Easy Reading Of The Atari Joystick

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

Converting Pet Basic Programs To ASCII

COMPUTE!

$2.50
August, 1981
Issue 15
Vol. 3, No. 8
63379

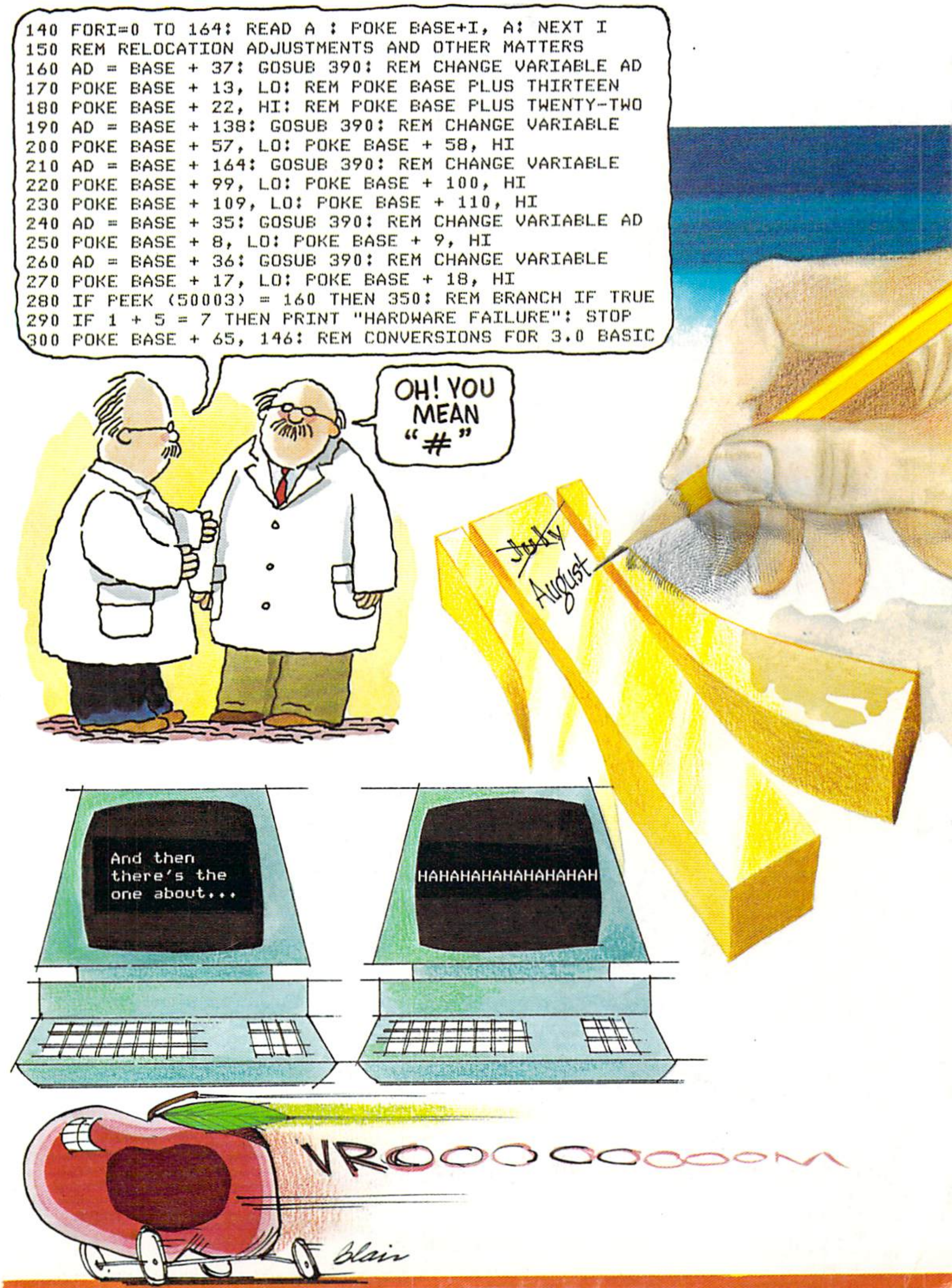
The Journal For Progressive Computing™

Keyword For The PET/CBM

Restoring And Updating Data On The Atari

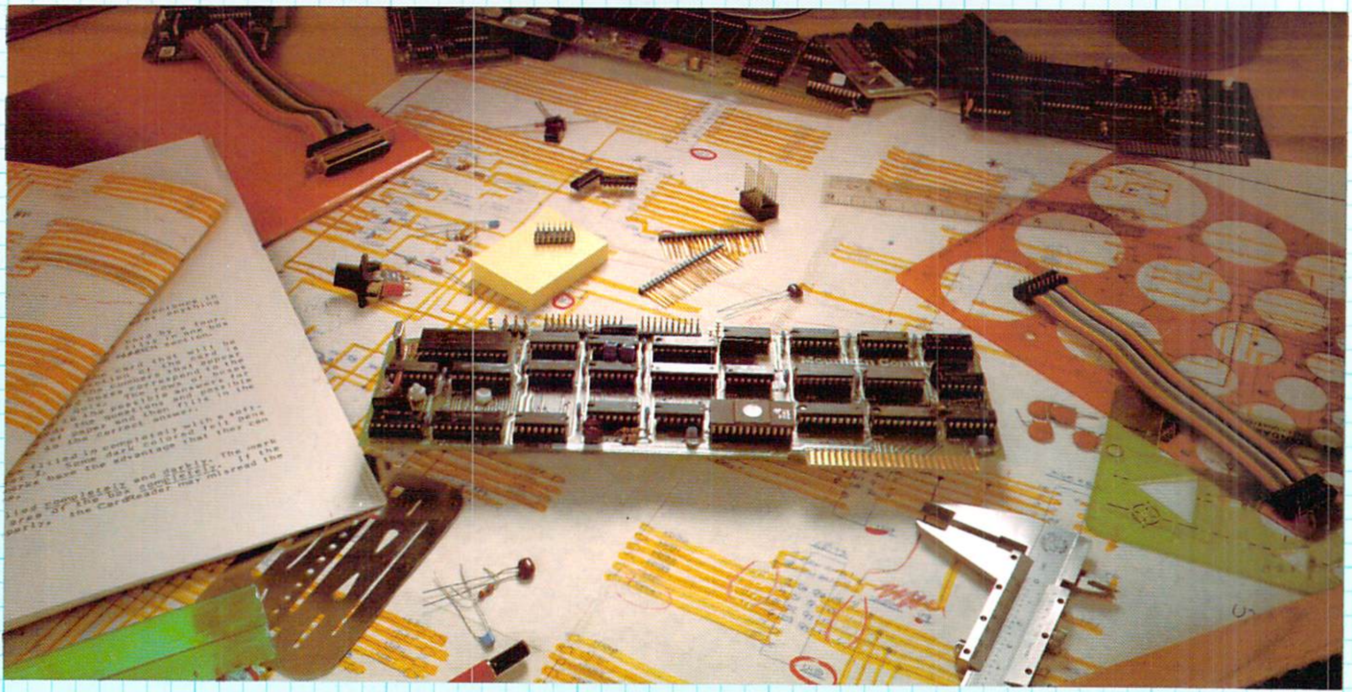
BASIC Oneliners: Minimize Code And Maximize Speed

Apple Disk Motor Control From Your Program



08
0 74470 63379

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

SPECIAL INTRODUCTORY PRICE
$239

Mountain Computer
INCORPORATED

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

™Apple Clock was the trademark of Mountain Computer Inc.

®Apple and Apple II are registered trademarks of Apple Computer Inc.

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

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

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

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

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

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

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

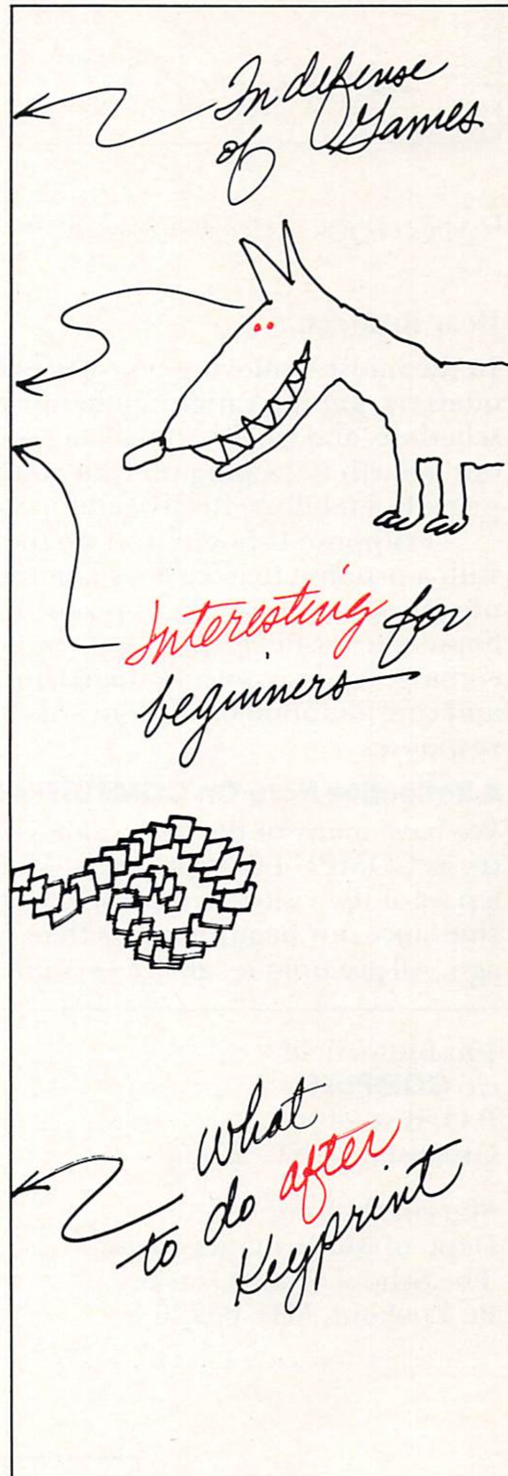
Complete information? It's at your dealer's now. Or, we'll send it to you and include a dealer list. Write us. Call us.

MICROSOFT
CONSUMER PRODUCTS

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

Table of Contents

August, 1981, Vol. 3, No 7



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.

The Editor's notes...

Robert Lock, Editor/Publisher

Dear Readers,

In the midst of moving our offices to larger quarters, getting caught up in our production schedule, and finally convincing ourselves that our growth is keeping up with yours, I'd like to provide a totally reflective editorial.

I suppose I should preface these remarks with a promise that we'll reopen the question of software and copyright protection in the September issue. Please keep your comments coming. And my sincere thanks for the time and consideration obvious in your recent responses.

A Reflective Note On COMPUTE!'s Columnists

We have many of the best writers in this industry as COMPUTE! supporters. Most have been a part of the vision of this user resource magazine since our beginning less than two years ago. All are unique, all are versatile, all are enthusiastic.

I realized this week, while contemplating our growth, the tremendous value to me of the simple mechanism of Editor's Feedback cards. This issue we're sharing. You'll find them replaced by several Author's Feedback cards. If you've never written your favorite columnist a letter, take the time to drop him or her a note. And thanks again, not only for your support of **COMPUTE!**, but also for your support *as* **COMPUTE!** We're that kind of magazine — our columnists share a belief that this industry of ours will soon touch — visibly — the lives of much of the world. And all encourage that.

Send a postcard. They'll appreciate it, and grow stronger from your feedback. I know I have in the last two years. Thanks!

Robert Lock

Jim Butterfield
c/o **COMPUTE!**
P.O. Box 5406
Greensboro, NC 27403

Marvin De Jong
Dept. of Mathematics-Physics
The School of the Ozarks
Pt. Lookout, MD 65726

Eric Rehnke
1067 Jadestone Lane
Corona, CA 91720

Susan Semancik
c/o Delmarva Computer Club
P.O. Box 36
Wallops Island, VA 23337

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

Gene Zumchak
1700 Niagara Street
Buffalo, NY 14207

Production Note: *As we move* **COMPUTE!** *to an earlier schedule, we've delayed the introduction of the first two* **COMPUTE! Books** *for four weeks. Thanks for your patience. RCL*

Subscription Information

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

TOLL FREE
Subscription
Order Line
800-227-1617
In CA 800-772-3545
Please ask for Extension 401.

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 represented by **The Gittelman Company.** You can reach them through the following offices:

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

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

If you're in the **Midwest, we're now represented by GB & Associates.** You can reach them through the office of:
GB & Associates
P.O. Box 335
Libertyville, IL 60048
312 362-1821
Gordon Benson

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

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

Telephone: (919) 275-9809

Ask The Readers

Robert Lock And Readers

Here are this month's questions... If you have an answer to any question here (or a question of your own), send it to "Ask The Readers," **COMPUTE!** Magazine, P.O. Box 5406, Greensboro, NC 27403.

"I am MOST happy to see articles beginning to appear that deal with the newer disc DOS (especially DOS 2.1). I am looking forward to more, I would particularly like to find a "map" that would tell me where the flag is that tells DOS 2.1 when the disc has been inserted, and assorted things like that.

Another important area that has not been covered by ANYBODY (including COMMODORE) is maintenance of the disc drives! My 8050 has been operating in a NON-airconditioned environment for nearly a year now with virtually NO maintenance! I find it hard to believe that it is not due for a "grease job & oil change" soon. The poor thing faithfully serves 2 PETs, an ancient 2001 (with 24K Expandapet & an MTU Integrated Video Memory), and a newer 'religious' 2001 (holy main PCB that I wired around to bring it up to 32K). Both units use BASIC 3.0, one has a DISC-O-PRO/TOOLKIT). I want to do right by it — HOW?

I just got hold of an English PET magazine — WOW the ROMs that they advertise! Why haven't US companies been producing like that?

I'd also like to mention that a friend of mine recently purchased AB Computers' PAPERMATE (on my advice) and got the same good service that I got last year when I bought PAPERMATE (good software at a reasonable price I won't copy for a friend)." R. Vanderbilt Foster

Here are some excellent ideas. The CBM disk drives are full of still unexplored code, waiting to be mapped. Also, **COMPUTE!** is looking into the question of drive maintenance — we hope to provide some definitive answers soon. Anyone with experience in this area, please share your views.

"An electronics instructor at our school is interested in purchasing film, filmstrips, cassettes, and videos to explain computer capabilities and functions for his students. Do you have sources of such information?" Rita Norton

---

"I wonder if you could give me some helpful information on a problem that I have with my computer system. I have an ATARI 800–48K RAM computer.

I recently purchased an ATARI Assembler Editor cartridge and I have had trouble ever since. Here's what happens: on page 68 of the Assembler Editor Users Manual is an Example No. 1 program.

I type this program into my computer and save on tape recorder 410 model. It appears to me that it seems to be being recorded on the 410 recorder. I get about 12 tone beeps while listening to the TV speaker while saving the assembler program and then it stops.

Then I replace the Assembler Editor cartridge with the BASIC cartridge and attempt to load the program from the tape recorder. But after about 3 tone beeps, the recorder stops and the screen will display either a dark screen or an endless loop of fast moving, what appears to be lowercase characters and graphic symbols. Almost too fast to read.

I have spoken to the ATARI people in Caifornia several times — received their errata manual — followed the corrected instructions and still have no success in loading tape program into the computer.

I haven't gotten past the Example 1 program on page 68 or the Assembler Editor Users Manual. If I don't solve this problem I might just drown the Assembler cartridge in the Potomac River. HELP!!!" Tony Pilato

Several readers have raised this question. See if this solves the problem for you — before heading for the river.

```
100 GRAPHICS 0:TRAP 230
110 PRINT "Insert tape, press RETURN"
120 OPEN #1,4,0,"C:"
130 GET #1,X:GET #1,X
140 GET #1,X:GET #1,Y
150 START=X+256*Y
160 GET #1,X:GET #1,Y
170 FINISH=X+256*Y
180 PRINT "Code being loaded at ";START
190 FOR I=START TO FINISH
200 GET #1,X:POKE I,X
210 NEXT I
220 PRINT "Code ends at ";FINISH
230 CLOSE #1
240 END
```

"Is there any way of saving a PET/CBM program and have its variables and strings saved with it, in such a way that it can be reloaded and continue processing, from a

cost of the world.

Access to all the CompuServe services, including electronic mail; bulletin board; CB simulation; newsletters from Radio Shack®, Atari®, and others; up to 128K storage free plus all the services listed here are yours. Access time is only 8⅓ cents per minute between 6 PM and 5 AM weekdays and all day weekends, billed to your charge card. It's a local phone hook-up in more than 260 U.S. cities.

There's an energy management system, home repair tips, discount information, personal health information and more. Better Homes & Gardens provides household information such as recipes, nutritional analyses and meal plans.

All kinds of games (including some for the kids), book and movie reviews, astrology and the latest sporting event news — even point spreads and injury lists. There's the CompuServe line printer art gallery and even a national user opinion poll for instant feedback on current events.

A complete program. The AP financial wire, newspaper financial pages, Raylux Financial Advisory Service, the Commodity News Service and personal financial programs. At a slight additional charge, you can retrieve fundamental financial data on more than 1,700 companies from Value Line, and MicroQuote offers current and historical data and special market reports on more than 40,000 stocks, bonds and options, updated daily.

While the CompuServe Information Service is designed for everyone, MicroNET Personal Computing is for the knowledgeable computer hobbyist or business person. You can buy software and have it downloaded directly to your person computer. Use programming languages such as *APL-SF, EXTENDED BASIC, MACRO-10, SNOBOL, EXTENDED FORTRAN* and *PASCAL,* use word processing and text editing programs.

Not 1985, NOW!

See for yourself what a state-of-the-art electronic information service can do. Get a demonstration at a Radio Shack® computer center or store — or write to us for further information.

CompuServe

Information Service Division
5000 Arlington Centre Blvd.
Columbus, Ohio 43220
(614) 457-8600

main menu?

I have tried saving the BASIC vital pointers, from location 122 to 135 (original ROMs) and saving all of memory. I would then reload all of the memory and restore the pointers and it would still not work.

Can someone please help me?" John Lemkelde

"I am an Amway distributor and am looking for a distributor software package for the Apple II+ computer. I have gotten several brochures based upon other computers; however, as sales representative with two companies which handle Apple, I tend to lean to it — that's the one I'm more familiar with." McBee Barbour

---

"I have an Atari 800 and have taught myself to write programs for it. However, I want to use a sort program to alphabetize a list of names and I can't work it out.

I would appreciate it if you would ask your readers if they can help me, or have **COMPUTE!** *publish such a program if there is a demand for one. I have tried sorts written for other computers, but they don't work on the Atari." Irwin Kaplan*

You are probably trying to use sort programs written for Microsoft BASIC (Atari's BASIC is significantly different). **COMPUTE!** will soon be publishing an Atari machine language sort.

"I am writing a program for a CBM 8032 which involves input of possibly large data sets by the user. Is there a way to 'trap' an overflow error, caused by erroneous entry of a datum, so the BASIC program won't bomb out completely? I would like a way to allow 'graceful' recovery by the user, so he won't have to start his data entry again from the beginning if he makes such an error." Don Barr

---

"I have a problem — maybe someone out there knows the answer.

I bought the "VOTRAX TYPE'N TALK" text to speech synthesizer for my ATARI 800 computer. Does anyone know how to program and interface the synthesizer to the ATARI 800?

They (VOTRAX Company) must think I have ESP. The 30 page booklet they provide doesn't tell you much on programming or whether or not the synthesizer is even compatible with ATARI or any other computer for that matter.

I'm only a computer hobbyist, not a programmer. $345 is nothing to sneeze at on something I don't know how to use." Edward A. Sweeney

---

"A keypad for entering repetitive numerical data is certainly a convenience. Is the Atari 'keypad' used in the video game computer compatible with the Atari 800 and can it be used for this purpose?" Jerry Stern, M.D. ©

Computers And Society

Some Speculations On The Well-Played Game, Part 2...

David D. Thornburg
Innovision
Los Altos, CA

Last month I responded to a **COMPUTE!** guest editorial by Alfred D'Attore in which he suggested that the major legitimate use of personal computers in schools was for the generation of rote drill and practice exercises. From my perspective, based on several years experience conducting workshops, talking with teachers, and, most importantly, talking with children, I have observed that there is a great deal of benefit in having carefully chosen computer games in the classroom.

This month I want to explore two issues. First, I want to describe some games which I have found to be both enjoyable to students and well stocked with educational content. Second, I want to explore some of the possible hidden reasons behind some teachers' reluctance to allow computers in the classroom.

The games I will describe represent only a small sampler of high quality educational software. There are many more games which are both high in intrinsic motivation and which provide practice in traditional school subjects.

Because I have acquired these games from many sources over the years, and because these games reside on various computers, I want to make a brief disclaimer. First, I am not the author of any of these games, nor, in most cases, do I know who the author is. Second, several of these games appear under several names in the marketplace. I have made no attempt to identify the originator of any of these games, nor do I know the copyright status of any of them. I am describing these games only to illustrate their value in the classroom. Any reader who wishes to generate an implementation of any of these games has the personal responsibility for first determining the copyright status of the game.

Since Mr. D'Attore was most concerned with math drill, I will cover that area first. As many of you know, there are many games which require various levels of computational skill. Here are two of my favorites:

Knockdown – This game is an excellent tool for providing drill. The game can be played against the computer or against another player. Each player has a number line with the numerals one through nine on it. Each player takes turns having the computer roll a pair of dice. The player then takes the numeral corresponding to the sum of the points on the dice. If that number has already been taken, or is larger than nine, then the numerals corresponding to each die are taken. If one or both of *these* numbers is covered, then any other two numerals which generate the same sum can be taken. Play alternates until one player cannot complete a move. The second player then continues playing until he or she is similarly stuck. The player with the highest score (determined as the sum of the covered numerals) wins.

> **... describe some games which I have found to be both enjoyable to students and well stocked with educational content.**

During the course of the play, each player gets several opportunities to add two numbers as well as to perform relational tasks, such as figuring out that 3 + 4 is the same as 2 + 5.

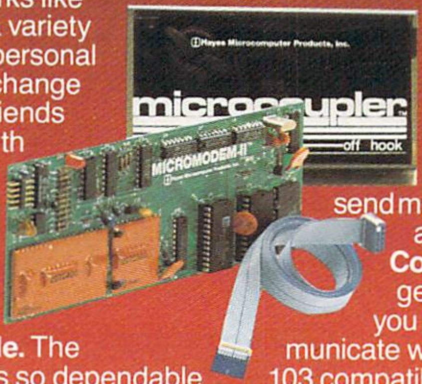
I have used this game in a first grade classroom where it was well received by both students and teacher alike. Several weeks later I was conducting a school-wide workshop at the same site and several first grade students specifically asked me if I would let them play this game again.

Not a bad response for a pure math drill game!

Maxit – My first exposure to this game came from the PET version written by Harry Saal. I have since seen implementations under different names on the Interact and on the Atari computers. A nice board-game version called "Tally Up" is made in Israel.

In Maxit, a square grid is formed of randomly chosen numbers. In some implementations of this game, both positive and negative numbers are used. Play is against the computer or against another player. At the start of the game, a marker is placed randomly on the grid. The first player is allowed to move the marker horizontally, choosing a number to capture. Choosing the number fixes the column from which the other player (or the computer) must take a number. Once a number is taken, it is added to the player's score. When the next player chooses a number from the column, this fixes the row from which the first player must take the next number. A sample grid is shown

With the Hayes direct-connect Micromodem II,™ your Apple II can communicate by phone with the outside world. You can access information networks like The Source for a variety of business and personal applications, exchange programs with friends anywhere in North America, and even use your Apple II when you're away from your home or office.



Dependable. The Micromodem II is so dependable it comes with a *two-year* limited warranty. That's another reason why it's the largest selling direct-connect modem for Apple II computers.

Programmable. Automatic dialing and answering? Of course! We include programs on disk that dial phone numbers for you, send messages while you're away, and much more!

Complete. You get everything you need to communicate with other Bell 103 compatible modems at 110 or 300 baud. The serial interface is built-in, and we even include our FCC-approved Microcoupler™ that plugs directly into any modular telephone jack in the U.S. — you don't even need the phone!

S-100, too. The Hayes Micromodem 100 gives S-100 microcomputers all the advantages of our Micromodem II!

Put the outside world *inside* your computer with a data communications system from Hayes. Available at computer stores nationwide — call or write for the location nearest you. And don't settle for anything less than Hayes.

Hayes™

The Hayes Micromodem II opens up your Apple II† to the outside world.

Hayes Microcomputer Products Inc. 5835 Peachtree Corners East, Norcross, Georgia 30092 (404) 449-8791

Micromodem II, Micromodem 100 and Microcoupler are trademarks of Hayes Microcomputer Products Inc.
†TM Apple Computer Inc. Micromodem II can also be used with the Bell & Howell computer. © 1981 Hayes Microcomputer Products Inc.

below:

3	4	3	0	2
7	1	4	3	8
5	*	6	4	1
1	5	9	4	2
0	1	9	8	1
2	1	6	5	0

The asterisk is the starting marker position.

Suppose it is the first player's turn. To maximize the score, the player could take the 6. However, the second player could then take a 9 for a net gain of 3 points. If the first player took the 5, the second player could only gain 2 points by taking the 7, in which case the first player could take the 8 and stick the second player with very low numbers.

As you can see, the game can be played at many levels. All computer-based versions I have seen play perfect games at one level of "look-ahead." Since many of the children I have seen are able to attain this level of play fairly easily, evenly matched games can be achieved with a small amount of practice.

While most players do not consciously add up each possibility and analytically determine the best choice, many people do perform the subtraction of the present choice from the probable next move in order to assess the validity of each choice. In addition to providing incentive for acquiring simple subtraction skills, this game also hones children's intuition in numeric relationships. Many children make explicit statements of their reasoning process during the play of the game; e.g., "Well, if I take the 9 he has to take the 4, but then he can stick me with the -10 and take the 15."

I have seen children spend hours on this game.

Among *reasoning* games which help develop intuition in the solution of multi-variable constraint problems, my favorites are the simulations Lemonade (from the Minnesota Educational Computer Consortium) and Hammurabi.

In both of these games the player is given control of an economic environment based on a model which has a certain level of uncertainty built into it. In Lemonade, the player runs a lemonade stand and has to purchase advertising and raw stock for each day's transactions. After setting a price, the results of the day's sales are determined. Generally, increased advertising increases sales, an increase in prices decreases sales, and so on. Chances of rain (and occasional hot summer days) are just two of the many external factors which are used to make this game more challenging.

Hammurabi (sometimes called Kingdom) has been around for ages. This simulation involves the planting, harvesting, and sale of grain, with resource management being the major goal. Success brings more workers to the city (thus requiring that they be fed), shortages of grain cause starvation, thus reducing the work force available to plant the next year's crop.

Both of these simulations are easy enough for most school age children to handle. While aiding in the development of general reasoning ability, these games help develop decision-making intuition, and the ability to make trade-offs.

Many more strategy games are useful in the classroom as well. Games such as Othello (a.k.a. Reversi, Roman Checkers) and Mastermind (or Logicolor) come to mind. The point is that there are hundreds of games which are well suited for use in the classroom. The well stocked school should

A curriculum which is too inflexible to allow the use of these games should probably not be using computers in the classroom at all.

have many games to choose from. Games like Hurkle, Hangman, Wumpus, Quest, Stars, Darts, Concentration, etc. can all find utility in the classroom for the acquisition and reinforcement of various skills in math, language, and general reasoning.

A curriculum which is too inflexible to allow the use of these games should probably not be using computers in the classroom at all.

The second topic for this month is my speculation on the origins of the concern I have heard expressed by teachers who want to keep computers out of the classroom. While many of the teachers I meet are quite excited about the use of personal computers, there are some who are adamant in their refusal to consider the use of this technology.

Many of the reasons given arise from important and potentially valid concerns. Are computers going to be under-used, high priced gimmicks, as some of the early and heavily-pushed audio/visual equipment was? Can a teacher, working against a fixed lesson plan, depend on the availability of textbook-specific software to provide an integrated program of instruction? The answers to these and other questions provide some valid points of concern which will diminish the use of computers in certain schools. It is not sufficient to say that, someday, these problems will be solved. Teacher's workloads and budgets leave little time for conducting sky-blue experiments. And yet there is one area where computers are most appropriately used in our schools, and that is in the development of computer literacy. The fact is that computers are here to stay. This technology is too powerful and too ubiquitous to be left out of the school curriculum much longer.

There are those teachers who say that the

SOFT ROM

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

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

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

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

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

$129.00

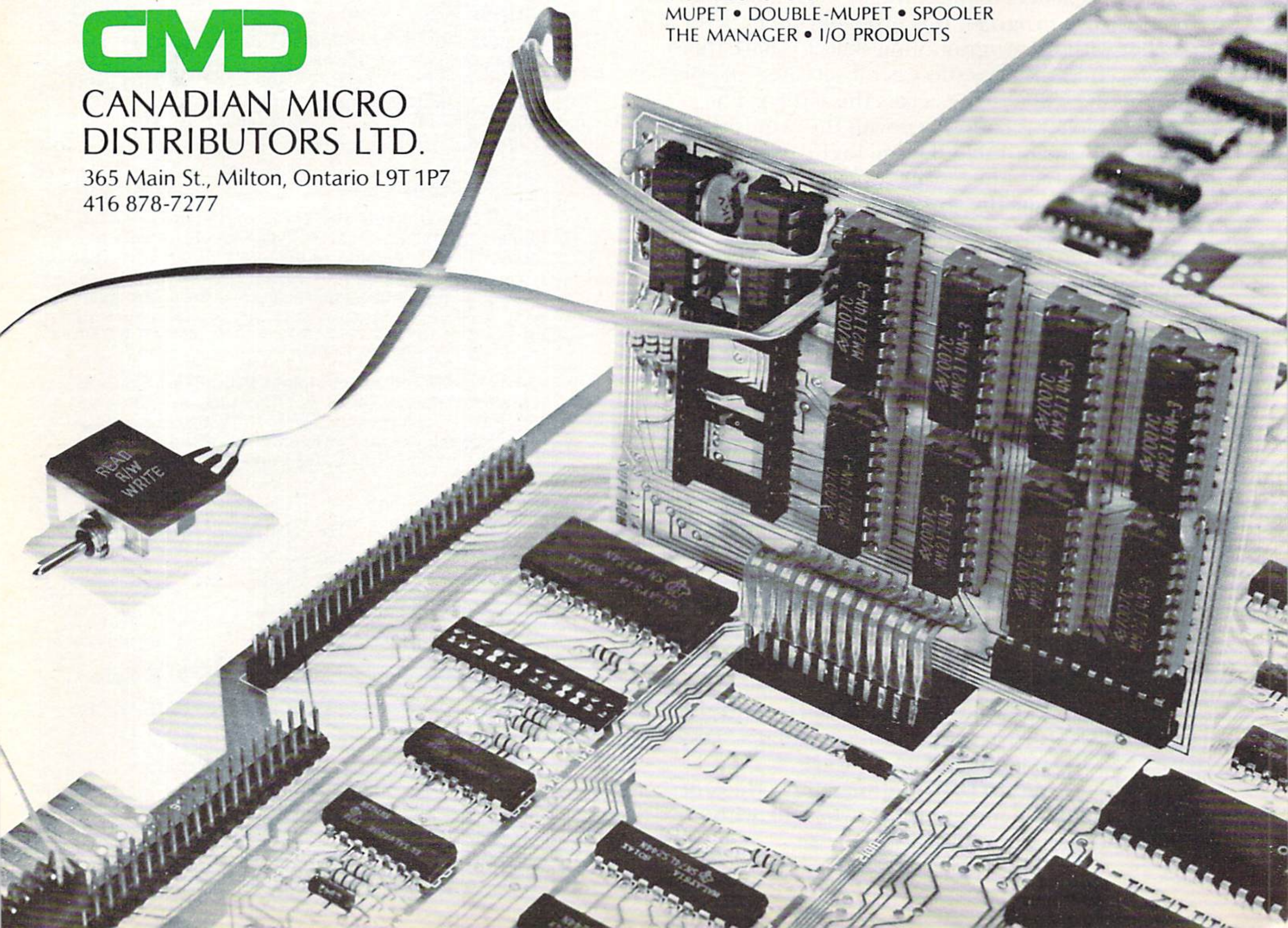
FROM

CMD

CANADIAN MICRO DISTRIBUTORS LTD.

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

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



computer is a waste of the child's time. The fervor with which I have heard this argument expressed suggests that many of these teachers have deeper fears regarding this technology than they are prepared to acknowledge.

Some youngsters gain mastery of the computer before the teacher does. Some teachers feel threatened by this. Perhaps an even deeper cause for computer-phobia among some teachers is the concern that the computer gives the student too much control over the school environment. When discoveries are being made by a child who is programming his own computer, there is no clear way for many teachers to measure, or even monitor, the student's progress. It is probably not easy for a teacher with no programming skills to share the joy of a child who has successfully created an animation of a ball moving across the screen. The more a teacher is concerned with the acquisition of specifically measurable skills in highly compartmentalized areas, the less likely she or he will be able to appreciate the tremendous amount of "learning" which takes place during the child's time with the machine.

Perhaps an even deeper cause for computer-phobia among some teachers is the concern that the computer gives the student too much control over the school environment.

From the perspective of the child, the creation of the truly low-cost computer and the associated creation of the consumer computer industry will allow our children to gain mastery of these devices anyway. It remains for the schools to decide what role, if any, they want to have in providing educational opportunities for our children in this area.

Frank Herbert Made Simple...

I have received a great many letters from readers of my review of "Without Me, You're Nothing" by Frank Herbert. Those of you who wrote, called, or electronic-mailed responses to me shared my concern for the lack of technical accuracy I noticed in the book. Of special concern was Mr. Herbert's constant use of "disk driver" when he was referring to a disk drive.

I am especially indebted to Dan Howard, from Oakland California, for pointing out that a "Disk Driver" is the pilot of a flying saucer!

Many thanks to Dan and the many others of you who took the time to write. After all, without *you*, this column is nothing! ©

The Beginner's Page

Richard Mansfield
Assistant Editor

Subroutines are one of the most important tools used in computer programming. They are, simply, small programs. (The words *routine* and *program* mean essentially the same thing.) Subroutines nestle within larger programs and provide essential services to the host program. Often a subroutine is *called* by the host many times during the *run* of the program.

The key difference between subroutines and larger programs can be illustrated by the fact that the human thinking process works the same way: large jobs are broken into smaller jobs. Each smaller task can then be solved as a separate unit. If we were not able to think in this way, we would not be able to think at all.

Subroutines and Pizza

The power of subroutines derives from two important factors: *repeatability* and *portability*. If you make a telephone call to order a pizza, your mind first breaks the problem down into sub-problems (some of them are so small that you are not even conscious that they are involved). You walk to the telephone, dial the number, hear a busy signal, replace the receiver, dial again, place your order, give your home address, and replace the receiver again. In this entire "program," your brain was mainly using subroutines.

Within the pizza-ordering job are many small jobs. Both replacing the receiver and dialing had to be done twice within the program. This is repeatability. The instructions for replacing the receiver, walking, recognizing busy signals, and so forth, would not appear *each time they were used* within a program. Rather, they are written only once, and in one place, and can be used whenever needed. Subroutines are also portable. They do not belong specifically to one particular task. They can be lifted out of one program and inserted into a different program. Dialing is also a subroutine when calling the police.

Whenever you are doing something "unconsciously," like driving, or walking to the phone, you are on auto-pilot and you are thinking about something else altogether. Long ago, your brain set aside a zone called "how to walk" and you do not need to figure out these complex muscle interactions each time you get off a chair. Instead, you call a subroutine. Your brain, like a computer, already knows how to perform many of the often-repeated jobs which, once learned, need not be "programmed" again. In the same way, you will want to build a library of subroutines for your computer.

When you first get a computer, it knows many subroutines (BASIC is a library of subroutines), but you will find that your own programming efforts will result in a growing collection of short, common routines. You might be writing a game which simulates a financial disaster and need to round some numbers to two decimal places. Before trying to write that section of the program, you

GENERAL GLOSSARY

K. Kilobyte. 1024 bytes (units of computer memory).

6502 or **CPU.** The chip (a piece of plastic with a lot of electronics inside it) within the CBM that "thinks." As opposed to the chips that "remember" such as RAM and ROM chips.

RAM. Read/write memory (can be changed).

ROM. Read Only Memory. A permanent memory. BASIC is in ROM so that when the computer is turned off, it can still remember BASIC.

Disk. A magnetic disk for storing data. Looks like a limp 45 RPM record. Also called "floppy." Used in a machine called a Disk Drive.

MODEM. MOdulator-DEModulator. Allows a computer to call other computers on the telephone and send or receive information.

IEEE-488. Pronounced I-triple-E. A standard interfacing (connecting) scheme to add peripherals (disk drives, printers, etc.) to a computer.

ASCII. Pronounced ASK-EE. American Standard Code for Information Interchange. Assigns a number (0-255) to each letter of the alphabet and other familiar symbols such as quotes, commas, etc. This is how the computer, which deals in numbers only, can also manipulate text. After A=65 and B=66, the computer can easily alphabetize.

Machine language. As opposed to BASIC and other "higher-level" languages — machine language (ML) is the *computer's* language, it's way of seeing a list of instructions (a program). What BASIC does is to stand between the human and the machine and translate English words into ML. PRINT, for example, is transformed into a long list of ML instructions which can communicate the idea of PRINTing to the 6502 "brain" of the computer.

Monitor. Or "machine language monitor" is a program which helps to simplify machine language programming. Several popular "extensions" add power and versatility to the monitor. Original ROM CBM's do not have a built in monitor.

Assembler. A program which makes machine language programming even easier by recognizing alphabetic information and translating it into true, numeric machine language.

Hexadecimal. Or "hex," is a way of counting which uses groups of 16 rather than the familiar decimal 10. When it gets to 9, it starts using A through F. It is sometimes convenient in ML work.

Garbage collection. Sometimes the computer must get rid of old strings (text variables) which are not needed anymore by a program. This frees some memory. In 4.0 BASIC this takes only seconds, but in other BASICS it can lock up the computer for minutes, with no explanation.

Integer. A whole (not fractional) number.

Floating point. A way for the computer to store long numbers efficiently, in a few bytes. FP can handle numbers from -10^{38} to 10^{38} on CBM. This is the ordinary type of numeric variable used by default in most computers. It gets its name from the fact that a decimal point can float around within numbers to represent various fractions.

Interrupt, Interrupt Request, or **IRQ.** Every "jiffy" (sixtieth of a second), the CBM stops whatever it's doing, jumps to a special little program, and then jumps back and continues whatever it was doing. Among other things, it checks to see if the STOP key is down. Other computers relate to interruptions in other ways, but all 6502 machines have this mode.

Crash (or **lockup**, or **endless loop**). The cursor usually disappears and the computer will not respond, even to the STOP key.

Parallel (and **Serial**) **Interfacing.** The two primary ways to connect devices to allow data to be accurately sent between them. Parallel is more expensive and faster. It sends an entire byte at a time (eight bits simultaneously) and needs thicker cables since it must have a minimum of nine wires. Serial sends data a bit at a time.

remember that this same rounding job was part of your income tax program. It is far simpler to lift the rounding routine out of the other program than to figure out how to program it all over again. Your financial disaster game and income taxes have nothing in common (well...), but the same subroutine can be taken from your library for both of the larger, unrelated programs.

The Rounding Subroutine

```
30000 ZQ=INT (ZQ * 100) / 100
30010 RETURN
```

Looking at the rounding subroutine, we can see a couple of odd things about it — it starts with line number 30000 and it uses a variable named ZQ. These are both good programming habits. When you are building your library of subroutines, you will want to be able to merge them easily into the larger, host programs. To do this, you will want to give them high line numbers, the highest permitted by your version of BASIC. In this way, you can stick them easily onto your other programs without needing to renumber your lines.

Interestingly, some people advocate the opposite — they suggest putting subroutines at the *start* of programs. Their reasoning is that the host program will run faster since it looks for a subroutine by going through the program from the lowest line number on up. This would be fine — as long as you always start programs with a line number such as 100 to leave room for any added subroutines. Many programmers start with line 100 to also allow space for DATA tables, REMark statements, and so forth. The essential thing is to decide on a reasonable style and stick to it each time. This permits you to more easily understand your programs when reviewing or modifying them later. It also makes the main components of *all* your programs (initialization, tables, subroutines, main loop) second nature to you. (We will explore these four primary program components in the next issue.)

The other odd thing about the rounding subroutine is the variable name "ZQ." It is chosen because it is so unlikely sounding that you will probably never pick it to use as a variable name in a host program. You would run into problems if your subroutine used a variable which was also being used by the host program for something unrelated. For example, if the subroutine caused a variable such as "N" to be defined as 53.25 and your host program were also using the variable "N" for the number of days in a week ... it would be a mess. So, before you GOSUB to the subroutine, you change the host program's variable into the subroutine's variable (ZQ=N) and, when you get back from the GOSUB, you give the result back to N again. The subroutine "call" would look like this:

```
150 ZQ=N: GOSUB 30000: N=ZQ
```

Subroutines have a major impact on programming style and they must be handled carefully. Books have been written on the subject of good programming practices. In general, it is less important that you follow arbitrary "practices" than that you develop consistent programming habits. Of course, it would be ideal if the rules of program composition had settled down into a universally accepted canon of guidelines. Many fine minds are presently occupied by attempts to develop a "best" programming language which will, it is hoped, force programmers to write clear, easily understood, and efficient programs.

Pessimists argue that the grammars of the world's languages never settle down in this fixed way, that BASIC has taken hold and will never yield, or that even a fixed set of linguistic rules cannot eliminate confusing, wasteful, or illogical programming. In addition, some feel that clarity and efficiency are mutually exclusive goals (when efficiency is defined in terms of memory space and speed of program execution). Nevertheless, these dark possibilities aside, it is surely worthwhile to attempt to be clear and efficient in your own programs.

Recursion

You are bound to see this term sooner or later. You would look for it in vain in most dictionaries — even unabridged editions will not list it. The word came into computing from both mathematics and logic. It means different things to each of those disciplines and, you guessed it, there is a third meaning in programming lingo. It has a meaning very close to recur, but it is a peculiar kind of recurrence. It means a subroutine that "calls" *itself*. This mind-bending idea is sometimes not allowed by a computer language — FORTRAN, for instance prohibits recursion. Most programming style experts suggest that you avoid it. (They also suggest you avoid GOTO, whenever possible, and self-modifying programs, but we'll deal with that another time.)

To visualize a subroutine which uses itself, we can write a slightly more complex subroutine:

```
150 ZQ=N: GOSUB 30000
............
30000 ZQ=ZQ/2
30010 IF ZQ>5 THEN GOSUB 30000
30020 RETURN
```

This complicated situation will cause a series of GOSUBs until the number is finally lower than five. You can see the problem here: we might have a large number of GOSUBs nested within each other for the computer to keep track of. (The computer remembers an "address" when it GOSUBs, so it can RETURN later and pick up where it left off.) So, when the program finally gets to RETURN ... where does it go? Back to the original GOSUB on line 150? Or back to line 30010?

The computer keeps track of these "return addresses" by stacking them up (in its *stack*) as the

Osborne AWAKENS the ATARI™ and Puts it to Work.

All too often, computer users who've been used to entertainment programs have trouble coming up with ideas for **practical** computing. Your computer plays space games well enough, but when you have a down to earth problem like the future of an investment or federal withholding taxes, can it perform? It can now.

Osborne/McGraw-Hill introduces **Some Common BASIC Programs for the Atari™**. This book contains 76 short, brilliantly documented programs that you can key directly into your **Atari™ 400 or 800** computer. You'll get a lot of math power including personal finance, taxes, and statistics. All the programs are listed below.

- Future of an Investment
- Future Value of Regular Deposits (Annuity)
- Regular Deposits
- Regular Withdrawals from an Investment
- Initial Investment
- Minimum Investment for Withdrawals
- Nominal Interest Rates on Investments
- Effective Interest Rate on Investments
- Earned Interest Table
- Depreciation Rate
- Depreciation Amount
- Salvage Value
- Discount Commercial Paper
- Principal on a Loan
- Regular Payment on a Loan
- Last Payment on a Loan
- Remaining Balance on a Loan
- Term of a Loan
- Annual Interest Rate on a Loan
- Mortgage Amortization Table
- Greatest Common Denominator
- Prime Factors of Integers
- Area of a Polygon
- Parts of a Triangle
- Analysis of Two Vectors
- Operations on Two Vectors
- Angle Conversion: Radians to Degrees
- Angle Conversion: Degrees to Radians
- Coordinate Conversion
- Coordinate Plot
- Plot of Polar Equation
- Plot of Functions
- Linear Interpolation
- Curvilinear Interpolation
- Integration: Simpson's Rule
- Integration: Trapezoidal Rule
- Integration: Gaussian Quadrature
- Derivative
- Roots of Quadratic Equations
- Real Roots of Polynomials: Newton
- Roots of Polynominals: Half-Interval Search
- Trig Polynomial
- Simultaneous Equations
- Linear Programming
- Matrix Addition, Subtraction, Scalar Multiplication
- Matrix Multiplication
- Matrix Inversion
- Permutations and Combinations
- Mann-Whitney U Test
- Mean, Variance, Standard Deviation
- Geometric Mean and Deviation
- Binomial Distribution
- Poisson Distribution
- Normal Distribution
- Chi-square Distribution
- Chi-square Test
- Student's t-distribution
- Student's t-distribution Test
- F-distribution
- Linear Correlation Coefficient
- Linear Regression
- Multiple Linear Regression
- Nth Order Regression
- Geometric Regression
- Exponential Regression
- System Reliability
- Average Growth Rate, Future Projections
- Federal Withholding Taxes
- Tax Depreciation Schedule
- Check Writer
- Recipe Cost
- Map Check
- Day of the Week
- Days Between Two Dates
- Anglo to Metric
- Alphabetize

Some Common BASIC Programs ed. by Lon Poole:

Atari™ edition - #53-5, $14.99 ☐
PET/CBM™ edition - #40-3, $14.99 ☐
TRS-80™ level II edition - #54-3, $14.99 ☐
Generalized BASIC edition - #06-3, $14.99 ☐

Practical BASIC Programs ed. by Lon Poole
Contains 40 additional programs with a wide range of home and business applications. (Generalized BASIC edition only) #38-1, $15.99 ☐

Make check payable to: **OSBORNE/McGraw-Hill**
630 Bancroft Way, Berkeley, CA 94710 **Dept. G6**

Phone: (415) 548-2805
TOLL FREE: 800-227-2895 Outside California

Name ______________________

Address ______________________

City/State/Zip ______________________

Plus ☐ .75/item 4th class ☐ $1.50/item UPS ☐ $2.50/item Air Mail ☐ $10.00/item Overseas
(California Residents add applicable tax.) ☐ Please send me your free catalog.

Total amount enclosed $__________ or charge my ☐ Visa ☐ Mastercard #__________ Exp. Date ____

Signature ______________________

GOSUB's come in (and removing each one when a RETURN is completed). Only the more twisted among us will ever want to enter the house of mirrors that this sort of self-referential programming involves. It is fine to have one subroutine call another subroutine. We can all easily see that the subroutine of walking calls the subroutine of balancing on our feet. But recursion, walking that calls walking, is hard to visualize.

So, at least until later, forget recursion. If, on the other hand, you enjoy Alice-in-Wonderland thinking, and most of us do in certain moods, you might want to look into recursion a bit more deeply. There is a fine book dealing with self-referential systems which is excellent brain exercise: **Godel, Escher, Bach: An Eternal Golden Braid**, by Douglas R. Hofstadter. It asks that you try to imagine things like starting to walk while you are walking. ©

Basically Useful BASIC

Checking Randomness Of Random Number Generator

Rick Keck
Overland Park, KS

Many computer programs make use of a random number generator to provide a random stimulus for use by programs. A random number generator is often used in setting up game situations. This ensures a different set-up or "situation" each time the game is played. More professional uses of the random number generator include its use as a stimulus for statistical analysis. Whatever the use may be, random number generators have become an important part of any computer system and are used by a variety of software applications.

Today, computer systems have a built in random number generator. A common name for the built in function is RND. Thus to use it to retrieve a random number, an invocation of the function would be made as follows: RND(.52819446). This would return a value greater than zero but less than one. This first value used to get the random number generator going is referred to as a "seed." All numbers used in calculating another random number can be the results from the previous random number retrieval. Hence, a program can be set up so that the random number generator "feeds" itself.

You can make your own random number generator and use it instead of the one the computer has. Random number generation theory is a field of its own with many books written on the subject. Generally speaking, a combination of mathematical operations (multiplication, division, addition) is performed with prime numbers along with a user-provided number which varies in value each time the function is used.

This brings about the important question "How good is my random number generator?". This can be determined by finding out how "random" the values are that it returns. The best way to find this out is to test it with an analytical program that will show whether a random number generator has a tendancy to return values in one range over values of another range. Ideally, the resulting value range count numbers generated by the program shown would all be nearly equal in value. Be sure to test your random number generator a few times with a different "seed" value each time to ensure a valid testing. ©

"How good is my random number generator?"

Trenton: The Original Computer Festival

Jim Butterfield
Toronto, Canada

The Trenton Computer Festival was a going concern back in 1976 (yes, Virginia, there were microcomputer systems then) and is still going strong. This year it was held on the weekend of April 25th. It tends to be an informal gathering. The commercial exhibit area is modest in size, the flea market is huge, and there's good opportunity to set up shirt-sleeve sessions in one of the many available classrooms.

Touching upon a few highlights that caught my eye ...

The PET/CBM Forum: Is there a compatability problem?

Ron Kushnier, the forum chairman, opened the panel discussion with a history of Commodore products and commented on compatibility of the various lines. He solicited attendees' opinions on whether there was a need for greater liaison on present and future systems to ease the pain of program adaptation.

The panelists were generally optimistic and felt that past evolution hadn't really been a major stumbling block to progress. A sampling of panelist opinion: "*The IEEE bus isn't completely implemented in the CBM/PET; despite its limitations (which can be gotten around) it's a valuable feature of the product line*" — Chet Nowicki. "*You quickly get used to coding to allow for the various machine configurations*" — Frank Covitz. "*We should welcome the changes, since changes mean progress*" — Elizabeth Deal. "*VIC and Mini-Mainframe systems will herald new changes ... but you'll write better programs if you code to be fairly machine-independent*" — Jim Butterfield. "*We've* all had difficult moments — but things today look good in terms of support and information" — Gene Beals.

The group was small but enthusiastic; and when the forum's allotted time was up the whole session moved over to an empty classroom and kept going for an extra hour.

One of the objectives set out by Ron in his opening remarks was the possibility of setting up a steering group whose purpose would be to co-ordinate questions of change and compatibility. The session ended without a clear position on this; it was assumed the panel members formed a good basis for the steering group, but it wasn't established how such a group would function, or how badly it was, in fact, needed.

The questions didn't end with the meeting, however. Ron passed out questionnaires which solicited opinions from attendees, and will be looking over the situation and recommending further action. If you have a comment on the subject, or would like to fill out the questionnaire yourself, you should contact: Ron Kushnier, 25 Wendy Way, Richboro, PA 18954, Phone (215) 364-2711.

The Long-awaited MTU-100 Systems: Still waiting.

Rumours said that the MTU-100 system might be shown at Trenton. Not so: but MTU were talking in confidence to a few selected people. Details are still confidential on this 6502-based system, but it's not hard to make some close guesses as to what the system will finally look like. All you have to do is to look at the current MTU catalogue and you can see the trends ...

MTU have made a name for themselves on high-resolution graphics for the 6502, and on music synthesis hardware and software. They have introduced floppy disk controller boards, and a comprehensive 6502 disk operating system called CODOS. Memory expansion has always been part of MTU's hardware offerings, and a recent MTU memory system ("The Banker") is built for "... the 18 bit address bus 6502 based systems of the future". Eighteen bit addressing on a 6502? That would mean you could fit up to 256K of memory. How is it done? MTU aren't talking ... yet.

So ... with a little cutting and pasting on MTU's existing catalogue, we come up with a system that will have up to 256K of memory, a strong disk system with both hardware and software, and built-in high-resolution graphics and full sound capability. It sounds quite tantalizing.

The Osborne I: A new style of packaging.

The Osborne I is a Z-80 machine, so a detailed summary would be out of place here. It was on display for one day only. The fascinating thing about this machine is the way it is packaged, both physically and price-wise.

The complete system — disks, CRT and keyboard — folds together into a portable package that is compared with an attache case. It's a rather oversize attache case, but nevertheless, the whole thing is truly portable. The screen is rather small, and has a 40-character line width; I understand that larger screens can be fitted externally. There's no printer, which is rather disappointing.

More interesting than physical size is the pricing philosophy. For an advertised price of

$1795, you get not only the computer, keyboard, screen and disks ... you also get two major software packages: a word processor and a financial worksheet program.

This type of software "bundling" may have an impact throughout the industry. Users typically don't want to buy boxes — they want working systems, and a package that includes software may be well-received. Other manufacturers may need to ponder the Osborne approach.

The Osborne system, with its Z-80, will interface CP/M; and this will give the user access to a large public domain library. It's hard to say whether this will generate extra commercial interest, but a lot of S- 100 hobbyists perked up at the mention of CP/M.

General Impressions

Frank Covitz and Cliff Ashcraft were at Trenton to demonstrate their latest achievements in computer music synthesis. For those who have not been exposed to the Diatonic Duo, they put on a virtuoso display of 6502-synthesized instruments and music. Listeners familiar with their work will find their latest exploits of interest: using a Fast Fourier Transform program developed by MTU they have been analyzing real world sounds and then resynthesizing them ... with the result that Frank strikes a key on his PET and the machine sings "Listen!".

Hal Chamberlin of MTU was also present at the music synthesis session and played some of his off-line generated music: the computer works much longer to make up the music than its actual playing time. Result: sound of remarkable quality.

A firm called Robot Mart was selling robot arms for $289. The mind boggles: I can visualize the PET reaching around and turning itself off. I quite liked an Atari game called NERD, which was a quiz for which there were no correct answers. MTU's visible memory was shown for 80-column CBM machines.

The flea market was huge. It was largely home-brew oriented, but there was a choice of everything from books to lapel pins.

In one corner of the flea market area, a lonely vendor was sitting by a fairly recent CBM system with a sign: Greatly Reduced! I asked him if he was moving on to another machine, or getting out of home computers. The reply: it had to be sold as part of a divorce settlement.

I thought about that one for a moment, and then asked: "Was the computer in any way responsible for the divorce?" The answer was affirmative: financial problems, including the computer, brought the whole situation about.

I knew the PET was a versatile machine ... but being named, more or less, as co-respondent? ©

BASIC Oneliners: Minimize Code And Maximize Speed

G. H. Watson
Physics Department
University of Delaware
Newark, DE

Editor's Note: While this article refers to the PET, many of the suggestions apply to all BASICs. ...RM

"I wish this would run faster!" Whether you analyze data or play games, sooner or later you will wish that your BASIC programs were faster. A frequently-used program which is time inefficient is wasting your valuable time with each use. Perhaps you've begun to speed things by deleting REM statements and spaces. What else may be done? My suggestion is to pack as much as possible into each line of the BASIC program.

By cramming each line we will be able to delete many lines from the program. With fewer lines the program will take up less memory and execute faster. The reasons are as follows:

1) Five bytes of RAM (programmable memory) are used for each line in addition to its contents: two bytes for the line number, two bytes for the line link (memory location of next line), and one byte to signify the end of the line. Every line we eliminate will save five bytes. OK, only four bytes; one byte will be used for a colon which is needed as a delimiter between multiple statements.

2) When a GOTO or GOSUB is encountered, the line number to which the program transfers must be found by beginning at the first line number in the program and searching through each consecutive line number until a match is made. In general, if there are fewer line numbers the program will run faster. For the same reason, subroutines which are called many times should be placed at the beginning of the program instead of at the end (TEST 1).

3) Programs execute faster when there are fewer line transfers. By keeping control in one line as much as possible, needless time-consuming hopping is avoided (TEST 2).

Obviously most BASIC programs are not one line in length. There are three serious limitations to the minimum number of lines which may represent a program.

1) The BASIC *editor* is the set of ROM subroutines which transfer BASIC statements entered from the keyboard into programmable memory and allows editing of a program. The editor limits the length of each BASIC line to two screen lines (80 characters). Since BASIC statements (several characters in length) are represented in memory by keywords (one byte) the line will usually be shorter than 80 bytes. The maximum length of a BASIC line which can be run is 255 bytes. A different editor could allow longer lines than the current screen editor (see R. Baker's COMPACTOR in **COMPUTE!** Sept./Oct. 1980).

2) The conditional statement IF..THEN.. drops program control to the next line when the IF.. fails. Each IF..THEN.. requires a succeeding line number.

3) Each GOSUB or GOTO requires a line number to which program control is transferred.

...BASIC statements are stored in memory as keywords by the editor.

By examining possible ways of sidestepping these limitations, we can stuff BASIC lines and make a faster running program. Limitation 3) is strict and may be avoided only through reduced use of GOSUB and GOTO. Most of my suggestions will deal with overcoming limitation 1) by placing as much as possible in each line.

A) As I've mentioned, BASIC statements are stored in memory as keywords by the editor. In many cases, the editor understands an abbreviated form of the statement — the first letter of the statement followed by the shifted second letter; e.g. OPEN may be entered as oP. Be careful if different statements begin with the same two letters; e.g. READ may be entered by rE. RETURN must be entered as reT. (No abbreviation is available for INPUT.) This is a great saving of line space — PRINT TAB(9) is ?T sh A 9), five characters in place of ten.

Care must be used with this technique. If a line is entered with many abbreviations and then LISTed, it may fill more than two screen lines. If a change is made in the line the editor will transfer only 80 characters. To make changes properly, the line must be reduced again, via abbreviations.

B) The 80 character limit to line length includes the line number. By reducing line numbers from the 10000's to 100's you will save 2 characters on each line, making room for more instructions. Note though, that no RAM memory is saved since


CBM

each line number occupies 2 bytes regardless of size.

In addition, any GOTO or GOSUB statement will be shorter with the lower line numbers. Here RAM memory will be saved since these numbers are stored as ASCII characters. Time will also be saved with the smaller numbers since less time is spent reading the number (TEST 3).

C) By using single character variable names, you save one character in line space and one byte of memory each time the variable is referenced. A slight savings in time results since the second character need not be read by the BASIC interpreter (TEST 4).

D) When an element of an array is used more than once without changing value, time, and possibly space, will be saved by assigning the element to a simple variable (TEST 5). Note that you cannot use this trick when reassigning the element; e.g. A(I,J) = A(I,J)/5 cannot be replaced with A = A(I,J): A = A/5.

E) Eliminate unnecessary parentheses in algebraic and logical expressions to save line space. Sometimes you gain speed (TEST 6); sometimes you lose speed (TEST 7).

F) Replace IF A<>B THEN.. with IF A-B THEN.. and IF A<>0 THEN.. with IF A THEN.. for a small saving in line space and a large saving in time (TEST 8 and 9). When appropriate, one character may be saved by using IF P<Q THEN.. rather than IF Q> = P THEN.. with a slight reduction in time (TEST 10)

G) It is generally recommended to drop variable references after the NEXT statement. One character is saved and the time spent in turning the loop is lowered dramatically (TEST 11). A very slight increase in speed will occur by terminating the NEXT with a colon (TEST 12). NEXT:NEXT is faster than NEXT J,K and the line space saved by the latter is eliminated when the abbreviation nE is used.

H) Make full use of the multiple argument capability of statements where appropriate; e.g. READ M,N instead of READ M:READ N saves space.

I) Print statements will generally work without the semicolon as delimiter; PRINT X;"+";Y"=";Z may be replaced with PRINT X"+"Y"="Z.

K) If a number is used several places in a program it may save space to represent it with a variable. In addition, if the variable table is not exceedingly large, it will be quicker for the PET to look up the variable in the variable table rather than convert the number to binary floating point each time (TEST 13). Note also that often it is quicker and shorter to do a simple operation rather than to convert a number to floating point (TEST 14).

While studying this problem I noticed two cases where a constant need never be defined by a variable. One is the case of pi, where the PET has a special key. Notice how fast it is assigned (TEST 15). I suspect that no conversion is done — the floating point number is stored in the ROMs. Almost unbelievably the other constant is zero. As far as I can tell the period (.) can play the rolls of zero (0) on the PET; e.g. V = 0 may be replaced with V = . (TEST 16). No space is saved but notice the substantial saving in time. Think of how many times the number zero is used in a program! Use this trick and puzzle your friends.

L) Boolean algebra statements may be used to form advanced conditional statements. The additional line needed for two IF..THEN..'s is avoided by using IF..OR..THEN... Where possible, though, I try to avoid these statements as they tend to be slow. Note that IF..THENIF..THEN.. is equivalent to IF..AND..THEN.. and much faster (TEST 17).

If a number is used several places in a program it may save space to represent it with a variable.

M) Use of the ABS statement (absolute value) may save time and space. Replace IF P<2 OR P>2 THEN.. with IF ABS(P)>2 THEN.. and replace IF P>-2 AND P<2 THEN.. with IF ABS(P)<2 THEN.. (TEST 18).

Limitation 2) is a severe one. By its very nature an IF..THEN.. is used to determine failure or satisfaction of a condition. On failure the program automatically drops to the next line, with no way around it.

One possibility is to replace IF..GOTO.. with ON..GOTO... Consider the following example ON X GOTO 220,240. When X = 1 the program will transfer to line 220, when X = 2 the transfer will be to 240, otherwise the program will finish out the line. That is, we can make a conditional branch without dropping to the next line on failure. This has limited utility in replacing IF..GOTO.. since the argument of the ON must be between 0 and 255 to avoid an ?ILLEGAL QUANTITY ERROR. The technique can be used readily on flags and counters with maximum value less than 255. For other types of variables, SGN (sign of) and ABS may be used to keep the argument in range. The following examples are possible although speed and space will be sacrificed because of the extra computations required:

Testing if a flag is set to one —
ON FG GOTO.. for IF FG GOTO..

Detecting a negative number —
ON SGN(X) + 2 GOTO.. for IF X<0 GOTO..

Detecting zero —
ON SGN(X) + 1 GOTO.. for IF X = 0 GOTO..


JINSAM™ 8.0
space
age
micro
software
USA
Used at NASA,
Kennedy Space Center
With Multiple Applications Related
to the Columbia Space Shuttle Project

Another possibility exists for handling conditionals used to increment or decrement counters. Try the following quick experiment: Enter X = 1 into the PET. -1 will be printed when you enter ?X = 0 is entered. That's right; an expression results in -1 if true and 0 if false. Thus the following are possible ways to avoid IF..THEN..:

Incrementing counter if X = 0
CT = CT-(X = 0) for IF X = 0 THEN CT = CT + 1 (TEST 19)

Decrementing twice if Y<0
CT = CT + 2*(Y<0) for IF Y<0 THEN CT = CT-2

Doubling if Y>1
CT = CT-CT*(Y>1) for IF Y>2 THEN CT = CT + CT

The program continues on the same line whether the expression is true or false. You are limited only by your imagination, although very complicated expressions will be much slower than the simple IF..THEN...

Most of my experience has been in reducing programs used for numerical computations. Many computations involve a number of lengthy FOR ..NEXT loops. Repetition is what takes time, so scrutinize carefully the contents of these loops. When starting to condense a program, begin on the loops; discard operations which may be done satisfactorily outside of the loop. I have had the most success by starting and ending loops on the same line.

I hope that you find your programming bag of tricks heavier now. I have never been able to reduce a significant program to just one line but I'll keep trying. If you succeed or have some additional tricks please drop me a line (of BASIC, that is).

The availibility of an internal timer in the PET allows timing tests to be done conveniently. TI$ is a string of 6 digits representing 24-hour time (102235 – 10 hours, 22 minutes, and 35 seconds) from the time the machine was turned on. The timer may be reset by entering TI$ = "000000". The number TI is also present and measures time in jiffies (1/60 second). TI$ and TI measure the same time, but in different units.

The difference in time taken to execute different operations will usually not be obvious if the operation is performed only once. For this reason I've placed all operations in a FOR..NEXT loop which will exaggerate any differences. By sandwiching the loop between TI$ = "000000" and PRINT TI we may easily make timing tests. Bear in mind that an empty loop also takes time which should be considered when making quantitative comparisons. For reproducible results, perform the tests on a PET immediately after turning it on — don't initialize the DOS WEDGE or the PROGRAMMER'S TOOLKIT as these will affect the execution time.

All results shown for PET BASIC 2.0 (upgrade ROM). Differences which are less than two jiffies may not be significant. ©

```
     TEST 1:
1 GOTO3                             ...
2 RETURN                            ...
3 TI$="000000"                      ...
4 FORJ=0TO999:GOSUB6:NEXT           4 FORJ=0TO999:GOSUB2:NEXT
5 PRINT TI:END                      ...
6 RETURN                            ...
     TI=149                              TI=132

     TEST 2:
1 TI$="000000"                      1 TI$="000000"
2 FORJ=0TO999                       2 FORJ=0TO999:Y=X:X=Y:NEXT
3 Y=X                               3 PRINT TI
4 X=Y
5 NEXT
6 PRINT TI
     TI=233                              TI=223

     TEST 3:
1 TI$="000000"                      10000 TI$="000000"
2 IFTI<3600THENJ=J+1:GOTO2          20000 IFTI<3600THENJ=J+1:GOTO20000
3 PRINT J                           30000 PRINT J
     J=1772                              J=1677
```

The following tests should be placed between
1 TI$="000000
and
5 PRINT TI

```
     TEST 4
3 FORJ=0TO999:AA=BB:NEXT            3 FORJ=0TO999:A=B:NEXT
     TI=149                              TI=143

     TEST 5
2 DIM A(999)                        ...
3 FORJ=0TO999:S=A(J):R=A(J):NEXT    3 FORJ=0TO999:S=A(J):R=S:NEXT
     TI=507                              TI=374

     TEST 6
3 FORJ=0TO999:A=(B*C):NEXT          3 FORJ=0TO999:A=B*C:NEXT
     TI=222                              TI=199

     TEST 7
2 B=2:C=2                           ...
3 FORJ=0TO999:A=1/(B*C):NEXT        3 FORJ=0TO999:A=1/B/C:NEXT
     TI=478                              TI=499

     TEST 8
2 A=2:B=3                           ...
3 FORJ=0TO999:IFA<>BTHENNEXT        3 FORJ=0TO999:IFA-BTHENNEXT
     TI=206                              TI=177
```

TEST 9

```
2 A=2                                 ...
3 FORJ=0TO999:IFA<>0THENNEXT          3 FORJ=0TO999:IFATHENNEXT
    TI=231                                TI=110
```

TEST 10

```
2 P=2:Q=3                             ...
3 FORJ=0TO999:IFQ>=PTHENNEXT          3 FORJ=0TO999:IFP<QTHENNEXT
    TI=206                                TI=203
```

TEST 11

```
3 FORJ=0TO999:NEXTJ                   3 FORJ=0TO999:NEXT
    TI=79                                 TI=63
```

TEST 12

```
3 FORJ=0TO9999:NEXT                   3 FORJ=0TO9999:NEXT:
    TI=684                                TI=677
```

TEST 13

```
2 A=.875                              ...
3 FORJ=0TO999:B=.875:NEXT             3 FORJ=0TO999:B=A:NEXT
    TI=732                                TI=143
```

TEST 14

```
3 FORJ=0TO999:B=.875:NEXT             3 FORJ=0TO999:B=7/8:NEXT
    TI=729                                TI=355
```

TEST 15

```
2 A=π                                 ...
3 FORJ=0TO999:B=A:NEXT                3 FORJ=0TO999:B=π:NEXT
    TI=142                                TI=126
```

TEST 16

```
3 FORJ=0TO999:V=0:NEXT                3 FORJ=0TO999:V=.:NEXT
    TI=167                                TI=131
```

TEST 17

```
2 A=1:B=2:C=3                         ...
3 FORJ=0TO999:IFA<BANDA<CTHENNEXT     3 FORJ=0TO999:IFA<BTHENIFA<CTHEN
                                        NEXT
    TI=468                                TI=341
```

TEST 18

```
2 P=1                                 ...
3 FORJ=0TO999:IFP>-2ANDP<2THENNEXT    3 FORJ=0TO999:IFABS(P)<2THENNEXT
    TI=537                                TI=256
```

TEST 19

```
3 FORJ=0TO999:IFX=0THENCT=CT+1:       3 FORJ=0TO999:CT=CT-(X=0):NEXT
  NEXT
    TI=410                                TI=362
```

BUSINESS and UTILITIES

SPELLGUARD™ (CP/M only) — **Price: $269.95 Disk**

SPELLGUARD is a revolutionary new product which increases the value of your current word processing system (WORD-STAR, MAGIC WAND, ELECTRIC PENCIL, TEXTED EDITOR II and others). Written entirely in assembly language, SPELLGUARD™ rapidly assists the user in eliminating spelling and typographical errors by comparing each word of the text against a dictionary (expandable) of over 20,000 of the most common English words. Words appearing in the text but not found in the dictionary are "flagged" for easy identification and correction. Most administrative staff familiar with word processing equipment will be able to use SPELLGUARD™ in only a few minutes.

MAIL LIST 2.2 (Apple, Atari and North Star diskette only) — **Price: $34.95**

This program is unmatched in its ability to store a maximum number of addresses on one diskette (minimum of 1100 per diskette, more than 2200 for "double density" systems!). Its many features include alphabetic and zip code sorting, label printing, merging of files and a unique keyword seeking routine which retrieves entries by a virtually limitless selection of user defined codes. Mail List 2.2 will even find and delete duplicate entries. A very valuable program!

FORM LETTER SYSTEM (FLS) (Apple and North Star diskette only) — **Price: $21.95**

Use FLS to create and edit form letters and address lists. Form letters are produced by automatically inserting each address into a predetermined portion of your letter. FLS is completely compatible with MAIL LIST 2.2, which may be used to manage your address files.

FLS and MAIL LIST 2.2 are available as a combined package for $49.95.

SORTIT (North Star only) — **Price: $29.95 Diskette**

SORTIT is a general purpose sort program written in 8080 assembly language. This program will sort sequential data files generated by NORTH STAR BASIC. Primary and optional secondary keys may be numeric or one to nine character strings. SORTIT is easily used with files generated by DYNACOMP's MAIL LIST program and is very versatile in its capabilities for all other BASIC data file sorting.

PERSONAL FINANCE SYSTEM (Atari and North Star only) — **Price: $34.95 Diskette**

PFS is a single diskette, menu-oriented system composed of ten different programs. Besides recording your expenses and tax deductible items, PFS will sort and summarize expenses by payee, and display information on expenditures by any of 26 user defined codes by month or by payee. PFS will even produce monthly bar graphs of your expenses by category! This powerful package requires only one disk drive, minimal memory (24K Atari, 32K North Star) and will store up to 600 records per disk (and over 1000 records per disk by making a few simple changes to the programs). You can record checks plus cash expenses so that you can finally see where your money goes and eliminate guesswork and tedious hand calculations.

FAMILY BUDGET (Apple only) — **Price: $34.95 Diskette**

The FAMILY BUDGET is a very convenient financial record-keeping program. You will be able to keep track of cash and credit expenditures as well as income on a daily basis. You can record tax deductible items and charitable donations. The FAMILY BUDGET also provides a continuous record of all credit transactions. You can make daily cash and charge entries to any of 21 different expense accounts as well as to 5 payroll and tax accounts. Data is easily retrieved giving the user complete control over an otherwise complicated (and unorganized!) subject.

THE COMMUNICATOR (Atari only) — **Price: $49.95 Diskette**

This software package contains a menu-driven collection of programs for facilitating efficient two-way communications through a full duplex modem (required for use). In one mode of operation you may connect to a data service (e.g., The SOURCE or MicroNet) and quickly load data such as stock quotations onto your diskette for later viewing. This greatly reduces "connect time" and thus the service charge. You may also record the complete contents of a communications session. Additionally, programs written in BASIC, FORTRAN, etc. may be built off-line using the support text editor and later "uploaded" to another computer, making the Atari a *very* smart terminal. Even Atari BASIC programs may be uploaded. Further, a command file may be built off-line and used later as controlling input for a time-share system. That is, you can set up your sequence of time-share commands and programs, and the Atari will transmit them as needed; batch processing. All this adds up to saving both connect time and your time.

DYNACOMP also supplies THE COMMUNICATOR with an Atari 830 modem for a combined price of $219.95. The modem is available separately for $189.95.

TEXT EDITOR II (CP/M) — **Price: $29.95 Diskette/$33.45 Disk**

This is the second release version of DYNACOMP's popular TEXT EDITOR I and contains many new features. With TEXT EDITOR II you may build text files in chunks and assemble them for later display. Blocks of text may be appended, inserted or deleted. Files may be saved on disk/diskette in right justified/centered format to be later printed by either TEXT EDITOR II or the CP/M ED facility. Futher, ASCII CP/M files (including BASIC and assembly language programs) may be read by the editor and processed. In fact, text files can be built using ED and later formatted using TEXT EDITOR II. All in all, TEXT EDITOR II is an inexpensive, easy to use, but very flexible editing system.

DFILE (North Star only) — **Price: $19.95**

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

FINDIT (North Star only) — **Price: $19.95**

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

GRAFIX (TRS-80 only) — **Price: $14.95 Cassette/$18.95 Diskette**

This unique program allows you to easily create graphics directly from the keyboard. You "draw" your figure using the program's extensive cursor controls. Once the figure is made, it is automatically appended to your BASIC program as a string variable. Draw a "happy face", call it H$ and then print it from your program using PRINT H$! This is a very easy way to create and save graphics.

EDUCATION

HODGE PODGE (Apple only, 48K Applesoft or Integer BASIC) — **Price: $19.95 Cassette/$23.95 Diskette**

Let HODGE PODGE be your child's baby sitter. Pressing any key on your Apple will result in a different and intriguing "happening" related to the letter or number of the chosen key. The program's graphics, color and sound are a delight for children from ages 1½ to 9. HODGE PODGE is a non-intimidating teaching device which brings a new dimension to the use of computers in education.

TEACHER'S PET I (Available for all computers) — **Price: $11.95 Cassette/$15.95 Diskette**

This is the first of DYNACOMP's educational packages. Primarily intended for pre-school to grade 3, TEACHER'S PET provides the young student with counting practice, letter-word recognition and three levels of math skill exercises.

MORSE CODE TRAINER (TRS-80 only) — **Price: $12.95 Cassette/$16.95 Diskette**

MORSE CODE TRAINER is designed to develop and improve your speed and accuracy in deciphering Morse Code. As such, MCT is an ideal software package for FCC test practice. The code sound is obtained through the earphone jack of any standard cassette recorder. You may choose the pitch of the tones as well as the word rate. Also, various modes of operation are available including number, punctuation and alphabet tests, as well as the keying of your own message. A very effective way to learn code!

MISCELLANEOUS

CRYSTALS (Atari only) — **Price: $ 9.95 Cassette/$13.95 Diskette**

A unique algorithm randomly produces fascinating graphics displays accompanied with tones which vary as the patterns are built. No two patterns are the same, and the combined effect of the sound and graphics are mesmerizing. CRYSTALS has been used in local stores to demonstrate the sound and color features of the Atari.

NORTH STAR SOFTWARE EXCHANGE (NSSE) LIBRARY

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

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

AVAILABILITY

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

STATISTICS and ENGINEERING

DIGITAL FILTER (Available for all computers) — **Price: $29.95 Cassette/$33.95 Diskette**

DIGITAL FILTER is a comprehensive data processing program which permits the user to design his own filter function or choose from a menu of filter forms. The filter forms are subsequently converted into non-recursive convolution coefficients which permit rapid data processing. In the explicit design mode the shape of the frequency transfer function is specified by directly entering points along the desired filter curve. In the menu mode, ideal low pass, high pass and bandpass filters may be approximated to varying degrees according to the number of points used in the calculation. These filters may optionally also be smoothed with a Hanning function. In addition, multi-stage Butterworth filters may be selected. Features of DIGITAL FILTER include plotting of the data before and after filtering, as well as display of the chosen filter functions. Also included are convenient data storage, retrieval and editing procedures.

DATA SMOOTHER (Not available for Atari) — **Price: $14.95 Cassette/$18.95 Diskette**

This special data smoothing program may be used to rapidly derive useful information from noisy business and engineering data which are equally spaced. The software features choice in degree and range of fit, as well as smoothed first and second derivative calculation. Also included is automatic plotting of the input data and smoothed results.

FOURIER ANALYZER (Available for all computers) — **Price: $16.95 Cassette/$20.95 Diskette**

Use this program to examine the frequency spectra of limited duration signals. The program features automatic scaling and plotting of the input data and results. Practical applications include the analysis of complicated patterns in such fields as electronics, communications and business.

TFA (Transfer Function Analyzer) — **Price: $19.95 Cassette/$23.95 Diskette**

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

HARMONIC ANALYZER (Available for all computers) — **Price: $24.95 Cassette/$28.95 Diskette**

HARMONIC ANALYZER was designed for the spectrum analysis of repetitive waveforms. Features include data file generation, editing and storage/retrieval as well as data and spectrum plotting. One particularly unique facility is that the input data need not be equally spaced or in order. The original data is sorted and a cubic spline interpolation is used to create the data file required by the FFT algorithm.

FOURIER ANALYZER, TFA and HARMONIC ANALYZER may be purchased together for a combined price of $49.95 (three cassettes) and $59.95 (three diskettes).

REGRESSION I (Available for all computers) — **Price: $19.95 Cassette/$23.95 Diskette**

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

REGRESSION II (PARAFIT) (Available for all computers) — **Price: $19.95 Cassette/$23.95 Diskette**

PARAFIT is designed to handle those cases in which the parameters are imbedded (possibly nonlinearly) in the fitting function. The user simply inserts the functional form, including the parameters (A(1), A(2), etc.) as one or more BASIC statement lines. Data and results may be manipulated and plotted as with REGRESSION I. Use REGRESSION I for polynomial fitting, and PARAFIT for those complicated functions.

MULTILINEAR REGRESSION (MLR) (Available for all computers) — **Price: $24.95 Cassette/$28.95 Diskette**

MLR is a professional software package for analyzing data sets containing two or more linearly independent variables. Besides performing the basic regression calculation, this program also provides easy to use data entry, storage, retrieval and editing functions. In addition, the user may interrogate the solution by supplying values for the independent variables. The number of variables and data size is limited only by the available memory.

REGRESSION I, II and MULTILINEAR REGRESSION may be purchased together for $51.95 (three cassettes) or $63.95 (three diskettes).

ANOVA (Available for all computers) — **Price: $39.95 Cassette/$43.95 Diskette**

In the past the ANOVA (analysis of variance) procedure has been limited to the large mainframe computers. Now DYNACOMP has brought the power of this method to small systems. For those conversant with ANOVA, the DYNACOMP software package includes the 1-way, 2-way and N-way procedures. Also provided are the Yates 2^{K-P} factorial designs. For those unfamiliar with ANOVA, do not worry. The accompanying documentation was written in a tutorial fashion (by a professor in the subject) and serves as an excellent introduction to the subject. Accompanying ANOVA is a support program for building the data base. Included are several convenient features including data editing, deleting and appending.

BASIC SCIENTIFIC SUBROUTINES, Volume 1 (Not available for Atari)

DYNACOMP is the exclusive distributor for the software keyed to the popular text *BASIC Scientific Subroutines, Volume I* by F. Ruckdeschel (see the BYTE/McGraw-Hill advertisement in BYTE magazine, January 1981). These subroutines have been assembled according to chapter. Included with each collection is a menu program which selects and demonstrates each subroutine.

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

All three collections are available for $39.95 (three cassettes) and $49.95 (three diskettes).
Because the text is a vital part of the documentation, *BASIC Scientific Subroutines, Volume I* is available from DYNACOMP for $19.95 plus 75¢ postage and handling.

ROOTS (Available for all computers) — **Price $10.95 Cassette/$14.95 Diskette**

In a nutshell, ROOTS simultaneously determines all the zeroes of a polynomial having real coefficients. There is no limit on the degree of the polynomial, and because the procedure is iterative, the accuracy is generally very good. No initial guesses are required as input, and the calculated roots are substituted back into the polynomial and the residuals displayed.

ACTIVE CIRCUIT ANALYSIS (ACAP) (48K Apple only) — **Price: $25.95/$29.95 Diskette**

ACAP is the analog circuit designer's answer to LOGIC SIMULATOR. With ACAP you may analyze the response of an active or passive component circuit (e.g., a transistor amplifier, band pass filter, etc.). The circuit may be probed at equal steps in frequency, and the resulting complex (i.e., real and imaginary) voltages at each component juncture examined. By plotting the magnitude of these voltages, the frequency response of a filter or amplifier may be completely determined with respect to both amplitude and phase. In addition, ACAP prints a statistical analysis of the range of voltage responses which result from tolerance variations in the components.

ACAP is easy to learn and use. Simply describe the circuit in terms of the elements and their placement, and execute. Circuit descriptions may be saved onto cassette or diskette to be recalled at a later time for execution or editing. ACAP should be part of every circuit designer's program library.

LOGIC SIMULATOR (Apple only; 48K RAM) — **Price: $24.95 Cassette/$28.95 Diskette**

With LOGIC SIMULATOR you may easily test your complicated digital logic design with respect to given set of inputs to determine how well the circuit will operate. The elements which may be simulated include multiple input AND, OR, NOR, EXOR, EXNOR and NAND gates, as well as inverters, J-K and D flip-flops, and one-shots. The response of the system is available every clock cycle. Inputs may be clocked in with varying clock cycle lengths/displacements and delays may be introduced to probe for glitches and race conditions. At the user's option, a timing diagram for any given set of nodes may be plotted using HIRES graphics. Save your breadboarding until the circuit is checked by LOGIC SIMULATOR.

LOGIC DESIGNER (North Star and CP/M only) — **Price: $34.95 Diskette**

LOGIC DESIGNER is an exceptional Computer Aided Design (CAD) program. With it you may convert a large and complicated digitial truth table (the functional specification) into an optimized Boolean logic equation. This equation may then be *easily* converted into a circuit design using either NAND or AND/OR gates. Operationally, LOGIC DESIGNER is composed of a BASIC program which calls in a machine language routine to reduce execution time. Example: For a 7 variable by 127 line table, the processing time is only two minutes. LOGIC DESIGNER is clearly a fast and powerful tool for building digital circuitry.

ORDERING INFORMATION

All orders are processed and shipped within 48 hours. Please enclose payment with order and include the appropriate computer information. If paying by VISA or Master Card, include all numbers on card.

Shipping and Handling Charges
Within North America: Add $1.50
Outside North America: Add 10% (Air Mail)

Delivery
All orders (excluding books) are sent First Class.

Quantity Discounts
Deduct 10% when ordering 3 or more programs. Dealer discount schedules are available upon request.

8" CP/M Disks
Add $2.50 to the listed diskette price for each 8" floppy disk (IBM soft sectored CP/M format). Programs run under Microsoft MBASIC or BASIC-80.

5¼" CP/M Disks
All software available on 8" CP/M disks is also available on 5¼" disks, North Star format.

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

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

VISA

MasterCard

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

Guest Commentary:

Computer Assisted Instruction-Worth The Effort?

Fred Keplinger
Los Gatos, CA

Let me answer that question with an unequivocal yes! I am a French teacher who has been experimenting with computers in the high school classroom for two years now. I am quite excited about the use of computers for private drill work, *individual tutoring* if you will.

Two years ago our administration purchased one 8K PET with the hope that it might lead to experimentation. Having long felt that computers might be of great aid in individualizing instruction, I appropriated the PET, took it home over the summer, and gradually learned to program it. This was my first experience with computers, but I did manage to make some programs for use in French and Spanish.

During the school year 1979-80, we acquired three more PET's — through donations from the community. I used these four machines for review work with my fifth year French students as well as for remedial work with my first year students. I required all beginning students whose progress was not satisfactory to meet with me after school to work with the computer drills. They had to do this until a minimum proficiency was attained. The success of this effort was much greater than I anticipated. For the first time in my twenty-four years of teaching French I had no students with grades of F or D at the end of the first semester.

In the meantime, our Principal and district Superintendent began searching for funds to build a computer laboratory. Generous local citizens and service clubs donated enough money to provide a laboratory equipped with fifteen PET's, two printers, and two disk drives. The school has used general fund money to pay for only one machine.

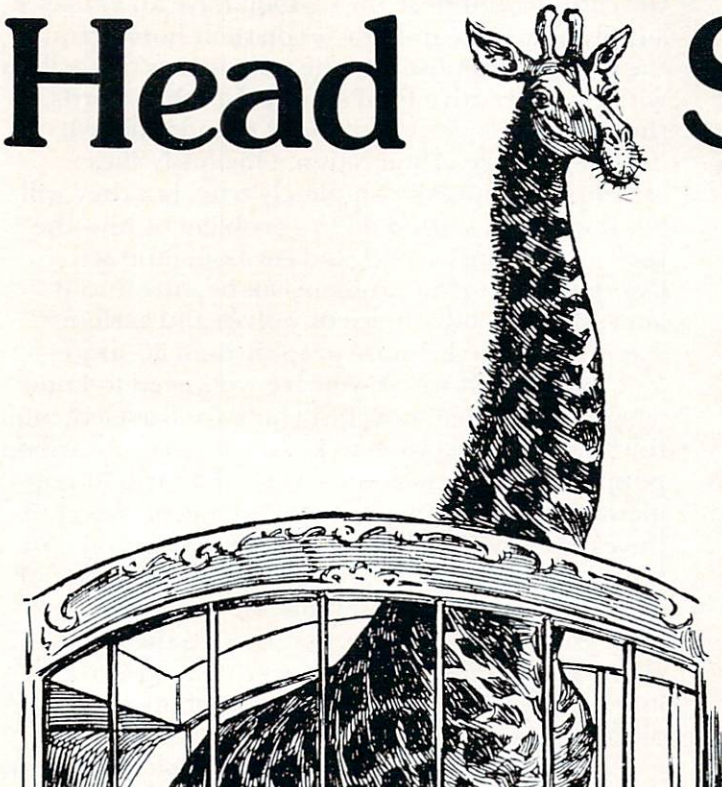
It is obvious that our program is off to a good start because of generous, enthusiastic local support. But what else is necessary? Let me make some observations. Speaking as one who has dealt with commercial software for the past twenty-two years (I have long been involved with oral-aural language labs and the writing of materials for them), I feel strongly that no adequate commercial software will ever be developed for school use — the students are just too different to allow for a mass market approach. Therefore a good CAI program will exist only when there are teachers throughout the school who are willing to learn enough programming to make their own drills. This is not difficult to accomplish.

Last summer I presented a crash course in programming in BASIC to our Foreign Language Department. The result is that we now have computers that speak French, Spanish, Italian, and Latin. I also offered an after school course to a number of teachers from several departments, as well as to some administrators and members of the clerical staff. We are well on the way to developing our own library of appropriate software.

If CAI is to succeed, another factor is most necessary. Students must have access to the computers. Obvious? Yes, but too often overlooked. We chose the PET computer because it is an integrated unit that is easily mounted on a small cart. It is quite easy to wheel our machines to wherever they are needed, whenever they are needed. I have also trained P.T.A. aides to operate the computers in the lab so that teachers may send students there for supervised drill whenever it is needed.

CAI is not a fad, an instructional gimmick. It is a technique that helps a student study and develop mastery of any material that requires repetition. I find it to be a very exciting technique. ©

Head and Shoulders



Above the Crowd!

PET®

educational software from MICRO-ED

Send for our free catalog*

*please specify PET

MICRO-ED, Inc. • P.O. Box 24156 • Minneapolis, MN 55424

or telephone us at (612) 926-2292

PET is the registered trademark for Commodore Business Machines.

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

Wolves, Caribou, And Other Problems

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

For a moment let us postpone the problem of wolves and caribou, which is only a specific example of a large class of problems that may be studied with mathematics and a computer. A *mathematical model* attempts to describe a complex social, biological, or physical system using mathematical techniques, with the hope that new insights into the behavior of the system will result. Attempts are made to project the behavior of the system into the future. Perhaps the most popular example of a mathematical model is the now old-fashioned lunar lander game. A good lunar lander game will simulate the *actual* conditions involved in a landing, using Newton's Universal Law of Gravitation and Newton's second law of motion to provide the necessary differential equations. These equations are solved by the computer while the operator controls such parameters as the "burn rate," or thrust. Fortunately, the crash that most of us experience is not actual.

Constructing a mathematical model of a system is not only entertaining and challenging — it may be genuinely useful. Some very useful models have been constructed that project population growth, availability of natural resources, and the growing economic distance between various groups of people on this planet. One of the most pressing needs is a model of the atmosphere of our planet that will allow long-range weather forecasting. In fact, such a model may be crucial if agriculture is going to supply enough food for the world. The vast amount of data needed for this project will require the use of many large and fast computers.

The goal of this article is more modest. It will describe a simple biological system involving predators, the wolves, and prey, the caribou. The credit for the original work on this subject belongs to Alfred Lotka and Vito Volterra whose names have become attached to the Lotka-Volterra Predator-Prey Equations. Lotka's concern was with a herbivorous animal population that preyed upon a plant population. Obviously, their ideas can be extended to a large number of complex biological and social problems because the highest form of life on this planet is a predator.

To begin, we will simplify the problem by making some assumptions about the wolves and the caribou. Suppose the caribou have an *unlimited* supply of mosses and grasses on their home ground, the tundra. Suppose that the wolves prey on caribou with no alternative food supply. In other words, if the caribou are scarce, the wolf population will decline because of starvation. Obviously these assumptions are not completely true, but they will allow us to get started on the problem of how the two populations, wolves and caribou, interact. Complexities in the problem can be introduced later. Besides, our choice of wolves and caribou was intended to be more graphic than accurate.

To find out where you are, you need to know where you started, how fast you have traveled, and for how long. Likewise, to know the current caribou population, you must know what the caribou population was at a previous time, the *rate of change* of the caribou population, and the time interval. For the moment, we will be concerned with the rate of change of the caribou population with respect to time. Just as speed (miles per hour) is the rate of change of distance with respect to time, we are interested in the rate of change (caribou per day) of the caribou population.

Let x be the number of caribou that are alive at any time t. The rate of change of the number of caribou with time is symbolized by:

$$\frac{dx}{dt} = \text{change in the number of caribou per unit time} \quad (1)$$

Do not let the strange looking symbol disturb you. It is no different than any other rate, such as miles per hour or gallons per minute. If dx/dt is positive, the number of caribou is increasing, while, if dx/dt is negative, the number of caribou is decreasing.

What determines dx/dt? Any population (people, rabbits, bacteria) with an unlimited food supply increases at a rate that is proportional to itself. The more rabbits you have, the more baby rabbits you get. Thus, one term in the equation for dx/dt is Ax. That is,

$$\frac{dx}{dt} = Ax \quad (2)$$

where A is some number that depends on many biological factors. These factors include the number of calves a caribou mother bears, at what age she starts bearing caribou babies, the natural death rate of caribou in the absence of predators, and many others.

This brings us to the effect of the wolves on the caribou. Let y be the number of wolves that are alive at any time. The quantity xy is the product of the number of caribou and the number of wolves. If we were to identify each of the caribou and each of the wolves, the product xy would give us the number of possible caribou-wolf encounters. Clearly, the negative component of dx/dt is related

to the number of caribou-wolf encounters. However, not every encounter leads to a caribou death; the wolf may not be hungry. Thus, the negative component of dx/dt is -Bxy where B is a number that represents the fraction of caribou-wolf encounters in a given period of time that lead to a caribou death. Thus, the final expression for the rate of change of caribou per unit time is:

$$\frac{dx}{dt} = Ax - Bxy \qquad (3)$$

Turning next to the wolves, we must find their rate of change with respect to time; that is, dy/dt, the change in the number of wolves per unit time. The "xy" term in Equation 3 is "bad" for caribou, but good for wolves because it represents more caribou eaten. In general, we can say that what is bad for caribou is good for wolves, and write:

$$\frac{dy}{dt} = Cxy - Dy \qquad (4)$$

where C and D are numbers that depend on a large number of biological factors. Equation 4 shows that, for predators with a limited food supply, an excess of predators (the "Dy" term in Equation 4) can have a detrimental effect on the number of predators.

Before continuing, let me point out the more or less obvious fact that I am not a biologist and I do not pretend to be able to determine the constants A, B, C, and D from biological data. Also, the purpose of this paper is not to reflect in any negative way on the wolf. As a predator he is probably far less harmful than human beings.

Meanwhile, back on the tundra, we would like to find how the number of caribou, *x*, and the number of wolves, *y*, vary with time. This problem will be solved in the same way you would calculate a simple rate problem, such as driving a car. If you are presently 50 miles from home, and you travel for an additional three hours at 55 miles per hour, then you will be 215 miles from home. 215 = 50 + 55*3.

In the same way, the number of caribou at time t is equal to the number of caribou at a time t_0 (earlier) plus the rate of change of caribou times the time elapsed between t and t_0. In equation form:

$$x(t) = x(t_o) + \frac{dx}{dt}(t - t_o) \qquad (5)$$

Note the similarity between this statement, Equation (5), and the example in the preceding paragraph. A similar equation may be written for the wolves:

$$y(t) = y(t_o) + \frac{dy}{dt}(t - t_o) \qquad (6)$$

Our calculation proceeds in very small steps because both dx/dt and dy/dt change with x and y. We begin with some initial population, say x_o and y_o. We calculate dx/dt and dy/dt at these two values of x_o and y_o. Next, multiply by the small time interval between t and t_o, and add the results to x_o and y_o as in Equations 5 and 6. Repeating the process over and over again produces a table of values for *x* and *y* at various times *t*, subsequent to our starting time.

Before giving a BASIC program to predict the number of caribou and the number of wolves, we note that a stable condition does exist in which the number of wolves and the number of caribou are constant. If x_o and y_o are the beginning values for the number of caribou and the number of wolves respectively, and if:

$$x_o = \frac{D}{C}, \qquad y_o = \frac{A}{B} \qquad (7)$$

then both dx/dt and dy/dt are zero, and there will be no change in either population.

For purposes of illustration, we chose A = 2, B = .01, C = .01, and D = 8. With these choices for the constants, a choice of x_o = 800 caribou and y_o = 200 wolves gives a stable population for both species. These constants and initial animal populations do not have their foundation in biological reality; they merely serve to illustrate situations and trends that can occur in predator-prey relationships. Perhaps a much smaller population of wolves can *actually* control a much larger population of caribou, but for our purposes it is nice to keep the numbers of each population in a range so they can both be plotted on the same graph. Nature is rarely that accomodating. The program in Listing 1 can be used to see how the number of caribou and the number of wolves change with time, depending on the initial values chosen for x_o and y_o.

Before describing some of the results, let us examine the details of the program. The first four statements should be obvious. Statement 50 simply prints the current value of the time, the caribou population *x*, and the wolf population *y*. Note that the print statement "dresses up" the output values of *t* and rounds *x* and *y* to the nearest whole number. The data are much simpler to read when placed in this form. Statements 60 and 70 calculate the rates of change of the caribou and wolf populations. They correspond to Equations 3 and 4. Statements 80 and 90 are BASIC equivalents of Equations 5 and 6. Note that our time interval is 0.005 (the units are arbitrary; if you wish, you may think of the units as decades). Statement 100 updates the time each time around the loop. Since there are numerous calculations, we chose not to print all of the results. Statement 110 selects values of t = 0, .1, .2, ..., 1.0, 1.1, ... for printing the current population data, skipping intermediate values of *t*.

Listing 1.
A BASIC program to solve the Lotka-Volterra predator-prey equations.

```
10 PRINT "INPUT THE STARTING POPULATIONS
   OF CARIBOU AND WOLVES."
20 INPUT X,Y
30 PRINT "INPUT THE CONSTANTS A, B, C, AND D
   FOR THIS PROBLEM."
```

```
 40 INPUT A, B, C, D
 50 PRINT INT(10*T+.001)/10; INT(X+0.5); INT(Y+0.5)
 60 XP=A*X-B*X*Y
 70 YP=C*X*Y-P*Y
 80 X=X+XP*0.005
 90 Y=Y+YP*0.005
100 T=T+0.005
110 IF ABS(10*T-INT(10*(T+.00001)))<0.00001 THEN 50
120 GO TO 60
140 END
```

Table 1. A sample run of the program in Listing 1.

```
INPUT THE STARTING POPULATIONS OF CARIBOU
AND WOLVES.
? 1000,  200
INPUT THE CONSTANTS A, B, C, AND D FOR THIS
PROBLEM
?2,     .01  .01,  8
0      1000  200
.1      980  243
.2      921  283
.3      836  307
.4      750  305
.5      682  280
.6      640  244
.7      624  205
.8      631  172
.9      657  147
1       698  130
1.1     753  120
1.2     817  118
1.3     885  124
```

Table 1 shows how the computer prints the results on my AIM 65. The results are best studied by graphing the number of caribou and the number of wolves as a function of time. However, even a brief study of the results in Table 1 can lead to some conclusions. Recall that 800 caribou and 200 wolves give a stable (nature is balanced) population for both caribou and wolves. Try these values in the program. Note that in Table 1 we began with 1000 caribou and 200 wolves, representing an excess of caribou. Perhaps it was a better than average calving year for caribou. In any case, the caribou population begins to decline, while the wolf population increases. The wolves also "benefit" from the excess of caribou. However, at about $t=.3$, the caribou are near their stable level but now there is an excess of wolves. Thus, the caribou population declines below the stable level and the wolf population also declines because there are too few caribou to sustain their numbers. When the wolves have declined below about 200 wolves at $t=.7$, the caribou population begins to climb because there are too few caribou to sustain their numbers. When the wolves have declined below about 200 wolves at $t=.7$, the caribou population begins to climb because there are so few wolves. The decline in the wolf population ends at $t=1.2$ when the caribou population exceeds 800 again. The entire cycle repeats itself over and over again. A graph of these results is shown in Figure 1. Note that the peaks in the wolf population *follow* the peaks in the caribou population. The dips in the wolf population also come at later times than the dips in the caribou population. A slight increase in the peaks and a slight decrease in the dips is observed as time goes on. This is a result of the crudeness of our technique. An actual solution will repeat itself. At least we obtain some idea of what happens when the stable population is disturbed.

Figure 1 suggests some things to try for yourself. What happens when there are too few caribou (less than 800) and too many wolves (more than 200)? What happens when there is an excess of each species? What happens if there are no wolves? No caribou? Suppose that caribou has suddenly become very popular in restaurants, and that the caribou herd is reduced to 80 before we realize that the caribou are almost extinct, and we suddenly quit hunting them (leaving 80 caribou and 200 wolves). What will happen in this case? Try to predict what happens before you run the program.

Equations 3, 4, 5, and 6; the computer program; and the graph of the results make up our mathematical model of the caribou and the wolves. It can be modified by adding additional complexities. What if the supply of grasses on the tundra is limited and there is intraspecific competition? What happens if the caribou sometimes kill wolves?

I would like to acknowledge the inspiration I received from my Numerical Analysis class (Summer 1980). I would also like to acknowledge David A. Smith's book, **INTERFACE: Calculus and the Computer**, (Houghton-Mifflin, Boston, 1976) where I first encountered the predator-prey problem. This book is an excellent source for a large variety of calculus-related problems that can be solved with a computer. Anyone who has had one or more courses in calculus should profit from this book.

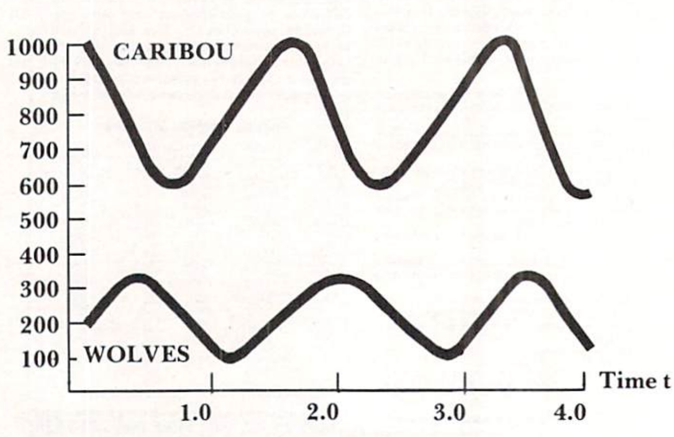


Figure 1. The populations of caribou and wolves as a function of time. Initial populations were $X_o=1000$ caribou and $y_o=200$ wolves.

©

Add A Programmable Sound Generator

Connecting The General Instruments AY-3-8910 Programmable Sound Generator To The 6502/6800 Bus

Michael Stevens
Buntingford, England

The AY-3-8910 sound generator is a particularly versatile device capable of generating three simultaneous tones, each of which can be separately controlled in amplitude and/or mixed with noise to produce a wide range of sound effects. The particular merit of the GI chip, compared to other sound generators, is that its operation is entirely digitally controlled, making it suitable for use with a microprocessor.

In addition to its sound generator functions, the AY-3-8910 also features two 8-bit wide general purpose I/O ports (labelled IOA and IOB in the pin diagram of Fig. 1). All functions are controlled by sixteen internal registers accessed by a combined data and address 8 bit port (DA0-7 in Fig. 1). The AY-3-8910 is designed principally for use with GI's PIC1600 and 1650 series of microprocessor with bus control pins BC1, BC2 and BDIR determining whether the DA0-7 lines are to be interpreted as address or data lines.

The combined function of the DA0-7 lines do not allow for easy interfacing to other microprocessors such as the 6502 and 6800 series. One method of interfacing that has been proposed uses a 6820 programmable interface adaptor (PIA) with 8 lines of port A connected to the DA0-7 pins of the sound generator and three of the port B lines for the three bus control pins.

This means of interfacing makes programming the sound chip cumbersome. One needs to simulate the bus waveforms shown in Fig 2. Assuming that one is writing in BASIC, then two POKE commands are needed to set up the 6820 ports as outputs. Then one needs a POKE to send the address of the required internal register to the DA0-7 pins, another POKE to send LATCH ADDRESS to the bus control pins, a third POKE to send BUS INACTIVE, followed by a fourth POKE to send the data to the DA0-7 pins, a fifth POKE to send WRITE DATA to the bus control pins, and a sixth POKE to return the bus control pins to BUS INACTIVE. These last six POKES must be repeated for each of the sixteen internal registers needing input.

Why can one not make the sixteen registers in the sound generator part of the addressable memory of the microprocessor? Then a single POKE to the relevant address would be all that is needed.

The reason that this is not straightforward is that the AY-3-8910 is too slow to respond to the 1uS processor cycle of the 6502/6800 families. Following a falling edge of the 02 clock the minimum time intervals needed are:

delay until the processor address is valid	**300 nS**
AY-3-8910 address set up time	**400 nS**
AY-3-8910 address hold time	**100 nS**
AY-3-8910 data set up time	**50 nS**
AY-3-8910 data pulse width	**500 nS**
AY-3-8910 data hold time	**100 nS**
	1.45uS

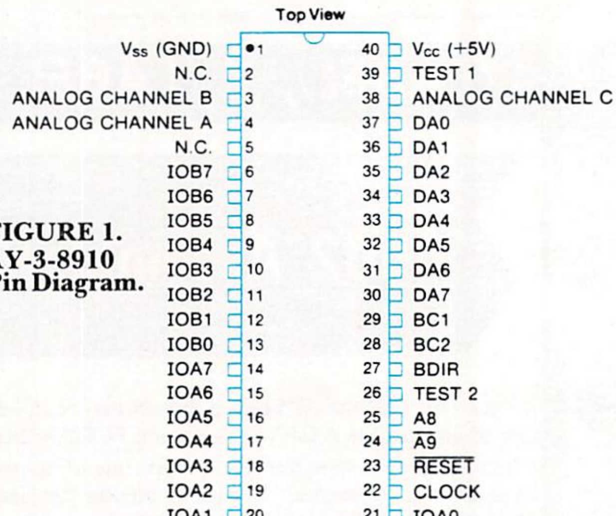


FIGURE 1. AY-3-8910 Pin Diagram.

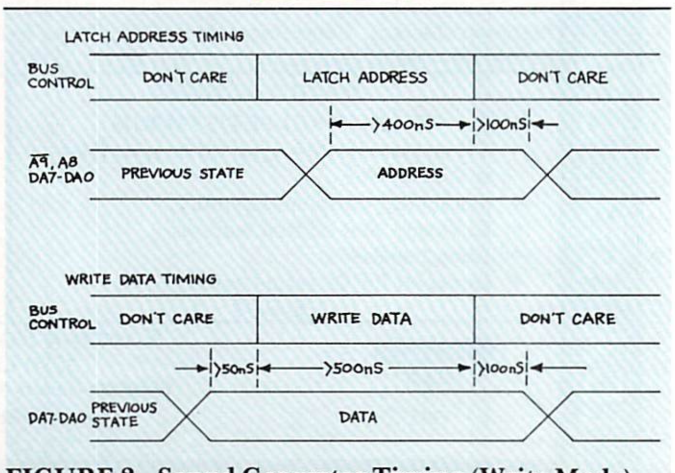


FIGURE 2. Sound Generator Timing (Write Mode).

The solution is to deliberately spread the writing to the sound chip over two processor cycles. The circuit in Fig. 3 does this. The relevant signal waveforms are shown in Fig. 4. Circuits IC1 and IC2 decode the top eight address lines A15-8 and their outputs feed the additional address select pins A8 and A9 on the sound generator IC8. The quiescent state of IC5 leaves a one on its terminal Q and a zero on Q, the latter enabling the tristate output of IC3 and thereby applying the lower address lines A7-A0 to the combined data/address pins of the sound generator. The first half of the dual monostable IC7 triggers off each falling edge of 02 and produces a 300 nS pulse. The back edge of this pulse triggers the second half generating a delayed 550nS pulse. This latter pulse is applied to pin BDIR of the sound generator, which together with the output Q of IC5 on BC1 (also high during this period) codes LATCH ADDRESS on the bus control pins. The LATCH ADDRESS condition terminates 150nS before the computer address lines A15-A0 change. Only the lowest four address lines A3-A0, of the eight A7-A0 lines, address a register in the sound generator, if the other four lines A7-A4 are not zero, or if A8 and A9 are not 10 respectively then the address is invalid and the sound generator takes no further action.

On the next falling edge of 02 (rising edge of 02), two things happen. The data on the data lines D7-D0 from the microprocessor are latched into IC4, and the low, now on pin D of IC5, is clocked through to the Q terminal. This latter signal enables the tristate output of IC4 at the same time

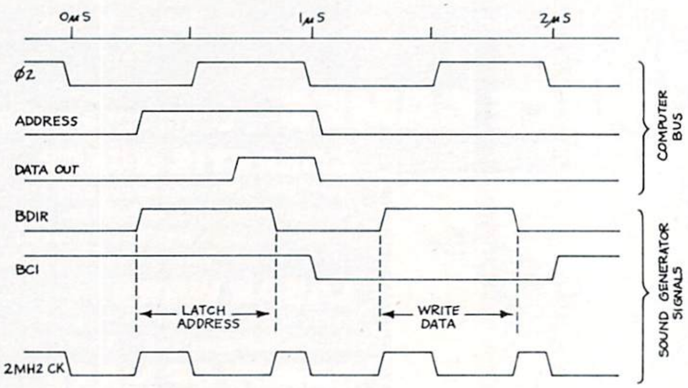
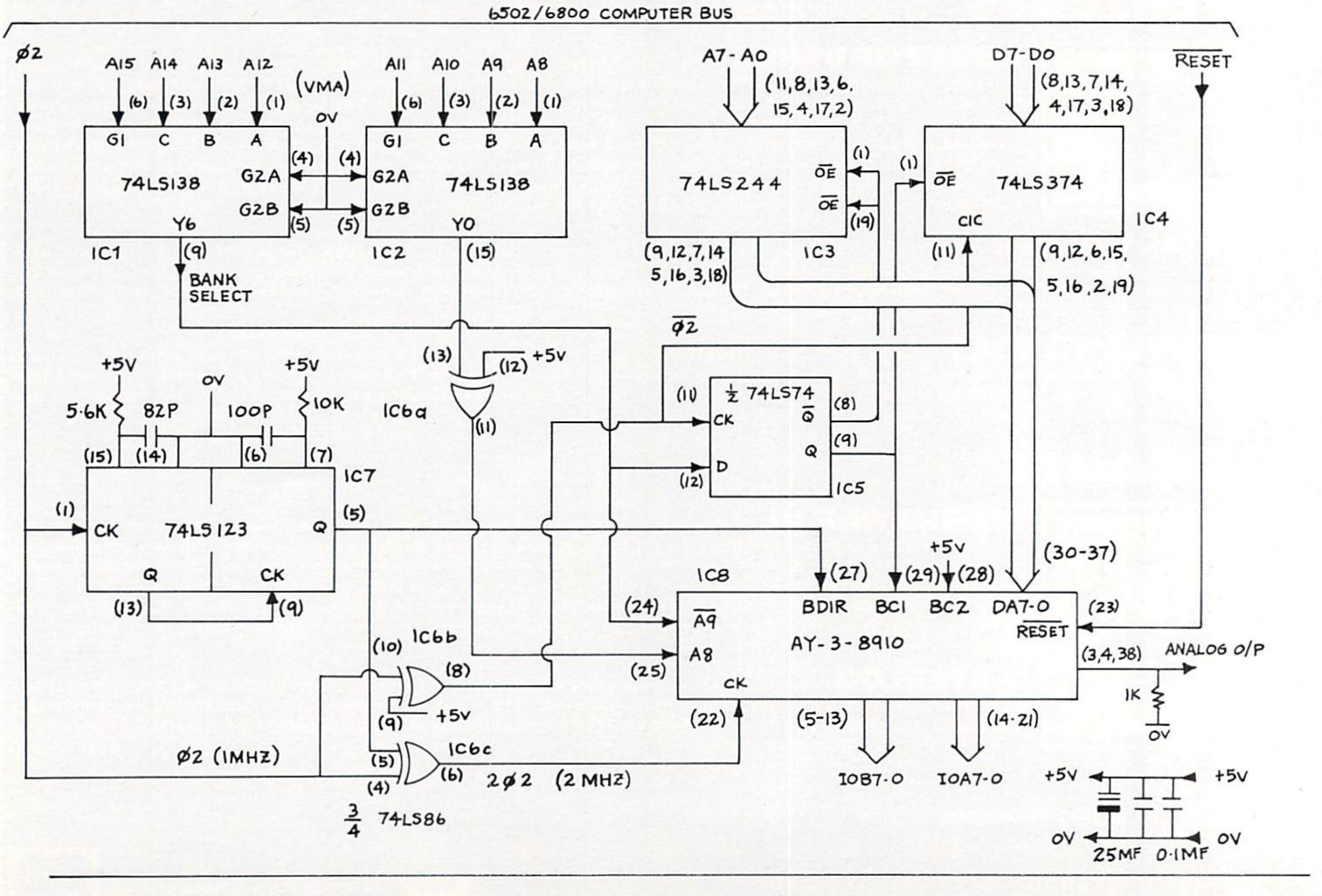


FIGURE 4. Signal Timing.

FIGURE 3. AY-3-8910. Memory Mapped On The 6502/6800 Bus.



as the output of IC3 is disabled. Data, not address, is now applied to the combined data/address lines of the sound generator IC8. The monostables continue to produce delayed 550nS pulses and, when BDIR is high again, BC1 is now low giving a READ DATA command to the sound generator. Circuit IC6c is an exclusive OR which inverts 02 whenever the monostable output is high. The result is to convert the 1 MHz 02 into a 2MHz clock which is fed to the clock terminal of the sound generator. This 2MHz clock enables the generation of more precise tones at the higher frequencies than a 1MHz clock would allow. (See Table 1.) Circuits IC6a and IC6b are simple inverters and could be replaced by a 74LS04 if the frequency doubling function of IC6a is not required.

I have constructed the circuit for use with a Commodore PET. For this purpose I do not really need IC1, since it duplicates the Bank Select signals which are available on this expansion bus. I have chosen to place the sound generator at memory addresses E800 to E80F, which is in the I/0 area of the PET. Other addresses are possible by choosing different outputs of IC1 and IC2, and possibly the use of the G2 chip select pins. If the circuit is used with the 6800 processor, then VMA should be connected to one of the G2 select pins.

The circuit achieves the objective of permitting POKES directly to the sound generator registers and is only slightly more complex than the use of a 6820 PIA, if one includes the chip select decoding that is also necessary with the PIA. One function that my circuit does not allow is the reading of the register data. To add this feature is not difficult, although it will require a double PEEK: one to strobe in the address, and the second to read the data. There seems little purpose in having a read function. I have yet to use the two output data ports IOA and IOB, but plan them for two D to A converters to provide, additionally, two directly synthesized tone channels.

TABLE 1. Coarse (HI) And Fine (LO) Tuning Register Values Using A 2MHZ Clock.

NOTE		--- FREQUENCY (HZ)--- IDEAL	ACTUAL	ERROR	POKE HI	LO
C	1	32.70	32.71	.0%	14	238
C#	1	34.65	34.65	.0%	14	24
D	1	36.71	36.71	.0%	13	77
D#	1	38.89	38.89	.0%	12	142
E	1	41.20	41.20	.0%	11	218
F	1	43.65	43.66	.0%	11	47
F#	1	46.25	46.24	.0%	10	143
G	1	49.00	49.00	.0%	9	247
G#	1	51.91	51.91	.0%	9	104
A	1	55.00	54.99	.0%	8	225
A#	1	58.27	58.28	.0%	8	97
B	1	61.74	61.73	.0%	7	233
C	2	65.41	65.41	.0%	7	119
C#	2	69.30	69.29	.0%	7	12
D	2	73.42	73.40	.0%	6	167
D#	2	77.78	77.78	.0%	6	71
E	2	82.41	82.40	.0%	5	237
F	2	87.31	87.29	.0%	5	152
F#	2	92.50	92.52	.0%	5	71
G	2	98.00	97.96	.0%	4	252
G#	2	103.83	103.82	.0%	4	180
A	2	110.00	110.04	.0%	4	112
A#	2	116.54	116.50	.0%	4	49
B	2	123.47	123.52	.0%	3	244
C	3	130.81	130.75	.0%	3	188
C#	3	138.59	138.58	.0%	3	134
D	3	146.83	146.89	.0%	3	83
D#	3	155.56	155.47	-.1%	3	36
E	3	164.81	164.91	.1%	2	246
F	3	174.61	174.58	.0%	2	204
F#	3	185.00	184.91	.0%	2	164
G	3	196.00	195.92	.0%	2	126
G#	3	207.65	207.64	.0%	2	90
A	3	220.00	220.07	.0%	2	56
A#	3	233.08	233.21	.1%	2	24
B	3	246.94	247.04	.0%	1	250
C	4	261.63	261.51	.0%	1	222
C#	4	277.18	277.16	.0%	1	195
D	4	293.66	293.43	-.1%	1	170
D#	4	311.13	310.95	-.1%	1	146
E	4	329.63	329.82	.1%	1	123
F	4	349.23	349.16	.0%	1	102
F#	4	369.99	369.82	.0%	1	82
G	4	392.00	391.85	.0%	1	63
G#	4	415.30	415.28	.0%	1	45
A	4	440.00	440.14	.0%	1	28
A#	4	466.16	466.42	.1%	1	12
B	4	493.88	494.07	.0%	0	253
C	5	523.25	523.01	.0%	0	239
C#	5	554.37	555.56	.2%	0	225
D	5	587.33	586.85	-.1%	0	213
D#	5	622.25	621.89	-.1%	0	201
E	5	659.26	657.89	-.2%	0	190
F	5	698.46	698.32	.0%	0	179
F#	5	739.99	739.64	.0%	0	169
G	5	783.99	786.16	.3%	0	159
G#	5	830.61	833.33	.3%	0	150
A	5	880.00	880.28	.0%	0	142
A#	5	932.33	932.84	.1%	0	134
B	5	987.77	984.25	-.4%	0	127
C	6	1046.50	1050.42	.4%	0	119
C#	6	1108.73	1106.19	-.2%	0	113
D	6	1174.66	1179.25	.4%	0	106
D#	6	1244.51	1250.00	.4%	0	100
E	6	1318.51	1315.79	-.2%	0	95
F	6	1396.91	1404.49	.5%	0	89
F#	6	1479.98	1488.10	.5%	0	84
G	6	1567.98	1562.50	-.3%	0	80
G#	6	1661.22	1666.67	.3%	0	75
A	6	1760.00	1760.56	.0%	0	71
A#	6	1864.66	1865.67	.1%	0	67
B	6	1975.53	1984.13	.4%	0	63
C	7	2093.00	2083.33	-.5%	0	60
C#	7	2217.46	2232.14	.7%	0	56
D	7	2349.32	2358.49	.4%	0	53
D#	7	2489.02	2500.00	.4%	0	50
E	7	2637.02	2659.57	.9%	0	47

F	7	2793.83	2777.78	-.6%	0	45
F#	7	2959.96	2976.19	.5%	0	42
G	7	3135.96	3125.00	-.3%	0	40
G#	7	3322.44	3289.47	-1.0%	0	38
A	7	3520.00	3472.22	-1.4%	0	36
A#	7	3729.31	3676.47	-1.4%	0	34
B	7	3951.07	3906.25	-1.1%	0	32
C	8	4186.01	4166.67	-.5%	0	30
C#	8	4434.92	4464.29	.7%	0	28
D	8	4698.64	4629.63	-1.5%	0	27
D#	8	4978.03	5000.00	.4%	0	25
E	8	5274.04	5208.33	-1.2%	0	24
F	8	5587.65	5681.82	1.7%	0	22
F#	8	5919.91	5952.38	.5%	0	21
G	8	6271.93	6250.00	-.3%	0	20
G#	8	6644.88	6578.95	-1.0%	0	19
A	8	7040.00	6944.44	-1.4%	0	18
A#	8	7458.62	7352.94	-1.4%	0	17
B	8	7902.13	7812.50	-1.1%	0	16

©

The Carry Bit-What It Is And How It Works

Eric Brandon
Ontario, Canada

When writing a machine language program, you often make what are called conditional branches. The 6502 makes conditional branches depending on the status of 7 status bits. How to use these bits, what changes their status, exactly what they can tell you: these are the questions that often plague the neophyte machine language programmer. Of the seven, the zero bit and the carry bit are the most used. Since the zero bit is extremely straightforward (when the result of an operation is zero it is set; otherwise it is clear), this article will deal with the carry bit.

The operations that affect the carry bit are:

Operation	Effect On Carry Bit
ADC	Sets carry if result of addition is more than 255 ($FF). Otherwise carry is cleared.
ASL	Bit 7 (MSB) goes into carry bit.
CLC	Clears Carry.
CMP	If byte compared is less than or equal to the contents of the accumulator, carry is set. Otherwise carry is clear.
CPX	Same as CMP, but byte is compared to X register.
CPY	Same as CMP, but byte is compared to Y register.
LSR	Bit 0 (LSB) goes into carry bit.
PLP	Bit 0 of the first byte on the stack goes into carry.
ROL	Same effect as ASL.
ROR	Same effect as LSR.
RTI	Carry restored to what it was before interrupt.
SBC	If the subtrahend (number you are subtracting) is greater than the minuend (number you are subtracting from) the carry bit will be cleared.
SEC	Sets carry.

The tests you can use are:

Test	Meaning
BCC	Branch if carry is clear.
BCS	Branch if carry is set.

The carry bit is most often used with ADC, SBC, and CMP. Because of this I will explain in depth how to use it with these instructions.

ADC means Add With Carry. It operates thus:
accumulator = accumulator + addend + carry

Before adding, the carry bit should always be cleared with the CLC instruction. For example, to add 40 to memory location VALUE, the sequence to use is:

```
LDA     VALUE
CLC
ADC     #40
STA     VALUE
```

When adding numbers of more than 8 bits in length, more than one ADC instruction must be used. Carry should only be cleared before adding the least significant byte. The other bytes will "take care of themselves." For example, to add the number in MSB1 and LSB1 to the number in MSB2 and LSB2 and store the result in VALUE, VALUE + 1, and VALUE + 2 the sequence to be used is:

```
LDA     LSB1
CLC
ADC     LSB2
STA     VALUE+2
LDA     MSB1
ADC     MSB2
STA     VALUE+1
LDA     #0
ADC     #0
STA     VALUE
```

Similar methods can be used for adding numbers of any length. Note that when the decimal mode flag is set, the carry bit will be set when the addition results in a number greater than 99 decimal.

Another frequent carry-related operation is SBC. To understand how carry works with this instruction, we must understand the relationship between the carry bit and an imaginary bit called "borrow." Most books on 6502 machine language programming will tell you that carry is an inverted borrow. What this means is that whenever this imaginary borrow bit would be set, carry is cleared. And, whenever borrow would be cleared, carry is set. SBC works thus:

accumulator = accumulator-subtrahend-borrow

Obviously, borrow must be cleared before subtracting. This is done by *setting* carry. For example, to subract 40 from memory location VALUE, this sequence could be used:

```
LDA     VALUE
SEC
SBC     #40
ST      VALUE
```

As with addition, when adding numbers longer than 8 bits in length, the least significant bits are subtracted first. Carry is set only before the first subtraction.

CMP is an instruction that subtracts the operand from the accumulator without changing either the contents of the accumulator or the operand. So what is it? When it does the subtraction, it adjusts the status bits as if a subtraction had been performed by an SBC.

There is a confusing aspect to interpreting the status bits after a CMP instruction. If the operand is less than or equal to the contents of the accumulator, the carry bit will be set. If the operand is greater than the contents of the accumulator, CMP will clear carry. Armed with this knowledge (and the fact that the zero bit will be set if the bytes were equal) you can test for any relationship between them. In other words:

	Carry	Zero
Operand is greater than accumulator	clear	clear
Operand is equal to accumulator	set	set
Operand is less than accumulator	set	clear

The carry bit can provide you with a wealth of information. Use it to its full advantage and you will find that your machine language programs will become shorter and easier to understand.

6502 Assembly Language Programming, by Lance A. Leventhal, clearly shows what effect each instruction has on the status bits. If you have any questions or problems, read this fine book, or write to me at:

Eric Brandon
36 Hartfield Road
Islington, Ontario
Canada
M9A 3C9

A Floating-Point Division Routine

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

I. Introduction

In three previous articles in **COMPUTE!** we described:

1) a program that converts a decimal number (with a sign and an exponent) to a floating-point binary number (**COMPUTE!** #9)
2) a program that converts a floating-point binary number to a decimal number (**COMPUTE!** #11)
3) a program that multiplies two signed binary floating-point numbers (**COMPUTE!** #12).

In this article we describe a program that divides two floating-point binary numbers. Most of the programming described in this series has been relocatable allowing the user to move the programs or to put them in EPROMs with relative ease. Furthermore, the routines that were used to input and output the numbers can usually be found in a monitor, so that most of the code should be easily adapted to anyone's machine.

II. The Division Routine

Just as the multiplication routine does, the division routine uses three accumulators. The contents of accumulator A (ACCA) is divided *into* the contents of accumulator B (ACCB), and the quotient is stored temporarily in the result accumulator (RES) before the answer is moved back to the accumulator used by the output (floating-point binary to BCD routine) program.

Accumulator A occupies locations with addresses $0000 through $0003 with the most-significant byte in location $0000. The mantissa of the divisor is located in accumulator A. Location $0004 is used as a guard byte, permitting a 34-bit division before rounding the final answer to 32 bits. Thirty-two bits gives an answer that is accurate to approximately nine decimal digits. Accumulator B occupies locations with addresses $0020 through $0023 with a guard byte at location $0024. Accumulator B contains the dividend mantissa. The exponent and sign locations are the same as for the multiplication routine described earlier. The quotient is moved into RES at locations $0010 to $0014 as it is being calculated. When the calculation is finished, the quotient is moved to the accumulator that is used by the floating-point binary to BCD routine to output the answer. The accumulator architecture is exactly the same as for the multiplication routine described in the previous article.

The division algorithm is almost identical to the one you used in elementary school to do long division. Try one of these problems in decimal and then in binary if you want to understand the algorithm. Basically, it proceeds as follows:

1. Set COUNT = 34 = $22 to do a 34 bit division.
2. Calculate DIVIDEN – DIVISOR. If the carry flag is set then the DIVIDEND is greater than the DIVISOR, go to (3). Otherwise go to (4).
3. Replace the DIVIDEND with DIVIDEND – DIVISOR.
4. Shift the CARRY left into the LSB of the QUOTIENT.
5. Shift the new DIVIDEND left. (This is analogous to "bringing down" the next digit.)
6. Decrement COUNT. If COUNT is not zero, go to (2), otherwise go to (7).
7. Normalize and round the quotient.

As in the case of multiplication, the sign of the result is found by forming an exclusive-or with the signs of the divisor and the dividend. Recall from algebra that the exponent of the quotient is found by subtracting the exponent of the divisor from that of the dividend. If the exponent exceeds 127 or is less than -128, the program executes a BRK instruction. It is left to your imagination what you want your BRK routine to do for underflow or overflow. In my case the program simply jumps to the monitor. If the divisor is zero, the program also executes a BRK instruction. If the dividend is zero, the entire division routine is bypassed and the correct answer of zero is placed in the accumulator.

One final important point needs to be made. This division routine uses the same normalize and round instructions that the multiplication routine used. These instructions started at DETOUR ($0C7D) in the previous article and are not repeated here. Thus, you will find a JSR DETOUR instruction just before the routine ends.

In listing 2 you will find a short program to test the division routine. It also makes use of the subroutines published in the previous article in this series. In fact, it differs only in that it jumps to the division subroutine rather than the multiplication subroutine. It duplicates almost exactly Listing 5 in "A Floating Point Multiplication Routine," and you may wish to refer to that article for details.

Listing 1. The Floating-Point Division Routine.

```
$0000 = ACCA; Most-significant byte of the mantissa in accumulator A.
$0005 = ACCX; Exponent for accumulator A.
$0007 = ACCS; Sign byte for accumulator A.
$0010 = RES;  Most-significant byte of the quotient accumulator.
$0020 = ACCB; Most-significant byte of accumulator B, the dividend.
$0025 = BCCX; Exponent of the dividend.
$0027 = BCCS; Sign of the dividend.
$0A70 A5 00    START   LDA ACCA       Is the divisor zero?
 0A72 D0 01            BNE BR1        No.
 0A74 00               BRK            Yes.
 0A75 A5 20    BR1     LDA ACCB       Is the dividend zero?
 0A77 D0 05            BNE BR2        No.
 0A79 A9 00            LDA #00        Yes. Make the answer zero.
 0A7B 85 01            STA ACCA+1
 0A7D 60               RTS            Then return.
 0A7E A5 07    BR2     LDA ACCS       Calculate the sign of the quotient.
 0A80 45 27            EOR BCCS
 0A80 45 07            STA ACCS       Return sign to answer location.
 0A84 38               SEC            Now calculate the exponent.
 0A85 A5 25            LDA BCCX
 0A87 E5 05            SBC ACCX       Subtract exponents when dividing.
 0A89 50 01            BVC BR3        Overflow or underflow?
 0A8B 00               BRK            Yes. Go to BRK routine.
 0A8C 85 05    BR3     STA ACCX       No. Put result into answer location.
 0A8E 18               CLC
 0A8F A2 FC            LDX #$FC       Both the mantissa of the divisor and
 0A91 76 04    BR4     ROR ACCA+4,X   the mantissa of the dividend will now
 0A93 E8               INX            be shifted one bit to the right. It
 0A94 D0 FB            BNE BR4        just makes the division routine easier
 0A96 18               CLC            to write.
 0A97 A2 FC            LDX #$FC
 0A99 76 24    BR5     ROR ACCB+4,X
 0A9B E8               INX
 0A9C D0 FB            BNE BR5        So far so good. Next we will clear
 0A9E A9 00            LDA #00        the locations to store the answer.
 0AA0 A2 04            LDX #04
 0AA2 95 10    LOOP    STA RES,X
 0AA4 CA               DEX
 0AA5 10 FB            BPL LOOP       Answer locations cleared.
 0AA7 A0 22            LDY #$22       Bit count = $22 = 34. Start division.
 0AA9 38       CIRCLE  SEC
 0AAA A2 04            LDX #04        Start by comparing divisor to dividend.
 0AAC B5 20    BR6     LDA ACCB,X     Is the dividend greater than divisor?
 0AAE F5 00            SBC ACCA,X
 0AB0 CA               DEX
 0AB1 10 F9            BPL BR6
 0AB3 90 0B            BCC BR8        No. Then put a zero in the quotient.
 0AB5 A2 04            LDX #04        Yes. Subtract divisor from dividend
 0AB7 B5 20    BR7     LDA ACCB,X     and use the result as the new
 0AB0 F5 00            SBC ACCA,X     dividend. The carry flag will be
 0ABB 95 20            STA ACCB,X     set after this operation, and it
 0ABD CA               DEX            will be moved into the quotient.
 0ABE 10 F7            BPL BR7
 0AC0 A2 04    BR8     LDX #04        Here is where the carry flag gets
 0AC2 36 10    BR9     ROL RES,X      put into the quotient.
 0AC4 CA               DEX
 0AC5 10 FB            BPL BR9
 0AC7 A2 04            LDX #04        Now rotate the new dividend left.
 0AC0 18               CLC
 0ACA 36 20    BR10    ROL ACCB,X
 0ACC CA       DEX
 0ACD 10 FB            BPL BR10       Mission accomplished.
 0ACF 88               DEY            So decrement the bit counter.
 0AD0 D0 D7            BNE CIRCLE     Then branch back if it's not zero.
 0AD2 A0 00            LDY #00        Actually, you don't need this instruction.
 0AD4 A5 10    BR11    LDA RES        Here we normalize the mantissa and
 0AD6 30 0B            BMI BR13       adjust the exponent for all the shifting
 0AD8 18               CLC            done earlier.
 0AD9 A2 04            LDX #04
```

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**

Integrated
Computer
Technologies

```
0ADB 36   10        BR12     ROL RES,X
0ADD CA                      DEX
0ADE 10   FB                 BPL BR12
0AE0 C8                      INY            Increment shift counter.
0AE1 D0   F1                 BNE BR11       Branch back until mantissa is normalized.
0AE3 84   0B        BR13     STY TEMP       Calculate the exponent adjustment.
0AE5 A9   07                 LDA #07
0AE7 38                      SEC
0AE8 E5   0B                 SBC TEMP
0AEA 18                      CLC
0AEB 65   05                 ADC ACCX

0AED 50   01                 BVC BR14       Overflow or Underflow?
0AEF 00                      BRK            Yes.
0AF0 85   05        BR14     STA ACCX       Final result into exponent.
0AF2 20   7D  0C             JSR DETOUR     Round and final normalization in
0AF5 60                      RTS            multiplication routine.
```

Listing 2. An Input/Output/Divide Calling Program.

```
$0050  20  00  0E  AGAIN  JSR INPUT    Call the BCD to Floating-Point Binary Routine.
 0053  20  B0  0F         JSR SUB1     Call the subroutine to modify the accumulator.
 0C56  20  C0  0F         JSR SUB2     Transfer ACCA to ACCB.
 0059  20  00  0E         JSR INPUT    Get the second number (divisor).
 005C  20  B0  0F         JSR SUB1     Fix the accumulator again.
 005F  20  70  0A         JSR DIVIDE   Divide the first number by the second.
 0062  20  00  0B         JSR OUTPUT   Convert the result to BCD and output it.
 0065  4C  50  00         JMP AGAIN    Try another pair of numbers.
```

NEW - LOW COST FLOPPY DISK FOR PET*!

PEDISK II

WHY PEDISK?

FAST — The *Pedisk* system loads directly to main memory from the disk and loads — saves information at least 3 times faster than any other disk available.

SIMPLE — A simple command syntax makes its BASIC commands easy to use. At the same time, its sophisticated indexed, sequential, or relative file handling makes it powerful.

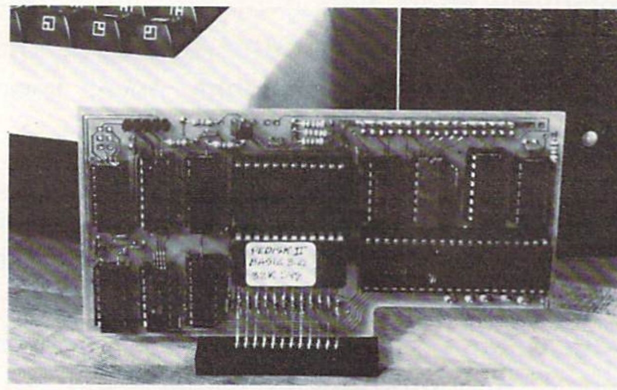
RELIABLE — The ultra-simple circuit design with its LSI disk controller chip provides maximum timing margins for error free operation.

5¼″ system — 1 drive, double density (143K)	$595.00	2 drives, double density (286K)	$945.00
5¼″ system — 1 drive, quad density (286K)	$795.00	2 drives, quad density (572K)	$1,195.00
8″ system — 1 drive, IBM 3740 business (295K)	$1,295.00	2 drives, IBM 3740 business (590K)	$1,895.00

Introducing **PEDISK II**, a low cost high performance floppy disk system consisting of: 1) *PEDISK II* controller, 2) flat disk cable, 3) Disk Drive Assembly, and 4) PDOS II software. The *PEDISK II* is a full function peripheral that provides high speed program storage and a sophisticated file handling package.

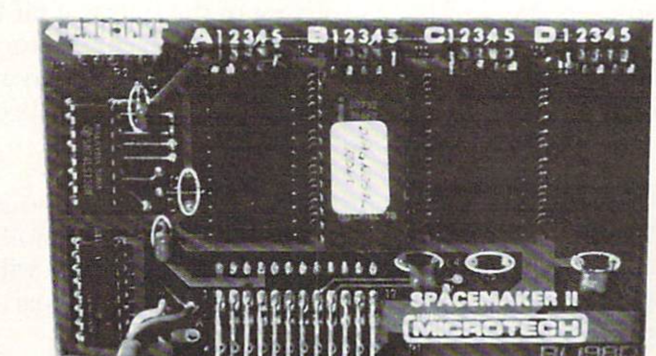
The small *PEDISK II* controller board mounts inside any 2000, 4000, or 8000 series machine. Standard systems are available with one or two disk drives. An optional third drive can also be added. The *PEDISK II* System can be operated simultaneously with any Commodore disk system for data exchange purposes.

The PDOS II software links BASIC by adding a new repertoire of disk commands. !OPEN, !CLOSE, !INPUT, and !PRINT provide the basis of the powerful file handling package. !LOAD, !SAVE and !RUN allow complete disk control. PDOS II also offers a full DOS-mode of operation for all disk diagnostic and utility functions. Diskette format, backup, diagnostic, and reorganization capabilities are provided.



PEDISK II CONTROLLER BOARD

NEW - SPACEMAKER II switch between one of FOUR ROMS



SPACEMAKER II is the new ROM switch from Microtech. It allows either manual or software controlled switching of up to four ROMs in a single ROM expansion socket. The switching is accomplished with a side-mounted slide switch or via ROMDRIVER, an accessory board which allows software controlled switching and keyboard controlled switching. ROM I/O is a special software package available on disk to implement full keyboard control of the ROMs. In addition, it adjusts for memory differences found in various utility ROMs.

SPACEMAKER II	$39.00
ROMDRIVER	$39.00
ROM I/O Commodore or Pedisk	$ 9.95

SEE YOUR LOCAL DEALER FOR ALL cgrs MICROTECH PRODUCTS

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

*PET IS A TRADEMARK OF COMMODORE

Part Two:

The Practical Aspects Of Assembly Language Programming

Bruce D. Carbrey,
Raleigh, NC

Editor's Note: Last month, in the first part of this article, the author explored some methods of handling flags. At the end, he discussed setting aside bytes for flags. Here he introduces some additional techniques. To begin with, he proposes a more efficient method of storing and testing flags. RM

If instead you choose $80 to represent true and $00 to represent false, you can use the BIT instruction to test the flag without having to save the A register:

```
BIT   ALFALK   ;TEST THE FLAG
BPL   FOLD1    ;BRANCH IF NO "FOLDING" DESIRED
```

You don't have to save A because the BIT instruction sets the sign flag according to the status of bit 7 of the operand, without altering the accumulator. This saves you 4 bytes in your program, as shown in Listing 3. It also runs faster. You now know two rules to improve efficiency:

Rule 1: Use bit 7 of a byte as a flag.

Rule 2: A flag in memory can be tested without "clobbering" a register by using the BIT instruction.

Now that you know how to test the flag, you will want to be able to set or clear it. This may seem terribly obvious, for example,

```
LDA   #$80
STA   ALFALK   ;ENABLE ALPHA-LOCK MODE
```

sets the flag and,

```
LDA   #$00
STA   ALFALK   ;DISABLE ALPHA-LOCK MODE
```

clears the flag. This method uses one less byte to set the flag and two less bytes to clear the flag! On the

negative side, it takes two machine cycles longer than the first method to set the flag, but is equally fast for clearing the flag. The shift-method also does not clobber the A register, which may often be useful. Again on the negative side, you could argue that the shift method is not as straightforward as the first method, and also that it leaves the remaining seven bits of the flag "undefined". However, this can also be useful, as I shall now demonstrate.

Suppose at some point in your program you want to *temporarily* allow entry of lower case letters, and then *restore* the previous mode (either alpha-lock or non-alpha-lock, whichever was previously in effect). One method might be:

```
LDA   ALFALK  ;RECALL PRESENT ALPHA-MODE
              STATUS FLAG
PHA           ;SAVE ON STACK
LDA   #0
STA   ALFALK  ;DISABLE ALPHA LOCK TEMPORARILY
...
(code using lower case input...)
...
PLA           ;RECALL ORIGINAL ALPHA-LOCK
              STATUS
STA   ALFALK  ;RESTORE OLD MODE
```

This program segment uses the stack to save and restore the flag status. Now consider this alternative:

```
LSR   ALFALK  ;SAVE OLD MODE, CLEAR ALPHA LOCK
...
(code using lower case input...)
...
ASL   ALFALK  ;RESTORE PREVIOUS ALPHA LOCK
              MODE
...
```

This program segment performs the same function in 6 bytes instead of 13, runs faster, and doesn't clobber the accumulator! It illustrates a simple but powerful fact:

Rule 3: A single byte can be used as an 8-level push-down stack for flags.

Shifting the flag byte right moves the previous status into bit 6; shifting the flag left restores the old flag back into bit 7. This rule has several corolaries which are occasionally useful:

Rule 4: You can test the previous (saved) flag by using a BIT instruction followed by a BVC or BVS instruction.

... programs will have fewer branches, will use less memory, and will run faster ...

Rule 5: You can test both flags (bit 7 and bit 6) with only one BIT instruction.

For example:

```
BIT   FLAG    ;TEST THE FLAG
BMI   NEWSET  ;BRANCH IF PRESENT FLAG IS SET
BVS   OLDSET  ;BRANCH IF PREVIOUS FLAG WAS SET
...
```

Another side effect is:

Rule 6: You can test a flag and restore it to its previous state at the same time by using ASL followed by BCC or BCS.

For example:

```
ASL   ALFALK  ;DISCARD PRESENT, RESTORE OLD
              FLAG
BCS   ISSET   ;BRANCH IF DISCARDED FLAG WAS
              SET
...
```

The same sequence can be used to clear the flag instead if it was initialized to 0 originally and was not used as a stack. All these functions have the advantage of not disturbing any registers (except the PSW). Since they are slightly "tricky", you should document your code with clarifying comments.

As you can see, there's more to the simple little flag than meets the eye! Properly used, flags can greatly simplify and improve your programming. If you try the techniques presented here, I think you will find that your programs will have fewer branches, will use less memory, and will run faster. In next month's installment, we will look at methods for improving machine language loops.

Listing 3: Improved Keyboard Driver With Alpha-Lock Flag Using Bit 7 = 1 = True

```
;
;      SUBROUTINE INCH: KEYBOARD DRIVER FOR ASCII-ENCODED
;      KEYBOARD WITH PARALLEL INTERFACE.
;
;
;      ADDRESSES SHOWN ARE FOR 6530 ON KIM-1 COMPUTER.
;      KEYBOARD DATA LINES TO PORT A BITS 0 TO 6,
```

```
                    ;       NEGATIVE-GOING STROBE TO BIT 7.
                    ;
                    ;       ON ENTRY: IF ALFALK BIT 7 IS 1, THEN LOWERCASE LETTERS WILL
                    ;       BE RETURNED AS THE EQUIVALENT UPPERCASE ALPHA.
                    ;       ON RETURN: REGISTER A = ASCII CODE FOR KEY PRESSED;
                    ;       X AND Y PRESERVED.
                    ;
1700                PAD     =       $1700       ;KIM PORT A DATA REGISTER ON 6530
1701                PADD    =       $1701       ;KIM PORT A DATA DIRECTION REGISTER
                    ;
0000                        *=      $1780       ;PROGRAM ORIGIN
                    ;
1780 A900           INCH    LDA     #$00
1782 8D0117                 STA     PADD        ;SET PORT DIRECTION = INPUTS
1785 AD0017         INCH1   LDA     PAD         ;TEST PORT
1788 30FB                   BMI     INCH1       ;WAIT FOR STROBE PULSE
178A 2C0017         INCH2   BIT     PAD
178D 10FB                   BPL     INCH2       ;WAIT FOR END OF STROBE
                    ;
                    ;       IF ALPHA-LOCK FLAG IS SET, FOLD ANY LOWERCASE LETTERS TO
                    ;       EQUIVALENT UPPERCASE LETTERS.
                    ;
178F 2C9F17         FOLD    BIT     ALFALK      ;TEST "ALPHA LOCK" FLAG
1792 100A                   BPL     FOLD1       ;BRANCH IF NO FOLDING DESIRED
1794 C97B                   CMP     #$7B        ;LOWER CASE "Z" + 1
1796 B006                   BCS     FOLD1       ;BRANCH IF PUNCTUATION
1798 C961                   CMP     #$61        ;LOWER CASE "A"
179A 9002                   BCC     FOLD1       ;BRANCH IF NOT LOWER CASE ALPHA
179C E920                   SBC     #$20        ;ELSE FOLD TO EQUIVALENT UPPERCASE
179E 60             FOLD1   RTS
                    ;
                    ;       ALPHA LOCK FLAG (DEFAULT = ALLOW LOWER CASE)...
                    ;
179F 00             ALFALK  .BYTE   0           ;"ALPHA LOCK" FLAG; NON-0=UPPERCASE ONLY.
                    ;
0000                        .END
NO ERROR LINES
```

©

THE apple® GAZETTE

Apple Disk Motor Control

William W. Martin
Seven Valleys, PA

The purpose of this article is to demonstrate a method of software motor control operation for the APPLE II DISK system.

The original reason for adoption of this performance patch was to decrease the time required to load the data file for a TEXT EDITOR I normally use at work. This design can be used anytime maximum performance is desired. The improvement seen for the demonstration program is about 20 percent, but may be greater for TEXT FILES, especially when time is taken to process the data as it is input from the disk.

This performance patch stops the APPLE DOS from turning off the disk motor as normally done during a disk operation. This function can now be controlled by the user with a definite performance improvement without modifying the normal DOS skew parameters.

EXAMPLE: First, since we will be changing the instructions of the DOS, please use a scratch disk in case we destroy it.

Now, please perform these steps in the following order, from the immediate mode.

VERIFY ORIGINAL DATA..

1.'PRINT PEEK (-16834)' - DOS VERSION 3.2 or 3.2.1
OR
'PRINT PEEK (-16819)' - DOS VERSION 3.3
THE VALUE RETURNED SHOULD BE: 189

TO SET DOS PATCH..

2. 'POKE -16834,96' - DOS VERSION 3.2 OR 3.2.1
OR
'POKE -16819,96' - DOS VERSION 3.3

3. NOW TYPE 'CATALOG'

Notice that the disk is still spinning after completion of the 'CATALOG'.

4. TYPE 'CATALOG' AGAIN

Notice how the 'CATALOG' function is performed faster since the motor is already up to speed. Repeat step four to test this again.

TO RESTORE TO NORMAL..

5. 'POKE' -16834,189' - DOS VERSION 3.2 OR 3.2.1
OR
'POKE -16819,189' - DOS VERSION 3.3

This will restore the DOS back to its original value.

6. 'POKE -16152,0' - ALL DOS VERSIONS

This will turn the motor off.

NOTE: 'POKE -16151,0' - TURN DISK MOTOR ON
'POKE -16152,0' - TURN DISK MOTOR OFF

This same method can be used under program control to obtain the same results.

The Demo Program

After entering the program and verifying that it is correct, insert your scratch disk. Type RUN and observe how the DISK motor stops while loading normally, but continues to spin while loading the "TEST" file with the patch set.

Program Description

Lines 100–150 — These lines print the header, change the text window setting, and send the DOS command 'MON I,O,C'. There is a centering routine used here by setting 0$ to the desired output and then executing a GOSUB 970.

Lines 160–179 — This sets up the variable 'D' to a value (200 in this example) that is used to simulate a short delay which would normally be encountered if the input data is processed while being input.

Lines 180–310 — This section establishes the dummy T$ as 'ABCDEFGHIJKLMMNOPQRSTU VWXYZ' and stores it to the scratch disk 100 times.

Lines 320–500 — The 100 T-strings are now read back from file 'TEST' under the normal DOS. Notice how the motor turns on and off while reading the 'TEST' file. Each time the motor starts up again, some time is wasted while waiting for the disk to come to the proper speed.

Lines 510–760 — The 100 T-strings are again read back, but this time using our disk motor patch. Notice that with the patch, the disk motor no longer stops and always maintains proper speed. Besides the time improvement, it is probably a little easier on the APPLE power supply by reducing the on/off cycle for the motor.

NOTE: Line 590–610 — The ONERR GOTO at line 610 is necessary when using this patch because, without it, if a disk error is encountered, the program would break without restoring the original values and turning off the motor. Notice that the ONERR GOTO routine at line 950 insures that the original will be restored due to a disk error of some type.

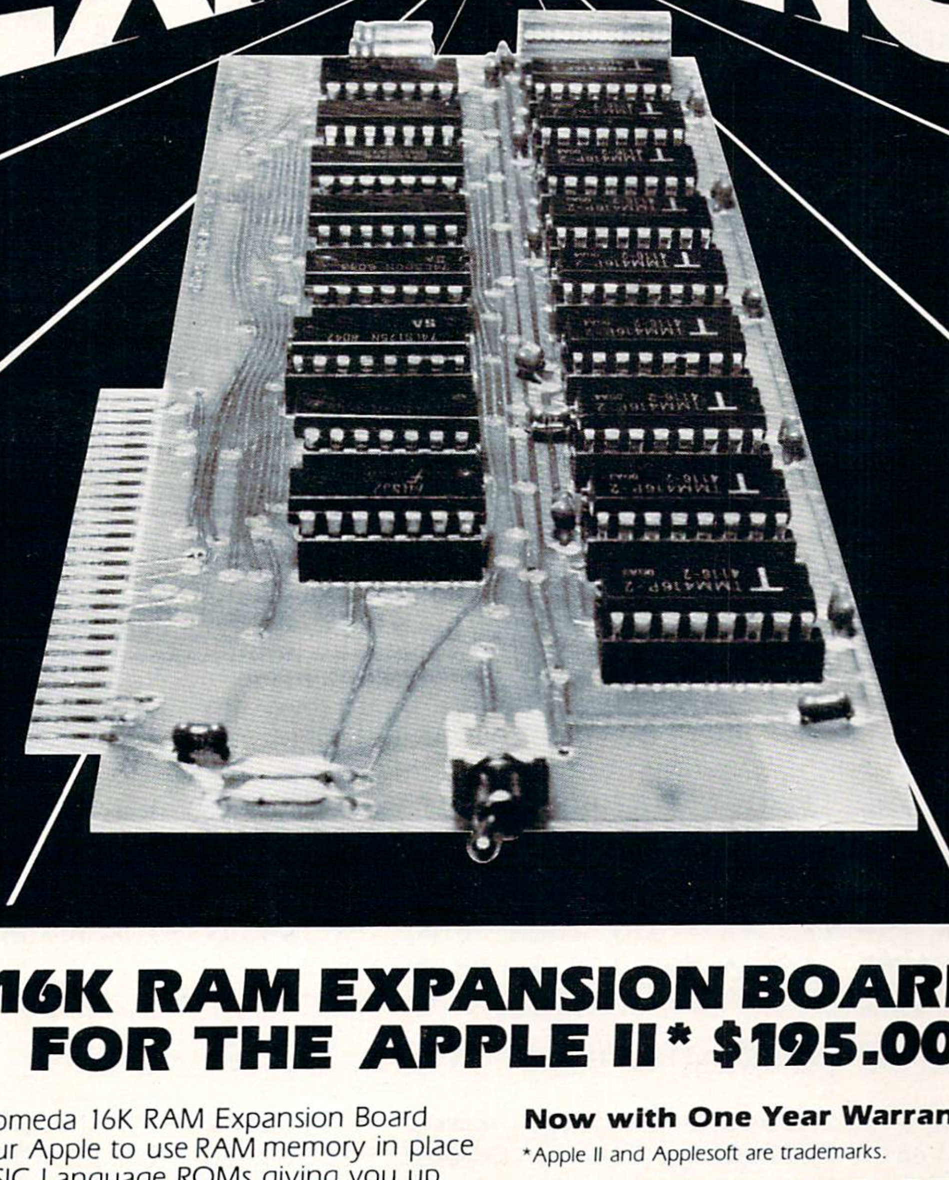
Assembly Language Notes

What this routine does is replace an LDA ,X instruction that normally stops the disk motor with an RTS instruction. This tricks the DOS into thinking that it turned off the motor.

NOTE: I wish to give special thanks to Mr. E. L. Didion for the information to make this program possible.

```
100  REM   MOTOR CONTROL DEMO BY BILL MARTIN
110 :
120  TEXT : HOME : PRINT :O$ = "DOS MOTOR CONTROL DEMO": GOSUB 97
     0: PRINT :O$ = "BY BILL MARTIN": GOSUB 970
130 O$ = "---------------------------------------": GOSUB 970: POKE
     34,5: HOME
140 D$ =  CHR$ (4): PRINT D$;"MON I,O,C"
150 :
160 D = 200: REM    VARIABLE 'D' IS LOOP DELAY TO SIMULATE INPUT
     DATA PROCESSING
170 :
180 T$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
190  REM    SAVE T$ 100 TIMES
200 :
210  HOME : PRINT "PRESS A KEY TO SAVE 'TEST' FILE..";: CALL  - 7
     56: PRINT : HOME
220 O$ = "SAVING 'TEST' FILE": GOSUB 970:O$ = "------------------
     ---------------------": GOSUB 970: POKE 34,8: HOME
230 :
240  PRINT D$;"OPEN TEST"
250  PRINT D$;"DELETE TEST"
260  PRINT
270  PRINT D$;"OPEN TEST"
280  PRINT D$;"WRITE TEST"
290  FOR N = 1 TO 100: PRINT T$: NEXT
300  PRINT D$;"CLOSE"
310 :
320  REM  READ BACK NORMAL
330 :
340  REM  TAKES ABOUT 51 SECONDS
350 :
360  HOME : PRINT "PRESS A KEY TO READ 'TEST' FILE NORMAL";: CALL
      - 756: PRINT : HOME
370  POKE 34,5: HOME :O$ = "READING 'TEST' FILE - NORMAL": GOSUB
     970:O$ = "---------------------------------------": GOSUB 970: POKE
     34,8: HOME
380 :
390  PRINT D$;"OPEN TEST"
400  PRINT D$;"READ TEST"
410  FOR N = 1 TO 100
420  INPUT "";T$
430 :
440  REM  SIMULATE PROCESSING INPUT DATA
450  FOR C = 1 TO D: NEXT C
460 :
470  NEXT N
480  PRINT D$;"CLOSE"
490 :
```

EXPAND ✧ ENHANCE



16K RAM EXPANSION BOARD FOR THE APPLE II* $195.00

The Andromeda 16K RAM Expansion Board allows your Apple to use RAM memory in place of the BASIC Language ROMs giving you up to 64K of programmable memory. Separate Applesoft* or Integer BASIC ROM cards are no longer needed. The 16K RAM Expansion Board works with the Microsoft Z-80 card, Visicalc, DOS 3-3, Pascal, Fortran, Pilot, and other software. A switch on the card selects either the RAM language or the mainboard ROMs when you reset your Apple.

The Andromeda 16K RAM Expansion Board has a proven record for reliability with thousands of satisfied customers.

Now with One Year Warranty.

*Apple II and Applesoft are trademarks.

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

Distributed By:

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

```
500  PRINT : PRINT "NORMAL READ COMPLETE": PRINT  CHR$ (7): PRINT

510  POKE 34,5: HOME :O$ = "READING 'TEST' FILE - IMPROVED": GOSUB
     970:O$ = "------------------------------": GOSUB 970: POKE
     34,8: HOME
520  PRINT "PRESS A KEY TO READ WITH MOTOR CONTROL";: CALL  - 756
     : PRINT : PRINT
530 :
540 :
550  REM  READ WITH MOTOR CONTROL
560 :
570  REM  TAKES ONLY 41.5 SECONDS WITH MOTOR PATCH ACTIVE
580 :
590  REM   IMPORTANT TO USE 'ON ERR GOTO' !
600 :
610  ONERR  GOTO 930
620 :
630  GOSUB 820: REM  SET MOTOR CONTROL PATCH
640 :
650  PRINT D$;"OPEN TEST"
660  PRINT D$;"READ TEST"
670  FOR N = 1 TO 100
680  INPUT "";T$
690 :
700  REM  SIMULATE PROCESSING INPUT DATA
710  FOR C = 1 TO D: NEXT C
720 :
730  NEXT N
740  PRINT D$;"CLOSE"
750 :
760  PRINT : PRINT "IMPROVED READ COMPLETE": PRINT  CHR$ (7): PRINT

770  PRINT D$;"DELETE TEST"
780  POKE 216,0: REM  DON'T NEED ONERR GOTO NOW
790  GOSUB 870: REM  RESTORE NORMAL DOS
800  HOME : PRINT "TEST COMPLETE....": TEXT : END
810 :
820  REM  SET MOTOR PATCH HERE
830 :
840  IF  PEEK ( - 16834) = 189 THEN  POKE  - 16834,96: REM  DOS 3
     .2.1 (CHANGE -16834 TO -16819 FOR DOS 3.3)
850  RETURN
860 :
870  REM  RESTORE MOTOR PATCH HERE
880 :
890  IF  PEEK ( - 16834) = 96 THEN  POKE  - 16834,189: REM  RESTO
     RE DOS 3.2.1 ORIGIONAL VALUE (CHANGE -16834 TO -16819 FOR DO
     S 3.3)
900  POKE  - 16152,0: REM  TURN MOTOR OFF
910  RETURN
920 :
930  REM  ON ERR ROUTINE
940 :
950  GOSUB 870: REM  SHOULD RESTORE ORIGIONAL DOS VALUES AND STOP
      MOTOR
960  POKE 216,0: HOME : VTAB 5: PRINT "DISK ERROR": PRINT : PRINT
     "PRESS ANY KEY TO TRY AGAIN.. ";: CALL  - 756: PRINT : GOTO
     550
970  HTAB 20 -  LEN (O$) / 2: PRINT O$: RETURN : REM  CENTERING R
     OUTINE
```

©

Interfacing The Apple To 6500 Family Peripherals

David Paul and James Wisman
Department of Chemistry
University of Arkansas
Fayetteville, AR

It has been stated previously that 6500 peripheral chips (6522, 6551, etc.) accessed from the Apple II peripheral bus must undergo complete address decoding. This results from the fact that 6500 chips require stable address and chip select lines 180 nanoseconds before the positive edge of the Φ2 clock (1). In this paper, a simple delay circuit allows use of the Apple device select lines to avoid the complexity of full address decoding.

Initially, a test program was written to explore the timing between Φ0 and device select on the Apple II peripheral bus. The test program was as follows:

```
0300 AD  B0  C0  over  LDA slot address
0303 4C  00  03        JMP over
```

This simple program produces continuous low going pulses on the address device select line. Observation via dual channel oscilloscope showed the relationship between Apple peripheral bus lines, device select, and Φ0. Figure 1 illustrates why 6500 family chips have difficulty working with the Apple II peripheral bus using the device select lines. Note that the falling edge of device select corresponds with the rising edge of Φ0. The device select line represents the address decode of the high order address lines that have been logically ANDed with Φ0. For the 6500 family, address lines must be stable 180 nanoseconds before the positive transition of the Φ2 clock line (2). (The Φ2 of 6522 is Φ0 from the Apple II Peripheral Bus.) Figure 1 shows absence of the needed delay. The solution calls for development of circuitry to delay the positive edge of Φ0, making the device select line useful.

A positive edge delay circuit is shown in Figure 2, with a corresponding timing diagram, Figure 3. In the resulting output waveform, the positive edge has been delayed several nanoseconds. The basic concept of the delay circuit centers around the voltage required to trigger a TTL gate. For the 74LS08 AND gate, potentials greater than a threshold of 1.0 volts are considered to be logical "highs." Rationalization for the delay circuit is as follows. When the input goes low, the output of the first AND gate (Pin 3) goes low: the diode is forward biased, discharging the capacitor so that both inputs to the second AND gate are low, resulting in a low output. When the input rises high, the output of the first AND gate goes high to reverse bias the diode. Current is then in the RC network charging the capacitor. The voltage across the capacitor rises until the 1.0V threshold level is reached. At this potential, the second AND gate recognizes the voltage level as a high (Pin 5), changing the state of the output.

The magnitude of the delay can be adjusted by changing the value of the RC time constant. The voltage drop across the capacitor (V_{TTL}) can be expressed by:

$$V_{TTL} = V(1-e^{-t/RC})$$

where: V - the TTL high (4.0V)
t - time
RC - resistor and capacitor values.

Rearranging, this equation takes the form:

$$t_{delay} = [-R\,\ln(1-\frac{V_{TTL}}{V})]C$$

where: t_{delay} - the rising edge delay
V_{TTL} - threshold voltage 1.0V

Therefore, for a constant resistance R, the amount of delay is proportional to the size of the capacitor (3). For example, for R = 1.5kΩ and C = 330pf, the predicted delay would be 140 nanoseconds, which agrees with that obtained experimentally. The upper limit of the delay results from an RC time constant that will not allow the capacitor to reach the 1.0V threshold during the 1/2 cycle time of ~500 nanoseconds. For R = .5kΩ, the maximum capacitor size is about 2000pf.

Advantages of this circuit are: 1) a TTL level signal results, 2) the falling or trailing edge is unaffected, and 3) inexpensive and simple to incorporate into existing peripheral design.

Implementation of delay circuit utilizing a 6500 family peripheral is shown in Figure 4. A 6522 VIA was chosen to test the applicability of the delay circuit. A simple inspection of memory through the Apple monitor can determine if the 6522 registers are communicating with the Apple. Simple software routines to define data direction, to read and write data to both 8 bit ports, and to set and decrement timers were used to determine that the 6522 was fully functional when using the delay circuit. All features of the 6522 are non-functional with no positive edge delay of Φ0.

Even though all 6500 family chips have not been tested, the authors presume that this delay technique is applicable to all 65XX peripheral devices. This investigation has revealed one excep-

tion to the Φ0 delay requirement. The 6520 PIA will function properly when connected directly to the Apple Φ0 and device select lines.

In summary, the delay of the positive edge of Φ0 from the Apple peripheral bus results in simpler interfacing with the 6500 family. Eliminating full address decoding utilizes slot independence of peripheral cards that are less expensive and easier to construct.

References

1. De Jong, Marvin C. **COMPUTE!**, 3, #3. pg. 142
2. Rockwell Data Sheet Document NO: 29000 D47, Revision, 2, June 1979.
3. Diefenderfer, A. James; Principles of Electronic Instrumentation, 2nd Ed. W. B. Sanders; Co. 1979.

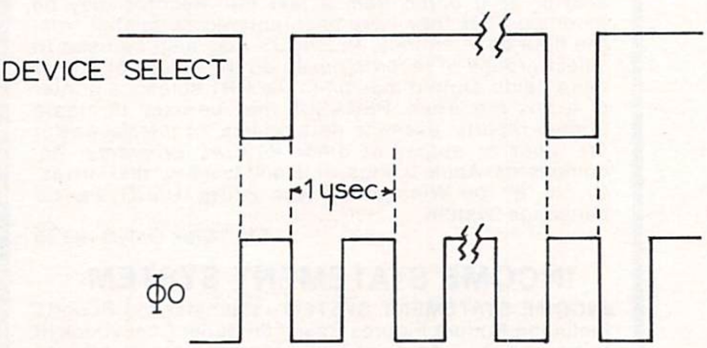


Figure 1.

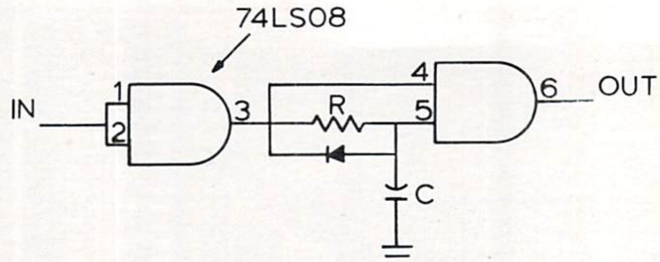


Figure 2.

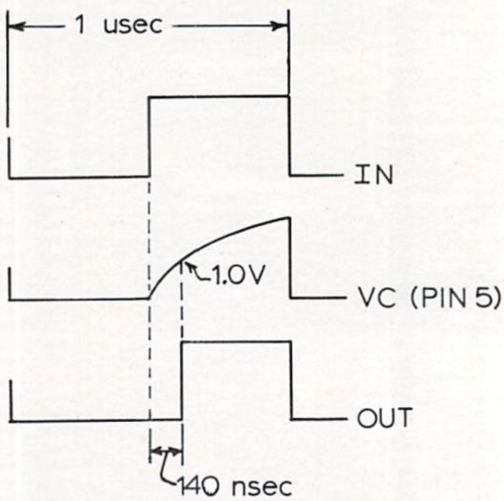


Figure 3.

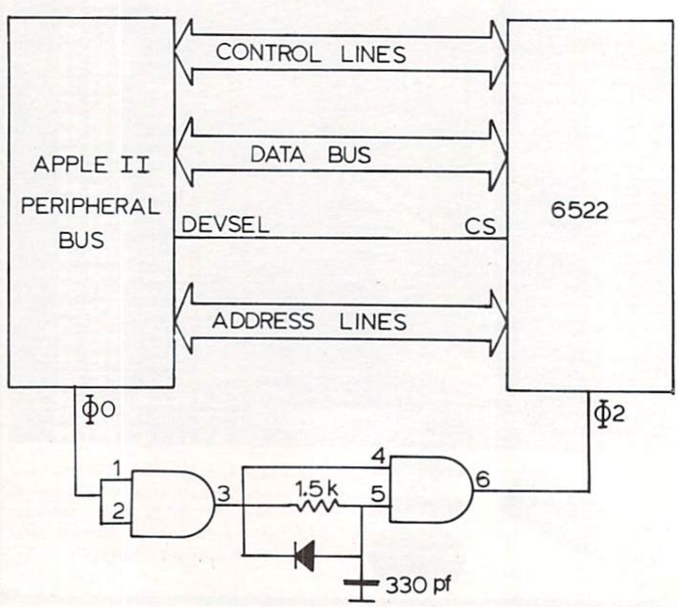


Figure 4.

A Cassette Tape Monitor For The Apple

Jim Lowell
Whitehouse Station, NJ

"Type LOAD; do not press RETURN yet. Remove the plug from the earphone jack of the tape recorder. Press the PLAY button on the recorder and advance the tape until you hear the leader tone. Stop the recorder and replace the plug in the earphone jack. Start the tape, and immediately press RETURN."

The Tape Monitor

To most tape-recorder-based APPLE computer users, the complex instructions above are a source of continued annoyance and frustration. Fortunately, they are also unnecessary.

The tape monitor unit described in this article eliminates the need to remove the plug from the earphone jack and simplifies the tape loading instructions to: "Type LOAD; do not press RETURN yet. Press the PLAY button on the tape recorder. When you hear the leader tone, press RETURN."

For about one hour's work and one-half the cost of a good tape program (about $12.00), you can build a tape monitor unit for your APPLE. To simplify the construction process, I have included a set of templates to locate the required mounting holes and both a schematic diagram (for those of you who are electrically inclined) and a wiring diagram (for those of you who aren't).

Using The Monitor

Use of the monitor unit has several advantages over the remove-the-plug method of tape loading:

- It simplifies the loading procedure.
- It allows the user to listen to the tape as it is loaded.
- It eliminates the need to change the tape recorder volume control setting to avoid disturbing others while listening to the leader tone.
- It reduces the wear and tear on the earphone jack of the recorder and the mini-plug of the computer connecting cable.

To use the monitor for the first time, connect the cord from the *tape in* jack of the APPLE to the mini-jack of the monitor. Then, insert the mini-plug of the monitor into the earphone jack of the tape recorder. Put a tape program into the machine and press PLAY. Adjust the volume control on the monitor (don't change the setting on the recorder yet) until it is at the desired level. Now, rewind the tape and go through the following loading steps:

- Type LOAD without hitting return.
- Start the tape.
- When you hear the leader tone, hit RETURN.
- When the cursor re-appears, rewind the tape.

If the tape fails to load properly, the problem could be that the speaker in the monitor is drawing some of the required signal away from the computer (this has never been a problem with my Panasonic recorder — I load most tapes at a volume setting of "4"). If the problem occurs, however, turn the recorder volume up a notch or two — that should fix it.

Building The Monitor

The parts list below uses mostly Radio Shack components because they are widely available; any comparable parts will do.

Here are my suggested steps for building the monitor.

1. Using the templates, locate and drill all holes in both the top and case.

2. Fasten all the proper parts to the top of the project case (i.e. everything but the mini-jack, mini-plug, control knob, and cord). Be sure you mount the speaker cloth over the right hole before mounting the speaker.

3. Connect these parts together as per the schematic or wiring diagrams (using single-conductor copper wire). The wires coming from the switch and volume control to the mini-jack should be about eight inches long. This is to allow the top of the project case to be easily removed if necessary. Don't connect the wires to the mini-jack yet.

4. Put the rubber grommet in the proper hole in the side of the project case (the one where the cord will go).

5. Connect the mini-plug to about one foot of shielded cable (the shield goes to the "ring" terminal of the plug), and thread the other end of the cable (with the shield and conductor already stripped and tinned) through the grommet into the case.

6. Twist the shield of the cord and the end of the wire from the volume control together, and solder them to the "ring" terminal of the mini-jack.

7. Twist the conductor of the cord and the end of the wire coming from the switch together and solder them to the "tip" terminal of the mini-jack.

8. Fasten the mini-jack in position in its hole in the side of the case.

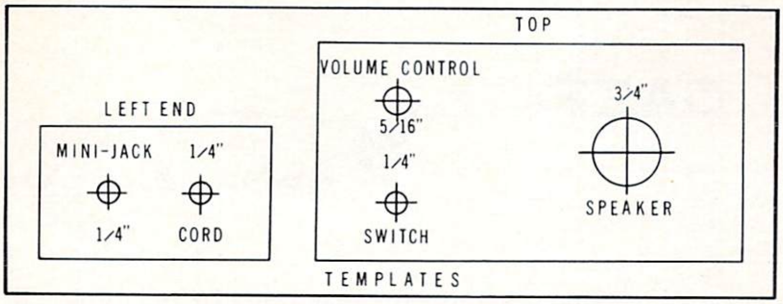
9. Screw on the top of the case, put the control knob onto the volume control post and you're done.

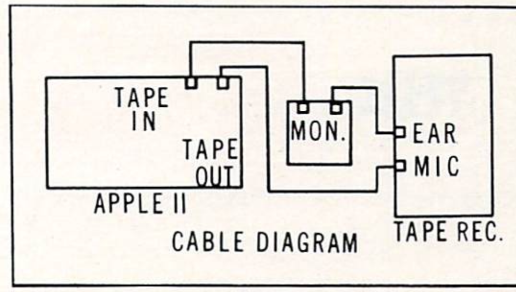
I've tried to supply all the necessary instructions to make building the tape monitor an easy evening's project. If, however, you encounter any problems, drop me a line at the address below:

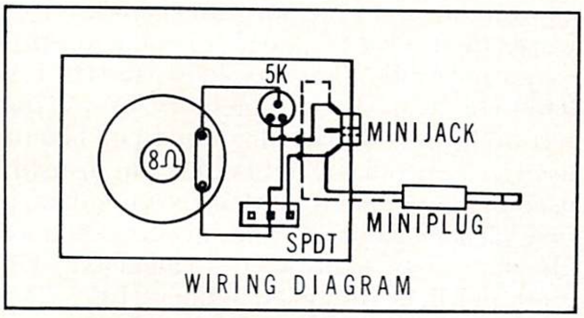
Jim Lowell
P.O. Box 364
Whitehouse Station, NJ 08889

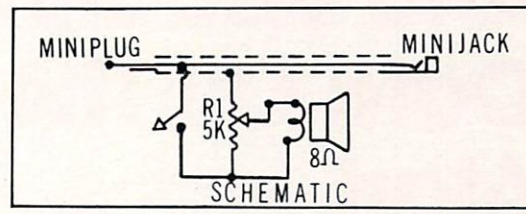
PARTS LIST

COMPONENT	PART#
1. Project case (4-3/4" x 2-1/2" x 1-2/5")	270-222
2. 2" speaker (8 ohm)	40-245
3. Mini-switch	275-662
4. Mini-volume control (5K ohm)	271-214
5. Control knob	274-415
6. Mini-jack	274-296
7. Mini-plug (shielded)	274-288
8. 1½ feet of single-conductor shielded cable.	
9. 1½ feet of single conductor copper wire.	
10. 1¼ inch piece of speaker cloth (or loose-weave fabric).	
11. A rubber grommet to fit a ¼-inch hole.	









©

Diskette Sector Space In A Greeting Program

R. R. Hiatt
St. Catherines, Canada

Most Apple users include the system command CATALOG in their greeting program so that when the DOS is booted, the diskette's contents are automatically listed. CATALOG, of course, displays the number of sectors used for each file as well as the file name. What it does not report is the total number of used sectors, or the remaining number unused. To find out this rather useful information, the user must sum the individual numbers and subtract from 403. It's a small chore, but annoying, particularly as the computer should be doing the arithmetic, not the user.

There are two problems: The first, that the numbers are non-resident (except in the screen memory area) is easily solved by PEEKing them out. My program does this, and works quite well as long as the diskette holds fewer than 24 files. It will always sum and report for the last 23 files, but that can be misleading. Consequently, I restrict myself to 23 files per diskette.

The real problem lies in the construction of the CATALOG routine itself. The output is paged, but control is not returned to the program until the entire contents have been screened. Thus, the program stands by helplessly as the numbers it's supposed to PEEK scroll upwards, off the screen, and into oblivion.

It would be an easy matter to amend CATALOG if one knew where it was. But searching the 12K bytes of disassembled DOS 3.2 in the hope of recognizing the routine seems more work than it's worth. Until some thoughtful person publishes a DOS map, I'll be content with my present, admittedly imperfect, but useful, program.

```
10  REM GREETING W CATALOG & ADD
20  REM SECTOR ADD FOR 23 FILES OR FEWER
30  REM CATALOG MUST FINISH BEFORE ADD
WILL START
40  PRINT "SLAVE DISK INIT ON 48K APPLE"
50  PRINT "BY R. HIATT, 12/1/80"
60 D$ = CHR$(4)
70 S = 128:D = S + 48
80  FOR I = 1 TO 2000: NEXT I: HOME
90  PRINT D$;"CATALOG"
100 T = 0:CT = 0
110  FOR BASE = 1024 TO 1104 STEP 40
120  FOR J = 0 TO 7
130 ROW = BASE + J * S
140 T1 = T
150  FOR COL = 3 TO 5
160 A = PEEK (ROW + COL) - D
170  IF A < 0 OR A > 9 THEN 190
180 T = T + A * 10 ↑ (5 - COL)
190  NEXT COL: IF T > T1 THEN CT = CT + 1
200  NEXT J: NEXT BASE
210  PRINT : PRINT "SECTORS USED TOTAL
";T
220  PRINT : PRINT "UNUSED SECTORS =
";403-T
230  PRINT "NUMBER OF FILES = ";CT
240 END
```

©

THE ATARI® GAZETTE

Restoring Data And Updating Data On The Atari

Bruce Frumker
Cleveland Heights, OH

Editor's Note: Many Atari users have been waiting for this. For years, one of the most useful programming techniques for the PET involved "dynamic keyboard" programming. This technique permits the machine to automatically "press" its own RETURN key and, in effect, program itself. This permitted automatic line-numbering, block deletion, DATA statements automatically generated, etc.

In this article, Mr. Frumker opens this important door for Atari programmers. Note that POKE 842,13 puts Atari into "RETURN-key mode." The cursor can be made to travel over a GOTO, sending it to a line containing POKE 842,13, to halt the "dynamic" mode. In the program below, text and background are set to identical colors, hiding the action. There is much to experiment with here. When you design a new utility using "dynamic keyboard," send it in to **COMPUTE!**. *—RTM*

One of the very nice features of Atari Basic is the ability to RESTORE to a specific line number of DATA. You can even RESTORE to a variable where that variable represents a line number of DATA. This opens up some nice possibilities. For example, you can generate random line numbers for random testing. In addition, if one of the items in a DATA line is a repetition of the line number, then that can be stored and retrieved later.

I used these possibilities in a homework practice program for my daughters. They often get lists of words in school whose meanings have to be memorized. That led to the following program where they can be randomly tested on the words (in this case, prefixes). When an error is made, the line number is saved in an array, so at the end those words and definitions that need more study can be RESTOREd and printed out.

Along the way, the program evolved into allowing my daughters to enter new lists of practice words themselves. In order to make this process as easy as possible, all of the old words (DATA lines) had to be cleared out of memory, in case the second list was shorter than the first. The method used to clear out the old DATA can instead be used in a program to update existing data, or save new variable values, or whatever else you would like that can be expressed as a legal line. This method will be discussed at lines 2100 – 2150.

I've allowed for 100 lines of practice space, although that many are seldom used. In the program that follows, only lines 1 through 10 are used, as an example. The format is: line number DATA repetition of the line number, question, answer. There are often two definitions for a word, so I've allowed for two answers (B$ for the first, and C$, if needed, for the second). Also, if a word is difficult and needs extra practice, it can be repeated (as I have done on lines 8 and 9) to increase its random frequency of occurence.

The Program

Lines

0 – Line 2000 is the introductory part of the program. It returns to line 110.

110 – The OPEN to screen editor here makes it possible to have INPUT without a ? prompt (by using INPUT #1; var). It also clears the screen when first OPENed, so I put it here where I wanted a screen clear anyway.

120 – CTRL M (44 times). This string is the underlining for the correct definitions. If, instead, you used a for-next loop for underlining, you would see it underline from left to right, which is distracting.

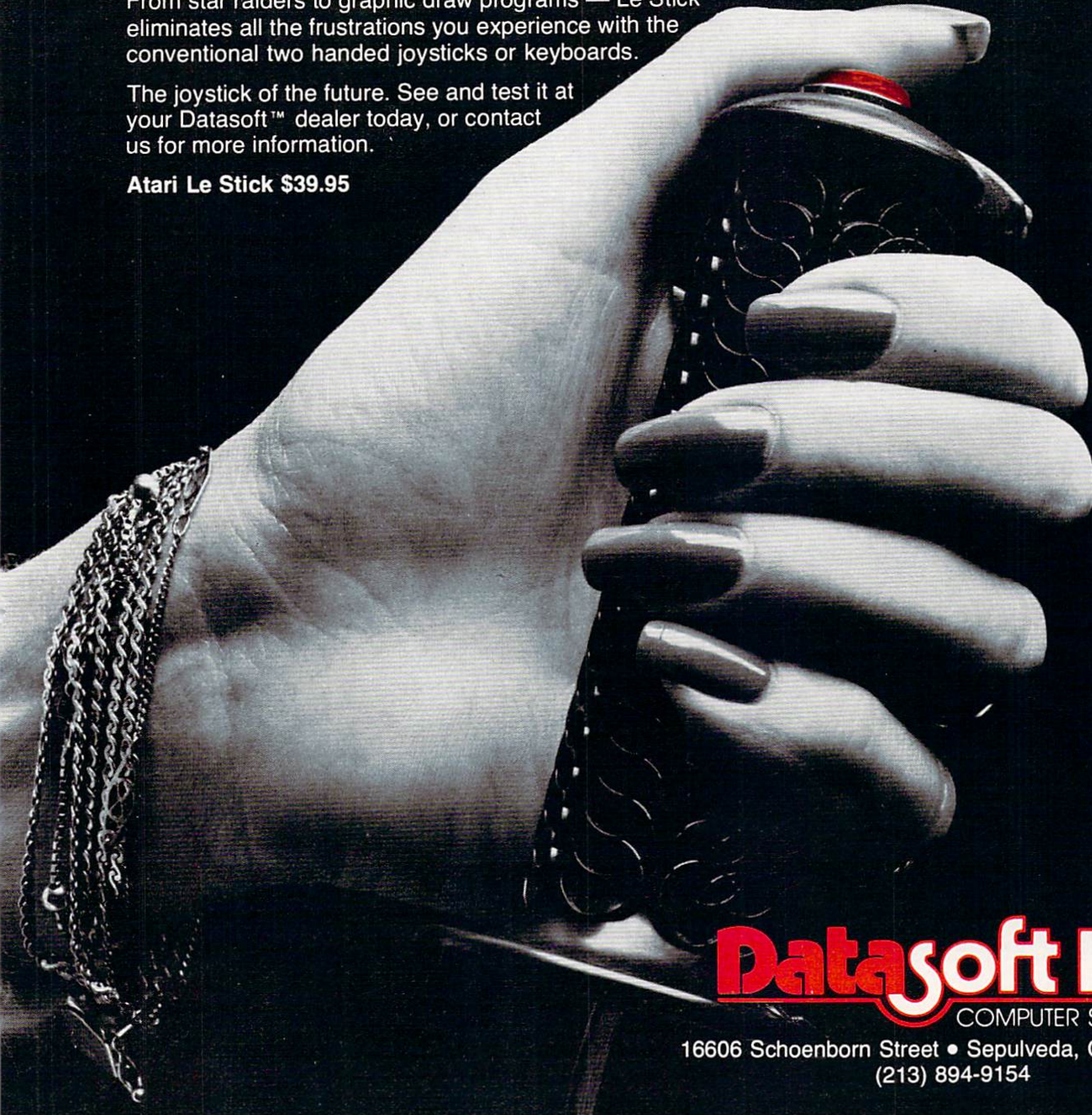
125-150 – The TRAP is set for the expected out of data error. All of the DATA items are read, and then the TRAP is sprung. At this point, P has the line number of the highest DATA line. Let TOTAL = P and use TOTAL in line 160 to set the random number limit. By doing all of this, my daughters don't have to go into the program to change the RND size.

135 – If C$ is a number (that is, less than ASC 65), then there is only one answer, and C$ is ignored.

160-170 – Here is where we RESTORE to random DATA lines, thanks to Atari Basic.

171-177 – This avoids an out of data error if the last DATA line has only one answer. As you see, TRAPs can be used for much more than just avoiding an error crash.

220 – Aha! The "missing" TAB in Atari BASIC really does exist! It's just called POKE 85, avar instead. Memory location 85 is the current cursor column, and you can POKE it with anything that you would use as a TAB argument. Remember that the Atari columns start at 0, but default to 2.



230 – If I OPENed the screen editor in order to avoid ? prompts, how come I put in my own ? (line 220) for this input? Here I felt that the extra space after the ? helps to visually set off the guessed answer.
270 – P(Q) = P. Remember that P is the repeat of the line number, so that the line number of the DATA on which an error was made is now saved in P(Q). When Q = 16 there is a chance that the list of mistakes will scroll when printed, so I go to that list here, just in case.
360-380 – The POKEs center the answer and U$ underlines it.
540 – I like to use CHR$(125) for a screen clear because it reads correctly on the printer. ESC SHIFT CLEAR prints out as a bracket.
600 – For all of you out there who like green screens ...
630 – FOR R = 1 TO the total number of mistakes.
650 – RESTORE the line number of the line on which an error was made.
730 – I like friendly computers.
745 – The two brackets are ESC CTRL 2. That beeps the internal speaker of the computer.
100-1080 – The flashing CORRECT! subroutine. The POKE 85,15 along with the semicolon holds it all together in the same place on the screen. This way you are not row-dependant, as you are with a POSITION.
2000-2330 – Program introduction and old DATA clearout.
2025 – WARN is a flag to warn that data has been changed. The warning is used in line 740.
2040 – Make the screen printing the same color and brightness as the background, so that it is "invisible." Otherwise, there are lots of distractions on the screen during the DATA clearout.
2050-2070 – How many DATA lines are there to clear out? This is the same kind of count-through as in lines 125–150.
2085 – ERASE is the line number to be "erased."
2090 – While all of this is going on, let's have some musical entertainment.
2100-2150 – The people at the Atari technical information WATS line (800-538-8547) supplied this routine. They have been very helpful, answering all kinds of questions. This routine takes information generated by the program and newly printed on the screen, and enters it into the program as part of the program. The new information must be a complete legal line (new line or changed existing line) including the line number. In this example, on line 2110, the ERASE is where the new line is printed. I have used the routine here with just line numbers in order to erase the old DATA lines. You can enter more than one new line by having more PRINT lines (after 2110 and before 2120). Just be careful to not scroll the screen with too many lines. Possible uses for this routine include updating specific data on existing lines. you might want to do this in a budget program to update expense totals and the projected inflation rate. It is used at lines 2200 to 2360 here to enter the new DATA lines as part of the program. This will work with anything that can be expressed as a complete line (see line 2310). And the result, since it is now part of the program, can be saved.
2200-2360 – The data update routine again. Here it takes INPUTs (lines 2250, 2270, and 2290), organizes them into a legal DATA line at line 2310 (with D incrementing the line number), and then adds this new line to the program.

```
0 DIM A$(20),B$(20),C$(20),U$(44),X$(20)
,Y$(1),P(16):BEEP=3000:GOTO 2000
1 DATA 1,IM____,NOT
2 DATA 2,MIS____,WRONG
3 DATA 3,CON____,WITH,TOGETHER
4 DATA 4,SUB____,UNDER
5 DATA 5,SUPER____,OVER
6 DATA 6,PRE____,BEFORE
7 DATA 7,INTER____,BETWEEN,AMONG
8 DATA 8,EX____,OUT OF
9 DATA 9,EX____,OUT OF
10 DATA 10,TRANS____,ACROSS
110 OPEN #1,4,0,"E:"
115 Q=0:M=0:POKE 752,1
120 U$="
        "
125 TRAP 150
130 READ P,A$,B$,C$
135 IF ASC(C$)<65 THEN RESTORE P+1
140 GOTO 130
150 TOTAL=P
160 X=INT(TOTAL*RND(0)+1)
170 RESTORE X
171 IF X<>TOTAL THEN 180
172 TEST=1:GOTO 174
173 TEST=0:RESTORE X
174 TRAP 173
175 IF TEST=1 THEN READ P,A$,B$,C$
176 IF TEST=0 THEN C$="0":READ P,A$,B$
177 TRAP 40000:GOTO 190
180 READ P,A$,B$,C$
190 B1=LEN(B$):C1=LEN(B$)+LEN(C$)+4
200 ? "             HOMEWORK PRACTICE"
205 ? "             _________________"
210 ? :? :? :? :? :POKE 752,0
220 POKE 85,17-LEN(A$)/2:? A$;" ? ";
230 INPUT #1;X$
240 POKE 752,1:? :?
250 IF X$=B$ OR X$=C$ THEN GOSUB 1000:GO
TO 500
260 ? "WRONG......TRY AGAIN."
270 M=M+1:Q=Q+1:P(Q)=P:IF Q>15 THEN ? CH
R$(125):GOTO 600
280 ? :? :POKE 752,0
290 POKE 85,17-(LEN(A$)/2):? A$;" ? ";
300 INPUT #1;X$
310 POKE 752,1
320 IF X$=B$ OR X$=C$ THEN GOSUB 1000:GO
TO 500
330 ? :? :? "NOPE..........THE CORRECT A
NSWER IS":? :?
335 IF ASC(C$)<65 THEN 370
340 POKE 85,19-C1/2:? B$;" or ";C$
360 POKE 85,19-C1/2:? U$(1,C1):GOTO 500
370 POKE 85,19-B1/2:? B$
380 POKE 85,19-B1/2:? U$(1,B1)
500 ? :? :? :?
510 ? "  PRESS [RETURN] FOR ANOTHER PROBLE
M":? "  OR TYPE [L] FOR A LIST OF MISTAKES
.";
520 INPUT #1;Y$
530 IF Y$<>"" AND Y$<>"L" THEN 510
```

Now for adults.

Textwizard™ transforms Atari into a powerfully serious word processor.

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

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

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

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

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

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

Datasoft Inc.™
COMPUTER SOFTWARE

Software for people who aren't easy to please.

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

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

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

```
540 ? CHR$(125)
550 IF Y$="L" THEN 600
560 GOTO 160
600 SETCOLOR 1,14,4:SETCOLOR 2,14,0
610 ? "           LIST OF MISTAKES"
615 ? "           ────────────────"
620 IF M<1 THEN 700
630 FOR R=1 TO Q
650 RESTORE P(R)
651 IF P(R)<>TOTAL THEN 660
652 TEST=1:GOTO 654
653 TEST=0:RESTORE P(R)
654 TRAP 653
655 IF TEST=1 THEN READ P,A$,B$,C$
656 IF TEST=0 THEN C$="0":READ P,A$,B$
657 TRAP 40000:GOTO 670
660 READ P,A$,B$,C$
670 PRINT A$;
680 IF ASC(C$)<65 THEN ? "......";B$:GOT
O 690
685 ? "...";B$;" or ";C$
690 NEXT R
700 PRINT
710 ? "DO YOU WANT MORE PRACTICE ";:INPU
T Y$
720 IF Y$="Y" THEN FOR R=1 TO 16:P(R)=0:
NEXT R:M=0:Q=0:GRAPHICS 0:POKE 752,1:GOT
O 160
730 ? :? "O.K., GOODBYE FOR NOW."
735 IF WARN=0 THEN GOSUB BEEP:GOTO 750
740 ? :? "YOU ADDED NEW PROBLEMS THIS TI
ME.":? "BE SURE TO CSAVE THIS PROGRAM!"
745 ? "}":FOR TIME=1 TO 10:NEXT TIME:? "
}":FOR TIME=1 TO 400:NEXT TIME
750 FOR TIME=1 TO 400:NEXT TIME
997 GRAPHICS 0:POKE 752,0
998 CLOSE #1
999 END
1000 ? :? :? :GOSUB BEEP
1010 FOR FLASH=1 TO 7
1020 POKE 85,15:? "CORRECT !";
1030 FOR TIME=1 TO 8:NEXT TIME
1040 POKE 85,15:? "CORRECT !";
1050 FOR TIME=1 TO 8:NEXT TIME
1060 NEXT FLASH
1070 POKE 85,15:? "CORRECT !"
1080 RETURN
2000 ? CHR$(125)
2005 POSITION 10,2:? " HOMEWORK PRACTICE
 "
2010 POSITION 9,10:? "DO YOU WANT TO ENT
ER":? "        NEW HOMEWORK PROBLEMS ";:I
NPUT Y$
2015 IF Y$<>"Y" AND Y$<>"N" THEN 2010
2020 IF Y$="N" THEN 110
2025 WARN=1
2030 POKE 752,1:? CHR$(125):POSITION 3,1
1:? "PLEASE WAIT WHILE I GET READY....":
FOR TIME=1 TO 250:NEXT TIME
2040 SETCOLOR 1,9,4:POKE 752,0
2050 TRAP 2085
2060 READ P,A$,B$,C$
2065 IF ASC(C$)<65 THEN RESTORE P+1
2070 GOTO 2060
2085 FOR ERASE=1 TO P
2090 SOUND 0,4*ERASE*RND(1),12,8
2100 ? CHR$(125)
2110 ? "↓";ERASE
2120 ? :? :? "CONT"
2130 POSITION 0,0
2140 POKE 842,13:STOP
2150 POKE 842,12
2160 NEXT ERASE
2200 SOUND 0,0,0,0:? CHR$(125):SETCOLOR
1,9,10:A$="":B$="":C$="":D=0
2210 POKE 752,1:POSITION 8,10:? "O.K., I
'M READY FOR YOUR"
2211 ? "         NEW HOMEWORK PROBLEMS.":F
OR TIME=1 TO 300:NEXT TIME:POKE 752,0
2220 ? CHR$(125):D=D+1:IF D>100 THEN 250
0
2230 IF D>1 THEN POSITION 8,21:? "PRESS
 RETURN  IF THE LIST":? "         OF PROBL
EMS IS COMPLETE."
2240 POSITION 14,9:? "PROBLEM #";D:? "
   Please type the  QUESTION ."
2250 POSITION 14,12:INPUT A$:IF A$="" AN
D D=1 THEN 2200
2255 GOSUB BEEP:IF A$="" THEN 2500
2260 ? CHR$(125):POSITION 14,9:? "PROBLE
M #";D:? "   Now type the  ANSWER , plea
se."
2270 POSITION 14,12:INPUT B$:GOSUB BEEP:
IF B$="" THEN 2260
2280 ? CHR$(125):POSITION 6,8:? "PROBLEM
 #";D;" - SECOND ANSWER "
2285 POSITION 6,10:? "If there is no sec
ond ANSWER,":? "         Just press  RETU
RN ."
2290 POSITION 14,13:INPUT C$:GOSUB BEEP
2300 SETCOLOR 1,9,4:? CHR$(125)
2310 ? "↓";D;" DATA ";D;",";A$;",";B$;:I
F C$<>"" THEN ? ",";C$
2320 ? :? :? "CONT"
2330 POSITION 0,0
2340 POKE 842,13:STOP
2350 POKE 842,12
2360 SETCOLOR 1,9,10:GOTO 2220
2500 ? CHR$(125):POKE 752,1:POSITION 4,1
0:? "O.K., I'M READY TO TEST YOU ON":? "
  YOUR NEW HOMEWORK PROBLEMS...."
2510 FOR TIME=1 TO 300:NEXT TIME:A$="":B
$="":C$="":RESTORE :POKE 752,0:GOTO 110
3000 SOUND 0,170*RND(0)+30,10,10:FOR TIM
E=1 TO 20:NEXT TIME:SOUND 0,0,0,0:POKE 7
64,255:RETURN
9999 REM - HOMEWORK PRACTICE
           by BRUCE FRUMKER
```

©

Easy Reading Of The Atari Joystick

Or, Which Way Is Up?

Edward P. McMahon
Potomac, MD

The ATARI joystick seems to be, at first glance, strangely encoded (Fig. 1) and the most common way of determining which way the stick is pushed in a BASIC program is by coding a series of "IF - THEN - GOTO" statements which test each of the apparently unrelated decimal values which may be in variable STICK. A deeper look at ATARI's scheme reveals the logic of the STICK(I) values and points the way to efficient decoding of a joystick's position.

First, let us label each joystick position with the binary values of each deflection assignment as shown in Figure 2. Second, remember that the implementation of switches such as the trigger (or fire) button on joysticks and paddles uses the value "0" to indicate that the button is pushed and a value "1" to mean that the button is not pushed.

Now, it becomes apparent that the joystick is made from four switches, each of which controls one bit in the STICK(I) value, and is set to "0" by moving the stick in one of the cardinal directions.

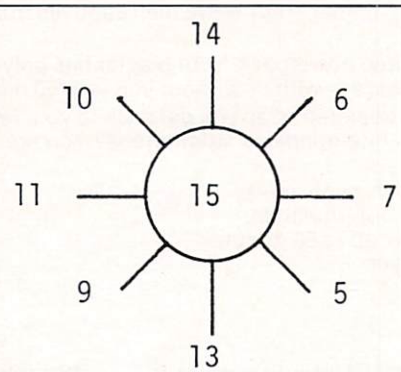


Figure 1.

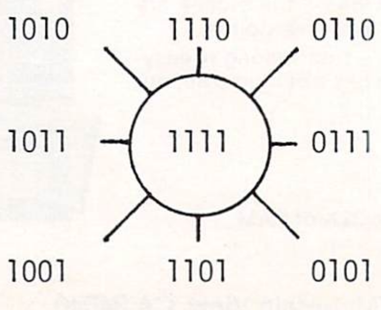


Figure 2.

Assigning bit numbers from right to left (bits 0 to 3) we see that bit 0 controls "forward" or "down", bit 1 controls "back" or "up," bit 2 for "left," and bit 3 for "right." If a control bit is zero, the switch has been pushed in that direction. Diagonal stick movement is encoded by two bits being pushed to zero.

I wanted values for STICK(I) in my programs to be usable in "ON value GOTO" statements, and, for strictly personal preference, wanted a value of "1" to mean that the stick was not deflected. Two values are needed — one for X deflection, one for Y deflection. The BASIC code in Program 1 (lines 10 through 40) produces these values. The rest of the code demonstrates the application of this stick decoding routine.

To be even faster while conforming to my own particular (not peculiar, I hope) conventions, I coded an assembly language routine which is tucked up near the top of page 6 of memory, that memory page thoughtfully left untouched by ATARI for use by us hackers. (Everybody else seems to put their code at the beginning of page 6, so I put this code almost at the end — \$06E4 to \$06FE.)

Note that there are two entry points to the code — one for the X value (\$06E4 or decimal 1764) and one for the Y value (\$06F5 or decimal 1781). This routine can be accessed by a BASIC program from the statements

X = USR (1764, STICK(I))
Y = USR (1781, STICK(I))

for whichever stick you're using (I = 0,1,2 or 3). See Program 1-a.

Program 2 is the assembled code for this routine which can be used to create your own binary program. Program 3 is a BASIC program to POKE the assembly language DATA where it belongs, if you would rather take that approach. Once you enter this code into page 6 of RAM, it should stay there undisturbed by anything but a deliberate write-over by you or a system reinitialization (power off-on).

```
10 P=STICK(0)
20 X=USR(1764,P)
30 Y=USR(1781,P)
50 ON X GOTO 80,60,70
60 ? "LEFT ";:GOTO 80
70 ? "RIGHT ";:GOTO 80
80 ON Y GOTO 110,90,100
90 ? "DOWN":GOTO 10
100 ? "UP":GOTO 10
110 IF X<>1 THEN ? " "
120 GOTO 10
```

Program 1-a.

```
10 P=STICK(0)
20 Q=INT(P/4)
30 X=4-Q
40 Y=4-P+Q*4
50 ON X GOTO 80,60,70
60 ? "LEFT ";:GOTO 80
70 ? "RIGHT ";:GOTO 80
80 ON Y GOTO 110,90,100
90 ? "DOWN":GOTO 10
100 ? "UP":GOTO 10
110 IF X<>1 THEN ? " "
120 GOTO 10
```

Program 1.

```
10 REM POKES STICK READER INTO LOCATIONS $06E4 TO $06FE
20 FOR I= 1764 TO 1790: READ X: POKE I,X: NEXT I :END
30 DATA 104,104,133,213,104,41,12,74,74,73,3,24,105,1
40 DATA 133,212,96,104,104,133,213,104,41,3,76,237,6
```

Program 3.

Poem Writer

Frank Roberts
Ft. Wayne, IN

Want to improve your love life? Enlist your Atari computer to write poems for that "special person." Here is a program designed to help you write Haiku poetry – a Japanese form of short verse having seventeen syllables and, generally, describing some observed natural phenomenon or personal gestalt. Haiku form is traditionally very structured, and the program can be modified to fit such rules, but this one was written just for the fun and love of it. Nevertheless, some remarkable verses can be "composed" with it. This program will turn out enough birthday, anniversary, and Valentine poems for a lifetime – and with no more effort than a few minutes of typing!

The program utilizes the XIO command to delete a temporary disk file upon user termination. LINE 50 opens the temporary file for storage of user input. LINES 600,660 store sector and bite

```
                   10 ;
                   20 ;INTERPRETS STICK(I) AND RETURNS
                   30 ;FOR X: 1=NOTHING
                   40 ;       2=LEFT
                   50 ;       3=RIGHT
                   60 ;FOR Y: 1=NOTHING
                   70 ;       2=DOWN  (PUSH STICK)
                   80 ;       3=UP    (PULL STICK)
                   85 ;
                   86 ; X=USR(1764,STICK(I))
                   87 ; Y=USR(1781,STICK(I))
                   90 ;
0000               0100        *=     $06E4
00D4               0110 RESLT  =      $D4
00D5               0115 RESLTH =      $D5
06E4 68            0120 XSTK   PLA                 ;THROW AWAY # ARG'S IN STACK
06E5 68            0130        PLA                 ;STICK HAS NO HI-ORDER BITS
06E6 85D5          0135        STA    RESLTH       ;STUFF ZERO IN HI-RETURN
06E8 68            0140        PLA                 ;THIS IS STICK(I)
06E9 290C          0150        AND    #$0C         ;GET BITS 2 AND 3
06EB 4A            0160        LSR    A            ;SHIFT 'EM LOW
06EC 4A            0170        LSR    A            ; TO BITS 0 AND 1
06ED 4903          0180 FIN    EOR    #$03         ;INVERT BITS
06EF 18            0190        CLC                 ;CLEAR CARRY BEFORE ADD
06F0 6901          0200        ADC    #$01         ;ADD 1 SO WE CAN "ON . GOTO .."
06F2 85D4          0210        STA    RESLT        ;PUT ANSWER IN RETURN LOCATION
06F4 60            0220        RTS                 ;GO HOME
06F5 68            0230 YSTK   PLA                 ;ENTRY FOR Y-STICK, AS ABOVE
06F6 68            0240        PLA                 ;NO HI-ORDER DATA
06F7 85D5          0245        STA    RESLTH       ;ZERO HI-ORDER RETURN DATA
06F9 68            0250        PLA                 ;THIS IS STICK AGAIN
06FA 2903          0260        AND    #$03         ;GET BITS 0 AND 1
06FC 4CED06        0270        JMP    FIN          ;GOTO CODING
06FF               0280        .END
```

Program 2.

©

numbers of the filed data into arrays SEC and BIT for later random selection. LOOP 700,730 selects sector and bite numbers at random and stores them into arrays A and B. LINES 800,880 choose string data from the file and print user input phrases in groups of four (traditionally, Haiku is a three line poem, and, if desired, this can be accomplished by changing LINE 870 to IF K/3=INT(K/3) THEN LPRINT ;delete LINE 850, and add 790 TRAP 890). LINE 1090 deletes the temporary file without going to DOS.

A word of WARNING! A few weeks ago my wife took a vacation ... well, not being one to enjoy letter-writing, I sent her several pages of Haiku "composed" with my Atari. She has since returned. Now I'm too tired to write more programs. C'est l'amour!

Haiku Poems

By Frank Roberts

Fly away and high
Idly, the ship glides by
I might enjoy tonight's moon
Summer and swimming and smiles
*****

Companions
A baby wren sings his first song
Mellons are ripening
Fly away and high
*****

A voiceless flower speaks
Hidden among the leaves
Companions
Summer and swimming and smiles
*****

Wind
Snow
Hidden in the leaves
Escape!
*****

Do not bother about mountains
Escape!
Summer and swimming and smiles
Where are the flowers
*****

Crickets
Ice and water are forgotten
A voiceless flower speaks
Crickets

Sayanara, Frank-San

```
1 REM ********************************
2 REM
3 REM HAIKU WRITING
4 REM by Frank Roberts
5 REM
6 REM ********************************
7 REM
9 REM
50 DIM A$(100),N$(30),R$(10),A(100),B(100),
   SEC(100),BIT(100)
60 REM
70 GRAPHICS 0:POSITION 5,5
100 PRINT "WHAT IS YOUR NAME, SIR "
110 INPUT N$
115 ? :? :?
120 ? :? "THANK YOU, ";N$
121 ? "DO YOU WANT EXPLANATION ?";
130 INPUT R$
140 IF R$(1,1)="N" THEN 400
150 GRAPHICS 0:POSITION 5,4
200 PRINT "I will help you write Haiku"
201 PRINT "poems.  If you will type in a"
202 PRINT "list of lovely phrases, I will"
203 PRINT "mix them up and put them back"
204 PRINT "together in my own Oriental way."
205 PRINT "  Some of them may not make much"
206 PRINT "sense, but that is probably"
207 PRINT "because language always looses"
208 PRINT "something in translation...."
209 PRINT "RIGHT?  some of them, however,"
210 PRINT "may be quite interesting."
300 ? :? :?
400 ? :? "ARE YOU READY, ";N$;" ";
410 INPUT R$:IF R$(1,1)<>"Y" THEN 400
415 GRAPHICS 0:POSITION 5,5
420 ? "AH......SO.:?"
430 ? :? "HOW MANY POEMS TO YOU WANT (MORE THAN
   3, PLEASE) ";
440 INPUT R
450 P=R*2
460 REM
500 OPEN #1,8,0,"D:HFILE"
570 ? :? "LIST ";P;" LOVELY PHRASES:"
600 FOR K=1 TO P
610 NOTE #1,Y,Z
620 SEC(K)=Y:BIT(K)=Z
630 PRINT K,
640 INPUT A$
650 PRINT #1;A$
660 NEXT K
670 CLOSE #1
675 REM
680 LPRINT CHR$(27);CHR$(14)
681 LPRINT "              HAIKU POEMS"
683 LPRINT CHR$(27);CHR$(15)
684 LPRINT "                                  BY ";N$
685 LPRINT :LPRINT :LPRINT
700 FOR K=1 TO P*2
710 X=INT(P*RND(0)+1)
720 A(K)=SEC(X):B(K)=BIT(X)
730 NEXT K
740 REM
760 REM
800 OPEN #1,4,0,"D:HFILE"
810 FOR K=1 TO P*2
820 Y=A(K):Z=B(K)
830 POINT #1,Y,Z
840 INPUT #1,A$
850 IF K/2=INT(K/2) THEN LPRINT " ","    ";
   A$:GOTO 870
860 LPRINT " ",A$:GOTO 880
870 IF K/4=INT(K/4) THEN LPRINT
880 NEXT K
890 CLOSE #1
900 GRAPHICS 0:POSITION 5,5
1000 ? "DO YOU WANT MORE FROM THE SAME PHRASES ";
1010 INPUT R$:IF R$(1,1)="N" THEN 1030
1020 GOTO 700
1030 ? :? "DO YOU WANT TO MAKE NEW PHRASES ";
1040 INPUT R$:IF R$(1,1)="N" THEN 1060
1050 GOTO 430
1060 FOR K=1 TO 5:LPRINT :NEXT K
1061 REM
1062 FOR K=1 TO LEN(N$)
1063 IF N$(K,K)<>" " THEN 1065
1064 N$=N$(1,K-1):GOTO 1066
1065 NEXT K
1066 REM
1070 LPRINT "              SAYANARA, ";N$;"-SAN"
1080 FOR K=1 TO 10:LPRINT :NEXT K
1090 XIO 33,#1,0,0,"D:HFILE"
```

©

Supercube Update

Mike Kinnamon
Perry, OK

Here is an updated version of Steve Steinberg's Supercube program which appeared in **COMPUTE!** #11.

Here is a list of the keys and their functions.

Function	Key
1. Automatic drawing mode	Logo key
2. Change cube color	Spacebar then number
3. Cube size	Insert key
4. Background color	Escape key
5. Background shade	Delete key
6. Clear screen	Clear key
7. Save picture to disk	R key
9. Start and stop erasing cursor	E key

```
1 DIM A$(20),B$(20),C$(4),PR$(1),PR1$(80
)
2 SHADE=2
3 HUE=0
4 X=80:Y=40
5 CV=1
6 FX=1
10 GRAPHICS 0:? "           3-D DRAWING
           "
12 ? "      USE JOYSTICK #1 TO PLAY"
14 ? " YOU MAY PLACE A 3-D CUBE ON THE "
:? "SCREEN BY MOVING THE CURSOR TO THE "

16 ? "APPROPRIATE SPOT AND PRESSING THE
":? "RED BUTTON THEREBY CREATING A PLEAS
ANT";:? "DESIGN."
18 ? :? " YOU HAVE SEVERAL OPTIONS TO CH
OOSE":? "FROM:"
20 ? "1.CUBE SIZE-FROM 0 TO ?":? "CHANGE
 THIS BY PRESSING THE INSERT KEY";
22 ? "THEN ANSWER THE SIZE QUESTION WITH
 A":? "NUMBER,AND PRESS RETURN"
23 ? "numbers larger than 10 may not wor
k"
24 ? "2.CUBE COLORS-PRESS THE SPACEBAR T
HEN":? "CHOOSE FROM THE 10 COLORS AND PR
ESS":? "RETURN."
26 ? "3.BACKGROUND COLOR AND SHADE-ESC K
EY":? "CONTROLS COLOR  DELETE KEY CONTRO
LS":? "SHADE."
28 ? "4.CLEAR KEY CLEARS THE SCREEN."
30 TRAP 30:? "ENTER CUBE SIZE NOW";:INPU
T SQ:TRAP 0
40 GRAPHICS 7
45 SETCOLOR 4,0,4
70 GOSUB 600
80 LOCATE X,Y,Z:ZZ=5:IF Z<>2 THEN ZZ=2
81 IF CV/2=INT(CV/2) THEN Z=4
82 IF PEEK(764)=39 THEN FX=FX+1:POKE 764
,255
84 IF FX/2=INT(FX/2) THEN GOSUB 700
85 IF STRIG(0)<>0 THEN GOSUB 500:GOTO 14
0
120 POKE 77,0
130 IF STRIG(0)=0 THEN GOSUB 700
140 GOSUB 1000
150 X=X+XDIF:Y=Y+YDIF
200 IF X>143 THEN X=0:GOTO 300
210 IF Y>78 THEN Y=7:GOTO 300
300 IF X<0 THEN X=143:GOTO 400
310 IF Y<7 THEN Y=78
400 GOTO 80
500 COLOR ZZ:PLOT X,Y
500 COLOR ZZ:PLOT X,Y
501 IF PEEK(764)=55 THEN POKE 764,255:?
">ENTER NEW DIMENSION FOR CUBE";:INPUT S
Q:? CHR$(253):GOSUB 3000
502 A=PEEK(764):IF A=33 THEN POKE 764,25
5:? CHR$(253):GOSUB 600
503 IF A=54 THEN GRAPHICS 7:SETCOLOR 1,C
,12:SETCOLOR 2,C,6:SETCOLOR 0,C,4:SETCOL
OR 4,HUE,SHADE:POKE 764,255:? CHR$(253):
GOSUB 3000
504 IF A=52 THEN SHADE=SHADE+2:POKE 764,
255:IF SHADE>14 THEN SHADE=0:? CHR$(253)
:GOSUB 3000
504 IF A=52 THEN SHADE=SHADE+2:POKE 764,
255:IF SHADE>14 THEN SHADE=0:? CHR$(253)
:GOSUB 3000
505 IF A=28 THEN GOSUB 2000:POKE 764,255
:HUE=HUE+1:GH=1:IF HUE>15 THEN HUE=0:? C
HR$(253)
506 IF A=62 THEN GOSUB 4000:GOSUB 3000:G
OTO 80
507 IF A=40 THEN GOSUB 5000:GOSUB 3000:C
OLOR 80
508 IF A=42 THEN CV=CV+1:POKE 764,255:?
CHR$(253):GOSUB 3000
509 IF A=10 THEN GOSUB 7000
510 SETCOLOR 4,HUE,SHADE
549 COLOR Z:PLOT X,Y
550 RETURN
600 POKE 764,255:? ">PRESS A NUMBER TO C
HANGE COLORS":? "1-GOLD 2-ORANGE 3-RED 4
-PINK 5-PURPLE":? "6-BLUE 7-GRAY";
601 ? " 8-VIOLET 9-GREEN ":? "0-TURQUOIS
E";
602 IF PEEK(764)=255 THEN 602
603 A=PEEK(764)
604 IF A=31 THEN C=1:GOTO 650
605 IF A=30 THEN C=2:GOTO 650
606 IF A=26 THEN C=3:GOTO 650
607 IF A=24 THEN C=4:GOTO 650
608 IF A=29 THEN C=6:GOTO 650
```

```
609 IF A=27 THEN C=7:GOTO 650
610 IF A=51 THEN C=0:GOTO 650
611 IF A=48 THEN C=12:GOTO 650
612 IF A=50 THEN C=10:GOTO 650
613 IF A=53 THEN C=5:GOTO 650
620 POKE 764,255:GOTO 602
625 ? "}"
650 SETCOLOR 1,C,12:SETCOLOR 2,C,6:SETCO
LOR 0,C,4:? "}"
651 GOSUB 3000
670 RETURN
700 REM
701 IF GH=1 THEN GH=0:GOSUB 3000
710 TRAP 1002
720 COLOR 1
730 FOR I=0 TO SQ
740 PLOT X,Y+I:DRAWTO X+SQ,Y+I
750 NEXT I
760 COLOR 2
770 FOR I=1 TO INT(3*SQ)/5
780 PLOT X+I,Y-I:DRAWTO X+I+SQ,Y-I
790 NEXT I
800 COLOR 3
810 FOR I=1 TO INT(3*SQ)/5
820 PLOT X+SQ+I,Y-I:DRAWTO X+SQ+I,Y+SQ-I
+1
830 NEXT I
840 IF STRIG(0)<>0 THEN X=X-1
850 RETURN
1000 WHAT=STICK(0):XDIF=0:YDIF=0
1002 IF FX/2=INT(FX/2) THEN X=INT(RND(0)
*140):Y=INT(RND(0)*70):SQ=INT(RND(0)*20)
:POKE 77,0:GOTO 1200
1100 IF WHAT=15 THEN RETURN
1110 IF WHAT=14 THEN YDIF=-1:RETURN
1120 IF WHAT=13 THEN YDIF=1:RETURN
1130 IF WHAT=11 THEN XDIF=-1:RETURN
1140 IF WHAT=10 THEN XDIF=-1:YDIF=-1:RET
URN
1150 IF WHAT=9 THEN XDIF=-1:YDIF=1:RETUR
N
1160 IF WHAT=7 THEN XDIF=1:RETURN
1170 IF WHAT=6 THEN XDIF=1:YDIF=-1:RETUR
N
1180 IF WHAT=5 THEN XDIF=1:YDIF=1:RETURN

1200 IF RND(0)>0.9 THEN C=RND(0)*15:SETC
OLOR 1,C,12:SETCOLOR 2,C,6:SETCOLOR 0,C,
4:HUE=RND(0)*15:SHADE=RND(0)*15
1201 IF RND(0)>0.95 THEN SQ=44
1202 RETURN
2000 ? "}BACKGROUND HUES ARE:ORANGE ":?
"GRAY-GOLD-ORANGE-RED-PINK-PURPLE":? "BL
UE-LIGHT BLUE-TUROUOISE-GREEN/BLUE"
2002 ? "GREEN-YELLOW/GREEN-ORANGE/GREEN"
;
2004 RETURN
3000 POKE 752,2:POKE 82,0:? "}S-SAVE PIC
 TO DISK R-RETRIEVE PIC":? "DELETE-BCKGN
D SHADE CLEAR-CLEAR SCREEN"
3001 ? "SPACEBAR-CUBE COLORS INSERT-CUBE
 SIZE":? "ESC-BCKGND COLORS E-START/STOP
 ERASE";
3002 RETURN
4000 POKE 764,255:? "}Name of picture to
 save";:INPUT A$:B$="D:":B$(LEN(B$)+1)=A
$:C$=".PIC":B$(LEN(B$)+1)=C$
4001 ? "Insert proper disk and hit any k
ey ":? "It will take about 4 1/2 minutes
 ":GOSUB 6000
4002 OPEN #1,8,0,B$:POKE 559,0:? CHR$(25
3);
4003 FOR YY=0 TO 79:FOR XX=0 TO 159
4004 LOCATE XX,YY,ZZZ:IF ZZZ<>0 THEN PUT
 #1,XX:PUT #1,YY:PUT #1,ZZZ
4005 NEXT XX:NEXT YY:POKE 559,34:CLOSE #
1:FOR XX=1 TO 5:? CHR$(253);:NEXT XX
4006 POKE 764,255:RETURN
5000 POKE 764,255:? "}Name of picture to
 get";:INPUT A$:B$="D:":B$(LEN(B$)+1)=A$
:C$=".PIC":B$(LEN(B$)+1)=C$
5001 TRAP 5006
5002 OPEN #1,4,0,B$:POKE 559,0:? CHR$(25
3);
5003 FOR C=1 TO 2 STEP 0
5004 GET #1,XX:GET #1,YY:GET #1,ZZZ:COLO
R ZZZ:PLOT XX,YY:NEXT C
5006 CLOSE #1:POKE 559,34:POKE 764,255:F
OR XX=1 TO 5:? CHR$(253);:NEXT XX
5008 RETURN
6000 OPEN #1,4,0,"K:":GET #1,R:CLOSE #1:
RETURN
7000 POKE 764,255:LPRINT CHR$(27);CHR$(6
5);CHR$(6)
7001 FOR XX=0 TO 159:FOR YY=79 TO 0 STEP
 -1
7003 A=PEEK(764):IF A=33 THEN XX=159
7004 LOCATE XX,YY,ZZZ
7005 IF ZZZ=0 THEN PR$=" "
7006 IF ZZZ=1 THEN PR$="#"
7007 IF ZZZ=2 THEN PR$="+"
7008 IF ZZZ=3 THEN PR$="*"
7009 IF ZZZ=4 THEN PR$="="
7010 IF ZZZ=5 THEN PR$="%"
7015 PR1$(LEN(PR1$)+1)=PR$
7020 NEXT YY:LPRINT PR1$:PR1$="":NEXT XX

7025 POKE 764,255
7027 LPRINT CHR$(27);CHR$(50);
7028 FOR CG=1 TO 32:LPRINT :NEXT CG
7040 RETURN
```

©

Atari Sound Utility

Jimmy Mork
Winger, MN

Have you ever wished you could get sound out of that little speaker hidden somewhere in your Atari? The one that is responsible for the buzz that occurs in the cassette input/output routines. Well, I think I have a routine that just may be able to add those little clicks and buzzes, and create some pretty good sound effects.

First let us try a USR call that will jump right into the middle of one of those routines ... try this:

```
U=USR(61530)
```

Does that sound familiar? If you have problems stopping it, try the SYSTEM RESET key.

What you just USRed to was the routine in the operating system that gives you the buzz in the CLOAD command.

How about that little click you hear when you push the SYSTEM RESET key? That is done in a little simpler way! The RAM location for the speaker is 53279 (D01F). Sound kind of familiar? It is also the location of the console switches. Which means that POKING to this location will activate the speaker, and PEEKing into it will give you console switch status.

The click you hear when you press the SYSTEM RESET can be duplicated as follows:

```
POKE 53279,0
```

So far, the use of the speaker to create sound effects has yielded little value. There is obviously a need to dig deeper. Here is an assembler routine similar to that at ROM location 61530.

```
0100            *=$600
0110  SPKR=$    D01F
0120            PLA
0130            LDX    #$FF
0140  MAIN      LDA    #$FF
0150            STA    SPKR
0160            LDA    #$00
0170            LDY    #$F0
0180  LOOP1     DEY
0190            BNE    LOOP1
0200            STA    SPKR
0210            LDY    #$F0
0220  LOOP2     DEY
0230            BNE    LOOP2
0240            DEX
0250            BNE    MAIN
0260            RTS
```

To load the assembler routine, type in the following program and RUN it:

```
100 DATA 104, 162, 255, 169, 255, 141, 31, 208, 169,
  0, 160, 240, 136, 208, 253, 141, 31, 208, 160, 240, 136,
  208, 253, 202, 208, 233, 96
110 RESTORE: FOR A=0 TO 26: READ H: POKE
  1536+A,H: NEXT A
```

Now that you have the subroutine loaded into memory do a:

```
U=USR(1536)
```

I suppose by now you are saying, "OK great, but who wants a game that sounds like CLOAD?" If you don't want it to sound like that, all you have to do is simply rearrange the machine language subroutine. There are two variables you can change that will change the pitch of the sound: the "LDY's" that set the number of iterations to loops "LOOP1" and "LOOP2" in the subroutine. Thus, by changing locations 1547 and 1555, you will have changed the pitch.

```
POKE 1547,120
U=USR(1536)
```

If you tryed the two instructions above, you should have heard a higher pitch than the one before. If the sounds are too long for you, all you have to do to change the length is change RAM location 1538 (the LDX). POKEing 1538 to 10 will give you a short chirp.

If you want to do some experimenting with different pitches and lengths, add the next four lines to your program:

```
120 PRINT "LENGTH, PF1, PF2 ";: INPUT
  L,PF1,PF2
130 POKE 1538,L: POKE 1547,PF1: POKE 1555,PF2
140 FOR A:=1 TO 100:NEXT A: REM This line will
  delay the routine to prevent 'key click'/sound
  confusion.
150 U=USR(1536):GOTO 120
```

©

Blockade For The Atari

Douglas Pinho
Valley Cottage, NY

Surround (or Blockade) was a popular arcade game in the early days of video games. The format of the game is not complex in itself, but it still is enjoyable and challenging. The object of the game is to build walls to trap the opposing player and force him to collide with: 1) his own walls, 2) the opposing player's walls, 3) the boundaries of the playfield. When this occurs, the player who did not crash receives a point. Upon every collision, the walls of the player who crashed will blink. The screen is then cleared and the game continues.

The first player to reach nine points is the winner. To start the next game just press the fire button. To play, plug joysticks into the middle two joystick ports (sticks 1 and 2).

Program Description And Explanation

Lines 1–2 set up the title display. Line 5 sets up a mixed graphics mode with 1 line of GR. 1 followed by 1 line of GR. 2 and 44 lines of GR. 5. START calculates the address of the display list in memory. This pointer is needed since the location of the display list is dependent upon the amount of memory installed in the Atari. The two POKES then place instructions for the desired graphic modes at the appropriate memory locations. Line 10 initializes the variables X and Y, the starting locations of player 1, and S and T which give the location of player 2. Variables X1 and Y1 and S1 and T1 are the increment or decrement values for plotting the walls on the screen. F is a flag to determine whether there was a simultaneous collision between the two players. H1 and B1 are used to keep score. Line 12 plots the boundaries of the playing field in blue. Poking memory location 87 (current screen mode) with 5 directs the computer to plot in GR. mode 5. This is only needed in a mixed graphics mode. Line 14 goes to a subroutine at line 300 which prints the score in GR. 2 characters. Line 15 checks for the end of the game.

Lines 20–120 contain the main game loop. Lines 25 to 43 check for joystick movement and assign the move variable (X1, Y1, S1, T1), and a value for P and L. One of the nice features of Atari Basic is that you can use a variable as a GOSUB address. This feature is used in line 50 to branch to different subroutines depending upon the value of P (player 1) and L (player 2). Note that in line 23, you must POKE 5 into memory location 87 again because it was changed during subroutine 300 (line 14). Lines 150 to 185 first check for a collision. If there is none, it plots the new block. A collision is found by locating the next position in front of the plotted block and finding its color. If the color is 0 (which is the background default color), it continues and plots the next block. If it is any other color, there is a collision. If the first player has collided, the program branches to line 201 to check for a simultaneous collision by the other player. Flag F is set if a simultaneous collision is found. Lines 210-220 update the score and blink the losing player's walls. Subroutine 300 prints the score at the top of the screen in GR. 2 characters. Subroutine 350 blinks the colors of the colliding player's walls. Lines 400-410 check if you want to start a new game (prints in GR. 1 characters).

If you haven't played "Blockade" before, grab a friend and try it. It requires quick decisions and good strategy. You'll like it.

```
1 GRAPHICS 2+16:SETCOLOR 4,5,5:POSITION
6,5:? #6;"BLOCKADE"
2 FOR D1=1 TO 6:FOR E1=0 TO 89:SOUND 1,E
1,10,10:NEXT E1:NEXT D1:SOUND 1,0,0,0
5 GRAPHICS 5+16:START=PEEK(560)+PEEK(561
)*256+4:POKE START-1,71:POKE START+2,6
10 X=13:Y=23:X1=1:Y1=1:S=66:T=23:S1=-1:T
1=1:P=160:L=170:F=0
12 POKE 87,5:COLOR 3:PLOT 0,3:DRAWTO 0,4
6:DRAWTO 78,46:DRAWTO 78,3:DRAWTO 0,3
14 GOSUB 300
15 IF H1=9 OR B1=9 THEN GOTO 400
20 B=STICK(1):H=STICK(2)
21 SOUND 3,200,10,15
23 POKE 87,5
25 IF B=14 THEN Y1=-1:P=150
27 IF H=14 THEN T1=-1:L=180
30 IF B=13 THEN Y1=1:P=150
32 IF H=13 THEN T1=1:L=180
35 IF B=7 THEN X1=1:P=160
37 IF H=7 THEN S1=1:L=170
40 IF B=11 THEN X1=-1:P=160
43 IF H=11 THEN S1=-1:L=170
44 SOUND 3,150,10,15
50 GOSUB P:GOSUB L
120 GOTO 20
150 Y=Y+Y1:COLOR 1:LOCATE X,Y,Z:IF Z<>0
THEN GOTO 201
155 PLOT X,Y:RETURN
160 X=X+X1:COLOR 1:LOCATE X,Y,Z:IF Z<>0
THEN GOTO 201
165 PLOT X,Y:RETURN
170 S=S+S1:COLOR 2:LOCATE S,T,U:IF U<>0
THEN GOTO 220
175 PLOT S,T:RETURN
180 T=T+T1:COLOR 2:LOCATE S,T,U:IF U<>0
THEN GOTO 220
185 PLOT S,T:RETURN
201 IF L=170 THEN S=S+S1:POSITION S,T:LO
CATE S,T,U:IF U<>0 THEN F=1
202 IF L=180 THEN T=T+T1:POSITION S,T:LO
CATE S,T,U:IF U<>0 THEN F=1
203 GOTO 210
210 SOUND 3,0,0,0:SOUND 1,100,14,14:FOR
W=1 TO 300:NEXT W:B1=B1+1:GOSUB 300:Q1=0
```

```
:GOSUB 350:SOUND 1,0,0,0
211 IF F=1 THEN GOTO 220
212 GOTO 5
220 SOUND 3,0,0,0:SOUND 2,150,12,14:FOR
W=1 TO 300:NEXT W:H1=H1+1:GOSUB 300:Q1=1
:GOSUB 350:SOUND 2,0,0,0:GOTO 5
300 POKE 87,2:POSITION 5,0:? #6;H1:POSIT
ION 15,0:? #6;CHR$(B1+16):RETURN
350 FOR P1=1 TO 7:FOR U1=1 TO 40:NEXT U1
:SETCOLOR Q1,9,4:FOR G1=1 TO 40:NEXT G1:
SETCOLOR Q1,4,6:NEXT P1:RETURN
400 POKE 87,1:POSITION 0,1:? #6;"PRESS f
ire TO BEGIN!"
405 IF STRIG(1)=0 OR STRIG(2)=0 THEN H1=
0:B1=0:GOTO 5
410 GOTO 405
```

©

Define A Line On The Atari

Wayne Kinzebach
Federal Way, WA

As we all know by now, the Atari has the capability of only diplaying one color at two luminance levels in Graphics mode 8. The background color may be any one of 128 color combinations, but the Graphics Point color must be the same as the background. Only the luminance can be different. The background is set by loading data into Setcolor Register 2, while the Graphics Point luminance is set using Setcolor Register 1. For the entire TV frame, you're stuck with one color at two luminance levels.

What if you could reload Setcolor Registers 2 and 1 at the beginning of each TV horizontal scanning line? The small program shown allows you to do this. Thus you can display all 128 color variations at once. Let's go through the program:

Line 10010: This simply gets into Graphics mode 8 and sets the border color. You can choose any color you want. Your main program would follow this line.

Line 10020: Obviously if you are going to define a background color and a luminance for each TV line, there must be a block set aside somewhere in memory that contains the necessary data. This block will be called COLUMN. COLUML and COLUMH are the low and high address pointers that point to the beginning of this block. Since there are 192 lines and 2 bytes are needed for each line, it takes 384 bytes of memory to store the data.

Lines 10030 and 10040: These lines load the machine language program contained in lines 10060 and 10070. The program is loaded into an area in memory that does not steal any memory from your BASIC programs.

Line 10050: This executes the machine language program.

Lines 10060 and 10070: This is the machine language program.

As an example, if you wanted your COLUM block to start at location 3840 in memory, COLUML would be 0 while COLUMH would be 15(0+15*256=3840). At location 3840 you would store the background color for line 0. At 3841 the Graphics Point luminance for line 0 is stored. At 3842 and 3843 you would store the values for line 1 and so on. The background color is a number between 0 and 255. The four most significant bits select the color, while the rest of the bits (except for the least significant bit, which is not used) specify the luminance. The Graphics Point luminance is set in the same manner as the background luminance.

The machine language program simply reloads the color registers of the ANTIC Graphics Processor chip at the beginning of each TV line. It is shown below. To get out of the program, hit System Reset.

```
PLA
SEI              Disable interrupt
LDX $CD          Save COLUMH
STA $D40A        Halt CPU until start of next line
STA $D018        Store background color in Color
                 Register 2
INY
LDA $(CC),Y      Load Graphics Point luminance
STA $D017        Store in Color Register 1
INY
BNE $02          Test to see whether COLUMH should
                 be inc.
INC $CD          Increment COLUMH
LDA $D40B        Load Vertical Line counter
CMP #$10         Test for line 16
BPL $04          Branch if greater than or equal to 16
LDY #0
STX $CD          Reload COLUMH
LDA $(CC),Y      Load background color
CLC
BCC $DF          Branch to CPU halt
```

To give you a good idea of what the program can do, type in the following lines and enter 0 for COLUML and 250 for COLUMH.

```
10012 COLOR 1
10015 FOR Y=0 TO 191
10018 PLOT 150,Y:DRAWTO 170,Y:NEXT Y
```

©

OSI **OSI**

AARDVARK NOW MEANS BUSINESS!

WORD PROCESSING THE EASY WAY— WITH MAXI-PROS

This is a line-oriented word processor designed for the office that doesn't want to send every new girl out for training in how to type a letter.

It has automatic right and left margin justification and lets you vary the width and margins during printing. It has automatic pagination and automatic page numbering. It will print any text single, double or triple spaced and has text centering commands. It will make any number of multiple copies or chain files together to print an entire disk of data at one time.

MAXI-PROS has both global and line edit capability and the polled keyboard versions contain a corrected keyboard routine that make the OSI keyboard decode as a standard typewriter keyboard.

MAXI-PROS also has sophisticated file capabilities. It can access a file for names and addresses, stop for inputs, and print form letters. It has file merging capabilities so that it can store and combine paragraphs and pages in any order.

Best of all, it is in BASIC (0S65D 51/4" or 8" disk) so that it can be easily adapted to any printer or printing job and so that it can be sold for a measly price.

MAXI-PROS — $39.95

THE EDSON PACK ALL MACHINE CODE GAMES FOR THE 8K C1P

INTERCEPTOR —You man a fast interceptor protecting your cities from Hordes of Yukky Invaders. A pair of automatic cannon help out, but the action speeds up with each incoming wave. It's action, action everywhere. Lots of excitement! $14.95

MONSTER MAZE — An Arcade style action game where you run a maze devouring monsters as you go. If one sees you first, you become lunch meat. Easy enough for the kids to learn, and challenging enough to keep daddy happy. $12.95

COLLIDE — Fast-paced lane-switching excitement as you pick up points avoiding the jam car. If you succeed, we'll add more cars. The assembler code provides fast graphics and smooth action. $9.95

SPECIAL DEAL—THE ENTIRE EDSON PACK— ALL THREE GAMES FOR $29.95

THE AARDVARK JOURNAL

FOR OSI USERS — This is a bi-monthly tutorial journal running only articles about OSI systems. Every issue contains programs customized for OSI, tutorials on how to use and modify the system, and reviews of OSI related products. In the last two years we have run articles like these!

1) A tutorial on Machine Code for BASIC programmers.

2) Complete listings of two word processors for BASIC IN ROM machines.

3) Moving the Directory off track 12.

4) Listings for 20 game programs for the OSI.

5) How to write high speed BASIC — and lots more —

Vol. 1 (1980) 6 back issues - $9.00

Vol. 2 (1981) 2 back issues and subscription for 4 additional issues - $9.00.

ACCOUNTS RECEIVABLE — This program will handle up to 420 open accounts. It will age accounts, print invoices (including payment reminders) and give account totals. It can add automatic interest charges and warnings on late accounts, and can automatically provide and calculate volume discounts.

24K and 0S65D required, dual disks recommended. Specify system.

Accounts Receivable. $99.95

*** * * SPECIAL DEAL — NO LESS! * * ***

A complete business package for OSI small systems — (C1, C2, C4 or C8). Includes MAXI-PROS, GENERAL LEDGER, INVENTORY, PAYROLL AND ACCOUNTS RECEIVABLE — ALL THE PROGRAMS THE SMALL BUSINESS MAN NEEDS. $299.95

P.S. We're so confident of the quality of these programs that the documentation contains the programmer's home phone number!

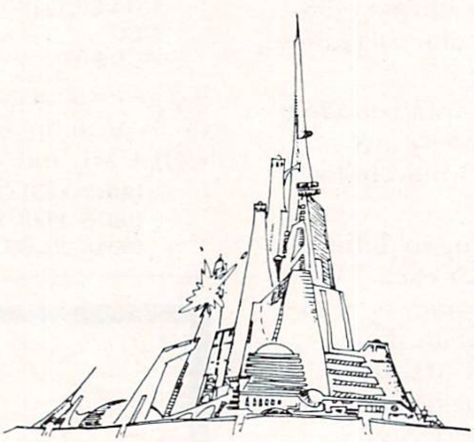
SUPERDISK II

This disk contains a new BEXEC* that boots up with a numbered directory and which allows creation, deletion and renaming of files without calling other programs. It also contains a slight modification to BASIC to allow 14 character file names.

The disk contains a disk manager that contains a disk packer, a hex/dec calculator and several other utilities.

It also has a full screen editor (in machine code on C2P/C4)) that makes corrections a snap. We'll also toss in renumbering and program search programs — and sell the whole thing for — SUPERDISK II $29.95 (5 1/4") $34.95 (8").

AND FUN, TOO!



BOOKKEEPING THE EASY WAY — WITH BUSINESS I

Our business package 1 is a set of programs designed for the small businessman who does not have and does not need a full time accountant on his payroll.

This package is built around a **GENERAL LEDGER** program which records all transactions and which provides monthly, quarterly, annual, and year-to-date PROFIT AND LOSS statements. GENERAL LEDGER also provides for cash account balancing, provides a BALANCE SHEET and has modules for DEPRECIATION and LOAN ACCOUNT computation.

GENERAL LEDGER (and MODULES) $129.95.

PAYROLL is designed to interface with the GENERAL LEDGER. It will handle annual records on 30 employees with as many as 6 deductions per employee.

PAYROLL - $49.95.

INVENTORY is also designed to interface with the general ledger. This one will provide instant information on suppliers, initial cost and current value of your inventory. It also keeps track of the order points and date of last shipment.

INVENTORY - $59.95.

GAMES FOR ALL SYSTEMS

GALAXIAN - 4K - One of the fastest and finest arcade games ever written for the OSI, this one features rows of hard-hitting evasive dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. Specify system — A bargain at $9.95

MINOS - 8K - — Features amazing 3D graphics. You see a maze from the top, the screen blanks, and when it clears, you are in the maze at ground level finding your way through on foot. Realistic enough to cause claustrophobia. — $12.95

NEW — NEW — NEW

LABYRINTH - 8K - This has a display background similar to MINOS as the action takes place in a realistic maze seen from ground level. This is, however, a real time monster hunt as you track down and shoot mobile monsters on foot. Checking out and testing this one was the most fun I've had in years! — $13.95.

TIME TREK - 8K - Real Time and Real graphics Trek. See your torpedoes hit and watch your instruments work in real time. No more unrealistic scrolling displays! — $9.95

SUPPORT ROMS FOR BASIC IN ROM MACHINES — C1S/C2S. This ROM adds line edit functions, software selectable scroll windows, bell support, choice of OSI or standard keyboard routines, two callable screen clears, and software support for 32-64 characters per line video. Has one character command to switch model 2 C1P from 24 to 48 character line. When installed in C2 or C4 (C2S) requires installation of additional chip. C1P requires only a jumper change. — $39.95

C1E/C2E similar to above but with extended machine code monitor. — $59.95

Please specify system on all orders

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.

VISA

AARDVARK TECHNICAL SERVICES, LTD.
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

master charge

OSI **OSI**

THE OSI GAZETTE

OSI RS232 Port And The High Speed Printer Interface

Paul Lilly
Pelham, AL

What About It?

I believe the thing I like most about my Superboard II is that it is full of surprises. Whatever OSI lacked in documentation they made up for it in utility, you just have to find it. While users of other systems pay upwards from $100 for a serial interface, you can put one on a Superboard II for around $5. And that includes a software selectable baud rate with handshaking. This article will show you how to install the port, and how to use it as a high speed printer interface.

How To Do It

OSI shows the schematic for this interface in their user manual, and, although the board has the runs etched in it, the parts were not installed by OSI. But don't try going by the board layout to plug in your parts, it's not exactly correct. Figure 1 is a correct board layout. If you only need a one-way port, you can omit Q2,D16,R62 and R66. If you want true RS232 voltage swings, you can cut between the 2 pads shown in the Note, and hook the high end (pin 7 of J3) to your negative supply. I have an Anadex DP8000 printer hooked up to my Superboard and although the manual for it says it needs a minimum of -3 volts for its RS232 input, I've left the 2 pads connected (voltage to printer swings +5 to 0 volts) and the printer works just fine anyway. Of course that doesn't mean yours will, so you may want to add the negative supply. If you don't need or want the extra speed from your printer, you need not hook up the handshake signals. This way you would be transferring information to your printer at 300 baud, which would allow you to LIST a program on your printer and tape at the same time. But if you want or need to run your printer at high speed (we can run 4800 baud with no hardware changes), for printing reports, making listings of one program, etc., then you will need the handshake capability. For the handshake capability, we can connect one of four different ways (depending on our printer), to the Superboard's CTS (Clear to Send) bus. If your printer uses an RS232 compatable BUSY/READY signal, then connect it to pin 3 of J3 and:

1) If the signal is high (+) when the printer is ready, connect pin 6 of J3 to pin 9 of J3. See figure 3.

2) If the signal is low (-) when the printer is ready, connect pin 10 of J3 to pin 9 of J3. See figure 3.

If your printer uses a TTL compatable BUSY/READY signal (here we can omit Q2,D16,R62,R66) and;

1) If the signal is high (+) when the printer is ready, connect the signal to pin 6 of J3, the connect pin 10 of J3 to pin 9 of J3.

2) If the signal is low (-) when the printer is ready, connect the signal to pin 9 of J3. See figure 3.

Then in all cases, in figure 2 cut one foil and connect between 2 pads as shown, to route this CTS signal to the ACIA. That's it for the hardware. You now have your serial port, ready to run your printer. Although this article describes a hook-up to a printer we can also hook up to a terminal, and use the serial port as an audio port. (More on the audio port at the end of this article).

A little about the ACIA

Before going further, some understanding of the ACIA is necessary. OSI uses the Motorola 6850 ACIA chip, which has 4 registers in it. Two registers we can read from only; the Receive Data Register (read hex addr. F001), and the Status Register (read hex addr. F000). The other two registers we can only write into; the Transmit Data Register (write hex addr. F001), and the Control Register (write hex addr. F000). The Control and Status registers are what give us the programmable flexibility to get the most out of our $5 interface. The control Register allows us to select the format we use to transmit and receive bits, select our transmit rate (baud), enable or disable control and interrupt signals, and reset the ACIA. Bits D0 and D1 are the counter divide and reset bits. Table 1 shows the usage for these bits.

As shown in the table we can select 1 of 3 possible divisions of our transmit clock merely by programming it. If you have a stock Superboard II, the TX clock input is approx. 4800 Hz. The monitor ROM (during system initialization) sets the clock divide to 16, which gives us our baud rate of 300 for our cassette interface. Now here's the trick, prior to outputting to our printer, we write zeros into D0 and D1, setting our clock divide to 1. With our existing TX clock input of 4800 Hz, this will give us a baud rate of 4800. Now we have 300 baud for our cassette, and 4800 baud for our printer, with no hardware changes to go back and forth between them. Bits D2, D3, and D4 select different combinations of word length, parity, and stop bits, Table 2 gives their usage. The monitor sets these bits such that we transmit 8 bits, then 2 stop bits. That setup works fine with my printer so I kept that format in my programming. You will want to check the manual for your printer, determine what format in which it needs to receive bits, and program the ACIA accordingly. Bits D5,D6 and D7 are used to program the IRQ and RTS functions, which are not needed at this time, so we will leave these bits cleared. Therefore when you want to set up the ACIA to run the printer at 4800 baud you will first want to write the binary word 00000011 into hex addr, F000 (POKE61440,3) top reset the ACIA, then write the binary word 00010000 into F000 (POKE61440,16), to select the format and baud rate.

The Status Register will (among many other things) let us know when the Transmitter Register is ready to accept another word. Bit 1 of the Status Register is the Transmistter Register Ready bit. If it is set to 1, we can send another word to the Transmit Register (F001). Two things will keep this bit from being set; 1) If the ACIA has not yet transferred the last word that was written and 2) If the CTS line to the ACIA is high, indication the peripheral cannot accept a word. Although the Status Register has a bit (D3) reserved to indicate the condition of the CTS line, it will not be necessary to check it since it is going to inhibit D1 bit anyway. The monitor ROM checks the Status Register prior to loading a word into the Transmit Register, and will continue looping to check D1 in the Status Register until it finds it set, then the monitor will load the word we want to transfer into the Transmit Register. We need no extra programming to support our handshake signal, as the monitor is already taking care of this for us.

How To Test

We can now test our system to see if it is working properly. We don't need to hook the printer up to make these checks, in fact it will probably be better if we don't. The first thing we should do is load a fairly long program into the system. As it's loading, notice the speed it is listing on the screen, that's 300 baud. Now from the keyboard, type POKE71440,3 (reset ACIA) then type POKE61440,16 (program ACIA). Next type SAVE, then LIST. Now you should see the program listing at a speed 16 times faster than it is loaded at, or 4800 baud. Now, as the program is listing, connect your CTS input to either +5 volts or ground (depending on how we configured it earlier) such that you simulate a peripheral BUSY state. The program should stop listing and not continue until we remove the input. If these tests pass then we are ready to hook up our printer; if the tests fail, you will need to go back and recheck the hardware modifications and programming. Check the manual for your printer and see what is necessary to set it up for an RS232 input at 4800 baud. The Anadex printer has 3 sets of dip switches that can be set for a wide range of baud rates, different types of interfaces, paper length, number of lines to skip over perforation etc. Hook the BUSY/READY output from the printer to J3 of your Superboard as described earlier for your system. Hook the RS232 output from your Superboard J3 pin 2 to the printer input. Hook pin 1 of J3 to the printer COM line (ground). Turn on your printer and away you go.

Add A Speaker

If you don't have a printer at this time but want to go ahead and make the hardware addition anyway, you can use this port to hook up a speaker. You can connect an 8 ohm speaker between pins 1 and 2 of J3. Now set up for 4800 baud and see how your program sounds when you LIST it. You can expand on this idea, and add some interesting effects to your existing programs. Remember, when you are in SAVE mode, any PRINT argument goes to the ACIA as well as your screen when you LIST it.

Some Finals

You should be careful not to program the ACIA while you are in the SAVE mode. I don't know why, but it hangs up the system sometimes when you try to do it. POKE517,1 will put you in SAVE mode and POKE517,0 will take you out of it. Although you can now run your printer at 4800 baud, you are still limited to 300 baud for your cassette due to the audio frequencies generated in the cassette portion of your Superboard. Since the PRINT statement sends its argument to the ACIA when in the SAVE mode, we can output to the printer any results we normally send to the video screen. Most small system printers provide 80 columns per line. The Superboard software is set up to force a LF/CR, after 72 consecutive characters to the ACIA. During a cold start, the system defaults to a terminal width of 72 characters per line if you try to request a larger line length. This can be fixed by writing 80 (Hex 50) in the terminal width storage location (hex F) after initialization, by the statement POKE15,80.

Table 1 – Control Register Usage, Bits D0 & D1

D1	D0	RESULTS
0	0	TX Clock input equals the transmit rate (baud)
0	1	TX Clock input is divided by 16 to give transmit rate (baud)
1	0	TX Clock input is divided by 64 to give transmit rate (baud)
1	1	Reset ACIA

Table 2 – Control Register Usage, Bits D2, D3, D4

D4	D3	D2	RESULTS (TRANSMIT FORMAT)
0	0	0	7 bit word, 1 parity bit (even), 2 stop bits
0	0	1	7 bit word, 1 parity bit (odd), 2 stop bits
0	1	0	7 bit word, 1 parity bit (even), 1 stop bit
0	1	1	7 bit word, 1 parity bit (odd), 1 stop bit
1	0	0	8 bit word, 2 stop bits
1	0	1	8 bit word, 1 parity bit (even), 1 stop bit
1	1	1	8 bit word, 1 parity bit (odd), 1 stop bit

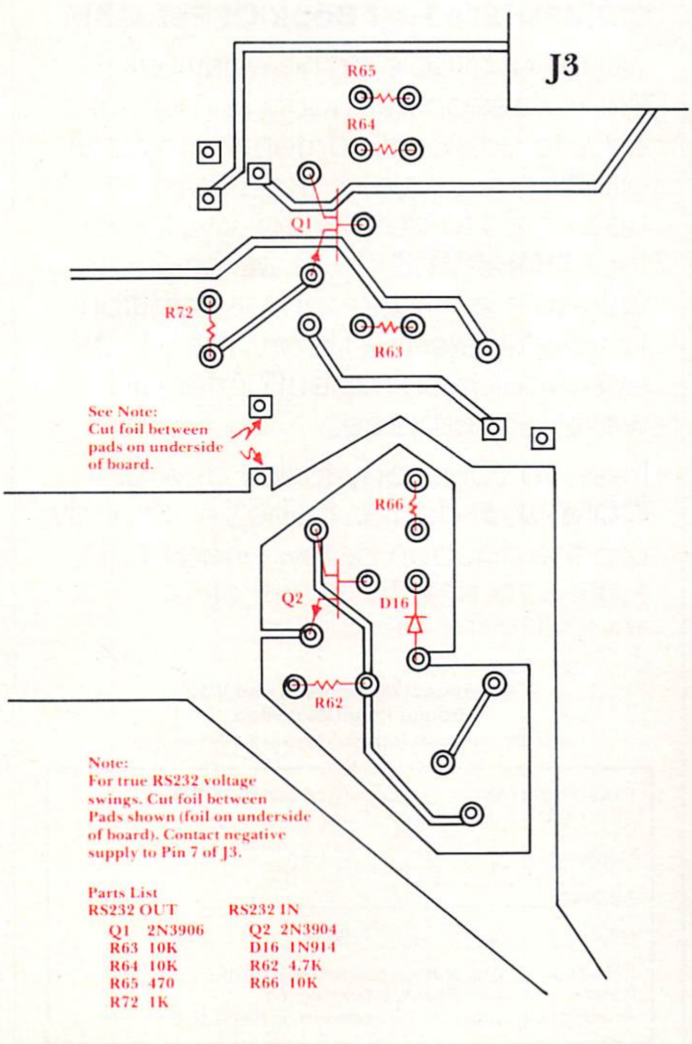


Figure 1

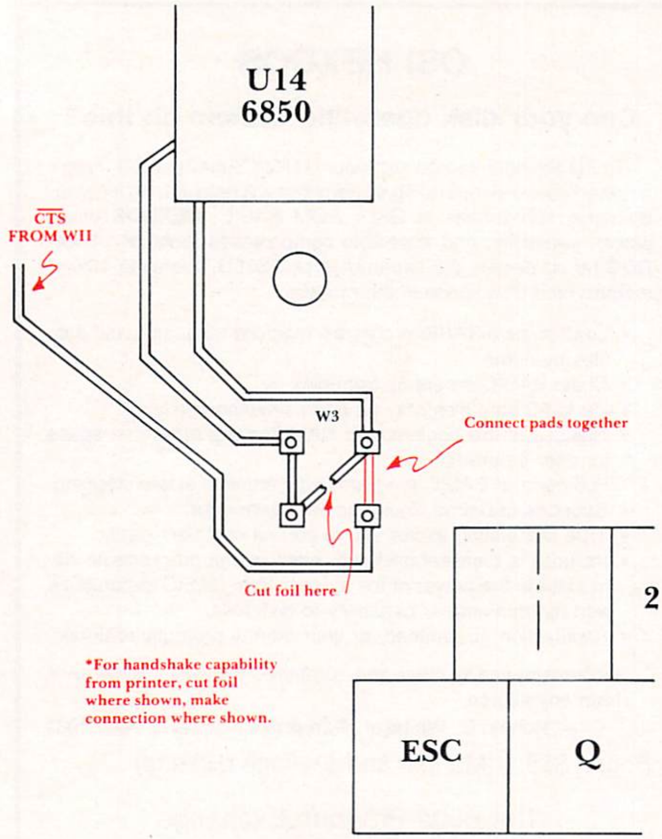


Figure 2

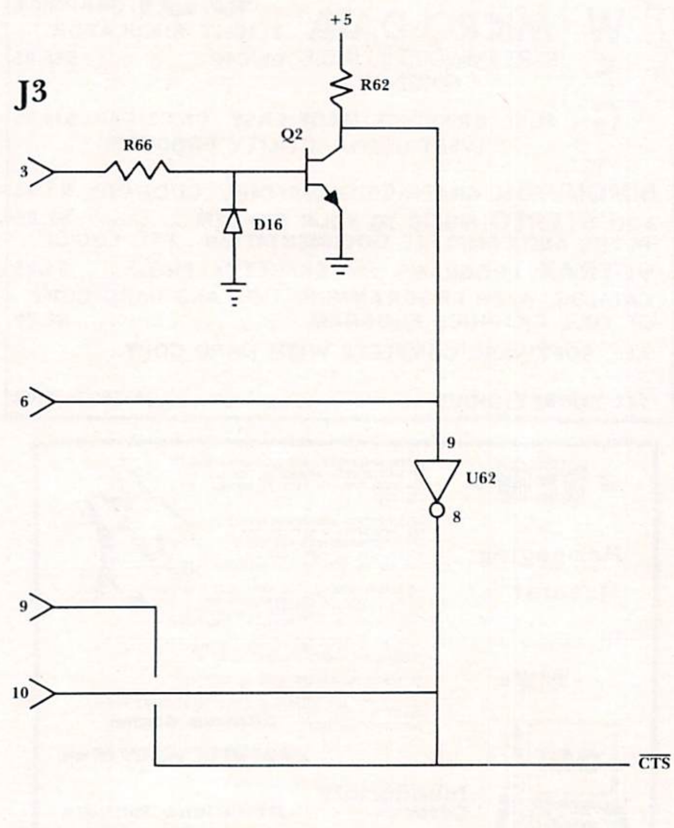


Figure 3 ©

The CBM "Fat 40"— Boon Or Bane?

Jim Butterfield
Toronto, Canada

A new style of 40-column PET has arrived. It looks like an 80-column CBM, and it has many of the 8000 series features, but it's still 40 columns wide. Because of the big twelve-inch screen, screen characters are large and crisp. I call it "The Fat 40" or, more formally, "The Fat 4.0 40", since it's Basic 4.0 in a new package.

It looks like an 80 column unit on the outside, except for the label and the graphics keyboard. It looks like an 80 column on the inside — the processor board has the same architecture as the 8000 series. One wonders: is Commodore just upgrading the 40 column units and incidentally standardizing their processor layout ... or will we be seeing a dual 40/80 column unit one of these days?

The Fat 40 arrived with no advance fanfare. That's something of a surprise, since some effort has been made to enhance the features of the 40 column machine. I've seen no documentation yet — I hope it will be available soon, for there are many features which are not immediately visible to the user.

Convenience Features

The moment they switch on, users will notice a new feature of the Fat 40 — there's a sounder or chiming device built in. It rings on power up and whenever you approach the end of a line. If you have two 40-column rows linked together to make an 80-column line, the bell will correctly sound only when you near the end of the second row. The sounder is tied into the CB2 line, which is used for many game sound effects. Now your games will chuckle, chime, chirp and chatter without the need for an external amplifier and speaker.

A number of keys now repeat automatically if you hold them down for a moment. These include the cursor movement keys, the Space bar, and the Insert/Delete key. It's handy to be able to zip around the screen, especially for those of us who make mistakes.

The screen can now be switched from its normal Graphics mode to Text mode (upper and lower case) with PRINT CHR$(14), and returned back to graphics with PRINT CHR$(142). The effect is somewhat different than the POKE 59468... command. When the screen switches to text mode, the lines are spaced more widely, giving a more pleasing appearance. Some computers may need a minor adjustment of the screen size control if the top and bottom lines partly disappear from the display.

Memory Reallocation

New memory locations have been called into use to allow implementation of the Fat 40's extra features. Standard Basic programs will not be affected, but users should check some of the following locations closely if they have problems.

Location 151 decimal (hexadecimal 97) is often used to tell if a key is being held down or not. It's still useful for that purpose: it will contain 255 as before if no key is pressed. If you have a program that peeks this address to see which particular key is down, you may have problems: the coding has changed and each key is now represented by a new value.

Locations 1001 to 1017 (hex 03E9 to 03F9) are now used by the operating system; if your program uses these locations, better rewrite and use some other part of memory. Some of these locations are changed by the system, and others are used but not changed; leave them alone.

A particularly deadly location is 1003 (hex 03EB). It governs the size of the keyboard input buffer: change it to a higher value (it's normally 9) and you'll be able to type ahead more characters. The extra space used for your characters is in the first cassette buffer — safe enough if you're not using it for something else. The catch is that a value of zero placed in location 1003 gives you a keyboard buffer size of zero, and your keyboard is out of business. It can happen accidentally ... even by just loading some programs.

As a result of these new memory allocations, it appears that the second cassette can no longer be used for reading or writing sequential tape files. Most serious file people use disk these days, but users who are thinking of a twin cassette system should approach the Fat 40 with caution. A brief summary of the new area is given in the following table.

Decimal	Hexadecimal	Usage	80-col
1001	03E9	New key marker	E6
1002	03EA	Key repeat countdown	E5
1003	03EB	Keyboard buffer limit	E3
1004	03EC	Chime time	E7
1005	03ED	Decisecond timer	–
1006	03EE	Key repeat flag	E4
1007	03EF	Tab work value	–
1008-17	03F0-9	Tab stops	03EE-F9

Equivalent locations in 80-column machines are given when applicable. The Decisecond timer is active only in unusual circumstances.

New Programmed-Cursor Commands

The Fat 40 doesn't have an Escape key, which makes it a bit harder to implement new commands as programmed-cursor characters. It's worthwhile knowing how to do them since it gives your program compactness. A couple of examples will illustrate how it's done.

Example 1: Ringing the bell. We wish to print HELLO and ring the bell and then continue the line with THERE and ring the bell again.

Start by typing in PRINT "" — be sure to type in both sets of double quotes — and then press the Delete key once. You'll be left with PRINT " and you'll be out of programmed-cursor mode. Now type in the HELLO, and then press the RVS (Reverse) key. If you've done it right so far, you will *not* get a reverse letter R printed on the screen. Now we press the G key and should get a reverse G on the screen. Turn the reverse feature off by pressing shifted RVS, and then type in THERE (don't forget the space before the word). To arrange for the bell to ring the second time, we repeat the RVS, G, RVS-OFF sequence.

If you've followed directions, you should have a line that looks like:

PRINT "HELLO G THERE G

... with the G's showing in reverse. Press RETURN and printing and bell-ringing will all take place as part of the one statement.

This works on the Fat 40 and on 80 column machines, but will not work on earlier (skinny) 40 column machines ... even with an external speaker attached.

Example 2: Setting and using tabulation stops. We wish to set up tab stops and then print something using them.

Type a line onto the screen that would print, say, the letter T wherever you wanted to put a tab stop. Be sure it starts with PRINT" ... and ends with a final quotation mark, but don't press RETURN yet. Now: back up the cursor to each T you have typed, and type the following sequence: RVS, Shift-I, RVS-OFF, Insert (Shift-DEL), Space. The Shift-I should print as a "curved corner" graphic in reverse. The line will get longer, but that's all right: the tab-set characters will end up in the right place.

When you've replaced all the T's with the reversed graphic, press RETURN and the tab stops are set. You can clear them later with exactly the same statement. Now let's use these tabs in a print statement.

Enter a line such as PRINT "THIS IS A NICE FEATURE", but don't press RETURN. Move the cursor back and replace all of the spaces with RVS I. This time, you won't get a graphic, just the reversed letter I between the words. When you're finished press RETURN and the words will arrange themselves at the tab positions you have previously set.

It takes a little practice to get used to these new features. They are worth the trouble: they are quite handy. The 8000 series machines have the same — and more — features, plus an Escape key to make coding easier. In either case, you may also use the CHR$(..) equivalent to do the same job.

Here's a table of the Fat 40 features:

CHR$	Programmed Char (reverse)	Effect
7	G	Ring Bell
135	Shift G	Ring Bell Longer
9	I	Tabulate
137	Shift I	Set/Clear Tab Stop
14	N	Switch to Text mode
142	Shift N	Switch to Graphics mode
16*	P*	Clear line from cursor
150	Shift V	Clear line before cursor

The control character which clears from the cursor to end-of-line appears to be an error. Compatibility with the 80-column unit and consistency with the other characters suggest that CHR$(22), or reverse V, should do the job. Instead, it's CHR$(16) that works on the Fat 40. Since decimal 22 equals hexadecimal 16, one suspects a simple programming oversight has taken place, and that Commodore will correct this one in a subsequent release. In the meantime, I would suggest using both in any programs that you write using this feature: PRINT CHR$(22);CHR$(16) .. will work nicely on any machine. The clear-from-cursor feature is very useful by the way: it allows you to clear a screen line before writing on it.

Not Quite An 80 Column

Some of the features of the 80 column machine don't seem to be available on the Fat 40. You can't set up screen "windows"; the commands for line insertion and deletion are not in place; and programmers will search in vain for the input-output vectors such as the 80 column provides.

Something that is common to the Fat 40 and the 80 column machines is a standard set of screen entry points. These will be useful to machine language programmers who want to do certain screen/ keyboard operations. The table is given here without details:

E000	**Initialize screen**
E003	**Get a key**
E006	**Input a line**
E009	**Print a character**
E00C	**Interrupt routine**
E00F	**Timer, keyboard service**
E012	**Interrupt exit**
E015	**Clear screen**
E018	**Set text mode**
E01B	**Set graphics mode**
E01E	**Set CRT controller**
E021	**Scroll down**
E024	**Scroll up**
E027	**Scan keyboard**
E02A	**Ring Bell**

It should be emphasized that these entry points do not exist on previous (skinny) 40 column machines. Only 80 column machines and the Fat 40 have them.

Adding It All Up

The Fat 40 has many advantages and only a few drawbacks compared to its 40 column predecessors. The operating system offers us some splendid new features for the screen. On the negative side, the second cassette appears to be virtually out of business and a number of programs will need cleaning up before they can graduate to Fat 40 operation.

All in all, it's a welcome change. Most users will be quite pleased with the bonus features that come with the Fat 40.

```
EXAMPLE 1:

PRINT"HELLO THERE
HELLO THERE

READY.

EXAMPLE 2:

PRINT "                                  "


READY.
PRINT "THISISANICEFEATURE"
THIS  IS      A    NICE    FEATURE

READY.
```

©

David Heise is a Professor of Sociology at the University of North Carolina, Chapel Hill. His books include **Causal Analysis** *and* **Understanding Events**, *and he edited* **Microcomputers in Social Research** *(Beverly Hills, CA: Sage, 1981). He has created commercial microcomputer programs for word processing and statistical analysis.*

Digital Arrayment

David R. Heise
Department of Sociology
University of North Carolina
Chapel Hill, NC

A PET in a store window points to a product on its left and prints the price in large numerals, then the price for a product on its right, then the price for a product on its top, cycling on all day. The referee at your club's ping pong tournament presses a key for each point, and results are displayed on the PET electronic scoreboard. Children in a classroom watch the big numbers appear on the PET screen as teacher types out 1 + 1 = 2.

This article provides a program that makes such applications practical. The program generates large numerals from PET graphic characters. The digits are highly visible from a distance — three columns wide and seven rows high. You call the routine from BASIC programs simply by specifying what digit you want and where on the screen to put it. The 400 bytes of assembly language are protected from overwriting while you program in BASIC.

Operation is fast. Digits appear at once. You can program applications that would not be effective if large numerals had to be constructed with the PRINT command.

The Digits Program

Problem: Create an assembly language routine that will print large digits, 0 to 9, anywhere on the PET screen on command from BASIC.

The logic for such a routine is simple. We determine which graphic characters are needed to form each digit and store the screen values for these characters in RAM. To put a digit on the screen, transfer the stored values to the appropriate cells of screen memory.

Calling the routine from BASIC provides a complication. We have to pass three parameters from BASIC to the special routine: the number to be displayed, and the row and column numbers defining where on the screen to display it. POKEing parameters and calling the routine with SYS is possible, but slow and clumsy. The USR function is preferable because it passes parameters naturally while it calls its routine. However, USR takes only a single argument, not three. How can we evade that limitation?

USR transfers its argument to the floating point accumulator (FACC) as a five-byte floating point number. The five byte format is difficult to decode, but a built-in PET subroutine converts the contents of FACC to a two-byte integer format, which gives 16 bits of easily interpreted information. We pass the required parameters using 15 bits as follows. (The 16th bit — the sign of the integer — is not needed).

—Use four bits to identify the desired **numeral** (n), 0 to 9 plus 10 as a code for blank.

—Use five bits to identify the **screen row**, 0 to 18, where the top of the numeral is to be positioned.

—Use six bits to identify the **screen column**, c, ranging from 0 to 37, for positioning the left side of the numeral.

Combining the three parameters by the expression:

ARGUMENT = N + 16*r + 512*c

or

ARGUMENT = N + ROW*r + COLUMN*c
where ROW = 16 and
COLUMN = 512

will position the parameters in the two byte integer representation of the argument as follows.

```
|       HighByte        |        LowByte        |
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
| |      c      |     r     |   n   |
SGN
```

To recover n we simply find the result of the expression [low-byte AND 15] which masks out the information in the four leftmost bits of the low byte (binary 15 equals 00001111). Then we move all bits rightward four places and mask again with the expression [low-byte AND 31] to get the value of r. We move the bits over another five places (adding 0 bits on the left), leaving c in the low byte.

Problem solved!

Assembly Language Routine

Listing 1 shows the source and object listings for the program.

When the USR function is executed, it transfers its argument to FACC and begins executing the code at line 750. Subroutine FLPINT in the PET's memory converts the floating point number to an integer.

Lines 760-770 apply the mask that recovers the digit to be displayed. Lines 780-830 multiply the digit by 21, yielding an index (line 840) to the relative place in memory where the screen values for the digit are stored.

Lines 850-860 roll the bits of the integer rightward four places with a subroutine at lines 1360-1400. A mask then is applied to get the row number for the top of the digit, and this is stored

on the stack (lines 870-890).
Bits are rolled rightward again leaving the desired column number in the fourth byte of FACC (lines 900-910). Line 920 zeros the fifth cell of FACC. The column number is added to the address for the beginning of screen memory and stored in cells 4 and 5 of FACC — lines 930-990. (SCREEN is the beginning of screen memory minus 40 in order to simplify the next step.)

The desired row number, r, is pulled from the stack, and 40 is added r + 1 times to the number in FACC 4 and 5 (lines 1000-1100). The final result in FACC bytes 4 and 5 is the address of the screen memory cell where construction of the digit is to begin. This address, with the 6502's Y index set to zero (line 1110), can be used to addess the cell by indirect addressing.

Lines 1130-1140 fetch the first character for the digit and transfer it to the screen. The X and Y indexes are incremented, and the second character is transfered (lines 1150-1190). This is repeated for the third character (lines 1200-1250). Then the X index is incremented as usual (line 1250), but the Y index is increased so it points all the way into the next screen row, once again at the left side of the character being generated (lines 1260-1290). The above steps are repeated, forming the next row of the digit. The cycle repeats seven times (lines 1300-1310), and then execution returns to BASIC (line 1320).

The screen values for the 21 graphics that form each digit are given in lines 1440-2130. Code 10 produces a blank created from 21 spaces (lines 2140-2200).

Auxiliary Assembly Routines

The code in Listing 1 begins with a routine that sets the pointer for the USR function to the beginning of the Digit code (lines 310-340) and protects the assembled routines from being overwritten by BASIC (lines 360-400). This is called by a SYS statement before loading or entering any BASIC lines.

A routine to reverse the entire screen (lines 470-670) does not aid the construction of digits, but it is typically useful in applications where large numbers are used. This procedure also is invoked by a SYS command.

Entering the Code:

The lefthand side of Listing 1 gives the hexadecimal code for the program, which can be entered using the PET monitor. The code is saved as a separate file to be loaded by the Loader Program given in the next section. You should enter and save the Loader Program first.

The listed code is assembled for placement at the top of RAM on an 8K machine. The same code can be used on 16K or 32K PETs as long as your application program fits in the space left for BASIC — about 6500 bytes. Alternatively, the code can be relocated for positioning at the top of RAM in larger machines, leaving more space for BASIC programming. This involves doing two things. (1) Enter the code beginning at address $3E70 for 16K, or at $7E70 for 32K. (2) Change the high bytes of some addresses in the code to $3E or $7E so they correspond to the relocation. Places where a change is required are marked by <SIZE> in the comments column.

The given code is for Operating System 3.0. Changes must be made for operation with "old ROMs" or the newer Operating System 4.0. Conversion values for old ROMs are given in brackets in the comments column. The code at line 750 must be changed to [20 A7 D0] for old ROMs; to [20 D1 CD] for O.S. 4.0.

After all of the code has been entered, it can be saved on tape using the following monitor command.

```
.S "DIGITS CODE",01,1E70,1FF4
```

The file should be written on tape following the Loader Program.

The code can be saved on a disk which has been initialized in unit #0 with the command:

```
.S "0:DIGITS CODE",08,1E70,1FF4
```

Naturally, the addresses have to be changed in these save commands if the code is relocated.

Loader Program

Listing 2 presents a short BASIC program that automatically loads the DIGITS CODE and then loads and runs a subsequent BASIC file named MAIN. (See the end of Listing 3 for explanation of special symbols.) The subsequent BASIC program is your applications program. It must be named MAIN to accord with the Loader Program.

A BASIC program always must be loaded after DIGITS CODE in order to reset the PET to its normal operating parameters. Otherwise you will get an OUT OF MEMORY error when you try to enter BASIC lines. Of course, MAIN could be a

dummy program consisting of nothing more than: 10 REM.

Using the Routines in BASIC

Set the multiplier variables at the beginning of your program as follows:

ROW = 16:COLUMN = 512

Then the following statement will print a digit, D, with its top left corner at screen row R and screen column C.

X = USR(D + ROW*R + COLUMN*C)

If D equals 10, then a blank is printed, which is useful for erasing old digits. (The value returned in X is meaningless.)

Screen reversal is obtained by

SYS 7809 (for 8K),
SYS 15984 (for 16K), or
SYS 32368 (for 32K).

Digits, erasure, and screen reversal can all be obtained by entering the BASIC commands in direct mode.

My Application

I have a classroom show for my university course in social statistics that goes like this.

"Professor Dummie at Nihil University has proposed a theory that males are lower class more often than females. He interviewed 100 people, computed the cross tabulation of class by gender, and found his hypothesis was supported. The news created a sensation among researchers, scores of whom are repeating the study.

In fact, this machine (the PET) is going to inform us how the results are coming out. For the rest of the period you will see the results of study after study. In each case, there will be 100 cases in a cross-tab along with the chi-square statistic revealing whether the results are important or not. The screen will turn white when the results are statistically significant. Interrupt my lecture when that happens so I can keep tabs on how Dummie's idea is faring. When the results support him, I'll put a mark in this column (I write "males are lower class" on the blackboard). When the results are opposite, I'll mark this column ("females are lower class").

"We don't need to bother with findings that aren't statistically significant. They wouldn't be published anyway.

"Of course," I add in a stage whisper, "the tables really are *random*."

The sideshow allows students to see chi square values for nearly 200 tables — a quick dose of "experience." They see a demonstration of the statistical principle that about one out of twenty statistics really is significant by chance at the .05 level. They are shown that a stupid theory can actually lead to reporting of significant results, even with sequences of confirmations.

The computer in the classroom provides the dramatization, but the show works only because numbers on the screen are big enough for everyone in the room to see.

Listing 3 presents the Chi Square Program. Enter it and save it as MAIN to get a nice demonstration of all the features in the Digits routine. (Be patient when you run it: significant results producing a reversed screen may not appear until the 20th or 30th table.)

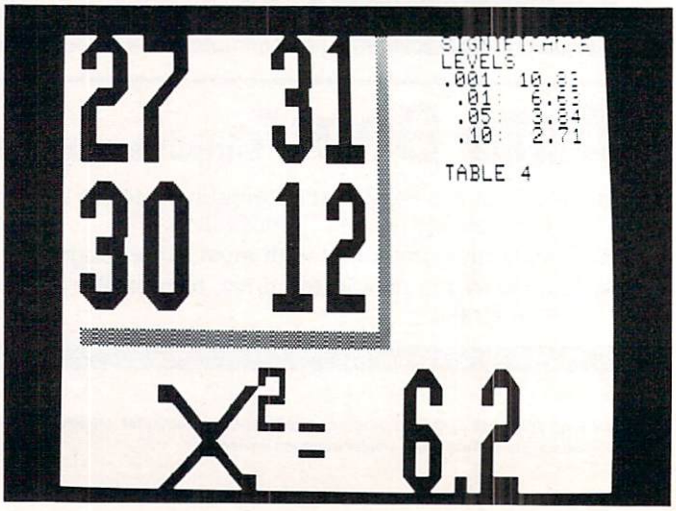
Program 1.

```
800 FOR ADRES=7792TO8191:READ DATTA:
    POKE ADRES,DATTA:NEXT ADRES
7792 DATA 169, 164, 133, 1, 169, 30
7798 DATA 133, 2, 169, 111, 133, 52
7804 DATA 169, 30, 133, 53, 96, 160
7810 DATA 0, 169, 232, 133, 97, 169
7816 DATA 131, 133, 98, 177, 97, 24
7822 DATA 105, 128, 145, 97, 56, 165
7828 DATA 97, 233, 1, 133, 97, 165
7834 DATA 98, 233, 0, 133, 98, 201
7840 DATA 127, 208, 232, 96, 32, 154
7846 DATA 208, 169, 15, 37, 98, 168
7852 DATA 169, 235, 24, 105, 21, 136
7858 DATA 16, 250, 170, 160, 4, 32
7864 DATA 5, 31, 169, 31, 37, 98
7870 DATA 72, 160, 5, 32, 5, 31
7876 DATA 132, 99, 24, 169, 216, 101
7882 DATA 98, 133, 98, 169, 127, 101
7888 DATA 99, 133, 99, 104, 168, 24
7894 DATA 165, 98, 105, 40, 133, 98
7900 DATA 169, 0, 101, 99, 133, 99
7906 DATA 136, 16, 240, 160, 0, 189
7912 DATA 13, 31, 145, 98, 232, 200
7918 DATA 189, 13, 31, 145, 98, 232
7924 DATA 200, 189, 13, 31, 145, 98
7930 DATA 232, 152, 24, 105, 38, 168
7936 DATA 192, 24, 208, 227, 96, 70
7942 DATA 97, 102, 98, 136, 208, 249
7948 DATA 96, 233, 160, 223, 160, 32
7954 DATA 160, 160, 32, 160, 160, 32
7960 DATA 160, 160, 32, 160, 160, 32
7966 DATA 160, 95, 160, 105, 233, 160
```

```
7972 DATA 32, 32, 160, 32, 32, 160
7978 DATA 32, 32, 160, 32, 32, 160
7984 DATA 32, 32, 160, 32, 160, 160
7990 DATA 160, 233, 160, 223, 160, 32
7996 DATA 160, 32, 32, 160, 233, 160
8002 DATA 105, 160, 32, 32, 160, 32
8008 DATA 32, 160, 160, 160, 233, 160
8014 DATA 223, 160, 32, 160, 32, 32
8020 DATA 160, 32, 160, 234, 32, 32
8026 DATA 160, 160, 32, 160, 95, 160
8032 DATA 105, 160, 32, 160, 160, 32
8038 DATA 160, 160, 32, 160, 160, 160
8044 DATA 160, 32, 32, 160, 32, 32
8050 DATA 160, 32, 32, 160, 160, 160
8056 DATA 160, 160, 32, 32, 160, 160
8062 DATA 223, 32, 32, 160, 32, 32
8068 DATA 160, 160, 32, 160, 95, 160
8074 DATA 105, 233, 160, 223, 160, 32
8080 DATA 160, 160, 32, 32, 160, 160
8086 DATA 223, 160, 32, 160, 160, 32
8092 DATA 160, 95, 160, 105, 160, 160
8098 DATA 160, 32, 32, 160, 32, 233
8104 DATA 105, 233, 105, 32, 160, 32
8110 DATA 32, 160, 32, 32, 160, 32
8116 DATA 32, 233, 160, 223, 160, 32
8122 DATA 160, 160, 32, 160, 244, 160
8128 DATA 234, 160, 32, 160, 160, 32
8134 DATA 160, 95, 160, 105, 233, 160
8140 DATA 223, 160, 32, 160, 160, 32
8146 DATA 160, 95, 160, 160, 32, 32
8152 DATA 160, 160, 32, 160, 95, 160
8158 DATA 105, 32, 32, 32, 32, 32
8164 DATA 32, 32, 32, 32, 32, 32
8170 DATA 32, 32, 32, 32, 32, 32
8176 DATA 32, 32, 32, 32, 170, 170
8182 DATA 170, 170, 170, 170, 170, 170
8188 DATA 170, 170, 170, 170, 170, 170
READY.
```

Program 2.

```
 100 : REM: LOADER PROGRAM
 110 :
 120 : REM: ROUTINE TO LOAD THE ASSEMBLED CODE, THEN THE MAIN PROGRAM.
 130 : REM: DELETE NEXT LINE TO LOAD FROM TAPE, INCLUDE IT TO LOAD FROM DISK.
 140 DV$=",8":CU$="<q>"
 150 : REM: SET GRAPHICS MODE.
 160 POKE 59468,12
 170 : REM: SET OPERATING SYSTEM LOCATIONS
 180 COUNTER=158  :REM 525 FOR OLD ROM
 190 BUFFER=623   :REM 527
 200 : REM: SET SUBROUTINE LOCATION
 210 INVOKE=7792  :REM 15984 IN 16K; 32368 IN 32K
 220 : REM: STORE 4 CARRIAGE RETURNS IN INPUT BUFFER,
 230 POKE COUNTER,4
 240 POKE BUFFER,13:POKE BUFFER+1,13:POKE BUFFER+2,13:POKE BUFFER+3,13
 250 : REM: SET UP SCREEN TO GET THE SUBROUTINE FILE -- 'DIGITS CODE',
 260 PRINT "<s><Q><Q>LOAD" CHR$(34) "DIGITS CODE" CHR$(34) DV$
 270 : REM: AND PROTECT THE CODE
 280 PRINT CU$ "<Q><Q><Q><Q><Q>SYS" INVOKE
 290 : REM: THEN LOAD PROGRAM 'MAIN'
 300 PRINT "<Q><Q>LOAD" CHR$(34) "MAIN" CHR$(34)DV$
 310 PRINT CU$ "<Q><Q><Q><Q><Q>RUN<S>";:END
READY.
```

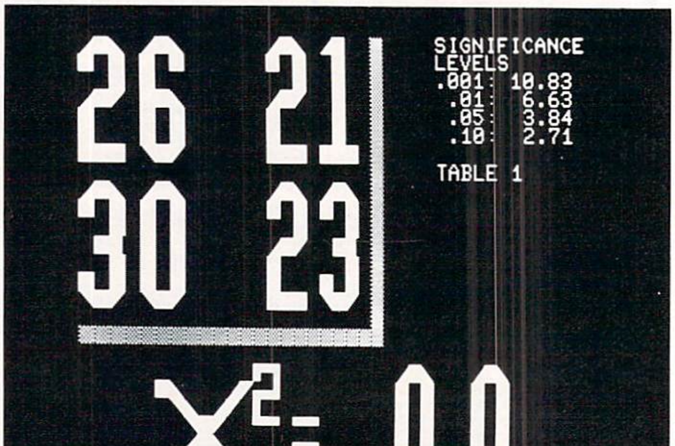



Program 3.

```
100 REM  *** CHI SQUARE ***
110 :
120 REM  SET THE ROW AND COLUMN MULTIPLIERS FOR THE USR FUNCTION
130 ROW=16:COLUMN=512
140 :
150 REM  READ THE ROW AND COLUMN COORDINATES FOR POSITIONING DIGITS IN TABLE
160 FOR I=1 TO 8
170 READ R(I),C(I)
180 NEXT I
190 DATA  0,1,  0,5,  0,14,  0,18,  8,1,  8,5,  8,14,  8,18
200 REM  AND THE DIGITS IN CHI-SQUARE VALUE
210 READ R(9),C(9), R(10),C(10)
220 DATA  18,24,  18,30
230 :
240 REM  MAKE A SMALL REFERENCE TABLE SHOWING PROBABILITIES
250 REM  FOR CHI-SQUARES WITH ONE DEGREE OF FREEDOM
260 PRINT "<s>" TAB(27) "SIGNIFICANCE
270 PRINT TAB(27) "LEVELS
280 PRINT TAB(27) ".001: 10.83
290 PRINT TAB(27) " .01:  6.63
300 PRINT TAB(27) " .05:  3.84
310 PRINT TAB(27) " .10:  2.71
320 :
330 REM  DRAW THE TABLE OUTLINE
340 PRINT "<S>";
350 FOR I=1 TO 16
360 PRINT TAB(22) "[&]
370 NEXT I
380 PRINT "[ ][&][&][&][&][&][&][&][&][&][&][&][&][&][&][&][&][&][&][&][&][&]
390 :
400 REM  DRAW A CHI-SQUARE FIGURE ("[_]" STANDS FOR LEFT-ARROW WITH SHIFT)
410 PRINT "<Q>[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]<R>[,][;]<r>
420 PRINT "[ ][ ][ ][ ][ ][ ]<R>[ ][_]<r>[ ][ ][ ]<R>[)]<r>[)]["]<R>[>]<r>
430 PRINT "[ ][ ][ ][ ][ ][ ][ ][_]<R>[_]<r>[ ]<R>[)]<r>[)][ ][!]
440 PRINT "[ ][ ][ ][ ][ ][ ][ ][ ][_]<R>[ ]<r>[)][ ][ ]<R>["]["]<r>[ ]<R>["]["]<r>
450 PRINT "[ ][ ][ ][ ][ ][ ][ ][ ]<R>[)][ ][_]<r>[ ][ ][ ][ ][ ]["]["]
460 PRINT "[ ][ ][ ][ ][ ][ ][ ]<R>[)]<r>[)][ ][_]<R>[_]<r>
470 PRINT "[ ][ ][ ][ ][ ][ ]<R>[)]<r>[)][ ][ ][ ][_]<R>[ ]<r>";
480 PRINT "[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]<R>[ ]<S>
490 :
500 REM  RANDOMLY ASSIGN 100 CASES TO 4 CELLS OF ARRAY
510 V=RND(-TI)
520 FOR I=1 TO 100
530 V=INT(RND(1)*3.999999)
540 A(V)=A(V)+1
550 NEXT I
560 :

READY.
```

```
570 REM  REVERSE SCREEN IF IT IS HIGHLIGHTED
580 IF RV THEN SYS 7809:RV=0:FOR I=1 TO 1000:NEXT I
590 :
600 REM  ERASE LAST DIGITS IN TABLE
610 FOR I=1 TO 10
620 V=USR( 10 + ROW*R(I) + COLUMN*C(I) )
630 NEXT I
640 :
650 REM  DISPLAY THE ARRAY AS A TABLE
660 FOR I=0 TO 3
670 T=INT(A(I)/10)
680 REM  USE BLANKS IN PLACE OF LEADING ZEROS
690 V=T:IF V=0 THEN V=10
700 V=USR( V + ROW*R((2*I)+1) + COLUMN*C((2*I)+1) )
710 V=USR( A(I)-10*T + ROW*R((2*I)+2) + COLUMN*C((2*I)+2) )
720 NEXT I
730 :
740 REM  COMPUTE THE VALUE OF CHI-SQUARE FOR THE TABLE
750 CHI=100*(A(0)*A(3)-A(1)*A(2))^2
760 CHI=.05+CHI/((A(0)+A(2))*(A(1)+A(3))*(A(0)+A(1))*(A(2)+A(3)))
770 CHI=INT(10*CHI)/10
780 IF CHI>9.9 THEN CHI=9.9
790 :
800 REM  DISPLAY CHI-SQUARE VALUE
810 V=USR( INT(CHI) + ROW*R(9) + COLUMN*C(9) )
820 V=USR( 10*(CHI-INT(CHI)) + ROW*R(10) + COLUMN*C(10) )
830 :
840 REM  SHOW HOW MANY TABLES HAVE BEEN DONE
850 M=M+1:PRINT "<S><Q><Q><Q><Q><Q><Q><Q>" SPC(27) "TABLE" M
860 :
870 REM  PAUSE FOR FIVE SECONDS
880 T=TI
890 IF (TI-T)<300 THEN 890
900 :
910 REM  IF SIGNIFICANT, REVERSE FIELD, HOLD LONGER, AND SET HIGHLIGHT FLAG
920 IF CHI>3.84 THEN SYS 7809:FOR I=1 TO 5000:NEXT I:RV=-1
930 :
940 REM  CLEAR TABLE
950 FOR I=0 TO 3
960 A(I)=0
970 NEXT I
980 :
990 REM  MAKE ANOTHER TABLE
1000 GOTO 520
1010 :
1020 REM  SYMBOLS USED IN LISTING
1030 "[X]"      PRESS KEY WITH SHIFT
1040 "<s>"      CLEAR SCREEN
1050 "<S>"      CURSOR HOME
1060 "<q>"      CURSOR UP
1070 "<Q>"      CURSOR DOWN
1080 "<}>"      CURSOR LEFT
1090 "<]>"      CURSOR RIGHT
1100 "<R>"      REVERSE ON
1110 "<r>"      REVERSE OFF
READY.
```



Keyword

Charles Brannon
Greensboro, NC

KEYWORD is another in a family of labor-saving programs. As with KEYPRINT, the mere touch of one key does a lot.

The purpose of KEYWORD is to save typing time. When you type any shifted letter of the alphabet, a whole BASIC word (or keyword) is inserted at the cursor position. For example, when you type SHIFT-A, the word PRINT is spelled out instantly. SHIFT-P will give you POKE, another commonly-used command. There are 26 letters, so you can access 26 keywords. These keywords are found in a 26-byte "dictionary" in memory. I have defined the 26 keys as the most commonly-used BASIC keywords, in a mnemonic (easily learned) arrangement. However, you can redefine the dictionary to supply *your* favorite set of keywords. When using the 4.0 version, for example, you may want to define several keys as disk commands, giving you a clear, easy DOS SUPPORT.

On to application ... KEYWORD will work on either the Upgrade ROM or 4.0 ROM with any memory size. You load it and lock it into memory with the supplied BASIC program. Then you can either activate or deactivate it with a single SYS command. The last five lines of DATA statements are the tokens that correspond to each letter of the alphabet. If you change them before running the program, KEYWORD will then have a different dictionary. You can have several versions of KEYWORD on disk or tape to suit your needs. More on how you can do this later.

Be very careful when entering the KEYWORD loader program. It reads the numbers in the DATA statements, which are the decimal equivalents of 6502 machine language code. The 6502 will not tolerate mistakes; one typo and it crashes, so do your best to type it in accurately. For insurance, *be sure* to save it on tape or disk before using it. Then you can re-LOAD it and proofread it. What the BASIC program does is:

1) Find the top of memory
2) Reduce BASIC's top-of-memory pointer by one page (256 bytes)
3) Place KEYWORD at the top of memory
4) Make corrections due to relocation
5) Correct, if necessary, for Upgrade ROM (3.0) BASIC
6) Display the ON/OFF address and TABLE address
7) Return control back to you!

When the program displays the addresses, write them down for further reference. The first one displayed is ON/OFF. When you SYS to it, it will "activate" KEYWORD, or if KEYWORD is already activated, it will turn it off. The second number is the starting address of the token table. If you didn't change the DATA statements, you can use POKE to change any of the 26 tokens in the dictionary. What the heck is a *token*? A token is a single character that represents an entire keyword. BASIC stores its programs in tokenized format, and you can see what an incredible memory-saver this is. PRINT condenses down into a single character, whose ASCII value is 153. The first byte in the token table is 153, and the first of 26 keys is the letter A. That's why SHIFT-A gives you PRINT, and that's the secret to changing the library. Just change any of the 26 numbers in the last four DATA lines to the tokens you want to substitute. Remember to save your modified version. Included at the end of this article is a list of the tokens and their matching keywords.

Finally, some more advanced programming issues need to be mentioned. Like KEYPRINT, the first thing KEYWORD does is "activate" itself by changing the IRQ vector at \$90-\$91 to point to the KEYWORD handler routine. This section handles the input. KEYWORD checks location \$D9 (ASCII value of last key pressed) for the range of shifted A to shifted Z, and skips out if not found. It then converts this value to a number from zero to 25 and uses it as an index to look up the proper token in the TOKEN TABLE. Finally, it uses a modified version of BASIC's LIST routine to "un-crunch" the token into its expanded form and insert these ASCII characters into the keyboard buffer, where they'll be treated as keystrokes and typed out by BASIC just as if the user had typed them. Note that the first character in the buffer must be ASCII 20 (delete) in order to erase the shifted character that appears on the screen.

Table 1.

A	=	PRINT	N	=	NEXT
B	=	PEEK	O	=	OPEN
C	=	PRINT#	P	=	POKE
D	=	CMD	Q	=	CLOSE
E	=	INPUT#	R	=	RUN
F	=	FOR	S	=	SAVE
G	=	GOTO	T	=	THEN
H	=	LEFT$	U	=	RND
I	=	INPUT	V	=	VERIFY
J	=	GET	W	=	LOAD
K	=	READ	X	=	SAVE
L	=	LIST	Y	=	GOSUB
M	=	TAB(	Z	=	SYS

Table 2.

END	- 128	AND	- 175
FOR	- 129	OR	- 176
NEXT	- 130	>	- 177
DATA	- 131	=	- 178
INPUT#	- 132	<	- 179
INPUT	- 133	SGN	- 180

NEW PRODUCT

SWARM-100

JUST PLUG IT IN

- No soldering
- No messy wires

SOFTWARE SELECTABLE

1. Software select one of two operating systems. (BASIC 2.0 / BASIC 4.0)
2. Software select utility ROMs at conflicting addresses.

$125.00 (U.S.)
($150.00 Canadian)
Add $3.00 shipping to all points outside Canada.

Master Charge and VISA accepted.

BATTERIES INCLUDED

Village by the Grange
71 McCaul Street
Toronto, Ontario
Canada M5T 2X1
(416) 596-1405

ASERT yourself... with CFI's new Database Retrieval System

WHO CAN USE ASERT?

libraries
personnel departments
dating services

OFFICIALLY APPROVED BY COMMODORE

schools
employment agencies
accountants

ANY BUSINESS THAT KEEPS RECORDS CAN USE ASERT TO:

- Create up to 21 fields per record
- Restructure fields at any time
- Sort on any field at any time
- Use FREE-TEXT area for comments
- Create up to 90 searchwords
- Search & retrieve on any combination of 90 searchwords
- Search with MUST HAVE, MAY NOT HAVE and OPTIONAL operators
- Print out hardcopy including labels
- Output to any word processor
- Compile summary statistics
- Maintain 1900 records per disk with "virtual" 5K record length

ASERT — Aid for Search & Retrieval of Text — $495 complete
For the 8032 CBM and 8050 disk drive — Commodore Approved Software

OTHER CFI SOFTWARE

Federal Income Tax Preparation System*
Personal Tax Calculator*
Emergency Control Program*
VIC Animation Tutorial
*Distributed for CFI under the Commodore label

ALL CFI SOFTWARE AVAILABLE
from your local Commodore dealer
or direct from CFI

CFI . . . *computer solutions,* 875 West End Ave., New York, NY 10025

```
DIM      - 134      INT        - 181
READ     - 135      ABS        - 182
LET      - 136      USR        - 183
GOTO     - 137      FRE        - 184
RUN      - 138      POS        - 185
IF       - 139      SQR        - 186
RESTORE  - 140      RND        - 187
GOSUB    - 141      LOG        - 188
RETURN   - 142      EXP        - 189
REM      - 143      COS        - 190
STOP     - 144      SIN        - 191
ON       - 145      TAN        - 192
WAIT     - 146      ATN        - 193
LOAD     - 147      PEEK       - 194
SAVE     - 148      LEN        - 195
VERIFY   - 149      STR$       - 196
DEF      - 150      VAL        - 197
POKE     - 151      ASC        - 198
PRINT#   - 152      CHR$       - 199
PRINT    - 153      LEFT$      - 200
CONT     - 154      RIGHT$     - 201
LIST     - 155      MID$       - 202
CLR      - 156      GO         - 203
CMD      - 157      Following are 4.0 Disk
SYS      - 158      BASIC Commands:
OPEN     - 159      CONCAT     - 204
CLOSE    - 160      DOPEN      - 205
GET      - 161      RECORD     - 206
NEW      - 162      HEADER     - 207
TAB(     - 163      COLLECT    - 208
TO       - 164      BACKUP     - 209
FN       - 165      COPY       - 210
SPC(     - 166      APPEND     - 211
THEN     - 167      DSAVE      - 212
NOT      - 168      DLOAD      - 213
STEP     - 169      CATALOG    - 214
 +       - 170      RENAME     - 215
 -       - 171      SCRATCH    - 216
 *       - 172      DIRECTORY  - 217
 /       - 173
 I       - 174
```

```
100 REM KEYWORD LOADER
110 POKE 53,PEEK(53)-1:CLR
120 BASE=PEEK(52)+256*PEEK(53)
130 PRINT"ĥPATIENCE..."
140 FOR I=0 TO 164:READ A:POKE BASE+I,A:
    ¬NEXT I
150 REM RELOCATION ADJUSTMENTS
160 AD=BASE+37:GOSUB390
170 POKE BASE+13,LO
180 POKE BASE+22,HI
190 AD=BASE+138:GOSUB390
200 POKE BASE+57,LO:POKE BASE+58,HI
210 AD=BASE+164:GOSUB390
220 POKE BASE+99,LO:POKE BASE+100,HI
230 POKE BASE+109,LO:POKE BASE+110,HI
240 AD=BASE+35:GOSUB390
250 POKE BASE+8,LO:POKE BASE+9,HI
260 AD=BASE+36:GOSUB390
270 POKE BASE+17,LO:POKE BASE+18,HI
280 IF PEEK(50003)=160 THEN 350
290 REM CONVERSIONS FOR 3.0 BASIC
300 POKE BASE+65,146
310 POKE BASE+69,192
320 POKE BASE+136,46
330 POKE BASE+137,230
340 PRINT"ĥ*** KEYWORD 3.0 ***↓":GOTO360
350 PRINT"ĥ*** KEYWORD 4.0 ***↓"
360 PRINT"ON/OFF:   SYS ɼ";BASE
370 PRINT"↓TABLE AT:      ɼ";BASE+138
380 END
390 HI=INT(AD/256):LO=AD-256*HI:RETURN
400 DATA 120, 165, 144, 72, 165, 145
410 DATA 72, 173, 35, 32, 208, 2
420 DATA 169, 37, 133, 144, 173, 36
430 DATA 32, 208, 2, 169, 32, 133
440 DATA 145, 104, 141, 36, 32, 104
450 DATA 141, 35, 32, 88, 96, 0
460 DATA 0, 72, 138, 72, 152, 72
470 DATA 165, 217, 201, 193, 144, 82
480 DATA 201, 219, 176, 78, 56, 233
490 DATA 193, 170, 189, 138, 32, 162
500 DATA 0, 134, 158, 170, 160, 178
510 DATA 132, 31, 160, 176, 132, 32
520 DATA 160, 0, 10, 240, 16, 202
530 DATA 16, 12, 230, 31, 208, 2
540 DATA 230, 32, 177, 31, 16, 246
550 DATA 48, 241, 200, 177, 31, 48
560 DATA 17, 8, 142, 164, 32, 230
570 DATA 158, 166, 158, 157, 111, 2
580 DATA 174, 164, 32, 40, 208, 234
590 DATA 230, 158, 166, 158, 41, 127
600 DATA 157, 111, 2, 169, 20, 141
610 DATA 111, 2, 230, 158, 104, 168
620 DATA 104, 170, 104, 76, 85, 228
630 REM * KEYWORD TOKEN LIBRARY HERE:*
640 DATA 153,194,152,157,132,129
650 DATA 137,200,133,161,135,155
660 DATA 163,130,159,151,160,138
670 DATA 148,167,187,149,147,148
680 DATA 141,158,0
READY.
```

©

CBM/PET Loading, Chaining, and Overlaying

Jim Butterfield
Toronto, Canada

It can be useful for a Basic program to load in another program. For example, if you are writing an ambitious program and you run out of memory, you could build your program in two or more parts. Each part loads in the next one and your program will run even though there isn't enough memory for the whole thing at one time. This technique is called "chaining" — each part chains to the next.

Then again, you might have a series of programs to do various types of work, and use a menu program to ask the user to select which job he wants done. The menu program will then load in the program needed to do the work. This is not called chaining — each new program stands by itself and doesn't logically connect to a previous part. It can be called "Loading by Program."

Finally, there's a need to leave the existing program in place, but to bring information into memory that supplements the program. For example, machine language coding might be brought in to do a certain job; when the job is done, a new machine language program might be loaded into the same memory slot ... and all the time the controlling Basic program remains undisturbed. Similarly, you can load screen memory with information directly from disk. The technique is called "overlay" — new data goes where old data used to be.

Chaining

The PET is very well adapted to program chaining. Each program module can load in the next and keep going. No variables are lost and the new program module runs from the first statement.

It's easy to do. If your program has finished its particular job, and wants to bring in part two of the program stored on disk and called "PART2", just code:

```
870 LOAD "PART2",8
```

Part two will load and start running from the first statement. It will be a true program continuation since your variables are preserved.

Important precautions: No program in the chained sequence must be bigger than the first program. Any defined functions (DEF FNX..) must be redefined as each new program is brought in. DATA statements are brought in fresh with each program module, of course. Strings which are fixed in the program must be redefined in each module. A fixed string is one that is given completely within the program: H$ = "HELLO" produces H$ as a fixed string which must be defined again upon chaining; INPUT Y$ or Z$ = "HELLO" + "THERE" produces Y$ and Z$ as computed strings which will be carried through chaining without the need for redefinition.

Chaining can also be done from cassette tape. The same rules apply and the chaining line would read: 870 LOAD "PART2".

Chaining Example

Try this simple example of program chaining. This program calculates a series of square roots (in part 1) and then prints them (in part 2).

```
100 DIM X(20)
110 FOR J=1 TO 20
120 X(J)=SQR(J)
130 NEXT J
140 REM  XXXXXXXXXXXXXXXX
150 REM  XXXXXXXXXXXXXXXX
160 LOAD "ROOTPRINT",8
```

Type in the above program and save it under the name ROOTCALC. The REM statements seem odd; they are there to make sure that this first program is longer than the second part.

Now we're ready for the second program. Type NEW and enter:

```
100 FOR K=3 to 15
110 PRINT K,X(K)
120 NEXT K
130 END
```

Save this one with name ROOTPRINT. Now we're ready to go.

Load the first program (ROOTCALC) and type RUN. There will be a pause while the twenty square roots are calculated, and then you'll see the second program being loaded. The table will be printed, and the program will stop. If you LIST, you'll see that ROOTPRINT is there.

Loading From Programs

When you want a program to load another program — not as a continuation but as a fresh start — you'll be tempted to use chaining anyway. It works very neatly and is easy to code. And it doesn't really matter if there are some variables left over from the previous program.

This will work well, but there are a few things you should watch closely. Remember that chaining insists that your first program is as large or larger than any other to be loaded — the first program allocates memory space for all the others. Watch out for array dimensioning: you should set up all your arrays in your first, initializing program or you may bomb out on a REDIM'D ARRAY error. And remember that with this method, you can't assume that all variables are zero when the program

EPROM PROGRAMMER

No fuss just plug in an do your own **EPROMS**.

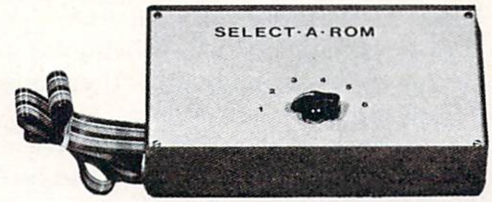
PROGRAM
COPY
VERIFY

Program you own single 5V supply EPROMS like 2716, 2532. Plugs directly into PET/CBM. Nothing else to buy completely assembled and tested including software. **$195.00**

ALSO EPROMS **2716 — $12.50 Each**
2532 — $28.00 Each

SHIPPING — ADD $5.00

MRJ
7951 No. 4 Road, Richmond, B.C.
Canada, V6Y 2T4
Telephone (604) 273-3651



SELECT — A — ROM

For the Commodore PET/CBM

With one rotary switch select 1 to 6 separate ROMS or EPROMS, without damaging your computer board or rom pins. Now you can use **Wordpro, Toolkit, Visicalc, Eproms** as **2716's** or **2532's** etc.

Assembled _________ $80.00
Kit_______________ $45.00

ROM SWITCH

A switch between old basic 2.0 and new 4.0 basic ROMS. Now you can utilize your computer with new and old software.

$125.00

VISA ORDERS — PHONE

(604) 273-3651 — JOHN (604) 273-3416 — JOHN
(604) 325-1122 — STAN

Progress
Computers

HARDWARE, SOFTWARE, PERIPHERALS
CALL (916) 722-2208

Call for price on...Commodore Computers, BEC Software Epson and TEC and Tally Printers

New Hardware

Sacramento, CA
(916) 722-2208

MFS-2 IEEE-488 To < RS232 Serial / 8 Bit Parallel — On ONE Board / Buy adding parts

The MFS-2 is addressable, fully 488 compatible, can mount inside the Commodore 8032, anywhere on the bus, or on the printer. Has an instruction book. Comes as a kit or factory assembled and tested.

	Price	
8 Bit Parallel	Kit 58.00	Tested 69
RS232	Kit 69.00	Tested 82

(cables not included)

Dealer Inquiry Welcome

New Software

File Cabinet 1.2 by Mike Kouri, an excellent data base with 24 pages of documentation, user defined files, can interface with word pro files. Does print labels and make mail or any kind of list. Has the fastest sort in microcomputing and uses very little ram.

$65 per copy

Business Check Book: Lets you know where your money came from and went. Prints several reports. This program does not work for inventory but is made for service business.

In Sacramento it's Progress Computers for Commodore Equipment

6962 Sunrise Blvd., Citrus Heights, CA 95610 (916) 722-2208 starts; you'll have to set them yourself.

If you'd rather do a true load — bringing in the next program and starting over — there are two methods you can use.

Method one uses the "dynamic keyboard" technique. You know that everything would be OK if you typed in your LOAD command and pressed RETURN ... so that's what you have the PET do. To load a disk program called WORK, you would code:

```
840 X$=CHR$(34)
850 PRINT "[clear, down3] LOAD";X$;
    "WORK";X$;",8'
860 PRINT "[down4] RUN [home]"
870 POKE 623,13:POKE 624,13:POKE 158,2
880 END
```

Note that we are using cursor movements here: on line 850, screen-clear followed by three cursor-downs; on line 860, four cursor-downs and later cursor-home. We are writing the LOAD and RUN instructions carefully onto the screen, and then forcing them to be executed by stuffing a couple of RETURN keystrokes into the keyboard input buffer.

The same techniques will work with cassette tape (change ",8" to ",1" in line 850). If you have Original ROM Basic, line 870 must change to: POKE 527,13:POKE 528,13:POKE 525,2.

Method two starts off with conventional chaining. The first statement in the newly loaded program is:

```
POKE 42,PEEK(201):POKE 43,PEEK(202):CLR
```

On Original ROMs, use:

```
X1=PEEK(229):X2=PEEK(230):POKE 124,X1:
  POKE 125,X2
```

Overlays

If a Basic program wants to load memory with data or programs, and these programs are so located that they won't interfere with Basic, the technique is called overlaying.

Normal chaining methods will do the trick nicely, with one minor exception: the BASIC program will go back to its first statement. This seems odd, but it's for a logical reason: the interpreter, seeing a LOAD, assumes that the BASIC program might be changed; on the assumption that a new program is in there, it starts executing from the beginning.

How do you handle this? Your program is zipping along at line 600, say, and decides to load in a piece of machine code; suddenly, you're back at the start.

The method is simple. As your first line of code, write something like:

```
10 ON Z GOTO 610
```

Now: when the program really starts, Z will be zero and you'll drop through this first line and continue with your program. But at line 600, you'll code: Z=1:LOAD "WHATEVER" ... and even though you'll go back to the start, line 10 will zip you right back to your working program at line 610 because of the value of Z. When you have multiple overlays, your first line will start to look like:

```
10 ON Z GOTO 610, 840, 1050, 1370
```

It's easy. By the way, be sure to check that you are loading your overlays into "safe memory." Cassette buffers are often fairly safe; if you use high memory, don't forget to adjust your top-of-memory pointers to protect your overlay programs.

Conclusion

Yes, you can run a 300K program on an 8K machine with overlays. It may take quite a bit of work to arrange it all, but the techniques are in place and waiting for you.

Converting Pet BASIC Programs To ASCII Files

Harvey B. Herman
Department of Chemistry
University of North Carolina
at Greensboro
Greensboro, NC 27412

Why A Terminal?

If one reads the ads in **COMPUTE!** and other hobbyist magazines you are sure to note a few which purport to convert a personal computer into a terminal. You may have wondered why anyone would want to communicate with a remote computer after spending so much for a stand-alone system. Well there can be advantages. Calling up a service, like The Source or CompuServe will allow you to get current information about Commodore stock (down as I write this) or about a late breaking news bulletin from the AP wire (trouble in Poland today). Calling up a big mainframe at work to make corrections on an errant statistical program will save a round trip and whatever gas that takes. A little reflection will enable you to come up with additional advantages.

Down-Loading

I recently discussed (**COMPUTE!**, Vol. 1, Issue 6, 1980) one of my frequent uses of PET as a terminal. The operation described in "Converting ASCII Files to PET BASIC", is sometimes called down-loading, that is, transferring a program saved on a large remote computer to a small one. What I did was establish contact with a remote computer using a modified PET terminal program. I then listed a BASIC program (written on the remote computer) on the screen while simultaneously saving the received ASCII characters to a reserved area in high memory. Contact with the remote computer was broken and a PET BASIC program (called ASCII) loaded and run normally. This program converted the saved ASCII file into a PET BASIC program which was then available for use on the PET.

Up-Loading

The opposite procedure is also very helpful. This process converting a PET BASIC program into an ASCII file and transferring this file to a remote computer using a terminal program, can be called up-loading. It works like this. First, load any PET BASIC program normally. Append (using TOOLKIT or the like) the BASIC program Disinterpreter, the listing of which is given in this article. Run the disinterpreter program (RUN 63000) whose function is to make an ASCII file in high memory of the first BASIC program loaded. Finally, load the terminal program, establish contact with the remote computer and transmit the ASCII file to it. Editing can be done "on the fly" with the disinterpreter program or by a text-editor on the big machine. Up-loading (or down-loading for that matter) saves a lot of time retyping BASIC programs for use on other machines. For example, I have had little trouble converting PET BASIC programs to work on a DEC VAX-11/780 machine (watch RIGHT$, however).

Program Disinterpreter

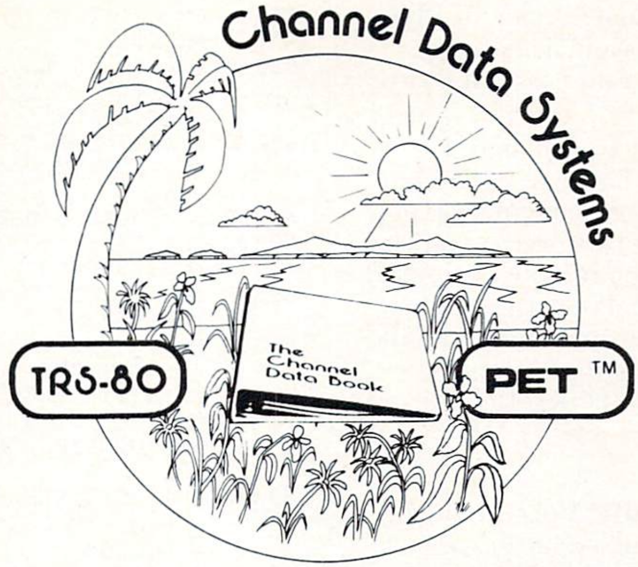
The program which makes an ASCII file (Disinterpreter) is a much expanded version of a preliminary one described in PET User's Notes (Vol. 1, issue 3). The program as listed is for "old" ROMs (version 2) but changes required for "upgrade" ROMs (version 2) are indicated (tested briefly). It can be difficult to follow a program written by someone else (an understatement). I have included remarks to help make the task easier. It will also be helpful for novices to review how PET programs are stored in memory (see PET User Manual, for example).

A number of points in the program need further elaboration:

1. This program was designed to be appended to programs which may or may not have data statements. Since PET BASIC does not have a RESTORE line number it was necessary to use a trick in lines 63190 to 63220. A unique data string ('end of data') was defined which stopped a dummy read loop. The data statement pointer was now positioned correctly for reading BASIC keywords into an array.

2. My intention was to send the ASCII files to VAX-11/780 mainframe. Several places in the program are specific for that application. For example, I could not use PET cursor controls in a VAX program so a space was substituted (line 63280 and lines 63330 to 63350) for cursor controls and unused tokens. A cursor control may be indicated if the quote mode flag is set (F = +1) and the peeked number is less than 32 or greater than 127. The program was *not* intended to be used in applications that have extensive graphics use but lower case will come out correctly with the original character generator ROM. I also substituted INPUT for GET as a BASIC keyword (see line 63690).

3. Old ROM/New ROM changes are given in lines 63150 to 63160 and 63380 to 63390. Also, add one more keyword ('GO') after MID$ in line 63730 and change 202 to 203 to 204 in lines 63310 and 63340,



OMNIFILE CBM or TRS-80 $30

Omnifile is a versatile, in-memory database program with sorting, formatting, and computational features. Records can be entered, edited, and processed with a single letter command. **Omnifile** applications include inventory records, mailing lists, sales journals and collection lists. Records can be stored on the Commodre floppy disks or on the tape cassette. **Omnifile** uses approximately 6k of RAM memory. Up to 500 records can be contained in memory in a 32k CBM at any time. Multiple files are easily accessed from disk or tape. Items can be sorted, moved, inserted and reformatted. Calculations can be made and totals can be printed. The **Omnifile** package includes the program with sample data, listing and manual, and will operate on the large keyboard Commodore PET or CBM computers with at least 16k memory. Also available on diskette for $36. An abbreviated version, **Data Logger,** requiring only 1k of RAM is available on cassette for $15.

GENERAL LEDGER/PERSONAL LEDGER CBM or TRS-80 $30

General Ledger is a complete double entry bookkeeping system with provisions for budgeting and keeping records of income, deductible and non-deductible expenses, assets and liabilities. Simple interactive features allow entering transactions, adding or editing accounts, and printing of a detailed income statement and balance sheets. Data can be stored on the Commodore floppy disks or cassette. **General Ledger** occupies about 6.2k of RAM memory, allowing approximately 200 accounts on a 16k machine. Transaction files can be accessed by our **Omnifile** database program for complete analysis, sorting by date, account number, etc. The **General Ledger** program will operate on the new Commodore PET or CBM microcomputer systems and comes with sample data, listing, and manual. Also available on diskette for $36. An abbreviated version allowing about 35 accounts on a 1.0 or 2.0 BASIC 8k PET is available on cassette for $20.

EXPLORE CBM $15

Inspired by the computerized fantasy simulation "Adventure," **Explore** is a conversational program which operates on the Commodore PET with only 8k bytes of memory. **Explore** contains four adventures in which you operate a computerized tank, hunt treasure in a magic cave, explore the mall in Washington D.C., and survive in a haunted castle. **Explore** package includes introduction, five data files, and complete manual. Available from Channel Data Systems on cassette for $15. Indication of old or new ROMs is requested.

CHANNEL DATA BOOK for PET CBM $20

A complete hardware and software reference service listing descriptions for over 1400 software programs and over 200 peripheral devices for PET. Also includes an information sources section, and addresses for Commodore Dealers in the USA and Canada, Commodore Vendors, and PET user groups. Designed to organize documentation, newsletters, listings, and other user selected information in an attractive 3-ring binder.

ACCESSORIES

PORT NOISE COMPLETE Speaker-amplifier for Commodore PET or CBM with connector (M & N pin) all ready to plug in—$20.

IEEE/USER PORT CONNECTOR 24 pin connector. With backshell—$7. Without backshell—$4.

C-10 TAPE CASSETTES High quality AGFA tape with screw type case. $10/10 or $75/100.

Cash, Checks, Money Orders, VISA, Mastercharge
Add $2 shipping for each order—COD-$3.50
For foreign orders please inquire for pricing

CHANNEL DATA SYSTEMS
4141 State Street Santa Barbara, CA 93110 805-964-6695

respectively. I made these changes on an 8K PET with upgrade ROMs and the original character generator ROM, and the program passed a short test.

4. As a convenience to the user I calculated the memory location in hex of the last byte (zero) sent to high memory. If the user wishes, memory from hex 4000 to last byte saved can be stored on tape or disk for transmission later to the remote computer. Furthermore the user's terminal program could key on the zero byte to stop transmission of the file automatically. The beginning of high memory ($4000 up) is set in lines 63160 (POKE 135,64) and 63240 (4*4096).

Summary

The programs ASCII (**COMPUTE!** Vol. 1, Issue 6, 1980) and disinterpreter (discussed here) are complimentary. They allow down-loading and up-loading of BASIC programs developed on one computer for use on another. If you have access to a remote computer, I urge you to purchase a terminal program which will allow you to communicate over telephone lines. It is a relatively inexpensive way to upgrade the capability of a personal computer. I have found it invaluable and I trust you will too.

```
READY.

 63ØØØ REM DISINTERPRETER
 63Ø1Ø REM
 63Ø2Ø REM APPEND TO BASIC PROGRAM
 63Ø3Ø REM MAKES AN ASCII FILE OF BASIC ¬
       ¬PROGRAM IN HIGH MEMORY
 63Ø4Ø REM
 63Ø5Ø REM HARVEY B. HERMAN
 63Ø6Ø REM
 63Ø7Ø REM ORIGINALLY FROM PET USER ¬
       ¬NOTES VOL 1 ISSUE 3
 63Ø8Ø REM EXPERIMENTAL DISINTERPRETER
 63Ø9Ø REM AUTHOR- WARREN D. SWAN
 631ØØ REM
 6311Ø REM 'OLD' ROMS. SEE REMS FOR ¬
       ¬UPGRADE CHANGES
 6312Ø REM
 6313Ø PRINT "ĥ↓↓↓↓↓↓                PLEASE ¬
       ¬BE PATIENT"
 6314Ø REM PROTECT HIGH MEMORY($4ØØØ UP)
 6315Ø REM UPGRADE-POKE 52,Ø:POKE53,64
 6316Ø POKE 134,Ø:POKE 135,64:CLR:F=-1:
       ¬REM QUOTE FLAG(MINUS-NO QUOTE)
 6317Ø POKE 59468,14
 6318Ø DATA "START OF DATA"
 6319Ø REM READ PAST DATA IN BASIC ¬
       ¬PROGRAM
 632ØØ FOR I=1 TO 1ØØØ:READ C$
 6321Ø IF C$="START OF DATA" THEN I=1ØØØ
 6322Ø NEXT I
 6323Ø REM ASCII FILE MADE FROM $4ØØØ UP
 6324Ø J=4*4Ø96
 6325Ø DIM C$(255):C$(Ø)=CHR$(13)
 6326Ø DIM D$(255):D$(Ø)=CHR$(13)
 6327Ø FOR I=1 TO 31:C$(I)=" ":NEXT I
 6328Ø FOR I=1 TO 31:D$(I)=" ":NEXT I
 6329Ø FOR I=32 TO 127:C$(I)=CHR$(I):
       ¬NEXT I
 633ØØ FOR I=32 TO 127:D$(I)=CHR$(I):
       ¬NEXT I
 6331Ø FOR I=128 TO 2Ø2:READ C$(I):
       ¬C$(I)=" "+C$(I)+" ":NEXT I:
       ¬REM SPACES FOR VAX
 6332Ø C$(127+5Ø)=">":C$(127+51)="=":
       ¬C$(127+52)="<":REM NO SPACES FOR ¬
       ¬VAX
 6333Ø REM MAKE UNUSED TOKENS INTO SPACES
 6334Ø FOR I=2Ø3 TO 254:C$(I)=" ":NEXT I:
       ¬C$(255)=" "
 6335Ø FOR I=128 TO 254:D$(I)=" ":NEXT I:
       ¬D$(255)=" "
 6336Ø FOR I=193 TO 22Ø:D$(I)=CHR$(I):
       ¬NEXT I:REM ALLOW FOR LOWER CASE
 6337Ø PRINT "ĥ":REM START DISINTERPRETIN
       ¬G
 6338Ø REM UPGRADE-PEEK(42)+256*PEEK(43)-
       ¬3
 6339Ø L=1Ø25:U=PEEK(124)+256*PEEK(125)-4
       ¬:REM MEM POINTER AND END OF ¬
       ¬PROGRAM
 634ØØ Q=PEEK(L)+256*PEEK(L+1):REM LINK
 6341Ø N$=STR$(PEEK(L+2)+256*PEEK(L+3)):
       ¬REM STATEMENT NUMBER
 6342Ø REM
 6343Ø REM WHEN STATEMENT 63ØØØ REACHED ¬
       ¬STOP AND FILL NEXT MEM. LOC. WITH ¬
       ¬ZERO
 6344Ø REM CONVERT TO HEX AND DISPLAY ¬
       ¬FOR REFERENCE
 6345Ø IFVAL(N$)>=63ØØØTHENPOKE J,Ø:
       ¬PRINT"ZERO AT LOCATION $";:N=J:
       ¬GOSUB 6374Ø:END
 6346Ø FOR I=2 TO LEN(N$)
 6347Ø CH$=MID$(N$,I,1):PRINT CH$;:
       ¬POKE J,ASC(CH$):J=J+1:NEXT I
 6348Ø PRINT CHR$(32);:POKE J,32:J=J+1
 6349Ø FOR I=L+4 TO Q-2
 635ØØ P=PEEK(I):REM P=32 IF QUOTE
 6351Ø REM F<1 IN QUOTE MODE, P=32 START ¬
       ¬QUOTE MODE
 6352Ø IF F<1 OR P=34 THEN S$=C$(P):
       ¬GOSUB 6359Ø:GOTO 6354Ø
 6353Ø S$=D$(P):GOSUB 6359Ø
 6354Ø IF P=34 THEN F=-F:REM TOGGLE ¬
       ¬QUOTE MODE
 6355Ø NEXT I
 6356Ø REM RESET QUOTE MODE,POKE CR,
       ¬LF AND CHECK FOR PROGRAM END
 6357Ø F=-1:L=Q:PRINT:POKE J,13:POKE J+1,
       ¬1Ø:J=J+2:IF L<U THEN 634ØØ
 6358Ø END:REM PROBABLY DON'T NEED THIS
 6359Ø FOR K=1 TO LEN(S$)
 636ØØ CH$=MID$(S$,K,1):IF F<1 THEN IF ¬
       ¬CH$=":" THEN CH$="\":REM FOR VAX
 6361Ø REM PRINTS TO SCREEN AND POKES ON ¬
       ¬HIGH
 6362Ø REM NOTE CONVERSION FROM PET ¬
       ¬ASCII IF ASC>=193 ("A") (TRUE = ¬
       ¬-1)
 6363Ø PRINT CH$;:POKE J,ASC(CH$)+(ASC(CH
       ¬$)>192)*96:J=J+1
 6364Ø NEXT K
 6365Ø RETURN
 6366Ø DATA END,FOR,NEXT,DATA,INPUT#,
       ¬INPUT,DIM,READ,LET
 6367Ø DATA GOTO,RUN,IF,RESTORE,GOSUB,
       ¬RETURN,REM,STOP
 6368Ø DATA ON,WAIT,LOAD,SAVE,VERIFY,DEF,
       ¬POKE,PRINT#
 6369Ø DATA PRINT,CONT,LIST,CLR,CMD,SYS,
       ¬OPEN,CLOSE,INPUT:REM GET
 637ØØ DATA NEW,TAB(,TO,FN,SPC(,THEN,NOT,
```

```
         ¬STEP,+,-,*,/,^
 63710 DATA AND,OR,>,=,<,SGN,INT,ABS,USR,
         ¬FRE,POS,SQR
 63720 DATA RND,LOG,EXP,COS,SIN,TAN,ATN,
         ¬PEEK,LEN,STR$
 63730 DATA VAL,ASC,CHR$,LEFT$,RIGHT$,
         ¬MID$
 63740 REM DECIMAL TO HEX CONVERSION-JIM ¬
         ¬B. IDEA
 63750 REM
 63760 X=N/4096:FORJ=1TO4:A=INT(X)
 63770 IFA>9THENPRINTCHR$(A+55);:
         ¬GOTO63790
 63780 PRINT CHR$(A+48);
 63790 X=(X-INT(X))*16:NEXTJ:PRINT:RETURN
READY.
```

©

Using *your* computer in an interesting application? Write it up for other **COMPUTE!** readers to use.

The Single Board 6502

Eric Rehnke
Anaheim, CA

What's New At MTU

Thought you'd like to hear a little about two of the latest software releases from Micro Technology Unlimited (POB 12106, Raleigh, NC 27605). This isn't really a product review, just an update. A review on both these products will follow after I've had a chance to use them awhile.

One of them, The Keyword Graphics Package, is a machine language program which interfaces AIM 65 BASIC to the MTU Visible Memory board (a 320 x 200 bit mapped graphics display) and actually extends BASIC with over 45 graphics-oriented commands to make programming the graphics display akin to programming the APPLE II Hires graphics — but lots more powerful!

According to the documentation, many things are now possible with the Visible Memory such as having plotting graphs and other images from mathematical functions or measured data; configuring up to four independent text or graphic "windows" on the screen; character scaling and rotation; the ability to store and display up to 255 user defined "shapes" each of which can be drawn anywhere on the screen; etc., etc. Lots of features.

Sounds really neat. Trouble is, I don't have enough memory on my AIM 65 at this moment so will have to wait a little while to try this package out. But, the point is, users of MTU hardware now appear to need very little knowledge of assembly language programming to get full use out of their MTU peripherals.

The second package from MTU is the Instrument Synthesis Package. This software can be used on AIM, KIM, SYM, or PET computers and gives those folks with some music ability (I'm not in that group yet, unfortunately) the tools to compose songs with up to four voices and a great degree of control over the tones synthesized. You can fiddle with the waveforms to mimic almost any musical instrument you could imagine. (You can even "invent" your own instrument sound. MTU sells a demo tape that can be played on your stereo and gives you a good idea of the kinds of sounds you can expect to produce. If you're in doubt, buy the demo tape before you take the plunge.)

I have the package running on my KIM and am really taken with the different kinds of sounds my computer can make. And it's all done in software, no less.

Now, the user interface to this music package hasn't quite been developed to the extent that the graphics have, but it's definitely more advanced than the earlier MTU four part harmony software.

If we can characterize a complete music composing/playback system as consisting of two parts — the user interface (graphic and/or joystick input package), and the actual music interpreter that makes the music from the data assembled by the user interface — then the MTU Instrument Synthesis Package can be thought of as the inner interpreter. The user interface still needs to be written. MTU says they are working on such a user interface and suggest that, in the interim, a simple music "compiler" can probably be written using BASIC or using macros (if you have a macro assembler).

The "interpreter" actually executes opcodes from memory in the same manner that your 6502 executes machine language opcodes from memory. So at this point, even though you would need to go through about the same sort of programming exercise you had to go through before you had an assembler, the operation codes for the music interpreter are considerably more powerful in function than those for the 6502 and can be viewed in the same manner as tokens in the BASIC language.

The documentation is quite informative on just how to use the package and presents a musical sound theory section for those of us needing a little brush-up in physics. One of the demo songs that comes on the tape is listed in "source" format as an example of how to write a song "program." Software tools are included in the package to ease the task of constructing instrument waveforms and there's even a library of pre-defined instrument definitions to help you get going. Lots of good material for experimentation.

No, the MTU software doesn't enable your computer to sound *exactly* like a guitar, or a piano, or a trumpet. But, it comes pretty close and is quite a bit less expensive than the hardware-based instrument synthesizers I've seen. (And they don't sound all *that* much better.)

The first thing I did after loading the Instrument Synthesizer into my machine and running it to make sure it all worked was to see how I could interface it to my HDE disk system. Because of the way the software is configured, that turned out to be very easy.

There are no system dependencies in the interpreter itself. All system specific data is contained in what's known as a CONFIGURATION STRING and the address of this string of data gets passed to the interpreter through the X and Y registers when called.

The following bit of code replaces the configuration string, the stop key test, and the calling routine which normally reside at \$0200-\$0234 in the KIM version of the software. The major differences in my interface are that I assume there is an ASCII terminal hooked up, and I return to my operating system (HDE) by way of the BRK opcode.

If you wish to interface the MTU package to other than HDE disk systems, you can use my code as a guide. Thanks to the folks at MTU for making things so easy.

I'm just beginning to understand just how sophisticated this music package really is so I'll have more for you later after I've had a chance to play with it awhile.

What The Bell?

Have any of you come up with any interesting sounds for the AY3-8910 Programmable Sound Generator yet? Lately, I've been trying to figure how to program it to sound like the bells you might hear in some far eastern temple.

I imagine there are some folks out there really getting into this chip. How about sharing some of that hard won information with the rest of us?

The AY3-8910 can also be used as a waveform generator in an electronic organ or synthesizer which needs a much more accurate tone generator than is possible with the PSG.

I would also like to design an electronic music box for my baby-to-be (we're expecting around the end of August). (Wonder if I'll have time for all those projects?) ©

Nuts and Volts
Build Your Own Controllers: Part II

Gene Zumchak

In my last column, I suggested that all it took to make a budget development system for bringing up controllers was a computer system (as little as a KIM will do), an EPROM programmer, and the key ingredient, an EPROM emulator.

An EPROM emulator (called a simulator by some) is a particular example of a two-port RAM. A two-port RAM can be independently accessed (addressed) by two separate systems (though not simultaneously). Two-to-one selector gates are used to apply one of the two sets of addresses. For an EPROM emulator, one of the sets of addresses is the host computer system. The other set of addresses comes from a socket that is wired to look like the particular EPROM being emulated. The RAM data is bidirectionally buffered to the host system, and unidirectionally to the "emulate" socket, since the EPROM is read-only. Access to the RAM is determined by a manual switch. A block diagram of an EPROM emulator is shown in Figure 1.

With the switch in the "Master System" position, the RAM is addressed by the host system just like any other RAM in the system. Correspondingly, it can be loaded with a program using the host system's operating system, preferably using an assembler. Similarly, any program in emulator RAM can be saved with the host system's tape or disk facility.

With the switch in the "Emulate" position, the RAM is addressable via the EPROM socket on the emulator. Connection to the controller board under development (often called the target system) is via a ribbon cable with a 24-pin DIP plug on both ends. The target system thinks that it has an EPROM in its socket, when it is really looking at the emulator RAM. This connection is shown in Figure 2.

Typically, the host system is used to put a program in the emulator RAM. Then the emulator is switched to the emulate mode and an attempt is made to run the program in the target system. If the program needs modification, the RAM is switched back to the host and changes made to the program. Changes can be made and tried out very rapidly, with dozens of changes made per hour. This contrasts sharply with the method of having to burn in a new EPROM after each and every change.

While it is clear that the emulator is practically indispensable for developing target system software, it is equally useful for completely checking out the target system hardware *before* any attempt is made to debug the software. This important step is often omitted and much time is needlessly wasted trying to track down problems in the software that don't exist. The use of an emulator to check out hardware will be amplified later.

The Design

The design of an EPROM emulator is relatively straightforward. The size of the emulator RAM will depend upon the EPROMs to be emulated. Althugh the 2K 2716 is the present standard, with 2732s dropping below $20, it would be foolish not to provide for at least 4K of RAM. If 4K of RAM is to be supported, the designer may wish to consider having two emulate sockets, to emulate two 2716s simultaneously. The design should take care to prevent the host system from writing into emulate RAM when it is in the emulate mode.

The heart of the design (no need to show) is a block of RAM. The low-cost, and readily available, 2114 types will do fine. Organized 1K x 4, eight chips will provide 4K of RAM. In the figures which follow the circled A's are the address lines of the RAM block, the circled D's are the data lines, and chip selects are shown for four 1K pairs of 2114's.

Figure 3 shows the "emulate" socket which is used to interface the emulator to the target system. The emulate socket or target system addresses are labeled with circled lower case a's. The target system data bits are labeled with circled lower case d's. Note that for all the popular 2K and 4K (5 volt only) EPROMs, 21 of the 24 pins are identically defined. Three pins, pin 18, pin 20 and pin 21, will vary according to EPROM type. This variation is accommodated with an 8-pin DIP jumper socket. The two outputs of the socket are "a"11 and S (Select). The later signal is used to gate the RAM contents to the emulate socket data pins. The customizing socket is shown jumpered for a 2532. For this chip, pin 20 serves as the chip select and should be jumpered to S. Pin 18 provides A11 and should be jumpered to "a"11. Thus the same emulate socket can be customized for any EPROM type with the associated jumper socket. For 2716s, usually either CS or OE is used for ROM selection and the other pin is just permanently grounded. If both are required and are independent, an external OR gate will be required to combine them into the single S signal.

Interfacing of RAM data to the target and host systems is shown in Figure 4. An octal three-state gate chip such as the 74LS244 is used to buffer the data to the emulate socket. The chip select signal from the target system EPROM socket provides the enable (S) for the chip. The RAM is bidirectionally buffered to the master or host system. A 74LS245 is shown. Other bidirectional or three-state buffers could be used.

Figure 5 shows address selection for the RAM. Common 74LS157 quad 2-to-1 selector gates are used. The "A" inputs are the target addresses from the emulate socket. The "B" inputs are the addresses from the master system. Selection is made with a manual switch. In the closed position, the target addresses are chosen. An LED can be used to indicate the switch position and corresponding mode. The ten low selected address bits go to all 1K RAM pairs. The two high bits, A10 and A11 go to any 2-to-4 or 3-to-8 decoder to provide individual chip select signals for the RAM pairs. With this scheme, one 1K block of RAM is enabled at all times. It is therefore necessary that the write enable signal for the RAM be gated with master system address selection so that RAM is written to only when specifically desired by the master system. Note that all the selected address bits may be optionally monitored with an LED as shown for bit A0. This feature will prove to be so valuable that it probably should not be considered an option.

Master system address selection and RAM write enable generation is shown in Figure 6. A 74LS85 4-bit comparator is used to compare the high four address bits of the master system, MA12 to MA15, with the settings of a 4-bit DIP switch (or jumpers) to place the emulator RAM into any 4K block of the master system. The output of the comparator is combined with the R/$\overline{W}$ signal and the Φ signal to provide a write enable (strobe) for the RAM which can occur only when selected. The output of the comparator is also inverted to provide a low-true enable for the data bus buffer shown in Figure 4. Note that the comparator output is disabled by the $\overline{E}$ signal when the mode switch is in the emulate position, preventing erroneous writes by the master system.

A design like the one just suggested will take eight 2114 RAM chips, ten support chips, some resistors, LEDs, and a few .1 mFd. decoupling capacitors. Before loading your breadboard with chips and trying it out with your system, carefully ring out all of the wiring with a continuity tester. This "extra" step will save you a lot of time. Verify that it works as RAM in your master system first. If you have any problems you can test it DC-wise by applying power to it without connecting it to your system. As it sits there, all address lines will be open and appear as ones. To select the RAM, you will only need to bias the top four address lines, grounding any address lines that should be zero, and leaving open any lines that should be ones. Similarly, you can control the Φ2 and R/$\overline{W}$ lines and trace all signals with a voltmeter.

Using The Emulator

Assuming that you have a working emulator, you can now use it with your system to bring up another microcomputer system. The first step is to verify that all the target system hardware works. While there is a temptation to load the target system with chips and try it out, a better approach is to add chips and test them a function at a time.

Start by adding only those chips necessary to run the simplest program. This will be the processor, address decoding chips, and any gates used to buffer or generate any control signals required. Let us assume that our controller has a 2K EPROM which is decoded to fall at $2000. The response to the reset address $FFFC should occur at $27FC in your EPROM. The first thing that should be done, then, is to use your system to install the reset vector at $27FC and $27FD. For testing purposes, that will probably point to $2000. Now when the target system is reset, it will try to execute a program at $2000. Note carefully, that the addresses to which the emulator RAM responds in the target system are unrelated to the addresses at which the RAM responds in the master system.

For testing, always use the simplest program that will do the job, since the object is to test hardware and not software. The first job is to verify that the target processor can run a program. That's easy. Just enter:

```
$2000  4C 00 20        LOOP   JMP LOOP
```

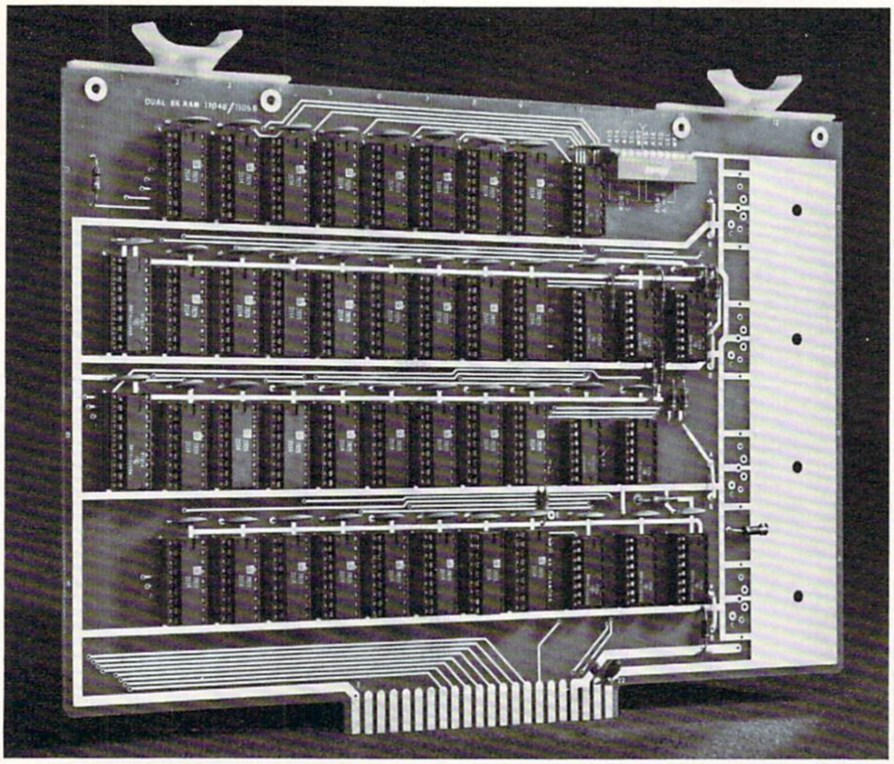
If your controller can run this program, you're on your way. But how do you know that the program is indeed running, since it doesn't do anything? That's easy, just look at the row of LEDs which monitor the selected emulator address. All LEDs but the least two, A1 and A0, should be dark. These should be glowing since the address in the target system is cycling $2000, $2001, $2002, $2000, etc., as the JMP instruction is executed. The LEDs are similar to those found on the front panel of a minicomputer. They allow us to "see" where in the EPROM space the program is running, (if it is running at all). When the processor has "bombed," all the LEDs will glow with equal brightness.

Note that the LEDs are monitoring *all* activity of the address bus, which includes references to memory and I/O locations. The following program cycles through addresses $2000 thru $2005. Every seventh cycle, however, the address $0180 will appear on the address lines and the LEDs for A8 and A7 will glow faintly.

```
$2000  8D 80 01        LOOP STA  $0180
$2003  4C 00 20             JMP  LOOP
```

Next, you might want to test RAM. Plug in the RAM, and use the emulator to enter a small program that writes a specific byte of data to a RAM location (or all RAM locations), and then reads the data back, comparing it with what was sent. The success or failure of this test can be noted by sending the program to one of two distinct loop locations, depending on the outcome of the test. The loop address will glow on the LEDs and you will be able to see the result.

VAK-2 8K STATIC RAM BOARD
VAK-4 16K STATIC RAM BOARD



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:

VAK-4	$\$325^{00}$
VAK-2	$\$239^{00}$

Add $\$4^{50}$ for UPS Shipment in continental U.S. All others call or write for shipping charges

The VAK-2/4 was specifically designed for use with the KIM-1, SYM-1 and the AIM 65 Microcomputer Systems. The VAK-4, 16K Ram Board, consists of two (2) separate 8K blocks. Each block has it's own address, write protect and block enable switches.

The VAK-2, 8K Ram Board, is identical to the VAK-4 with sockets for all 16K of Ram, but it has only one of the 8K blocks populated with IC's. Therefore, the VAK-2 is user expandable to a full 16K with the purchase of the VAK-3 Expansion Kit.

Both the VAK-2 and the VAK-4 Boards are made with 1st quality, Industry Standard 450 *nsec.* 2114 RAM Chips. They plug directly into the VAK-1 Motherboard, or with addition of voltage regulators plug into the KIM-4* Motherboard.

SPECIFICATIONS:

- Completely assembled, tested and burned-in.
- All IC's are in sockets
- Fully buffered address and data bus
- Standard KIM-4* Bus (both electrical Pin-out and card size)
- Designed for use with a regulated Power Supply such as our VAK-EPS, but has provisions for adding regulators for use with an unregulated Power Supply.
- Each 8K Block Address is independent and switch selectable.
- Separate write-protect switch for each 8K block.
- Board size: 10 in. Wide x 7 in. High (including card-edge)
- Power requirements: VAK-2—5V.DC @ 1.2 AMPS.
  Power requirements: VAK-4—5V.DC @ 2.4 AMPS.

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

We have moved to a new, larger facility. Please make note of our new address.

RNB ENTERPRISES INCORPORATED

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

master charge

VISA

If some hardware element does not operate as expected, it's very easy to observe what is happening with a scope, as long as the program is in a very tight two to five instruction loop. Using the emulator in this way, it is relatively routine to completely check out all of a controller's hardware before any attempt is made to run and debug any complex programs.

The development of software with the emulator can be accelerated if your target controller has an eight bit output port. This can be used to drive eight LEDs which can serve to indicate the result of any program. The use of an EPROM emulator and a latched LED display to debug prototype hardware and develop target system software is detailed in my book *Microcomputer Design and Troubleshooting*.

It should be clear that, with an EPROM emulator, any microcomputer that has an operating system that can read and write, load and store RAM, can serve as a very adequate development system for bringing up dedicated controllers. In a future column, we'll detail a design for a general-purpose controller board.

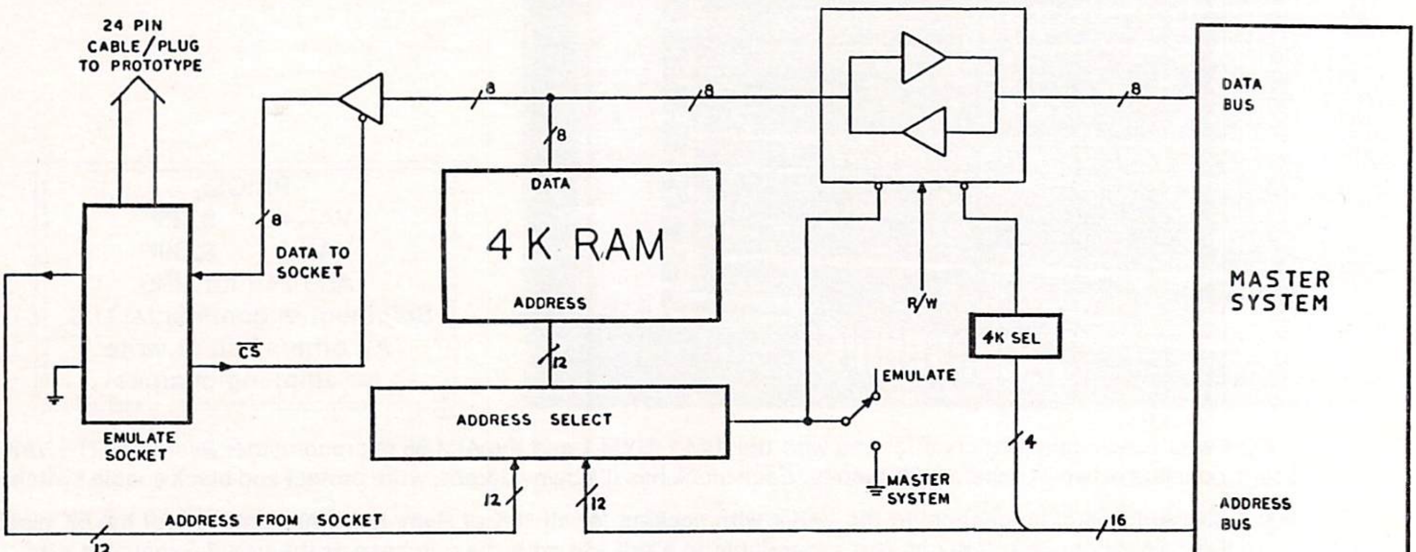


Figure 1. Block Diagram of a 4K EPROM Emulator

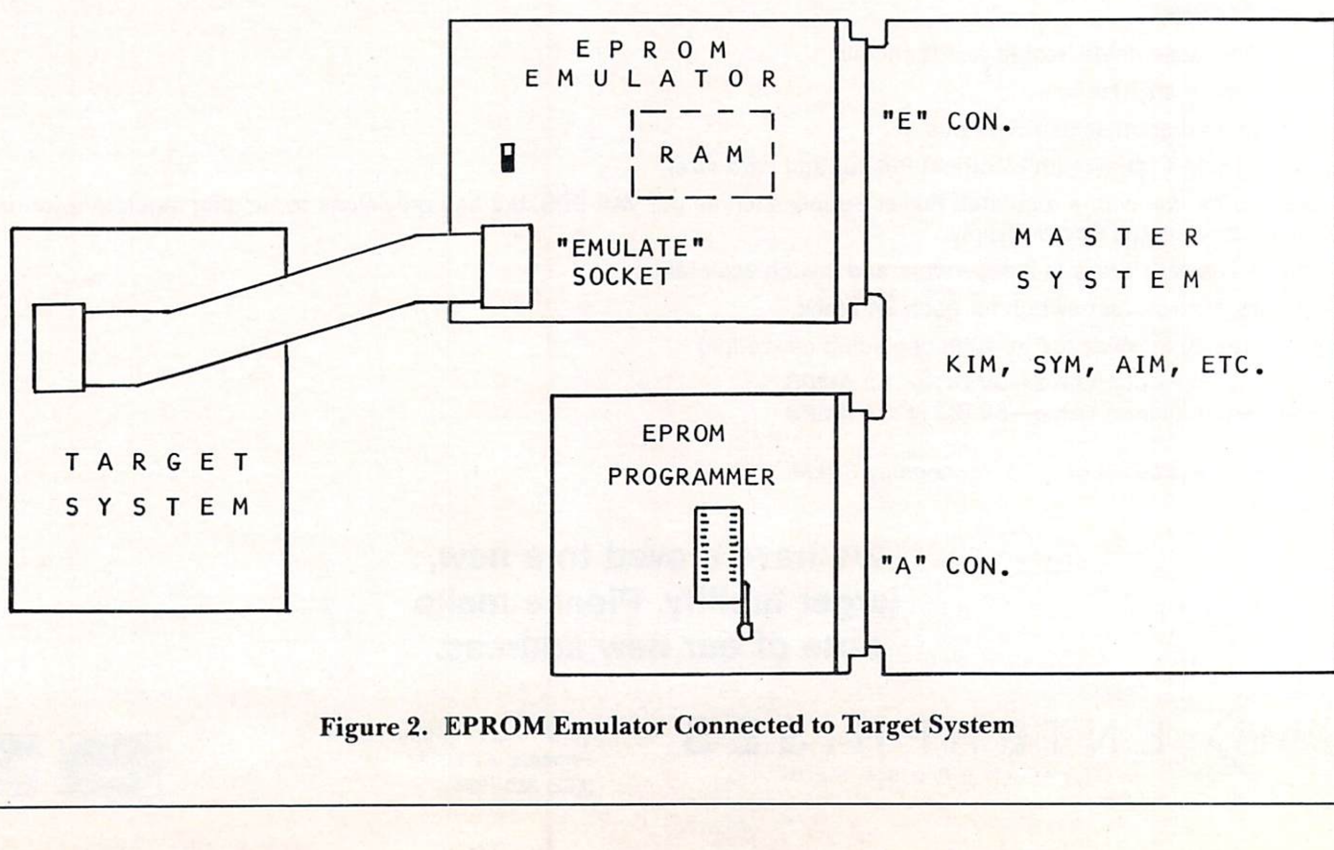


Figure 2. EPROM Emulator Connected to Target System

Photo credit: GREAT GALAXY IN ANDROMEDA: Palomar Observatory, California Institute of Technology

THE MTU FLOPPY DISK CONTROLLER WITH 16K RAM GIVES YOUR AIM-65 ION DRIVE POWER!

HARDWARE

- 16K 2 PORT RAM ONBOARD WITH WRITE PROTECT
- USES THE NEC-765 DISK CONTROLLER CHIP
- ROM BOOTSTRAP LOADER SPEEDS LOADING
- DMA OPERATION ALLOWS INTERRUPTS
- SUPPORTS 8 INCH DRIVES 1 OR 2 SIDED
- MAXIMUM STORAGE IS 4 MEGABYTES
- ANALOG PLL DATA SEPERATOR

SYSTEM FEATURES

- FORMAT UTILITY LOGS OUT DEFECTIVE SECTORS
- DISK/FILE COPY WITH WILDCARD SELECTION
- SYSTEM CUSTOMIZATION UTILITY
- VISIBLE MEMORY TERMINAL DRIVER PROVIDED
- INTERCHANGE CODOS SOFTWARE AMONG KIM, SYM, AIM, PET SYSTEMS
- IN FIELD USE FOR OVER 6 MONTHS

CODOS SOFTWARE

- CODOS DISK OPERATING SOFTWARE
- 8K RAM RESIDENT ALLOWS UPGRADES
- FINDS AND LOADS 32K BYTES IN 3 SECONDS
- STARTUP FILE EXECUTES AT BOOT-UP
- COMMAND FILE EXECUTION FROM DISK
- DYNAMIC DISK STORAGE ALLOCATION
- DEVICE-INDEPENDENT I/O
- TRUE RANDOM ACCESS TO RECORD IN ONE ACCESS
- MONITOR WITH 29 BUILT-IN COMMANDS
- FULL ENGLISH ERROR MESSAGES
- FILE NAMES 12 CHARACTERS + EXTENSIONS
- FILE SIZE UP TO 1 MEGABYTE
- UP TO 247 FILES PER DISK DRIVE
- INDIVIDUAL WRITE PROTECT ON FILES
- WORKS WITH AIM EDITOR, ASSEMBLER, BASIC AND MONITOR ROMS
- SUPERVISOR CALLS AVAILABLE TO USER PROGRAM

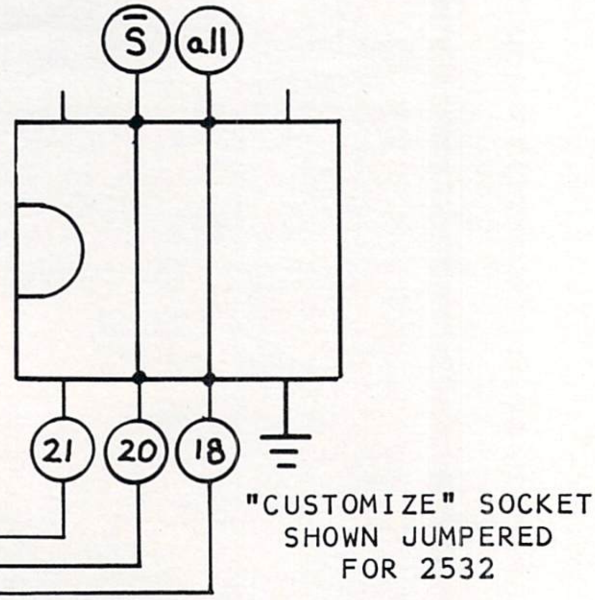
K-1013M Hardware Manual-$10, K-1013-3M CODOS manual-$25, K-1013-3D RAM/Disk controller with CODOS-$595, Floppy drives, cables, power supply also available.

MASTERCARD & VISA accepted

WRITE OR CALL TODAY FOR OUR 48 PAGE FALL 1980 CATALOG DESCRIBING ALL MTU 6502 PRODUCTS, INCLUDING 320 BY 200 GRAPHICS, AIM GRAPHIC/TEXT PRINT SOFTWARE, BANK-SWITCHABLE RAM/ROM/I-O, AIM CARD FILE, POWER SUPPLY AND MORE!

Micro Technology Unlimited • 2806 Hillsborough St. • P.O. Box 12106 • Raleigh, N.C. 27605 • (919) 833-1458

TYPE	21	20	18
2732	A11	$\overline{OE}$	$\overline{CE}$
2532	+5	$\overline{CS}$	A11
2716	+5	$\overline{OE}$	$\overline{CS}$
2516	+5	$\overline{CS}$	GND



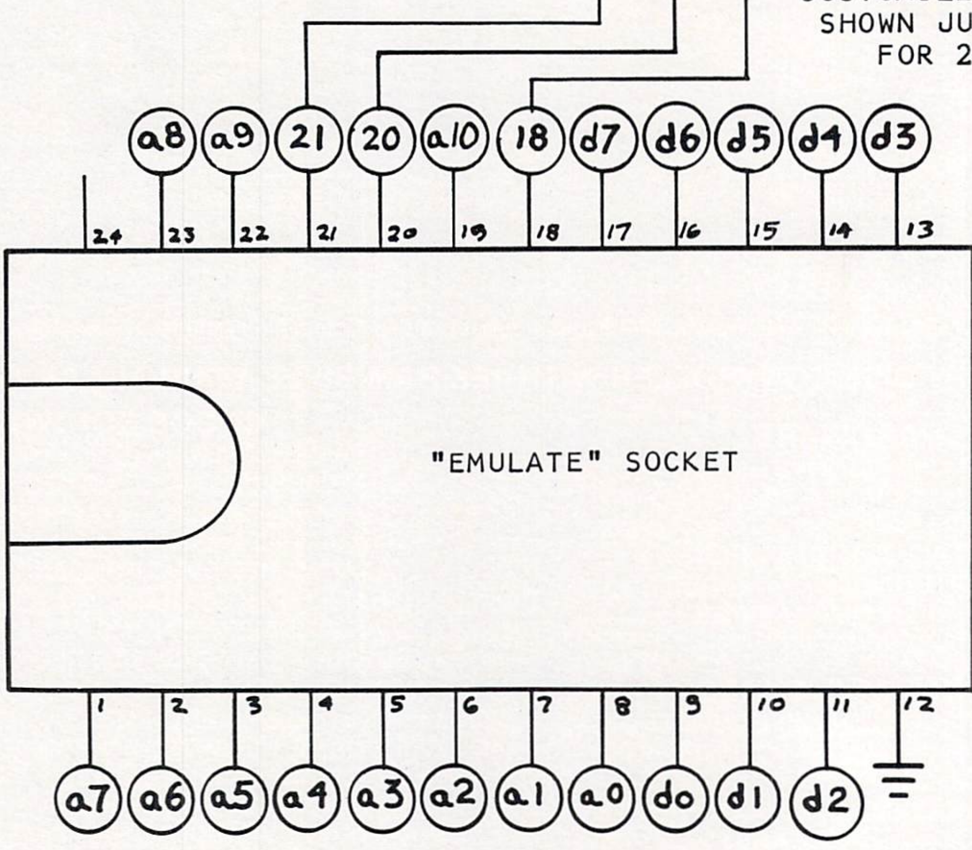


Figure 3. "Emulate" Socket

— AIM 65 —

P/N		QTY 1-9
A65-1	AIM-65 w/1K RAM	**$399**
A65-4	AIM w/4K RAM	**$439**
A65-A	Assembler ROM	**$ 85**
A65-B	BASIC ROMS	**$100**
A65-PL	PL/65 ROMS	**$125**
A65-F	FORTH ROMS	**$150**

SPECIALS

A65-4AB	**AIM-65 w/4K RAM, Assembler & BASIC**	**$599**
A65-4B	**AIM-65 w/4K RAM, BASIC**	**$529**

Power Supplies (AIM-65 Compatible, Industrial Quality Open Frame)

PRS3	+5V at 3A, +24V at 1A w/mtg hardware, cord, etc	**$ 75**
PRS4	+5V at 2A, +24V at .5A w/mtg hardware, cord, etc	**$ 60**
PRS5	+5V at 2A, +24V at .5A, ± 12V to ± 15V at .4A	**$ 75**
PRS6	+5V at 3A, +24V at .5A, + 12V at 2A	**$ 85**

From The Enclosure's Group

ENC1	AIM-65 case	**$ 45**
ENC1A	AIM-65 case w/space for one expansion bd	**$ 50**

Cases With Power Supplies (for ENC1A Add $5)

ENC3	ENC1 w/PRS3 mounted inside	**$125**
ENC4	ENC1 w/PRS4 mounted inside	**$110**
ENC5	ENC1 w/PRS5 mounted inside	**$125**
ENC6	ENC1 w/PRS6 mounted inside	**$135**

From Optimal Technology

ADC1	A/D eight channels, D/A 2 channels, requires ± 12V to ± 15V at 100 MA & 2-I/O Ports from AIM-6522	**$126**
	w/Cable for AIM-65	**$150**

From The Computerist

MCP1	Mother Plus™ Dual 44 pin mother card & card cage, fully buffered, 5 expansion slots underneath the AIM	**$150**
MEB1-2	DRAM Plus™ 32K RAM, 16K PROM sockets, 2-6522 I/O chip and programmer for 5V EPROMS	**16K RAM $325**
		32K RAM $395
PTC1	Proto Plus II™ Prototype card same size as KIM-1, MEB1-2, VIB1 (Bare Bd $60) assembled	**$ 75**
VIB1	Video Plus™ bd with 128 char, 128 user char, up to 4K display RAM, light pen and ASCII keybd interface	**$325**
CBL1	CABLE for MEB1-2, VIB1, PTC1	**$ 25**

From Seawell MicroSystems

MCP2	Little Buffered Mother™ Single 44 pin (KIM-4 style) mother card. Has on bd 5V regulator for AIM-65, 4 expansion slots. Routes A&E signals to duplicates on sides with 4K RAM	**$199**
MEB2	SEA 16™ 16K static RAM bd takes 2114L with regulators and address switches	**$250**
MEB2-3	CMOS RAM, realtime clock, EPROM bd, up to 8K RAM, 16K EPROM. (w/1k CMOS, 7K NMOS)	**$395**
PGR2	Programmer for 5V EPROMS with ROM firmware, regulators, low force sockets, up to 8 EPROMS simultaneously, can execute after programming	**$299**
PIO2	Parallel I/O bd with 4-6522's	**$260**
PTC2	Proto/Blank™ Prototype card that fits MCP2	**$ 49**
PTC2A	Proto/Pop™ with regulator, decoders, switches	**$ 99**
FDC2	Floppy disk controller bd & DOS, up to four 8" drives, double sided, double density (DD/DS)	**$425**
SBC2	SBC/CPU card, 9K RAM, 18K EPROM, 3 serial ports, 1 parallel port, audio tape interface	**$495**

From Micro Technology Unlimited (MTU)

DAC3	8 bit DAC bd	**$ 49**
FDC3	Floppy disk controller bd & DOS, up to four 8" drives, double sided, double density, 16K DRAM, Boot PROM	**$595**
MCP3	Card file w/4 slot expansion mother bd w/keybd brackets	**$ 85**
MEB3-2	Banker Board™, low power, 32K DRAM	**$450**
PIO3	24K PROM, 4-8 bit I/O ports w/RS-232 port to 4800 bps, PROM Programmer.	**$295**
VIB3	8K DRAM bd, low power, w/composite video out in 200 lines 320 dot/line format	**$240**
PTC3	Prototype Bd w/regulators	**$ 42**
MPS3	AIM-65 Power Supply w/12V for MTU Bds, can drive one 8" disk drive.	**$ 65**
CBL3	Cable for MEB3-2, VIB3, FDC3	**$ 25**

All MTU Software Available For These Products

Miscellaneous

TPT3	Approved Thermal Paper Tape, 3/165' rolls	**$7.50**
MEM6	6/2114 RAM Chips	**$ 45**
CAS1	Audio Cassette Recorder	**$ 40**
CAS1-1	CAS1 w/cable	**$ 65**
2716	16K 5V EPROM	**$ 10**
2532	32K 5V EPROM	**$ 35**
A65-P	Printer	**$ 70**
A65-DM	Display Module-DL1416	**$ 30**
FDD8	QUME Data Trak 8, DD/DS Disk Drive/up to 1 Megabyte	**$650**
FD8C-1	Cable set for 1 drive w/AC cord	**$ 50**
FD8C-2	Cable set for 2 drives w/AC cords	**$ 65**
MON1	9" composite video monitor w/80 char line resolution. Requires 12V DC only at .8A	**$125**

All AIM-65 Spare Parts Are Available
All "Microflex" Products Available
COMING SOON!
AIM-65/40 w/32K RAM- $1995

ASSEMBLED & TESTED SYSTEMS

We have been specializing in assembled and tested systems made from the above items for over 2 years. Normally, the price will be the total of the items, plus $5 for handling; shipping is extra on all COD's or invoiced orders. Please call or write for exact prices or if questions arise. **Six month warranty on all systems.**

Mail Check or Money Order to:

Educational Computer Division
EXCERT INCORPORATED

- SALES
- SERVICE
- INSTALLATION
- CONSULTING

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

Higher quantities quoted upon request, COD's accepted, shipping will be added. Add $5 for shipping, insurance, and handling on prepaid orders. Minnesota residents add 4% sales tax. Prices subject to change without notice.

April 1, 1981

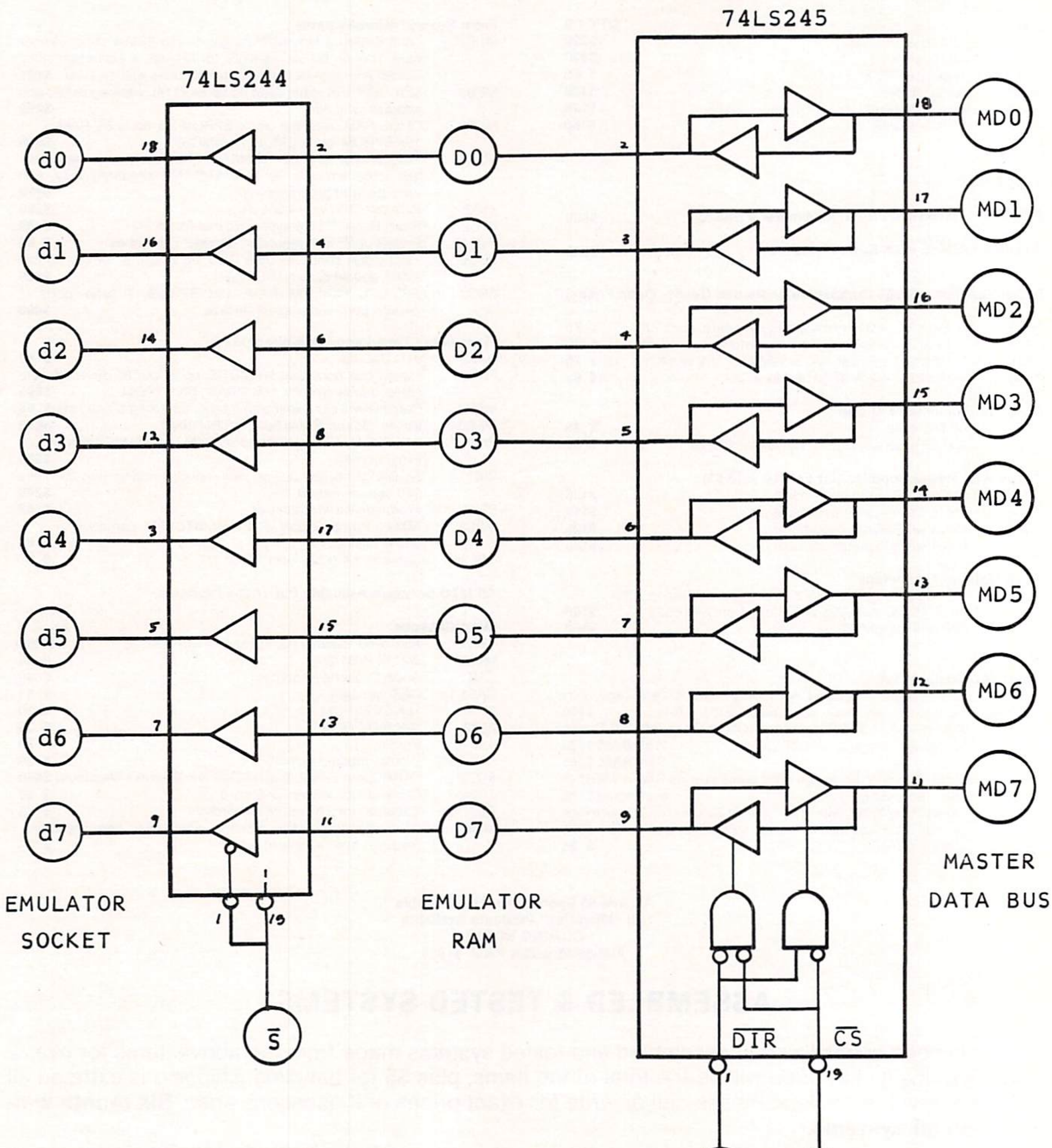


Figure 4. Data Interface to Master System and Emulator Socket

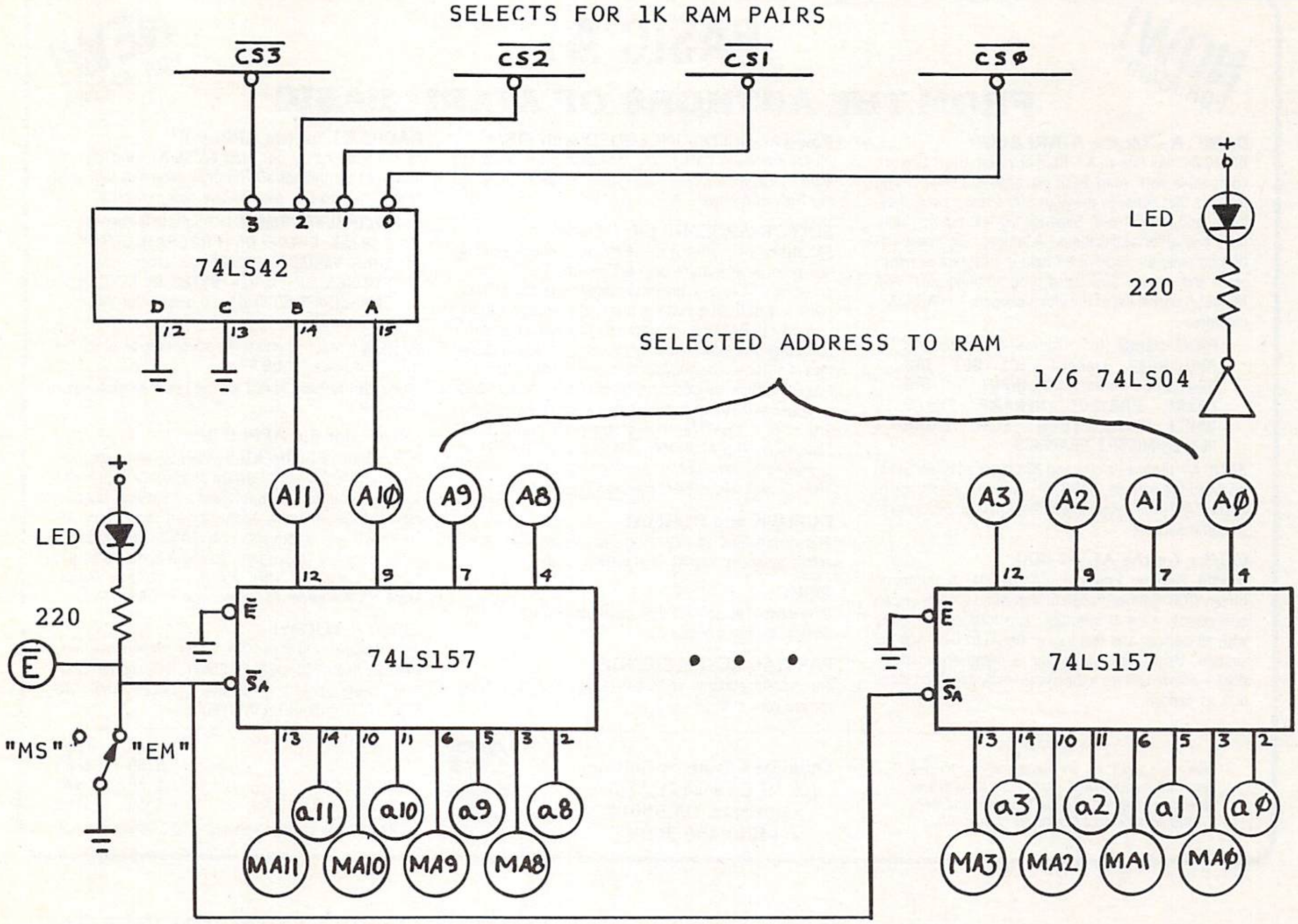


Figure 5. Address Selection for Emulator RAM

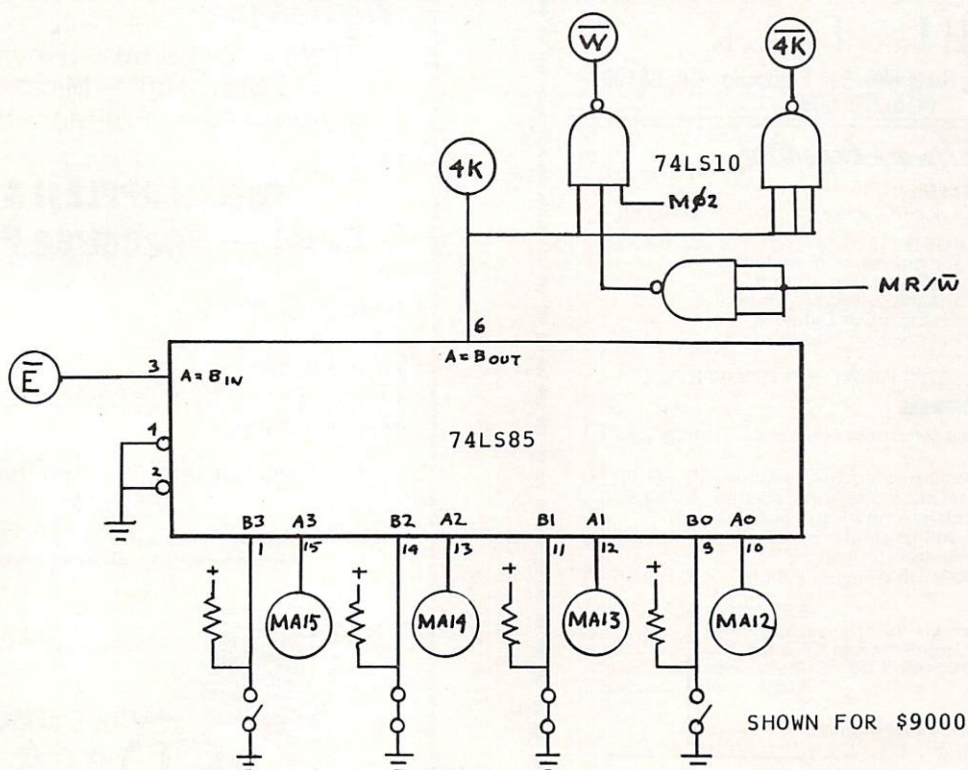


Figure 6. Master System RAM Selection

©

New Complete Guide To Bar Code Technology

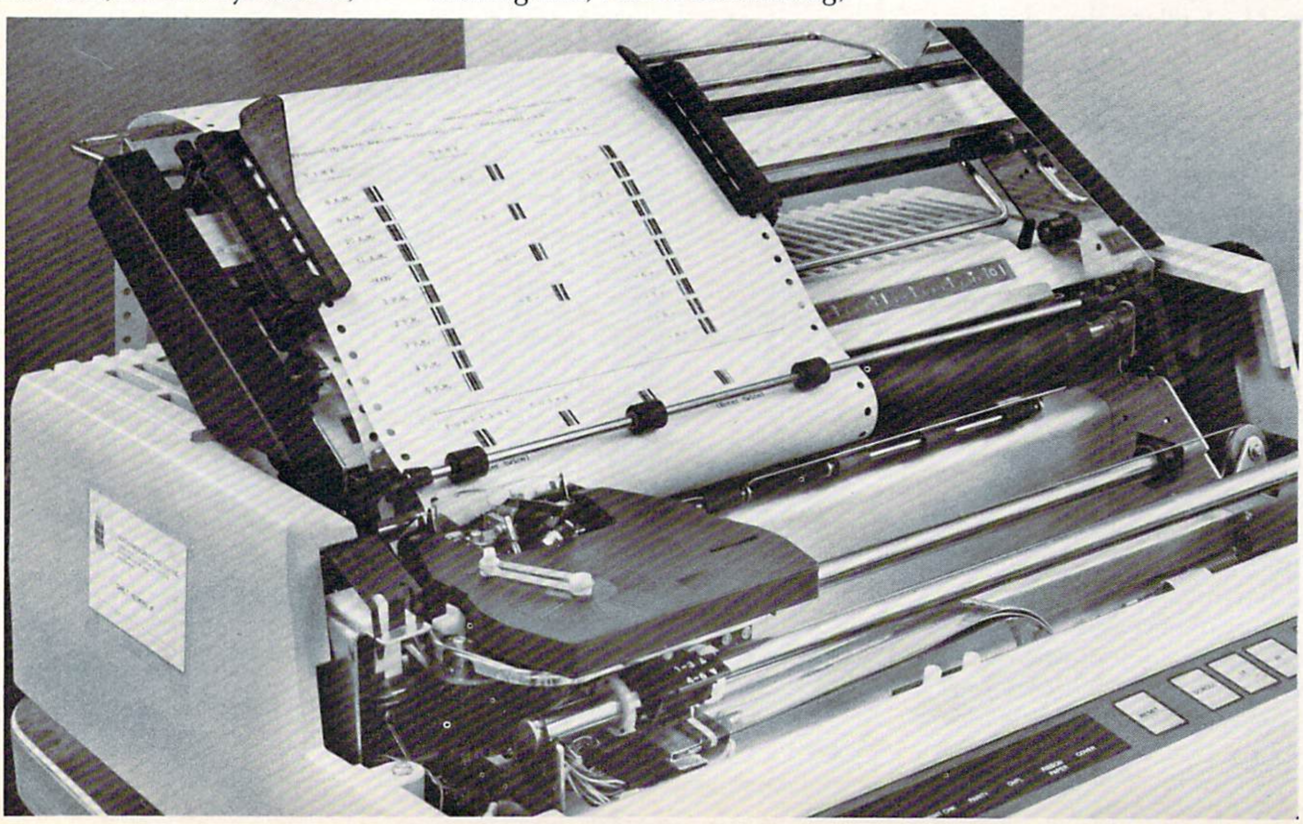
North American Technology, Inc., announces the availability of its major new applications report, "Contemporary Applications of Optical Bar Code Technology." The report is written by the originators of BYTE magazine's experiments with publication of software in printed bar code form, Walter Banks and Carl Helmers. It is a complete document with machine independent UCSD Pascal applications software in bar code readable form which details the uses, techniques, application situations and theory of optical bar code technology.

The bar code report is designed to be a complete guide to a unique, inexpensive coding system highly useful for software distribution, inventory control, sales transactions, customer billing, generalized data entry, customer-account access, and a host of ordinary business processes. The most important characteristic of bar codes is that they provide an inexpensive means of entering information into a computer without detail processing by a human operator. Keyless bar code information entry eliminates translation errors and permits the use of unskilled personnel for a whole class of data entry problems.

The North American Technology bar code report focuses on bar code systems using small computers, and gives complete information on code selection, required hardware, the economics of bar code use, and the extraordinary range of situations in which bar codes can productively be employed. Uses outlined range from household applications to mail order retailing, publication mailing lists, return accounting, and a host of other applications. A major feature is a complete detailed description of the software publication problem and an implementation of a bar code solution.

Manufacturers will be especially interested in those aspects of the report which deal with integrating bar code systems into home and office products. It is assumed that computer system and peripheral designers, manufacturers and users will use the report to evaluate their potential uses for bar codes, and will refer to the software provided for design ideas about implementing their own bar code systems.

For a free brochure detailing the contents of the report, contact North American Technology, Inc., 174 Concord Street, Strand Building, Peterborough, NH 03458. Questions can be answered by calling North American Technology at (603) 924-6048.



CompuServe

Columbus, Ohio — The College Board is now a provider of educational information on the CompuServe Information Services.

The College Board, an educational association of more than 2,500 secondary schools, colleges and educational associations, provides guidance and admissions information to college-bound students and adults nationwide, and helps schools and colleges respond to the educational needs of students.

Owners of personal computers and computer terminals who subscribe to CompuServe can receive information from The College Board on choosing a college, the types of financial aid available to students and adults, preparing for the Scholastic Aptitude Test and adult education.

Future services from The College Board could include exercises related to the building of decision-making skills and a schedule detailing the cycle of activities associated with college selection, application and admission.

CompuServe subscribers can access The College Board information for the standard fee of $5 per hour weekday evenings, all day weekends and holidays. Weekday daytime access is also available. The equipment needed would be a personal computer or computer terminal, a telephone and modem.

CompuServe also offers the latest news from major newspapers and a national wire service, financial information on stocks and commodities, home banking, electronic mail and real-time communications, computer games, family information and computing power for programming activities.

CompuServe is available through a local telephone call in more than 260 U.S. cities. CompuServe is an H&R Block Company.

T.H.E.S.I.S Software For The Apple II+*

T.H.E.S.I.S. software, originally available only for the ATARI Personal Computer**, is now being developed for the APPLE II+*. Many educational programs should be available this September (1981). Scheduled for immediate re-release on the APPLE II+* are: Fishing for Homonyms, Word-Scramble, Word-Mate and Preschool Fun. Most programs will be available on both cassette and disk (DOS 3.3). For a free catalog listing all available programs, write:

T.H.E.S.I.S
P.O. Box 147
Dept. AS
Garden City, MI 48135
or call (313) 595-4722.

* Indicates trademark of Apple Computer, Inc.
** Indicates trademark of Atari, Inc.

Universal AIM-65 Interface Card

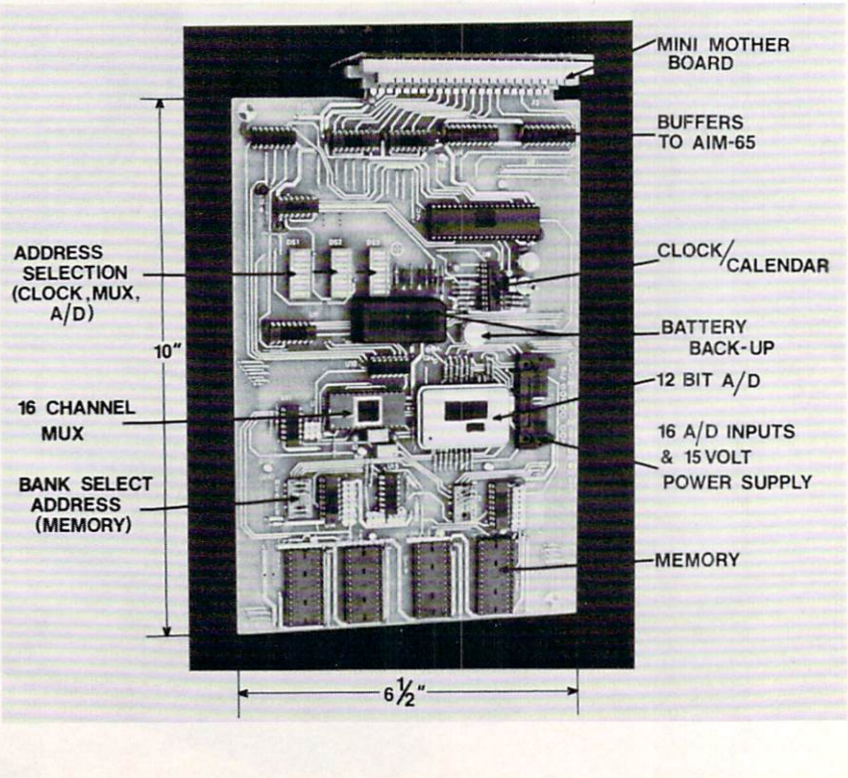
Columbus Instruments presents an addition to any Rockwell AIM-65 computer which enhances its capability in the real time acquisition and control area, while allowing computer program size to be expanded ten times to 40K bytes. Columbus Instruments has incorporated the following on ONE board:

1. Real Time Clock/Calendar with Battery Back-up.
2. Sixteen (16) Channel Twelve (12) Bit High Speed A/D Converter.
3. Universal Memory Capability for RAM, ROM or EPROM. Up to 64K Bytes if Needed.
4. Basic Programs Included to Demonstrate all of the above.

This card plugs into mating mini-mother board into the J-3 expansion plug on the AIM-65 which allows up to a total of three (3) boards to be installed.

Prices range from $390 for Additional Memory, to $1,270 for the A/D plus Calendar/Clock plus Memory Space.

For more information contact: Columbus Instruments International Corporation, 950 N. Hague Avenue, Columbus, OH 43204, (614) 488-6176.



we carry it all....

everything for Commodore and Atari

Atari® Software

VisiCalc	149
CX4101 Invitation to Programming 1	17
CX4104 Mailing List	17
CX4102 Kingdom	13
CX4103 Statistics	17
CX4105 Blackjack	13
CX4106 Invitation to Programming 2	20
CX4107 Biorhythm	13
CX4108 Hangman	13
CX4109 Graph It	17
CX4111 Space Invader	17
CX4110 Touch Typing	20
CX4115 Mortgage & Loan Analysis	13
CX4116 Personal Fitness Program	13
CX4117 Invitation to Programming 3	20
CX4118 Conversational French	45
CX4119 Conversational German	45
CX4120 Conversational Spanish	45
CX4121 Energy Czar	13
CX4125 Conversational Italian	45
CX8108 Stock Charting	20
CXL4001 Educational System Master	21
CXL4002 Basic Computing Language	46
CXL4003 Assembler Editor	46
CXL4004 Basketball	30
CXL4005 Video Easel	30
CXL4006 Super Breakout	30
CXL4007 Music Composer	45
CXL4009 Chess	30
CXL4010 3-D Tic-Tac-Toe	30
CXL4011 Star Raiders	33
CXL4015 TeleLink	20
Talk & Teach Courseware: CX6001 to CX6017	23

Atari® Peripherals:

400 16K	$349
410 Recorder	59
810 Disk	469
815 Disk	1199
822 Printer	359
825 Printer	629
830 Modem	159
850 Interface Module	139

Atari® Accessories

CX853 16K RAM	89
CX70 Light Pen	64
CX30 Paddle	18
CX40 Joystick	18
CX86 Printer Cable	42
CO16345 822 Thermal Printer Paper	5
CAO16087 825 80-col. Printer Ribbon (3/box)	17
Microtek 16K RAM	79
Microtek 32K RAM	179

ATARI 800™ with 32K RAM only $759

commodore

VIC-20	$ 279
4032N	1020
8032	1175
CBM 4022 Printer	630
CBM 4040 Drive	1020
CBM 8050 Drive	1420
CBM C2N Drive	87
PET-IEEE Cable	37
IEEE-IEEE Cable	46

Disks

Maxell Disks	10 for $36
Syncom Disks	10 for 29
Atari Disks	5 for 22

Software

EBS Accounts Receivable Inventory System	$595
OZZ Information System	329
BPI General Ledger	329
Tax Package	399
Dow Jones Portfolio Management	129
Pascal	239
WordPro 3 (40 col.)	186
WordPro 4 (80 col.)	279
WordPro 4 Plus (80 col.)	339
Wordcraft 80	319

Printers

NEC 5530	$2495
Diablo 630	2195
Trendcom 100	299
Starwriter	1495
Trendcom 200	489
Paper Tiger 445G	769
Paper Tiger 460G	1219
Epson MX-80, MX-80FT, MX-70	Call for new prices!
Tally 8024	1699

Please Call Between 11AM & 6PM (Eastern Standard Time)

No Risk -

No Deposit On Phone Orders -

COD or Credit Card - Shipped Same Day You Call*

Prepaid Orders Receive Free Shipping

(800) 233-8950

VISA master charge

* on all in stock units

Computer Mail Order 501 E. Third St., Williamsport, PA 17701 (717) 327-9575

Hayden Applesoft™ Compiler Available

This product translates a standard Applesoft BASIC program into true machine code that runs three to more than 12 times faster than normally interpreted code. The longer and more complex the original program, the greater the increase in execution speed.

The Compiler is easy to use and compilation requires only one step before the user may BRUN the Compiler program.

The Hayden Applesoft Compiler provides an economical solution to a long standing problem: BASIC is easy to work in, but slow to execute; machine code is quick running but difficult to write. The Compiler now allows you to write and debug programs in Applesoft BASIC, compile them and enjoy the speed inherent in executing machine language files. Designed by Jonathan Eiten and published by the Hayden Book Company, Inc. of Rochelle Park, New Jersey, the Applesoft Compiler is being introduced for a price of $200.

The Compiler can be run with the following requirements: 48K of RAM, Applesoft, the Autostart ROM and at least one disk drive. No other hardware is required and the product will operate in a 64K environment, with a RAMcard or Language card.

The following is a summary of the features and advantages of Hayden's Applesoft Compiler: Sub-expressions with integer operand(s) are calculated with faster integer arithmetic routines that result in faster execution of Applesoft intrinsics which require integer values. The Compiler generates sub-array look-up vectors for multi-dimensional arrays, thereby eliminating the time-consuming multiplication that usually occurs during the subscript calculation. The result is increased speed for programs with many multi-dimensional array references. The more dimensions, the greater the increase in speed.

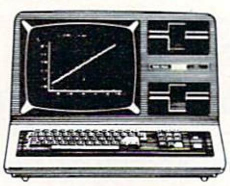
The starting address for a compiled module is specified prior to compilation, a CALL to this address executes the program. Adding two to the start address executes the program without CLEARing its variable space or RESTOREing its data-pointer. This allows modules to retain their data between CALLS (local

REAL PLUS

WE HAVE THE LOWEST POSSIBLE FULLY WARRANTEED PRICES.

Color Computer 4K $310
w/16K Ext. Basic $489

Pocket Computer $189



Model III 16K $859

Model II 64K $3300

Here are just a few of our fine offers . . . call toll-free for full information.

COMPUTERS	
Model II 64K	$3300
Model III 4K LEV I	599
Model III 16K	859
Model III 32K	981.50
+ Model III 32K	915.50
Model III 48K	1104
+ Model III 48K	972
Model III 32K 2 Disc & RS232 c	2100
Color Computer 4K	310
Color Computer 16K	432.50
+ Color Computer 16K	366.50
Color Computer 16K w/extended basic	489
Pocket Computer	199
VIDEOTEX	310
APPLE 48K only	1279
ATARI 800 32K	789

+ Computer Plus New Equipment. 180 Day Extended Warranty

PERIPHERALS	
Expansion Interface OK	$249
Expansion Interface 16K	359
+ Expansion Interface 16K	305.50
Expansion Interface 32K	469.95
+ Expansion Interface 32K	362
16K RAM N.E.C. 200 N.S. chips	39
MODEMS	
Lynx Direct Connect	219
COMM 80 Interface	159.95
Chatterbox Interface	239
Telephone Interface II	169
R.S. Modem I D.C.	130
Atari 830 Modem	159
Atari 850 Interface	183
PRINTERS	
Line Printer IV	849
Daisy Wheel II	1695
Line Printer VI	999
Epson MX80	499
Line Printer VII	315
Line Printer V	1610
Epson MX80 FT	599
Atari 825	
Pocket Computer Printer	130
Centronics 737	737
DISK DRIVES	
R.S. Model III 1ST-Drive	712
TEAC 40 Track MI	319
R.S. 1 Drive Exp MII	999
R.S. 2 Drive Exp MII	1518
R.S. 3 Drive Exp MII	2040
Seimans 8″ MII	799
Atari 810 Disk	499
SOFTWARE	
R.S. Software 10% off list	
Atari Software 10% off list	
Newdos + (40) track MI	88
Newdos 80 MIII	149
ST80III	149
ETC.	
Verbatum 5″ Double Density	32
Verbatum 8″ Data Life	49.95
Ctr-80A recorder	52
C.C Joysticks	22

PLUS real back-up warrantees —

Factory warrantees on Apple and Atari equipment. Other equipment carries manufacturer's warranty or Computer Plus 180 day extended warranty. Combined warrantees carry Computer Plus 180 day warranty or original manufacturer's warranty.

DEALER INQUIRIES ARE INVITED

Prices subject to change without notice.
Not responsible for typographical errors.

TRS-80 is a registered trademark of Tandy Corp.

call TOLL FREE **1-800-343-8124**

computer plus

Dept. B
245A Great Road
Littleton, MA 01460
617-486-3193

Write for your free catalog

PAY LESS.

ATARI
A Warner Communications Company

GET MORE.

PRICE EXAMPLES:

ATARI 800 with 32K memory	$759
ATARI 400 with 16K memory	$399

ATARI Software 25% OFF

Mail Order Customers

- Indicate Unit(s) you want
- Shipping Address • Mass. res. 5% sales tax
- Include check, money order, MC or VISA number

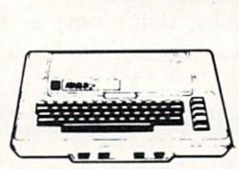
We ship by best means possible freight collect

WARREN'S COMPUTER SYSTEMS

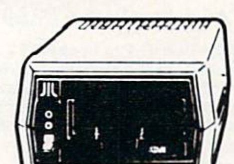
4 Elm St., Braintree, MA 02184 (617) 843-1133
After 5:30 EST please dial (617) 961-3285

SEND FOR CATALOG

LOWER THAN DISCOUNT
SUPER SPECIAL



ATARI 800
$735.00



ATARI D.D.
$ 449.00

ATARI SOFTWARE	20% off
EPSON MX80	475.00
DC HAYES MICROMODEM II	289.00
Z-80 SOFTCARD	295.00
MICROSOFT 16K	150.00
NOVATION DCAT	160.00
BMC MONITOR	225.00

unicomm ™
MARKETING CORPORATION

THE PURCHASING AGENT FOR HOME AND BUSINESS ELECTRONICS
PO BOX 77 DEPT."C" BEVERLY HILLS, CA. 90213 (213) 451-8089

CALL OR WRITE FOR FREE CATALOG

variables). Modules may CALL one another and/or may be CALLed from a real Applesoft program running under the interpreter.

All Applesoft features are supported except dynamic array dimensioning and those few statements that no longer make sense outside of an interpreter environment, such as LIST and TRACE. The advantage, in executing the source program, is that the fact of compilation is transparent to the user.

The 3.3 DOS version of the Hayden Applesoft Compiler will soon be available. HAYDEN APPLESOFT COMPILER, 08809, Apple II Disk, $200. The Hayden Applesoft Compiler is a trademark of the Hayden Book Company, Inc. It is available through local computer dealers or by calling 800-631-0856.

Microcomputers In Education Conference Proceedings

In January 1981 the College of Education at Arizona State University hosted a special microcomputer conference designed to introduce educators and administrators to the many applications of microcomputers in the classroom. Over 400 people attended the conference from all over the United States.

The conference proceedings are now available in a 340-page booklet and include over 30 articles. A few of the titles included are, "Instructional Techniques for Teaching BASIC Programming to Elementary Children"; "Using Computers with Blind and Deaf Children"; "Managing Instruction with a Micro"; "The Challenge of the 1980's: Computer Literacy"; "Microcomputers in High School Physics."

The proceedings are available at a cost of $10.00 each. Please send your request along with a check payable to Arizona State University to:

Dr. Gary Bitter
Arizona State University
Payne B203
Tempe, Arizona 85287

Computer Conferencing

First Edition, by the Communi-Tree Group, is a computer conferencing software package with general data base and "electronic mail" features. Designed to run on an Apple II™ (modem not required for local data base use), the system allows users to enter messages and to retrieve them via regular phone lines using a computer or terminal and a standard 30 cps or 10 cps modem. Each new message is attached to any other already in the data base, allowing users to organize a "conference tree" of messages — any one of which can "grow" into a new conference or subconference. This "tree" structure makes it easy to find specific information, and to enter information in the most appropriate part of the data base. Maximum message length is 50 lines of 80 characters, with up to 320 different messages, depending on available disk space.

Messages entered can be either public — readable by everyone using the system — or private, and both kinds can share the same system. This means that confidential or preliminary conferences can take place prior to publication in the "public" sections, if so desired.

Seven different levels of password protection take the system from "completely open/public" to "completely closed/private." (The latter might be

used for a personal or professional messaging system/network, for example.) Passwords are easy to change.

Applications include on-line publishing; an "electronic bulletin board" (BBS); and others.

A backup utility for making copies of the program and data disks is included in the package.

The source code, written in FORTH, will be available separately (4th quarter, 1981).

In addition to technical and start-up data, the manual contains an introduction to microcomputers with an emphasis on telecommunications and its applications/implications. The appendix has related data on other telecommunications software, data base directories, and various other telecommunications software, data base directories, and various telephone services now available to microcomputer owners. The manual alone serves as an introduction to computing for non-technical people. An extensive bibliography, glossary and index are included.

Introductory price for the complete conference tree system is $95; the program (on 5¼" floppy disks) alone is $80; the manual alone is $20. The disks include a start-up conference with complete online documentation ("help files") for remote users.

For more information, call or write the CommuniTree Group, 470 Castro Street, Suite 207-3002, San Francisco, CA 94114. (415) 474-0933. To get an on-line demonstration of a running, publically available system, dial either of these "flagship" trees: (415) 938-0641 or (415) 526-7733. Type two carriage returns to start; TRS-80 users type two "enters."

Two-Player Invasion Orion

Mountain View, CA — Automated Simulations, Inc. has added a new two-player option to its EPYX game, Invasion Orion.

Invasion Orion, a tactical space battle game, can now be played by either one or two players.

In Invasion Orion, each player controls up to nine spaceships armed with destructor beams, missiles and torpedos. Ships spend energy on movement, shields and firing weapons, and each player must decide how to allocate that energy to defeat his opponent.

Invasion Orion includes 10 scenarios that provide a variety of settings and objectives for both beginners and experts. When battling against the computer, the player can adjust the difficulty of the game by choosing from three skill levels. An expert player can make a challenging game against a novice human by allowing his opponent more powerful ships.

A second program, also included, lets the players create more scenarios of their own, and even design their own starships.

Invasion Orion comes in a four-color box with game program and 48-page battle manual. The game is available on cassette for the Apple (32K with Applesoft in ROM), TRS-80 (Level II, 16K), Pet (16K) and Atari 400/800 (32K), or on disk for the Apple (48K and Applesoft in ROM), TRS-80 (32K), and Atari 400/800 (32K). The suggested retail price is $24.95.

Invasion Orion is available from Automated Simulations, Inc., P.O. Box 4247, Mountain View, CA 94040.

The Atari Assembler

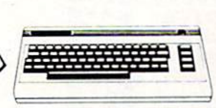
Reston Publishing Company is pleased to announce the publication of a new computer book, THE ATARI ASSEMBLER, by Don Inman and Kurt Inman.

Since the explosion in the use of personal microcomputers, more and more people are acquiring computers of their own. After conquering BASIC lan-

commodore

summer spectacular

8032-32K 80 COL CRT REG. $1495	**$1195**
4032-32K 40 COL CRT REG. $1295	**$995**
4016-16K 40 COL CRT REG. $995	**$795**
8050-DUAL DISK 950K REG. $1795	**$1395**
4040-DUAL DISK 343K REG. $1295	**$995**
C2N-CASSETTE DRIVE REG. $95	**$79**
4022-80 COL PRINTER REG. $795	**$649**
8024-MANNSMAN TALLEY REG. $1995	**$1595**
25CPS-STARWRITER REG. $1895	**$1445**
CBM-IEEE MODEM REG. $279	**$229**
VOICE SYNTHESIZER REG. $395	**$329**
VIC 20 ★★★★ REG. $299	**$269**
PET TO IEEE CABLE REG. $39.95	**$34**
IEEE TO IEEE CABLE REG. $49.95	**$39**



SOFTWARE

	LIST	YOU PAY
OZZ—The Information Wizard	$395	$299
Wordcraft 80	$395	$299
IRMA—Info Retrieval & Mgmt. Aid	$495	$399
Dow Jones Portfolio Mgmt.	$149	$119
Pascal Development Pkg.	$295	$229
EBS—Receivables, Inventory	$750	$579
BPI—General Ledger	$395	$299
Word Pro 3—40 Column	$250	$179
Word Pro 4—80 Column	$375	$269
Word Pro 4 Plus	$450	$329

CUSTOM PRINTED CONTINUOUS MINI-FORMS

- LOOK PROFESSIONAL
- LOW PRICES FOR SMALL RUNS

1000-11"x9½" BLACK

PARTS	LETTERHEAD	INVOICE	STATEMENT
1	37.50	—	37.50
2		63.50	63.50
3		79.50	
5		115.50	

- SPECIAL LOGOS, 2 COLOR PRINTING & SPECIAL STOCK AVAILABLE

MINI-FORMS PRICES & ORDERS:
(800) 523-3442
PA & West of Rockies (215) 455-9705

Main Line Computer Discount

512 W. LANCASTER AVE.
WAYNE, PA 19087
215-687-8502

MAIL ORDER PHONE
800-345-8112
IN PA CALL 1-800-662-2444

- PREPAID ORDERS SHIPPED FREE
- VISA & MASTERCARD ADD 3%
- COD-UPS
- PA RESIDENTS ADD 6% PA TAX

guage, an assembler is the next step. THE ATARI ASSEMBLER gives you detailed directions for using the Atari Assembler Cartridge with the Atari 400 or 800 Model Computer. Inman and Inman help you take what you already know about BASIC and apply it to learning Assembly language.

Slow paced for beginners, the instructions are explained in words and pictures too. You'll see sketches of the video screen for each stage of entry and execution of demonstration programs, and you'll find each program described in easy-to-understand terms so you can immediately begin experimenting in this new language.

Topics in this book include Memory Use, Machine Language, Assembler Review, Program Design, and Addressing Modes.

Don Inman is a control system engineer and Kurt Inman is the editor of Dymax Gazette.

To order directly from Reston call 800-336-0338. $9.95 paper, $14.95 cloth.

CAI Software

EICONICS, INC. of Taos, New Mexico, announces the release of The Eureka™ Learning System, a software system for the generation of Computer Aided Instruction courses on microcomputers. The Eureka Learning System has been under development since 1979, and has been field tested in a junior high school in Vermont during the 1980-81 school year.

With The Eureka Learning System, teachers may develop CAI courses of their own design, without any knowledge of programming. The menu-driven Text Writer program displays all options available to the teacher at each step in the development process. The text that is developed by the teacher is then presented to the students through the use of the Educator program, which is the system's other main component.

"We were not sure when we started almost three years ago what could be achieved on the 32k personal computer," said Reginald Barrow, Director of Product Development. "We have achieved a method for teachers to present instructional material on any subject to the student with a level of flexibility previously unavailable," he said in a recent conversation. The system provides special symbols and graphic presentation as well as text.

"Our system will be marketed to educational institutions for use in their work as well as to those individuals who wish to develop courses for sale," said June Barrow, President of the firm founded in 1974. "We see The Eureka Learning System as a service to education as well as a business with fantastic potential," she continued.

A demonstration kit consisting of three sample lessons in punctuation, geography, and geometry, together with a copy of the Teacher's Guide, is available from EICONICS. The Teacher's Guide shows teachers how to use the system in a 'learn by doing' type of environment. By following the guide, teachers develop three courses, and cover most areas of the system. The $25 fee for the kit will be deducted from the license price when a system license is purchased.

A dealer program has been started with the first dealer in Santa Fe, New Mexico. Other dealers are being sought for other locations in a nationwide network. Product support and continued development of other educational products will be done from company offices in Taos.

For further information, contact Eiconics, Inc., 200 Cruz Alta, Taos, NM 87571, Tel. 505-758-1696.

WHAT'S NEW?

Consumer Computers announces it's lowest prices ever.

Apple Add-Ons

Language System w/Pascal	379
Hayes Micromodem II	299
Novation Apple-Cat II	339
Videx Videoterm 80 w/graphics	269
Z-80 Microsoft Card	269
16K Ram Card Microsoft	159
ABT Numeric Keypad (old or new kybrd)	110
ALF 3 Voice Music Card	239
Heuristics Speechlink 2000	249
Alpha Syntauri Keyboard System	1599
Corvus 10 MB Hard Disk	CALL
Lazer Lower Case Plus	50
Micro-Sci Disk Drives (A40 & A70)	CALL
SSM AIO Serial/Parallel Card A&T	189
ThunderClock Plus	CALL
Integer or Applesoft II Firmware Card	145
Graphics Tablet	619
Parallel Printer Interface Card	135
Hi-Speed Serial Interface Card	135
Smarterm 80 Column Video Card	299
MusicSystem (16 voices)	479
A/D + D/A Interface	289
Clock/Calendar Card	225
Supertalker SD-200	239
Romplus + Card	135
Clock/Calendar Module (CCS)	99
Asynchronous Serial Interface Card (CCS)	129
We carry all CCS hardware	Please Call

FREE CATALOG!

Please mail us your name and address.

Solution Software for the Apple II.

Visicalc 3.3	169
CCA Data Management	85
DB Master	169
WordStar (Apple 80 col. ver.)	299
Desktop Plan II	169
Applewriter	65
Easywriter	225
Appleplot	60
Peachtree Business	1200
VisiTerm	129
VisiTrend/VisiPlot	219
DOS Toolkit	65
Dow Jones Portfolio Evaluator	45
Dow Jones News & Quotes Reporter	85
Apple Fortran	165

Atari Personal Computer

ATARI 800 16K $749

Atari Accessories

410 Program Recorder	59
810 Disk Drive	449
850 Interface Module	159
16K Ram Memory Module	89

Please call us for an Atari Software List.

S-100 Cards

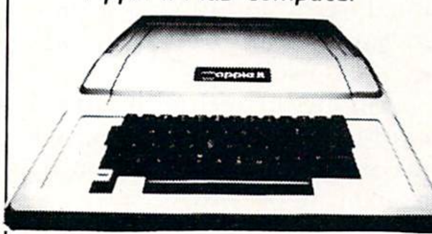
SSM

	Kit	Assm & Tested
Z-80 CPU CB-2	219	279
I04 2 P + 2 S I/O	189	249
VB-3 80 x 24 VIDEO	329	379
VB-3 80 x 48 VIDEO	369	429
BV-2 VIDEO	159	219
PB1 PROM PRGMMR & EPROM BOARD	139	199

CCS

Z80 CPU 2810	n/a	229
64K Dynamic RAM card	n/a	499
16K Static RAM 2 MHz	n/a	249
16K Static RAM 4 MHz	n/a	269
Dbl. Density Floppy Disc Controller	n/a	299
12 Slot S-100 Mainframe	n/a	349

Apple II Plus Computer



16K **$1025** 48K **$1129**

Disk II Drive w/controller **$519**

Disk drive w/out controller 439

We carry the Apple III

CALL FREE!

800-854-6654

In California and outside continental U.S.

(714) 698-8088

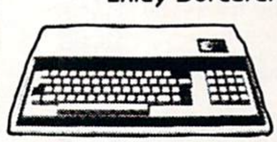
Warehouse (714) 698-0260
Service (714) 460-6502
TELEX 695000 BETA CCMO

AVAILABLE NOW . . .

The NEC Microcomputer

Please call for more details.

Exidy Sorcerer II



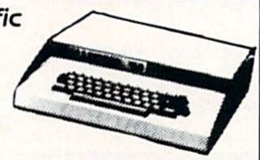
Please call for our price.

S-100 Unit	449
Word Processing Pac	179
Development Pac	89

PMC-80 Micro Computer

$599

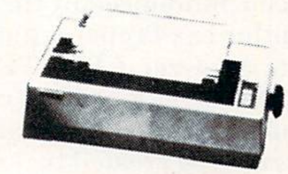
Ohio Scientific Challenger Computer.



4P **$549**

1P MOD II 399

Printers, Etc.

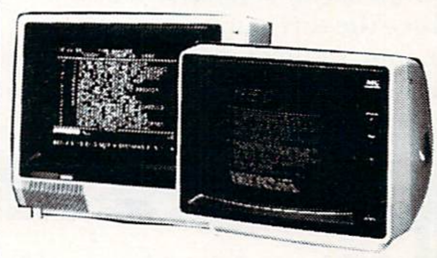


Epson MX-80

$499

Silentype w/Apple Interface	349
Epson MX-80 F/T	649
Epson MX-100	829
Epson MX-70	339
Paper Tiger IDS-445	729
Paper Tiger IDS-460	1099
Paper Tiger IDS-560	1450
Qume Sprint 5/45	2495
Anadex DP-9500/ w/2K Buffer	1349
C. Itoh Starwriter 25 CPS	1649
C. Itoh Starwriter 45 CPS	2249
Centronics 737	699
Watanabe DigiPlot	1149

Save on Video displays.



Amdek/Leedex Video 100 12" B&W	139
Amdek (Hitachi) 13" Color	359
NEC 12" P31 Green Phospher	CALL
NEC 12" RGB Hi-Res Color	CALL
Panasonic 13" Color	449
Sanyo 9" B&W	159
Sanyo 12" B&W	239
Sanyo 12" P31 Green Phospher	279
Sanyo 13" Color	419

We carry much more than listed. Please call our toll free order line to request our complete price list.

How to Order

Ordering information: Phone orders using VISA, MASTER-CARD, AMERICAN EXPRESS, DINER'S CLUB, CARTE BLANCHE, bank wire transfer, cashiers or certified check, money order, or personal check (allow ten days to clear.) Unless prepaid with cash, please add 5% for shipping, handling and insurance. (minimum $5.00). California residents add 6% sales tax. We accept CODs, OEM's, Institutions and corporations please send for a written quotation. All equipment is subject to price change and availability without notice. All equipment is new and complete with manufacturer's warranty (usually 90 days). Showroom prices may differ from mail order prices.

Send orders to:

consumer computers
Mail Order

8314 Parkway Drive
La Mesa, California 92041

Creative Financing Software For Apple

A new software package for evaluating loans and investments has just been announced by the makers of the Tax Preparer and Real Estate Analyzer. CREATIVE FINANCING by HowardSoft is sophisticated but friendly software that clears the fog created by today's sophisticated loan packages, complex tax laws, runaway inflation, and cash draining investments. It produces numerous tabular printouts that not only provide you with important payment and depreciation schedules, but also provide you with vital information on the projected profitability of your investments, even for the most complicated loan packages. It is now possible to objectively compare dissimilar investments and loan packages and see the direct impact of inflation, time value of money, taxes, and cash flow timing, and to answer "what if" questions with the confidence that all important factors are taken into account. Whether you invest in bonds, stocks, income property, or trust deeds, or you just want to determine the true cost of your home mortgage or consumer loan, this package is useful.

Among the technical capabilities of the software are computations of return-on-investment, internal rate-of-return, net present value, equivalent compound interest rate, and after-tax cash flows. The package can be used to evaluate wraparound mortgages, complex loan portfolios, yield to maturity of discounted bonds, lump-sum value of annuities, ownership vs. sale-leaseback decisions, and many other involved situations that are common today. Cascaded and overlapping leases and loans may be handled. Printouts can be submitted by financial consultants directly to their clients.

The software comes complete with detailed instructions in a padded loose-leaf notebook, including instructive examples and a tutorial on investment analysis. The software is available on disk for Apple Computers with 48K and Applesoft, and retails for $150 at dealers nationwide. For further information, contact HOWARD SOFTWARE SERVICES, 6713 Vista Del Mar, La Jolla, CA 92037, 714-454-5079.

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.

Business Graphics Packages

Cupertino, CA — June 30, 1981 — Apple Computer Inc. and Business and Professional Software, Inc., a developer of microcomputer software packages, have produced a family of business graphics software packages designed for the $300 million business-professional personal computer market.

David Solomont, president of the Cambridge, Mass. based Business and Professional Software, said, "Business graphics enable managers and professionals to cope effectively with the current information explosion. We want to provide this and other sophisticated decision support tools for the individual user of a desktop computer."

Apple, which is refining and marketing the new software family, has set delivery of the first package — a graphics program for the Apple II — for fall, 1981.

MOUNTAIN SOFTWARE

FREE OFFER

WITH PURCHASE OF ANY 3 PROGRAMS, YOU WILL RECEIVE FREE THE ORIGINAL ADVENTURE GAME, CONVERTED TO LOAD ENTIRELY INTO 48K RAM ON APPLE. NOTHING LEFT OUT. AMAZING!

APPLE SOFTWARE
DISCOUNTS FROM 10% TO 30%

☐ MAGIC WINDOW	$99.95	SALE	$89.50
☐ SUPER TEXT 11	$150.00	SALE	$135.50
☐ APPLE PIE (40 OR 80 COL)	$129.95	SALE	$119.50
☐ SUPERSCRIBE	$99.95	SALE	$84.50
☐ EASY WRITER 80 COL	$250.00	SALE	$225.50
☐ MODIFIABLE DATABASE	$79.50	SALE	$69.50
☐ MAILING LIST DATABASE	$50.00	SALE	$34.50
☐ GEN. LEDGER (CONT.)	$175.00	SALE	$140.50
☐ ACCT RCVBLE (CONT.)	$175.00	SALE	$140.50
☐ ACCT PAYABL (CONT.)	$175.00	SALE	$140.50
☐ PPTY MANGMT (CONT.)	$175.00	SALE	$140.50
☐ VISICALC 3.3 (PERSONAL)	$199.95	SALE	$169.50
☐ HOME MONEY MINDER	$34.95	SALE	$29.50
☐ WIN AT RACES (HANDICAP)	$39.95	SALE	$33.50
☐ L A LAND MONOPOLY	$29.95	SALE	$25.50
☐ WARP FACTOR	$39.95	SALE	$35.50
☐ ZORK	$39.95	SALE	$35.50
☐ MISSION ASTEROID	$19.95	SALE	$17.50
☐ PRO FOOTBALL POINT PRED	$26.95	SALE	$22.50
☐ ODYSSEY ADVENTURE	$30.00	SALE	$26.50
☐ COMPU-MATH ARITHMETIC	$49.95	SALE	$39.50
☐ COMPU-MATH FRACTIONS	$39.95	SALE	$35.50
☐ COMPU-MATH DECIMALS	$39.95	SALE	$35.50
☐ COMPU-SPELL (REQ. DATA DISK)	$29.95	SALE	$25.50
☐ DATA DISK LEVEL 4	$19.95	SALE	$16.50
☐ DATA DISK LEVEL 6	$19.95	SALE	$16.50
☐ DATA DISK SECRETARIAL	$19.95	SALE	$16.50
☐ STATISTICS 3.0	$29.95	SALE	$25.50
☐ PERCEPTION 3.0	$24.95	SALE	$22.50
☐ ALGEBRA 1	$39.95	SALE	$33.50
☐ SPACE EGGS	$29.95	SALE	$25.50
☐ ALIEN RAIN (GALAXIAN)	$24.95	SALE	$22.50
☐ ALIEN TYPHOON	$24.95	SALE	$22.50
☐ SNOGGLE (NEW PUCKMAN)	$24.95	SALE	$22.50
☐ WIZARD AND THE PRINCESS	$32.95	SALE	$29.50
☐ MYSTERY HOUSE	$24.95	SALE	$22.50
☐ PULSAR 11	$29.95	SALE	$25.50
☐ ORBITRON	$29.95	SALE	$25.50
☐ GAMMA GOBLINS	$29.95	SALE	$25.50
☐ HI-RES SOCCER	$29.95	SALE	$25.50
☐ SARGON 11	$34.95	SALE	$29.50
☐ ADAMS ADVENTURE #1,2,3	$39.95	SALE	$34.50
☐ ADAMS ADVENTURE #4,5,6	$39.95	SALE	$34.50
☐ ADAMS ADVENTURE #7,8,9	$39.95	SALE	$34.50
☐ ADVENTURE HINT BOOK	$7.95	SALE	$6.50
☐ TEMPLE OF APSHI	$29.95	SALE	$24.50
☐ HELLFIRE WARRIOR	$29.95	SALE	$24.50
☐ FASTGAMMON	$24.95	SALE	$22.50
☐ THREE MILE ISLAND	$39.95	SALE	$35.50
☐ PHANTOMS FIVE	$29.95	SALE	$25.50
☐ E-Z DRAW 3.3	$49.95	SALE	$39.50
☐ THE PRISONER	$29.95	SALE	$25.50
☐ TERRORIST	$29.95	SALE	$25.50
☐ SPACE	$29.95	SALE	$25.50
☐ A.B.M.	$24.95	SALE	$22.50
☐ CYBER STRIKE	$39.95	SALE	$34.50
☐ FLIGHT SIMULATOR	$35.50	SALE	$31.50
☐ AUTOBAHN	$29.95	SALE	$25.50
☐ GALACTIC TRADER	$24.95	SALE	$22.50
☐ Z-80 SOFTCARD	$349.00	SALE	$299.50
☐ 16K RAM CARD	$195.00	SALE	$175.50
☐ M/R SUPERTERM 80 COL	$375.00	SALE	$325.50

★ SEND FOR FREE PRICE LIST & CATALOG ★

DEDUCT 3% IF PAYMENT ACCOMPANIES ORDER. WE PAY SHIPPING AND IF YOU PHONE YOUR ORDER WE WILL **CREDIT $1.00 FOR CALL.** CALIFORNIA RESIDENTS ADD 6% SALES TAX. ALL ORDERS SHIPPED FROM STOCK WITHIN 48 HOURS. WE ACCEPT MASTER CARD AND VISA.

NAME ....

STREET ....

CITY .... STATE .... ZIP ....

CARD # .... PHONE ....

EXP. DATE .... SIGNATURE ....

★ P.O. BOX 796, DEPT. C-6, TWIN PEAKS, CA 92391 ★

PHONE ORDERS (714) 337-4063

LO•BALL COMPUTERS LO•BALL COMPUTE

apple®

ATARI®

EPSON

... and more

Apple® II

32K	$1099.00
48K	$1149.00
Disk II w/3.3 DOS	$529.00
Disk II	$450.00
Apple III w/128K	$3600.00

EPSON

MX-70	$399.00
MX-80	CALL
Apple card & cable	$99.00

ATARI®

400 w/16K	$439.00
800 w/16K	$775.00
810 Disk Drive	$449.00

We'll meet or beat any advertised price.

HEWLETT PACKARD

HP-85A w/16K	$2697.30

NEC

5510-2 w/Tractor	$2550.00
5520-2 w/Tractor	$2850.00

SOROC

IQ 120	$725.00
IQ 135	$799.00

C.ITOH

Comet	$499.95
Starwriter	$1450.00

Northstar, Altos and Zenith All 25% Discount

LO-BALL COMPUTERS
7677 S.W. Cirrus Dr.
Beaverton, OR. 97005

TO ORDER CALL (503) 641-0211

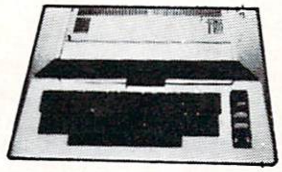
Ordering Information: For fastest service, send money order, cashier's check or bank wire. Visa and MC orders, add 3%. Personal checks accepted (allow minimum 10 days to clear). Hours 9-5, M-F.

Call for our Free Catalog.

LO•BALL COMPUTERS LO•BALL COMPUTE

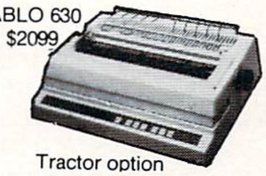
WEST COAST / EAST COAST

EPSON MX-80
$ 479

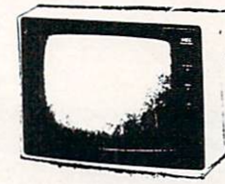


ATARI 800 32K $ 779

DIABLO 630
$2099



Tractor option
$ 245



NEC 12" MONITOR
$ 229

NEC
OKIDATA
APPLE
TELEVIDEO
COMMODORE
RADIO SHACK
AMDEK
ATARI

DIABLO 630	$2099
INTERTEC SUPERBRAIN 64K RAM	$2799
QD SUPERBRAIN	$3195
NEC 5510 SPINWRITER	$2495
NEC 5530 SPINWRITER	$2495
OKIDATA MICROLINE-80	$ 399
OKIDATA MICROLINE-82	$ 529
OKIDATA MICROLINE-83	$ 799
APPLE II PLUS 48K	$1189
APPLE DISK w/3.3 DOS Controller	$ 545
APPLE DISK w/o Controller	$ 435
BASE II PRINTER	$ 599
HAZELTINE 1420	$ 799
NORTHSTAR HORIZON II 32K QD	$2975
ANADEX DP-9500	$1295
TELEVIDEO 912C	$ 669
TELEVIDEO 920C	$ 729
TELEVIDEO 950	$ 959
CBM 8032 COMPUTER	$1225
CBM 8050 DISK DRIVE	$1449
CBM 4032 COMPUTER	$1090
CBM 4040 DISK DRIVE	$1090
CBM 4022	$ 679
CBM VIC-20	$ 289
CBM C2N	$ 85
RADIO SHACK II 64K	$3245
RADIO SHACK III 16K	$ 839
LEEDEX/AMDEK 100	$ 139
LEEDEX/AMDEK 100G	$ 169
LEEDEX/AMDEK COLOR-1 13" Color Monitor	$ 349
MICROTEK 16K RAMBOARD for Atari	$99.95
MICROTEK 32K	$ 165
ATARI 400 16K	$ 349
ATARI 825 PRINTER	$ 619
ATARI 850 INTERFACE	$ 139
or both together	$ 749
ATARI 810 DISK DRIVE	$ 449
ATARI 800 32K	$ 769
Call for price list of ATARI software	
NEC 12" MONITOR	$ 229

PRICES ARE SUBJECT TO CHANGE WITHOUT NOTICE.

No Surcharge for Credit Cards Orders • We Accept C.O.D.'s
All Equipment Factory Fresh With MFT. Warranty • Stock Shipments Same Day or Next

WEST COAST
1-800-235-3581

OMEGA SALES CO.
3533 Old Conejo Rd. #102
Newbury Park, CA 91320
1-805-499-3678
CA. TOLL FREE 1-800-322-1873

ΩMEGA SALES CO.

EAST COAST
1-800-556-7586

OMEGA SALES CO.
12 Meeting St.
Cumberland, RI 02864
1-401-722-1027

ΩMEGA SALES COMPANY

Advertiser's Index

The 6502 Resource Magazine

COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-227-1617, Ext. 401
In California call 800-772-3545, Ext. 401

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**____________
☐ **Don't yet have one...**

☐ Please enter my 1 year (12 issue) subscription to **COMPUTE!**
☐ New subscription ☐ Renewal subscription
☐ $20.00 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______/ ______/

7 8 9 10 11 12

The Editor's Feedback:

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**____________
☐ **Don't yet have one...**

During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

Content:
Best Article This Issue (page #, title)

____

Other suggestions:

7 8 9 10 11 12

The 6502 Resource Magazine

COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-227-1617, Ext. 401
In California call 800-772-3545, Ext. 401

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**____________
☐ **Don't yet have one...**

☐ Please enter my 1 year (12 issue) subscription to **COMPUTE!**
☐ New subscription ☐ Renewal subscription
☐ $20.00 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______/ ______/

7 8 9 10 11 12

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 2312 GREENSBORO, NC

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE! Magazine
Post Office Box 5406
Greensboro, NC 27403

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 2312 GREENSBORO, NC

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
Post Office Box 5406
Greensboro, NC 27403

Author Feedback:

COMPUTE!

Author Feedback:

COMPUTE!

Author Feedback:

COMPUTE!

Place
Stamp
Here

To:

Place
Stamp
Here

To:

Place
Stamp
Here

To:

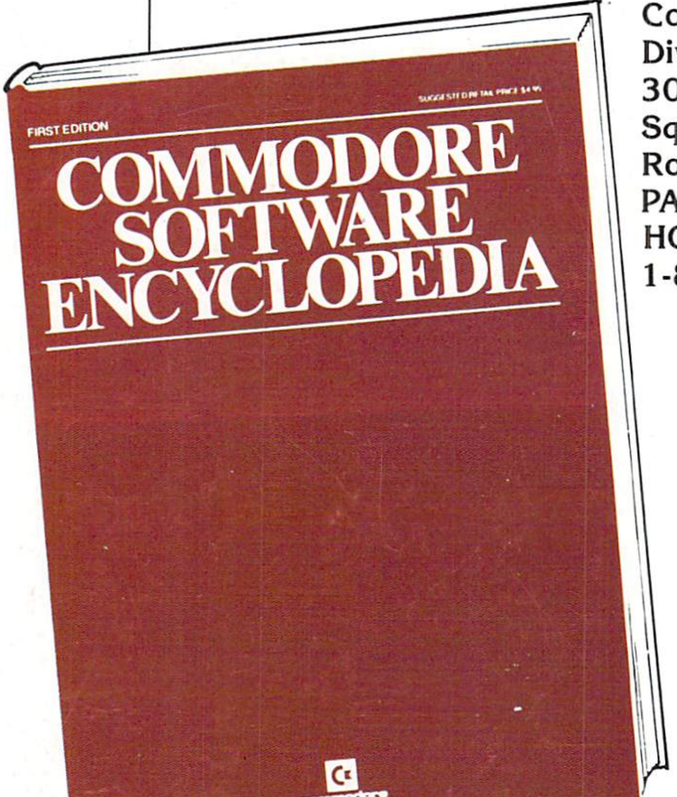
Commodore Software Encyclopedia

The COMMODORE SOFTWARE ENCYCLOPEDIA is now available from your authorized COMMODORE Dealer, for . . .

$4.95

The next edition of the COMMODORE SOFTWARE ENCYCLOPEDIA will be available in 90 to 120 days. If your software is not listed and you would like to have it listed in the next edition, please submit details to:

Software Department
Commodore Business Machines, Inc.
Computer Systems Division
300 Valley Forge Square, 681 Moore Road, King of Prussia, PA 19406.
HOTLINE Number
1-800-523-5622



commodore

CIRCLE 123