there, and upon hitting an RTI instruction, reloads all of its registers and continues with whatever it was doing. By changing the hardware Interrupt Vector at 537 and 538 (144 and 145 new ROMs) we can make the PET execute our program every 60th of a second, while BASIC operates normally.

I have included here two sample programs using this technique, one is for ROM 2.0 (old ROMs) and the other is for ROM 3.0. What they do, is that after a SYS 826, the contents of the BASIC Input Buffer are constantly displayed on the top two lines of the screen. Hitting the ampersand (&), BREAKs the machine language program, and hitting the slash to the right of the ampersand on the keyboard, clears the buffer if you find that it is getting too cluttered. These programs were written only as examples of using the Hardware Interrupt Vector and are meant to show you how to use this with your own programs. Before we proceed, I wish to clarify just one feature of my assembler; the plus sign on lines 11 and 37 means add one to the value of the symbol. On most assemblers this should be substituted with HIV 1.

Lines 2-6 simply set the values of some symbols. INBUF is the first memory location of the BASIC Input Buffer. KEY is the location that contains the keyboard matrix value of the key presently depressed. INTRPT is the routine to which the Hardware Interrupt Vector usually points. HIV is the location of the first byte of the two byte Hardware Interrupt Vector. Finally, SCRN is the top lefthand corner of the screen.

Lines 7-13 are essential and should be looked at in detail. Line 7 has the Set Interrupt Mask instruction. This is necessary to prevent the PET from being interrupted with only one byte of the pointer changed. Line 12 clears the interrupt mask. If the mask wouldn't be cleared, the PET would "hang up" and need to be turned off. Lines 8–11 make the pointer point to 0347 (0345 new ROMs). Note that the least significant byte goes in 537 (144 new ROMs), and that the most significannt byte goes in 538 (145 new ROMs). The RTS in line 13 returns you to BASIC after your SYS. The effect of an SYS 826 is to make the cursor reappear nearly immediately, but now the PET executes the machine language program at 0347 (0345 new ROMs) every 60thy of a second. The actual operation of the program is quite straightforward to anyone familiar with machine language programming.

Lines 33–39 are the standard procedure for setting the Hardware Interrupt Vector back to normal. Note that POKE 537,133:POKE 538,230 (POKE 144,46:POKE 145,230 new ROMs) has the same effect. This procedure must be done before any cassette I/O.

The last thing that deserves notice are lines 25,32, and 39. The only safe way to leave a program that has been called by the Hardware Interrupt Vector is to jump somewhere into the interrupt handling routine. Since it begins at E685 (E62E new ROMs), that is where you will most often go. You <u>cannot</u> end your program with a RTS or a BRK.

I learned this technique from disassembling KEYPRINT by Charles Brannonn, a program in a previous issue of **COMPUTE!**. I hope you find this useful, and if you have any questions, you can write me at:

Eric Brandon
36 Hartfield Road
Islington, Ontario
Canada
M9A 3C9

```
INPBUF =   $0200
KEY =      $0097
INTRPT =   $E62E
HIV =      $0090
SCRN =     $8000
START =    $0345
LOOP =     $0351
CLEAR =    $035F
LOOP2 =    $0363
NORMAL =   $036E
                         *       =    $33A
                         INPBUF  =    $200
                         KEY     =    151
                         INTRPT  =    $E62E
                         HIV     =    144
                         SCRN    =    $8000
      033A 78                    SEI
      033B A9 45                 LDA  #$45
      033D 85 90                 STA  HIV
      033F A9 03                 LDA  #$03
      0341 85 91                 STA  HIV+
      0343 58                    CLI
      0344 60                    RTS
      0345 A5 97         START   LDA  KEY
      0347 C9 45                 CMP  #69
      0349 F0 14                 BEQ  CLEAR
      034B C9 4D                 CMP  #77
      034D F0 1F                 BEQ  NORMAL
      034F A2 00                 LDX  #0
      0351 BD 00 02      LOOP    LDA  INPBUF,X
      0354 9D 00 80              STA  SCRN,X
      0357 E8                    INX
      0358 E0 50                 CPX  #80
      035A D0 F5                 BNE  LOOP
      035C 4C 2E E6              JMP  INTRPT
      035F A2 00         CLEAR   LDX  #0
      0361 A9 20                 LDA  #32
      0363 9D 00 02      LOOP2   STA  INPBUF,X
      0366 E8                    INX
      0367 E0 50                 CPX  #80
      0369 D0 F8                 BNE  LOOP2
      036B 4C 2E E6              JMP  INTRPT
      036E 78            NORMAL  SEI
      036F A9 2E                 LDA  #$2E
      0371 85 90                 STA  HIV
      0373 A9 E6                 LDA  #$E6
      0375 85 91                 STA  HIV+
      0377 58                    CLI
      0378 4C 2E E6              JMP  INTRPT
```

NEW ROM (3.0) VERSION

```
INPBUF =   $000A
KEY =      $0203
INTRPT =   $E685
HIV =      $0219
SCRN =     $8000
START =    $0347
LOOP =     $0354
CLEAR =    $0361
LOOP2 =    $0365
NORMAL =   $036F
                         *       =    $33A
                         INPBUF  =    $0A
                         KEY     =    515
                         INTRPT  =    $E685
                         HIV     =    537
                         SCRN    =    $8000
      033A 78                    SEI
      033B A9 47                 LDA  #$47
      033D 8D 19 02              STA  HIV
      0340 A9 03                 LDA  #$03
      0342 8D 1A 02              STA  HIV+
      0345 58                    CLI
      0346 60                    RTS
      0347 AD 03 02      START   LDA  KEY
      034A C9 45                 CMP  #69
      034C F0 13                 BEQ  CLEAR
      034E C9 4D                 CMP  #77
      0350 F0 1D                 BEQ  NORMAL
      0352 A2 00                 LDX  #0
      0354 B5 0A         LOOP    LDA  INPBUF,X
      0356 9D 00 80              STA  SCRN,X
      0359 E8                    INX
      035A E0 50                 CPX  #80
      035C D0 F6                 BNE  LOOP
      035E 4C 85 E6              JMP  INTRPT
      0361 A2 00         CLEAR   LDX  #0
      0363 A9 20                 LDA  #32
      0365 95 0A         LOOP2   STA  INPBUF,X
      0367 E8                    INX
      0368 E0 50                 CPX  #80
      036A D0 F9                 BNE  LOOP2
      036C 4C 85 E6              JMP  INTRPT
      036F 78            NORMAL  SEI
      0370 A9 85                 LDA  #$85
      0372 8D 19 02              STA  HIV
      0375 A9 E6                 LDA  #$E6
      0377 8D 1A 02              STA  HIV+
      037A 58                    CLI
      037B 4C 85 E6              JMP  INTRPT
```

OLD ROM VERSION ©

Odds & Ends on the 2040 Disk

Jim Butterfield

The disk's ID is written over 600 times. There's no quick way of changing it — just copy everything over to a new disk if you really need to do this.

* * * * * * * * *

Odds & Ends on the 2040 Disk

Jim Butterfield

Yes, the disk checks every read by using a checksum. You can depend on a good read being correct.

* * * * * * * * * *

Pet As An IEEE-488 Logic Analyzer

Jim Butterfield
Toronto, Canada

If you'd like to see what's going on on the GPIB — and if you can borrow an extra PET and IEEE interface cable — this program will help.

It shows the current status of four of the GPIB control lines, plus a log of the last nine characters transmitted on the bus.

The four control lines are NRFD, NDAC, DAV, and EOI. It would be nice to show ATN too, but I couldn't fit this in: it's detected in a rather odd way in the PET so that fitting it in is rather too tricky for this simple program.

The last nine characters are shown in "screen format". This means that you'll have to do a little translation work to sort out what some of them mean. On the other hand, it allows you to see characters that otherwise wouldn't be printed. A carriage return, for example, shows up as a lower case m; this is a little confusing at the start, but you'll quickly get used to it and it's handy to see everything that goes through. Don't forget that original model PETs may show upper and lower case reversed.

I had hoped to show which characters were accompanied by the EOI signal. It turned out that time is critical — the bus works very fast — and that adding this feature would cut down the number of displayed characters from nine to five. I opted for the bigger count, and dropped the EOI log feature.

The high speed of the bus makes it difficult to watch the control lines in real time. When the "active" PET is exchanging information with disk or printer everything is happening very fast, and the "logic analyzer" PET will show an amazing flurry of activity on the control lines. Only when the activity stops or hangs up will you be able to see the lines in their static conditions.

You may use the program to chase down real GPIB problems, or just to gain insight on how the bus works. Either way, it will come in handy if you can borrow that extra PET unit.

Even at the speed of program operation, a few signals come too fast to catch on the fly. If you must see everything in the select and unselect sequences, you'll have to cut down the number of characters displayes. Try changing the contents of $04F0 to, say, 5 if you want to do this.

```
                          ; IEEE WATCH   JIM BUTTERFIELD
110:   04B0                          *=    $4B0
120:   04B0              DFLAG       =     $B1
130:   04B0              DNNSAV      =     $B2
140:   04B0              EOISAV      =     $B3
200:   04B0 46 B1        START       LSR   DFLAG
210:   04B2 78                       SEI
220:   04B3 AD 12 E8 MAIN            LDA   $E812
230:   04B6 C9 EF                    CMP   #$EF
240:   04B8 D0 02                    BNE   CONT
250:   04BA 58                       CLI
250:   04BB 60                       RTS
288:   04BC AC 10 E8 CONT            LDY   $E810       ;EOI
290:   04BF AD 40 E8                 LDA   $E840       ;DAV, NRFD, NDAC
300:   04C2 AE 20 E8                 LDX   $E820       ;DATA
310:   04C5 29 C1                    AND   #$C1        ;EXTRACT BITS
320:   04C7 C5 B2                    CMP   DNNSAV
330:   04C9 D0 11                    BNE   DNN
340:   04CB 98                       TYA
350:   04CC 29 40                    AND   #$40        ;EXTRACT EOI
360:   04CE 0A                       ASL   A
370:   04CF 49 A0                    EOR   #$A0
380:   04D1 C5 B3        EOI         CMP   EOISAV
390:   04D3 F0 DE                    BEQ   MAIN
400:   04D5 85 B3                    STA   EOISAV
410:   04D7 8D 61 80                 STA   $8061
420:   04DA D0 D7                    BNE   MAIN
                          ;ACTIVITY SEEN - UPDATE SCREEN
430:   04DC 85 B2        DNN         STA   DNNSAV
```

```
04DE 29 80              AND  #$80
04E0 49 A0              EOR  #$A0
04E2 8D 52 80           STA  $8052
04E5 10 1D              BPL  NDAV       ;NO DAV SEEN
04E7 A4 B1              LDY  DFLAG
04E9 30 1B              BMI  DCONT      ;DAV SEEN BEFORE
04EB 85 B1              STA  DFLAG
04ED 85 B2              STA  DNNSAV
04EF A0 00              LDY  #0
04F1 B9 A2 80 SCROL     LDA  $80A2,Y
04F4 99 A1 80           STA  $80A1,Y
04F7 C8                 INY
04F8 C0 08              CPY  #8
04FA D0 F5              BNE  SCROL
04FC 8A                 TXA
04FD 49 FF              EOR  #$FF
04FF 8D A9 80           STA  $80A9
0502 B0 AF              BCS  MAIN
0504 85 B1    NDAV      STA  DFLAG
0506 A5 B2    DCONT     LDA  DNNSAV
0508 29 40              AND  #$40       ;NRFD
050A 0A                 ASL  A
050B 49 A0              EOR  #$A0
050D 8D 57 80           STA  $8057
0510 A5 B2              LDA  DNNSAV
0512 29 01              AND  #$1        ;NDAC
0514 4A                 LSR  A
0515 6A                 ROR  A
0516 49 A0              EOR  #$A0
0518 8D 5C 80           STA  $805C
051B D0 96              BNE  MAIN
```

```
 10 REM IEEE WATCH
 20 REM JIM BUTTERFIELD
 30 POKE59468,14:PRINT"□ DAV NRFD
NDAC  EOI":PRINT"  ↑      ↑      ↑      ↑□"
 40 PRINT"=123456789=□"
 50 SYS1200
READY.
```

©

Odds & Ends on the 2040 Disk

Jim Butterfield

Yes, the disk checks every write by reading the block back and verifying every byte for correctness. You can depend on a good write having gotten to the diskette correctly.

* * * * * * * * * *

Odds & Ends on the 2040 Disk

Jim Butterfield

The first files written to disk will cluster around track 18, the directory track. This minimizes head movement on a lightly-used disk. By the same token, you might arrange to write your most-used programs and files first on the disk, to save both time and wear and tear.

* * * * * * * * * *

Running 40 Column Programs On A CBM 8032

Chuan Chee
St. Catherine's
Ontario Canada

Good news for those who own a Commodore 80 column CBM. I have developed a method of making the computer act almost like a 40 column PET.

Over the years, many programs have been developed for PET 2001 computers. There had been one ROM upgrade from BASIC 1.0 to 2.0 but many people and software companies got over that hurdle. Now Commodore has introduced a BASIC 4.0 for their PET 40XX and CBM 80XX computers. Again many programmers must change any SYS commands into the ROM locations. However, some programs can still run on the PET 40XX because the programmers were careful enough to avoid any of the ROM routines; especially the BASIC part as opposed to the Operating System. Luckily, most of the first 1024 bytes remained the same as promised by Commodore.

80 Column Problem

But hold on before you start attacking your programs, the CBM 80XX is a completely different animal — it has an 80 column CRT (or screen). All the programs are assuming that there are 40 bytes per line as in the case of a PET, but a CBM has 80. Therefore, any programs that store characters on the CRT memory will have every other line on columns 41 to 80. This is certainly a dissaster.

The Solution

In solving this problem, there must be some way of convincing the microcomputer that there are only 40 bytes per line as in the PET. Commodore was wise enough to implement their newly developed Video Interface Controller (or CRT Controller) into the CBM. They are also using this chip in the VIC 20 (Video Interface Computer). When the power is turned on, the operating system instructs the chip to do various functions such as the height of the 25 lines in normal or graphics modes. My program instructs the Controller to display 40 bytes per line and shift the first column to the right to center the display instead of being on the left side of the CRT.

That is just fine for the programs that store characters on the CRT. But what about those that simply PRINT. Now whenever the PRINT finishes a line (40 characters) of output, the ROM routines will make the next PRINT occur 80 bytes from the start of the current PRINT line. This will make the output appear on every other line.

Well, there just happens to be an "Output a byte on the CRT" jump vector at locations $00EB to $00EC. The CBM 4032 program will change this vector to intercept any character before it gets printed. The routines included in the program were modified from a PET 4032 Operating System ROM, set so that it will behave exactly like a 40 column PET. It will handle RETURN, cursor movements, INST, DEL, and even wrap around lines properly.

Bonuses Not Available On A Pet

There are several features that make this simulation of a PET 40XX even better since they are not available on any PET computer. Such bonuses include the automatic repeat of the cursor control and editing keys and the use of the REPEAT key with all other keys. There will also be the usual warning "bell" when six characters from the end of the line. To disable the "bell", type POKE 231.0. This RAM location contains the duration time of the "bell" which usually is 16. Try poking various values and notice how the duration changes.

I also decided to keep the functions of "◊" and ":" during scrolling the same as before because those who are used to them should not have to use the RVS key. Along the same lines, the ESC key is still fully functional. By the way, did you know that the ESC key not only gets you out of quote and insert modes but also the reverse mode, thus functioning similar to the OFF key?

The CBM 4032 Program

The program is in two parts — a BASIC and a data part. After turning on the computer or typing NEW, type in the BASIC part exactly as shown without any extra spaces. LIST it again to be sure. Next, get into the Machine Language Monitor by SYS4 and type in the data, making sure not to make any mistakes. The next important step is to save the program through the Monitor by .S "0:CBM 4032",08,0400,07A8 for a disk drive or .S "CBM 4032",01.0400,07A8 for a tape cassette drive. Now exit the Monitor and prepare to RUN the program.

The data is actually the machine language routines required. The BASIC portion will transfer it into the second half of the 2K CRT memory. As it transfers the data, you will see "garbage" appear on the CRT. This is an ideal spot to put the routines because the CRT will only use 40 bytes per line by 25 lines (= 1000 bytes), the second half of the CRT memory will never be printed on.

After the transfer, the BASIC portion will SYS 33876 ($8454) to have the routines set up the necessary parameters. It will give the CRT Controller the proper instructions and then CLR the CRT. A READY. will appear on the CRT and control is returned to the user. Now you are ready to RUN any programs meant for a 40 column PET with the proper ROM charges if necessary.

If for any reason you wish to go back to the original 80 column format, you can switch off and on the CBM. Alternatively, you can type SYS 58982 ($E666) and press both SHIFTs and the quote keys simultaneously. The latter method will again display the data on the second half of the CRT but you risk printing or typing over it.

Conclusion

Essentially, any program that can RUN on a PET 40XX, that is with BASIC 4.0, will work with this program. There is no need to alter the program to add anything extra to the programs to artificially perform what this program does. The only side effect is that the characters appear narrower than usual but the advantage of having the program displayed far exceeds this small deviation.

I would like to thank **Batteries Included**, in Toronto for allowing me to use their computers for the development of this program.

```
10 REM * CBM 4032 - BY CHUAN CHEE *
20 REM SEE ARTICLE IN COMPUTE!
30 A=32672:FORI=1136TO1998:
   POKEI+A,PEEK(I):NEXT:SYS33876
READY.

C*
     PC   IRQ  SR AC XR YR SP
.;   B780 E455 34 33 38 36 FA
.
.:   0470 31 14 1F 0F 28 05 19 21
.:   0478 00 07 00 00 10 00 00 00
.:   0480 00 00 00 28 50 78 A0 C8
.:   0488 F0 18 40 68 90 B8 E0 08
.:   0490 30 58 80 A8 D0 F8 20 48
.:   0498 70 98 C0 20 53 62 7D 80
.:   04A0 94 A0 B3 C2 20 02 19 20
.:   04A8 03 08 15 01 0E 20 03 08
.:   04B0 05 05 20 20 78 A9 6F A2
.:   04B8 84 85 EB 86 EC 86 A7 58
.:   04C0 20 75 84 A2 00 86 A7 A9
.:   04C8 10 A2 84 20 86 E0 60 20
.:   04D0 4F 85 4C 9D E1 A0 83 A2
.:   04D8 18 98 9D 3B 84 E0 14 F0
.:   04E0 08 E0 0D F0 04 E0 07 D0
.:   04E8 01 88 CA 10 EC E8 86 9F
.:   04F0 86 C4 A9 20 9D 00 80 9D
.:   04F8 00 81 9D 00 82 9D 00 83
.:   0500 CA D0 F1 A0 00 84 C6 84
.:   0508 D8 A6 D8 BD 3B 84 09 80
.:   0510 85 C5 BD 22 84 85 C4 A9
.:   0518 27 85 D5 E0 18 F0 09 BD
.:   0520 3C 84 30 04 A9 4F 85 D5
.:   0528 A5 C6 C9 28 90 04 E9 28
.:   0530 85 C6 60 09 40 A6 9F F0
.:   0538 02 09 80 A6 DC F0 02 C6
.:   0540 DC 20 06 E6 E6 C6 A4 D5
.:   0548 C4 C6 B0 30 A6 D8 C0 4F
.:   0550 D0 0B 20 1D 85 20 67 86
.:   0558 A9 00 85 C6 60 E0 18 D0
.:   0560 09 20 8B 86 C6 A3 C6 D8
.:   0568 A6 D8 1E 3C 84 5E 3C 84
.:   0570 20 1D 85 A5 C6 48 20 A9
.:   0578 84 68 85 C6 60 E0 17 B0
.:   0580 08 BD 3D 84 09 80 9D 3D
.:   0588 84 60 A0 27 A6 D8 D0 05
.:   0590 86 C6 68 68 60 BD 3A 84
.:   0598 30 06 CA BD 3A 84 A0 4F
.:   05A0 CA 86 D8 85 C5 BD 22 84
.:   05A8 85 C4 84 C6 84 D5 60 A9
.:   05B0 00 85 AC A5 D9 29 7F C9
.:   05B8 1B D0 07 68 68 4C BD E3
.:   05C0 EA EA A4 C6 A5 D9 30 68
.:   05C8 C9 0D D0 03 4C 7E 86 C9
.:   05D0 20 90 08 29 3F 20 6A E1
.:   05D8 4C D5 84 A6 DC F0 03 4C
.:   05E0 D9 84 C9 14 D0 10 88 84
.:   05E8 C6 10 06 20 2A 85 4C 5C
.:   05F0 E2 68 68 4C 51 E2 A6 CD
.:   05F8 F0 03 4C D9 84 C9 12 D0
.:   0600 03 85 9F 60 C9 13 D0 03
.:   0608 4C A3 84 C9 1D D0 10 C8
.:   0610 84 C6 88 C4 D5 90 07 20
.:   0618 67 86 A9 00 85 C6 60 C9
.:   0620 11 D0 FB 18 98 69 28 C5
.:   0628 D5 90 F1 F0 EF 4C 67 86
.:   0630 29 7F C9 7F D0 02 A9 5E
.:   0638 C9 20 90 03 4C D3 84 C9
.:   0640 0D D0 03 4C 7E 86 A6 CD
.:   0648 D0 2F C9 14 D0 27 A4 D5
.:   0650 B1 C4 C9 20 D0 04 C4 C6
.:   0658 D0 07 C0 4F F0 16 20 ED
.:   0660 86 A4 D5 88 B1 C4 C8 91
.:   0668 C4 88 C4 C6 D0 F5 A9 20
.:   0670 91 C4 E6 DC 60 A6 DC F0
.:   0678 05 09 40 4C D9 84 C9 11
.:   0680 D0 2A A5 C6 C9 28 90 05
.:   0688 E9 28 85 C6 60 A6 D8 F0
.:   0690 FB BD 3A 84 10 07 C6 D8
.:   0698 20 A9 84 90 EF CA CA 86
.:   06A0 D8 20 A9 84 A5 C6 18 69
.:   06A8 28 85 C6 60 C9 12 D0 04
.:   06B0 A9 00 85 9F C9 1D D0 08
.:   06B8 88 84 C6 10 EE 20 2A 85
.:   06C0 C9 13 D0 E7 4C 75 84 38
.:   06C8 46 A3 A6 D8 E8 E0 19 D0
.:   06D0 03 20 8B 86 BD 3B 84 10
.:   06D8 F3 86 D8 4C A9 84 A9 00
.:   06E0 85 DC 85 9F 85 CD 85 C6
```

```
.:  06E8 4C 67 86 A0 00 84 C4 A9
.:  06F0 80 85 C8 85 C5 A9 28 2C
.:  06F8 3C 84 30 02 A9 50 85 C7
.:  0700 B1 C7 91 C4 C8 D0 F9 E6
.:  0708 C8 E6 C5 A9 84 C5 C8 D0
.:  0710 EF A9 E8 85 C4 C6 C5 A9
.:  0718 20 C6 C4 C6 C7 91 C4 D0
.:  0720 F8 A2 19 86 D8 A2 00 C6
.:  0728 D8 BD 3B 84 29 7F BC 3C
.:  0730 84 10 02 09 80 9D 3B 84
.:  0738 E8 E0 19 D0 EC A9 83 8D
.:  0740 53 84 AD 3B 84 10 DE 20
.:  0748 0B E4 A6 D8 60 A6 D8 E8
.:  0750 E0 18 F0 36 90 03 4C 01
.:  0758 85 A2 17 BD 3C 84 09 80
.:  0760 85 C8 BC 3B 84 30 02 29
.:  0768 7F 9D 3C 84 98 09 80 85
.:  0770 C5 A0 27 BD 23 84 85 C7
.:  0778 BD 22 84 85 C4 B1 C4 91
.:  0780 C7 88 10 F9 CA E4 D8 D0
.:  0788 D2 E8 BD 3B 84 09 80 85
.:  0790 C5 29 7F 9D 3B 84 BD 22
.:  0798 84 85 C4 A0 27 A9 20 91
.:  07A0 C4 88 10 FB 58 4C A9 84
.
.
READY.

    READY.

    C*
         PC  IRQ  SR AC XR YR SP
    .;  B780 E455 34 33 38 36 FA
    .
    8454 78          SEI
    8455 A9 6F       LDA #$6F
    8457 A2 84       LDX #$84
    8459 85 EB       STA $EB
    845B 86 EC       STX $EC
    845D 86 A7       STX $A7
    845F 58          CLI
    8460 20 75 84    JSR $8475
    8463 A2 00       LDX #$00
    8465 86 A7       STX $A7
    8467 A9 10       LDA #$10
    8469 A2 84       LDX #$84
    846B 20 86 E0    JSR $E086
    846E 60          RTS
    .
    .
    846F 20 4F 85    JSR $854F
    8472 4C 9D E1    JMP $E19D
    .
    .
    8475 A0 83       LDY #$83
    8477 A2 18       LDX #$18
    8479 98          TYA
    847A 9D 3B 84    STA $843B,X
    847D E0 14       CPX #$14
    847F F0 08       BEQ $8489
    8481 E0 0D       CPX #$0D
    8483 F0 04       BEQ $8489
    8485 E0 07       CPX #$07
    8487 D0 01       BNE $848A
    8489 88          DEY
    848A CA          DEX
    848B 10 EC       BPL $8479
    848D E8          INX
    848E 86 9F       STX $9F
    8490 86 C4       STX $C4
    8492 A9 20       LDA #$20
    8494 9D 00 80    STA $8000,X
    8497 9D 00 81    STA $8100,X
    849A 9D 00 82    STA $8200,X
    849D 9D 00 83    STA $8300,X
    84A0 CA          DEX
    84A1 D0 F1       BNE $8494
    .
    .
    84A3 A0 00       LDY #$00
    84A5 84 C6       STY $C6
    84A7 84 D8       STY $D8
    84A9 A6 D8       LDX $D8
    84AB BD 3B 84    LDA $843B,X
    84AE 09 80       ORA #$80
    84B0 85 C5       STA $C5
    84B2 BD 22 84    LDA $8422,X
    84B5 85 C4       STA $C4
    84B7 A9 27       LDA #$27
    84B9 85 D5       STA $D5
    84BB E0 18       CPX #$18
    84BD F0 09       BEQ $84C8
    84BF BD 3C 84    LDA $843C,X
    84C2 30 04       BMI $84C8
    84C4 A9 4F       LDA #$4F
    84C6 85 D5       STA $D5
    84C8 A5 C6       LDA $C6
    84CA C9 28       CMP #$28
    84CC 90 04       BCC $84D2
    84CE E9 28       SBC #$28
    84D0 85 C6       STA $C6
    84D2 60          RTS
    .
    .
    84D3 09 40       ORA #$40
    84D5 A6 9F       LDX $9F
    84D7 F0 02       BEQ $84DB
    84D9 09 80       ORA #$80
    84DB A6 DC       LDX $DC
    84DD F0 02       BEQ $84E1
    84DF C6 DC       DEC $DC
    84E1 20 06 E6    JSR $E606
    84E4 E6 C6       INC SC6
    84E6 A4 D5       LDY SD5
    84E8 C4 C6       CPY SC6
    84EA B0 30       BCS S851C
    84EC A6 D8       LDX SD8
```

```
84EE C0 4F       CPY #$4F
84F0 D0 0B       BNE $84FD
84F2 20 1D 85    JSR $851D
84F5 20 67 86    JSR $8667
84F8 A9 00       LDA #$00
84FA 85 C6       STA $C6
84FC 60          RTS
84FD E0 18       CPX #$18
84FF D0 09       BNE $850A
.
.
8501 20 8B 86    JSR $868B
8504 C6 A3       DEC $A3
8506 C6 D8       DEC $D8
8508 A6 D8       LDX $D8
850A 1E 3C 84    ASL $843C,X
850D 5E 3C 84    LSR $843C,X
8510 20 1D 85    JSR $851D
8513 A5 C6       LDA $C6
8515 48          PHA
8516 20 A9 84    JSR $84A9
8519 68          PLA
851A 85 C6       STA $C6
851C 60          RTS
.
.
851D E0 17       CPX #$17
851F B0 08       BCS $8529
8521 BD 3D 84    LDA $843D,X
8524 09 80       ORA #$80
8526 9D 3D 84    STA $843D,X
8529 60          RTS
.
.
852A A0 27       LDY #$27
852C A6 D8       LDX $D8
852E D0 05       BNE $8535
8530 86 C6       STX $C6
8532 68          PLA
8533 68          PLA
8534 60          RTS
8535 BD 3A 84    LDA $843A,X
8538 30 06       BMI $8540
853A CA          DEX
853B BD 3A 84    LDA $843A,X
853E A0 4F       LDY #$4F
8540 CA          DEX
8541 86 D8       STX $D8
8543 85 C5       STA $C5
8545 BD 22 84    LDA $8422,X
8548 85 C4       STA $C4
854A 84 C6       STY $C6
854C 84 D5       STY $D5
854E 60          RTS
.
.
854F A9 00       LDA #$00
8551 85 AC       STA $AC
8553 A5 D9       LDA $D9
8555 29 7F       AND #$7F
8557 C9 1B       CMP #$1B
8559 D0 07       BNE $8562
855B 68          PLA
855C 68          PLA
855D 4C BD E3    JMP $E3BD
8560 EA          NOP
8561 EA          NOP
8562 A4 C6       LDY $C6
8564 A5 D9       LDA $D9
8566 30 68       BMI $85D0
8568 C9 0D       CMP #$0D
856A D0 03       BNE $856F
856C 4C 7E 86    JMP $867E
856F C9 20       CMP #$20
8571 90 08       BCC $857B
8573 29 3F       AND #$3F
8575 20 6A E1    JSR $E16A
8578 4C D5 84    JMP $84D5
857B A6 DC       LDX $DC
857D F0 03       BEQ $8582
857F 4C D9 84    JMP $84D9
8582 C9 14       CMP #$14
8584 D0 10       BNE $8596
8586 88          DEY
8587 84 C6       STY $C6
8589 10 06       BPL $8591
858B 20 2A 85    JSR $852A
858E 4C 5C E2    JMP $E25C
8591 68          PLA
8592 68          PLA
8593 4C 51 E2    JMP $E251
8596 A6 CD       LDX $CD
8598 F0 03       BEQ $859D
859A 4C D9 84    JMP $84D9
859D C9 12       CMP #$12
859F D0 03       BNE $85A4
85A1 85 9F       STA $9F
85A3 60          RTS
85A4 C9 13       CMP #$13
85A6 D0 03       BNE $85AB
85A8 4C A3 84    JMP $84A3
85AB C9 1D       CMP #$1D
85AD D0 10       BNE $85BF
85AF C8          INY
85B0 84 C6       STY $C6
85B2 88          DEY
85B3 C4 D5       CPY $D5
85B5 90 07       BCC $85BE
85B7 20 67 86    JSR $8667
85BA A9 00       LDA #$00
85BC 85 C6       STA $C6
85BE 60          RTS
85BF C9 11       CMP #$11
85C1 D0 FB       BNE $85BE
85C3 18          CLC
85C4 98          TYA
85C5 69 28       ADC #$28
85C7 C5 D5       CMP $D5
```

```
85C9 90 F1       BCC $85BC
85CB F0 EF       BEQ $85BC
85CD 4C 67 86    JMP $8667
.
.
85D0 29 7F       AND #$7F
85D2 C9 7F       CMP #$7F
85D4 D0 02       BNE $85D8
85D6 A9 5E       LDA #$5E
85D8 C9 20       CMP #$20
85DA 90 03       BCC $85DF
85DC 4C D3 84    JMP $84D3
85DF C9 0D       CMP #$0D
85E1 D0 03       BNE $85E6
85E3 4C 7E 86    JMP $867E
85E6 A6 CD       LDX $CD
85E8 D0 2F       BNE $8619
85EA C9 14       CMP #$14
85EC D0 27       BNE $8615
85EE A4 D5       LDY $D5
85F0 B1 C4       LDA ($C4),Y
85F2 C9 20       CMP #$20
85F4 D0 04       BNE $85FA
85F6 C4 C6       CPY $C6
85F8 D0 07       BNE $8601
85FA C0 4F       CPY #$4F
85FC F0 16       BEQ $8614
85FE 20 ED 86    JSR $86ED
8601 A4 D5       LDY $D5
8603 88          DEY
8604 B1 C4       LDA ($C4),Y
8606 C8          INY
8607 91 C4       STA ($C4),Y
8609 88          DEY
860A C4 C6       CPY $C6
860C D0 F5       BNE $8603
860E A9 20       LDA #$20
8610 91 C4       STA ($C4),Y
8612 E6 DC       INC $DC
8614 60          RTS
8615 A6 DC       LDX $DC
8617 F0 05       BEQ $861E
8619 09 40       ORA #$40
861B 4C D9 84    JMP $84D9
861E C9 11       CMP #$11
8620 D0 2A       BNE $864C
8622 A5 C6       LDA $C6
8624 C9 28       CMP #$28
8626 90 05       BCC $862D
8628 E9 28       SBC #$28
862A 85 C6       STA $C6
862C 60          RTS
862D A6 D8       LDX $D8
862F F0 FB       BEQ $862C
8631 BD 3A 84    LDA $843A,X
8634 10 07       BPL $863D
8636 C6 D8       DEC $D8
8638 20 A9 84    JSR $84A9
863B 90 EF       BCC $862C
863D CA          DEX
863E CA          DEX
863F 86 D8       STX $D8
8641 20 A9 84    JSR $84A9
8644 A5 C6       LDA $C6
8646 18          CLC
8647 69 28       ADC #$28
8649 85 C6       STA $C6
864B 60          RTS
864C C9 12       CMP #$12
864E D0 04       BNE $8654
8650 A9 00       LDA #$00
8652 85 9F       STA $9F
8654 C9 1D       CMP #$1D
8656 D0 08       BNE $8660
8658 88          DEY
8659 84 C6       STY $C6
865B 10 EE       BPL $864B
865D 20 2A 85    JSR $852A
8660 C9 13       CMP #$13
8662 D0 E7       BNE $864B
8664 4C 75 84    JMP $8475
.
.
8667 38          SEC
8668 46 A3       LSR $A3
866A A6 D8       LDX $D8
866C E8          INX
866D E0 19       CPX #$19
866F D0 03       BNE $8674
8671 20 8B 86    JSR $868B
8674 BD 3B 84    LDA $843B,X
8677 10 F3       BPL $866C
8679 86 D8       STX $D8
867B 4C A9 84    JMP $84A9
.
.
867E A9 00       LDA #$00
8680 85 DC       STA $DC
8682 85 9F       STA $9F
8684 85 CD       STA $CD
8686 85 C6       STA $C6
8688 4C 67 86    JMP $8667
.
.
868B A0 00       LDY #$00
868D 84 C4       STY $C4
868F A9 80       LDA #$80
8691 85 C8       STA $C8
8693 85 C5       STA $C5
8695 A9 28       LDA #$28
8697 2C 3C 84    BIT $843C
869A 30 02       BMI $869E
869C A9 50       LDA #$50
869E 85 C7       STA $C7
86A0 B1 C7       LDA ($C7),Y
86A2 91 C4       STA ($C4),Y
```

```
86A4 C8          INY
86A5 D0 F9       BNE $86A0
86A7 E6 C8       INC $C8
86A9 E6 C5       INC $C5
86AB A9 84       LDA #$84
86AD C5 C8       CMP $C8
86AF D0 EF       BNE $86A0
86B1 A9 E8       LDA #$E8
86B3 85 C4       STA $C4
86B5 C6 C5       DEC $C5
86B7 A9 20       LDA #$20
86B9 C6 C4       DEC $C4
86BB C6 C7       DEC $C7
86BD 91 C4       STA ($C4),Y
86BF D0 F8       BNE $86B9
86C1 A2 19       LDX #$19
86C3 86 D8       STX $D8
86C5 A2 00       LDX #$00
86C7 C6 D8       DEC $D8
86C9 BD 3B 84    LDA $843B,X
86CC 29 7F       AND #$7F
86CE BC 3C 84    LDY $843C,X
86D1 10 02       BPL $86D5
86D3 09 80       ORA #$80
86D5 9D 3B 84    STA $843B,X
86D8 E8          INX
86D9 E0 19       CPX #$19
86DB D0 EC       BNE $86C9
86DD A9 83       LDA #$83
86DF 8D 53 84    STA $8453
86E2 AD 3B 84    LDA $843B
86E5 10 DE       BPL $86C5
86E7 20 0B E4    JSR $E40B
86EA A6 D8       LDX $D8
86EC 60          RTS
.
.
86ED A6 D8       LDX $D8
86EF E8          INX
86F0 E0 18       CPX #$18
86F2 F0 36       BEQ $872A
86F4 90 03       BCC $86F9
86F6 4C 01 85    JMP $8501
86F9 A2 17       LDX #$17
86FB BD 3C 84    LDA $843C,X
86FE 09 80       ORA #$80
8700 85 C8       STA $C8
8702 BC 3B 84    LDY $843B,X
8705 30 02       BMI $8709
8707 29 7F       AND #$7F
8709 9D 3C 84    STA $843C,X
870C 98          TYA
870D 09 80       ORA #$80
870F 85 C5       STA $C5
8711 A0 27       LDY #$27
8713 BD 23 84    LDA $8423,X
8716 85 C7       STA $C7
8718 BD 22 84    LDA $8422,X
871B 85 C4       STA $C4
871D B1 C4       LDA ($C4),Y
871F 91 C7       STA ($C7),Y
8721 88          DEY
8722 10 F9       BPL $871D
8724 CA          DEX
8725 E4 D8       CPX $D8
8727 D0 D2       BNE $86FB
8729 E8          INX
872A BD 3B 84    LDA $843B,X
872D 09 80       ORA #$80
872F 85 C5       STA $C5
8731 29 7F       AND #$7F
8733 9D 3B 84    STA $843B,X
8736 BD 22 84    LDA $8422,X
8739 85 C4       STA $C4
873B A0 27       LDY #$27
873D A9 20       LDA #$20
873F 91 C4       STA ($C4),Y
8741 88          DEY
8742 10 FB       BPL $873F
8744 58          CLI
8745 4C A9 84    JMP $84A9
.
.
READY.
```

©

THE SBC GAZETTE

Build Your Own Controllers
Nuts And Volts

Gene Zumchak

Part I

If you have a personal computer of any kind, you probably already appreciate the power of a general-purpose computer system to serve as a controller. While tying up your APPLE or PET to control the thermostat may not seem overly attractive, you can usually try out a control idea or scheme using your existing computer system and small amount of custom I/O. Eventually, however, you will want to dedicate a separate computer system to your controller application.

It wasn't so long ago that such a thought would be prohibitive. Computer systems were dream machines that cost several thousands of dollars. Of course, if you have a console type computer system which includes a CRT and perhaps one or more disks, then your console system can easily cost three or more thousand dollars. On the other hand, a great many controller applications require little more than a handful of chips that cost well under $100. In fact, if your application has any merit and a significant market, it may be quite possible to integrate the design into a single-chip microcomputer costing only a couple of dollars, and you can be on your way to making your first million.

While your particular application may never make you rich, it is fairly easy to put together a prototype or a one-of-a-kind microcontroller system for a reasonable price. A 6502 will cost less than $10. A 2716 will cost about the same. Figure $5 for a 128 x 8 RAM chip, (Motorola 6810), or $8 for a pair of 2114's for 1K of RAM. A 6522 for $8 will provide sixteen bits of I/O and a pair of timers (suitable for a real-time clock). Finally, a few more dollars for a crystal and some TTL for address decoding, and the electronic parts cost will come to not much more than $50.

If the parts really cost as little as mentioned, what's to prevent anyone with a little knowledge of computers from designing and building his own microcontrollers? The answer is absolutely nothing. But there is one small catch. While the cost of the end product may be minimal or even negligible, most companies or individuals who design microcomputer systems do it with the aid of a microprocessor "development system". Commercial development systems start at about $5,000, but typically range from $15,000 to $25,000.

In my book, **Microcomputer Design and Troubleshooting**, which is being typeset and will be in print in the Fall (Howard Sams, and the Blacksburg Series), I address the question of what comprises a typical development system, but more importantly, what is minimally required to put together your own low-budget development system. While the reader will want to read about the details in the book when it is published, the highlights of that discussion will be brought out here, in this first installment of several in which I will outline the procedures and equipment necessary to put together and bring up, your own microcomputer controllers.

The Development System

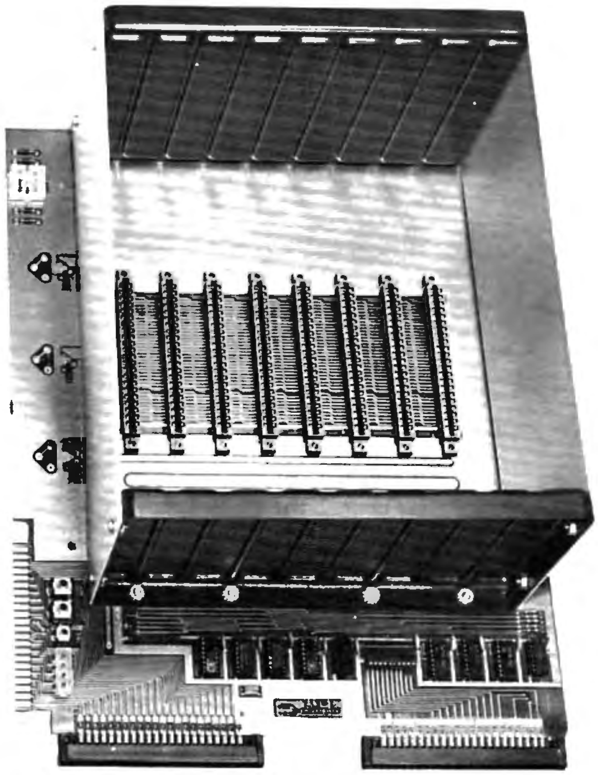
A development system is the hardware and software required to check out and debug both the hardware and software of a prototype microprocessor system. Ironically, the hardware and software debugging capabilities are not always reflected by the systems very high cost. Software debugging capabilities are usually satisfactory, provided that the system has an "optional" processor emulator module which typically costs $2,000 or more. Even with the emulator, the hardware debugging capabilities may be mediocre at best.

Typically, a commercial development system consists of the following items:

1. Microcomputer with software
2. Console device (CRT or Teletypewriter)
3. RAM memory blocks
4. Floppy disk(s)
5. Printer
6. EPROM programmer (with software)
7. Emulator (processor)

Why should such a system cost $15,000? The reason for the high cost is the law of supply and demand; there just aren't that many people in the world who need a microprocessor development system. However, except for some specialized software like an editor and assembler, the first five items in the list are not appreciably different than what you get in a BASIC oriented console computer system like a PET or APPLE. And, of course,

VAK-1 MOTHERBOARD



We also carry:

SYM-1	**$229.00**
AIM-65 w/1K	**389.00**
AIM-65 w/4K	**439.00**

We also do custom hardware and software for the 6502 microprocessor

PRICE: **$139.00**
Call or write for shipping charges and our complete catalog.

The VAK-1 was specifically designed for use with the KIM-1, SYM-1 and the AIM 65 Microcomputer Systems. The VAK-1 uses the KIM-4* Bus Structure, because it is the only popular Multi-Sourced bus whose expansion boards were designed specifically for the 6502 Microprocessor.

SPECIFICATIONS:

- Complete with rigid CARD-CAGE
- Assembled (except for card-cage). Burned in and tested.
- All IC's are in sockets
- Fully buffered address and data bus
- Uses the KIM-4* Bus (both electrical Pin-out and card size) for expansion board slots
- Provides 8 slots for expansion boards on 1″ centers to allow for wire-wrap boards
- Designed for use with a Regulated Power Supply (such as our VAK-EPS) but has provisions for adding regulators for use with an unregulated power supply.
- Provides separate jacks for one audio-cassette, TTY and Power Supply.
- Board size: 14.5 in. Long x 11.5 in. Wide x 8 in. High
- Power requirements; 5V.DC @ 0.2 Amps.

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

RNB ENTERPRISES INCORPORATED

2951 W. Fairmount Avenue
Phoenix, AZ 85017
(602) 265-7564

master charge VISA

editors and assemblers are easy to come by for most console systems. On the other hand, not many microprocessor development systems will allow you to run a program in BASIC, or Pascal, or FORTH. In other words, while a personal computer can be turned into a development system, a development system usually does not make a very good personal computer. It should be made clear, however, that a personal computer is not a development system without items 6 and 7 in the list above (or their equivalent). The EPROM programmer is easy. Such accessories are available for very reasonable prices. If you don't mind stuffing a blank board, you can put together your own universal EPROM programmer for less than $30. However, the "emulator" function is not quite so available.

The function of an emulator is to provide the prototype controller with the attributes of an operating system. Suppose you want to make a controller out of an existing single-board computer like a KIM or SYM. After attaching any additional I/O hardware required, you can hand assemble a controller program and enter it into the KIM or SYM's RAM using its built-in operating system. Programs under development can be saved on tape. Software debug functions are even available to get the program running. But what do you do if your prototype controller is not like a KIM or SYM? What if it has no keyboard or display, or any means (operating system) of entering a program into itself? There are two solutions to this problem. One is to use (abuse) an EPROM programmer. The second is to use some kind of emulator.

The first solution mentioned is actually used by owners of commercial development systems, who do not have an emulator module. It works as follows. First, a program is developed and entered into RAM in the development system's microcomputer. The RAM contents are now burned into an EPROM. The EPROM is now plugged into the prototype system and an attempt made to reset the prototype system and run the program. If the program does not run as expected, the program is modified and a second EPROM is programmed. In the meantime, the old EPROM is being erased. While this method can eventually produce a working program it is very tedious and ineffecient. To give you some idea of how really dumb this method is, consider using the same method to write a program in BASIC. That is, suppose you had to enter the program into RAM, burn the RAM contents into an EPROM and then plug the EPROM into a special socket to try out your program. Yet that is essentially what many, if not most, people do to bring up controllers. Clearly there must be a much better way.

The second approach is to give the prototype system a virtual operating system with some kind of emulator. Commercial development systems generally emulate the prototype's processor. Such a processor emulator is a very complex hardware and software system, usually requiring two or three large PC boards which live in the development system's card rack. The emulator physically connects to the prototype via a cable which plugs into the prototype's processor socket. The development system is used to create a program in a block of RAM. The emulator allows the block of RAM to be executed as if it resided in the prototype system. In addition, the program can be stepped, the register contents displayed, breakpoints set, etc. Effectively, the emulator runs considerable software "in the cracks" between prototype program instructions. One consequent limitation of this scheme, however, is that many emulators are unable to execute prototype programs at the full processor speed.

While a processor emulator can be quite useful for debugging software, it is somewhat less suitable for finding hardware bugs. Unfortunately, many users attempt to debug complicated software before even knowing whether the hardware is 100% functional. As mentioned, a processor emulator is very expensive, typically two or three thousand dollars, and cannot be used independently of the development system for which it was designed. Fortunately, another kind of emulator can be built that is usable with almost any computer system having an operating system, including one as simple as a KIM. Instead of emulating the prototype's processor, this emulator emulates the prototype system's ROM or EPROM. It is nothing more than a small block of RAM that can be alternately addressed as part of the host computer system, or via the EPROM socket in the prototype system. Aside from the fact that an EPROM emulator can be an efficient tool for debugging both hardware and software, the best thing about an EPROM emulator is that it can be put together for less than $100.

An EPROM emulator is used as follows. A program is assembled and placed into the emulator RAM block using your computer's operating system. Throwing a switch on the emulator now causes the RAM block to be addressed from a cable and plug inserted into the empty EPROM socket on the prototype system. If a change is required, the switch is flipped back into the host system position and any changes made in the emulator RAM. There is never any need to burn an EPROM until the program is completely debugged. At any point along the way, the RAM contents can be preserved on tape or disk.

In the next column, we will see what it takes to put together an EPROM emulator, and use it to debug both hardware and software. A very workable microprocessor development system can be had with as little as a KIM, an EPROM programmer,

and an EPROM emulator. With such a budget development system, you can bring up a controller based on ANY processor which can use EPROM. Of course, it's even easier if you have an assembler on your system for the processor you are using in the controller.

Perhaps you would like to control your electric train, or build a home environment control system. Or maybe you want to computerize your wife's loom. Maybe you'd like to cash in on your software experience and build custom controllers for local industrial clients. Whatever your bag, you probably already have with your present personal computer, most of what you need to put together a very practical microprocessor development system. Stay tuned for the next exciting installment. ©

A Kim-I Music File In Microsoft Basic

Part 1.

Anthony T. Scarpelli
N. Windham ME

Getting The System Together

If you have a KIM-I, don't have a printer, but do have a memory mapped video display, here's how I solved the problem of getting a software routine to cause an ASCII keyboard to act like a serial teleprinter with all the KIM-I's teletype operations. There's nothing that seems complicated about what I did, but it sure took some mental gyrations to get it working. Yet I did learn a lot about the KIM monitor routines which I'll tell you about. Also how to implement BASIC, and how to implement a Music File which I wrote for my wife. Here's the story.

I had a KIM-I up and running and was learning a lot about assembly language programming, when the opportunity of getting a high resolution video monitor for cheap came along. I bought a SWTP keyboard, and while I was at one of the computer fairs last year I purchased Microtechnology's 8K visible memory and a main frame. The price was good and it was completely compatible with the KIM. It's a dynamic memory system, but is completely invisible to normal computer use, and it has a standard video output. It works beautifully, and is fairly high in resolution with 64,000 bits as dots on the screen. Writing a "1" in a memory location lights up a dot, and, of course, a "0" turns the dot off. Microtechnology's SWIRL software routine shows the system off and provides hours of viewing enjoyment; and when company comes over it's great for showing off your computer.

Microtechnology also has a text display routine whereby, after an ASCII number is put into the accumulator, a subroutine call to the text display puts the ASCII character on the monitor screen. It provides a 53 character by 18 line display, with both upper and lower case letters. Having a software character generator gives you complete control over the configuration of the letters. For instance, I changed all my lower case letters, which I didn't need, into a table of 26 lines, dots, and other shapes for drawing on the screen. Also, the whole screen can be saved on tape. My wife was very pleased as a valentine message formed from a randomly patterned screen. Hypertape loaded the screen in under three minutes.

I also purchased from Microtechnology their bare board 16K memory, and purchased the I.C.'s and components at other sources. You can save about a hundred dollars this way, but you do have to get a few extra memory chips in case a bad one comes up and you do have to do all of your own soldering, and testing. If you go this route you might have a fault in the bare board. In the one I bought, a part of the PCB pattern wasn't etched away so I had no -5v supply. After I fixed the problem the board worked perfectly the first time running and onwards, and I have nothing but praises for the design.

Then came the job of getting my keyboard with parallel output ASCII to go serial. It turned out to be not too difficult when I found an interface in a series of articles by John Blankenship in Kilobaud. In the March '78 issue he shows how to build a parallel to serial interface for the KIM-I. It merely takes the parallel output of the keyboard, using three I.C.'s and a transistor, and the KIm's power and clock, and converts it to a serial output which is presented to the printer input of the KIM. It worked very well.

Then what? Well, here comes the hard part. In order to get the KIM to accept a printer input, you connected pins "21" and "V" on the applications connector, hit the RS button, press the RUBOUT key on your keyboard and type away. The only problem is that any ASCII characters that come in don't go anywhere except to the printer output of the same connector. The ASCII number is put into the accumulator, but how do you call up a subroutine in some other part of memory to display it? The solution wasn't too difficult. You write a little program that jumps to KIM's own GETCH subroutine which then puts the printer ASCII character into the accumulator, then jumps to the character display subroutine, then jumps back to the GETCH etc. You start out by going to the memory location where the program starts on the KIM keyboard, short the two pins together (best to get a switch to do this), hit RS, then RUBOUT, and G on the keyboard, and away you go. You're finally writing on the CRT. Now what?

With this method that's about all you can do because you are in a program of your own creation and are using KIM's ROM routines, and you have to stay there until you hit ST (stop). What I really wanted to do was have my keyboard act just like a printer: change memory, display it, and all the other things the user manual said you can do with a printer. I asked myself, how easily can this be done? More likely, how difficult is it. There were two possibilities open to me: hardware or software. My old teacher said you never learn enough by going the easy route. I didn't know whether hardware or software was the most difficult, but I chose software. You can judge the result; I probably would have bought a printer.

To go the software route meant rewriting some of the subroutines in the KIM's ROM. To show you what routines I had to include, let's go over what happens in the KIM when you hit RS. So get out your user manual, follow the diagrams and let's go.

First look at the listings starting at 1C22 in the User Manual and also at fig. 1.

1. When the RS (reset) button is pressed the data at locations 1FFC & 1FFD, which happens to be the address 1C22, is put into the program counter. This is the entry point for the program in ROM of the 6530-002. This address is fixed and cannot be changed. It is the KIM entry via RST.

2. The first thing that happens is the stack pointer is initialized to FF.

3. Then we go to a subroutine called INITS at 1E88. In INITS, the first thing done is to put 01 into the X register and then put it into the top of the stack at 00FF.

Next, the X-index gets 00 and is stored in PADD which is the 6530-002 A ports data direction register. This is at address 1741 and makes all the ports inputs so they can accept data from TTY or KB (keyboard).

Next X-index gets 3F and is stored at 1743 which is the 6530-002 B ports data direction register, PBDD, and it makes ports PB6 and PB7 inputs, and all the rest outputs. PB7 is connected to the audio tape interface circuits and is prepared to accept program loading from tape.

Next X-index is loaded with 07 and is stored in SBD (1742) which is the data to be sent out from the 6530-002 data ports. So PB 0, 1, & 2 now have 1's on them. PB0 is for TTY data out. PB 1, 2, 3, & 4 go to the 74145 I.C.'s inputs. With a 1 on 1 & 2 and 0 on 3 & 4, all the outputs of the 74145 are high except 03. This goes out to application connections A-V. When this pin is connected to A-21 (PA0), PA0 becomes low. This indicates TTY mode.

Next decimal mode is cleared and the interrupt disable status is set. Then a return from this subroutine.

4. Next back at 1C2A, FF is stored at 17F3 (CNTH 30) which is the TTY count, and 01 is stored in the accumulator. Then SAD (1740) is tested, specifically PA0. If it is not equal to zero, that is, if it's high, the program branches to START. PA7 is tested also. This is the input from the TTY keyboard. It tests for a rubout bit. PA7 is normally a one and the program will keep on testing this input until a zero is detected and also PA0 in case the TTY mode is not wanted any more.

If a zero is detected, the accumulator is loaded with FC and the carry flag is cleared, then 01 is

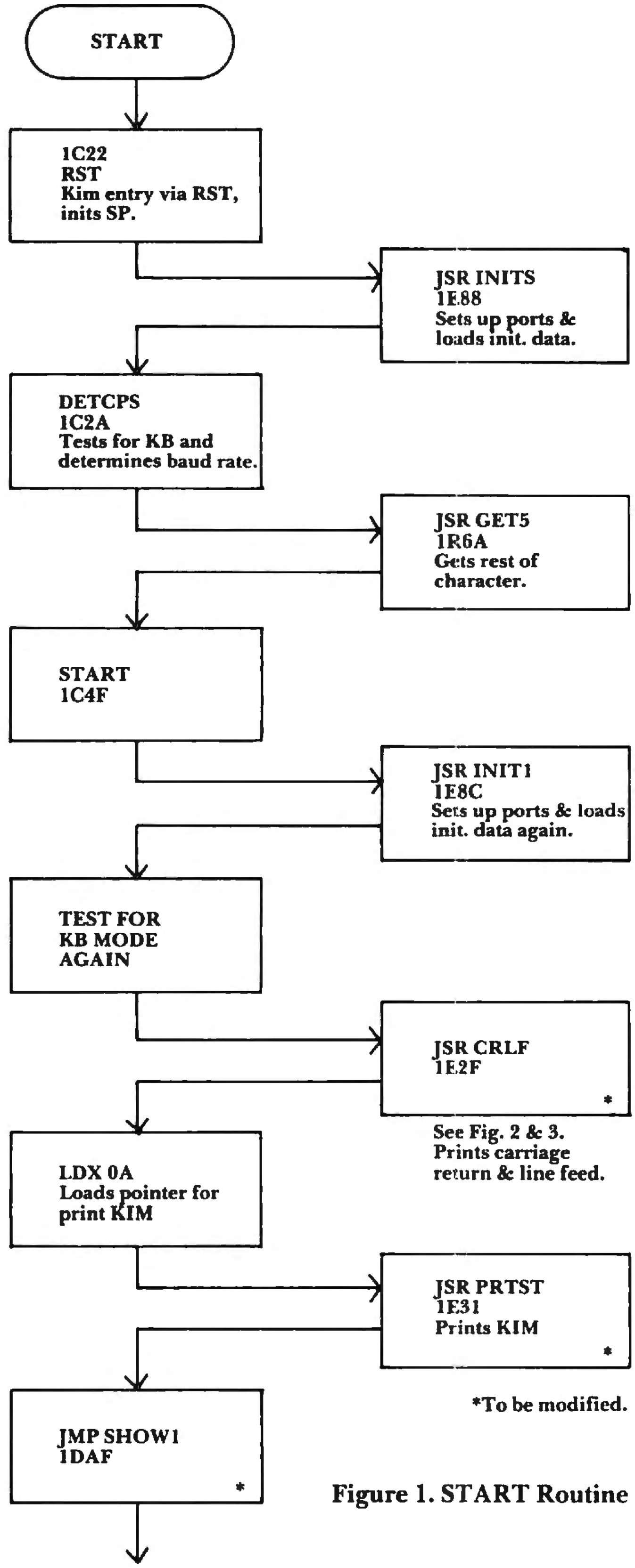


Figure 1. START Routine

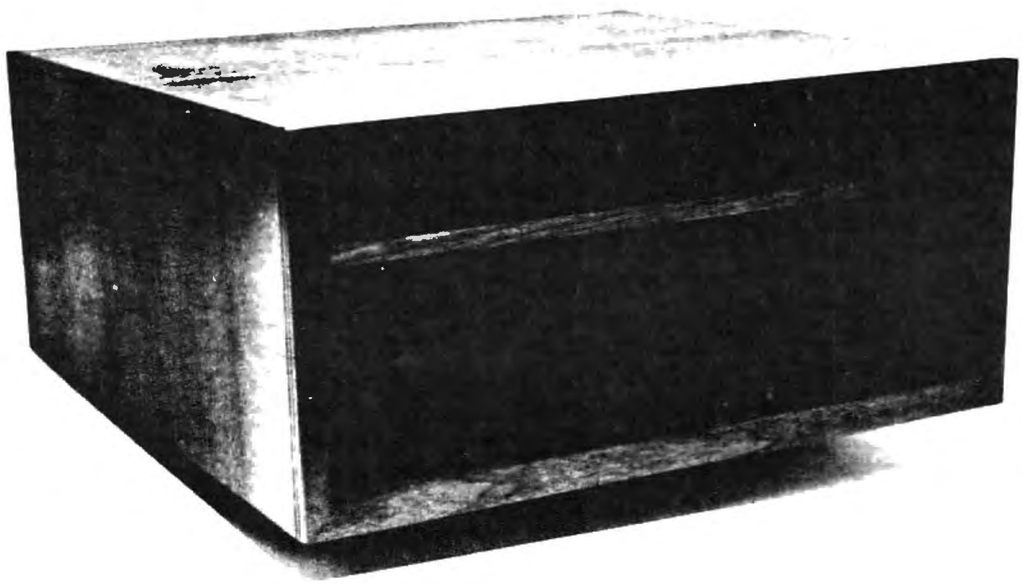
added to the accumulator (FC). If the carry flag is not set it will branch to DET 2. It will the first time around anyway. This part (DET 2) first loads Y-index with SAD (1740) and if the rubout bit is still there (a 0 at PA7) then it goes back to DET 3 and another 01 is added to the accumulator. When the

HDE inc.

BOX 120
ALLAMUCHY, N.J. 07820
201-362-6574

HUDSON DIGITAL ELECTRONICS INC.

ANNOUNCING THE HDE OMNIDISK 65/8



Now, you can "plug in" the latest in a successful series of flexible disk systems developed by HDE for the KIM, SYM and AIM microcomputers. The OMNIDISK 65/8 is a complete system, using 8 inch soft sectored diskettes with a formatted (IBM Standard) capacity of 256K. Of course, a disk formatting function is included as are system supporting utilities for file renaming, disk packing, copy (dual systems) and others.

TED, a full featured, line oriented editor is standard in KIM and SYM based versions to get you up and running on your project in a hurry. The AIM version uses the on-board editor. With the OMNIDISK 65/8 you can concentrate on your problem, the disk supports you all the way.

OMNIDISK 65/8 is available in an attractive walnut wood cabinet, or unpackaged for OEM applications in dual and single drive configurations. The HDE disk controller is a state-of-the-art 4½" by 6½" card electronically compatible with the 44-pin KIM-4 bus structure. The controller and disk-driver are designed to operate with the popular Shugart 801-R and compatible devices.

The OEM single drive is $1195, the dual, $1895 and the dual in the walnut cabinet, $2200. Price is another reason to step up to the proven quality of an HDE system.

HDE PRODUCTS – BUILT TO BE USED WITH CONFIDENCE
AVAILABLE DIRECT OR FROM THESE FINE DEALERS:

JOHNSON COMPUTER	ARESCO	PLAINSMAN MICROSYSTEMS	LONE STAR ELECTRONICS	PERRY PERIPHERALS
Box 523	P O Box 43	Box 1712	Box 488	P.O Box 924
Medina, Ohio 44256	Audubon Pa 19407	Auburn, Ala 36830	Manchaca, Texas 78652	Miller Place, N.Y. 11764
216-725-4560	215-631-9052	800-633-8724	612-282-3570	516-744-6462

accumulator reaches FF and 01 is added, the carry flag is set and CNTH30 (17F3) is incremented, it becomes 00. As long as the rubout bit is there the accumulator keeps on increasing and increases CNTH30. As soon as the bit ends the accumulator is stored in CNTL30 (17F2) and X-index gets an 08. Then the program goes to subroutine GET5 at 1E6A, where it goes to DEHALF (1EEB).

5. DEHALF first gets the high byte count time at CNTH30 and stores it in TIMH (17F4), then gets CNTL30. The accumulator and TIMH are shifted right (divides by two). If the 0 bit had a 0 the carry flag is cleared and a branch is taken to DE2, otherwise the accumulator is OR'd with 80 and it branches to DE4. If the DE2 branch was taken the carry flag has been set and next 01 is subtracted from the accumulator. The time is reduced and back with RTS. What is happening here is the keyboard baud rate in CNTL30 and CNTH30 is halved to get in the middle of the bit, then delayed one whole bit to read the next bit of the character. Cute, huh.

6. Back at 1E6D (GET2), the accumulator is loaded with SAD and the bit number 7 only is saved. 00FE is shifted right, then OR'd with the accumulator and stored in 00FE. Another delay and the process is repeated until the whole character is retreived, then another half delay, X-index is loaded with TMPX (00FD), and the accumulator gets CHAR which is the ASCII character. The accumulator is rotated left then shifted right, which gets rid of any parity bit that might be stuck on the character. Then a return to START.

7. START. First is a jump to subroutine INIT1 (1E8C) which is the same as before, it sets up the ports. The accumulator is loaded with 01, and SAD is tested again for TTY or KB mode. If there's a 1 in PA0 it branches to KB mode. If no KB mode, it then jumps to CRLF, Fig. 2 & 3, (1E2F), which

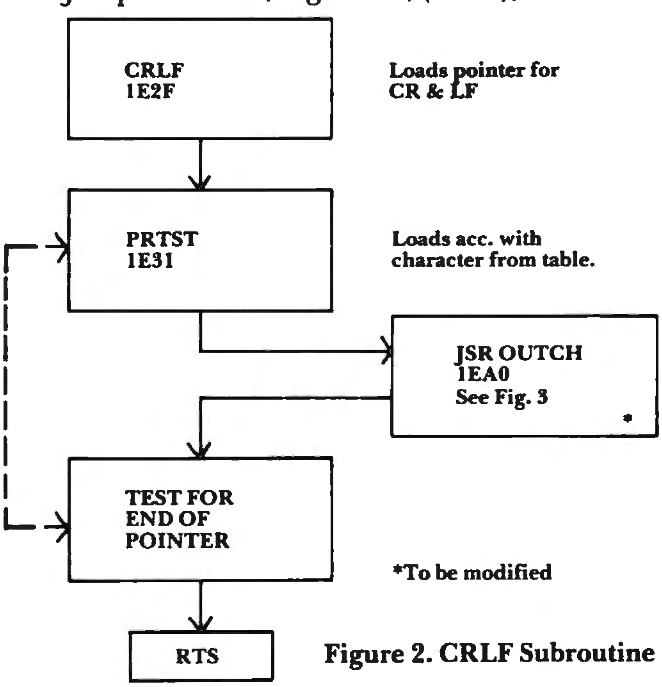


Figure 2. CRLF Subroutine

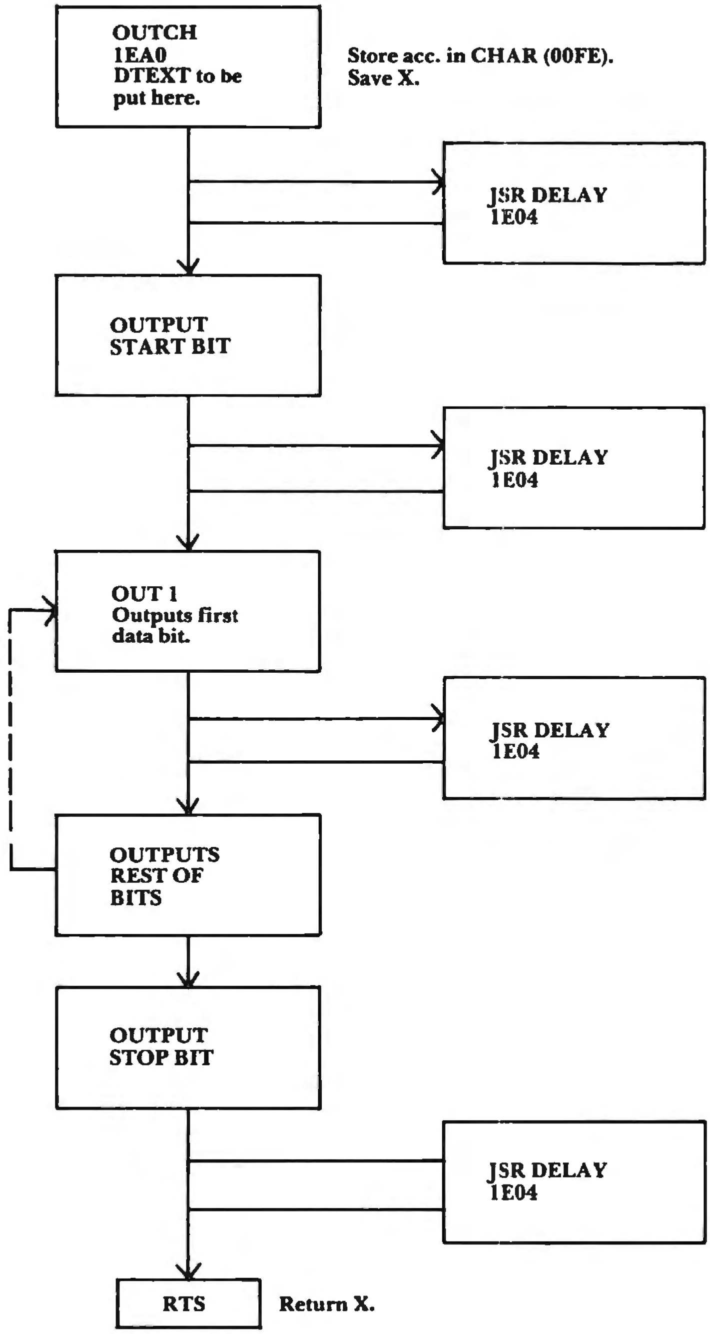


Figure 3. OUTCH Subroutine

prints a carriage return, then a line feed, then JSR PRTST prints "KIM", then jumps to SHOW1 (1DAF), Fig. 4, and then back to CLEAR, Fig. 7.

8. CLEAR. The accumulator gets loaded with 00 and is stored in INL & INH. The program tests for a character in GETCH, Fig. 8. In GETCH it stays in a loop waiting for a start bit. After the start bit, the rest of the character is retreived and loaded into the accumulator, the program then comes back, and we test for KB mode again. If no KB the character is changed into a hex number in PACK, Fig. 9, and then in SCAN, Fig. 10, the program determines if the hex number is an execute key. If not, it will get another character.

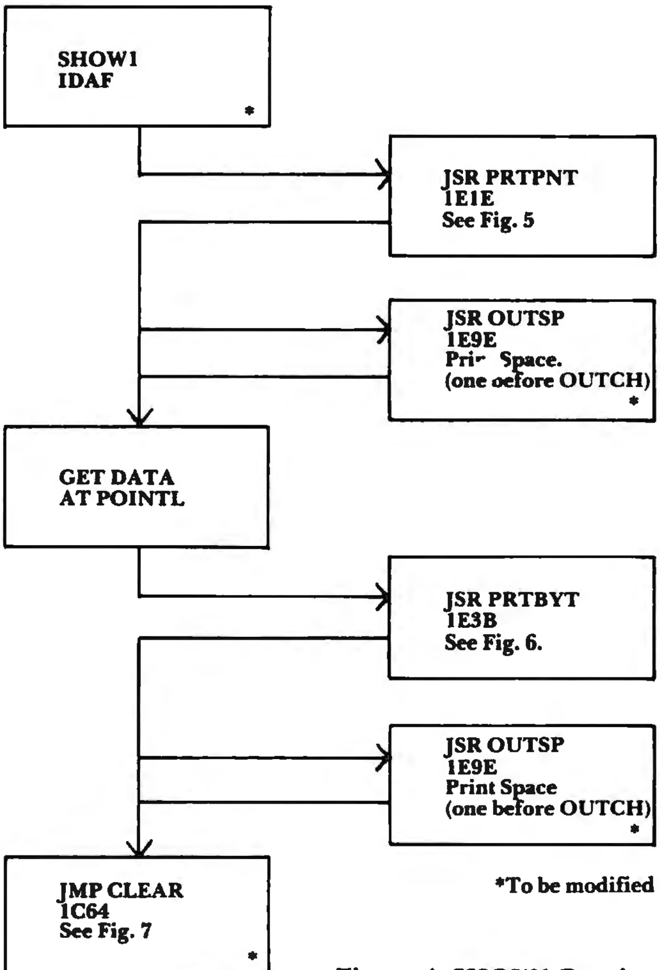


Figure 4. SHOW1 Routine

So this is the program I need to simulate a teletype. The problem now becomes, what are the subroutines I have to rewrite and which ones of the KIM's ROM subroutines can I use. Obviously, any part of the program that refers to a ROM address has to be rewritten, such as in a JMP. Also when the accumulator gets the ASCII character that is to be displayed, the program that does the displaying, in this case called DTEXT (the Microtechnology software routine), has to be addressed at the right point, and thus any subroutines involved here have to be rewritten. So definitely the subroutine OUTCH has to be changed to add DTEXT. We get to OUTCH from CRLF so that has to be rewritten. CRLF is addressed from START which is part of the whole RST routine. As you can see it starts to get involved. So if you go this route table I lists all the KIM ROM routines that must be rewritten. Of course in this rewriting, some branches have to be changed as well as addresses. (A SASE sent to me will get you a list of the changed addresses.)

Now my keyboard acts just as a teletype, and I can display all the teletype outputs from the KIM on the CRT. First I go to the RST program address, the one I rewrote, on the KIM display, switch to teletype mode, hit RS on the KIM, then press the rubout key on the keyboard. The SWTP keyboard doesn't have an actual rubout key, but there are two spare keys, one of which can be wired as rubout. Then I press the G key which puts me into the RST program, (rewritten). When the rubout key is pressed again the CRT will display "KIM" and also the address of the RST program; now we are as a teletype with all its functions. Simple, wasn't it.

Next time I'll go into the actual file program that creates a music file, and then can search it for any of a number of subjects.

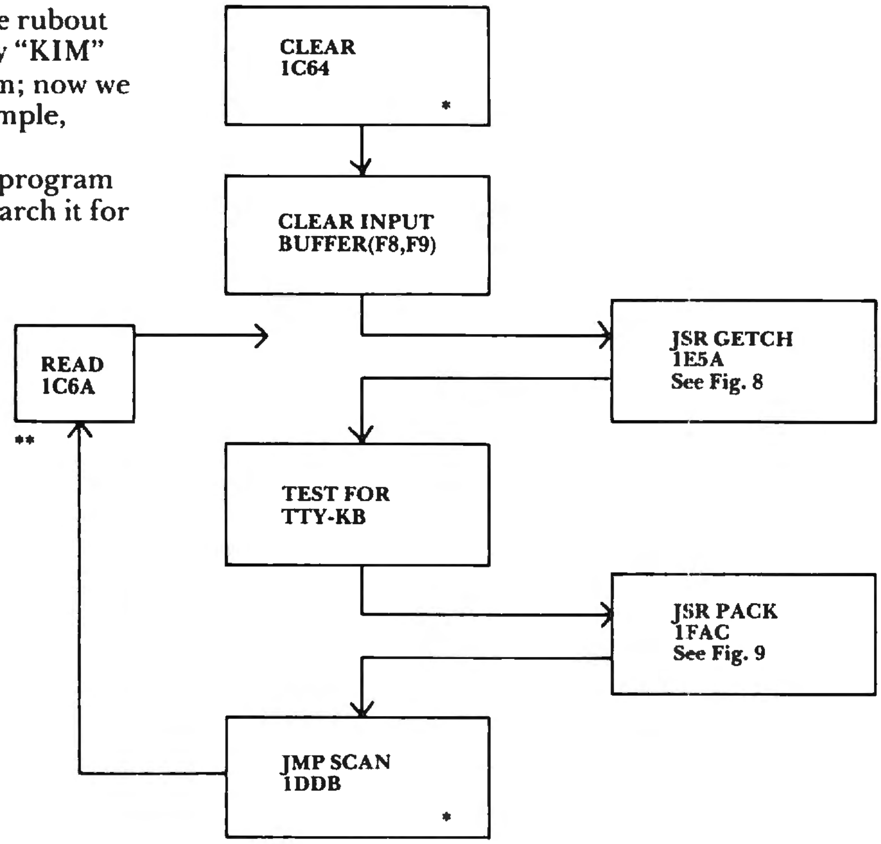


Figure 7. CLEAR Routine

*To be modified
**Re-entrance from SCAN

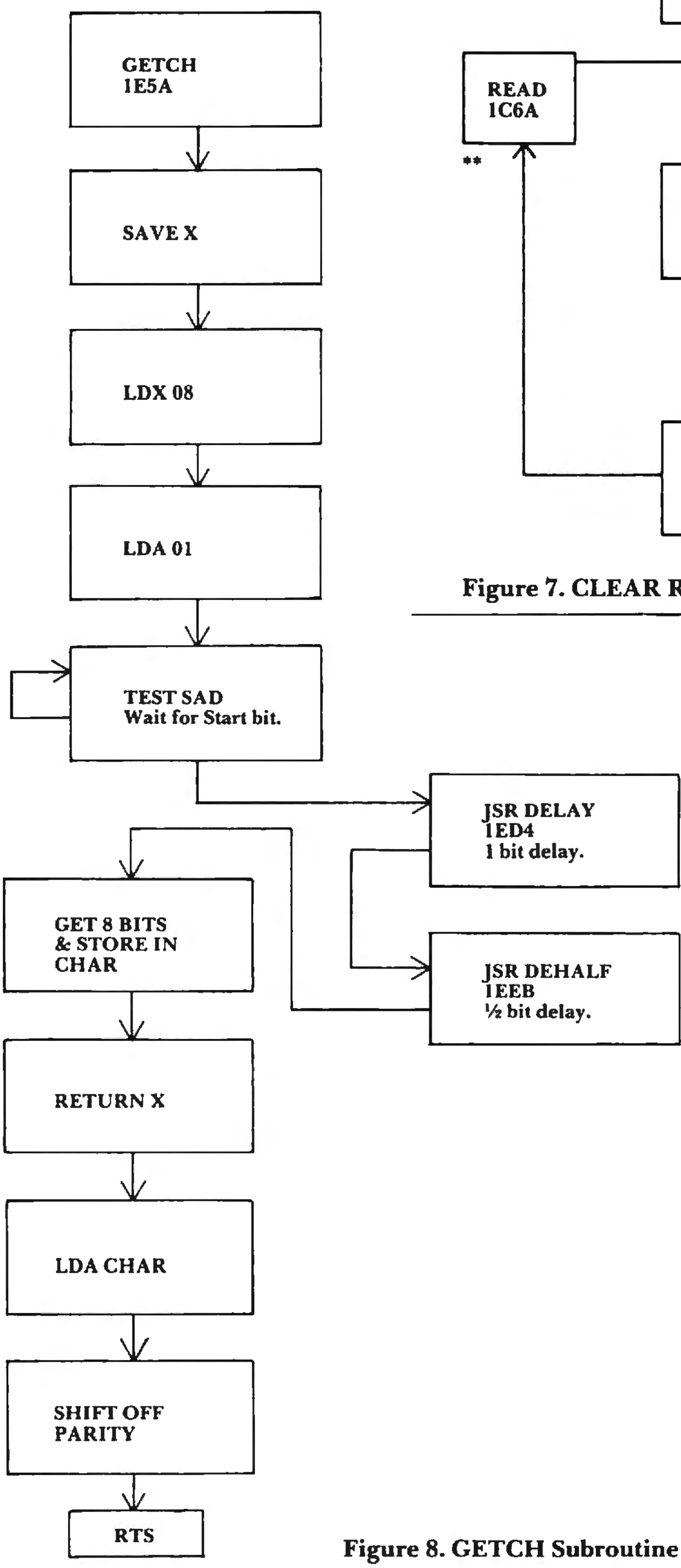


Figure 8. GETCH Subroutine

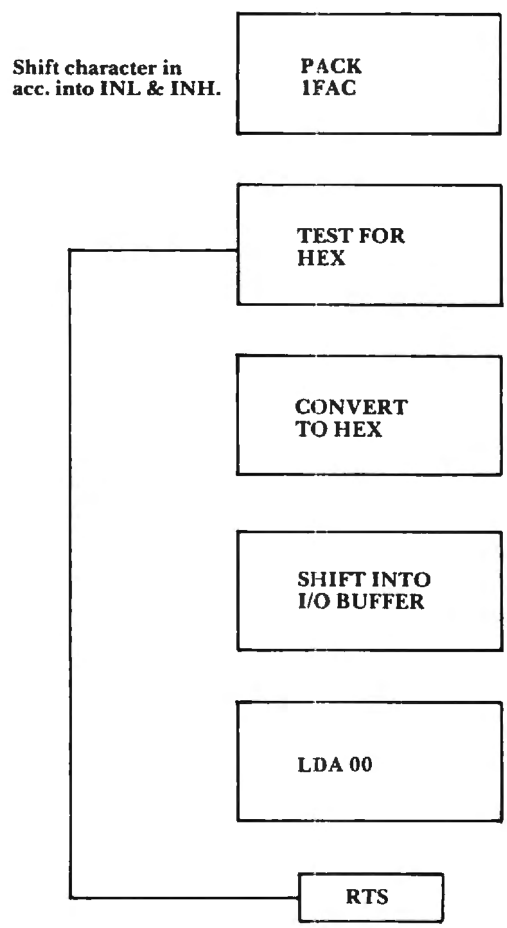


Figure 9. PACK Subroutine

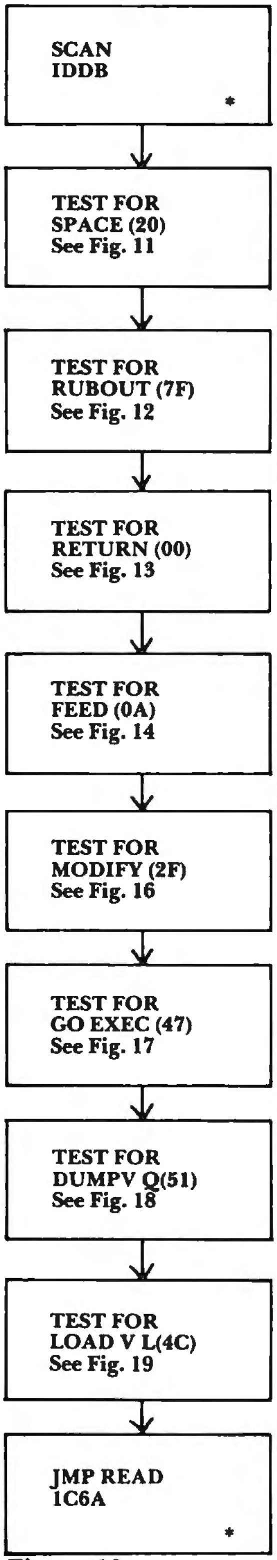


Figure 10.
SCAN Routine

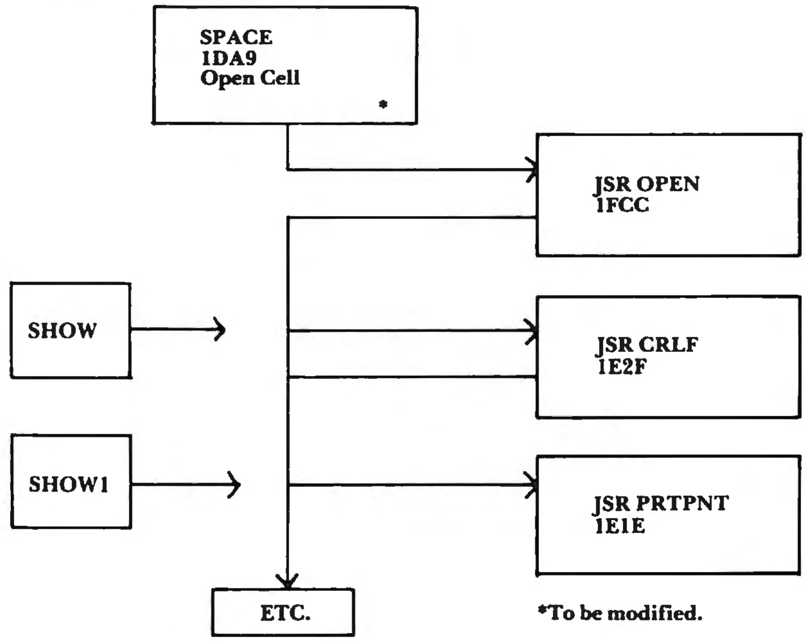


Figure 11. SPACE Routine

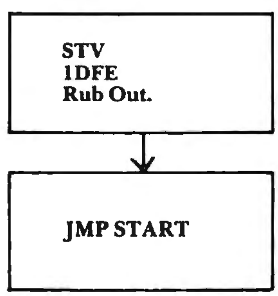


Figure 12. STV Routine

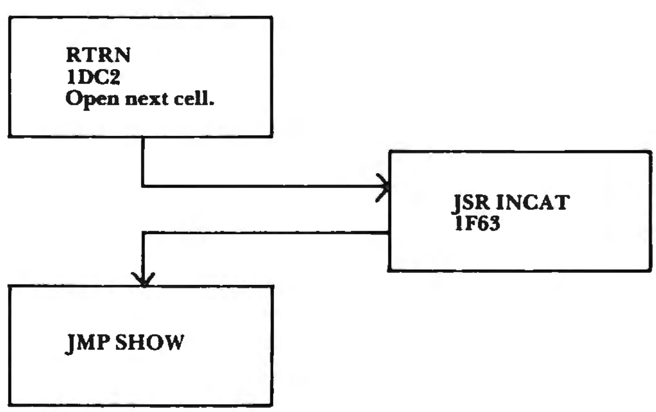


Figure 13. RTRN Routine

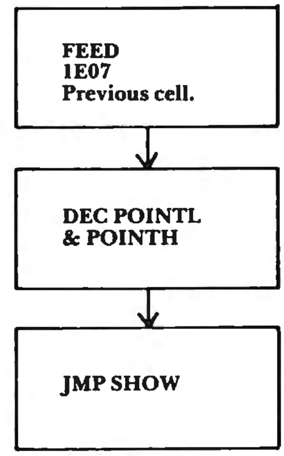


Figure 14. FEED Routine

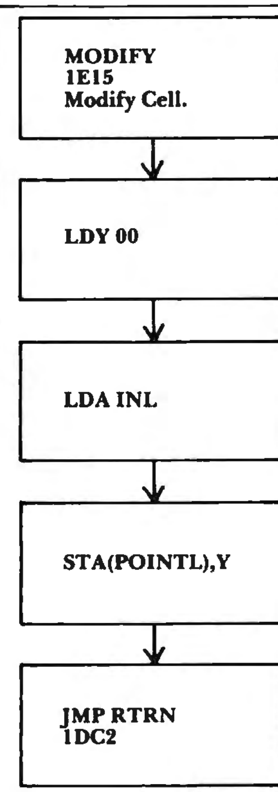


Figure 16. MODIFY Routine

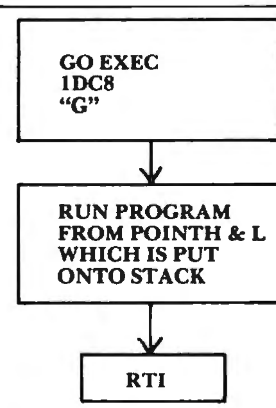
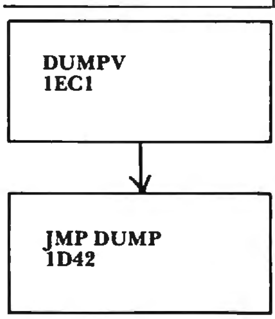


Figure 17. GOEXEC Routine



Dump from open cell to hi limit. Prints out memory.

Figure 18. Dump "V" Routine

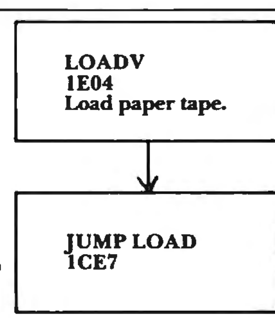


Figure 19. Load "V" Routine

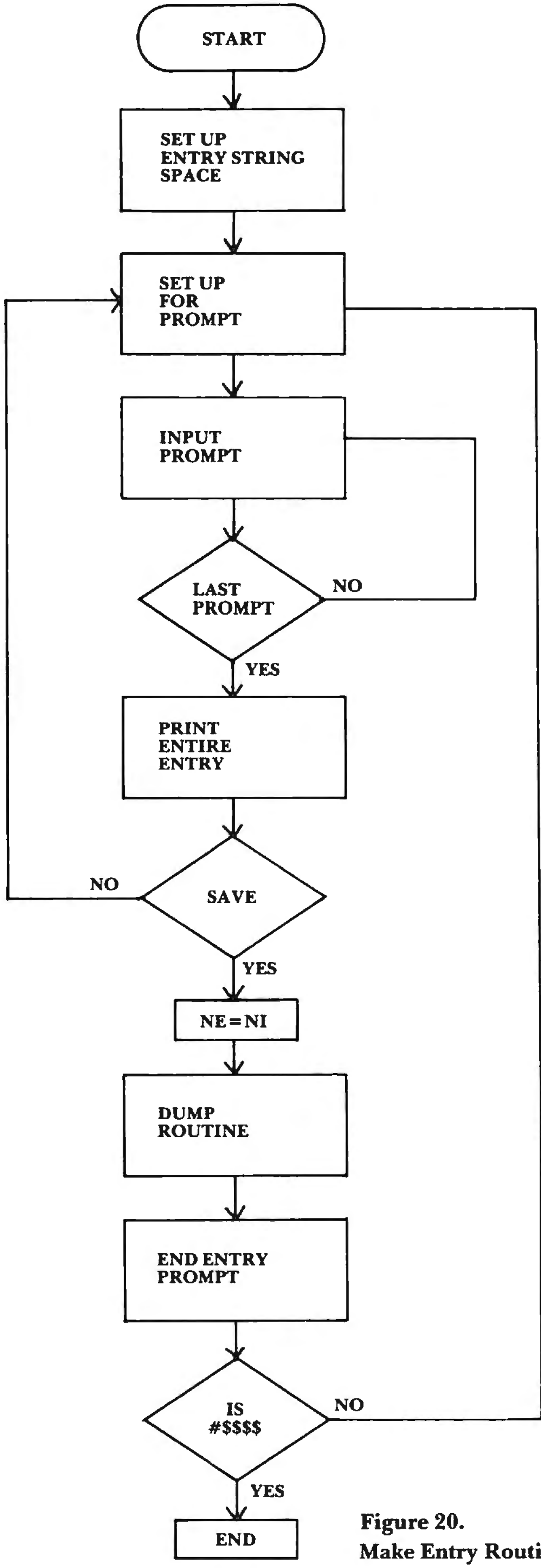


Figure 20.
Make Entry Routine

Assembly Language Program for Cassette DUMP & LOAD

```
0300 A5 01      DELAY  LDA DELTIM  Load delay
  02 85 EE             STA TIMEA     value.
  04 A9 30      DECB   LDA 30      Load
  06 8D 04 17          STA 1704      timer.
  09 2C 07 17   TEST   BIT 1707    Test timer.
  0C 10 FB             BPL TEST    Branch if not run out.
  0E C6 ED             DEC TIMEB   Reduce time value.
  10 D0 F2             BNE DELB    Start again.
  12 C6 EE             DEC TIMEA   Reduce delay value.
  14 D0 EE             BNE DELA    Branch if not done.
  16 60                RTS         Return.
0317 A9 02      TWRITE LDA #02     Turn tape on.
  19 10 02             BPL TAPE
  1B A9 01      TREAD  LDA #01     Turn tape off.
  1D 4D 03 17   TAPE   EOR 1703
  20 8D 03 17          STA 1703
  23 60                RTS         Return.
0324 20 17 03   WRITE  JSR TWRITE  Turn tape on.
  27 20 00 03          JSR DELAY   Delay for tape speed.
  2A 20 00 02          JSR HYPER   Record in hypertape.
  2D 20 17 03          JSR TWRITE  Turn tape off.
  30 20 8C 1E          JSR INITI   Open ports again.
  33 60                RTS         Return.
03D5 20 1B 03   READ   JSR TREAD   Turn on tape.
  D8 20 36 03          JSR LOADT   Load tape.
  DB 20 1B 03          JSR TREAD   Turn off tape.
  DE 20 8C 1E          JSR INITI   Open ports again.
  E1 60                RTS         Return.
```

Note: HYPER is taken from The First Book of KIM page 119 relocated to address 0200.

LOADT is taken from the KIM-1 User Manual Program listing page 6 relocated from address 1871-1931 to 0334-03D4.

If you wish to use the same routines in the same addresses as I did, send a SASE and I'll let you know what locations have to be changed in those listings to get it to run right. ©

CAPUTE!

Corrections/Clarifications

From Raymond Diedrichs ("Pet File I/O In Machine Language", April, 1981, Issue 11, pp. 144-145):

> "In the machine language open statement, the following lines are missing:
> **LDA #DEVICE-NUMBER**
> **STA $D4**
> **LDA #SECONDARY-ADDRESS**
> **STA $D3.**

They should appear directly below the line which reads:
STA $D2.

And here's the missing listing from Charles Brannon's "String Arrays in Atari BASIC," April, 1981, Issue 11, p. 103.

```
100 REM SIMPLE BAR GRAPH PROGRAM
110 GRAPHICS 0
120 PRINT "NUMBER OF COMPANIES";
130 INPUT NC
140 MAXLEN=20:DIM A$(MAXLEN*NC),L(NC),
    ¬A(NC),T$(MAXLEN)
150 FOR I=1 TO NC
160 T$="                    ":REM 20 ¬
    ¬SPACES
170 E=I:GOSUB 20000
180 PRINT "ENTER THE NAME OF COMPANY ";I
190 INPUT T$:GOSUB 20000
200 PRINT "AMOUNT FOR ";T$;
210 INPUT A:A(I)=A:IF A>HI THEN HI=A
220 PRINT:NEXT I
230 GRAPHICS 0
240 FOR I=1 TO NC
250 E=I:GOSUB 30000
260 PRINT:PRINT T$
270 FOR J=1 TO (A(I)/HI)*30
280 PRINT CHR$(160);
290 NEXT J
300 NEXT I
310 END
20000 L=LEN(T$):IF L>MAXLEN THEN ¬
    ¬L=MAXLEN
20010 L(E)=L:START=(E-1)*MAXLEN+1
20020 A$(START,START+L-1)=T$:RETURN
30000 START=(E-1)*MAXLEN+1
30010 T$=A$(START,START+L(E)-1):RETURN
READY.
```

Program Listings for COMPUTE

Cursor control characters will appear in source listings as shown below:

```
h=HOME          , ĥ=CLEAR SCREEN
↓=DOWN CURSOR   , ↑=UP CURSOR
→=RIGHT CURSOR, ←=LEFT CURSOR
r=REVERSE       , r̂=REVERSE OFF
```

Graphics (i.e. shifted) characters will appear as the unshifted alphanumeric character with an underline. This does not apply to the cursor control characters. The Spinwriter thimble doesn't have a backarrow symbol, so a "~" is used instead.

The "¬" is used to indicate the beginning of a continuation line. It is also used to indicate the end of a line which ends with a space. This prevents any spaces from being hidden.

New Products

THE CARVERY

The Carvery
156 Front Street West
Toronto, Ontario M5J 2L6
Telephone 977-7770

New Toronto Restaurant

Why does a computer pour the drinks at the Carvery, a downtown Toronto restaurant? Jim Butterfield, who is a small shareholder in the establishment, has no comment. Neither does he explain why the machine always pours him doubles.

Atari Launches Major Software Acquisition Program

Sunnyvale, California — April 3, 1981 — A major new effort to expand the library of consumer-oriented software for its personal computer systems is being launched by Atari, Inc. Atari is looking for high quality programs that can be used immediately, and easily, by people with little or no training in the use of computers.

"We want to acquire software in the areas of personal finance, self-improvement, education and home entertainment. We are encouraging the creation and marketing of software by vendors and developers, and want to help market appropriate materials from outside authors," Bruce W. Irvine, vice president of software for Atari's Computer Division said. "To start things off, we are sponsoring a $100,000 contest for software authors."

The acquisition program involves the creation of Atari Software Acquisition Program regional centers where qualified developers can work with Atari equipment and receive technical assistance, and Atari Program Exchange, a free quarterly catalog of user-written software to be distributed to Atari computer owners. In addition, Atari will offer periodic technical seminars for qualified software authors to familiarize them with the inner workings of Atari computer products and enable them to write programs that take advantage of all the advanced features of the ATARI 400™ and 800™ computers.

"We recognize that a broad selection of readily available software is a critical key to the ultimate consumer market. No one company can create the amount of material needed to properly address the market, so we are going to do our best to encourage our users and software vendors to create programs compatible with Atari computers. Often, a user or developer is an expert in a field we don't know much about; with our assistance, that person can make his or her programs available to the wide audience they deserve," Irvine added.

Acquisition Centers

Beginning with an initial installation in the Sunnyvale area which will open in mid-May, Atari will develop software acquisition centers in geographical areas where there are high concentrations of programmers and users, such as metropolitan areas with technical universities. No timetable has been announced for the opening of these additional facilities.

Qualified developers will be able to use the centers on an appointment only basis. Design of the centers will help insure the privacy of material under development. Each center will be equipped with Atari computers annd peripherals, all necessary reference materials and technical manuals. Center staff will help answer technical questions and review and evaluate completed software.

Once a program is completed, Atari may be interested in marketing it under the company name, or accept it for listing in the Atari Program Exchange catalog. Or, developers may wish to market the program on their own.

Pet User Group Celebrates Third Birthday

As shown in the birthday cake picture above, SPHINX celebrated their third year with a full sized Pet cake (complete with keyboard and message on the screen) at their meeting March 14, 1981 at the Lawrence Hall of

Science, Berkeley, California.

Originally formed by Niel Busey and Milt Lee, SPHINX, (Society For Pet Handlers Information Exchange), cooperated with Lawrence Hall of Science in putting out a newsletter which contained basic information about the Pet when there was little from the manufacturer.

Although the newsletter has been discontinued, they are still active in exchanging programs. At the sixth West Coast Computer Fair, April 5, 1981, a proposal was made that librarians from user groups across the United States trade programs on a disk basis. To this end SPHINX would like to receive 2040 or 4040 format disks from other groups and will return the diskette(s) with programs from our library (currently 13 diskettes and growing.)

Other current SPHINX projects are a nationwide Pet/CBM telephone network for Pet users with modems. Some interest in sponsoring this has been shown by Commodore. SPHINX also plans to start a library for the VICcolor computer because of the tape and software compatability. Many of their programs will run on a VIC with minor or no modifications.

For further information, please write to SPHINX C/O their sponsor:

PC Computers
10166 San Pablo Ave.
El Cerrito, CA 94530

Meetings in the Bay Area are the only way SPHINX currently exchanges individual programs — the second Thursday of the month at Lawrence Hall of Science, Chem. Lab, Berkeley, CA at 7:00 p.m.

New Low-cost 80-Column Dot Matrix Printer

MICROTEK, Inc. has announced a new low-cost (under $300) 80-column dot matrix printer. Dubbed the "BYTEWRITER-1", the printer accepts single sheet or roll paper up to 8½ inches wide and prints at 60 lines per minute using a 7 x 7 dot matrix.

The BYTEWRITER-1 interface is similar but not identical to a Centronics parallel interface, and has been designed specifically to operate with the Apple II, the Atari 400/800, and all models of the TRS-80. Using a print mechanism and logic board designed and manufactured in the U.S., the unit is priced at $299 (interface cable slightly extra). MICROTEK is directing its marketing efforts towards the personal computing and hobbyist segments of the market, and will sell the printer direct only. The BYTEWRITER-1 carries a 90-day limited warranty. Delivery is from stock to 60 days.

For further information, contact Diane Barney-Laukat at MICROTEK, INC., 9514 Chesapeake Drive., San Diego, California 92123. (714)-278-0633.

High Performance Data Communications System

Norcross, Georgia — Hayes Microcomputer Products, Inc., announces the Hayes Stack Smartmodem high performance data communications system for small computers.

The Smartmodem, an FCC-approved direct-connect device, is designed for use with RS-232C compatible computers or terminals to communicate via the telephone system with other computers or time sharing systems.

A unique feature is that the Smartmodem can be program controlled in any language by ASCII character strings.

This intelligent datacomm system analyzes and executes commands and in response sends result codes which, at the user's discretion, can be English words or decimal digits. The Smartmodem has auto dial and auto answer capabilities. A special design feature is that all circuitry required for auto dial and auto answer is installed within the Smartmodem. This eliminates the need for any auxiliary equipment and makes the Smartmodem a stand-alone system.

The Hayes Stack Smartmodem can be connected to any telephone system in the U.S. since dialing can be either Touch-Tone* or pulse. Furthermore, both dialing modes can be combined within a command with pulse being used, for example, to access a PBX board and Touch-Tone for dialing an outside number after the second dial tone is received.

An audio monitor permits the user to follow the progress of the call and be alerted to wrong numbers and busy signals. If a busy signal is encountered, by entering a repeat command, the Smartmodem will automatically redial the number at any time.

Operation can be in full or half duplex with a data rate of 0 – 300 baud. Power-on default options are controlled by the

positioning of seven option switches. Four of these options can be overridden by software command. LED status indicators on the front panel of the unit provide a visual check of the Smartmodem's operational status.

In addition, the unique "Set" commands allow the user to select (and change) various operational parameters such as dialing speed, escape code character and number of rings to answer on.

In announcing the release of the Smartmodem, Glenn Sirkis, Hayes Vice President, stated, "The Smartmodem, offers all the classic modem functions *plus* some special features — e.g., pulse and Touch-Tone dialing — that are available only with a limited number of modems. Add to this the features that are unique to the Smartmodem — e.g., programmable in any language and Set commands for customized operation — and you'll know why we believe the Smartmodem is everything you could ever want in a 300 baud modem."

The Smartmodem has a Two Year Limited Warranty. The suggested retail price for the Hayes Stack Smartmodem system is $279.00. Included in this price are the Smartmodem unit, a power pack, one modular telephone cable to connect the unit to the telephone line and an owner's manual.

The Smartmodem is the first product in a new series that features the exclusive Hayes Stack design. This compact design permits other Hayes components to be stacked on top of the Smartmodem, thereby eliminating clutter.

TM Trademark of Hayes Microcomputer Products, Inc.
• Trademark of American Telephone and Telegraph.

New Professional Applications Package For The Medical Profession

Charles Mann & Associates, Micro Software Division, has announced the release of a new professional applications package for the Medical Profession called "Medirec". The Medirec system is a total Medical History and Report Preparation System. The professional using the system can prepare office input forms, enter patient and family histories, record patient visit symptoms, diagnosis, and treatments, prepare referral requests, prepare patient history summaries, and prepare referral reports. The program compliments the firm's existing Medical Billing Package.

Medirec is designed with today's professional practice liability in mind. The system allows the diskette recording of up to 550 professional visits per diskette. Individual patient records can be recalled, linked together and printed either in whole or in part. The system allows the practitioner to search past history files for common symptoms, diagnosis or the administration of conflicting drug treatments.

The system can recall records for past due follow treatment, prepare reminder notices, prepare liability release forms and print file folder labels. The system comes with a full featured address data base system and a programmable form letter writing element. The system can be programmed to prepare referral report letters, and requests for specialist treatment.

The Medirec system requires a 48K Apple II, Apple II +, or Apple III, an 80 column printer, and two disk drives. A special Corvus Systems hard disk version is also available for system configurations up to 40MB of on line storage.

The system is available from over 700 CMA dealers worldwide for an introductory price of $199.95 (Corvus version $249.95). Preview Documentation is available for $25.00. Additional information and dealer location information can be obtained from Charles Mann & Associates, Micro Software Division, 7594 San Remo Trail, Yucca Valley, CA 92284. Phone (714) 365-9718.

NYSAEDS Conference

On October 18, 19 and 20, 1981, The New York State Association for Educational Data Systems (NYSAEDS) will hold its annual conference in Syracuse, NY. NYSAEDS, an affiliate of AEDS, is composed of people who have a common interest in computers and education.

The theme of this year's conference is "Software". The keynote speaker is Marge Kosel from MECC and the banquet speaker is Dr. Earl Joseph (Futurist) from Sperry Rand. A variety of workshops will be held concerning the uses of microcomputer software in education.

For further information, please contact Don Ross, Ardsley High School, Ardsley, NY 10502.

New Power Line Filter

Pilgrim Electric Company, Plainview, New York, introduces new, compact "Plug-In" style VOLTECTOR®. It provides the most cost-effective protection against voltage spikes, surges, transients and high frequency interference for Word-Processors, Microcomputers and other Microprocessor-based equipment.

A REMARKABLE MAGAZINE

David Ahl, Founder and
Publisher of Creative Computing

creative computing

"The beat covered by Creative Computing is one of the most important, explosive and fast-changing."—Alvin Toffler

You might think the term "creative computing" is a contradiction. How can something as precise and logical as electronic computing possibly be creative? We think it can be. Consider the way computers are being used to create special effects in movies—image generation, coloring and computer-driven cameras and props. Or an electronic "sketchpad" for your home computer that adds animation, coloring and shading at your direction. How about a computer simulation of an invasion of killer bees with you trying to find a way of keeping them under control?

Beyond Our Dreams

Computers are not creative per se. But the way in which they are used can be highly creative and imaginative. Five years ago when *Creative Computing* magazine first billed itself as "The number 1 magazine of computer applications and software," we had no idea how far that idea would take us. Today, these applications are becoming so broad, so all-encompassing that the computer field will soon include virtually everything!

In light of this generality, we take "application" to mean whatever can be done with computers, *ought* to be done with computers or *might* be done with computers. That is the meat of *Creative Computing*.

Alvin Toffler, author of *Future Shock* and *The Third Wave* says, "I read *Creative Computing* not only for information about how to make the most of my own equipment but to keep an eye on how the whole field is emerging.

Creative Computing, the company as well as the magazine, is uniquely lighthearted but also seriously interested in all aspects of computing. Ours is the magazine of software, graphics, games and simulations for beginners and relaxing professionals. We try to present the new and important ideas of the field in a way that a 14-year old or a Cobol programmer can understand them. Things like text editing, social simulations, control of household devices, animation and graphics, and communications networks.

Understandable Yet Challenging

As the premier magazine for beginners, it is our solemn responsibility to make what we publish comprehensible to the newcomer. That does not mean easy; our readers like to be challenged. It means providing the reader who has no preparation with every possible means to seize the subject matter and make it his own.

However, we don't want the experts in our audience to be bored. So we try to publish articles of interest to beginners and experts at the same time. Ideally, we would like every piece to have instructional or informative content—and some depth—even when communicated humorously or playfully. Thus, our favorite kind of piece is acessible to the beginner, theoretically non-trivial, interesting on more than one level, and perhaps even humorous.

David Gerrold of *Star Trek* fame says, "*Creative Computing* with its unpretentious, down-to-earth lucidity encourages the computer user to have fun. *Creative Computing* makes it possible for me to learn basic programming skills and use the computer better than any other source.

Hard-hitting Evaluations

At *Creative Computing* we obtain new computer systems, peripherals, and software as soon as they are announced. We put them through their paces in our Software Development Center and also in the environment for which they are intended—home, business, laboratory, or school.

Our evaluations are unbiased and accurate. We compared word processing printers and found two losers among highly promoted makes. Conversely, we found one computer had far more than its advertised capability. Of 16 educational packages, only seven offered solid learning value.

When we say unbiased reviews we mean it. More than once, our honesty has cost us an advertiser—temporarily. But we feel that our first obligation is to our readers and that editorial excellence and integrity are our highest goals.

Karl Zinn at the University of Michigan feels we are meeting these goals when he writes. "*Creative Computing* consistently provides value in articles, product reviews and systems comparisons . . . in a magazine that is fun to read."

Order Today

To order your subscription to *Creative Computing*, send $20 for one year (12 issues), $37 for two years (24 issues) or $53 for three years (36 issues). If you prefer, call our toll-free number, **800-631-8112** (in NJ 201-540-0445) to put your subscription on your MasterCard, Visa or American Express card. Canadian and other foreign surface subscriptions are $29 per year, and must be prepaid. We guarantee that you will be completely satisfied or we will refund the entire amount of your subscription.

Join over 80,000 subscribers like Ann Lewin, Director of the Capital Children's Museum who says, "I am very much impressed with *Creative Computing*. It is helping to demystify the computer. Its articles are helpful, humorous and humane. The world needs *Creative Computing*."

creative computing

Attn: Barbie
P.O. Box 789-M
Morristown, NJ 07960
Toll-free **800-631-8112**
(In NJ 201-540-0445)

The unique heavy duty design meets the latest industry Surge Voltage Standards and provides the highest level of commercially available two-way protection. It assures electronic equipment of clean incoming AC power ... and it prevents disturbances generated by equipment from being conducted back into the power line. The VOLTECTOR® Series 6 Plug-In Power Conditioner supplies the positive surge-protection and filtering that most manufacturers have left out of their equipment.

Available with current ratings of 1, 2, 5, 7 and 10 amps.

Prices: Begin at $79.50
Delivery: Stock

Manhattan Software Announces Four Atari Game Programs

Manhattan Software, long a publisher of programs for the TRS-80, has begun issuing a series of programs for the Atari Computer. The first four releases are:

Gin Rummy 3.0, with color card graphics and sound, which plays a full regulation game of Gin, and can hold its own against even skilled Gin players. Prices at $19.95, the program requires 32K memory and one joystick.

Casino Blackjack/Counter, a realistic simulation of playing at a casino table — card graphics show five hands dealt, and the user plays the center hand while the computer plays the rest. A major purpose of the program is to teach card-counting, a method which is claimed to give the player a statistical advantage over the house in some situations. Priced at $19.95, for 24K and one joystick.

Labyrinth Run, a test of skill and coordination, using the joystick to guide a fast-moving runner through twists, turns, reverses and slaloms, with thunderous crashes when the runner hits a wall. Three skill levels. $14.95, the game requires two joysticks.

These programs are available at dealers, and direct from Manhattan Software, P.O. Box 35, Pacific Palisades, CA 90272. Telephone (213) 454-8290.

Atari Adds Missile Command To Its Video Computer System Game Library

Missile Command™, a popular coin operated video game currently in arcades, is now available in a home video game version, it was announced today by Atari, Inc., creator and manufacturer of both products.

Largely due to Missile Command's success as an arcade game and in response to considerable consumer demand, Atari designed the game cartridge for its Video Computer System™ programmable TV game.

The Missile Command game cartridge is a one or two player game that uses joysticks and offers 34 game variations.

According to Michael J. Moone, president of the Consumer Electronics Division, "Missile Command is one of the most challenging skill and action video games ever created. We believe its popularity will be as pervasive as that of its predecessors, Space Invaders and Asteroids."

The game begins with wave after wave of enemy missiles raining down on an earth missile base and 6 surrounding cities. The player, as base commander, is responsible for protecting and defending the territory from enemy attack. To combat each wave of enemy missiles, the base commander is given 30 guided defense missiles which when exploded in the path of attacking missiles destroys them. Each successive wave of attacking missiles comes faster than the previous one and the game continues until all cities and the missile base are lost.

Additional features include game difficulty adjustment to correspond to player skill levels, slow game variations designed for young children and screen color changes as game progresses to reduce eyestrain during extended game play.

Suggested retail price is $31.95 and cartridges will be available nationwide by April.

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.

Advertiser's Index

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:

5 6 7 8 9 10 11 12

The 6502 Resource Magazine
COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**__________
☐ **Don't yet have one...**
During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

☐ Please enter my 1 year (12 issue) subscription to COMPUTE!
☐ New subscription ☐ Renewal subscription
☐ $20.00 U.S. subscription
☐ $25.00 Canada and international surface mail. Payment in U.S. funds, drawn on U.S. bank or money order
☐ Bill me (U.S. only) $1.00 billing fee

Name, Address: ____________________

____

____

____

Charge my: ☐ Visa ☐ MC
Number____________________ Expires______/______/
5 6 7 8 9 10 11 12

Place
Stamp
Here

COMPUTE.
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

IF YOU CAN WAIT A MINUTE, WE CAN SAVE YOU $1,000.

With the Starwriter™ Daisy Wheel 25 cps printer from C. Itoh.

A business letter, written on a 45 cps word-processing printer, might take about two minutes to print.

With the Starwriter, it might take closer to three.

The typical 45 cps printer retails for about $3,000.

But the Starwriter 25 retails for about $1,895—thus saving you about $1,000.

And therein lies the biggest difference between the Starwriter 25 and the more expensive, daisy wheel printers.

The Starwriter 25 comes complete and ready-to-use, requiring no changes in hardware or software. It uses industry-standard ribbon cartridges, and it's "plug-in" compatible to interface with a wide variety of systems, to help lower system-integration costs.

Using a 96-character wheel, it produces excellent letter-quality printing on three sharp copies with up to 163 columns, and offers the most precise character-placement available, for outstanding print performance.

C. Itoh's warranty;

3 months on parts and labor, supported by one of the best service organizations in the industry.

$1,000 OFF

Leading Edge Products, Inc., CM-5
225 Turnpike Street,
Canton, Massachusetts 02021

Dear Leading Edge:
I'd like to know more about the Starwriter, and how spending a minute can save me a grand. Please send me the name of my nearest dealer.

Name ______
Title ______
Company ______
Street ______
City ______
State ______ Zip ______
Phone: Area Code ______
Number ______

LEADING EDGE.

Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Dealers: For immediate delivery from the Leading Edge Inventory Bank™ call toll free 1-800-343-6833
In Massachusetts, call collect (617) 828-8150. Telex 951-624

The Great American Solution Machine.

Meet Commodore. The business computer that's providing solutions for more than 100,000 people all over the world. Built by one of the pioneers in office machines. With a reputation for quality that can only come from vertical integration of the total manufacturing process. Commodore builds, not assembles.

Compare Commodore's word and data processing capabilities with computers costing twice or even three times as much. You'll see why so many small businesses are turning to Commodore for solutions to problems as varied as these:

☐ A car leasing company's customers were terminating too early for account profitability. Solution: A 16K Commodore. It analyzes cash flow on over 1200 accounts, identifies those for early penalties, and even writes up lease contracts. Commodore paid for itself within weeks.

☐ A law firm needed a high quality, easy-to-use, affordable word processing system. Solution: Commodore plus its WordPro software package. At a $6,000 savings.

☐ A gasoline retailer needed to inventory, order and set prices; determine Federal and state income taxes; and comply with Federal pricing and allocation regulations. All done daily, weekly, monthly and yearly. Solution: Commodore. It keeps his business on track—and Uncle Sam off his back.

☐ A paint and wallpaper store had to inventory over 600 expensive wallpaper lines for profitability, monitor distributor sales, set and track salesmen's goals, and help the customer select the right size, pattern and quantity. Solution: Two 32K Commodore computers, floppy disk and printer. Commodore does it all—and accounting, too.

In applications like these, and many more, Commodore solves the problems that stand in the way of increased profitability. Commodore can provide the solution in your Great American business, too. Find out more by calling or writing any of Commodore's District Sales Offices.

COSTA MESA, CA 2955 N. Airway Avenue 92626. (714) 979-6307.
SANTA CLARA, CA 3330 Scott Boulevard 95051. (408) 727-1130.
DECATUR, GA 5360 Snapfinger Woods Drive 30035. (404) 987-3311.
BENSENVILLE, IL 790 Maple Lane 60106. (312) 595-5990.
NORRISTOWN, PA 950 Rittenhouse Road 19401. (215) 666-7950.
DALLAS, TX 4350 Beltwood Parkway South 75234. (214) 387-0006.

Commodore Business Machines, Inc., Computer Sales Division, Valley Forge Corporate Center. 950 Rittenhouse Road, Norristown, PA 19401.

commodore

COMMODORE