Animating Integer BASIC Low-Resolution Graphics

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

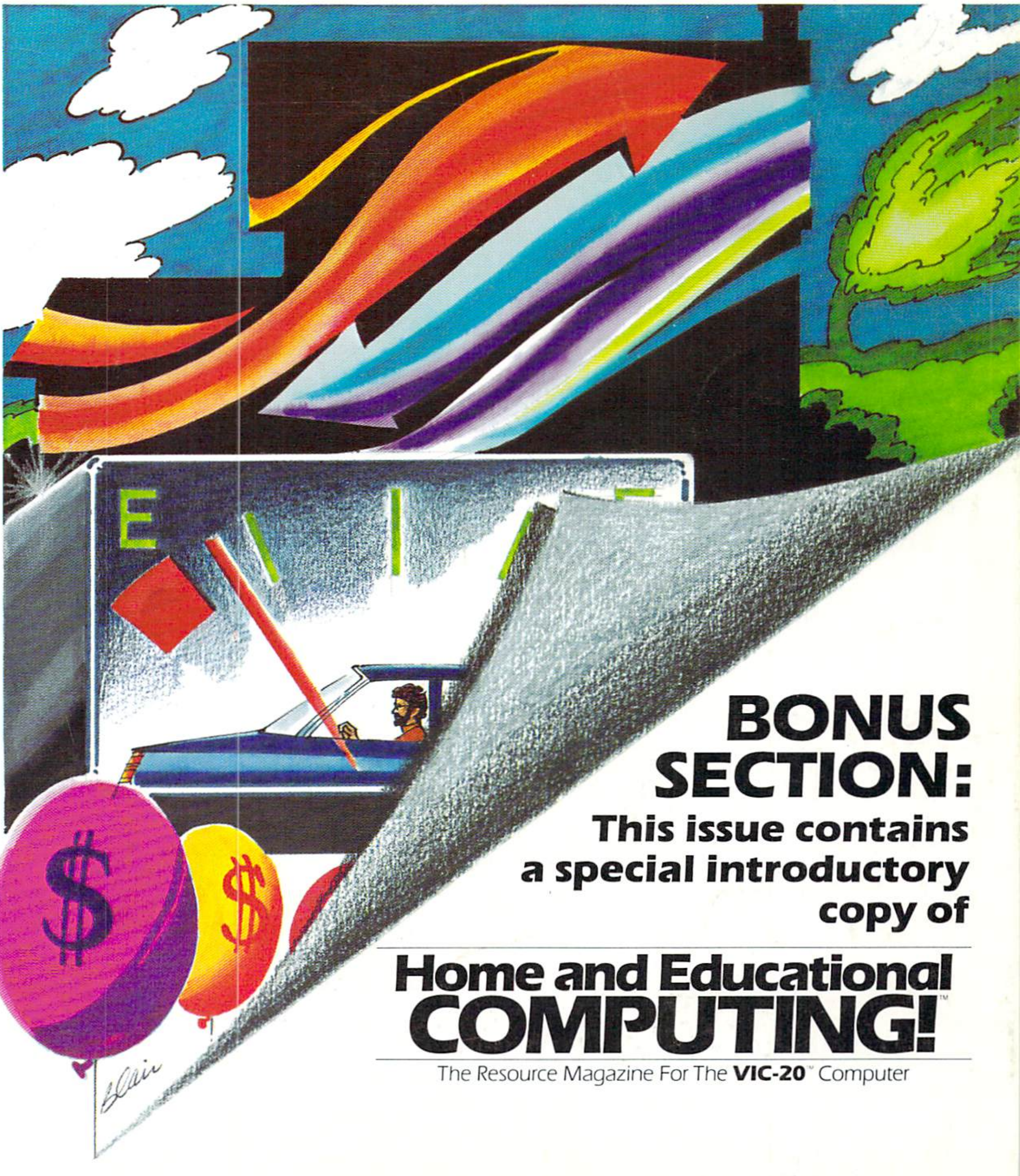
Adding A Voice Track To Atari Programs II

COMPUTE!

$2.50
July, 1981
Issue 14
Vol. 3, No. 7
63379

The Journal For Progressive Computing™

SPECIAL FEATURE: HOME APPLICATIONS



Home Heating And Cooling Audit

Estimating Gas Mileage On An Empty Tank

Inflation Adjusted Loans

BONUS SECTION:

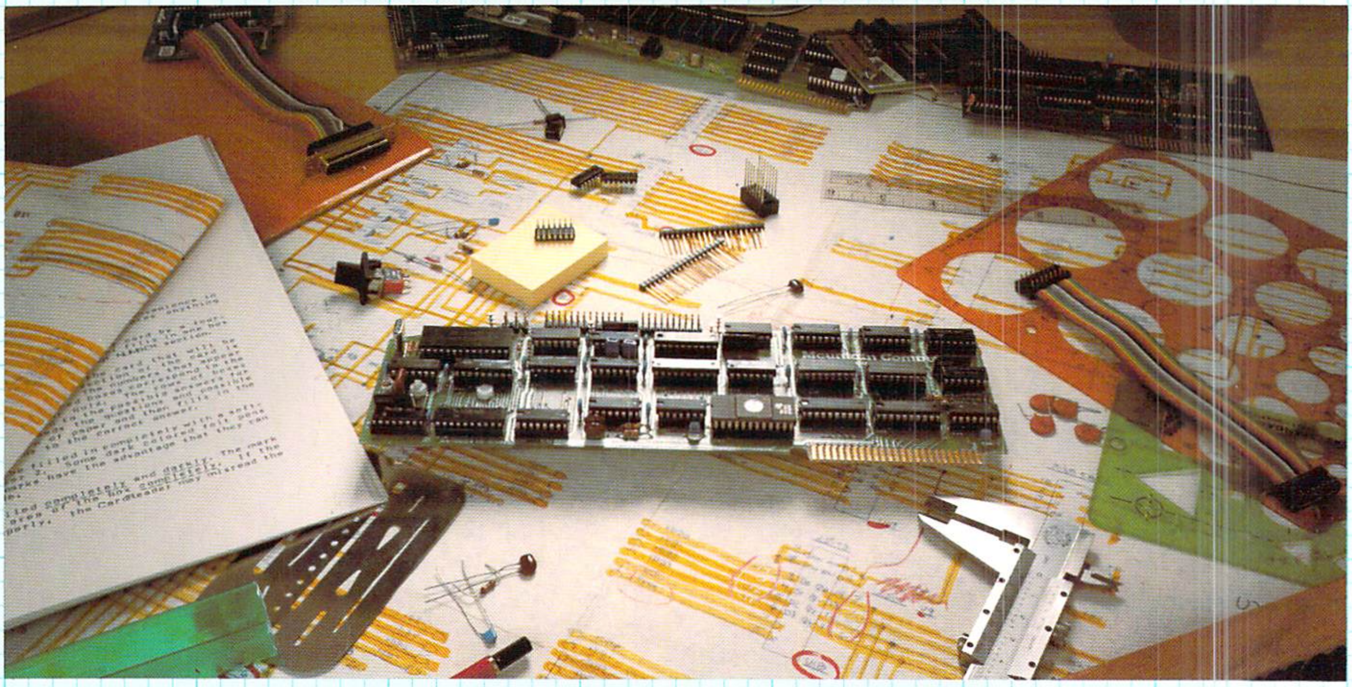
This issue contains a special introductory copy of

Home and Educational COMPUTING!™

The Resource Magazine For The **VIC-20**™ Computer

0 74470 63379 07

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

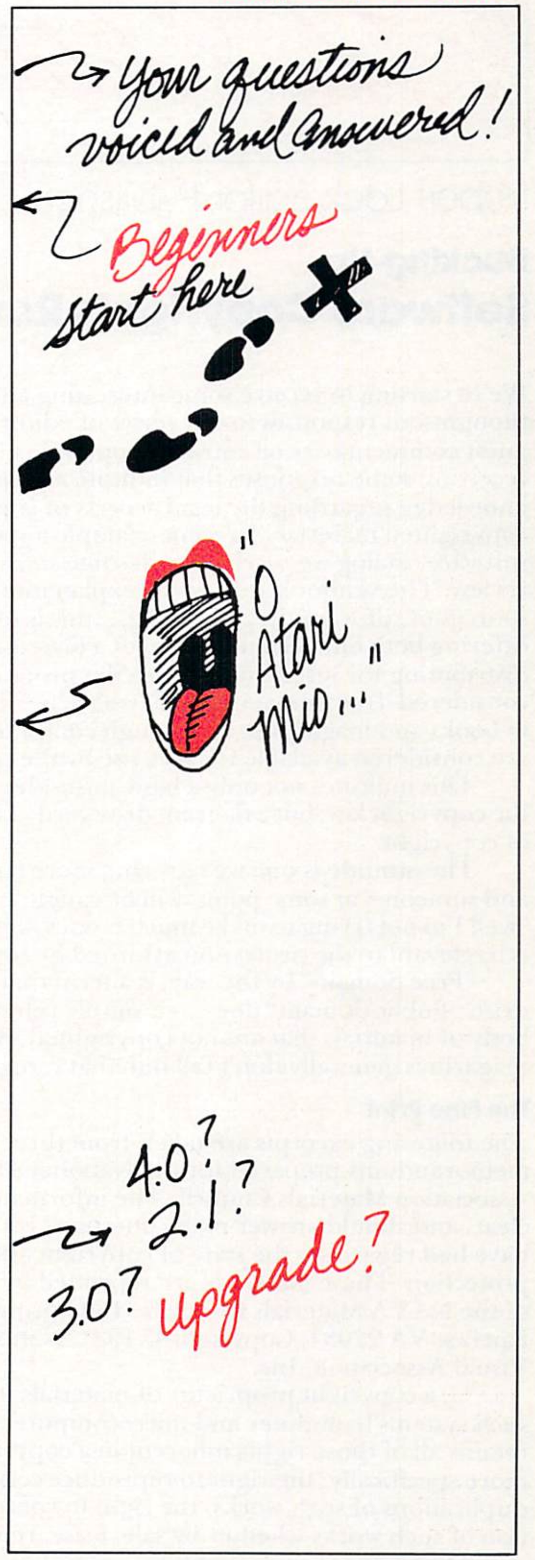
Mountain Computer
INCORPORATED

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


MICROSOFT
CONSUMER PRODUCTS
BASIC-80
CP/M
Z-80
apple II

Table of Contents

July, 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.

A Small System Services, Inc. Publication

COMPUTE!

The Editor's notes...

Robert Lock, Editor/Publisher

Backing-Up
Software Copyrights Revisited

We're starting to receive some interesting and well-thought-out responses to our series of editorials and guest commentaries on software copyrights. We're also receiving some responses that indicate a basic lack of knowledge regarding the legal aspects of copyright and copyrighted material. A recent example appears in a software catalog we received with some software for review. The vendor states, in the explanation regarding sources of software they're selling, ...this is to be done by offering both our own programs at a low cost, and by distributing for just a minimal fee the programs that are considered 'free domain', i.e. they have been given out in books and magazines, or through computer clubs and are considered available for free use by the public."

This indicates not only a basic misunderstanding of the copyright law, but a flagrant disregard of the strength of copyright.

The attitude is one we're seeing more frequently, and someone, at some point, will be caught by it. The "well I'm not trying to make much money at it" argument is irrelevant to the protection afforded by copyright.

"Free domain" by the way, is a term that doesn't exist. "Public domain" does — it simply refers to that body of materials that are not copyrighted. Books and magazines generally don't fall into that category.

The Fine Print

The following excerpts are taken from three legal memorandums prepared for the National Audio-Visual Association Materials Council. The information is quite clear, and should answer many questions you might have had regarding the state of copyright and software protection. These materials are reprinted by permission of the NAVA Materials Council, 3150 Spring Street, Fairfax, VA 22031. Copyright © 1981, National Audio-Visual Association, Inc.

"... a copyright proprietor of materials utilized in such systems [computer and microcomputer systems] retains all of those rights inherent in a copyright, being more specifically: the right to reproduce copies and/or duplications of such works; the right to control distribution of such works whether by sale, lease, rental, loan or any other form of dissemination; the right to use such works for purposes of adaptive or derivative creation; the right to perform or license others to perform such works publicly, with or without commercial gain; and the right to display or publicly show or exhibit such works. This bundle of rights is in no way diminished simply because a copyrighted work is utilized in conjunction with computer-like systems, which entails that a cassette or diskette cannot be reproduced or distributed without prior authorization of the copyright owner."

"The only extent to which there may be limitations on these rights are those concerning 'fair use' ... A school system, for example, may not purchase a cassette or diskette and simply reproduce unlimited copies to be disseminated around its various locations. It may make, perhaps *one copy* under "fair use" exemptions in the event that the original is lost, destroyed or becomes worn. If, however, the school system desires to have several such cassettes or diskettes in circulation, then it must purchase, rent or lease the additional copies.

"Particularly because today's technology allows copying to be done more easily, more quickly and less expensively than ever before, users of audio-visual and microcomputer software should be aware of the fact that they are still violating the law when they copy without permission of the copyright owner any copyrighted materials..."

"Unauthorized copying and distribution of video cassette software, subject only to the narrow confines of 'fair use' is illegal and can be prosecuted both civilly and criminally on a case by case basis."

Finally, the counsel for NAVA, reviews a December 12, 1980 amendment to the copyright code:

"What does the new law provide? Section 7(b) of Pub. L. 96-517 states:

> *"(b) Section 117 of title 17 of the United States Code is amended to read as follows:*
>
> *§117. Limitations on exclusive rights: Computer programs*
> *Notwithstanding the provisions of section 106, it is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program provided:*
>
> *(1) that such a new copy or adaptation is created as an essential step in the utilization of the computer program in conjunction with a machine and that it is used in no other manner, or*
>
> *(2) that such new copy or adaptation is for archival purposes only and that all archival copies are destroyed in the event that continued possession of the computer program should cease to be rightful.*
>
> *Any exact copies prepared in accordance with the provisions of this section may be leased, sold, or otherwise transferred, along with the copy from which such copies were prepared, only as part of the lease, sale, or other transfer of all rights in the program. Adaptations so prepared may be transferred only with the authorization of the copyright owner."*

"We do not believe that the above language changes our opinion (as expressed in our earlier statements) that the owner of a properly copyrighted, adequately "noticed" and otherwise copyrightable work is afforded adequate protection from illegal copying or other types of infringement... The above language, however, while admittedly over-riding the "bundle of rights" provisions of Section 106 of the 1976 Act, does not on its face diminish the power of the owner of a computer program copyright to control the duplication, distribution or transference of his works, *except* as to the narrow scope of duplication indicated above..."

Robert C. Lock, Publisher/Editor
Kathleen Martinek, Publication Assistant
Richard Mansfield, Assistant Editor

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

Contributing Editors
Charles Brannon, P.O. Box 6104, Greensboro, NC 27405
Marvin DeJong, Dept. of Mathematics -Physics, The School of the Ozarks Pt. Lookout, MD 65726
Eric Rehnke, 1067 Jadestone Lane, Corona, CA 91720
David Thornburg, P.O. Box 1317, Los Altos, CA 94022

Georgia Papadopoulos, Art Director/ Production Manager
Terry Cash, Typesetting/Production Assistant
Harry Blair, Director, Advertising and Promotion
Joretta Klepfer, Manager, North American Retail Sales
Alice Wolfe, Manager, Foreign Retail Sales
Bonnie Valentino, Circulation Assistant
Sonja Whitehead, Office Assistant
Dai Rees, Shipping Department

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

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

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

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

Advertising Sales

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

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

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

If you're in the East, we're now 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

Well group, here's the first installment. You'll find more questions than answers in this first column, but we expect to balance that out over the next few columns. Thanks for the tremendous response!

"Please (provide) the address of one manufacturer with the company name of 'Giltronix' — they manufacture a RS-232 switch. Thank you." Roy Partridge

---

"One of the many programming techniques Atari has neglected to mention is how to generate pure graphics modes without using the command GR.X. This would be most desirable when writing Assembly programs without the use of BASIC. I have purchased the Atari Hardware Manuals and found nothing to solve this problem. Any help you can give me will be greatly appreciated." Tracy Principio

---

"I would like to see an article in your magazine on television interference generated by microcomputers and how users or the computer manufacturers have solved this problem. I own an Apple II with 48K RAM, disk, and printer. It sits idle because it disrupts television channels 2 and 5. I had my computer modified by my local Apple dealer but the modification did not help one bit. Am I the only person who has made the mistake of buying an Apple II Computer and who lives in an apartment and can't use it and has virtually wasted $3,000 in cash? To sell my Apple seems to be the only solution but I stand to lose a lot of money to replace the system. I have tried getting help from Apple after the modification was made in vain. Just sign me: My Apple II: It does not compute!"

Comments on interference? With the FCC currently working on applying new standards, we'd be interested in some definitive comment.

"I am building a robot and have run into a problem with the speed control circuit for the motorized wheels. I have been using the book "How To Build A Computer Controlled Robot" for reference, without much success. This is due to the fact that I am forced to use Radio Shack parts, which are always being updated, and therefore I have trouble finding good replacements.

The robot uses two motorized wheels which draw seven amps each at twelve volts. The two wheels have separate Reverse/Forward controls, connected to the Micro., so that it might turn by putting one wheel in forward and one in reverse. The speed must also be Micro. controlled by use of on/off cycles. However the speed for each wheel does not have to be separately controlled. I need a design that uses up-to-date Radio Shack parts, and is easy to build." A Subscriber

---

"I am ten years old and have an Atari 800. I have a problem with a program I am writing and would like to have some help. I have an airplane flying on the screen, but I need something to simulate motion. One solution is to make the land scroll, or to make mountains and hills appear to pass you. I don't know how to do this, and if someone could write a few lines for this I would appreciate it. This is in graphics 8." Joseph Daniels

---

COMPUTE! *is printing more and more advertising for modems to connect my home computer to a communication net. Why should I invest in a modem? What exactly, do the various 'nets' offer? Any examples of actual PET/Apple use? What exactly, can I do with a modem — at what costs; at what actual experience speeds? What are the disadvantages — what can a modem not do? Any value to a modem if not net connected?*

Surely many of Compute's readers, as well as some of your advertisers, would like a thorough research of the Modem investment question from the user's viewpoint." Jim Sercile

Good news, Jim. With the addition of Richard Mansfield to our staff you'll see us doing ever expanding and more responsive "keeping up." We'll be starting telecommunications columns in both **COMPUTE!** Magazine and **Home and Educational Computing!** Magazine (our new VIC-20 magazine) sometime before Fall. We're **very** interested in hearing from reader/users on all the points in Jim's letter.

"I recently purchased an ATARI 800 computer. Inside are an ATARI 16K module and a Microtek 32K module, giving me a total of 48K RAM. However, when I ran: PRINT FRE (0), it came back with the number 37,902.

How do you account for the missing 11K? The computer store where I bought it seemed to think that it was normal to receive such a sum due to the operations of the computer. I really don't think it should use up that much RAM. Please inform me as to what the situation is. Thank you." Dennis Gallagher

I wonder if Atari, Inc. will be surprised that Dennis' 48K system came with two manufacturer's RAM boards ... hum? I was. Anyhow, when you ran **PRINT FRE(0)**, your 37,902 indicates there's nothing wrong. That's exactly what you have *left* as free memory in a 48K system, after Atari takes up the RAM it needs for "overhead" — the operating system, screen memory, BASIC, etc.

"I have an Apple II Plus, and was wondering if you could direct my letter — or me — to an Apple User's Group Tape Exchange.

If not, I'll understand — but if there is one around you know about, I'd sure like to hear from them!

I've read all your issues — and am glad to see more and more Apple news all the time!" Dave Wright

---

The following answers and questions are excerpted from that of a reader suggesting a column such as this one; our issue announcing this column had not yet reached him. Small world, right?

"The advantage, I think, of a questions column or columns, is that of eliciting contributions from persons willing to respond to a declared need to know, but uncertain about submitting unsolicited contributions. For instance, the Apple Manuals are not very forthcoming about HIMEM, and I wasn't at all sure that HIMEM set low to protect hex data in high memory wouldn't result in string garbage overwriting the arrays. In fact, it doesn't and I have a little program designed to show what happens. But maybe everyone else already knows. It's not the sort of thing I'd submit without a prompt. Similarly, FRE(0) is virtually instantaneous: Does it really do a garbage collection? Answer: No, it simply resets the garbage pointer.

Of course, as well as being willing to supply answers to questions, should I happen to know them, what I really want is to be able to submit questions, such as 'Has anybody figured out how to fix the bug in DRAW?' " R. R. Hiatt

---

"In the September-October 1978 Volume 1, Issue 6 of PET USER NOTES, there was described a program called "Index" by David Wilcox which allowed the original PET to locate by fast-forward any program on your tape, thereby saving much hunting time. However, this does not work on the new 4.0 machine. Would appreciate knowing what modifications are needed ..." A Reader ©

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

Computers And Society

Some Speculations On The Well-Played Game ...

David D. Thornburg
Los Altos, CA

For starters this month, those of you who use BASIC might like to enter and run this program:

```
10 PRINT "DO YOU PLAY GAMES ON YOUR
   COMPUTER?"
20 INPUT A$
30 IF A$<>"YES" THEN GOTO 10
40 PRINT "I'M NOT SURPRIZED!"
50 END
```

What is the point? Simply that you probably answered YES the first time through.

I have never seen a computer system yet which didn't have games on it. Over the years games have appeared on every electronic computational device from the pocket calculator to the massive computer systems connected to the Department of Defense's ARPA net. The creation of mechanized games is so natural that I wouldn't be surprised if I found out that Charles Babbage, in the 1840's, would have had Ada Byron write some game programs if they had ever finished building his Analytical Engine.

I don't expect that this comes as a surprise to most of you, since the small computers used by most readers of **COMPUTE!** provide extraordinarily versatile environments in which to play games.

There are several reasons for exploring this issue this month. As someone who, through workshops and conferences, has the pleasure of interacting with thousands of people every year, I hear comments about computer technology from many sources — hobbyists, educators, parents, children, business people, etc. More and more often I hear comments which disturb me. These comments include:

"The trouble with personal computers is that they are just toys."

"Why would I buy a computer made by __________; after all, it's just a game machine."

"I don't want personal computers in my classroom. All that the kids do on them is play games all day long."

Now I have no problem countering any of these arguments by showing that any personal computer can be used for much more than "playing games." Any of you can probably list at least ten non-game-related activities appropriately carried out by personal computers.

But that isn't the point I want to make. What I want to do is pause and ask: "What is wrong with playing games?"

Think about this yourself for a minute. Many people are expressing the view that games are somehow less valuable applications of computer technology than, for example, a program which provides drill in irreducible fractions.

Well, I'm sorry folks, but I just don't buy it.

Those of you experiencing the joy of having small children in the house have the opportunity to see a new living being develop into a conscious self-aware human being who acquires a tremendous amount of linguistic, computational, and social skill by the time he or she is four years old. The skills which are a significant fraction of the total skills that the child will acquire over his or her entire lifespan.

What is wrong with playing games?

And yet, what is it that children do during their first four years? By what magic do they accomplish so much? They play games, that's what.

It may look like "Peek-A-Boo" to you, but to an infant this game is a way of helping a child discover that he or she is separate from the parent and from the environment. It becomes a tool for the development of a self concept.

As children grow older they continue to play games. This activity continues into adulthood, with ever newer and more complex games replacing games which have been "outgrown." Game playing spans all cultures and is traceable into the darkest reaches of human history. Playing games is an intrinsic part of the human experience.

Does this mean that I think children should be playing Asteroids *instead* of learning Analytical Geometry? No, of course not. But on the other hand, I reject the hypothesis that you'll grow hair on your hands if you play Space Invaders too much.

And what this all brings me to is the guest commentary by Mr. Alfred D'Attore which appeared in **COMPUTE!** two months ago.

Mr. D'Attore suggested that computers should be used in the classroom for "drill". In his words, drill is supposed to turn students off. He goes on to say that, realistically, there isn't any other way to

acquire these basic skills. His experience is that children benefit from drill – disciplined, repeated, old-fashioned drill. The alternative to this is the use of computers for "endless, ever-present games."

I'll bet that I can list quite a few games in my library which provide tremendous opportunity for math drill, but which do it in a way in which the acquisition of these computational skills becomes a necessary part of winning the game. I am contrasting this type of game environment with the "sugar coated" CAI drill programs which intersperse rote computational drill with "cutesy" graphics (happy faces when you get the right answer, etc.). No, what I am talking about is the kind of game which gives more opportunities for the acquisition of computational skills than many so-called drill programs, but which are as appealing as many of the traditional computer games which Mr. D'Attore would prefer *not* seeing in schools.

An acquaintance of mine, Thomas Malone, wrote his Ph.D. thesis on this very topic. His work (entitled: "What Makes Things Fun to Learn? A Study of Intrinsically Motivating Computer Games") is concerned with the study of what it is that glues kids to the computer for hours on end. Perhaps, by finding the key to unlock the secret of traditional computer games, he would be able to generate a set of design criteria useful to those interested in writing intrinsically motivating educational programs.

Working with many children in several schools, Tom tried out several modifications of some popular computer games. He tested children's interest in the games when they consisted of pure non-interactive drill, and when the games were successively enhanced by the addition of performance feedback, scoring, constructive feedback, fantasy, sound, and graphics. He learned a lot from these experiments. For example, he found that the best goals are often practical or fantasy goals (like reaching the moon in a rocket) rather than simply goals of using a skill (like doing arithmetic problems). He found that fantasies often make programs more interesting if they are relevant to the skills being used. He found that constructive feedback is important for educational programs. He found that sound and graphics are important — especially to the extent that they can extend the representational system which the program creates.

A sheet full of computer generated raw addition problems fails to meet all of these criteria, in my estimation.

There is much to be said for thinking of games in a larger context as well. One person who has devoted much effort to this task is games designer and philosopher Bernard DeKoven, author of the (unfortunately out of print) book, **The Well Played Game** (write Anchor Press/Doubleday to see if they plan to reissue it!). This book deals with the sensation of "wellness" which we feel when a game is really working for all the players.

The common thread which binds all members of the play community is this quest for the well-played game — the feeling that, whatever our skills, we are playing well. It doesn't matter if we are playing chess or volleyball, the whole reason we participate in games is to gain a connection with the magical feeling we get when the game is working for us. When we are in this state, we are not playing for the score (although score can be important); we are not having to win (although winning can be important); we are playing to maintain this feeling of excellence in our connection with the game and with whomever we are playing at the time.

... they help us to maintain our self concept and our relationship with others.

Because of this, games take on even more importance in the sense that they help us to maintain our self concept and our relationship with others. As Bernie shows, play is important for everyone, adult and child alike. Through games we become a community. Through well-played games we are able to acknowledge our own and each other's genuine claim to excellence.

With this view in mind, think of your own experience with games. When you remember a game do you remember the score, or do you remember that terrific play when you almost dropped the ball and faked out your opponent to pull off a perfect shot from halfway across the court? Or the time when the invaders were on the bottom row of the screen and you knocked them all out to get a bonus? Or the time your partner made a grand slam in your first perfectly bid hand?

Whatever the game, whether the players number from one to a hundred or more, we who play have a common goal — playing well. This feeling is the ultimate motivator. It, more than anything else, makes us want to excel — makes us forget anything else we are doing — and provides us with more incentive for acquiring skills than any rote drill environment could ever provide.

Is my computer a game machine? You bet it is!

Next month ...
In August I will describe a few exceptional games which I have found to be quite useful in skills acquisition. ©

Guest Commentary

The "World Computer"

Marvin De Jong
The School Of The Ozarks
Pt. Lookout, MO

Editor's Note: Most of you are familiar with Marvin's work in the area of problem solving with machine language. We felt this commentary raised some interesting questions. Comments? RCL

From My Soapbox

A few days ago I sat down by my friendly personal computer to load a cassette tape. The tape contained a program intended for educational use at the elementary school level. The loading instructions were complex. Two hours, three tape recorders, and two computers later, I had succeeded only in obtaining the company logo on the video monitor. A phone call to the marketing firm brought the helpful remark "we've never had that problem before." My experience with tapes makes me doubt that statement. If a professional computer user with several computers and tape recorders cannot get a program to run, what do you think will happen in a third grade situation when 20–30 howling kids are left to themselves?

The computer industry, including software vendors, must recognize that entertainment grade cassette instrumentation is completely unsuitable for transporting or storing computer programs. At least that is true in the context of an educational environment.

I am quite happy with our new computer. It has the latest disk operating system. Of course, since most of the disks that are commercially available have a different format, I must boot two disks to use one. My dealer is having difficulty finding the right lower-case adapter for the word-processing system I requested because I have the very latest revision for this computer and disk system. I also just received a complimentary copy of a book containing a disk with a lot of very nice programs on it, for my particular brand of computer. Unfortunately, most of them won't work because they were written for a different version of BASIC. Is this planned obsolescence?

The problems mentioned above were acceptable nuisances when the industry was in its infancy. The same problems are no longer acceptable. A third grade teacher is not a computer expert, and the computer will soon be gathering dust in a corner if it is not easy to load and run programs. Furthermore, the educational system in this country is already in financial difficulty. The tremendous expense involved in operating and/or upgrading different systems will stifle the use of computers at many levels.

Perhaps up until now, industry competition has been fruitful, but competition and cooperation need not be mutually exclusive forces in terms of progress. Why not think about a "world computer," designed for personal and educational use? This computer would be designed by people from (say) "the big four" personal Computer manufacturers (Tandy, Apple, Commodore, and Atari). Each of the four could market their version, but all versions would be completely hardware, software, and graphics compatible. They would all load the same *disks*. They would *not* have cassette I/O, but they would come complete with one disk drive. Color might be optional. Printer interfaces would be compatible and networking would be supported.

Try to imagine the impact of such a development on personal computing, particularly in the educational world. It is difficult to think of a single negative effect. Computer manufacturers would still be able to manufacture their own "other" computers for specific applications, but an inexpensive computer with vast amounts of software would be available to many more individuals than is now the case.

Where would personal transportation be now if automobile manufacturers had produced a world car with universal parts available everywhere? Probably they would not be in financial trouble, and the use of a car would probably not be confined to a *small* percentage of the world's population. ©

The Beginner's Page

Richard Mansfield
Assistant Editor

When you first see BASIC, many of the ideas and techniques of programming are pretty straightforward. GOTO 100 is easy enough. You are telling the computer to jump from wherever it might be in a program to the instructions on line 100. PRINT "LARRY" is not too hard to grasp. (The word PRINT is used because BASIC, in the early days, sent program results — "output" — to a teletype machine instead of a TV. PRINT used to describe what it did exactly. Now, perhaps, another word would be more correct, but it's too late to change things at this point.

In general, though, BASIC words are more or less like what they mean in English. And ideas such as "less than" or "more than" are pretty familiar concepts. Its just a matter of learning the words and punctuation of BASIC and you're *programming* (telling the computer what you want it to do).

But some BASIC words are not so simple. RND, ON GOTO, VAL, GOSUB, STR$, DIM, DATA, are words which refer to ideas generally found only in the computing world. They take some getting used to and some practice. The best way to understand these *computer* words is to work with them in short programs and watch what they do. What effects do they have? How are they useful in getting jobs done?

Probably the most difficult BASIC word (or "statement") to quickly understand is ON GOTO. It usually appears something like this: 10 ON A GOTO 500, 600, 700. We know that GOTO means to jump to a certain line number. But here we are giving the computer three choices of line numbers to jump to. How will it decide? The decision depends "ON A." "A" in this line is a variable. That means that it is just like a box and what's in BOX A could be any number. The contents of the box we call "A" vary — sometimes A is 15 and sometimes A is 233.001 and so forth. A is *variable*, it changes whenever something in the program causes it to change.

The ON GOTO statement makes the computer look up A to see what is "in" A. If A is the number one, then the computer will GOTO the first line number in the list (500). If A is three, then we will GOTO line 700. We only put three line numbers in our program (500, 600, 700) so A had better not be a 5 or a -12. It must be one, two, or three. We could have written: 10 ON B GOTO 40, 50, 60, 70. In this case, the computer would look up variable "B" to see which number between one and four was "in" B. Then it would GOTO the correct line number.

Let's try a short program to illustrate this idea (this sort of programming is called "multiple branching"):

```
10 INPUT A
20 IF A < 1 OR A > 3 THEN GOTO 10
30 ON A GOTO 500,600,700
500 PRINT "WE WENT TO LINE 500"
501 END
600 PRINT "WE WENT TO LINE 600"
601 END
700 PRINT "WE WENT TO LINE 700"
```

INPUT makes the computer stop and wait for something to be typed in from the keyboard (followed by a carriage return). Line 20 is a common programming technique which checks to see if wrong, *illegal*, things were typed in. Since we don't want any numbers except one through three (that's all we provided for in our list on line 30) — we have the computer go back to line 10 for another INPUT if the first one was illegal.

We had to put two END's in or the program would have just gone on to print each message in turn. END stops everything, as you would expect. This "multiple branching" is very often used when you want to allow a choice among different parts of a program. Let's say that you have a program which is really several programs in one. It can do four different things, play four different card games. Common programming practice would begin such a program with a listing of the options. It would say, "Please choose: 1. Poker 2. High-card 3. Bridge 4. Patience." When this *menu* came onto the screen, the user would type in a number between one and four. This is perfect for the ON GOTO statement.

A similar statement is ON GOSUB, but we'll deal with the important topic of *subroutines* in the future. For now, try an experiment. See what would happen with our short multiple branching program if you typed in 1.5 as input instead of an integer (whole number). ©

Inflation Adjusted Loans

When Is It Worthwhile Paying Out Your Loan Early?

M. R. Smith
Alberta, Canada

In June, my house mortgage comes up for renewal. The interest rate on my mortgage is going to jump from 10.75% to 16%. My monthly payments are going to go from $463 principal plus interest to around $680. Since this is one great jump already, I wondered if it would be worthwhile increasing it another $50 a month in order to pay the loan off earlier?

The purpose of this program is to find out the best way to pay off a loan, taking into account inflation, which affects both your salary and the value of the dollar used to pay back your loan. It has been written in a general, machine independent form. The only change needed is in line 13010. Forms of this line for the Apple, PET and TRS-80 are given. The input has been designed to be quasi-foolproof and will accept a carriage return to indicate no change in value. The program investigates the interaction of changes in your salary, the dollar you use and the monthly payments.

Many people realize that paying out a loan early can result in a big saving in the amount of money you have to pay to the bank. Even a small amount of increased payment can make a very large saving appear. If, for example, you have a $50,000 loan at 16% interest, being paid back at $680/month. Over the length of the loan you will have to pay the bank $201,960. Paying back at $710/month, a mere $30 increase, results in a saving of more than $51,000. The table below shows the savings that occur for a number of monthly payments.

Payment Per Month	Total Paid Out	Saving	Loan Paid In
$680	$201,960	–	25 yrs
$710	$150,520	$51,440	17.7 yrs
$740	$129,500	$72,460	14.6 yrs
$770	$117,040	$84,920	12.6 yrs
$800	$108,800	$93,160	11.2 yrs

The purpose of this program is to investigate whether or not these enormous savings are real. After all, inflation means that 10 years down the line you are paying with really deflated dollars. Is it worthwhile to pay out your loan early when your early payments are made in non-inflated dollars? I have always presumed that there was some optimum monthly payment. Pay back too much, too early, then you put a strain on your budget and waste money by paying in non-inflated money. Pay back too little then you waste money by paying for too long. What is this optimum payment?

In addition, suppose you find that paying out early is a good idea, what is the best amount to pay a

... the best way to pay off a loan, taking into account inflation ...

month? Now the answer to this question will depend on what you can afford to pay. If you pay out a large amount each month then you will have little left to pay for other luxuries like food. It is therefore important to determine the impact of the payments on your monthly take-home pay.

As an example, suppose that a loan for $50,000 at an interest rate of 16% per year has been taken out. The net (after tax) income is $3,000 per month and expected to increase at a yearly rate of 12% to counteract inflation. The following table indicates what would happen to both the total cost of the payment and the inflation-adjusted payment. The percentage of the salary that goes into the loan is also indicated. The percentage over the length of the loan and the percentage in the first year are both shown. This information is graphed in Figure 1.

Monthly Payment	Total Payment	Inflation Adjusted Payment		% Of Income Used	
		10%	15%	Average	First Year
$680	$201,960	$81,268	$60,588	8.0%	22.6%
$710	$150,520	$76,301	$59,777	10.8%	23.6%
$740	$129,500	$73,321	$59,190	12.7%	24.6%
$770	$117,040	$71,217	$58,750	14.4%	25.6%
$800	$108,800	$69,709	$58,467	15.8%	26.6%

It can be seen from the graph that the line indicating total amount repaid drops very rapidly as the monthly payment increases. However, the lines for the inflation-adjusted total repayments are very flat, especially for the 15% inflation rate. The 10% inflation rate shows a large drop for the first increase in monthly payment, but it too is fairly flat. However, the average percentage of your salary spent on the loan rapidly increases as your monthly payments increase (column 5). This is because when you pay off very quickly, you are paying in non-inflated dollars and your salary is lower.

From these figures, it seems that it makes sense to pay a little extra a month. Paying a great deal extra only puts a strain on your budget with little overall saving.

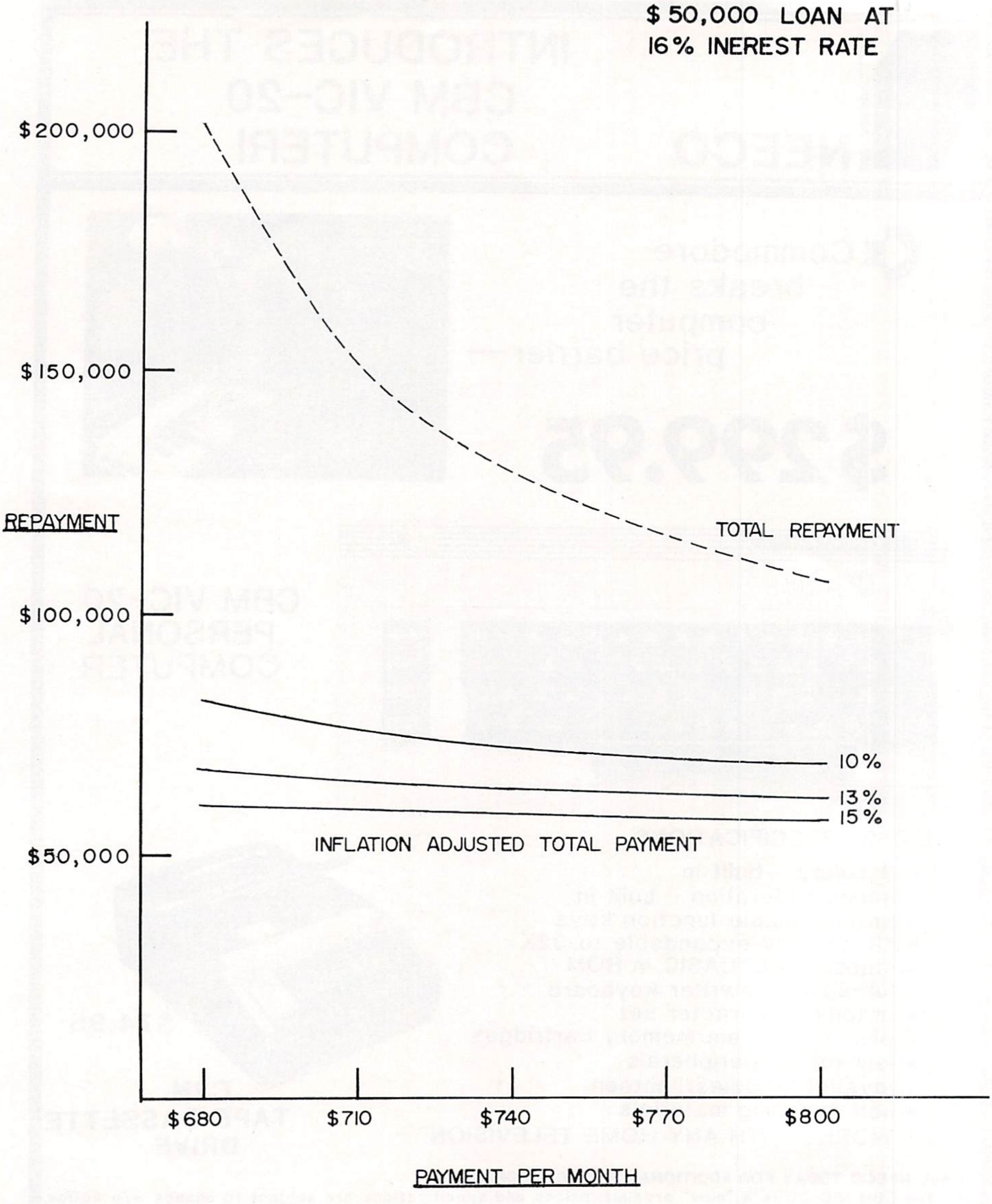

$ 50,000 LOAN AT
16% INEREST RATE
$200,000
$150,000
$100,000
$50,000
REPAYMENT
TOTAL REPAYMENT
10%
13%
15%
INFLATION ADJUSTED TOTAL PAYMENT
$680
$710
$740
$770
$800
PAYMENT PER MONTH

Figure 1

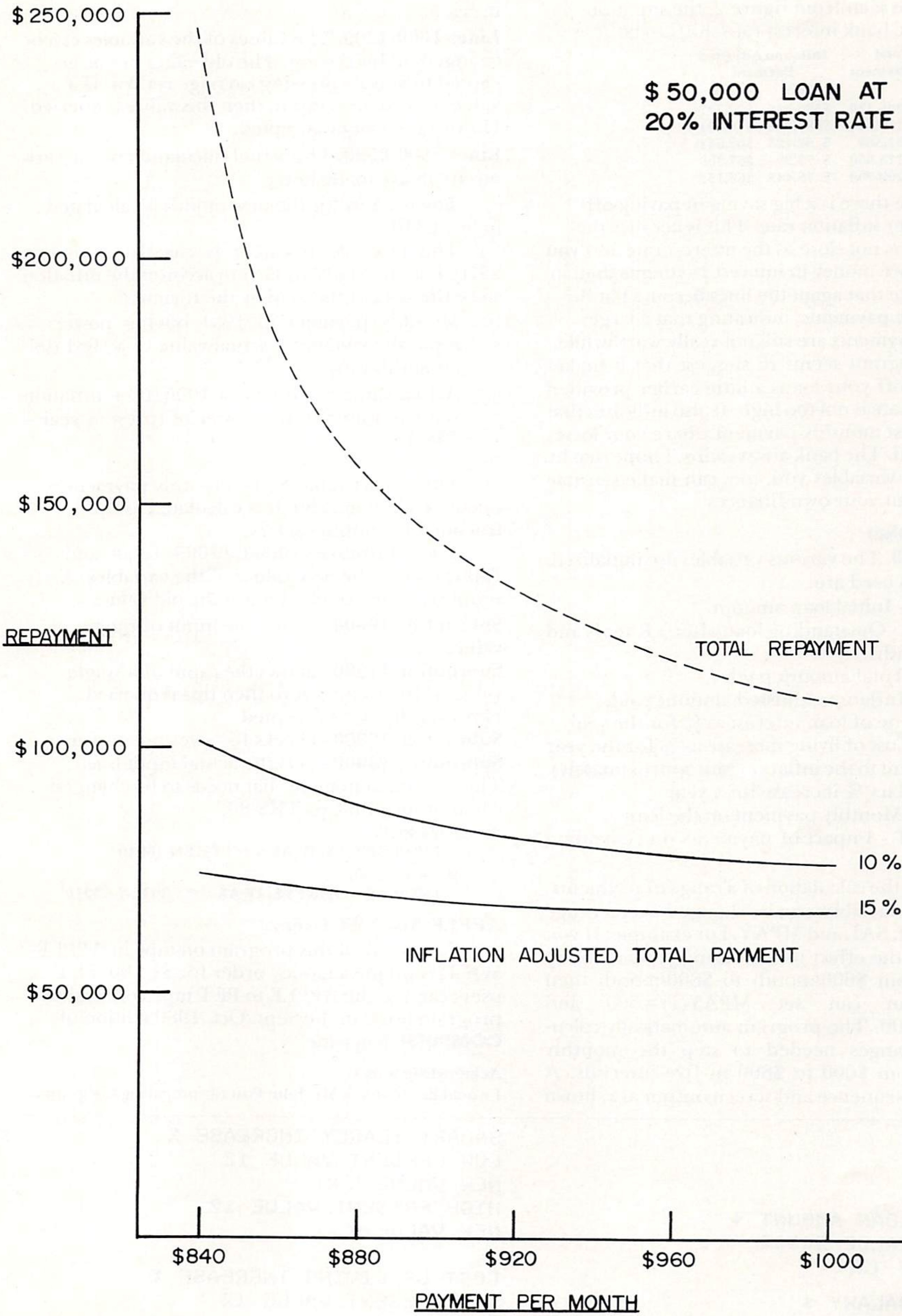

$ 250,000
$200,000
$ 150,000
$ 100,000
$50,000
REPAYMENT
$ 50,000 LOAN AT
20% INTEREST RATE
TOTAL REPAYMENT
10%
15%
INFLATION ADJUSTED TOTAL PAYMENT
$840
$880
$920
$960
$1000
PAYMENT PER MONTH

Figure 2

As can be seen from figure 2, the situation changes if the bank interest rate shifts to 20%.

Monthly Payment	Total Payment	Inflation Adjusted Payment	
		10%	15%
$ 840	$246,120	$100,049	$74,727
$ 880	$156,640	$ 87,888	$70,721
$ 920	131,560	$ 82,423	$68,622
$ 960	$118,080	$ 78,974	$67,200
$1000	$109,000	$ 76,443	$66,132

This time there is a big saving in paying off early for either inflation rate. This is because the inflation rate is not close to the interest rate and you are losing more money in interest payments than in inflation. Note that again the lines become flat for large monthly payments, indicating that a large increase in payments are still not really worthwhile.

This program seems to suggest that it makes sense to pay off your loans a little earlier provided the inflation rate is not too high. It also indicates that there is no best monthly payment where your losses are minimized. The bank always wins. I hope that by adjusting the variables you, too, can make sensible decisions about your own finances.

Program Design

Lines 100-490. The various variables are initialized. The variables used are:

- **ILOAN** - Initial loan amount.
- **OLOAN** - Outstanding loan after YR years and MM months.
- **TPAY** - Total amount paid.
- **APAY** - Inflation adjusted amount paid.
- **RYR** - Rate of loan interest as % for the year.
- **COLI** - Cost of living increase as % for the year (equivalent to the inflation rate approximately).
- **SAL** - Salary % increase for a year.
- **MPAY** - Monthly payment on the loan.
- **IMPACT** - Impact of payments on remaining salary.

To allow the calculation of a range of payments, the following variables can be changed over a range :- RYR, COLI, SAL and MPAY. For example: If you want to find the effect of increasing your monthly payments from $600/month to $800/month then the program can set MPAY(1)=600 and MPAY(2)=900. The program automatically calculates the changes needed to step the monthly payments from $600 to $800 in five intervals. A typical input sequence and screen output are shown in fig. 3.

Lines 1000-1300. The values of the variables can be changed in this section. The old value can be accepted by simply pressing carriage return. If a value of zero (0) is input, then this value is queried. However, it can be accepted.

Lines 2000-2590. The actual calculation of the various totals are made here.

The interest for the new month is calculated in line 2310.

The monthly payment is calculated in line 2370. It is adjusted to take into account the inflation since the start of the loan by the formula:

> Monthly payment in 1981 buying power = monthly payment x actual value of a 1981 dollar at this time.
>
> Actual value of a dollar = 100/(100 + inflation rate) all raised to the power of (present year – 1981).

The impact value of the monthly payment is calculated in line 2390. It is calculated as the fraction of a month's salary.

The subroutines 10000, 11000, 12000 and 13000 take in the new values of the variables. A simple carriage return retains the old value.

Subroutine 10000 - allows the input of ranges of values.

Subroutine 11000 - allows the input of a single value. If the value is zero then this is queried. However, it can be accepted.

Subroutine 12000 - checks for a yes/no answer.

Subroutine 13000 - gets the actual input line.

There is one statement that needs to be changed when using a PET or TRS-80.

For a PET:

13010 GET A$: IF A$ = "" THEN 13010

For a TRS-80:

13010 A$ = INKEY$: IF A$ = "" THEN 13010

APPLE And PET Users:

For a copy of this program on tape in APPLESOFT, send me a money order for $15.00. PET users can use the APPLE to PET tape conversion program listed in the Sept./Oct. 1980 edition of **COMPUTE!** magazine.

Acknowledgements

I would like to thank Mr. John Post for preparing the graphs.

Figure 3

```
 RUN
]

INITIAL LOAN AMOUNT $
PRESENT VALUE 50000
NEW VALUE <CR>

MONTHLY SALARY $
PRESENT VALUE 2000
NEW VALUE 3000 <CR>

SALARY YEARLY INCREASE %
LOW PRESENT VALUE 12
NEW VALUE <CR>
HIGH PRESENT VALUE 12
NEW VALUE <CR>

COST OF LIVING INCREASE %
LOW PRESENT VALUE 12
NEW VALUE <CR>
HIGH PRESENT VALUE 12
```

```
NEW VALUE <CR>

MONTHLY PAYMENT RATE $
LOW PRESENT VALUE 650
NEW VALUE <CR>
HIGH PRESENT VALUE 800
NEW VALUE <CR>

LOAN INTEREST RATE YEARLY %
LOW PRESENT VALUE 16
NEW VALUE <CR>
HIGH PRESENT VALUE 16
NEW VALUE <CR>

PRINT OUT END-OF-YEAR TOTALS? Y/N N
<CR>

LOAN $50000  INTEREST RATE 16%
INFLATION 12%  SALARY $3000/MONTH
SALARY INCREASE 12%

CAN'T BE DONE @ $650/MONTH

AFTER 24 YRS 9 M @ $680 /MONTH
TOTAL $201960   ADJUSTED $71545
IMPACT FACTOR 8

AFTER 17 YRS 8 M @ $710 /MONTH
TOTAL $150520   ADJUSTED $68765
IMPACT FACTOR 10

AFTER 14 YRS 7 M @ $740 /MONTH
TOTAL $129500   ADJUSTED $66981
IMPACT FACTOR 12

AFTER 12 YRS 8 M @ $770 /MONTH
TOTAL $117040   ADJUSTED $65685
IMPACT FACTOR 14

AFTER 11 YRS 4 M @ $800 /MONTH
TOTAL $108800   ADJUSTED $64762
IMPACT FACTOR 15
```

```
]LIST

10  REM
11  REM      CALCULATION OF WHEN TO PAY OUT YOUR LOAN
12  REM
13  REM  M.R.SMITH  MARCH 1981
14  REM
15  REM     304, 86TH AVENUE SE,
16  REM     CALGARY, ALBERTA,
17  REM     CANADA T2H 1N7
18  REM
100  REM    ****************************
110  REM
120  REM    INITIALIZE VARIABLES
130  REM
140  REM    ILOAN = INITIAL LOAN
150  REM    OLOAN = OUTSTANDING LOAN
160  REM
170 ILOAN = 50000:OLOAN = ILOAN
180  REM
190  REM   TPAY = TOTAL PAYMENTS
200  REM   APAY = INFLATION ADJUSTED TOTAL PAYMENTS
210  REM    IMPACT = SALARY IMPACT FACTOR
220  REM
230 TPAY = 0:APAY = 0:IMPACT = 0
240  REM
250  REM   RYR(1) = % RATE YEARLY INTEREST (LOW)
260  REM   RYR(2) = % RATE YEARLY INTEREST (HIGH)
270  REM   RYR(3) = STEP
280  REM
290 RYR(1) = 16:RYR(2) = 16
300  REM
310  REM   COLI(1) = % COST OF LIVING INCREASE YEARLY (LOW)
320  REM   COLI(2) = % COST OF LIVING INCREASE YEARLY (HIGH)
330  REM   COLI(3) = STEP
340  REM
350 COLI(1) = 12:COLI(2) = 12
```

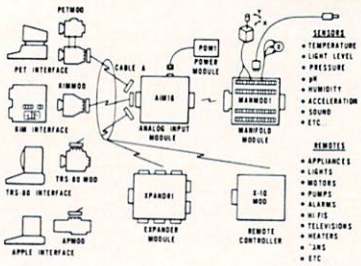
```
360  REM
370  REM  MPAY(1) = MONTHLY PAYMENT (LOW)
380  REM  MPAY(2) = MONTHLY PAYMENT (HIGH)
390  REM  MPAY(3) = STEP
400  REM
410 MPAY(1) = 650:MPAY(2) = 800
420  REM
430  REM  SAL(0) = MONTHLY SALARY
440  REM  SAL(1) = % RATE OF SALARY INCREASE PER YEAR (LOW)
450  REM   SAL(2) = % RATE OF SALARY INCREASE PER YEAR (HIGH)
460  REM  SAL(3) = STEP
470  REM
480 SAL(0) = 2000:SAL(1) = 12:SAL(2) = 12
490  REM
500  REM  SUBROUTINES
510  REM
520  REM  DUMMY AND DUMMY() ARE VARIABLES USED IN SUBROUTINES
530  REM  CARRIAGE RETURN ALONE RETAINS OLD ANSWER
540  REM
550  REM  GOSUB 10000 - GET A RANGE OF VARIABLES
560  REM  GOSUB 11000 - GET A VARIABLE
570  REM  GOSUB 12000 - GET A YES/NO ANSWER
580  REM  GOSUB 13000 - GET THE INPUT LINE
990  REM  GET THE NEW VALUES
992  REM
1000  PRINT : PRINT : PRINT "INITIAL LOAN AMOUNT $"
1010 DUMMY = ILOAN: GOSUB 11000:ILOAN = DUMMY
1020  REM
1030  PRINT : PRINT "MONTHLY SALARY $"
1040 DUMMY = SAL(0): GOSUB 11000:SAL(0) = DUMMY
1050  PRINT : PRINT "SALARY YEARLY INCREASE %"
1060 DUMMY(1) = SAL(1):DUMMY(2) = SAL(2): GOSUB 10000
1070 SAL(1) = DUMMY(1):SAL(2) = DUMMY(2):SAL(3) = DUMMY(3)
1080  REM
1090  PRINT "COST OF LIVING INCREASE %"
1100 DUMMY(1) = COLI(1):DUMMY(2) = COLI(2): GOSUB 10000
1110 COLI(1) = DUMMY(1):COLI(2) = DUMMY(2):COLI(3) = DUMMY(3)
1120  REM
1130  PRINT "MONTHLY PAYMENT RATE $"
1140 DUMMY(1) = MPAY(1):DUMMY(2) = MPAY(2): GOSUB 10000
1150 MPAY(1) = DUMMY(1):MPAY(2) = DUMMY(2):MPAY(3) = DUMMY(3)
1160  REM
1170  PRINT "LOAN INTEREST RATE YEARLY %"
1180 DUMMY(1) = RYR(1):DUMMY(2) = RYR(2): GOSUB 10000
1190 RYR(1) = DUMMY(1):RYR(2) = DUMMY(2):RYR(3) = DUMMY(3)
1200  REM
1210  PRINT "PRINT OUT END-OF-YEAR TOTALS? Y/N ";
1220  GOSUB 12000:EOY$ = YN$
1990  REM
1991  REM  ACTUAL CALCULATIONS
1992  REM
2000  REM     LOOP ON INTEREST RATES - ADJUST TO MONTH AND FRACTION
2010  FOR RYR = RYR(1) / 1200 TO RYR(2) / 1200 STEP RYR(3) / 1200
2020  REM
2030  REM    COST OF LIVING LOOP - ADJUST TO FRACTION
2040  FOR COLI = COLI(1) / 100 TO COLI(2) / 100 STEP COLI(3) / 100
2050  REM  AV IS ACTUAL VALUE OF A STARTING DOLLAR AT END OF YEAR
2060 AV = 1 / (1 + COLI)
2070  REM
```

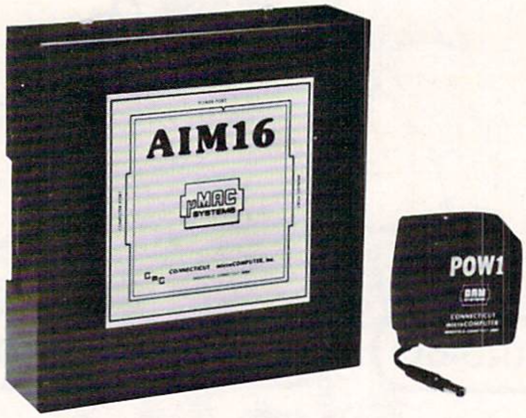
```
2080  REM     SALARY INCREASE LOOP - ADJUST TO FRACTION
2090  FOR SAL = SAL(1) / 100 TO SAL(2) / 100 STEP SAL(3) / 100
2100  REM
2110  PRINT : PRINT "LOAN $";ILOAN;"  INTEREST RATE "; INT (RYR * 12
   000 + 0.5) / 10"%"
2120  PRINT "INFLATION " INT (COLI * 1000 + 0.5) / 10"%  ";
2130  PRINT "SALARY $";SAL(0)"/MONTH"
2140  PRINT "SALARY INCREASE " INT (SAL * 1000 + 0.5) / 10"% "
2150  REM
2160  REM   MONTHLY PAYMENT LOOP
2170  FOR MPAY = MPAY(1) TO MPAY(2) STEP MPAY(3)
2180  REM  INITIAL VARIABLES
2190 OLOAN = ILOAN: REM  LOAN AMOUNT
2200 TPAY = 0: REM  TOTAL PAY
2210 APAY = 0: REM  ADJUSTED PAYMENT
2220 IMPACT = 0: REM  SALARY IMPACT
2230  REM  CAN'T BE DONE IF INTEREST INCREASE ABOVE REPAYMENTS
2240  IF OLOAN * RYR < MPAY THEN 2270
2250  PRINT : PRINT "CAN'T BE DONE @ $"; INT (MPAY * 100) / 100;"/MO
   NTH": GOTO 2530: PRINT
2260  REM  LOOP OVER A MAX OF 50 YEARS
2270  PRINT : FOR M = 1 TO 6000
2280  REM  IN WHAT YEAR AND WHAT MONTH?
2290 YR =  INT ((M - 1) / 12):MM = M - YR * 12
2300  REM  NEW INTEREST SINCE LAST PAYMENT
2310 MI = OLOAN * RYR
2320  REM  OUTSTANDING LOAN AFTER NEW PAYMENT
2330 OLOAN = OLOAN + MI - MPAY
2340  REM  TOTAL AMOUNT PAID
2350 TPAY = TPAY + MPAY
2360  REM  INFLATION ADJUSTED AMOUNT PAID
2370 APAY = APAY + MPAY * AV ^ YR
2380  REM  SALARY IMPACT
2390 IMPACT = IMPACT + MPAY / (SAL(0) * (1 + SAL) ^ YR)
2400  REM  LOAN PAID OUT?
2410  IF OLOAN < 0 THEN 2500
2420  IF EOY$ = "N" OR MM <  > 12 THEN  NEXT M: GOTO 2460
2430  PRINT "$"; INT (MPAY * 100) / 100;" YR "YR + 1" T $" INT (TPAY
   );
2440  PRINT " A $" INT (APAY)" I " INT (IMPACT * 100 / M)
2450  NEXT M
2460  PRINT "OVER FIFTY YEARS"
2470  REM
2480  REM  PRINT ON THE SCREEN
2490  REM
2500  PRINT "AFTER ";YR;" YRS ";MM;" M @ $";MPAY;" /MONTH"
2510  PRINT "TOTAL $"; INT (TPAY),"ADJUSTED $"; INT (APAY)
2520  PRINT "IMPACT FACTOR "; INT (IMPACT * 100 / M)
2530  NEXT MPAY
2540  NEXT SAL
2550  NEXT COLI
2560  NEXT RYR
2570  REM   DO IT AGAIN
2580  PRINT : PRINT "AGAIN Y/N ";: GOSUB 12000
2590  IF YN$ = "Y" THEN 1000
2600  STOP
9990  REM
9992  REM  GET A RANGE OF VALUES
9994  REM
```

```
10000  PRINT "LOW ";:DUMMY = DUMMY(1): GOSUB 11000:DUMMY(1) = DUMMY
10010  PRINT "HIGH ";:DUMMY = DUMMY(2): GOSUB 11000:DUMMY(2) = DUMMY

10020  IF DUMMY(2) < DUMMY(1) THEN  PRINT "INVALID": GOTO 10000
10030  REM   IF A RANGE THEN USE 5 STEPS
10040 DUMMY(3) = (DUMMY(2) - DUMMY(1)) / 5
10050  REM  WATCH FOR ZERO STEPS - NASTY THINGS
10060  IF DUMMY(3) = 0 THEN DUMMY(3) = 1
10070  PRINT : RETURN
10990  REM
10991  REM  GET A VALUE
10992  REM
11000  PRINT "PRESENT VALUE ";DUMMY: PRINT "NEW VALUE ";: GOSUB 1300
 0
11010  REM   RETAIN OLD VALUE?
11020  IF DUMMY$ = "" THEN DUMMY$ =  STR$ (DUMMY)
11030  REM  CHECK IF ZERO VALUE
11040 DUMMY =  VAL (DUMMY$): IF DUMMY > 0 THEN  RETURN
11050  IF DUMMY < 0 THEN  PRINT "INVALID": GOTO 11000
11060  PRINT "ZERO VALUE OKAY? Y/N ";: GOSUB 12000
11070  IF YN$ = "N" THEN  PRINT "CORRECT VALUE": GOTO 11000
11080  RETURN
11990  REM
11991  REM  GET A YES OR NO ANSWER
11992  REM
11993  REM   NULL RETURNS ARE ILLEGAL
12000  GOSUB 13000: IF DUMMY$ = "" THEN 12040
12010  REM  FIRST CHARACTER ONLY OF INTEREST
12020 YN$ =  LEFT$ (DUMMY$,1)
12030  IF YN$ = "Y" OR YN$ = "N" THEN  RETURN
12040  PRINT "ANSWER YES OR NO ";: GOTO 12000
12990  REM
12991  REM  GET AN INPUT LINE
12992  REM
13000 DUMMY$ = ""
13010  GET A$
13020  REM  IS IT A BACKSPACE? THEN DELETE WHOLE LINE
13030  IF A$ =  CHR$ (8) THEN  PRINT "**DELETED**": GOTO 13000
13040  PRINT A$;: IF A$ =  CHR$ (13) THEN  RETURN
13050 DUMMY$ = DUMMY$ + A$: GOTO 13010

]
```

©

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





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

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

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

Connectors

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

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

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

Analog Input Module

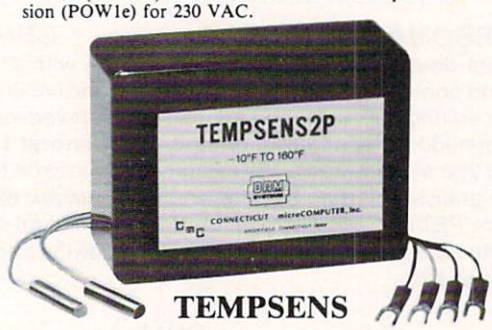
The AIM 16 is a 16 channel analog to digital converter designed to work with most microcomputers. The AIM 16 is connected to the host computer through the computer's 8 bit input port and 8 bit output port, or through one of the uMAC SYSTEMS special interfaces.

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

Power requirements are 12 volts DC at 60 ma.

POW1

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



TEMPSENS

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

Remote Controller-Clock and Calendar

AN INEXPENSIVE CONTROL SOLUTION FOR

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

SUPER X-10 MOD SPECS

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

XPANDR1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Printer And Communication Interfaces For The CBM/PET

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

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

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

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

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

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

ADA1600 • For Parallel NEC and Centronics Standard Printers

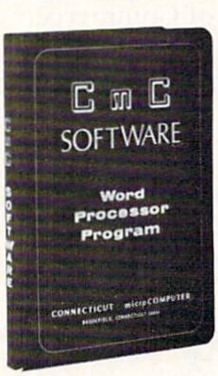
The ADA1600 is a low cost easy to use interface for the Commodore Computers. It allows the PET and CBM computers to use standard Centronics type printers (including the NEC 5530) for improved quality printing. The ADA1600 has a two foot cable which plugs into the PET IEEE port. Another IEEE card edge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

ADA1450 • Serial Printer Adapters

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

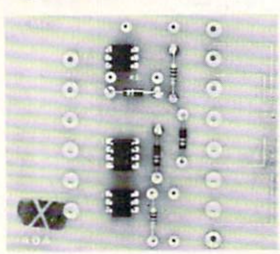
ADA730 Parallel • For the Centronics 730 and 737 Printers

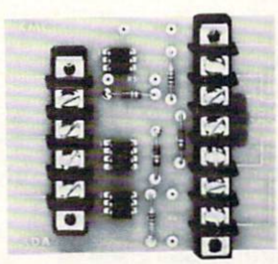
The ADA730 is a low cost easy to use interface for the Commodore Computers. It allows the PET and CBM computers to use Centronics type 730 and 737 printers. The ADA730 has a two foot cable which plugs into the PET IEEE port. Another IEEE card edge connector is provided for connecting disks and other peripherals to the PET. The ADA730 is addressable and does not tie up the bus. The address is switch selectable. A cable with a 36 pin card edge connector is provided. A switch selects upper/lower coase, upper/lower case reversed (needed for some Commodore machines) and upper case only for clearer program listings. Works with WORDPRO, BASIC and other software. No special programming is required. The case measures 3 1/2 x 5 3/4 inches. Comes complete, assembled and tested, with case and cables. Power is obtained from the printer or an external power supply may be used. Retail price for the ADA is $149.



Word Processor Program •

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





ADA400 RS-232 To Current Loop Adapter

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

ADA4005 - Solder Pads 24.50
ADA400B - Barrier Strips with screw terminals 29.50

QUANTITY	DESCRIPTION	PRICE	TOTAL

SUBTOTAL
Handling and shipping — add per order $4.00
Foreign orders add 10% for AIR postage
Conn. residents add 7% sales tax
TOTAL ENCLOSED
NAME ____
COMPANY ____
ADDRESS ____
CITY ____
STATE ____ ZIP ____
VISA O M/C O Expiration date ____
Card number ____
Mention this magazine and deduct 2% from the TOTAL

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

CmC

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

Home Heating And Cooling Audit

David E. Pitts
Houston, TX

Have you, like thousands of Americans, added insulation, storm windows, a setback thermostat, and caulking to improve the energy efficiency of your home? Other than the 15% energy credit you could claim on your taxes starting in 1979, it is difficult to know what savings one is achieving with these substantial investments of time and money. A colder than normal winter will cause increased fuel usage for heating which may or may not overshadow the energy savings by insulating. On the other hand, last winter (1979-80) was so mild in most parts of the United States that it brought significant fuel savings for most homeowners whether they insulated or not. However, energy costs have increased so much in some areas and for some fuels that these consumers may not have achieved a monetary savings.

The cost for heating or cooling a house is due to three things:

1) outside temperature
2) thermostat setting
3) insulation (including air infiltration)

Only the last two are under the homeowner's control. The most cost-effective action a homeowner can take is to raise the thermostat in the summer and lower the thermostat in winter. The next most effective is to increase the insulation. Having done this, the fuel use will still be driven by the outside temperature. In order to compare the severity and predict fuel use, meteorologists have developed two concepts:

1) Heating degree day
2) Cooling degree day

Heating degree day is an estimate of the heating necessary in the winter and cooling degree day is an estimate of the cooling necessary in the summer.

... energy fuel savings as well as economic savings can be calculated ...

Both are calculated from the maximum and minimum temperatures and summed each day to accumulate monthly and yearly totals. Heating degree days accumulate on days with an average temperature cooler than 65° F, and cooling degree days accumulate on days with an average temperature warmer than 65° F. These data are recorded for several hundred stations in the United States and are available in a publication "Local Climatological Data" from the U.S. Dept. of Commerce,

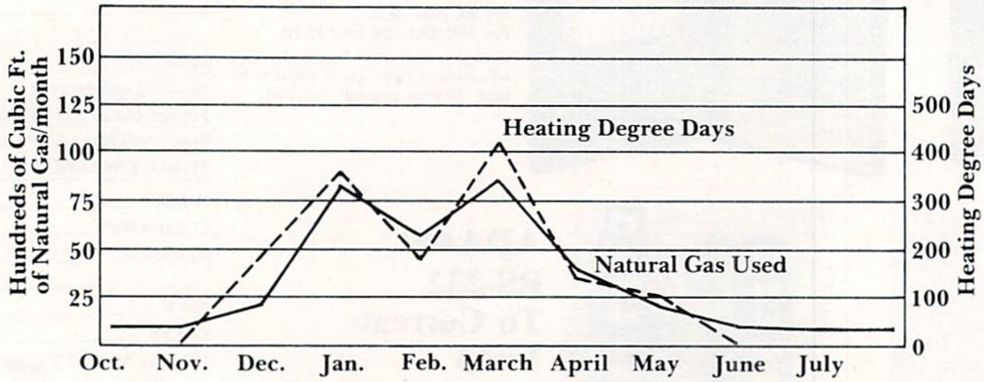


Figure 1. Heating Degree Days by Month for 1979-80 in Houston, TX and Natural Gas Used in the Author's Residence.

```
RUN
                YEAR BY YEAR HEATING CONSERVATION AUDIT


--------------------------------------------------------------
STATE (DON'T ABBREVIATE)? TEXAS

THE WINTER OF 1973-74 IS CALLED 74, CHOICES ARE 74 TO 80

STARTING YEAR? 78
LAST YEAR? 80

CHOICES OF INPUT ARE BY YEAR OR MONTH

BY YEAR (Y OR N)? Y

UNITS OF FUEL CAN BE ANYTHING: GALLONS, KWH, CUFT, 100CUFT
ALL FUEL ENTRIES MUST BE THE SAME UNITS
--------------------------------------------------------------

YEAR=  78
FUEL USE FOR OCT 1 TO MAY 1? 650
COST(DOLLARS)? 205.05
--------------------------------------------------------------
YEAR=  79
FUEL USE FOR OCT 1 TO MAY 1? 526
COST(DOLLARS)? 182.70
--------------------------------------------------------------
YEAR=  80
FUEL USE FOR OCT 1 TO MAY 1? 318
COST(DOLLARS)? 120.60
--------------------------------------------------------------
FUEL USE FOR JULY? 10
               STATE     CITY
           1   TEXAS     BROWNSVILLE
           2   TEXAS     AMARILLO
           3   TEXAS     FORT WORTH
           4   TEXAS     HOUSTON
CHOOSE # OF CITY? 4

--------------------------------------------------------------
CHOSEN CITY= HOUSTON                   RATE(1ST YEAR)=       .31

     YEAR        RATE      FUEL SAVINGS     SAVINGS(DOLLARS)
      79         .34           15.88          5.51
      80         .37           178.1          67.54

--------------------------------------------------------------
                    (+ = SAVINGS)(- = LOSS)
           1   TEXAS     BROWNSVILLE
           2   TEXAS     AMARILLO
           3   TEXAS     FORT WORTH
           4   TEXAS     HOUSTON
CHOOSE # OF CITY?

OK
```

National Climatic Center, Federal Building, Asheville, NC 28801. The concepts of the Cooling and Heating degree days have shown excellent correlation with fuel use in the author's residence (see figure 1 and 2) both in heating and air conditioning, on a month by month basis and an even higher correlation for an entire season. This correlation prompted the author to develop a BASIC program

The programs can evaluate efficiency from the year 1974, through and including 1980.

for calculating an energy usage rate in one year and predicting energy usage in the following years based on degree days. Using this technique, energy fuel savings as well as economic savings can be calculated even though the weather, energy cost, and energy efficiency of the home are changing month by month and year by year.

Two programs were written for the OSI-4PMF in "plain Jane" BASIC so as to be easily converted to other machines. Each program requires less than 8K and can be shortened considerably by selectively eliminating data statements to restrict the geographical coverage. Each program requires the homeowner to have records of fuel use and cost for two years or more. The programs can evaluate efficiency from the years 1974, through and including 1980. Any type of fuel can be utilized, just remember that the units you input will be the units calculated for the fuel savings. Likewise the rate is given as cost/fuel units, and so is dependent upon the units you input. Changing fuels or changing residences invalidates the technique. The heating season is from October 1 to May 1 and the cooling season is from April 1 to November 1 and are made extra long in order to accomodate the wide range of climates in the United States. Because many fuels are used for other purposes such as hot water heating, home lighting, etc., the off season minimum usage is used to remove these factors from the seasonal weather effects. Thus the heating program requests the July fuel use and the cooling program requests the January fuel use. Should a user live between the cities, listed runs for all cities in that region will allow interpolation. The following are some key variables:

ST$ = state
CT$ = city
H(1,I) = degree days for 1974 for city I
x = fuel use/degree days for base year
H = predicted fuel use minus actual fuel used
RATE(k) = cost/fuel unit
F(k) = fuel unit
D(k) = cost
k = year
MI = fuel use in minimum month

The precision of this technique is good, but may predict small savings or loss in years when no energy conservation practices were in effect. This uncertainty is due to the variance between day and night temperatures which is not always well represented by the mean temperature for the day.

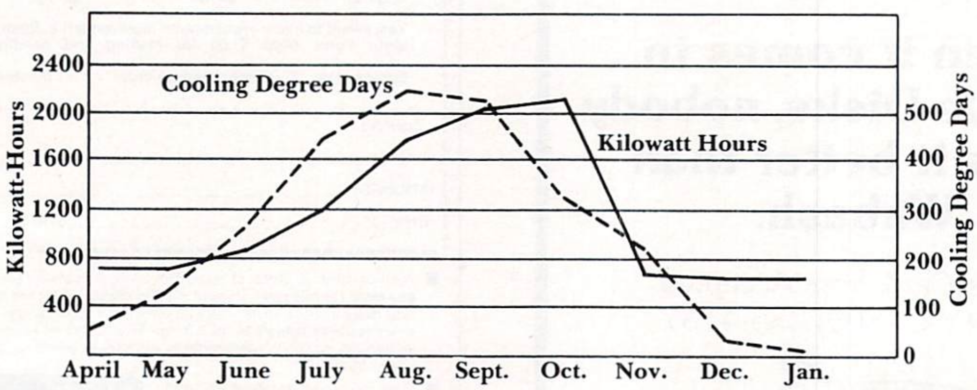


Figure 2. Cooling Degree Days by Month for 1979 in Houston, TX and Kilowatt Hours used in the Author's Residence

```
RUN
              YEAR BY YEAR COOLING CONSERVATION AUDIT

--------------------------------------------------------------
STATE (DON'T ABBREVIATE)? TEXAS

THE SUMMER OF 1974 IS CALLED 74, CHOICES ARE 74 TO 80

STARTING YEAR? 78
LAST YEAR? 80

CHOICES OF INPUT ARE BY YEAR OR MONTH

BY YEAR (Y OR N)? Y

UNITS OF FUEL CAN BE ANYTHING: GALLONS, KWH, CUFT, 100CUFT
ALL FUEL ENTRIES MUST BE THE SAME UNITS
--------------------------------------------------------------

YEAR=  78
FUEL USE FOR APR 1 TO NOV 1? 10422
COST(DOLLARS)? 374.28
--------------------------------------------------------------
YEAR=  79
FUEL USE FOR APR 1 TO NOV 1? 9483
COST(DOLLARS)? 402.56
--------------------------------------------------------------
YEAR=  80
FUEL USE FOR APR 1 TO NOV 1? 10204
COST(DOLLARS)? 528.08
--------------------------------------------------------------
FUEL USE FOR JANUARY? 679
                STATE      CITY
           1    TEXAS      BROWNSVILLE
           2    TEXAS      AMARILLO
           3    TEXAS      FORT WORTH
           4    TEXAS      HOUSTON
CHOOSE # OF CITY? 4
--------------------------------------------------------------
CHOSEN CITY= HOUSTON              RATE(1ST YEAR)=         .03

     YEAR        RATE        FUEL SAVINGS        SAVINGS(DOLLARS)
      79         .04             367.35           15.59
      80         .05             734.26           37.99

--------------------------------------------------------------
                   (+ = SAVINGS)(- = LOSS)
           1    TEXAS      BROWNSVILLE
           2    TEXAS      AMARILLO
           3    TEXAS      FORT WORTH
           4    TEXAS      HOUSTON
CHOOSE # OF CITY?

OK
```

```
1 REM COOLING FUEL AUDIT
2 REM BY DAVID PITTS, 16011 STONEHAVEN DR. HOUSTON, TX 77059
10 REM
11 REM PROGRAM REQUIRES COOLING FUEL USE (ANNUAL OR BY MONTH)
12 REM COOLING MONTHS ARE APRIL 1 TO NOV 1, 2 YRS OR MORE REQUIRED
13 REM OUTPUT IS FUEL SAVINGS, AND $ SAVINGS
15 REM ********************************************
17 DEFFNTRC(E)=INT(E*100)/100
20 PRINTTAB(12);"YEAR BY YEAR COOLING CONSERVATION AUDIT":PRINT:PRINT
22 GOSUB500:INPUT"STATE (DON'T ABBREVIATE)";B$
23 FORI=1TO7:READM$(I):NEXT:PRINT
25 PRINT"THE SUMMER OF 1974 IS CALLED 74, CHOICES ARE 74 TO 80"
27 PRINT:INPUT"STARTING YEAR";YS:INPUT"LAST YEAR";YE:PRINT
30 L=YE-YS+1:PRINT"CHOICES OF INPUT ARE BY YEAR OR MONTH":PRINT
40 INPUT"BY YEAR (Y OR N)";A$:IFASC(A$)<>89THEN100
41 PRINT:PRINT"UNITS OF FUEL CAN BE ANYTHING: GALLONS, KWH, CUFT, 100CUFT"
43 PRINT"ALL FUEL ENTRIES MUST BE THE SAME UNITS":GOSUB500:PRINT
47 FORI=1TOL:PRINT"YEAR= ";INT(YS+I-1)
48 PRINT"FUEL USE FOR APR 1 TO NOV 1";:INPUTF(I):INPUT"COST(DOLLARS)";D(I)
90 GOSUB500:NEXT:GOTO200
100 FORI=1TOL:GOSUB500:PRINT"YEAR = ";INT(YS+I-1):FORJ=1TO7
105 PRINT"FUEL USE FOR ";M$(J);:INPUTF:PRINT"COST FOR ";M$(J);:INPUTD
110 F(I)=F(I)+F:D(I)=D(I)+D:NEXT:NEXT
200 INPUT"FUEL USE FOR JANUARY";MI:I=1
220 READST$,CT$(I),H(1,I),H(2,I),H(3,I),H(4,I),H(5,I),H(6,I),H(7,I)
230 IFLEFT$(ST$,7)=LEFT$(B$,7)THENI=I+1
240 IFST$="END"THEN250
245 GOTO220
250 J=I-1:LL=YS-74+1:PRINTTAB(15);"STATE";TAB(25);"CITY"
255 FORI=1TOJ
260 PRINTTAB(10);I;TAB(15);B$;TAB(25);CT$(I):NEXT
270 INPUT"CHOOSE # OF CITY";I:PRINT:GOSUB500
280 X=(F(1)-7*MI)/H(LL,I):RATE(1)=D(1)/F(1):H=FNTRC(RATE(1))
282 PRINT"CHOSEN CITY= ";CT$(I);TAB(37);"RATE(1ST YEAR)=";TAB(58);H
285 PRINT:PRINTTAB(5);"YEAR";
290 PRINTTAB(16);"RATE";TAB(25);"FUEL SAVINGS";TAB(42);"SAVINGS(DOLLARS)"
295 FORK=2TOL
300 H=H(LL+K-1,I)*X+7*MI-F(K):RATE(K)=D(K)/F(K):C=H*RATE(K)
312 H=FNTRC(H):RATE(K)=FNTRC(RATE(K)):C=FNTRC(C)
320 PRINTTAB(5);INT(YS+K-1);TAB(15);RATE(K);TAB(28);H;TAB(42);C
340 NEXT:PRINT:GOSUB500:PRINTTAB(20);"(+ = SAVINGS)(- = LOSS)":GOTO255
500 PRINT"------------------------------------------------------------":RETURN
1999 DATAAPRIL,MAY,JUNE,JULY,AUGUST,SEPTEMBER,OCTOBER
2000 DATATEXAS,BROWNSVILLE,3871,3857,3327,4023,4188,3689,3756
2010 DATATEXAS,AMARILLO,1396,1235,1013,1700,1556,1168,1666
2020 DATATEXAS,FORT WORTH,2578,2609,2251,3017,2965,2509,3142
2030 DATATEXAS,HOUSTON,2821,2656,2225,2751,2866,2577,3127
2032 DATAALABAMA,BIRMINGHAM,1640,1858,1427,2272,1975,1719,2177
2034 DATAALABAMA,MOBILE,2548,2732,2405,2846,2884,2442,2680
2036 DATAALABAMA,MONTGOMERY,1941,2349,1730,2630,2388,2033,2375
2038 DATAARIZONA,FLAGSTAFF,232,88,98,191,152,85,334
2040 DATAARIZONA,PHOENIX,4285,3785,3965,4521,4343,4186,3872
2042 DATAARIZONA,TUCSON,2788,2592,2760,3099,3184,3052,2844
2044 DATAARKANSAS,LITTLEROCK,1787,1941,1602,2266,2358,1926,2486
2046 DATACALIFORNIA,LOSANGELES,627,505,864,602,827,845,494
2048 DATACALIFORNIA,SANFRANCISCO,127,80,192,88,144,182,102
2050 DATACOLORADO,DENVER,715,554,667,799,748,661,950
2052 DATACONNECTICUTT,HARTFORD,764,870,819,905,657,811,787
2054 DATADELAWARE,WILMINGTON,1109,1101,1003,1120,1016,990,1333
2056 DATAFLORIDA,JACKSONVILLE,2460,2784,2179,2717,2559,2483,2647
```

```
2058 DATAFLORIDA,MIAMI,4657,4570,4014,4202,4183,4218,3486
2060 DATAFLORIDA,TALLAHASSEE,2472,2604,2292,2498,2480,2198,2458
2062 DATAGEORGIA,ATLANTA,1506,1600,1254,1735,1773,1762,2370
2064 DATAGEORGIA,SAVANNAH,2289,2574,1954,2643,2570,2390,2501
2066 DATAIDAHO,BOISE,851,789,535,822,597,752,511
2068 DATAIDAHO,POCATELLO,460,440,372,519,369,479,225
2070 DATAILLINOIS,CHICAGO,770,1124,906,1218,982,812,929
2072 DATAILLINOIS,SPRINGFIELD,984,1200,1021,1369,1253,1201,1452
2074 DATAINDIANA,EVANSVILLE,1229,1500,1112,1779,1550,1238,1672
2076 DATAINDIANA,FORTWAYNE,727,833,664,1032,898,677,844
2078 DATAINDIANA,INDIANAPOLIS,850,1046,770,1363,1300,882,1142
2080 DATAIOWA,DESMOINES,974,1237,1050,1342,1226,984,1262
2082 DATAIOWA,SIOUXCITY,980,1013,969,862,928,865,1071
2084 DATAKANSAS,TOPEKA,1173,1474,1294,1563,1434,1275,1810
2086 DATAKANSAS,WICHITA,1466,1512,1417,1841,2047,1663,2286
2088 DATAKENTUCKY,LOUISVILLE,1055,1506,1130,1717,1539,1236,1676
2090 DATALOUSIANA,BATONROUGE,2641,2618,2248,2772,2781,2379,2670
2092 DATALOUSIANA,NEWORLEANS,2655,2637,2390,2962,3059,2895,3030
2094 DATAMAINE,CARIBOU,116,271,231,223,264,290,211
2096 DATAMAINE,PORTLAND,296,351,308,308,336,316,468
2098 DATAMARYLAND,BALTIMORE,1038,1245,1149,1474,1274,1137,1407
2100 DATAMASSACHUSETTS,BOSTON,646,862,895,897,668,789,894
2102 DATAMICHIGAN,DETROIT,620,731,706,873,760,522,672
2104 DATAMICHIGAN,GRANDRAPIDS,400,619,638,714,588,614,667
2106 DATAMICHIGAN,SAULTST.MARIE,126,243,167,86,125,145,96
2108 DATAMINNESOTA,DULUTH,149,229,271,122,224,169,241
2110 DATAMINNESOTA,INT.FALLS,259,328,281,187,219,131,327
2112 DATAMINNESOTA,MINNEAPOLIS,619,850,950,691,811,651,776
2114 DATAMISSISSIPPI,JACKSON,2036,2300,1943,2535,2421,1947,2578
2116 DATAMISSOURI,KANSASCITY,1159,1534,1296,1403,1535,1174,1721
2118 DATAMISSOURI,ST.LOUIS,1175,1431,1229,1667,1519,1578,1907
2120 DATAMISSOURI,SPRINGFIELD,1095,1349,1085,1692,1565,1122,1850
2122 DATAMONTANA,BILLINGS,572,478,547,545,429,716,670
2124 DATAMONTANA,GREATFALLS,473,315,389,293,332,396,302
2126 DATAMONTANA,MISSOULA,303,258,158,295,185,390,125
2128 DATANEBRASKA,LINCOLN,1128,1282,1178,1250,1175,1098,1552
2130 DATANEBRASKA,OMAHA,1021,1389,1249,1364,1321,1124,1374
2132 DATANEVADA,LASVEGAS,3403,2973,2809,3332,3300,3387,3035
2134 DATANEVADA,RENO,258,348,236,500,340,404,400
2136 DATANEW HAMPSHIRE,CONCORD,302,553,419,488,521,519,450
2138 DATANEW JERSEY,TRENTON,977,1070,1053,1163,976,936,1288
2140 DATANEW MEXICO,ALBUQUERQUE,1353,1100,1141,1335,1398,1508,1519
2142 DATANEW MEXICO,ROSWELL,1655,1437,1786,2428,1991,1684,2061
2144 DATANEW YORK,ALBANY,386,597,476,574,456,636,561
2146 DATANEW YORK,NEWYORK,1017,953,1158,1097,919,1049,1276
2148 DATANEW YORK,SYRACUSE,405,555,357,520,623,595,687
2150 DATANORTH CAROLINA,ASHEVILLE,731,795,545,1007,973,792,1166
2152 DATANORTH CAROLINA,RALEIGH,1325,1452,1459,1701,1671,1275,1742
2154 DATANORTH CAROLINA,WILMNGTN,1978,2373,1936,2310,2097,1966,2230
2156 DATANORTH DAKOTA,FARGO,444,553,766,487,604,504,580
2158 DATANORTH DAKOTA,WILLISTON,440,376,507,410,421,415,600
2160 DATAOHIO,COLUMBUS,836,1147,608,1073,968,808,1008
2162 DATAOHIO,TOLEDO,608,692,599,784,741,602,740
2164 DATAOKLAHOMA,OKLACITY,1651,1615,1702,2163,2418,1805,2479
2166 DATAOREGON,BURNS,406,347,227,451,277,367,106
2168 DATAOREGON,PORTLAND,409,342,216,375,343,462,270
2170 DATAPENNSYLVANIA,HARRISBURG,1163,1026,901,1053,1015,828,1183
2172 DATAPENNSYLVANIA,PITTSBURG,657,721,358,623,836,620,876
2174 DATAPENNSYLVANIA,PHILADELPHIA,1165,1243,1211,1237,1247,1097,1410
2176 DATARHODE ISLAND,PROVIDENCE,666,694,631,821,610,640,789
```

```
2178 DATASOUTH CAROLINA,CHARLESTON,2044,2408,1885,2584,2319,2204,2258
2180 DATASOUTH CAROLINA,GREENVILLE,1435,1547,1135,1958,1559,1296,1710
2182 DATASOUTH DAKOTA,RAPIDCITY,697,583,676,574,669,550,665
2184 DATASOUTH DAKOTA,SIOUXFALLS,751,910,1040,854,743,724,793
2186 DATATENNESSEE,CHATTANOOGA,1058,1365,1165,2095,1847,1432,1778
2188 DATATENNESSEE,KNOXVILLE,1340,1530,1133,1811,1612,1355,1814
2190 DATATENNESSEE,MEMPHIS,1840,2184,1800,2681,2357,2088,2683
2192 DATAUTAH,SALTLAKECITY,1191,900,943,1108,1018,1274,990
2194 DATAVERMONT,BURLINGTON,442,699,483,507,489,531,503
2196 DATAVIRGINIA,NORFOLK,1531,1744,1558,1930,1535,1433,1788
2198 DATAVIRGINIA,RICHMOND,1259,1433,1385,1814,1573,1375,1681
2200 DATAWASHINGTON,SEATTLE,196,197,129,232,210,171,57
2202 DATAWASHINGTON,SPOKANE,405,340,293,472,326,496,228
2204 DATAWEST VIRGINIA,CHRLSTON,910,1074,801,1227,1114,894,1123
2206 DATAWISCONSIN,GREENBAY,323,514,520,534,440,380,451
2208 DATAWISCONSIN,MADISON,457,742,627,622,589,450,630
2210 DATAWYOMING,CHEYENNE,349,193,217,252,297,352,415
2990 DATAWYOMING,LANDER,467,333,394,410,408,436,423
2995 DATAEND,END,0,0,0,0,0,0,0
3000 END
```

```
1 REM HEATING FUEL AUDIT
2 REM BY DAVID PITTS, 16011 STONEHAVEN DR. HOUSTON, TX 77059
10 REM
11 REM PROGRAM REQUIRES HEATING FUEL USE(ANNUAL OR BY MONTH)
12 REM HEATING MONTHS ARE OCT 1 TO MAY 1, 2 YRS OR MORE REQUIRED
13 REM OUTPUT IS FUEL SAVINGS, AND $ SAVINGS
15 REM ***********************************************************
17 DEFFNTRC(E)=INT(E*100)/100
20 PRINTTAB(12);"YEAR BY YEAR HEATING CONSERVATION AUDIT":PRINT:PRINT
22 GOSUB500:INPUT"STATE (DON'T ABBREVIATE)";B$
23 FORI=1TO7:READM$(I):NEXT:PRINT
25 PRINT"THE WINTER OF 1973-74 IS CALLED 74, CHOICES ARE 74 TO 80"
27 PRINT:INPUT"STARTING YEAR";YS:INPUT"LAST YEAR";YE:PRINT
30 L=YE-YS+1:PRINT"CHOICES OF INPUT ARE BY YEAR OR MONTH":PRINT
40 INPUT"BY YEAR (Y OR N)";A$:IFASC(A$)<>89THEN100
41 PRINT:PRINT"UNITS OF FUEL CAN BE ANYTHING: GALLONS, KWH, CUFT, 100CUFT"
43 PRINT"ALL FUEL ENTRIES MUST BE THE SAME UNITS":GOSUB500:PRINT
47 FORI=1TOL:PRINT"YEAR= ";INT(YS+I-1)
48 PRINT"FUEL USE FOR OCT 1 TO MAY 1";:INPUTF(I):INPUT"COST(DOLLARS)";D(I)
90 GOSUB500:NEXT:GOTO200
100 FORI=1TOL:GOSUB500:PRINT"YEAR = ";INT(YS+I-1):FORJ=1TO7
105 PRINT"FUEL USE FOR ";M$(J);:INPUTF:PRINT"COST FOR ";M$(J);:INPUTD
110 F(I)=F(I)+F:D(I)=D(I)+D:NEXT:NEXT
200 INPUT"FUEL USE FOR JULY";MI:I=1
220 READST$,CT$(I),H(1,I),H(2,I),H(3,I),H(4,I),H(5,I),H(6,I),H(7,I)
230 IFLEFT$(ST$,7)=LEFT$(B$,7)THENI=I+1
240 IFST$="END"THEN250
245 GOTO220
250 J=I-1:LL=YS-74+1:PRINTTAB(15);"STATE";TAB(25);"CITY"
255 FORI=1TOJ
260 PRINTTAB(10);I;TAB(15);B$;TAB(25);CT$(I):NEXT
270 INPUT"CHOOSE # OF CITY";I:PRINT:GOSUB500
280 X=(F(1)-7*MI)/H(LL,I):RATE(1)=D(1)/F(1):H=FNTRC(RATE(1))
282 PRINT"CHOSEN CITY= ";CT$(I);TAB(37);"RATE(1ST YEAR)=";TAB(58);H
285 PRINT:PRINTTAB(5);"YEAR";
290 PRINTTAB(16);"RATE";TAB(25);"FUEL SAVINGS";TAB(42);"SAVINGS(DOLLARS)"
295 FORK=2TOL
300 H=H(LL+K-1,I)*X+7*MI-F(K):RATE(K)=D(K)/F(K):C=H*RATE(K)
```

```
312 H=FNTRC(H):RATE(K)=FNTRC(RATE(K)):C=FNTRC(C)
320 PRINTTAB(5);INT(YS+K-1);TAB(15);RATE(K);TAB(28);H;TAB(42);C
340 NEXT:PRINT:GOSUB500:PRINTTAB(20);"(+ = SAVINGS)(- = LOSS)":GOTO255
500 PRINT"-------------------------------------------------------------------":RETURN
1999 DATAOCTOBER,NOVEMBER,DECEMBER,JANUARY,FEBRUARY,MARCH,APRIL
2000 DATATEXAS,BROWNSVILLE,418,520,518,974,800,728,640
2010 DATATEXAS,AMARILLO,3389,4163,3484,4515,4084,4540,4219
2020 DATATEXAS,FORT WORTH,1854,2281,1841,2967,2941,2730,2375
2030 DATATEXAS,HOUSTON,1157,1190,1309,2276,2103,1711,1545
2032 DATAALABAMA,BIRMINGHAM,2138,2570,2527,3488,3295,2777,2766
2034 DATAALABAMA,MOBILE,1037,1365,1393,2400,2206,1617,1608
2036 DATAALABAMA,MONTGOMERY,1643,1967,2119,3038,2403,1987,2028
2038 DATAARIZONA,FLAGSTAFF,6080,6740,6158,6032,4882,6813,6100
2040 DATAARIZONA,PHOENIX,1093,1558,1089,1071,692,1428,1022
2042 DATAARIZONA,TUCSON,1652,2183,1453,1644,1194,1840,1349
2044 DATAARKANSAS,LITTLEROCK,2645,3059,2763,3590,3723,3528,3142
2046 DATACALIFORNIA,LOSANGELES,1232,1305,1160,969,705,1452,808
2048 DATACALIFORNIA,SANFRANCISCO,2752,2918,2929,2594,1972,2774,2116
2050 DATACOLORADO,DENVER,5569,5826,5117,5258,4882,5937,5333
2052 DATACONNECTICUTT,HARTFORD,5540,5890,5349,6164,5711,6286,5569
2054 DATADELAWARE,WILMINGTON,3910,4676,4177,5206,4980,4883,4364
2056 DATAFLORIDA,JACKSONVILLE,933,1168,1390,2061,1791,1525,1406
2058 DATAFLORIDA,MIAMI,131,59,202,311,331,185,204
2060 DATAFLORIDA,TALLAHASSEE,1106,1547,1594,2199,2166,1746,1692
2062 DATAGEORGIA,ATLANTA,2305,2873,2697,3834,3298,2757,2737
2064 DATAGEORGIA,SAVANNAH,1274,1537,1735,2527,2253,1751,1881
2066 DATAIDAHO,BOISE,4977,5318,5376,5715,4287,5984,4792
2068 DATAIDAHO,POCATELLO,6387,6713,6252,6474,5103,7190,5839
2070 DATAILLINOIS,CHICAGO,5634,6039,5135,6613,6322,6686,5537
2072 DATAILLINOIS,SPRINGFIELD,4998,5433,4693,6157,6057,6075,5308
2074 DATAINDIANA,EVANSVILLE,3873,4424,3960,5236,5113,4979,4676
2076 DATAINDIANA,FORTWAYNE,5660,6093,5198,6723,6472,6271,6046
2078 DATAINDIANA,INDIANAPOLIS,4698,5477,4762,6260,5698,5748,5484
2080 DATAIOWA,DESMOINES,5908,6468,5268,6418,6606,7041,5827
2082 DATAIOWA,SIOUXCITY,6120,6924,5946,6961,7020,7912,6263
2084 DATAKANSAS,TOPEKA,4873,5225,4408,5455,5556,6023,5045
2086 DATAKANSAS,WICHITA,4540,4820,4035,4702,4855,5310,4620
2088 DATAKENTUCKY,LOUISVILLE,3697,4289,3694,5016,4896,4583,4392
2090 DATALOUSIANA,BATONROUGE,1050,1458,1548,2133,1996,1744,1762
2092 DATALOUSIANA,NEWORLEANS,931,1295,1430,2057,1860,1453,1447
2094 DATAMAINE,CARIBOU,8980,9024,8947,9140,8152,8638,7860
2096 DATAMAINE,PORTLAND,6472,6747,6709,7462,6600,7040,6427
2098 DATAMARYLAND,BALTIMORE,4241,4264,3857,4940,4542,4508,4271
2100 DATAMASSACHUSETTS,BOSTON,4998,5230,4620,5492,4963,5425,5017
2102 DATAMICHIGAN,DETROIT,5923,6375,5583,6754,6408,6538,6088
2104 DATAMICHIGAN,GRANDRAPIDS,6338,6987,5933,7167,6605,6944,5898
2106 DATAMICHIGAN,SAULTST.MARIE,8576,8602,8079,9047,8245,8848,8021
2108 DATAMINNESOTA,DULUTH,9292,9435,8662,9310,8657,9577,8351
2110 DATAMINNESOTA,INT.FALLS,9844,9755,9435,10044,9858,10745,9442
2112 DATAMINNESOTA,MINNEAPOLIS,7560,7969,6785,7800,7789,8132,7140
2114 DATAMISSISSIPPI,JACKSON,1746,2066,2058,2961,2881,2451,2568
2116 DATAMISSOURI,KANSASCITY,4775,5407,4401,5550,5671,5811,5106
2118 DATAMISSOURI,ST.LOUIS,4507,5001,4173,5466,5410,5368,4574
2120 DATAMISSOURI,SPRINGFIELD,3982,4659,3837,5033,4973,5116,4140
2122 DATAMONTANA,BILLINGS,6294,7106,6118,6076,7068,7878,5814
2124 DATAMONTANA,GREATFALLS,6810,7482,6503,6006,7606,8138,6164
2126 DATAMONTANA,MISSOULA,6797,7104,6668,6896,6423,8068,6439
2128 DATANEBRASKA,LINCOLN,6067,6504,5302,6131,6484,6881,5562
2130 DATANEBRASKA,OMAHA,6069,6316,5037,6045,6140,6391,5954
2132 DATANEVADA,LASVEGAS,2418,2610,2298,2150,1664,2517,2147
```

```
2134 DATANEVADA,RENO,5184,5820,5548,5196,4228,5679,4625
2136 DATANEW HAMPSHIRE,CONCORD,6924,7304,7194,7732,7094,7229,6479
2138 DATANEW JERSEY,TRENTON,4373,4763,4172,5355,5056,4818,4595
2140 DATANEW MEXICO,ALBUQUERQUE,4206,4707,4328,4761,3543,4020,3735
2142 DATANEW MEXICO,ROSWELL,3015,3660,2771,3469,2712,3585,3297
2144 DATANEW YORK,ALBANY,6539,6835,5999,6989,6315,6806,6023
2146 DATANEW YORK,NEWYORK,4333,4643,4131,5195,4804,4950,4434
2148 DATANEW YORK,SYRACUSE,6241,6439,5917,6806,6234,6573,5895
2150 DATANORTH CAROLINA,ASHEVILLE,3375,3947,3843,4755,4281,3882,3793
2152 DATANORTH CAROLINA,RALEIGH,2758,3550,2895,4258,3801,3286,3464
2154 DATANORTH CAROLINA,WILMNGTN,1683,2249,1908,2847,2658,2256,2464
2156 DATANORTH DAKOTA,FARGO,9171,8502,7937,8893,9012,9915,8402
2158 DATANORTH DAKOTA,WILLISTON,8714,8616,8081,8192,8867,9784,7787
2160 DATAOHIO,COLUMBUS,4701,5314,4860,6494,5860,5653,5253
2162 DATAOHIO,TOLEDO,5996,6243,5674,7093,6673,6520,5992
2164 DATAOKLAHOMA,OKLACITY,3278,3762,2950,3835,3977,4142,3543
2166 DATAOREGON,BURNS,6395,6587,6880,6102,5711,7093,5830
2168 DATAOREGON,PORTLAND,4070,3993,3992,4057,3715,4577,3690
2170 DATAPENNSYLVANIA,HARRISBURG,4509,5199,4498,5437,5059,4915,4422
2172 DATAPENNSYLVANIA,PITTSBURG,5005,5516,5105,6822,5636,5964,5536
2174 DATAPENNSYLVANIA,SCRANTON,5950,5691,5251,6642,5963,6348,5417
2176 DATARHODE ISLAND,PROVIDENCE,5184,5531,5172,6035,5497,5867,5029
2178 DATASOUTH CAROLINA,CHARLESTON,1393,1941,1837,2702,2340,1972,2195
2180 DATASOUTH CAROLINA,GREENVILLE,2730,3199,2920,3851,3392,3122,3166
2182 DATASOUTH DAKOTA,RAPIDCITY,6477,7045,6111,6622,6923,7626,6050
2184 DATASOUTH DAKOTA,SIOUXFALLS,7088,7598,6685,7484,7822,8393,6799
2186 DATATENNESSEE,CHATTANOOGA,2898,3694,3313,4113,3729,3349,3483
2188 DATATENNESSEE,KNOXVILLE,2833,3418,3340,4148,3822,3520,3467
2190 DATATENNESSEE,MEMPHIS,2500,2878,2526,3442,3355,3205,3013
2192 DATAUTAH,SALTLAKECITY,5402,5495,5392,5370,3982,5526,4722
2194 DATAVERMONT,BURLINGTON,7276,7306,6945,7726,7257,7623,6615
2196 DATAVIRGINIA,NORFOLK,2674,3210,2827,3817,3478,3432,3358
2198 DATAVIRGINIA,RICHMOND,3265,3944,3232,4389,4033,3861,3532
2200 DATAWASHINGTON,SEATTLE,4369,4537,4200,3699,3650,4414,3995
2202 DATAWASHINGTON,SPOKANE,6171,6613,6104,5978,5826,7368,5650
2204 DATAWEST VIRGINIA,CHRLSTON,3807,4813,3832,5487,4896,4534,4526
2206 DATAWISCONSIN,GREENBAY,7324,7794,7079,8319,7616,8227,7040
2208 DATAWISCONSIN,MADISON,6934,7161,6270,7837,7413,7827,6922
2210 DATAWYOMING,CHEYENNE,6561,7106,6274,6540,6100,6851,6179
2990 DATAWYOMING,LANDER,7122,7482,7058,6961,6445,8528,6891
2995 DATAEND,END,0,0,0,0,0,0,0
3000 END
```

©

DYNACOMP

Quality software for:*

ATARI	TRS-80 (Level II)**
PET	NORTH STAR
APPLE II Plus	CP/M Disks/Diskettes

CARD GAMES

BRIDGE 2.0 (Available for all computers) — **Price: $17.95 Cassette/$21.95 Diskette**
An all-inclusive version of this most popular of card games. This program both BIDS and PLAYS either contract or duplicate bridge. Depending on the contract, your computer opponents will either play the offense OR defense. If you bid too high, the computer will double your contract! BRIDGE 2.0 provides challenging entertainment for advanced players and is an excellent learning tool for the bridge novice. See the software review in 80 Software Critique.

HEARTS 1.5 (Available for all computers) — **Price: $15.95 Cassette/$19.95 Diskette**
An exciting and entertaining computer version of this popular card game. Hearts is a trick-oriented game in which the purpose is not to take any hearts or the queen of spades. Play against two computer opponents who are armed with hard-to-beat playing strategies. HEARTS 1.5 is an ideal game for introducing the uninitiated (your spouse) to computers. See the software review in 80 Software Critique.

STUD POKER (Atari only) — **Price: $11.95 Cassette/$15.95 Diskette**
This is the classic gambler's card game. The computer deals the cards one at a time and you (and the computer) bet on what you see. The computer does not cheat and *usually* bets the odds. However, it sometimes bluffs! Also included is a five card draw poker betting practice program. This package will run on a 16K ATARI. Color, graphics, sound.

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

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

THOUGHT PROVOKERS

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

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

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

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

NOMINOES JIGSAW (Atari, Apple and TRS-80 only) — **Price: $16.95 Cassette/$20.95 Diskette**
A jigsaw puzzle on your computer! Complete the puzzle by selecting your pieces from a table consisting of 60 different shapes. NOMINOES JIGSAW is a virtuoso programming effort. The graphics are superlative and the puzzle will challenge you with its three levels of difficulty. Scoring is based upon the number of guesses taken and by the difficulty of the board set-up.

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

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

CHOMP-OTHELLO (Atari only) — **Price: $11.95 Cassette/$15.95 Diskette**
CHOMP-OTHELLO? It's really two challenging games in one. CHOMP is similar in concept to NIM; you must bite off part of a cookie, but avoid taking the poisoned portion. OTHELLO is the popular board game set to fully utilize the Atari's graphics capability. It is also very hard to beat! This package will run on a 16K system.

DYNACOMP OFFERS THE FOLLOWING

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

AND MORE...

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

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

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

MOVING MAZE (Apple only) — **Price: $10.95 Cassette/$14.95 Diskette**
MOVING MAZE employs the games paddles to direct a puck from one side of a maze to the other. However, the maze is dynamically (and randomly) built and is continually being modified. The objective is to cross the maze without touching (or being hit by) a wall. Scoring is by an elapsed time indicator, and three levels of play are provided.

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

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

GIANT SLALOM (Atari only) — **Price: $14.95 Cassette/$18.95 Diskette**
This real-time action game is guaranteed addictive! Use the joystick to control your path through slalom courses consisting of both open and closed gates. Choose from different levels of difficulty, race against other players or simply take practice runs against the clock. GIANT SLALOM will run on 16K systems.

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

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

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

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

ADVENTURE

CRANSTON MANOR ADVENTURE (North Star and CP/M only) — **Price: $21.95 Diskette**
At last! A comprehensive Adventure game for North Star. CRANSTON MANOR ADVENTURE takes you into mysterious CRANSTON MANOR where you attempt to gather fabulous treasures. Lurking in the manor are wild animals and robots who will not give up the treasures without a fight. The number of rooms is greater and the associated descriptions are much more elaborate than the current popular series of Adventure programs, making this game the top in its class. Play can be stopped at any time and the status stored on diskette.

ABOUT DYNACOMP

DYNACOMP is a leading distributor of small system software with sales spanning the world (currently in excess of 40 countries). During the past two years we have greatly enlarged the DYNACOMP product line, but have maintained and improved our high level of quality and customer support. The achievement in quality is apparent from our many repeat customers and the software reviews in such publications as COMPUTRONICS, 80 Software Critique and A.N.A.L.O.G. Our customer support is as close as your phone. It is always friendly. The staff is highly trained and always willing to discuss products or give advice.

**ATARI, PET, TRS-80, NORTHSTAR, CP/M and IBM are registered tradenames and/or trademarks.* ***TRS-80 diskettes are not supplied with DOS or BASIC.*

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 (WORDSTAR, 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 disk menu oriented system composed of 10 programs designed to organize and simplify your personal finances. Features include a 300 transaction capacity; fast access; 26 optional user codes; data retrieval by month, code or payee; optional printing of reports; checkbook balancing; bar graph plotting and more. Also provided on the Atari diskette is ATARI DOS 2.

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.

COMPRESS (North Star only) — **Price: $12.95**

COMPRESS is a single-disk utility program which removes all unnecessary spaces and (optionally) REMark statements from North Star BASIC programs. The source file is processed one line at a time, thus permitting very large programs to be compressed using only a small amount of computer memory. File compressions of 20-50% are commonly achieved.

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 1 1/2 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 effort 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 rice. 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 I (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.

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
Office phone (9AM-5PM EST): (716)442-8960

VISA MasterCard

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

Estimating Gas Mileage With An Empty Tank

J. L. Christensen
El Cajon, CA

In days gone by, the estimation of gas mileage was routinely performed by many drivers to check the health of their favorite conveyance. Gasoline was cheap and the procedure was straightforward. One simply filled the gas tank, drove a couple hundred miles, then refilled the tank. The mileage driven divided by the gallons to refill the tank was the gas mileage.

Today, with the cost of gasoline competing with the cost of a nuclear power plant, not all of us can afford to fill the tank so casually. In fact, for long periods we may find ourselves driving consistently off the bottom half of the tank. This practice results in poor mileage estimates since we no longer have a solid benchmark (a full tank) against which to measure fuel consumption.

The BASIC program presented on the following pages helps to restore gas mileage accuracy while allowing us to drive off the botton of the tank. Fuel consumption is estimated from the gas gauge instead of the 'full tank' method. As everyone knows, automobile fuel gauges are notoriously non-linear, so this program is designed to calibrate the gauge by fitting a cubic equation to data supplied by the driver. The program estimates gas mileage over any selected mileage interval which allows the program operator to discriminate between city driving and the mileage obtained on a trip.

How To Take Data

To use the program, the driver must keep a log in the following format:

Miles	Gallons	Start Gauge	End Gauge

Figure 1. The Gasoline Log

Each time the driver visits his friendly service station, he should enter the miles from the odometer and the gallons of gas added to the tank. He must also enter the gas gauge readings before (Start Gauge) and after (End Gauge) adding the gas. For a bit more generality you may want to add other columns for the date, liters (in case you can't buy gallons) and price.

Now fuel gauges are normally provided with a scale of several divisions. The bottom line is marked with an 'E' while the top of the scale is marked with an 'F'. Automotive gas gauges are rarely provided with numeric annotation. However, the driver may supply the necessary quantitative meter scale by assigning an integer value to each division beginning with 0 at the 'E' position. Thus the gas gauge will appear as shown below:

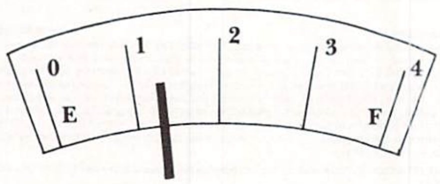


Figure 2. The Gas Gauge Scale

When recording the fuel gauge reading, the driver is expected to interpolate between divisions. This is normally possible to within about ± 10% of the interdivision scale. The fuel gauge reading in the above figure is estimated to be 1.4.

Be sure to read the gauge when the car is resting on level ground. Also some gauges have a very sluggish response so the driver may have to wait a minute or so after restarting the car to get an accurate reading.

If the fuel consumption during a trip is of interest, the driver should enter the odometer reading at the start of the trip, the fuel added, if any, and the gauge readings. If no fuel has been added, enter 0 in the Gallons column and the same fuel gauge reading in both the Start and End Gauge columns. At the conclusion of the trip, again enter the odometer reading, 0 gallons and the same gauge reading in both the Start and End Gauge columns.

Following data assimilation, the program asks the operator for the minimum mileage interval over which the fuel consumption will be computed. Proper account will be taken of the gas remaining in the tank at the start and end of each interval. If the input mileage interval corresponds to that of a trip, the gas consumption on the trip will be computed as well as the consumption over other similar intervals.

Program Operation

The flowchart in Figure 3 will be of assistance in identifying major functions and operations of the program.

The mileage and gas gauge data for the program is provided in a DATA statement beginning at line 201. During program development, DATA statements are preferred over INPUT

```
MILES          GAL         ST. GAUGE          END GAUGE
57146.         8.78         .4                 2.8
57272.         4.3          .3                 1.9
57311.         4.39         .2                 1.9
57349.         1.66         .1                 .6
57364.         4.35         0                  1.7
57394.         8.7          0                  2.3
57473.         4.3          0                  1.75
57506.         4.17         .1                 1.8
57550.         4.16         -.1                1.5

 TANK EQUATION:
 GL = A3*(GA)^3 + A2*(GA)^2 + A1*(GA) + A0
 A3=.609321        A2=-1.73937        A1=3.88762
-------------------------------------------------
GAUGE          GAL
0              0
.2             .712824
.4             1.31574
.6             1.83801
.8             2.30887
1              2.75757
1.2            3.21335
1.4            3.70547
1.6            4.26317
1.8            4.9157
2              5.69231
2.2            6.62224
2.4            7.73474
2.6            9.05906
2.8            10.6244
3.             12.4601
3.2            14.5954
3.4            17.0595
3.6            19.8816
3.8            23.091
4.             26.717

ST MI:57146.      END MI:57311.      GAL:13.6829     MI/GAL:12.0588
ST MI:57272.      END MI:57473.      GAL:24.4262     MI/GAL:8.22887
ST MI:57311.      END MI:57473.      GAL:19.8128     MI/GAL:8.17652
ST MI:57349.      END MI:57506.      GAL:19.01       MI/GAL:8.25881
ST MI:57364.      END MI:57550.      GAL:21.9268     MI/GAL:8.48278
ST MI:57394.      END MI:57550.      GAL:17.5768     MI/GAL:8.87535

 TOTAL GAL:42.3725            TOTAL MILES:404
AVE MILES/GAL:9.01353
OVER AN AVERAGE INTERVAL OF:171.167        MILES
```

statements since the data is not lost if (when) the program bombs. In either case, the value of M1 at line 153 must be set equal to the number of lines of data.

Following data input, the program asks for the "MINIMUM MILEAGE INTERVAL" over which the fuel consumption estimate is to be computed. This value is called U1 in subsequent calculations. At line 310, the program commences at the first mileage entry and searches through following entries until it finds the next entry and searches through following entries until it finds the next entry that equals or exceeds the first entry by U1. The gas consumption is computed over this interval and the program then returns to the second entry and performs the same task. If you would like the mileage between each visit to the pump, enter 0 for the minimum mileage interval.

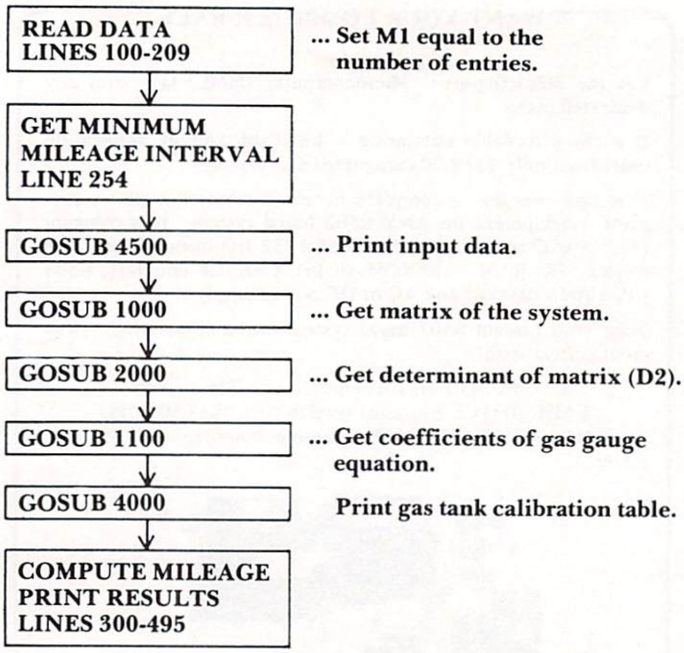


Figure 3. Program Flowchart

The fuel consumption, G1, between two mileage entries, M1 and M2, is computed at line 342. It is the total gallons added between M1 and M2 (including the fuel at M1 but excluding the fuel at M2) plus the fuel in the tank at M1 minus the fuel in the tank at M2. The fuel in the tank at the time of the mileage readings is computed from the gas gauge equations at lines 316 and 340. The program prints intermediate results at line 343 and final results are printed following the end of the primary FOR-NEXT loop at line 450. If the operator has entered a minimum interval which exceeds the available data entries, the program will print "INSUF. DATA OR INTERVAL TOO LONG" and recycle to get a lower value for the minimum interval. The total gas consumed over the entire data set, G4, and the total mileage interval of the data set, Z4, is computed at lines 396 and 397 respectively.

Insofar as practical there has been a deliberate effort to modularize the program through the use of subroutines. Each clearly definable function has been prepared as a subroutine. This technique makes program preparation logical and also simplifies later program modifications. For instance, after the tank is calibrated, the requirement to take End Gauge readings is unnecessary. You may therefore delete the call to these subroutines and insert the gauge coefficients by means of algebraic equivalents.

As illustrated in the flow diagram, the subroutine at 4500 reprints the gas data log. The gauge calibration is performed by subroutines 1000, 2000 and 1100. Subroutine 4000 prints the coefficients of the gas gauge equation with the exception of a constant, A0. The gas tank calibration table is also printed with the assumption that A0 is zero. This assumption may or may not be valid, depending on the characteristics of tank and gas gauge.

An important feature of this program is the method used to calibrate the gas gauge. The gauge is assumed to vary in accordance with the following cubic expression:

$$GL = A3 * (GA)^3 + A2 * (GA)^2 + A1 * (GA) + A0$$

where GL is gallons and GA is the gauge reading.

The program does a least squares curve fit to the data by finding coefficients of the cubic equation which best satisfy the data. The gallons added to the gas tank is assumed to be the difference between the values of the above cubic equation solved at the End Gauge reading and the Start Gauge reading. That is:

$$\blacktriangle GAL = A3 * (GA^3_e - GA^3_s) + A2 * (GA^2_e - GA^2_s) + A1 * (GA_e - GA_s)$$

where $\blacktriangle$GAL is the gallons of gas added to the tank at each data point. GA_s is the Start Gauge reading and GA_e is the End Gauge reading at the same data point. The value of A0 is not computed, nor is it needed since the program uses only the incremental change in gallons and does not need the absolute number of gallons in the tank as a function of the gauge reading. The method of least squares curve fitting will not be described here, but it is available in standard texts such as ref 1. Essentially this technique finds coefficients of the equation such as to minimize the difference between the gallons recorded in the data table and the gallons obtained by means of the above equation and the gauge readings. Since three coefficients must be found, at least three data points will be required. Naturally more data points will result in a better fit. Please note that the equation will be best in those areas where the most data has been taken. The equation should not be trusted in regions of the gas gauge where no data has been provided. If all the data has been taken on the low end of the gauge, the equation is likely to be inaccurate at the upper end of the gauge.

The subroutine at 1000 sets up the elements of a 3 by 3 matrix containing various summations of the data set of gauge readings. This matrix is referred to as the 'matrix of the system' or simply the system matrix. The subroutine at 2000 finds the determinant, D, of a 3 by 3 matrix. The subroutine at 1100 saves the determinant of the system matrix in D2 and sequentially exchanges each column of the system matrix for summations of the gallons data, B(K1). The determinant of the resulting matrix is then computed at line 1155 and the corresponding coefficient, A(J1) is then determined by dividing by D2. Note that before each column exchange is made, the original system matrix is restored at line 1141.

Results

The results of program operation are shown in the example below. The program first reprints the

input data in a form similar to the gasoline log. Next the tank equation and values of the first three coefficients of the equation are printed. If you would like the value of the final coefficient, A0, simply fill the tank to capacity and note the position of the gauge reading. Substitute the full tank gauge reading into the equation and, from your automobile owner manual, set GL equal to the capacity of the gas tank, The value of A0 is then determined to make the equation balance.

Following the gas tank equation, the program prints a table relating the gauge reading to the gallons of gas. Note that this table is useful in obtaining the gallons of gas added to the tank from the differences in gauge readings. It will not provide the absolute gallons in the tank as a function of gauge reading since it is offset by the value of A0. If desired, you may modify the table to include this value by adding A0 to the equation at line 4015.

The program now prints a table of intermediate results consisting of the consumption between successive mileage entries whose difference is not less than the specified MINIMUM MILEAGE INTERVAL. In this example the minimum allowed interval was 150 miles. You will note that the data set contained entries which resulted in an average interval of 171.167 miles.

Finally the program prints the total gallons and the total mileage of the data set followed by the average miles/gallon obtained over the average mileage interval.

Reference

1.) McCracken, D. D. and Dorn, W. S. "Numerical Methods and Fortran Programming", John Wiley and Sons, New York, 1964.

```
100  REM:  PROGRAM   $GASS
150  DIM G[50,3],M[50],C[3,3],B[3],F[3],A[3],D[3]
153  LET M1=9
160    FOR I=1 TO M1
162    READ M[I],G[I,1],G[I,2],G[I,3]
170    NEXT I
199  REM MILES, GALLONS, ST GAUGE, END GAUGE
201  DATA 57146.,8.78,.4,2.8
202  DATA 57272.,4.3,.3,1.9
203  DATA 57311.,4.39,.2,1.9
204  DATA 57349.,1.66,.1,.6
205  DATA 57364.,4.35,0,1.7
206  DATA 57394.,8.7,0,2.3
207  DATA 57473.,4.3,0,1.75
208  DATA 57506.,4.17,.1,1.8
209  DATA 57550.,4.16,-.1,1.5
250  REM*****START PROGRAM*********
254  PRINT "MINIMUM MILEAGE INTERVAL";
255  INPUT U1
257  GOSUB 4500
258  GOSUB 1000
260  GOSUB 2000
261  GOSUB 1100
262  GOSUB 4000
300  LET Z3=0
307  LET N1=0
308  LET Z2=0
309  LET G4=-G[M1,1]
310    FOR I=1 TO M1
312    LET T4=G[I,2]
314    LET S1=M[I]
315    LET G1=0
316    LET G2=A[3]*T4*T4*T4+A[2]*T4*T4+A[1]*T4
317    LET G4=G4+G[I,1]
320      FOR J=(I+1) TO M1
324      LET G1=G1+G[(J-1),1]
330      LET D5=M[J]-M[I]
332      IF (D5 <= U1) THEN 380
336      LET T5=G[(J),2]
```

WARNING!

We have all heard that nothing you can type in from the keyboard can hurt the computer. Not completely true.

There is a particular POKE which, on some CBM computers, risks burning the screen phosphor or blowing out a weak component on one of the computer's boards. To be on the safe side, never POKE 59458,62. This poke appears as line 130 of a program, "Epidemic," on page 46 of the May 1981, **COMPUTE!**. The intended effect — speeding up the output to the screen — might cause problems with some computers.

COMPUTE! mentions this danger on page 91 of the January/February 1980, issue. Unfortunately the early issues of **COMPUTE!** are now out of print.

You cannot, with absolute accuracy, comfort beginners with the assurance, "Go ahead. Nothing you type in could hurt the computer." This POKE can result in a shrinking of the size of the display on the CRT and a consequent intensification of the phosphor fluoresce. Such intensity can burn the screen. Associated with this is the possibility that components inside the computer might be damaged as well. Sometimes a faster PRINT to the screen is desirable, but this can be, for some machines, a damaging way to achieve it.

```
340       LET G3=A[3]*T5*T5*T5+A[2]*T5*T5+A[1]*T5
342       LET G1=G1+G2-G3
343       PRINT " ST MI:";M[I];"END MI:";M[J];" GAL:";G1;" MI/GAL:";D5/G1
346       LET Z2=Z2+D5
347       LET N1=N1+1
348       LET Z3=Z3+D5/G1
350       GOTO 390
380       NEXT J
390     NEXT I
395   LET T1=G[1,2]
396   LET G4=G4+A[3]*T1*T1*T1+A[2]*T1*T1+A[1]*T1-G2
397   LET Z4=M[M1]-M[1]
410   IF (N1#0) GOTO 440
420   PRINT " INSUF. DATA OR INTERVAL TOO LONG"
430   GOTO 254
440   PRINT
450   PRINT "    TOTAL GAL:";G4;"    TOTAL MILES:";Z4
480   PRINT " AVE MILES/GAL:";(Z3/N1)
490   PRINT " OVER AN AVERAGE INTERVAL OF:";(Z2/N1);"MILES"
495   STOP
500   REM*****************************
1000   REM: SUBROUTINE "MATRIX "
1002   REM: THIS SUBROUTINE SETS UP THE ELEMENTS
1003   REM: OF THE GAS TANK MATRIX.
1004   REM
1010     FOR J=1 TO 3
1015     LET B[J]=0
1020       FOR K=1 TO 3
1025       LET C[J,K]=0
1027       NEXT K
1030     NEXT J
1035     FOR I=1 TO M1
1037       FOR J=1 TO 3
1040       LET D[J]=G[I,3]^J-G[I,2]^J
1045       LET B[J]=B[J]+D[J]*G[I,1]
1050       NEXT J
1060       FOR K=1 TO 3
1065         FOR L=1 TO 3
1068         LET C[K,L]=C[K,L]+D[K]*D[L]
1070         NEXT L
1075       NEXT K
1080     NEXT I
1090   RETURN
1100   REM....SUBROUTINE " COEFFICIENTS"
1101   REM.......THIS SUB GETS THE COEFFICIENTS OF THE EQ.
1102   LET D2=D
1140     FOR J1=1 TO 3
1141     GOSUB 1000
1145       FOR K1=1 TO 3
1146       LET C[K1,J1]=B[K1]
1150       NEXT K1
1155     GOSUB 2000
1160     LET A[J1]=D/D2
1162     NEXT J1
1170   RETURN
1190   REM ..............................
2000   REM: SUBROUTINE "DETERMINANT"
2002   REM      THIS SUBROUTINE FINDS THE DETERMINANT OF THE
2003   REM         MATRIX.
```

```
2005  LET F[1]=C[2,2]*C[3,1]-C[2,1]*C[3,2]
2006  LET F[2]=-C[1,2]*C[3,1]+C[1,1]*C[3,2]
2007  LET F[3]=C[1,2]*C[2,1]-C[2,2]*C[1,1]
2008  LET D=0
2009    FOR W=1 TO 3
2010    LET D=D+F[W]*C[W,3]
2011    NEXT W
2015  RETURN
2016  REM..............................
4000  REM
4001  REM:  SUBROUTINE "PRINT TANK CALIBRATION"
4002  PRINT "    TANK EQUATION:"
4003  PRINT "    GL = A3*(GA)^3 + A2*(GA)^2 + A1*(GA) + A0     "
4004  PRINT "    A3=";A[3];"   A2=";A[2];"   A1=";A[1]
4005  PRINT "  ------------------------------------"
4010  PRINT "  GAUGE              GAL"
4011  LET R1=0
4012  LET R2=.2
4013  LET R3=4
4015  LET G=A[3]*R1^3+A[2]*R1^2+A[1]*R1
4016  PRINT TAB(2);R1,G
4017  LET R1=R1+R2
4020  IF (R1>R3) THEN 4050
4030  GOTO 4015
4050  PRINT
4051  RETURN
4052  REM..............................
4500  REM
4501  REM SUBROUTINE "INPUT DATA"
4502  REM .....THIS PRINTS THE INPUT DATA
4503  PRINT "  MILES          GAL        ST. GAUGE        END GAUGE"
4504    FOR I=1 TO M1
4505    PRINT TAB(2);M[I],G[I,1],G[I,2],G[I,3]
4506    NEXT I
4507  PRINT
4510  RETURN
4511  REM..............................
5000  END
```

©

Intermediate BASIC Tutorial:

How To Make Conversation With Your Computer

John Victor
Greenwich, CT

Computer programs that appear to have conversations with humans have fascinated computerists and non-computerists alike. Most of these programs focus on key words to determine the computer's reply.

For example, there is a popular computer program that simulates a psychiatrist. When the "patient" types in an answer to a question such as "what is the matter with you?", the program analyzes the typed answer for words such as father, mother, sex, etc. If key words are found, the program then produces a canned response such as "tell me more about your mother," or "does talking about sex bother you?"

Simple conversational programs can be written using the string manipulation features of Atari, Apple, PET or OSI BASIC. The techniques allow you to look at any string of characters for certain words that may be present in the string.

First, let's look at some of the fundamentals of string logic. We can write a BASIC statement to compare two strings of characters:

IF A$="YES" THEN GOTO 100

The computer will look at the string stored as A$ and, if it matches "YES" it will go to line 100. However, if there is any variation at all, the computer will consider the two strings as not matching. For example, if A$ contains "YES I DO," or even "YES " or " YES" (note the extra space) the computer will consider this as not matching "YES".

Programmers often ask program users to type in YES or NO responses, but they do not really care if the program user can spell. The only thing that concerns them is the first letter, the Y or the N, that indicates the user's intent. In Microsoft BASIC the programmer can get around the problem of the user's input by using the MID$ function.

INPUT A$
IF MID$(A$,1,1)="Y" THEN GOTO 100

The first number indicates what character the computer is going to start with. Here the computer is to start with the first character. The second number indicates how many characters over the computer is going to look at. Here the computer is to look at just one character. In the above statement, no matter what the user types in, if the first character is a Y, the program will go to line 100.

Atari BASIC works a little differently. MID$ is not used. Instead, the same operation is done like this:

INPUT A$
IF A$(1,1)="Y" THEN GOTO 100

The first number here represents the first character to be looked at, but the second number does not indicate the number of characters. Instead, it is the

... some of the fundamentals of string logic.

position of the last character to be looked at. Since the first and the last character are both 1, the computer will only look at one character.

The MID$ function can be used to find strings inside a larger string. For example, we can store a list of three-letter words in a string variable (rather than using DATA statements):

A$="DOG CAT RAT SAT GET KIN FIN SUN RUN"

MID$(A$,9,3) will give us the three-letter word starting with the 9th character, which is the word RAT. (Spaces count as characters).

We can now set up a FOR-NEXT loop that will print all of the three-letter words in the string:

```
10  DIM A$(40)
20 A$ = "DOG CAT RAT SAT GET KIN
    FUN SUN RUN"
30  FOR W = 1 TO 33 STEP 4
40  PRINT  MID$ (A$, W, 3)
41  REM                ATARI VERSION
    -- 40 PRINT A$(W, W+2)
50  NEXT W
```

When the loop starts, W equals one and the equivalent of MID$(A$,1,3) is printed. Next, W is made equal to five and the equivalent of MID$(A$,5,3), the word CAT, is printed.

We can change the above program to work for any length string stored in A$ by using the LEN function. LEN(A$) gives us the number of characters in A$. The following will count to the end of A$, no matter what its length:

30 FOR W = 1 TO LEN(A$)-2 STEP 2

The minus 2 allows for the last two letters in the three-letter word.

In the following example, we will take the process one step further. We will ask the program

user to type in a string of words, and we will look for one word from that string.

```
10  DIM A$(40)
20  PRINT : PRINT "TYPE IN A SENT
     ENCE. "
25  PRINT : PRINT
26  INPUT A$
27  PRINT
30  FOR W = 1 TO  LEN (A$) - 2
40  IF  MID$ (A$, W, 3) = "THE" THEN
      PRINT "YOU USED THE WORD 'T
    HE'": END
41  REM            ATARI VERSION US
     ES A$(W, W+2)="THE"
50  NEXT W
60  PRINT "YOU DID NOT USE THE WO
     RD 'THE'. "
```

In the above program, the computer looks at every three-letter group starting with the first character, and moving over 1 on each count of W. If THE is found, the program will print the appropriate message and stop the program. If not, the loop continues. Now for our final example. The next two programs illustrate a conversational technique whereby the computer user is asked a question, and the computer analyzes the answer for certain key words.

```
1   REM  APPLE VERSION
2   DIM A$(40)
5   DATA  BAD, LOUSY, AWFUL, TERRIBLE
     , NOT GOOD, NOT TOO GOOD, NOT V
     ERY GOOD, NOT WELL
20  TEXT : HOME : REM  CLEAR SCRE
     EN
25  PRINT : PRINT : PRINT
26  PRINT "HOW ARE YOU TODAY?"
27  INPUT A$
30  FOR V = 1 TO 8
40  READ B$
50  FOR C = 1 TO  LEN (A$) -  LEN
     (B$) + 1
60  IF B$ =  MID$ (A$, C, LEN (B$)
     ) THEN 100
70  NEXT C
80  NEXT V
90  PRINT : PRINT "I'M GLAD. ": END
100  PRINT : PRINT "THAT'S TOO BA
     D. . . . ": END
```

```
1 REM ATARI VERSION
2 DIM A$(40),B$(20)
5 DATA BAD,LOUSY,AWFUL,TERRIBLE,NOT TOO
GOOD,NOT GOOD,NOT VERY GOOD,NOT WELL
20 GRAPHICS 0
25 PRINT :PRINT :PRINT
26 PRINT "HOW ARE YOU TODAY?"
27 INPUT A$
30 FOR V=1 TO 8
40 READ B$
50 FOR C=1 TO LEN(A$)-LEN(B$)+1
55 IF LEN(A$)-LEN(B$)<0 THEN 80
60 IF B$=A$(C,C+LEN(B$)-1) THEN 100
70 NEXT C
80 NEXT V
90 PRINT :PRINT "I'M GLAD.":END
100 PRINT :PRINT "THAT'S TOO BAD....":EN
D
```

Lines 30 to 80 establish a nested loop. On each turn of the outer loop, a key word or phrase is read into B$ from a DATA line. The inner loop then tries to find the key word in the string typed into A$ by the program user.

Line 50 sets the number of turns of the inner loop equal to the length of the string typed in by

There are certain problems inherent in these types of programs.

the user, minus the length of the key word. This will allow the computer to try every group of letters in the string that could possibly match the key word.

In line 60 LEN(B$) replaces a constant in the MID$ statement because different key words of different lengths are going to be stored in B$. If the key word is found in A$, line 60 sends the program to line 100. If not, the next key word is read into B$ from DATA and the outer loop turns one more time. If no key words are found, the program ends in line 90.

There are certain problems inherent in this type of program. The first is that all possible key words must be accounted for or the program messages will not be appropriate. In our example, if the program user typed in I FEEL BLAH, the program would not recognize BLAH as a key word. Another possibility is that the program might misinterpret another word as a key word. For example, I FEEL AWFULLY GOOD is interpreted in the negative since the word AWFUL is imbedded in the string. However, even with its limitations, this sort of programming is both interesting to write and fun to use. ©

PETERM™

Terminal Program

Intelligent Communication

Uploads and Downloads

In Machine Language

Use your PET as an intelligent terminal. Access timeshare systems and networks such as the Source or Telenet. This is an interrupt driven system with buffered input/output. Unlike dumb terminals, the PET can do other tasks instead of having to wait for incoming data. This flexibility allows many advanced features.

These are complete assembled hardware and software packages. All include line editing/resend, auto-repeat, shift lock, output to IEEE printer and much more. Delivered on cassette or disk. Also, available for Commodore 8010 modem or Livermore Star 488 modem.

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

FORTH

A Programming Revolution

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

fig-FORTH, with editor, assembler $50.00
with floating point and strings $60.00
with floating point, strings, macro-assembler .. $70.00
cassette and disk interface available.

DEALERS INQUIRE

F S S

software for small computers

1903 Rio Grande
Austin, Texas
78705

1-512-477-2207

P.O. Box 8403
Austin, Texas
78712

Source: TCE995

THE STAR MODEM

From Livermore Data Systems

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

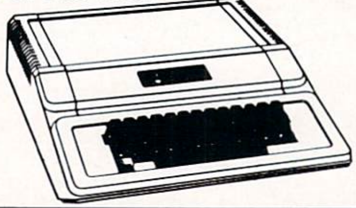
STAR Modem is the price performance leader with a full 2 YEAR FACTORY WARRANTY.

MIPLOT Intelligent Plotter
by Watanabe Instruments (Digiplot)

SPECIAL $1195

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

We carry Apple II+ from Bell & Howell



REVERSAL (Spracklen) 32K Apple	28.00
Super FORTH 48K Apple	40.00
Energy Miser for PET, Apple, or Zenith	24.50
Data Manager (Lutus) 24K Apple	40.00
Histo-Graph (Boyd) 48K Apple	24.50
Data-Graph (Boyd) 48K Apple	40.00
Apple II User's Guide (Osborne)	12.00
Introduction to Pascal (Sybex)	10.30
Pascal Handbook (Sybex)	12.00
Graphics Cookbook for Apple	9.90
Musical Applications of Micros (Chamberlin)	20.00
Basic FORTRAN (Coan)	7.25

SPECIALS

EPSON MX-80 Printer	
EPSON MX-70 Printer	
STARWRITER Daisy Wheel Printer	$1445
Centronics 737 Printer	725
NEC Spinwriter	2500
Diablo 630 Daisy Wheel	2150
Leedex Video 100 12" Monitor	135
ZENITH DATA SYSTEMS	
Z19 Video Terminal (factory assem.)	729
Z89 with 48K (factory assem.)	2150
SYM-1	209
SYM BAS-1 BASIC or RAE-1/2 Assembler	85
KTM-2/80 Synertek Video Board	349
Seawell Motherboard - 4K RAM	195
Seawell 16K Static RAM - KIM, SYM, AIM	320

PROGRAM YOUR OWN EPROMS

Branding Iron for PET/CBM $79

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

WORDCRAFT 80 Word Processor-CBM $300
Extremely comprehensive word processing package for 8032. Allows you to view full 117 character line on screen by using "window" approach. Supports all major printers.

Dow Jones Portfolio Management System $120
Online access to quotes and data for 6000 stocks from Barrons, WSJ, and Dow Jones News Service. Maintains portfolio accounting including tax records.

Legal Time Accounting $500
Allows automatic processing of matters by client, lawyer, and activity. Provides reports on client data, aging analysis, activity revenue, and productivity. Interfaces with Wordcraft 80.

EBS Business System for CBM $600
Provides extremely comprehensive integrated inventory and accounts receivable package including invoices, packing slips, mail labels, statements, deposit slips, and 17 reports.

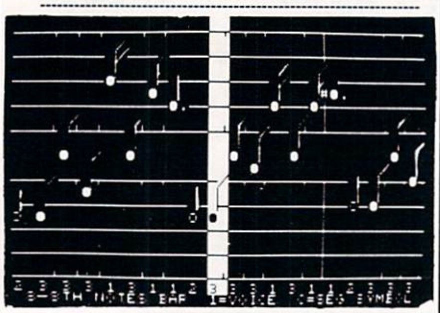
FORTH for PET/CBM or Apple $65
A full-featured FORTH with extensions conforming to Forth Interest Group standards. Includes assembler, string processing capabilities, disk virtual memory multiple dimensioned arrays, floating point and integer processing.

6502	7.45	10/6.95	50/6.55	100/6.15
6502A	8.40	10/7.95	50/7.35	100/6.90
6520 PIA	5.15	10/4.90	50/4.45	100/4.15
6522 VIA	6.45	10/6.10	50/5.75	100/5.45
6532	7.90	10/7.40	50/7.00	100/6.60
2114-L200 ns RAM		3.75	25/3.50	100/3.25
2114-L300 ns RAM		3.15	25/2.90	100/2.65
2716 EPROM (5 volt)		9.90	5/9.45	10/8.90
TMS 2532 EPROM				20.00
4116-200 ns RAM				**8 for 24.00**
S-100 Wire Wrap				2.65
Zero Insertion Force 24 pin Socket				$2.00

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

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

All other lengths available. Write for price list.



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

KL-4M Music Board with VMM Program $59.90

commodore

CBM-PET SPECIALS

FREE Up to $375 free merchandise with purchase of one of following CBM-PET items:

		FREE
8032 32K - 80 Column CRT	$1495	295
8050 Dual Disk Drive - 950,000 bytes	1795	375
4016 Full Size Graphics Keyboard	995	170
4032 Full Size Graphics Keyboard	1295	260
8096 96K - 80 Column CRT		
2040/4040 Dual Disk Drive - 340K	1295	260
4022 Tractor Feed Printer	795	140
CBM Voice Synthesizer	395	50
C2N External Cassette Deck	95	12
Used CBM/PET Computers		CALL
VIC Personal Computer	300	25

WRITE FOR SYSTEM PRICES

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

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

DISK SPECIALS

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

WE STOCK MAXELL DISKS

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

ATARI 800
All Atari Modules 20% OFF

ATARI EDUCATIONAL PLAN Write for details.

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

ALL BOOK and SOFTWARE PRICES DISCOUNTED

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

252 Bethlehem Pike
Colmar, PA 18915

215-822-7727 **A B Computers**

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

The Practical Aspects Of Assembly Language Programming

Bruce D. Carbrey
Raleigh, NC

Part I: Using Flags

It starts with a vague sense of dissatisfaction with the limitations of BASIC. Then you feel a twinge of jealousy towards that mysterious cult of software Gurus who seem to use black magic to exhort their machines to run devilishly fast and speak in strange tongues to devices like PIAs and UARTS. Before you know it you're TAKING THE PLUNGE (**COMPUTE!**, March, 1981), and after struggling down a river of addressing modes and across a sea of opcodes, you know you've passed the initiation rites and can call yourself an Assembly Language Programmer. But perhaps you still feel like something of a novice programmer when it comes to assembly language. If you know assembly language, but don't feel confident that your machine language routines are the best that they can be, this series of articles may help. Even if you are an "expert" assembly language programmer, you may find a useful technique or two presented. Or, you may know of better techniques, in which case I encourage you to write them up and send them in to **COMPUTE!**, so we can all benefit.

I'm going to cover a number of loosely-related topics in detail, putting emphasis on program efficiency. After all, it is almost axiomatic that if you are programming in assembly language at all, you are doing it either to improve execution speed or to reduce program size, or both. The rest of this article assumes that you have a basic working knowledge of 6502 assembly language. The first installment discusses the deceptively simple topic of flags.

Representing Flags

Flags are familiar to any experienced programmer. A flag is a variable which can have only two possible states: TRUE or FALSE. It can be represented by a single bit in memory, but for ease of manipulation by a program, a whole byte is usually used.

Since flags are so simple in concept, you may be surprised to know that many programmers use flags quite inefficiently. To demonstrate what I mean, first consider the example program in Listing 1. This subroutine, usually called a keyboard driver, reads one character from an ASCII-encoded keyboard. The keyboard is assumed to be connected to a parallel I/O port such as is found in a 6820, 6522, 6530, 6532, or similar device. The seven data lines from the keyboard are tied to bits 0 through 6, and a negative-going strobe is connected to bit 7 of the port. When bit 7 of the port becomes zero, the ASCII code for the key which is depressed can be read on the remaining 7 bits. Notice that the strobe is connected to bit 7 because bit 7 is always zero in the ASCII code anyway, and because we can test it easily using BMI or BPL instructions, since bit 7 is the sign bit in a word.

Now suppose that you discover that your Monitor program will accept only uppercase alphabetic letters for commands, but your keyboard only delivers lower case letters unless you hold down SHIFT. What can you do about this nuisance if you don't have an ALPHA LOCK key? You

Properly used, flags can greatly simplify and improve your programming.

could go to the parts box and build a circuit to modify your keyboard, or you can take the software approach and simply add some code to your driver to "fold" all lower-case alphabetic characters ($61 through $7A in the ASCII table) to their uppercase equivalents, as shown below:

```
FOLD   CMP  #$7B   ;LOWER CASE "Z" + 1
       BCS  FOLD1  ;BRANCH IF NOT LOWER CASE ALPHA
       CMP  #$61   ;LOWER CASE A
       BCC  FOLD1  ;BRANCH IF NOT LOWER CASE ALPHA
       SBC  #$20   ;ELSE FOLD LOWER TO UPPER CASE
                   ALPHA
FOLD1  ...
```

This code can simply be inserted at the end of the keyboard driver, just before the RTS. The trouble is, your driver will now *always* return upper case alphabetic characters. This may be desirable for entering commands to the Monitor, but when you're in the Editor you may want to be able to input lower case. The solution? You need an "Alpha-Lock Enable" flag to tell the driver whether to allow lower case or not. You can start by allocating space for your flag:

```
ALFALK  .BYTE  0  ;ALPHA LOCK FLAG FOR
                   KEYBOARD DRIVER
```

Now how do you use it? The natural choice is to set

Shortcuts to more Commodore capability

4 amazing new devices to make your Commodore friendlier, faster and more productive than ever

If you have a Commodore computer, life at the keyboard just got easier, thanks to four new products from Kansas City Computers, Inc.

THE SILENT SWITCH

If you use a switching device on your ROM based software, you may have experienced —select line transient problems.

Introducing our **Triple Flip** firmware. It's a remarkably simple switch that can be electronically *mechanically* switched. With the external switch running to internal switching, there are no address lines outside the computer. That nips select line transient problems right at the source.

Triple Flip lets you place three individual ROMS in a single ROM socket. A single switch can control up to seven **Triple Flips.** That gives you 21 ROM capability.

Versatile? **Triple Flip** is one-socket ROM selectable for Commodore computers 2716, 2532 and 2732. It also fits Pet 2001B, N and later models.

And like all KCCI products, it's backed by our full guarantee and 90-day warranty.

Don't change chips everytime you switch to another program. And don't risk select line transient problems. Put **Triple Flip** to work for you for just $39.95.

30 NEW COMMANDS

Until now, you've had to punch the keys 24 times or more to put some commands into action. Commands like transferring programs to disk or paper, or appending program modules.

Order our **UtiliRom** and start using your computer's memory instead of your own. **UtiliRom** includes an enhanced DOS universal wedge that lets you make most of your present commands in two keystrokes. Imagine having all this at your command:

- Wild card directory (DOS)
- Rename a file (DOS)
- Wild card file scratch (DOS)
- Copy files on disk (DOS)
- Reset disk system (DOS)
- Initialize 2000 series drives (DOS)
- Check error channel (DOS)
- New a disk (DOS)
- Validate a disk (DOS)
- Scroll down
- System cold start
- One key command to load a program (DOS)
- Send program listing to printer (with* or without* form feed at end)
- Send screen contents to printer (normal mode* or squeezed*)
- Send screen contents to disk file by any name*
- Disk program append*
- Repeat key function*
- Kill to turn off repeat*
- Escape to turn off ROM*
- Convert hex to decimal or
- Convert decimal to hex (with error detection)
- Fast jump to monitor
- Fast shift to upper or lower case
- Fast jump to cold start
- One key command to save a program
- Beep (programmable)*

**Asterisk indicates routines which can be called in basic as subroutines for increased computer power.*

Warning! Extensive testing has proven that even casual users of **UtiliRom** become hooked on its efficiency and speed. Serious programmers and custom software designers appreciate the fact that routines are included for fast disk I/O — up to 3 times faster than the system routines. In addition, **UtiliRom** can help protect software designs from theft.

UtiliRom firmware is available for any socket on any Commodore system (including the Address 9000). When used with **Chipmate** firmware (below), you can use UtiliRom in the same socket simultaneously with one other 2K ROM such as Toolkit™.

Order **UtiliRom** now at just $49.95.

Order **Chipmate firmware with UtiliRom** for same socket simultaneous use with most other 2K ROMs. **Chipmate with UtiliRom** is only $61.90. **Chipmate alone** is $14.95.

COMPUTERIZED SCRATCH PAD

Dream up anything you want and put it on your screen. Punch a couple of keys and put it on disk in less than two seconds. Retrieve it anytime, almost as fast as you can visualize it. Hit two more keys and it's generated on your printer in page-perfect format.

You're not dreaming. You're using **Screen Pro** software, the new disk based system that adds exciting new powers to your computer — and a faster response than ever before.

From compiling grocery lists to editing catalogs, from diagramming football plays to animating them, **Screen Pro** makes text and diagram work incredibly simple.

Screen Pro's entire 2000 character screen is displayed in a fraction of a second. You can easily place this subroutine into any software you write, for quick access later. In fact, an eight line program gives you access to any one of 10,000 pages of information — all within four screens!

Clear, understandable documentation shows you how to tailor **Screen Pro** to your exact needs. And it opens your eyes to uses limited only by your imagination — for only $39.95.

MONEY BACK GUARANTEE

All four products come to you with complete, easy-to-follow documentation. Installation instructions that are a snap. Plus step-by-step examples that walk you through the process.

You get more than documentation with your KCCI order. You get dependability. Dependability backed by our Money Back Guarantee. Plus our full 90-day Service Warranty (that's three times as long as most of our competitors' warranties).

Start investing less time in your computer — and start getting more out of it. Fill out and mail this coupon with your check today.

Or see your local Commodore dealer.

Toolkit™ is a trademark of Palo Alto ICS Inc.
™Commodore Business Machines

KCCI

Kansas City Computers, Inc.
5214 Blue Ridge Boulevard
Kansas City, Missouri 64133 • Phone 816/356-6502

Name __________
Address __________
City __________ State __________ Zip __________
Computer make __________ Model __________
Disk system make __________ Model __________

YES! Please send me these shortcuts for more computer capability. My check is enclosed.

☐ TRIPLE FLIP	**$39.95**	☐ CHIPMATE WITHOUT UTILIROM	**$14.95**
☐ UTILIROM	**$49.95**	☐ SCREEN PRO	**$39.95**
☐ CHIPMATE WITH UTILIROM	**$61.90**	☐ TOTAL ORDER	

(Add $1.50 shipping & insurance for U.S. orders. Add $3.10 shipping for foreign orders — payment must be in U.S. funds. In Missouri, add 4% sales tax.)

the flag to 1 if it's true and 0 if it's false. The complete driver routine using this method is shown in Listing 2.

This routine is satisfactory (because it works!), but it can be substantially improved. Notice that you had to temporarily save the returned character on the stack while you tested the Alpha-lock flag.

Next time I'll show a substantial improvement and more ways to improve efficiency.

See the introductory issue of
Home and Educational Computing!
in this issue of **COMPUTE!**

LISTING 1: SIMPLE KEYBOARD DRIVER ROUTINE

```
                    ;
                    ;       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: NO ARGUMENTS.
                    ;       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
178F 60                     RTS
                    ;
0000                        .END
NO ERROR LINES
```

LISTING 2: KEYBOARD DRIVER WITH ALPHA LOCK FLAG
USING 0 = FLASE AND NON-0 = 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 IS NON-0, THEN ALL LOWERCASE LETTERS WILL
                    ;       BE RETURNED AS THE EQUIVALENT UPPERCASE ALPHA.
                    ;       ON RETURN: REGISTER A = ASCII CODE FOR KEY PRESSED;
                    ;       X AND Y PRESERVED.
                    ;
1700                PAD     =       $1700   ;KIM PORT A DATA REGISTER ON 6530
1701                PADD    =       $1701   ;KIM PORT A DATA DIRECTION REGISTER
                    ;
0000                        *=      $1780   ;PROGRAM ORIGIN
                    ;
1780 A900           INCH    LDA     #$00
1782 8D0117                 STA     PADD    ;SET PORT DIRECTION = INPUTS
1785 AD0017         INCH1   LDA     PAD     ;TEST PORT
1788 30FB                   BMI     INCH1   ;WAIT FOR STROBE PULSE
178A 2C0017         INCH2   BIT     PAD
178D 10FB                   BPL     INCH2   ;WAIT FOR END OF STROBE
```

©

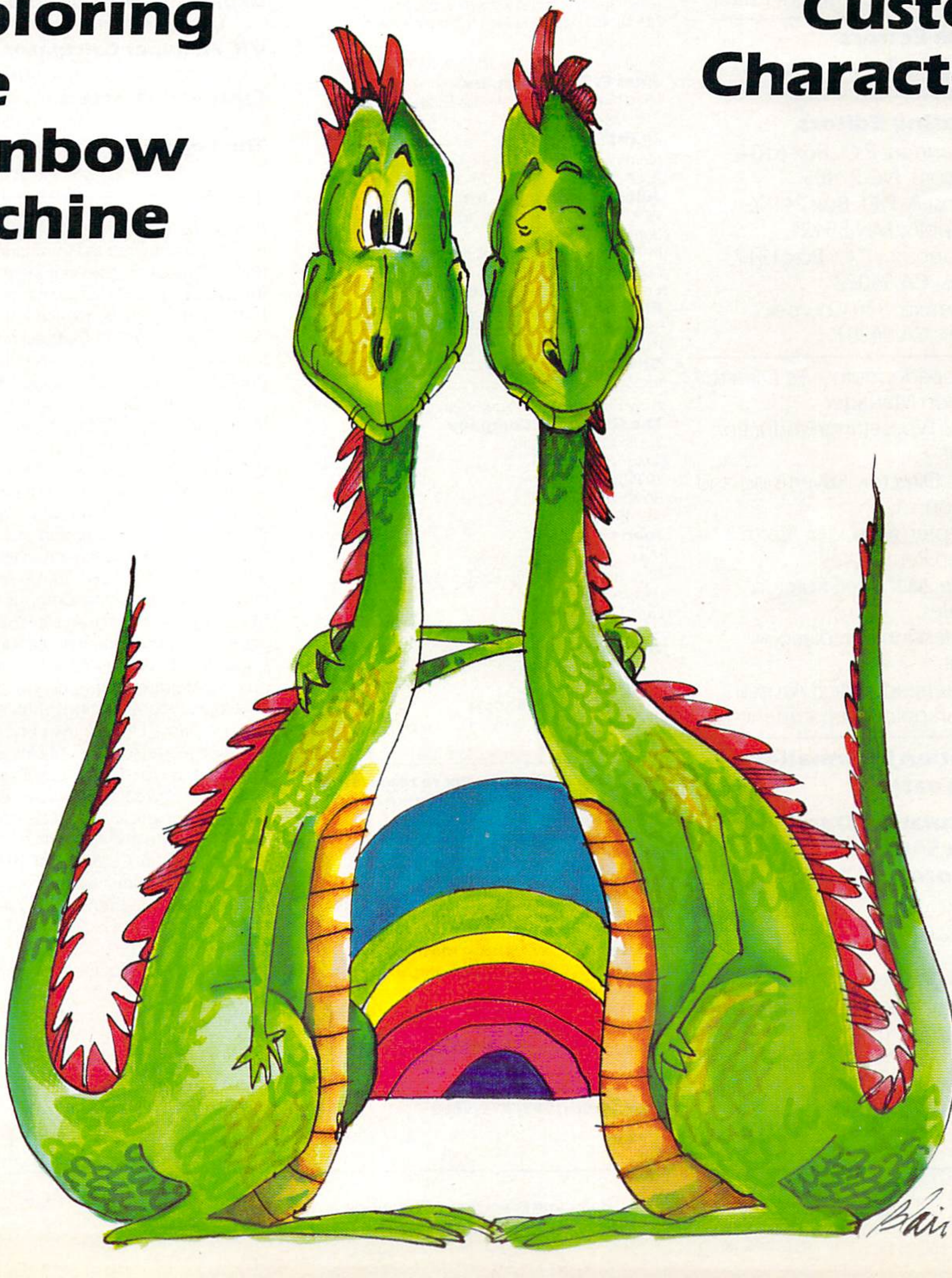
Special Introductory Issue Price 50¢

Home and Educational COMPUTING!™

The Resource Magazine For The **VIC-20**™ Computer

Exploring The Rainbow Machine

Custom Characters



Subscription Information (6 Issue Year):

HEC Circulation Dept.
P.O. Box 5406
Greensboro, NC 27403 USA

US: $10.00
Canada: $12.00
Europe, Air: $18.00
Elsewhere, Air: $24.00

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 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 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 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:
Home and Educational Computing!
625 Fulton Street
Greensboro, NC 27403 USA

Mailing address:
Home and Educational Computing!
Post Office Box 5406
Greensboro, NC 27403 USA

Telephone: (919) 275-9809

Table of Contents

The Editor's Notes

Robert Lock

Welcome to the introductory issue of **Home and Educational Computing!** In these pages you'll find a hint of what's to come in the months ahead. Our concept of a magazine is quite simple: we should be a source of useful, reliable information to users at all levels, from absolute beginner to most advanced. In each issue you'll find information valuable in learning to apply your computer in an ever greater variety of ways.

You're important to this magazine as a reader and contributor. We welcome your comments, suggestions, programs and articles. And please let us know what you think of the magazine.

On With The Notes

If you're already a VIC-20 owner, you may be experiencing some color problems. The units shipped, at least in the US, have experienced some variable amount of RF interference. Two things to try:

1. Reorient your VIC or modulator (as you would a portable TV antenna) to see if relative position improves your color.
2. Inside your VIC is a blue, variable resistor. This can be adjusted to vary color intensity. We strongly discourage this method for the novice, and recommend you ask your dealer for advice on "fine tuning" your color.

Our rumor mill says these problems will be corrected in the version of VIC that's fully FCC approved. These units, scheduled to begin shipment in late July or August, have an improved video circuit and better shielding. A modulator (for connection to your TV) will no longer be standard equipment. It will now cost you an extra $29.95. And that's the state of the color question at the moment. If your "Rainbow" machine isn't making rainbows, we suggest a heart to heart chat with your dealer. In our opinion Commodore should provide those of you with an "early model color problem" with a *cheerful* upgrade.

Where's Our Software?

Although Commodore has been expressing proper concern for getting support to all of you outside software vendors, our feedback from the field has been quite the reverse. Several of the established software houses have vented displeasure with the lack of support and information available from VIC headquarters. We'd like to hear from more software vendors, and to also hear from Commodore on plans to truly initiate some support.

New Products And Reviews

Next issue, we'll start our **New Products** section, bringing you timely information on the latest developments in the world of the VIC-20. You'll also enjoy our **Reviews** section; careful and fair analysis of new products, with the goal of helping you make informed buying decisions.

Exploring The Rainbow Machine

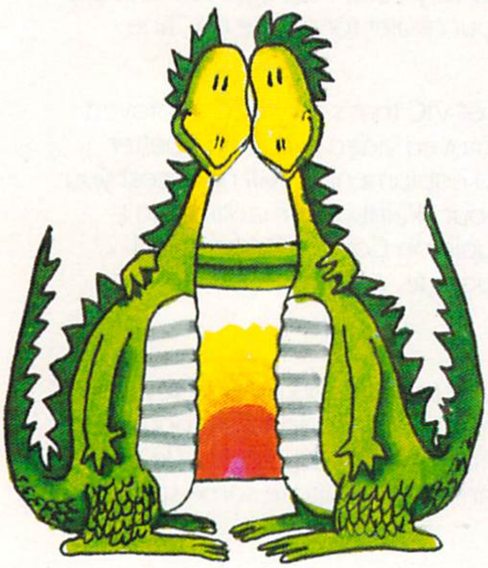
A Beginner's Guide ... To The VIC (Part I)

Ramon Zamora and George Firedrake
Palo Alto, CA

PreRamble

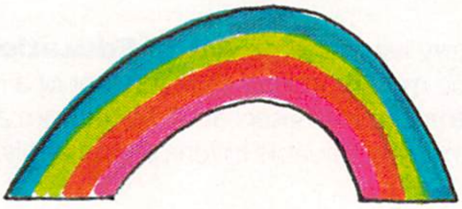
Let's begin with dragons. George and Ramon are two of the many Dragons of Menlo Park. They are friends of Bob Albrecht, known throughout the world, by both kids and adults, as *the* Dragon. (Many people believe that Bob is a real dragon, temporarily disguised as a human, so that we will listen to what he has to say about kids and learning.)

George Firedrake and Bob are old, old friends and claim to have adventured together several thousand years ago. George is often



overheard singing parts of an elaborate dragonsong about their escapades. Together, George and Bob have experienced a dream of an important task to be done. They are to find as many dragonfriends as possible and create dragonstories that help the people of the planet Earth, especially kids, acquire knowledge and mastery of the planet's new technologies.

Ramon appears to be a young dragon. Although he says he is only



a few hundred years old (a span of time that, to a dragon, is hardly any time at all), Ramon is a creator of dragonsongs and dragonstories that reach beyond his years. (Hmmm...perhaps he is older than he looks.)

Recently, Bob and Ramon have been busy establishing a new form of learning environment in the Menlo Park Public Library, called ComputerTown, USA!, and building various dragonlairs about the community.

What does all this dragon stuff have to do with a column about the new VIC computer? Well, dragons are not fond of divulging their larger plans; they like to let events unfold as they may. But, George has let it be known that he feels that the new VIC is an important tool for kids, parents, and teachers to start to use. He claims that the color, sound, and graphics features of the VIC make it, for now, an ideal instrument on which to develop innovative learning materials about how to use personal computers.

Whatever their reasons, the Dragons of Menlo Park are busy writing dragonsongs and dragonstories on the VIC. What you are about to read is part of one of their new adventurous explorations for beginners on the computer they have renamed the *Rainbow Machine.*

— DragonNotes
(March 1981)

The Beginning VIC Display

WARNING!

These columns are written for beginners, and for newcomers to computing and the Commodore VIC personal computer. If you are a skilled computer user, reading this material may cause any one of a number of strange and unpredictable reactions.

WARNING!

What word would you say you use for something that is useful but fun? Why, FUNctional, of course! The new Commodore VIC is a FUNctional computer. The VIC's Color, Sound, and Graphics features are fun to use. The same features play a double role and help you create useful applications for your home, business, or classroom.

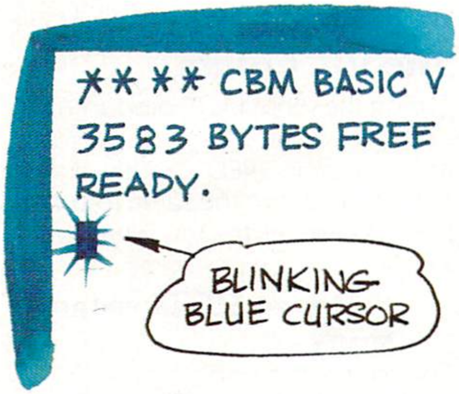
As beginners, you will explore the VIC's Color, Sound, and Graphics so that you teach yourself what the VIC can do. The material in these columns can be used by anyone wanting to learn about the VIC. (Anyone means both kids and adults.) You get to work at your own pace. If you like to explore the VIC s-l-o-w-l-y, then do so. If you discover that you already know some parts being discussed then skip ahead. *You* are in control.

Get ready for your adventures with the VIC. If you wish, find a friend to work with you. Learning about a computer is often more enjoyable (and you sometimes learn more) when done with a friend.

If you have not already done so, unpack your VIC and, following the directions in the VIC user's manual, connect the VIC to your TV set. When the VIC is ready to go ...

TURN ON THE VIC

The VIC screen comes to life, in brilliant color. (If the color is not brilliant, perhaps your VIC is connected to a black-and-white TV set.) Look closely at the screen for a few moments. Check to see that the display on your TV appears like this (we assume your VIC is connected to a color TV):



The messages that appear when you first turn on the VIC should show up as *blue* letters on a *white* screen. Around the Edge of the white central area is a colored border. The border color is *cyan*. Cyan is a lighter blue that has a greenish hue.

You can ignore the first two message lines. (Hmmm...did the VIC misspell *bites*? No, but more on *bytes* later.) The READY message is the VIC's way of saying to you that it is ready to do something. You will tell the VIC what to do by typing on the keyboard. In addition to the

Get ready for your adventures with the VIC.

READY message, the VIC also displays the blinking rectangle, called the *cursor*, to let you know that it is your turn to type. The position of the cursor on the screen indicates where what you type is likely to appear.

The Keys To The VIC

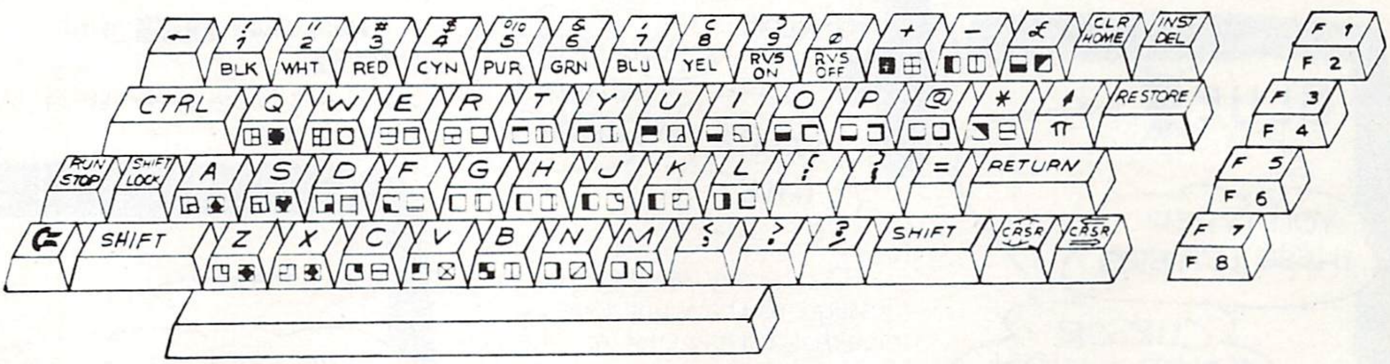
The keyboard is your control center for talking to the VIC. Whatever you type is sent to the VIC and, at the same time, placed on the TV screen so you can verify what you are entering. All the VIC's messages to you are also placed on the screen. Look at the keyboard for a few minutes. The number of keys, all the graphics symbols, and funny words like HOME, CTRL, and CRSR appear everywhere.

The VIC keyboard, however, is a lot like a typewriter in many ways. There are letter keys, number keys, shift keys, and a shift lock key. If you are not a typist, don't worry. Most of what you are going to enter, for a while, only requires a couple of fingers.

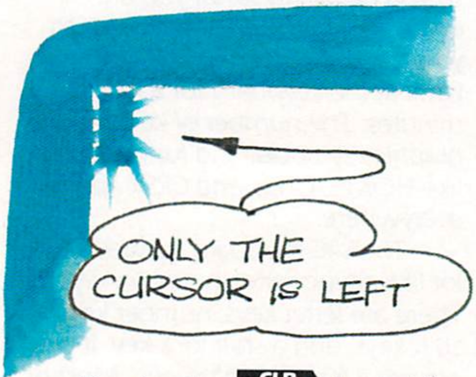
Time for your first experiment. Locate the **SHIFT** key on the left side of the keyboard. (Note: There are two **SHIFT** keys on the VIC, one on the left and one on the right, near the bottom corners. Also, there is a **SHIFT LOCK** key.) You want the **SHIFT** key on the left. When you locate that key, hold it down with a finger on your *left* hand. Now, locate this key:

CLR HOME

With the **SHIFT** key held down with your left hand, press the **CLR HOME** key. What happens? Do the messages on the TV disappear? What does the screen look like?



The VIC Keyboard



Pressing the CLR HOME key while holding down a SHIFT key CLeaRs the screen. Get it? The word CLR on the CLR HOME key comes from the word CLeaR. So to clear the screen:

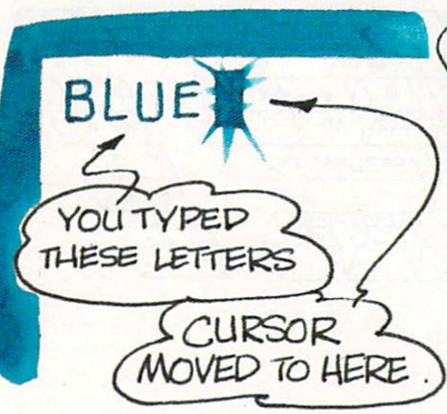
Hold down SHIFT and press CLR HOME

Congratulations! You completed your first communication with the VIC. By pressing the two keys, you told the VIC to do something, namely, clear the screen, and the VIC cheerfully responded. When the screen is clear, the VIC places the blinking cursor in the upper left corner of the display, telling you that it is again your turn to type something.

Type the following (press the keys shown below):

B L U E

Your screen should display:

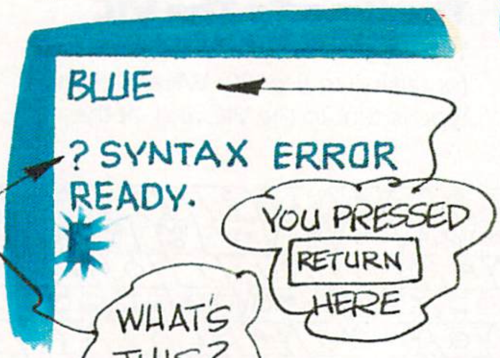


How do you tell the VIC what colors to use?

Each letter you typed appeared on the screen. The cursor moved to the right one position each time you pressed a letter key and sits at the end of the word. (Right now, it doesn't matter what word you typed. So, if you accidentally spelled BLUE as BLEW or BLEU or BLIP, you didn't blow it.)

You now have the word BLUE (or some word) in the color blue on the screen. Press the large RETURN key and observe the TV display.

The message that the VIC sent to the screen can be interpreted as *I Don't Understand What You Typed.* Anytime you enter something that the VIC does not understand the SYNTAX ERROR message will appear.



Don't worry when you see this message. You have not done anything that will harm the VIC. (In fact, short of standing on your keyboard, you cannot type anything into the VIC that will cause serious problems.) The message appeared because the VIC can only understand certain words; it has a limited vocabulary. But, more on that issue later. Back to your explorations with the VIC keys.

The VIC Colors

Typing the word BLUE produced a SYNTAX ERROR. Typing the word for any other color (RED, GREEN, and so forth) would give the same result. How do you tell the VIC what colors to use? Try this experiment:

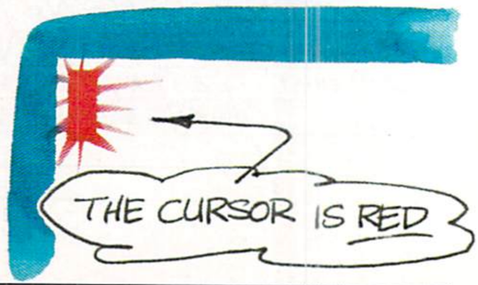
Hold down SHIFT and press CLR HOME

The screen clears and the cursor returns to the top left corner of the display area.

Now, locate the key labeled CTRL. Hold down the CTRL key and press the key with the number 3 on top.

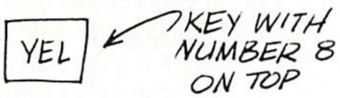
Hold down CTRL and press 3

Look closely at the screen. What color is the cursor?



The cursor is now *red*! Look again at the number 3 key; look at the face of the key. Do you see the letters RED on the front? Aha! All the number keys from 1 through 8 have color codes on their faces. Hold down the **CTRL** key and press another color key. Try this:

Hold down CTRL and press YEL



Is the cursor *yellow*? Your turn: change the cursor to all the colors indicated. (What happens when you press the **WHT** key? Does the cursor disappear?) Remember: you must hold the **CTRL** key down while you press the color keys.

Holding down the **CTRL** key and pressing these keys gives the following colors

BLK	**WHT**	**RED**	**CYN**
black	white	red	cyan
PUR	**GRN**	**BLU**	**YEL**
purple	green	blue	yellow

Try typing a few words in the various colors. Type RED in red; PURPLE in purple; CYAN in cyan. What about typing WHITE in white? The screen's *background* color is already white. If you attempt to type in white, nothing appears on the screen. Try it! Send some invisible messages.

The Rainbow Machine

Time to explore why we call the VIC the Rainbow Machine. Clear the screen. (By now you are probably remembering how this part is done.)

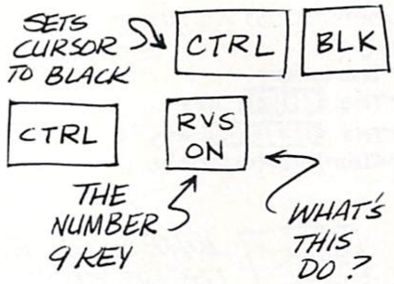
Hold down SHIFT and press CLR HOME

With the Screen clear, and the cursor back in the top corner, press the

Send some invisible messages.

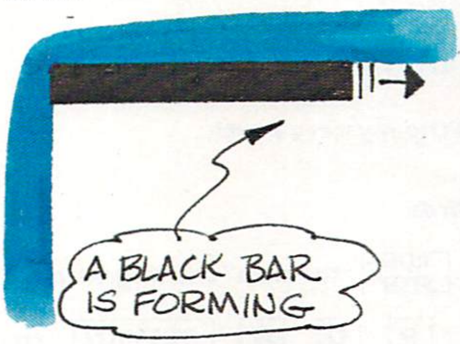
following keys:

Hold down CTRL and press BLK , then hold down CTRL and press RVS-ON

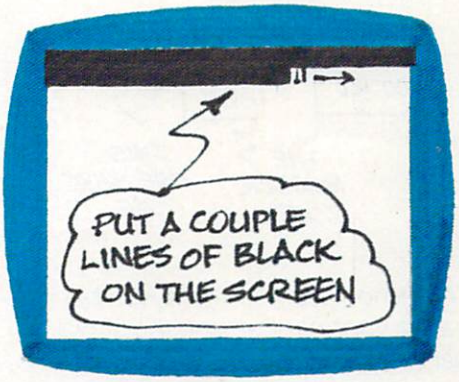


OK! Now, locate the SPACE bar. It is the large bar at the bottom of the keyboard. When pressed, it normally produces a space on the TV display, just like a typewriter produces a space on a piece of paper.

Press the SPACE bar and *hold it down*. What happens? Is the VIC drawing a black bar across the screen?



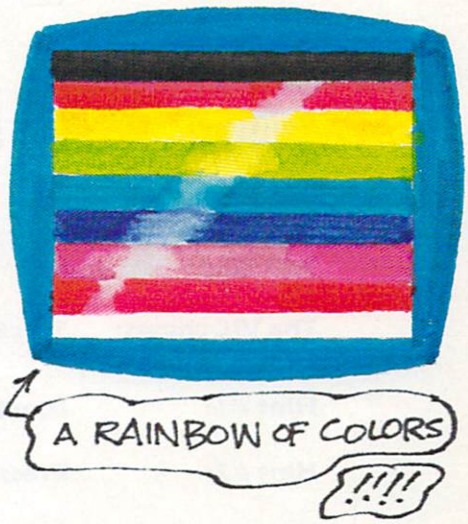
Continue holding the SPACE bar down until the bar reaches the right edge of the screen. What happens then? Why, the VIC keeps on drawing the black bar on the next line. Let the VIC draw until you get two full lines or so, on the screen, then release the SPACE bar.



Now, change to another color. For example, change RED.

Hold down CTRL and press RED

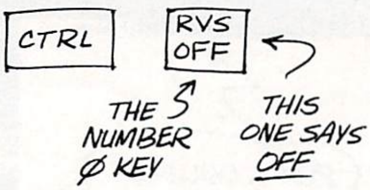
Press the SPACE bar once again. A *red* bar begins to appear. Keep drawing bars and changing colors until the screen is filled with the VIC Rainbow.



You can use the VIC Rainbow to adjust your TV's color settings. Tune the TV to the colors that look best on your screen.

When you finish making rainbows, do this:

Hold down CTRL and press RVS-OFF



This action sets the VIC SPACE bar back to *normal*. Pressing the SPACE bar after you press the RVS-OFF key now puts a space on the screen. You will have other chances to experiment with RVS-ON and RVS-OFF as you progress through this material.

... you will find a mystery exercise that you type into the VIC.

VIC Mystery Experiment #1

Up to this point you have explored or experimented with these VIC features:

***The cursor**
***The SHIFT key**
***The RETURN key**
***Clearing the screen**
***RVS-ON**
***SYNTAX ERRORs**
***Typing words**
***Color keys**
***The SPACE bar**
***The CLR HOME key**
***RVS-OFF**
***Rainbow-Making**

At the end of each column in this series you will find a *mystery* exercise that you type into the VIC, and that leads into material in future columns. Here is Mystery Experiment #1. Clear the VIC's screen (by now, you know how this is done). Type the following into the VIC. If you make a typing mistake, press the RETURN key and retype the line. Experiment and see what you get!

HOLD DOWN SHIFT WHEN ENTERING THESE CHARACTERS

			SHIFT	SHIFT		SHIFT	SHIFT	SHIFT	SHIFT	
You type:	1	0	?	"	SPACE	J	Q	K	"	;	RETURN
The VIC shows:	1	0	?	"	(SPACE)	◡	•	◡	"	;	
You type:	2	0	G	O	T	O	1	0	RETURN		
The VIC shows:	2	0	G	O	T	O	1	0			
You type:	R	U	N	RETURN							
The VIC shows:	**This is the mystery part!**										

Hint #1: **Bluebirds**

Hint #2: **Press RUN STOP to "stop" the migration.**

Hint #3: **Type R U N RETURN to re-start the migration.**

Have fun with your VIC! See you next time.

VIC As Super Calculator

Jim Butterfield
Toronto, Canada

Everyone knows that you can load programs into the VIC and get some pretty clever things to happen. Don't forget that you can also do useful tasks with your VIC without any programs at all.

The technique is called Direct Statements. These are lines that you type without a number at the beginning. For example, if you type PRINT "HELLO" or just ? "HELLO" for short, VIC will obligingly print HELLO. Not too useful, but we're just warming up for the good stuff.

Quick arithmetic is easy to do. To add five and six, type PRINT 5+6. It works just as you expect it to.

VIC uses an * (asterisk) character to signify multiplication, and a / (slash) for division. So PRINT 2*3/4 gives you an answer of 1.5 as you would think. By the way, you'll quickly learn that VIC ignores spaces: PRINT 2 * 3 / 4 gives the same result, and you may feel that it's neater.

When you start mixing multiplication/division with addition and subtraction you'll need to get used to a quick VIC trick: it always performs the multiplication and division first. This means that PRINT 2*3+4*5 will produce 26 (six plus twenty), not 50 as you might think at first. If you really multiply by five you can always force VIC to see things your way by using brackets: PRINT (2*3+4)*5 makes it work. You can use multiple brackets if you wish: PRINT ((2+3)*4)+5 is quite acceptable, and PRINT (2+3)*(4+5) produces the expected answer of five times nine or 45. Remember that you must close the brackets as many times as you open them, or you may get the dreaded ?SYNTAX ERROR notice that tells you that you've done something dumb. If you'd rather be exact about brackets and call them parentheses, that's OK — just remember to use them correctly.

VIC has special functions similar to an advanced scientific calculator.

You'll quickly discover that you can raise a number to a power with the upward arrow: PRINT 2↑3 gives you two cubed, which is eight. Powers are always performed before multiplication, division, addition or subtraction — unless you use brackets. By the way, you'll discover that powers of a number have one very nice feature: the sign of a number is handled correctly in almost all cases. If you have a mathematical bent, you can probably guess what will happen if you raise a number to a fractional or negative power; if you don't, you might like to try it anyway and see what happens. One last thing about powers: they don't work out exactly in all cases: three raised to the fourth power might give you a value just a shade higher or lower than 81.

We've still only just begun. VIC has special functions similar to an advanced scientific calculator. For example, PRINT SQR(5) calculates and prints the square root of five. You have quite a few trigonometric functions: SIN, COS, TAN and the arctangent ATN if you need them, but be careful: they are worked from angles in radians. If you measure your angles in degrees, be sure to convert using a factor of pi/180: for example, the sin of 30 degrees is calculated with: PRINT SIN(30*π/180). For the math whiz, there are logarithms and exponentials using the LOG and EXP functions. If you use these, you'll need to know that they are natural logarithms. If you prefer to use unnatural logarithms (base 10), use a factor of LOG(10) to divide or multiply: the common log of two can be calculated with PRINT LOG(2)/LOG(10).

Memories

Calculators use memories: and VIC the super-calculator gives you lots of memory. You get to name your memory: type A=17 and the value of 17 stored into a memory location called A. Later, you can use this value in other calculations such as PRINT A+9. You can change the memory value at any time with a statement like A=14. You can add or subtract to it with unusual (at first) syntax such as A=A+4 or A=A-11. When you do this kind of thing, remember that the new value is set only after the calculation is complete. So if A equals 5, the expression A=A*3-A would calculate five times three minus five, and then set the result (ten) into memory location A.

You may name memory locations (called "variables" in the VIC) almost anything you like: for example, HENRY=7 will work. You'll be much better off to use a single letter (A, B, C, etc.) or a letter followed by a number (D9, M4, etc.) since VIC can get confused with certain combinations of letters. For example, TANK would get mixed up with the TAN function.

One last thing: memory can get wiped out very easily in the VIC. Certain commands like NEW and

CLR will do it; and typing in any line starting with a number will clear all the variables. Be careful.

Multiple Calculations

If you want to calculate several things, you can do it with a single PRINT command. Just put a semicolon (;) or a comma (,) between the espressions you want to calculate. For example: PRINT 3+5,3*5,3/5 calculates three values and prints them neatly on a single line. If you used the semicolon: PRINT 3+5;3*5;3/5 the values would be printed close together rather than in neat columns.

You can put several commands together on a single line, separated by a colon (:) character. To add a value of one to variable X, and then print X+4, you might code: X=X+1, and then print X+4. The colon will separate the statements so that VIC will understand that they are to be performed separately.

Repeating Calculations

If you can ask VIC to do something once, you ask for the same calculation to be performed many times. All you need to do is put the job you want done between the two following statements: FOR J=1 TO ... : ... (your statement) ... : NEXT J.

Beginners like to see their name printed many times. They should code: FOR J=1 TO 100 : PRINT "JOE" :NEXT J to have the name JOE printed one hundred times. Since each name is printed on a separate line of the screen, there won't be room for all those JOEs. Try changing the PRINT statement by adding a little extra punctuation behind JOE — for example PRINT "JOE", with a comma, or PRINT "JOE"; with a semicolon. You have a lot of control over how things appear on the screen.

It doesn't seem to make much sense to print a fixed calculation such as the square root of ten over and over again. We have much more flexibility than that. As the central statement repeats, the value of the variable (J in the example above) will step through the values we have shown (1 to 100). You may use this variable as a memory value and calculate with it. To print a table of the square roots of numbers from ten to twenty, code: FOR J=10 TO 20 :PRINT J,SQR(J) :NEXT J and the job will be done.

Remember that everything between the FOR and NEXT statements will be repeated with the value of the variable (J in this case) stepping through its range. Don't forget that you can use any variable name you like: FOR M=3 TO 7 is perfectly good so long as you say NEXT M at the point that you want to go back and repeat.

Summary

Direct statements are a good way to learn some of the simple rules of Basic, and they are handy for quick calculations, too.

When you start writing BASIC programs, you'll find it handy to try some of the program lines as direct statements first, to make sure that they work properly. And if your program gives you trouble and stops, you'll find that statements from the program, entered in Direct mode, can give you a hint as to what's going on.

But no matter how advanced you get in your programming adventures, don't forget what a zippy little calculator you have at your fingertips.

Direct statements are a good way to learn ... ... simple rules of BASIC.

Custom Characters For The VIC

David Malmberg
Fremont, CA

One of the many innovations built into the new Commodore VIC is the ability to design our own special characters and have them available to BASIC programs. The possible uses are many. Now you can have different language fonts, such as Japanese, Chinese, or Arabic. You can display electronic schematic symbols. Or, you can write a program that transcribes stenographic characters into English. Greek alphabet characters are now available for tutorial math programs requiring special symbols. You can even design your very own space invader creatures.

This article explains how to make these custom characters. It also presents a utility program to make the job of designing these characters and incorporating them into your BASIC programs quite easy and straightforward. Finally, a sample program is given that demonstrates the custom character features of the VIC by displaying all of the special math symbols of the Greek alphabet.

VIC Character Sets

The character set to be used by the VIC is determined by the value in location 36869. (Note: all locations are given in decimal). This is similar to location 59468 in the PET or CBM machines. The various VIC character sets specified by POKEing this location are as follows:

POKE36869,240 gives upper case and graphics (when shifted)
POKE36869,242 gives lower case and upper case (when shifted)
POKE36869,255 cause the VIC to set aside the first 64 characters of the character set as user defined characters. These special characters will be

... you can have different language fonts, such as Japanese ...

determined by values in the 512 locations beginning at 7168.

VIC Character Representation

To understand how to design your own VIC characters, you must first understand how the VIC represents its characters internally. Just how this is done was demonstrated by Jim Butterfield in his article in the April 1981 issue of **COMPUTE!** Jim pointed out that the two "normal" character sets can be located by using the following equation:

CHR(I) = 32768 + B + 8*I
where B = 0 for the upper/graphics set
and B = 2048 for lower/upper characters
and I = the "screen POKE" value of the character
(e.g., @ is 0, A is 1, B is 2, etc.)

As an example, let's look at how the VIC stores an upper case "A." It has a "screen POKE" value of 1, so by using the above equation we see that it is stored in the eight consecutive bytes beginning at location 32776. If we were to PEEK these locations, we would find the following decimal values — which have the specific bit patterns which define the pixel (i.e., dot) pattern the VIC uses when its prints an "A."
The bit pattern corresponds to the binary representation of the decimal number found in the location. For example, location 32782 contains a

BYTE LOCATION	DECIMAL VALUE	7	6	5	4	3	2	1	0
32776	24				*	*			
32777	36			*			*		
32778	66		*					*	
32779	126		*	*	*	*	*	*	
32780	66		*					*	
32781	66		*					*	
32782	66		*					*	
32783	0								

decimal 66 which is 01000010 in binary — i.e., the pattern at the bottom of the VIC's representation of an "A".

Defining Your Own Characters

Let's see how you would go about designing and incorporating your own custom character into a BASIC program. For example, let's add the following vicious-looking creature to your version of Space Invaders or Dunjonquest.

	7	6	5	4	3	2	1	0	BINARY	DECIMAL
ROW 0	*							*	10000001	129
ROW 1	*			*	*			*	10011001	153
ROW 2		*	*			*	*		01100110	102
ROW 3			*	*	*	*			00111100	60
ROW 4	*	*	*	*	*	*	*	*	11111111	255
ROW 5			*	*	*	*			00111100	60
ROW 6		*					*		01000010	66
ROW 7		*					*		01000010	66

(Columns 7–0 are the PIXEL PATTERN.)

The binary and decimal values corresponding to the creature's pixel pattern are also given. To get the VIC to use this pattern as one of its characters, let's enter and run the following short program:

```
100 X = PEEK(56)-2: POKE 52,X: POKE
    56,X: POKE51,PEEK(55): CLR
110 CS =256*PEEK(52) + PEEK(51)
120 FOR I = CS TO CS+511: POKEI,
    PEEK(I+32768-CS): NEXT
130 FOR I = 0 TO 7: READ J: POKE
    CS+I,J: NEXT
140 DATA 129,153,102,60,255,60,
    66,66
150 POKE 36869,255: PRINT"CLR"
160 FOR I = 1 TO 11: PRINT"@ "0:
NEXT
```

After running the program you should see a row of 11 of your creatures on the top line of the screen.

Let's review this program line-by-line to understand how to use these special characters in other programs. Line 100 PEEK's two pages lower (a page is 256 bytes). Line 100 also changes the pointer to the beginning of the strinng variables (locations 51 and 52) to point to the beginning of these two pages. The CLR resets the user RAM boundaries so that these two pages are protected from the rest of the BASIC program. Line 110 calculates the starting location for the table containing the new character set.

Line 120 transfers the first 65 characters of the standard upper case character set from ROM into the new character set table in the top two pages of user BASIC RAM. This is not strictly required, but it is good practice because it allows you to have access to "normal" characters as well as your specially designed charcters on the same screen.

Line 130 reads the data in line 140 that defines the pixel pattern for the creature and POKEs it into the table space used by the first character of the new character set, i.e., the table space used by the "@" sign in the normal upper case character set.

Line 150 tells the VIC to use the custom character set where the first 64 characters are user defined. Line 160 tries to print a row of @'s but ends up printing your creature because its pixel pattern is in the table where the @ would normally be located.

You could continue to add to this simple program to build a complex game that would use your creature whenever you PRINTed "@" or POKEd the screen with a zero (i.e., the @'s normal screen POKE value). To return to the normal character set, give the direct command: POKE 36869,240 which will cause all your creatures to be transformed back to @'s.

The program has two operation modes: ... review mode ... and ... editing and new-character definition ...

A Utility Program

Listing 1 is a short BASIC program for the VIC which helps with the design, testing, and coding of special characters by essentially automating the process described above. The program has two operation modes: (1) a review mode which allows you to see how the current character set looks — including your custom characters, and (2) an editing and new-character definition mode.

Review Mode

When you first run the program in listing 1 you will initially be in the review mode and the screen will look like this:

```
@ABCDEFG     OPTIONS
HIJKLMNO
PQRSTUVW     N EW CHAR
XYZ[£]       E EDIT CHAR
■!"#$%&'     Q QUIT
()*+,-./
01234567
89:;<=>?
```

The characters shown in the first eight rows and eight columns of the screen are the currently defined custom character set. Note that you start the program with the normal upper case characters. As you redefine the characters, the new characters will be displayed in their appropriate place in this character table. For example, if we had used the utility program to create the creature in the previous example it would be displayed in place of the @ sign whenever we were in the review mode.

The blank in the first column of the fifth row will be red and serve as a "fake" cursor. You will be able to use all normal cursor controls, including HOME and CLR to position this fake cursor on any of the characters displayed in the character set. This fake cursor will also have automatic repeat key and automatic wraparound features.

To define a new special character, move the fake cursor to the position of the character in the normal character set you wish to replace. A good idea is to replace characters that are seldom used, so that you still have access to the more popular characters, i.e. the letters and digits. Once the cursor is positioned, hit "N" on the keyboard to define a new character in place of the one the cursor is on. If you are just reviewing a character that you have previously created and decide it needs more work, then position the cursor on the character and hit "E". Either "E" or "N" will shift the program into the EDIT mode.

Edit Mode

As an example, let's assume that you wanted to add serifs to the character "K". After placing the red cursor over the K you would hit an "E" to enter the Edit mode, and the screen would look like this:

```
■.*...*.        OPTIONS
..*..*..
..*.*...        + ADD DOT
..**....        – ERASE
..*.*...        = UPDATE
..*..*..        B BASIC
..*...*.        R REVIEW
........        Q QUIT
```

The screen shows the pixel pattern for the character K in a large 8-by-8 format. The cursor is "homed," but may be repositioned using the cursor control keys to rest on any of the large pixels. Once the cursor is properly positioned, a pixel may be "turned-on" by hitting a " + " or "turned off" by hitting a "–" sign. After you are satisfied with your handiwork, hit an " = " sign to put that character into the character set table. Then if you wish to see the character in its normal size and format, hit an "R" to go back to the Review mode.

After the design of the special character is complete, you may hit a "B" to have the VIC print the BASIC code needed to add this character to other programs. For example, if we had used this utility program to design the creature used in the previous example, and we hit a "B", the VIC would display the following lines of BASIC code:

```
200 READ X: FOR I = X TO X+7: READ
    Y: POKE X,Y: NEXT
210 DATA 7168, 129, 153, 102, 60,
    66, 66
```

You will recognize that these lines of code are essentially equivalent to lines 130-140 in the previous example. IMPORTANT — these lines of code will only work if lines 100-120 in the previous example or their equivalent have already been executed. Listing 2 gives an example of how the BASIC code generated by this utility program might be used to incorporate a number of special characters (specifically the math symbols in the Greek alphabet) into a BASIC program.

... two challenges to programmers ...

A word of caution — hitting a "B" or an "R" will both generate displays based on what is actually in the character set data table. This may not correspond to the current large-sized pixel pattern. Always be sure this table is correct by updating it via the " = " command prior to using the "B" or "R" commands.

Hitting a "Q" while in either the Edit or Review modes will cause both the memory size and the character set to be reset to their normal states and the program to end.

A Few Suggestions

If you want to use a custom pattern that is larger than just one character, use the utility program to design the pieces of the overall pattern into contiguous characters, as shown in the Review mode display. For example, if you want a 3-across by 2-down pattern, you could use the utility program to design the various parts into the character positions normally occupied by @, A, B, and H, I, J. Then whenever you PRINTed these characters in your BASIC program (in the correct configuration — of course) you would get your desired large custom pattern.

If the reverse character flag is on (i.e., the character you are PRINTing has been preceded by a reversed "R") the VIC will use the standard character set and not the custom character set. You will find this useful when you have already redefined various characters, and you want to use those same original characters on the same screen. You can simply PRINT those characters in reverse. This "trick" is used in the Review mode of the utility program to assure that the options are always printed properly.

Programming Challenges

Here are two challenges to programmers who would like to show they have mastered the VIC's custom character features and who want to write very useful programs that can be used and enjoyed by the growing VIC community: (1) Write a program that will draw a straight line (or as close to one as possible) between any two pixels on the VIC's screen. (2) Write a generalized graph program that can graph equations (one or more simultaneously) in high resolution by defining special characters — on the fly — as the shape of the equations require them.

Table 1

```
100 POKE36879,27:PRINT"Ɔ CHARACTER GENERATOR"
110 PRINT"▓▓  BY DAVID MALMBERG"
120 REM 43064 VIA MORAGA
130 REM FREMONT, CALIFORNIA
140 X=PEEK(56)-2:POKE52,X:POKE56,X:POKE51,PEEK(55):CLR
```

```
150 CS=256*PEEK(52)+PEEK(51)
160 FORI=CSTOCS+511:POKEI,PEEK(I+32768-CS):NEXT
170 S=7680:CL=22
180 CR=0:LN=200:P=12:BG=3:BR=1
190 POKE36879,BG*16+BR
200 DEFFNA(XX)=S+R*CL+C:REM SCREEN POKE LOCATION
210 DEFFNB(XX)=8*R+C:REM SCREEN POKE VALUE FOR CHARACTER
220 GOTO580
230 PRINT"♥":GOSUB810
240 PRINT"⌂";:FORI=0TO7:PRINT"........":NEXT:F=0
250 PRINT"⌂":R=0:C=0
260 Z=FNA(0)
270 IFF=0THENPOKEZ,PEEK(Z)+128:GOTO310
280 IFZ=ZLTHEN300
290 POKEZL,IL:POKEZL+30720,BC:ZL=Z:IL=PEEK(ZL)
300 POKEZ,32:POKEZ+30720,2
310 GETA$:IFA$=""THEN310
320 IFF=0THENPOKEZ,PEEK(Z)-128
330 REM CURSOR CONTROL OPTIONS
340 IFA$="Q"THENPOKE56,PEEK(56)+2:POKE36869,240:PRINT"♥":END
350 IFA$="⇨"ANDC=7THENC=0:GOTO260
360 IFA$="⇨"THENC=C+1:GOTO260
370 IFA$="⇦"ANDC=0THENC=7:GOTO260
380 IFA$="⇦"THENC=C-1:GOTO260
390 IFA$="⇩"ANDR=7THENR=0:GOTO260
400 IFA$="⇩"THENR=R+1:GOTO260
410 IFA$="⇧"ANDR=0THENR=7:GOTO260
420 IFA$="⇧"THENR=R-1:GOTO260
430 IFA$="⌂"THEN250
440 IFF=1THEN540
450 REM DEFINE NEW CHARACTER OPTIONS
460 IFA$="+"THENPOKEZ,81:GOTO260
470 IFA$="-"THENPOKEZ,46:GOTO260
480 IFA$="="THEN680
490 IFA$="♥"THEN240
500 IFA$="R"THEN580
510 IFA$="B"THEN770
520 GOTO260
530 REM REVIEW CHARACTER SET OPTIONS
540 CR=FNB(0)
550 IFA$="N"THENPOKE36869,240:GOTO230
560 IFA$="E"THENPOKE36869,240:F=0:GOTO730
570 GOTO260
580 POKE36869,255:R=4:C=0:ZL=FNA(0):IL=32
590 PRINT"♥@ABCDEFG":PRINT"HIJKLNMO":PRINT"PQRSTUVW":PRINT"XYZ[\]↑←":F=1
600 PRINT" !"+CHR$(34)+"#$%&'":PRINT"()*+,-./":PRINT"01234567":PRINT"89:;<=>?"
610 PRINT"⌂"SPC(12);"®OPTIONS■":PRINT
620 PRINTSPC(10);"®N NEW CHAR■"
630 PRINTSPC(10);"®E EDIT CHAR■"
640 PRINTSPC(10);"®Q QUIT■"
650 BC=PEEK(38400)
660 GOTO260
670 REM UPDATE CHARACTER DATA IN TABLE
680 PRINT"⌂";:X=CS+8*CR:FORR=0TO7:SM=0:FORC=0TO7:D=7-C
690 SM=SM-2↑D*(PEEK(FNA(0))=81):NEXTC
700 POKEX+R,SM:PRINTSPC(8);SM:NEXTR
710 R=0:C=0:GOTO260
720 REM EDIT CHARACTER FROM TABLE
730 X=CS+8*CR:PRINT"♥":FORR=0TO7:Y=PEEK(X+R):FORC=0TO7:Z=FNA(0)
740 Q=46:Y=Y*2:IFY>255THENQ=81:Y=Y-256
750 POKEZ,Q:NEXTC,R:R=0:C=0:GOSUB810:GOTO260
760 REM BASIC STATEMENTS TO DEFINE CHARACTER
770 X=CS+8*CR:PRINT"⌂⇩⇩⇩⇩⇩⇩⇩⇩⇩"
780 PRINTLN;"READ X: FOR I=X TO X+7: READ Y: POKE X,Y: NEXT":LN=LN+10
790 PRINTLN;"DATA";X;:FORI=XTOX+7:PRINT"⇦,";PEEK(I);:NEXTI:PRINT
```

```
 800 GOTO260
 810 PRINT"▣";SPC(13)"▣OPTIONS■":PRINT
 820 PRINTSPC(P);"▣+■ ADD DOT"
 830 PRINTSPC(P);"▣-■ ERASE"
 840 PRINTSPC(P);"▣=■ UPDATE"
 850 PRINTSPC(P);"▣B■ BASIC"
 860 PRINTSPC(P);"▣R■ REVIEW"
 870 PRINTSPC(P);"▣Q■ QUIT"
 880 RETURN
READY.
```

Table 2

```
 100 POKE36879,27:PRINT"□ VIC CHARACTER DEMO"
 110 PRINT"▤▤  BY DAVID MALMBERG"
 120 REM 43064 VIA MORAGA
 130 REM FREMONT, CALIFORNIA
 140 X=PEEK(56)-2:POKE52,X:POKE56,X:POKE51,PEEK(55):CLR
 150 CS=256*PEEK(52)+PEEK(51)
 160 FORI=CSTOCS+511:POKEI,PEEK(I+32768-CS):NEXT
 170 READ X:IFX<0THEN200
 180 FORI=X TO X+7:READJ:POKEI,J:NEXT
 190 GOTO170
 200 PRINT"□@ABCDEFG":PRINT"HIJKLNMO":PRINT"PQRSTUVW":PRINT"XYZ[\]↑←"
 210 PRINT" !"+CHR$(34)+"#$%&'":PRINT"()*+,-./":PRINT"01234567":PRINT"89:;<=>?"
 220 PRINT"▣"SPC(11);"▣OPTIONS■":PRINT
 230 PRINTSPC(11);"▣N NORMAL■"
 240 PRINTSPC(11);"▣L LOWER■"
 250 PRINTSPC(11);"▣G GREEK■"
 260 PRINTSPC(11);"▣Q QUIT■"
 270 PRINT:PRINT:PRINT
 280 GETA$:IFA$=""THEN280
 290 IFA$="N"THENPOKE36869,240
 300 IFA$="L"THENPOKE36869,242
 310 IFA$="G"THENPOKE36869,255
 320 IFA$="Q"THENPOKE36869,240:POKE56,PEEK(56)+2:END
 330 GOTO280
 340 DATA7168,24,24,36,60,102,66,66,0
 350 DATA7176,124,34,34,60,34,34,124,0
 360 DATA7184,126,34,34,32,32,32,112,0
 370 DATA7192,24,24,36,36,102,66,126,0
 380 DATA7200,126,34,32,56,32,34,126,0
 390 DATA7208,126,70,12,24,48,98,126,0
 400 DATA7216,102,36,36,60,36,36,102,0
 410 DATA7224,24,36,66,126,66,36,24,0
 420 DATA7232,28,8,8,8,8,8,28,0
 430 DATA7240,102,36,40,48,40,36,102,0
 440 DATA7248,24,24,60,36,36,102,102,0
 450 DATA7256,66,102,90,66,66,66,66,0
 460 DATA7264,66,98,82,74,70,66,60,0
 470 DATA7272,126,0,36,60,36,0,126,0
 480 DATA7280,24,36,66,66,66,36,24,0
 490 DATA7288,126,36,36,36,36,36,36,0
 500 DATA7296,124,34,34,60,32,32,112,0
 510 DATA7304,126,98,48,24,48,98,126,0
 520 DATA7312,62,42,8,8,8,8,28,0
 530 DATA7320,20,42,8,8,8,8,28,0
 540 DATA7328,8,28,42,42,28,8,8,0
 550 DATA7336,102,66,36,24,36,66,102,0
 560 DATA7344,42,42,42,28,8,8,28,0
 570 DATA7352,0,24,36,66,66,36,102,0
 580 DATA7360,0,0,0,0,0,0,0,0
 590 DATA7368,0,0,0,0,0,0,0,0
 600 DATA7376,0,0,0,0,0,0,0,0
 610 DATA-1
READY.
```

The Confusing Quote

Charles Brannon

As you type in a program, you will eventually come to a place where information is in quotes. This tells the computer to "take this exactly as shown," instead of interpreting it. For example, the instruction PRINT "PRINT" causes the word PRINT to be displayed on the screen. The PRINT in quotes is entirely different from the command PRINT. What, however, do you do when you're typing in a line like:

```
10 INPUT "WHAT IS YOUR
NAME";N$
```

and you make a mistake at the beginning of the line? You just cursor-left to the error, and correct it. Right? Nope. What you get are a bunch of reverse-field vertical lines. These are *control characters*, but that explanation doesn't help you retype that error.

When you typed that first quote you entered the twilight zone of *quote mode*, which is both one of the most frustrating and most useful features of the VIC. The trick is that cursor keys are not only for use in screen editing, but can also be programmed. When the VIC comes to one of those reverse-field vertical lines, it will attempt to actually move the cursor left one space. This can be used to produce animation. When a character is printed on the screen, the cursor moves to the right one space, just like on a typewriter. If, however, you move it back with a programmed cursor-left, you can replace the old character with a new one. Try this line:

```
10 FOR I=1 TO 20:PRINT" ●‖";
:NEXT I
```

Other cursor controls can be programmed as well. The most commonly-used one is the clear-screen character. This is at the start of most programs, and it appears as a reverse-field heart. Actually, all control characters on the VIC are in reverse-field. Cursor-down (Q) can be used to skip down to any line quickly, without a having to print a blank line. Hence the line:

```
10 PRINT:PRINT:PRINT:PRINT
```

can be replaced by

```
10 PRINT "QQQQ";
```

Used in conjunction with the HOME character, cursor down can act like a "vertical TAB statement." At the start of your program, define CD$ (or any string, really) to be equal to HOME and 21 cursor-downs. Now can place the cursor on any line with PRINT LEFT$ (CD$,L); where L is the screen line, from zero to 22. Reverse field on and off are also easy to use; just insert the appropriate characters before and after the text you want highlighted. The color control keys are used similarly, except that while reverse-field is cancelled by a carriage-return, the color command remains in effect until changed. Has your display ever disappeared?

The trick is that cursor keys are not only for use in screen editing, but can also be programmed.

Don't despair, you probably changed the text color to white (CONTROL2), and if the background was white, everything would disappear. Just type CTRL some other color to regain it, or reset with STOP/RESTORE.

Okay, now you're using control characters to do amazing things, but you may be experiencing another problem — you can't make the VIC print them. The problem here is that you are not in quote mode. Here is exactly how quote mode works:

1. If you type an odd number of quotes, you are in *quote-mode* — all cursor controls (except DELete) will show up for better or worse.
2. An even number of quotes, two or four, or none at all, lets you be in the *edit mode*, where you can move the cursor anywhere and type.
3. A special way to get into quote mode is by the INST key. When you insert a gap in text, you are temporarily in quote mode. If you type any control key, it will be printed. This is useful for placing cursor controls inside an already-typed line.

Finally, if you are going crazy trying to figure out what those quotes are doing to your line, just type SHIFTED RETURN to escape to the next line. SHIFTED RETURN does not act like an ENTER key. It just moves to the next line, and cancels reverse field and quote mode. You can then cursor up to the mangled line and fix it.

Remember, one of the VIC's strengths is its ability to manipulate the cursor, colors, and even select upper/lower or uppercase/graphics. Don't neglect this feature. And you can quote me on that.

THE apple GAZETTE

Animating Integer BASIC Low-Resolution Graphics

Leslie M. Grimm
Mt. View, CA

Animated low-resolution graphics can add a lot of pizazz to your Basic program. It takes longer to design the program, and requires the knowledge of a few peeks and pokes, but is not really too difficult to learn, and the results make the effort extremely worthwhile.

There are two basic techniques involved. The first is the design of the animated figures. This is similar to what is done in designing cartoon figures. The second technique involves "flipping pages" on the computer while successively drawing each section of your animated figure to create the illusion of movement. Each technique will be explained here, with a short Integer Basic program using the technique as an example.

Designing The Figure

Before you take pencil and paper in hand it is a good idea to spend some time observing or visualizing the object you wish to animate in the various phases of its motion. For the program example here, the figure to be animated was a girl, who was to be shown walking from left to right across the screen. Observation of people walking was followed by paper and pencil sketches of the various stages of the walking motion, as shown in fig. 1. In this example, the illusion of walking can be created with a succssion of four pictures, A, B, C, and D, as shown.

Once you have a sketch of your figure, you'll need some graph paper. Quadrille-ruled paper is fine, but the special Apple graphics paper available in some computer outlets is more accurate, because the "squares" on the screen are really rectangles, which can throw off the proportions of your finished figure.

Each phase of the movement of the figure should be drawn on its own rectangular section of the graph paper. Each rectangular section should be the same size, and the figure should be centered in exactly the same position in each rectangle. (See fig. 2). In the case of the girl, the rectangle had to be wide enough to accomodate the figure, whether the leg was projected forwards or backward. An extra space was also allowed on either side of the girl, but this is not essential. Each rectangle should be numbered as shown in Fig. 2, with the upper left-hand corner counted as 0,0 (just as the upper left corner of the graphics screen is 0,0).

Now you are ready to develop the subroutines that will draw the figure. You will need one subroutine for each phase of the movement, as a minimum. These subroutines should be assigned

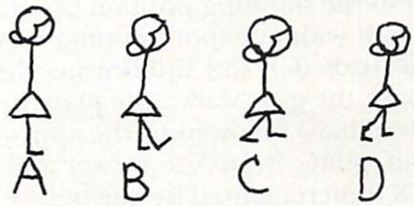


Figure 1

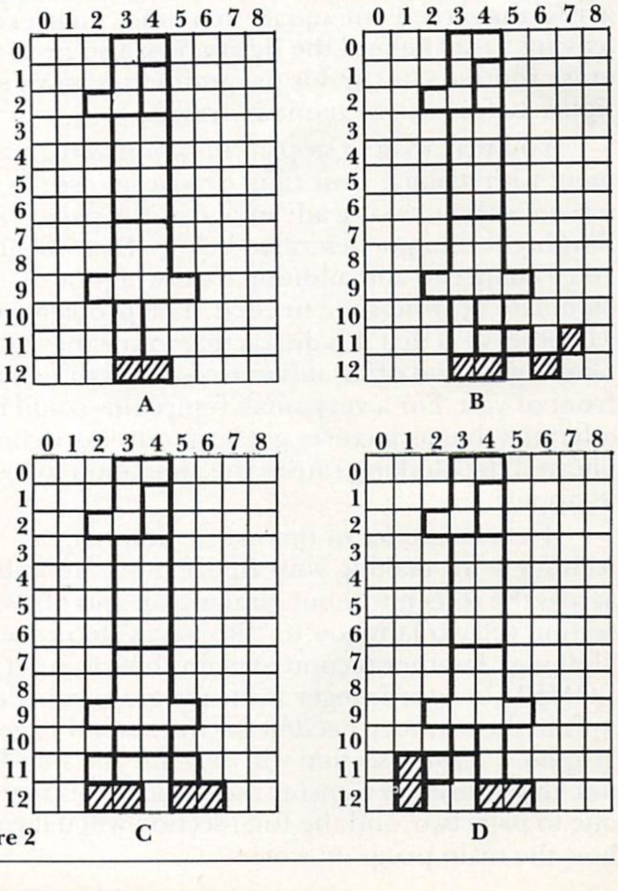


Figure 2

low line numbers, and as many commands as possible should be crammed into each line in order to speed up the drawing time. (The larger your picture is, the more critical this aspect becomes.) Use HLIN and VLIN commands wherever possible.

Since your figure is going to be moving on the screen, your subroutines should not plot specific points (such as PLOT 12,21), but should be written instead in general terms. To do this call the 0,0 square of each rectangle X,Y. Then write your commands in terms of X and Y. For example, the girl's face is drawn with a short vertical line at column 4 in the picture. The *specific* command to draw the face would be VLIN 1,2 at 4. The *general* command is VLIN Y+1,Y+2 at X+4. If X and Y are set to zero before the subroutine is called, the face will be drawn from Y=1 to Y=2 at X=4. It can be moved one space to the right by adding one to X before calling the subroutine.

The subroutines that draw the girl in this program are located from 100 to 200. Subroutine 100 draws the upper half of the girl, which is the same in each picture. Subroutine 120 draws the lower half in the standing position (A). 130 draws the lower half with one foot stepping forward (B), 140 is mid-stride (C), and 150 finishes the sequence (D). To make the girl "walk," the main program calls 100 (top half) followed by the appropriate bottom half, going from A to D over and over. The value for X is incremented by one before each successive step is drawn, so that the girl moves across the screen one square at a time. To prevent leaving a trail behind the figure, you also need an erase routine (160 in this program) to remove each figure before the next one is drawn.

You may wish to stop at this point and experiment with making your figure move across the screen without taking advantage of the page-flipping technique described below. This will give you a chance to add additional drawings or to eliminate drawings not needed. The problem you will observe is that it is distracting to see the figure blinking on and off as it is erased and redrawn in front of you. For a very small figure this could be tolerated, but as pictures get larger or more complicated the blinking causes the animation to lose its appeal.

The remainder of this article describes a technique for making your figure move smoothly across the screen without blinking on and off. One section will explain how to "flip pages" to prevent blinking. Another section explains how to reset LOMEM in your Integer Basic program in order to free the memory needed for the second page of graphics. A third section will describe the subroutine that is used to transfer the contents of page one to page two, and the final section will tell you how the main program works.

Flipping Pages

The Apple has two blocks of memory that can be used for low-resolution graphics. They are referred to as pages. Page one is the one you normally use when you enter the command GR. This page is also used for your text statements. Gaining access to page two requires a bit of trickery. One of the tricks is to prevent this page from being used to store the variables in your Integer program. This is performed by putting the statement LOMEM: 3072 as the first line of the program. Unfortunately, you can't just type that in. (Try it and see for yourself!) But there is a way to do it, which is explained below. The second trick is to put a picture on page two. But alas! there is no command to draw or print text on page two. It can be done, however, by making the drawing on page one, in the usual way, and calling a special subroutine in the Apple Monitor to make a copy of page one on page two. This is also explained below.

Setting LOMEM:

There is more than one way to accomplish this task, but the method described here results in the simplest program. It requires doing some things in the Apple Monitor, but each step will be carefully explained, so you should have no trouble. You may want to save the program you have typed so far (if any) before you begin. If you haven't started a program yet, this will be your first line.

Enter the following as the first line of your Integer program: 0 PRINT 3072. Now type CALL-151 to get into the Monitor. (Fig. 3 shows what you will see as we go along.) You should see the * prompt. The first task is to locate the machine language version of the Basic program line 0 which you just typed in. This is done by looking at the numbers stored in two special memory locations in the Monitor – memory locations 00CA and 00CB (or CA and CB for short.) These locations are called pointers, and contain numbers representing the address of the beginning of your program. (Note: The letters A, B, C, D, E, and F are numbers (10 through 15) in machine language. The 0's in machine language are all zeros, so when you see "0" in this part of the article, type a zero on your computer, not a letter 0.)

Type CB (return). You will see 00CB- ##. The actual number represented here by ## will depend on the size of your program and the amount of memory in your computer. In Fig. 3 the number is 90. Now type CA (return). You will see 00CA– ## (## = C3 in fig. 3). The two-digit numbers you just found are the two halves of the four digit address of line zero of your Basic program. The first half of the four digit number is the one you found at CB, and the second half is the one you found at CA. In the example in fig. 3, the whole four digit number is 90C3. By typing this number (using the actual numbers you found on

```
>CALL-151

*CB

00CB- 90
*CA

00CA- C3
*90C3

90C3- 08
*
 00 00 62 B3
*

90C8- 00 0C 01 1C 0A 00 64 36
*90C3:08 00 00 11

*
```

your computer) and pressing (return) two or three times you will see a series of numbers representing line zero of your Basic program.

The first eight two-digit numbers you should see are the following: 08 00 00 62 B3 00 0C 01. The number "08" means there are eight numbers for line zero; the numbers "00 00" mean the first line is line zero; the "62" means PRINT; "B3 00 0C" stands for 3072 and the "01" at the end means "end of this line." By replacing the number 62 with the number 11 you can change the PRINT in line zero to LOMEN:.

To change the 62 to 11, just type in the address you found in CB and CA again (90C3 in fig. 3, but different in your computer), and put a colon immediately after it – but *don't* press return just yet! The colon tells the computer that you are going to enter some new numbers in here. Type the following four numbers: 08 00 00 11, and press return. Type control-C to get back into Basic, and list line zero. It will now read: 0 LOMEM: 3072.

Copy And Flip Subroutine

The workhorse of this program is the subroutine at line 60. It does the job of copying what is on graphics page one to page two. The display on page one is not erased by the copying, and will stay there until your program changes it. Here's how the page-flipping routine works: While your subroutines are drawing or erasing on page one, the viewer is looking at a finished picture on page two. When the drawing on page one is complete, the viewer is switched to page one where he sees the next step in the movement of the figure. He hasn't seen any of the drawing or erasing, so it only appears that the figure has made a slight shift in position. While the viewer is looking at page one, the computer is busy copying page one to page two – invisibly. The subroutine then switches to page two, but the viewer is unaware that anything has happened since page two is now an exact copy of page one. Now, while the viewer is looking at page two, the cycle repeats and a new drawing is begun "behind the scenes" on page one.

The POKE-16300,0 at the beginning of subroutine 60 causes page one to be displayed, showing the viewer the drawing you made while he was looking at page two. The next six pokes specify that it is page one of graphics that is to be copied to page two. The CALL-468 does the actual copying. The last Poke, POKE-16299,0 causes page two to be displayed again, and the subroutine returns to the main program so that the next step in the drawing can be made out of sight on page one.

The speed of the animation can be slowed down if necessary with delay loops such as those in lines 50, 51, and 52. For large or very complicated drawings delay loops won't be necessary, but for a small object which is quickly drawn you may want them.

Main Program

The main program is found from line 1000 to 2000. You can see that it consists mostly of GOSUB's. There is a FOR...NEXT loop, and several X=X+1 commands, which cause the figure to move to the right. Line 1000 clears the graphics and text display and calls the page-flip routine so that the viewer will be looking at a blank screen for a few microseconds while the first drawing is made. Lines 1010 and 1020 draw a sidewalk and place the girl on the sidewalk at the left of the screen, standing still (drawing A). Note that X was set to zero to place her at the left edge, and Y was set to 10 to put her down on the sidewalk. At the end of line 1020 the page flip subroutine is called, followed by a delay subroutine, so that the viewer looks at the standing girl for a brief time before she begins to walk. Lines 1030 through 1050 contain a FOR...NEXT loop during which the drawings B, C, D, A will be made seven times. (The loop starts at B since A was made before entering the loop, and it was desirable to finish with the girl in standing position). Note that the first GOSUB in line 1030 erases the girl. X is then incremented by one before the next drawing is made, so that she will appear to have shifted one space to the right. Each time a new drawing is made it must be erased before incrementing X and making the next drawing, or the figure will leave a trail of itself behind on the screen. In this example program the girl will move across the screen from left to right one square at a time. The last step of the program is to do a POKE-16300,0 at the end, so that the viewer will be left on page one when the program ends.

Final Hints

When you are debugging your program and using the page-flipping subroutine, you may occasionally hear the ominous "syntax error" beep, but be unable to see a message on the screen, and also see no cursor. This probably means you have been caught on page two, and the error message that stopped your program and left you stranded on page two is being displayed on page one. Just type POKE-16300,0 (return), and you will see page one displayed, where the error message and cursor will be visible.

It is hoped that you have a lot of fun with animating your graphics routines, despite the extra effort involved. The same general methods apply to Applesoft programs, but different "tricks" are required to gain access to page two and to do the memory move required. These "tricks" will be described in a future article.

```
>LIST
  0 LOMEM:3072
 10 POKE -16300,0: TEXT : CALL
    -936: GR : GOTO 1000
 49 REM  ** DELAY ROUTINES **
 50 FOR J=1 TO 50: NEXT J: RETURN

 51 FOR J=1 TO 1000: NEXT J: RETURN

 52 FOR J=1 TO 2000: NEXT J: RETURN

 59 REM  ** PAGE FLIPPER **
 60 POKE -16300,0: POKE 60,0: POKE
    61,4: POKE 62,255: POKE 63,
    7: POKE 66,0: POKE 67,8: CALL
    -468: POKE -16299,0: RETURN

 98 REM  ** GRAPHICS FOR GIRL **
 99 REM  ** DRAW TOP HALF **
100 COLOR=8: HLIN X+3,X+4 AT Y:
    VLIN Y,Y+2 AT X+3: PLOT X+
    2,Y+2
105 COLOR=13: VLIN Y+1,Y+2 AT X+
    4: COLOR=6: VLIN Y+3,Y+6 AT
    X+3: VLIN Y+3,Y+6 AT X+4
110 COLOR=3: VLIN Y+7,Y+8 AT X+
    3: VLIN Y+7,Y+8 AT X+4: HLIN
    X+2,X+5 AT Y+9: RETURN
119 REM  ** STANDING LEGS **
120 COLOR=13: VLIN Y+10,Y+11 AT
    X+3: COLOR=8: HLIN X+3,X+4 AT
    Y+12: RETURN
129 REM  ** STEP FORWARD **
130 COLOR=13: VLIN Y+10,Y+11 AT
    X+3: PLOT X+4,Y+10: PLOT X+
    5,Y+11: COLOR=8: HLIN X+3,X+
    4 AT Y+12: PLOT X+6,Y+12: PLOT
    X+7,Y+11: RETURN
139 REM  ** MIDDLE OF STEP **
140 COLOR=13: HLIN X+3,X+4 AT Y+
    10: PLOT X+2,Y+11: PLOT X+5
    ,Y+11: COLOR=8: HLIN X+2,X+
    3 AT Y+12: HLIN X+5,X+6 AT
    Y+12: RETURN
149 REM  ** END OF STEP **
150 COLOR=13: VLIN Y+10,Y+11 AT
    X+4: PLOT X+3,Y+10: PLOT X+
```

```
   2,Y+11: COLOR=8: VLIN Y+11,
   Y+12 AT X+1: HLIN X+4,X+5 AT
   Y+12: RETURN
 159 REM  ** ERASE GIRL **
 160 COLOR=0: FOR J=X+1 TO X+7: VLIN
   Y,Y+12 AT J: NEXT J: RETURN
 999 REM  ** MAIN ROUTINE **
1000 GR : CALL -936: GOSUB 60: TAB
   14: PRINT "GIRL WALKING"
1010 COLOR=5: HLIN 0,39 AT 23
1020 X=0:Y=10: GOSUB 100: GOSUB
   120: GOSUB 60: GOSUB 51
1030 FOR I=1 TO 7: GOSUB 160:X=X+
   1: GOSUB 100: GOSUB 130: GOSUB
   60: GOSUB 50
1040 GOSUB 160:X=X+1: GOSUB 100:
    GOSUB 140: GOSUB 60: GOSUB
   50: GOSUB 160:X=X+1: GOSUB
   100: GOSUB 150: GOSUB 60: GOSUB
   50
1050 GOSUB 160:X=X+1: GOSUB 100:
    GOSUB 120: GOSUB 60: GOSUB
   50: NEXT I
1060 CALL -936: TAB 17: PRINT "THE EN
   D": POKE -16300,0: GOSUB 52
   : GOSUB 160: COLOR=0: HLIN
   0,39 AT 23
2000 TEXT : CALL -936: POKE -16300
   ,0: END
```

©

Oscilloscope

Rob Smythe
Ontario, Canada

Here is a program for physics teachers that makes good use of the Apple's high resolution graphics.

Unless your school's equipment is better than mine, you probably find it tricky to demonstrate waveforms in class. Stabilizing the pattern on an oscilloscope can require painstaking adjustment when the frequency of the inputted sound is changed. You wish to show how the shape of the wave is altered as overtones are added, but textbooks don't show enough. Demonstrating the effect of different separations in the frequencies of two notes requires diagrams, tedious to produce.

With this program you can show effects of varying amplitude and frequency on sine waves, add up to five overtones (each with their own amplitude) and show the resultant wave pattern for up to six different notes. This last facility is useful for demonstrating the cause of beats.

When you run this Applesoft program you will be presented with a table showing that there are no notes presently in memory and a menu prompting you for single Keystroke selection of commands. Touch 1, 2, 3, 4, 5 or 6 and you will be able to set the amplitude and frequency of a note. Enter as many notes as you wish, or change them one by one. Touch P to plot the resultant waveform. After the oscilloscope pattern is drawn and you have finished studying it, return to the menu by pressing any key.

Touching S will enable you to alter the plotting speed, which is initially set at 4. This determines the increment along the x-axis (time axis) between plotted points. When using frequencies over about 500 Hz, you might have to set speed at 1 or 2 (because at coarser settings significant change to the shape might occur between points and be missed: try 800 Hz at speed 4 and speed 1 to see this).

To clear all notes from the table, touch C and confirm with a Y.

Try notes of amplitude 10 to 20 in a frequency range of 100 to 500. Create a complicated note using all overtones, with amplitudes 10 or less (so that you don't go off the top of the "scope"). Beat patterns look nice when you play notes of frequency 1000 and 1050 together.

The Program:

1000's	**print table and menu routine**
1030	**format numbers in display**
1100	**wait for single keystroke input**
1110	**input data**
1120 on	**process data and reject invalid input**
2000's	**plot routine**
2000-2110	**draw axes**
2150-2160	**pick X value in radians**
2170	**sum the waves**
2190	**scale X and Y to fit screen**
2200	**check for off scale values**
2210	**plot**
3000's	**subroutine to check that points are not off scale**

Variables (in order of appearance)

G$	**bell**
SP	**speed (1 = slow to 5 = fast)**
I	**counter, local pointer**
A$	**local input variable**
AMP(I)	**amplitude of the I-th note**
FR(I)	**frequency of the I-th note careful: don't use FRE(I)**
F(I)	**frequency after scaling for plotting**
TIME	**a measure of length of X-axis**
S	**scaling factor**
J	**loop counter**
X	**horizontal coordinate of point**
Y	**vertical coordinate of point**

Suggested Modifications:

1. Very small changes are required to allow for more overtones.

2. Changing TIME in line 2120 will allow for a different range of suitable frequencies. You might add TIME input to the menu, so that beats can be shown effectively with frequencies that are very close together.

3. Of course, adding routines which would produce the sound of the note you have created on the Apple's speaker would make this program tremendously useful. Any volunteers?

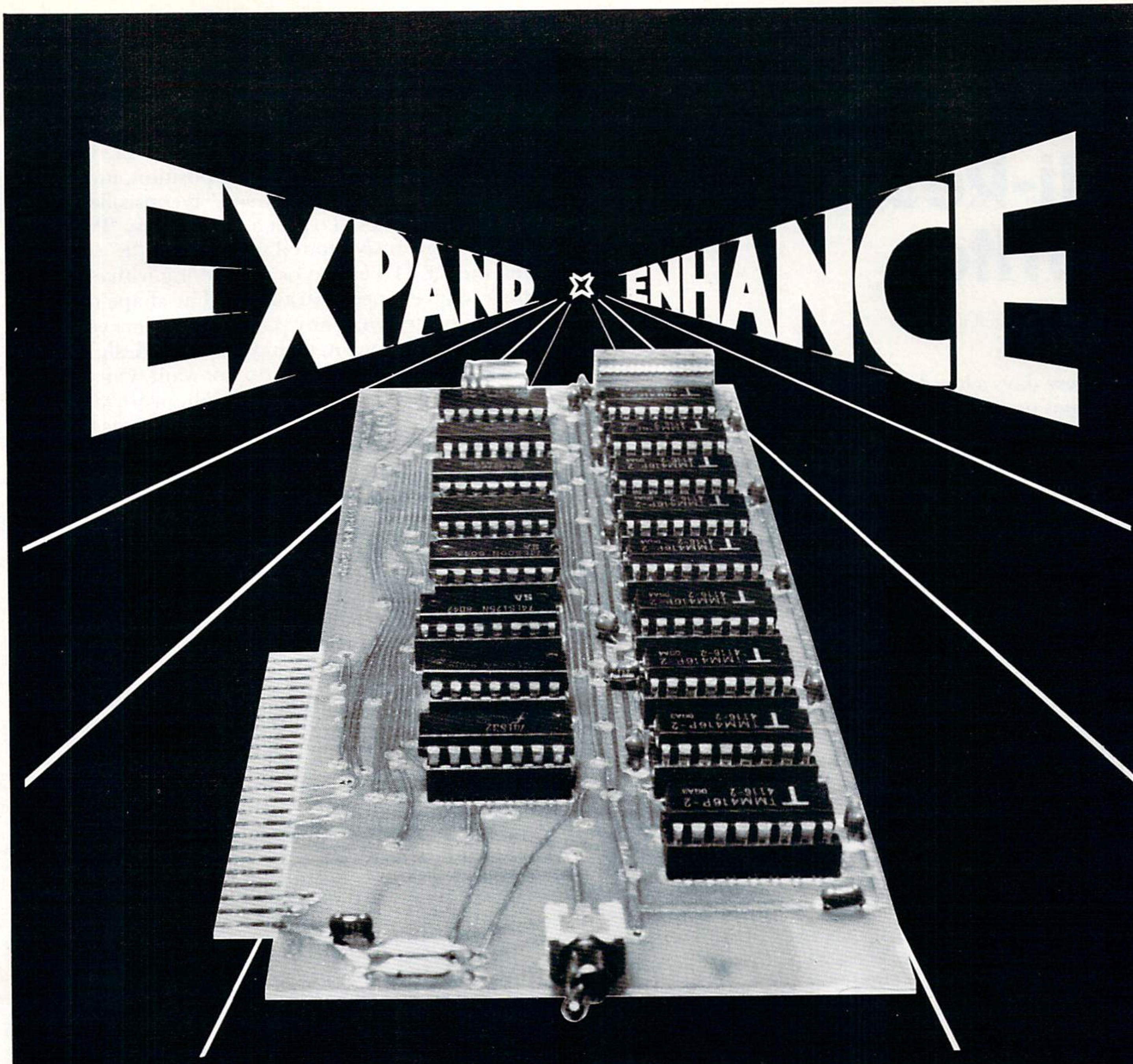
```
]LIST

10  REM   OSCILLOSCOPE

          R.M. SMYTHE
          1522 RUSHOLME CRES.
          BURLINGTON, ONTARIO
          CANADA
```

```
20  REM

        COPYRIGHT (C) 1981
        BY SOFTWARE UNLIMITED

50 G$ =  CHR$ (7): REM ERROR BEEP

100 SP = 4: REM  PLOTTING SPEED
     FROM 1 (SLOW=MOST ACCURATE)
     TO 5
997 :
998  REM  DATA INPUT
999 :
1000  TEXT : HOME
1010  PRINT "    NOTE    AMP    F
    REQ": PRINT
1020  FOR I = 1 TO 6: PRINT  TAB(
    7);I;"       ";
1030 A$ =  RIGHT$ ("      " +  STR$
    (FR(I)),6): IF AMP(I) < 10 THEN
     PRINT " ";
1040  PRINT AMP(I);"  ";A$
1050  PRINT : NEXT I
1060  PRINT : PRINT : PRINT "SPEE
    D - ";SP
1070  VTAB 21
1080  PRINT "CHANGE NOTE: 1/2/3/4
    /5/6    PLOT: P"
1090  PRINT "CLEAR NOTES: C   EXI
    T: E   SPEED: S"
1100  POKE  - 16368,0: WAIT  - 16
    384,128
1110  GET A$:I =  VAL (A$): IF I >
    6 THEN  PRINT G$: GOTO 1000
1120  IF I = 0 THEN 1180
1130  VTAB 21: CALL  - 958: PRINT
    "NOTE ";I;": ";: INPUT "AMPL
    ITUDE (1-10) ";A$:AMP(I) =  VAL
    (A$): IF AMP(I) = 0 THEN 113
    0
1140  IF AMP(I) > 20 THEN  PRINT
    G$;: GOTO 1130
1150  PRINT  TAB( 9);: INPUT "FRE
    QUENCY - ";FR(I): IF FR(I) <
    0 OR FR(I) > 99999 THEN  PRINT
    G$;: VTAB 22: CALL  - 868: GOTO
    1150
1160 F(I) = FR(I) / 27.75
1170  GOTO 1000
1180  IF A$ = "E" THEN  END
1190  IF A$ = "P" THEN 2000
1200  IF A$ = "C" THEN 1240
1210  IF A$ <  > "S" THEN  PRINT
    G$: GOTO 1000
1220  VTAB 21: CALL  - 958: INPUT
    "ENTER SPEED (1-5) - ";SP: IF
    SP < 1 OR SP > 5 OR  INT (SP
    ) <  > SP THEN  PRINT G$: GOTO
    1220
1230  GOTO 1000
1240  VTAB 21: CALL  - 958: PRINT
    "CLEAR ALL NOTES IN MEMORY?
    (Y/N) ": GET A$: IF A$ <  >
    "Y" THEN 1000
1250  FOR I = 1 TO 6:F(I) = 0:FR(
    I) = 0:AMP(I) = 0: NEXT : GOTO
    1000
1997 :
1998  REM  PLOTTING ROUTINE
1999 :
2000  HOME
2010  VTAB 24
2020  HGR
2030  HCOLOR= 3
2040  HPLOT 0,80 TO 279,80
2050  HPLOT 0,16 TO 0,143
2060  FOR I = 0 TO 279 STEP 70
2070  HPLOT I,78 TO I,82: HPLOT 2
     79,78 TO 279,82
2080  NEXT I
2090  FOR I = 16 TO 144 STEP 16
2100  HPLOT 0,I TO 4,I
2110  NEXT I
2120 TIME = 400
2130 S = 280 / TIME
2140  HPLOT 0,80
2150  FOR I = 0 TO TIME STEP SP
2160 X = I * 3.14159 / 180
2170 Y = 0: FOR J = 1 TO 6:Y = AM
     P(J) / 5 *  SIN (F(J) * X) +
     Y: NEXT J
2180 Y = 80 - Y * 16
2190 X = I * S
2200  GOSUB 3000
2210  HPLOT  TO X,Y
2220  NEXT I
2230  POKE  - 16368,0: WAIT  - 16
    384,128
2240  GET A$
2250  GOTO 1000
2997 :
2998  REM  SUBROUTINE CHECK RANGE

2999 :
3000  IF X < 0 THEN X = 0
3010  IF X > 279 THEN X = 279
3020  IF Y < 0 THEN Y = 0
3030  IF Y > 159 THEN Y = 159
3040  RETURN
```

©

Apple Authors

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


EXPAND ENHANCE

The Apple Hi-Res Shape Writer

Doug Hennig
Dallas, TX

These days a lot of people are writing their own games or applications software rather than paying for someone else's labor. After all, designing and writing software really is the best part about owning your own microcomputer. Games especially are fun to design because they tax not only your programming skills, but your imagination as well.

Many of today's popular computer games are of the "arcade" type where you are required to fire missles to destroy any number of different kinds of objects. Usually these objects are drawn and manipulated on the high resolution screen using hi-res shape tables that are stored in memory to define the object. Other types of games also use hi-res shape tables, including a number of recently developed high resolution adventure games. As you have no doubt already observed, the uses of shape tables in graphics programs is almost limitless. However, that is where the problem arises.

Anyone who has ever designed a high resolution shape table knows that it is not a lot of fun to do. You must draw the shape dot by dot, convert each dot into a number, convert the numbers into bytes by an obscure process, type in the long list of bytes, and hope that you have not made an error somewhere along the way (I invariably do!). After several attempts at such nonsense, I thought that there must be an easier way. Why not do just the fun part – drawing the shape – and leave the Apple to do the hard part? Thus the Apple Hi-Res Shape Writer was born.

The Apple Hi-Res Shape Writer allows you to draw any shape on the low resolution screen using the game paddles and then convert the lo-res shape into a high resolution shape table. You will soon see your graphics ideas take shape (no pun intended) quickly and painlessly. Create space ships, alien creatures, even new and exotic character sets.

Operation

Operation of the Apple Hi-Res Shape Writer is simple. First you tell the program how many shapes you want in the shape table (the maximum is ten). It will then draw a dark blue background in low resolution graphics and ask you for the number of vertical and horizontal elements in the shape that is to be drawn next; since the low resolution screen is 40x40, these are the limits for the shape. A black area that is the size of your shape will appear, which can be filled in as you wish. The location of the current plotting position, indicated by a flashing light blue "cursor," is controlled by the game paddles. To plot a point, press "P" and to erase a previously plotted point, press "E" (note that the REPT key can be used along with either of these to give "speed" drawing). The shape does not have to be drawn in any particular order; you can "doodle" if you want, trying different designs and erasing the parts that you do not want. Once the shape is drawn to your satisfaction, press "S" to construct the shape table. After the table is done, you may see the high resolution shape before starting on the next shape. Once you have drawn the desired number of shapes, you have the option of saving the entire shape table on disk.

Hi-res shapes will appear "skinnier" than the lo-res shapes drawn because low resolution blocks are rectangular rather than square. However, a little practice will allow you to easily visualize how your hi-res shape will look and to plan the lo-res shape accordingly.

Since the hi-res shape table is stored in the second page high resolution buffer (starting at 16384 or $4000), The Apple must have at least 24K and Applesoft in ROM.

The Program

There are few "fancy tricks" used in this program. If you are not familiar with the way shape tables are created or used, you should read Chapter Nine of the latest Applesoft manual before trying to follow how this program works. To help explain its operation, I have included a list of major variables and subroutines used in the program, with comments about the uses of each.

Variables Used (In Order of Appearance)

A$ Used for all input from the user.

SH The number of shapes to be drawn (a maximum of ten). This number is POKEd into the first two locations of the shape table.

BASE The starting address of the data for the current shape. It is initially set to 16384 + 2*SH + 2 because the table starts at 16384 and the table index consists of two bytes for the number of shapes and two bytes to point to each shape.

NU The number of the current shape.

TABLE () Holds the plot status of every point in the shape area of the screen. The status is based on the following system:

move right (no plot) = 1	plot and move right = 5
move down (no plot) = 2	plot and move down = 6
move left (no plot) = 3	plot and move left = 7

BLACK, AQUA, RED Hold the values of the corresponding low resolution colors.

COL A dual purpose variable: first it holds the number of horizontal elements in the shape and later it is used as a loop counter for the columns.

C1, C2 Contain the horizontal low resolution screen limits.

R1, R2 Contain the vertical low resolution screen limits.

X, Y Represent the coordinates of the current plot position.

OLD Holds the color of the screen under the "cursor" (black if nothing was plotted, red if a point was plotted).

MOVE Stores the plot value: either move right (1) or move left (3), depending on which row is being scanned (see the variables START, LAST).

DOWN Stores the plot value to move down (2).

START, LAST Hold the loop limits for the column counter. The first row is scanned left to right, the second right to left, and so on (always moving down one row when the end of the current row is reached), so START and LAST are switched at the end of each row.

INC Another dual purpose variable: initially, it holds the step value for the column loop (1 means scanning left to right and -1 means scanning right to left); later it is used to hold the number of bytes in the current shape.

B Holds the octal representation of the current byte to be put into the shape table. Octal is used because in hi-res shape tables, eight possible moves are allowed (see the Applesoft manual).

W, X, Y, Z Store intermediate results in the conversion of octal to decimal. X also holds the final (decimal) result.

D Holds the length of the shape table.

Sections And Subroutines Used

10–37 Introduction.
40–400 Print instructions.
410–450 Initial shape table setup.
460–500 Variable initialization.
510–700 Low resolution screen setup.
710–830 Draw the lo-res shape.
840–1130 Create the hi-res shape table.
1140–1240 Display the hi-res shape.
1250–1500 Save the shape table on disk.
11000–11080 Input user response (avoids the problems of using INPUT).
12000–12100 Print heading.
13000–13040 Go on to next screenful.

Changes And Modifications

A number of changes to the program are possible. The colors of the background, the flashing "cursor," and plotted points may be changed to suit your taste by changing the values in lines 490 and 530 (AQUA is the color of the "cursor" and RED is the color of a plotted point). To allow more shapes per table (up to a reasonable limit, of course), change the upper limit in line 425.

Perhaps the most important modification would concern the speed of execution. Creation of the shape table can take up to several minutes for large shapes because of the amount of data involved. If the time involved seems unreasonable to you, perhaps a machine language version of lines 840 to 1130 would be in order. Since I am not proficient in machine language, I chose to let Basic do the job for me, but I am sure that some enterprising soul can come up with a faster version. If you want to tackle this problem, feel free to contact me with any questions that you have about the program.

Using Shape Tables With Your Own Programs

There are two ways that you can use a shape table in your own program. The program can read in the table from disk (using BLOAD) or the program can POKE in the values for the table from data in DATA statements. The first method is obviously easier to program, but you must ensure that the disk containing the shape table is inserted in the drive or the user will get a nasty DOS error message.

The question with the second method is: how do I convert all those bytes in memory into numbers in DATA statements? One way (the hard way) is to write a short program which reads the shape table one byte at a time, prints the value, and lets you write it down before going on to the next one. Then you have to type all the values into DATA statements and POKE them into memory. The easier way is to EXEC a text file which will do all that for you. Listing 2 contains a program which will set up such a text file. To use this, set the variable LINE in line 90 to the line number that you want the POKE routine to start at, set the variable B in line 90 to the last memory location of the shape table (lines 1460 and 1470 in the Apple Hi-Res Shape Writer print this value for you), RUN the program and EXEC the text file. For example, for a shape table that the computer tells you ends at 17000, type (in the direct mode):

```
LOAD POKE WRITER
90 LINE = 5000 : B = 17000
RUN
```

Now just load your program, EXEC POKE ROUTINE, and you have added a routine, starting at line 5000 in this example, that will POKE your shape table into memory every time the program is run.

I hope that you enjoy Apple Hi-Res Shape Writer. I have found it extremely useful in designing some of my own games and educational software, and I am sure that you will find many uses for it too!

```
?SYNTAX ERROR
]LIST

10  REM ********************************
12  REM ** APPLE HI-RES SHAPE WRITER **
14  REM **       BY DOUG HENNIG       **
20  REM ********************************
25  HIMEM: 16384
30  GOSUB 12000
35  PRINT "DO YOU WANT INSTRUCTIONS? ";: GOSUB 11000
37  IF A$ = "N" THEN  HOME : GOTO 400
40  VTAB 10: CALL  - 868: HTAB 3
50  PRINT "THIS PROGRAM WILL ALLOW YOU TO CREATE"
60  PRINT "UP TO TEN DIFFERENT SHAPE TABLES FROM"
70  PRINT "DESIGNS THAT YOU HAVE DRAWN ON THE"
80  PRINT "SCREEN IN LOW-RESOLUTION GRAPHICS."
90  VTAB 15: HTAB 3
100  PRINT "THE PROGRAM WILL ASK FOR THE NUMBER"
110  PRINT "OF HORIZONTAL AND VERTICAL ELEMENTS "
120  PRINT "THAT YOU WANT. THESE NUMBERS MUST BE"
130  PRINT "BETWEEN 1 AND 40."
140  GOSUB 13000
150  HTAB 3
160  PRINT "A BLUE 'CURSOR' WILL BE DISPLAYED TO"
170  PRINT "INDICATE THE CURRENT PLOT POSITION. USE"
180  PRINT "THE GAME PADDLES TO MOVE THE CURSOR TO"
190  PRINT "THE DESIRED LOCATION AND PRESS ";: INVERSE : PRINT "P";: NORMAL
   : PRINT " TO"
200  PRINT "PLOT AT THE CURRENT POSITION; A PLOTTED"
210  PRINT "POINT IS INDICATED BY A RED SQUARE. TO"
220  PRINT "ERASE A PLOTTED POINT, PRESS ";: INVERSE : PRINT "E";: NORMAL
   : PRINT ". TO"
230  PRINT "BEGIN CREATION OF THE SHAPE TABLE,"
240  PRINT "PRESS ";: INVERSE : PRINT "S";: NORMAL : PRINT ". "
250  GOSUB 13000
260  HTAB 3
270  PRINT "THE SHAPE TABLES WILL BE STORED IN"
280  PRINT "THE HIGH-RESOLUTION SECONDARY PAGE AREA"
290  PRINT "(STARTING AT LOCATION 16384, OR $4000)."
300  PRINT "YOU WILL BE GIVEN THE OPTION TO SAVE "
310  PRINT "THE SHAPE TABLE ON DISK."
320  PRINT : HTAB 3
330  PRINT "TO USE THE SHAPE TABLE IN A PROGRAM,"
340  PRINT "SIMPLY USE A 'BSAVE' COMMAND WITHIN"
350  PRINT "THE PROGRAM. NOTE THAT THE HIGH-"
360  PRINT "RESOLUTION SECONDARY PAGE WILL BE"
370  PRINT "UNAVAILABLE FOR USE."
390  GOSUB 13000
400  POKE 34,0
410  PRINT "HOW MANY SHAPES WILL THERE BE? ";: GOSUB 11000
420 SH =  INT ( VAL (A$))
425  IF SH < 1 OR SH > 10 THEN  VTAB 10: CALL  - 958: GOTO 410
430  POKE 16384,SH: POKE 16385,0: REM PUT NUMBER OF SHAPES INTO START OF T
   ABLE INDEX
440 BASE = 16384 + 2 * SH + 2
450  POKE 232,0: POKE 233,64: REM TELL APPLE WHERE SHAPE TABLE IS
460 NU = 0
480  DIM TABLE(1600)
490 BLACK = 0:AQUA = 6:RED = 1
```

```
500 NU = NU + 1: IF NU > SH THEN 1250
510  GR
520  HOME : VTAB 21
530  COLOR= 2
540  FOR I = 0 TO 39: VLIN 0,39 AT I: NEXT I: REM DRAW BACKGROUND
550  PRINT "NUMBER OF HORIZONTAL ELEMENTS - ";: GOSUB 11000
560 COL =  INT ( VAL (A$))
570  IF COL < 1 OR COL > 40 THEN  VTAB 21: CALL  - 958: GOTO 550
580  HOME : VTAB 21
590  PRINT "NUMBER OF VERTICAL ELEMENTS - ";: GOSUB 11000
600 ROW =  INT ( VAL (A$))
610  IF ROW < 1 OR ROW > 40 THEN  VTAB 21: CALL  - 958: GOTO 590
620 C1 =  INT ((40 - COL) / 2):R1 =  INT ((40 - ROW) / 2)
630 X =  INT (COL / 2 -  INT (COL / 2) + 0.5):Y =  INT (ROW / 2 -  INT (RO
    W / 2) + 0.5)
640 C2 = 39 - C1 - X:R2 = 39 - R1 - Y
650  COLOR= BLACK: FOR I = C1 TO C2: VLIN R1,R2 AT I: NEXT I: REM CLEAR SP
    ACE FOR SHAPE
660  HOME : VTAB 21
670  INVERSE : PRINT "P": PRINT "E": PRINT "S": NORMAL
680  VTAB 21: HTAB 3: PRINT "- PLOT POINT"
690  VTAB 22: HTAB 3: PRINT "- ERASE POINT"
700  VTAB 23: HTAB 3: PRINT "- CREATE SHAPE TABLE"
710 X = C1:Y = R1
720 OLD =  SCRN( X,Y)
730  COLOR= AQUA: PLOT X,Y: REM FLASH CURSOR
740  FOR I = 1 TO 100: NEXT I
750  COLOR= OLD: PLOT X,Y
760 X =  INT ((C2 - C1 + 1) / 255 *  PDL (0)) + C1:Y =  INT ((R2 - R1 + 1)
     / 255 *  PDL (1)) + R1: REM GET NEW COORDINATES
770  IF X > C2 THEN X = C2: REM DON'T GO OUT OF BOUNDS
780  IF Y > R2 THEN Y = R2
790 KEY =  PEEK (49152):A =  PEEK (49168)
800  IF KEY < 128 THEN 720
810  IF KEY = 208 THEN  COLOR= RED: PLOT X,Y: REM PLOT POINT
820  IF KEY = 197 THEN  COLOR= BLACK: PLOT X,Y: REM ERASE POINT
830  IF KEY <  > 211 THEN 720
835  REM CREATE SHAPE TABLE
840  HOME : VTAB 21: PRINT "CREATING SHAPE TABLE"
850 MOVE = 1:N = 0:DOWN = 2
860 START = C1:LAST = C2:INC = 1
865  REM STARTING AT THE UPPER LEFT CORNER, SCAN BACK AND FORTH ACROSS ROW
     S
870  FOR ROW = R1 TO R2
880 : FOR COL = START TO LAST STEP INC
890 ::TABLE(N) = MOVE
900 :: IF COL = LAST THEN TABLE(N) = DOWN: REM IF END OF ROW GO DOWN
910 :: IF  SCRN( COL,ROW) = 1 THEN TABLE(N) = TABLE(N) + 4: REM A PLOT POI
     NT
920 ::N = N + 1
930 : NEXT COL
940 :TEMP = START:START = LAST:LAST = TEMP
950 MOVE = MOVE + INC * 2
960 :INC =  - INC
970  NEXT ROW
980 COL = 0:INC = 0
990 B = 0: FOR I = 0 TO 2: REM CONVERT MOVES TO BYTES
1000 :A =  INT (10 ^ I * TABLE(COL))
1010 :B = B + A
```

```
1020 : IF B > 199 THEN B = B - A: GOTO 1050
1030 :COL = COL + 1: IF COL = N THEN 1050
1040  NEXT I
1050 W =  INT (B / 100):X =  INT (B / 10)
1060 Y = X - W * 10:Z = B - X * 10
1070 X = 64 * W + 8 * Y + Z
1080  POKE BASE + INC,X:INC = INC + 1: REM PUT BYTE INTO TABLE
1090  IF COL < > N THEN 990
1100  POKE BASE + INC,0:INC = INC + 1: REM END OF THIS SHAPE
1110  POKE 16384 + 2 * NU,BASE - 16384 - 256 *  INT ((BASE - 16384) / 256)
    : REM POKE POINTERS TO THIS SHAPE INTO INDEX
1120  POKE 16385 + 2 * NU, INT ((BASE - 16384) / 256)
1130 BASE = BASE + INC
1140  HOME : VTAB 21
1150  PRINT  CHR$ (7)"WANT TO SEE HI-RES SHAPE #"NU"? ";: GOSUB 11000
1160  IF A$ = "N" THEN 1250
1170  HGR : SCALE= 1: ROT= 0
1180  HCOLOR= 3
1190  DRAW NU AT 140,80
1200  HOME : GOSUB 13000
1240  GOTO 500: REM NEXT SHAPE
1250  REM   SAVE ON DISK
1260  GOSUB 12000
1270  VTAB 10: PRINT "DO YOU WANT TO SAVE THE SHAPE TABLE ON"
1280  PRINT "DISK? ";: GOSUB 11000
1290  IF A$ <  > "Y" THEN 1480
1300 D = BASE - 16384: REM  LENGTH OF TABLE
1340  PRINT : PRINT "FILE NAME - ";: GOSUB 11000
1350  IF  LEN (A$) > 30 OR  VAL ( LEFT$ (A$,1)) <  > 0 THEN  VTAB  PEEK (3
    7) - 1: CALL  - 958: GOTO 1340
1355  FOR I = 1 TO  LEN (A$): IF  MID$ (A$,I,1) = "," THEN  VTAB  PEEK (37
    ) - 1: CALL  - 958: GOTO 1340
1357  NEXT
1360  PRINT : PRINT "INSERT THE DATA DISK INTO THE DRIVE"
1380  PRINT "AND THEN PRESS ANY KEY."
1390 KEY =  PEEK (49152): IF KEY < 128 THEN 1390
1400 KEY =  PEEK (49168)
1410  PRINT : PRINT "SAVING SHAPE TABLE"
1415  ONERR  GOTO 5000
1420  PRINT  CHR$ (4)"BSAVE"A$",A$4000,L"D
1430  POKE 216,0
1460  PRINT : PRINT "THE LAST LOCATION IN THE SHAPE TABLE"
1470  PRINT "IS "BASE - 1"."
1480  POKE 34,0
1490  PRINT : PRINT "GOOD LUCK WITH YOUR NEW SHAPE TABLE!"
1600  END
4999  REM CONVERT DECIMAL TO HEX
5000  PRINT : PRINT "THERE WAS A DISK I/O ERROR.": POKE 216,0: END
10999  REM "INPUT" SIMULATOR
11000 A$ = ""
11010  GET B$
11020  IF B$ =  CHR$ (13) THEN  PRINT : RETURN : REM IF RETURN PRESSED GO
    BACK
11025  IF B$ =  CHR$ (21) OR B$ =  CHR$ (10) THEN 11010
11030  IF B$ <  >  CHR$ (8) THEN  PRINT B$;:A$ = A$ + B$: GOTO 11010
11040  IF  LEN (A$) = 0 THEN 11010: REM IF NO CHARS ENTERED IGNORE BACKSPA
    CE
11050  PRINT B$" "B$;
11060  IF  LEN (A$) = 1 THEN 11000
```

```
11070 A$ =  LEFT$ (A$, LEN (A$) - 1)
11080  GOTO 11010
11999  REM PRINTHEADING AND SET TEXT WINDOW
12000  TEXT : HOME
12010  VTAB 2: INVERSE
12020  FOR I = 1 TO 40: PRINT " ";: NEXT I: PRINT
12030  VTAB 4: HTAB 10: NORMAL
12040  PRINT "HI-RES SHAPE WRITER"
12050  HTAB 13: PRINT "BY DOUG HENNIG"
12060  VTAB 7: INVERSE
12070  FOR I = 1 TO 40: PRINT " ";: NEXT I: PRINT
12080  POKE 34,9
12090  NORMAL : VTAB 10
12100  RETURN
12999  REM NEXT SCREEN ROUTINE
13000  VTAB 23: HTAB 8
13010  INVERSE : PRINT "PRESS RETURN TO CONTINUE": NORMAL
13020 KEY =  PEEK (49152): IF KEY < 128 THEN 13020
13030 KEY =  PEEK (49168)
13040  HOME : RETURN

]
```

©

THE ATARI® GAZETTE

Adding A Voice Track To ATARI Programs II

Mike Doleman
Richfield, MN

Deja vu? Not really. Those of you interested in making your ATARI speak, a la a pre-recorded voice track in "sync" with your program may recall an article in the July/Aug. '80 issue of **COMPUTE!** which shows one method of doing this. Since what will be explained in this article results in the same effect, it is titled the same. Hopefully however, you will agree that this new method has advantages.

Some problems do exist with using the other method since, in that situation, the two elements — the voice track and the program — advance independently of each other and in some situations are vulnerable to falling out of sync. Obviously if the two are to be in sync some care must be taken to see that they start out together. Not so obviously, if you happen to put the cassette into a cassette player that runs at a slightly different speed than the one used to obtain the time values, presto — your program and voice track gradually get separated, after all that work you put into synchronizing them. (Not to mention the whole sync procedure being a cumbersome hassle even if it works right.)

Take heart, you can do it the way the "pros" do, and you don't need any "special hardware" to accomplish it with your ATARI system. You probably already have it, an ordinary run-of-the-mill stereo cassette tape recorder with two microphones.

With the stereo cassette recorder and the aid of your ATARI you can put a signal on the digital track of a cassette tape which controls the advancement of the program, and do it at the same time you're recording the voice track — thereby eliminating the need for doing all that other "stuff" when using the other method.

Here's the nuts and bolts, or should I say peeks and pokes of how it's done.

Memory location 53775 (refer to Appendix I, Basic Reference Manual) can be made to return several values by the ATARI program recorder as it plays. These values are governed, as I mentioned before, by a signal on the digital track of the cassette tape, so the first thing to know is which signal does what to 53775 and how to put it on the tape at the proper time. The signal itself is simply an audio frequency and can be easily generated by the SOUND statement on your ATARI. Specifically just two tones are needed, one caused by a SOUND 0,5,10,15 statement and the other by a SOUND 0,8,10,3 statement.

If the signal is the audio frequency produced by a SOUND 0,5,10,15 being recorded on the tape, then on playback in the program recorder it will cause 53775 to return a 255, if the signal is changed to a sound 0,8,10,3 then 53775 will return a 239.

Now! If, in the program which is being used with the voice track, a simple subroutine is placed wherever you want the program to stop and wait for a cue from the tape (the subroutine monitoring 53775 and holding if 53775 = 255 or continuing if 53775 = 239), then you have total control, by the tape, over the progression of the program.

Now we're ready for the step-by-step procedure to actually make a tape that does the job! Presumably you already know where, on the script you have written for your voice track, you want the program to stop running and wait for the voice track to cue it to begin running again. If you haven't done so already, mark these places.

STEP 1. When you set up to record your voice track (in the left channel of your recorder) also put a right channel microphone directly in front of your TV speaker. (The TV you use with your ATARI).

STEP 2. Load the following program into your ATARI:

```
10 SOUND 0,5,10,15
20 FOR X=1 TO 500 : NEXT X
30 IF STRIG(0)=1 THEN 30
40 SOUND 0,8,10,3
50 FOR X=1 TO 100 : NEXT X
60 GOTO 10
```

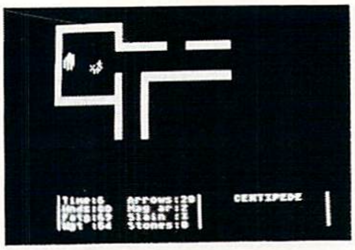
STEP 3. Plug a joystick into controller jack 1 of the ATARI. (Or a paddle and change the 'STRIG' in statement 30 to a 'PTRIG'. If you have neither then use:

```
30 IF PEEK(53279)<>7 THEN 30
```

and press any console switch rather than a trigger button.)

EPYX COMPUTER GAMES THINKERS PLAY

Announcing 3 Challenging New Games For Your ATARI!



Now you and your ATARI 800 can enter a world where fantasy is at your fingertips.

When you play EPYX games, *you* take command! *You* determine the course of history! *You* plan your strategy! Here's more! Our role-playing games are in real-time, so all your decision making has to be done within seconds.

We believe that the games you buy should be superior and uniquely rewarding. That's why we design the game before we design the program. And each game is playtested for hundreds of hours, so you can be sure you're getting the best game *and* the best quality computer program your money can buy.

Thousands of Apple and TRS-80 owners already enjoy these great simulation games — and now you too can add them to your collection. They take full advantage of the ATARI's unique graphics and sound capabilities. Joysticks are optional.

So turn on your ATARI and escape to a universe that's completely in your power!

Invasion Orion.

Turn on your computer and you've got an instant opponent, ready to challenge you in Invasion Orion, the complete tactical science-fiction game from EPYX.

You command up to nine starships. Each ship "spends" energy on moving, sheilding itself and firing its three weapon systems — destructor beam, missiles and torpedos.

The computer takes care of the details, making the game easy to play. And the simultaneous combat is resolved quickly, so you can proceed with your starfleet decisions.

Choose from three levels of play difficulty, 30 ship types and 10 fictional scenarios — or create more of your own! Invasion Orion is infinitely expandable!

- Color graphics and sound!
- Different every time you play!
- For ages 10 through adult.
- Complexity: Intermediate.
- Playing time: 20 minutes to hours of fun!
- For one player.

Suggested Retail Price: $24.95

Rescue at Rigel.

In Rescue at Rigel you've got 60 minutes to find your way thorugh a maze of corridors, chambers, gravshafts and teleportals to release 10 humans held somewhere within. Armed with powergun and blaster, you must battle the insectoid aliens that inhabit the complex, and then get the prisoners — and *yourself* — out alive — in real-time!

But the diabolical Tollah race makes your mission even harder! They move their captives from room to room, so each time you play you must search again!

Your powergun and shield draw energy from your limited powerpack. Your blaster has only a handful of charges, and your rescue ship is under orders to leave — with or without you — in 60 minutes!

Can you save the prisoners before your powerpack is depleted? Can you get back to your rendezvous point in time? Or will the 10 humans be transformed into mindless automatons? *You* are their only hope!

- Color graphics and sound!
- Real-time!
- Different every time you play!
- For ages 10 through adult.
- Complexity: Intermediate.
- Playing time: 20 to 60 minutes.
- For one player.

Suggested Retail Price: $29.95

The Datestones of Ryn.

The treasured datestones of Ryn have been stolen by a dastardly band of robbers! And your mission is to retrieve them before the thieves can escape!

Not only does the real-time action keep you on the edge of your seat, but you've got to finish your quest within 20 minutes! In The Datestones of Ryn, you'll explore a cave complex where the stones are hidden. Armed with sword and bow, you must battle thieves and monsters to reach the stones.

You choose to fire your bow or speak with monsters — parry or thrust, rest or run — from among 14 easy-to-use single-key or joystick-controlled commands. And The Datestones of Ryn has a built-in competitive scoring system, so you can compare your skills with other players!

- Color graphics and sound!
- Real-time!
- Complexity: Introductory.
- Playing time: 5 to 20 minutes.
- For one player.

Suggested Retail Price: $19.95

All of these great EPYX games are available on cassette for the ATARI 800 with 32K of RAM.

STEP 4. RUN the program and adjust the record level of the right channel so the signal is strong. Also adjust the left channel appropriately for recording the voice track.
STEP 5. Position the tape you are using for the voice track to the proper place for recording. (This will probably be immediately behind the program it goes with.)
STEP 6. Begin recording the voice track and every time you come to a place previously marked in the script for the program to continue simply press the trigger button. (Remember to release it fairly quickly as a safeguard to putting more than one signal change on the tape.)

The only thing you now need in the main program is a little subroutine that stops the program and then only allows it to continue on the cue (signal) you have just put on the tape along with your voice track. It is as follows:

```
0000 GOSUB 6000 (Placed wherever you want to stop
     the program.)
6000 FOR X=1 TO 500 : NEXT X
6010 X=PEEK(53775):IFX=255 OR X=127
     THEN 6010
6020 RETURN
```

Obviously the subroutine does not have to be numbered 6000; number it whatever you need for your own program. The delay at 6000 is needed in case the program execution to the next place you want stopped is so fast that it catches the tape still signaling for a change, which is not an improbability. And the number 127 needs to be included in 6010 because sometimes the ATARI decides to return it instead of 255.

And that is it! You have just eliminated any need to collect all those time values and go back to your program to put them in the proper place. Along with that, there is now no chance that the voice track and program will go out of 'sync'.

A word about an inconvenience which may manifest itself in the form of the TV speaker interfering with the voice track recording while making the signals for the digital track. It may be a good idea to make a direct hookup from the ATARI monitor jack to the right channel of the cassette recorder. Pin hole #3 (upper left) is the audio and a 1 mega ohm resistor should be used to prevent cross channel recording. Also note that the procedure for allowing the program to turn the program recorder on and off, a necessary function for making voice track/program combinations, is simply POKE 54018,52 for 'on' and 60 for 'off'. ©

Atari Tape Techniques

Richard M. Kruse
Wichita, Kansas

The Atari 400/800 personal computers have more built-in capabilities than most users realize. This article expands on one such area – the tape-handling functions. No special knowledge or equipment beyond the Atari computer and tape recorder is required, and the Atari user who is so equipped can try the new procedures immediately. Included is the listing for a BASIC program called "Hex Tape Dump," which allows display and examination of Atari tape records.

It is a safe assumption that nearly all owners of Atari 400/800 computers have also acquired the Atari model 410 cassette tape recorder. This peripheral device was originally shipped with each Atari 800 computer, but is now sold as an optional accessory useable with either model. The 410 is the basic mass storage device in the Atari system. It's simple, reliable, and economical (roughly one-eighth the cost of a single disk drive).

Yet, it's also a good bet that most Atari owners are not using their tape systems at anywhere near full potential. Now we Atari fans are, in general, no more lazy than the rest of the computerists out there, but, alas, we have been a little short on information concerning our machines. We recognize, though, that the Atari folks out in Sunnyvale have struggled mightily to correct this situation, and we applaud their efforts. The Atari Basic Reference Manual is, in fact, a model of organization that others would do well to emulate. Still, the Atari is a complex machine, and even this excellent manual barely scratches the surface of Atari's capabilities. This is the case with the more advanced cassette functions.

(An aside: In case you've heard about the new Atari tech manuals ... yes, they're real, and they're crammed with data. They are also VERY heavy reading. Unless you're on easy speaking terms with the 6502 and advanced machine-language programming techniques, stick with the material presented in these pages. The same high-class stuff will soon be showing up here; a little at a time and with enough explanation to make it useable to all.)

If you have had the feeling that maybe you were missing something along these lines, then read on. The information in this article will get you past "CSAVE" and "CLOAD", and show you how to get more 'bang per buck' out of your 410 recorder while sticking with BASIC language.

Atari fans ... to your keyboards!

THERE'S LIFE BEYOND CLOAD!

First, let's examine some tape functions whose commands are built-in; that is, no POKES or PEEKS are required. Take a few moments here to dust off your Atari Basic Reference Manual and reread the descriptions in the "Input/Output Commands" section for "ENTER", "INPUT", "LOAD", "OPEN/CLOSE", "PRINT", "PUT/GET", and "SAVE". Each of these commands is applicable to a tape function, and there are important differences between them which seem, on the surface, to be redundant.

THE LONG AND SHORT OF IT

All data written to tape by the Atari OS (operating System) is first formatted into 128-byte blocks called RECORDS. The OS then appends four additional bytes of control data to each record, forming 132-byte FRAMES. Since 128 bytes is a fairly small amount of data, several frames will normally be recorded sequentially to make a FILE. Adjacent frames in a file are never contiguous, however, but are separated by non-data spaces called INTER-RECORD GAPS (IRG's). Refer to Figure 1, which is a pictorial diagram that should help clarify the relationship between these entities.

We now have enough information to understand and appreciate the difference between CSAVE and SAVE"C". Both of these BASIC commands save a BASIC source program to the cassette, but CSAVE uses short IRG's (less than a

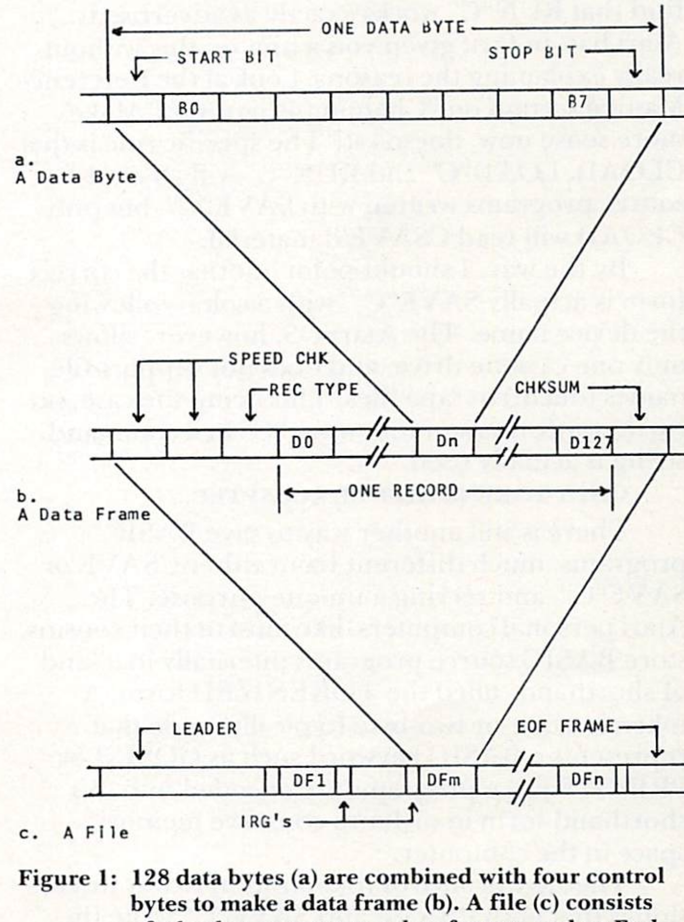


Figure 1: 128 data bytes (a) are combined with four control bytes to make a data frame (b). A file (c) consists of a leader, one or more data frames, and an End-Of-File frame.

second) while SAVE"C" uses long gaps (about three seconds).

What practical difference does this make? And why would I ever use SAVE"C" anyway, since it makes the snail-like pace of program loading even slower? Okay, here we go. The significance of long IRG's, and the reason for providing the option to use them, is this: The long gap provides just enough space between records to stop the tape drive motor, restart it when ready, and bring the tape back up to speed before the beginning of the next sequential frame of data. This cannot be accomplished with short gaps, and under some conditions the computer cannot accept data fast enough to accomodate non-stop operation. (In all fairness, there is another reason for the apparent redundancy of CSAVE and SAVE"C". While CSAVE applies specifically and only to the tape recorder, SAVE"C" is one form of a more general command that also applies to disk files.)

Now, I can almost hear you saying, "Well, that's nice to know, sort of, but the information doesn't really seem too useful." Right? You would be right, of course, except for a function called RUN"C" which, according to the Reference Manual, loads a BASIC program and immediately begins execution. The only problem is, it doesn't work with a CSAVE'd program ... Aha! You're way ahead of me. Yes, indeed, try SAVE"C" and you'll find that RUN"C" works exactly as advertised. Atari has, in fact, given you a hint on this without really explaining the reasons. Look at the Reference Manual section on "Chaining Programs." Makes more sense now, doesn't it? The specific rule is that CLOAD, LOAD"C" and RUN"C" will all read source programs written with SAVE"C", but only CLOAD will read CSAVE'd material.

By the way, I should point out that the correct form is actually SAVE"C:" with a colon following the device name. The Atari OS, however, allows only one cassette drive, and does not support file names (ouch!) in tape files. This being the case, no further information following "C" in a command string is actually used.

ONLY TOKENS WILL BE ACCEPTED...

There is still another way to save BASIC programs, much different from either CSAVE or SAVE"C", and serving a unique purpose. The Atari personal computers, like most of their cousins, store BASIC source programs internally in a kind of shorthand called the TOKENIZED form. A token is a one-or two-byte (typically) code that represents a BASIC keyword such as GOSUB or PRINT. Source programs are encoded into this shorthand form in order to conserve memory space in the computer.

The significance of tokenization to our discussion is that both CSAVE and SAVE"C" write the shortened, tokenized form of your BASIC program to the cassette. Not surprisingly, CLOAD, LOAD"C", and RUN"C" all recognize only this abbreviated code. There exists, however, a method of storing the original, expanded ASCII source code. And, just as with long IRG's, there are a couple of good reasons to use this method.

CAUTION: MERGING TRAFFIC

Every BASIC programmer is familiar with the LIST command, with which he can review previously-entered source code on the CRT or produce a permanent program listing on a printer. Atari's treatment of I/O devices, however, allows you to LIST your program to the cassette as well. The command format is LIST"C", to save all source lines; or LIST"C",X,Y to save only those from line X to line Y, inclusive. Either way, program lines will be stored in full ASCII form, including line numbers, REMarks, and (most) spaces. There has to be a way, of course, to retrieve such a listing, and this is done with the command ENTER"C".

The LIST"C"/ENTER"C" sequence has one particular characteristic that makes it indispensible in preparing BASIC programs. Unlike CLOAD and LOAD"C", which both clear any BASIC source lines from memory before loading, ENTER"C" does not necessarily disturb a resident program. This means that you can merge often-used routines into a BASIC program without having to retype them each time. Once you become familiar with this process, it can save lots of time and effort.

Actually, ENTERing a source program from tape is exactly equivalent to typing source lines at the keyboard, and the same rules generally apply. If a line is entered from tape, for example, having the same line number as a line already in memory, the old line will be replaced.

THE INCREDIBLE SHRINKING CODE

Another interesting and potentially useful effect of the LIST"C"/ENTER"C" sequence is that it can actually reduce the size of your BASIC programs. I stumbled across this undocumented characteristic while doing my "homework" for this article, and I do not at present know the reasons for it. Here's how to try it for yourself:

> CLOAD a BASIC program, preferably one that's undergone lots of editing. When it's loaded, type "PRINT FRE(0)", and write down the number of unused memory bytes. Now store the source back to tape, using LIST"C". Clear the computer's BASIC program area by typing "NEW," and reload the code, using ENTER"C", from the tape just made. Finally, type "PRINT FRE(0)" again, and compare the result with the one you write down.

I have tried this on several existing programs, and have gotten memory savings anywhere from a

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.

few bytes (who cares!) to over 500 bytes (impressive!). As I said earlier, I haven't delved into the reasons for this effect, but an observation of the results quickly shows one repeating pattern. The memory saving seems to be greatest when applied to programs that have been heavily "massaged" through editing, especially if variables have been used and then discarded. If true, the implication is that the Atari line-editing process is not one hundred percent effective in deleting program statements. Hopefully an inquisitive reader will determine exactly what it is worth for himself.

BASICALLY BASIC FILE HANDLING

As we have now seen, the Atari cassette system suffers from the worst drawbacks of all such systems. It is slow, file names are not supported, and only a single device can be used. Furthermore, except for the motor, the recorder is under manual control, and, in fact, the computer cannot even tell whether or not the tape is running! The combination of these characteristics pretty well precludes any but the most elementary file-handling functions. This section, therefore, may be seen mostly as an exercise with only minimum practical value.

It is possible to store and retrieve data via the cassette recorder from within a BASIC program using only built-in commands. Once again, the Atari BASIC Reference Manual alludes to this, but does not go into sufficient detail so that users can fully understand the methods. The problem is complicated, in this case, by an acknowledged OS "bug" which may thwart attempts to use the capability.

Due to the limited usefulness of the tape file functions, we will not spend much time on them, except to define how they are invoked.

The cassette recorder, like all Atari I/O devices, must be identified to the OS before it can be used. This is the function of the OPEN command, which links the cassette function to one of the INPUT/OUTPUT CONTROL BLOCKS (IOCB's). The command structure is OPEN#1,4,0,"C" to read from tape, or OPEN#1,8,0,"C" to write. You cannot open the cassette for simultaneous read and write, for obvious reasons. The first parameter in the OPEN command (#1, above) tells the system which of the eight IOCB's to assign. They are numbered from zero through seven, and as far as the OS is concerned they are all functionally identical. Stay away from #0 and #7, however, since these two IOCB's are "appropriated" by the OS for the screen (#0) and for its own tape functions (#7). The second parameter identifies the "direction" of data flow; either input to the computer ("4") or output from the computer ("8"). The third parameter is not used by the cassette system but must be present in the command ... use "0".

Once the cassette is opened, you may read or write data either as pure binary bytes or as ASCII-encoded character strings.

To write a single byte (any value from 0 to 255), use the command PUT#1,X. (X is a representative name for any variable which evaluates to an integer value in the range just stated.) GET#1,X is the corresponding command which will read these values back. It is important to realize that floating-point numerical values are represented internally as multi-byte groups, and, as such, cannot be handled directly by PUT or GET.

The other method of saving and retrieving tape data is to use ASCII representation, with one character per byte. This method is considerably less efficient with tape space, but does allow you to store anything that can be represented in a PRINT statement. The PRINT command, in fact, is what you use to write such information, using PRINT #1;data. The data can be numeric variables, string variables, literals, arrays, or any combination of these, just as in an ordinary PRINT command. (In this way floating point values can be saved. The value 2.45, for example, would be written as a series of four ASCII bytes: 32,2E,34,35 (hexadecimal representation).) Be aware that the system will append the End-of-Line character (9BH) to the last data item unless the PRINT#1 statement is terminated with a semicolon (;).

The corresponding command for reading ASCII data back is INPUT#1,var. Here is where the EOL character just mentioned becomes very important-the INPUT#1 command will attempt to keep reading until it finds this character. INPUT#1, like PRINT#1, uses the same general rules as does its keyboard-related counterpart.

Following the last PUT#1 or PRINT#1 command, a cassette file absolutely must be closed, using CLOSE#1. If you do not do this, you will lose some of the data that you thought was written.

At this point we need to look at some of the internal "mechanics" used by the Atari OS to format your data into the 128-byte records which are actually recorded. A 131-byte block of memory locations is reserved for use as a cassette buffer. (This is identified in your BASIC Reference Manual Memory Map as addresses 3FDH through 47FH.) Each time you execute a PUT#1 or PRINT#1 command, the resultant data is temporarily stored in the last 128 bytes of this buffer. Each time the 128th byte is stored, the Atari immediately suspends operation of your program, starts the cassette motor, writes the data block, and stops the motor. The internal buffer pointer is then reset to zero, and control is returned to your program (which could actually be in the middle of dumping data from a PRINT#1 statement!). In order for any of this to happen, of course, the cassette must first be OPENed as already described.

Opening the device for a write operation causes this sequence of events: (1) two beeps are emitted from the console speaker, and the computer waits for you to enable the recorder; (2) the cassette motor is started; (3) approximately 20 seconds of "leader" tone is written; and (4) the motor is stopped ... OOPS!!! Correction-the motor keeps on running! Here is the OS "glitch" mentioned earlier. The motor will not stop until the first data record has been written. The easiest way to get around this is to immediately write a "dummy" record using zeroes or spaces, or any other data. Here's how to do it:

```
.
.
100 OPEN#1,8,0,"C"
110 FOR I=0 TO 127:PUT#1,0:NEXT I
.
.
```

Your program will, of course, have to allow for this dummy record when retrieving the data. Once the first record has been written, the system is back on good behaviour ... the motor is started only when the buffer needs to be dumped, and stopped promptly after writing.

Realizing that there is an intermediate data buffer between your program and the cassette clarifies the need to close all files. The CLOSE command immediately causes the buffer to be dumped to the tape, even though it may not be full. (This is why you will lose data if a file is not closed properly). After the last data record is written, the OS (automatically) appends an End-of-File record before stopping the tape. NOTE: The END statement automatically closes all open files, but STOP does not.

Please note that the IOCB number need not be #1 as in these examples, but must be the SAME value in all OPEN, GET, PUT, INPUT, PRINT and CLOSE statements accessing this particular file.

BRINGING IT ALL TOGETHER

The BASIC program listing which follows accomplishes two things: it illustrates one possible method of reading a tape file from BASIC, and it provides a function that will prove indispensable to you when you try some of the procedures that we have explored. The program, called "Hex Tape Dump," reads any Atari-recorded file having long IRG's, one record at a time. After each record is read, the contents of that record are displayed on the screen in both hexadecimal and ASCII form.

This program makes use of a very important bit of information not yet discussed. One of the additional control bytes automatically appended to each record prior to writing contains a code describing the length and nature of that record. This byte shows up at location 1023 (decimal) following a read, and can have one of three values, with the following meanings: 252 (decimal) means that the record just read is a full record; that is, it contains 128 bytes of valid data. 250 (decimal) indicates that the record is only partly filled. In this case only, the last byte of the buffer will contain the actual number of valid data bytes. 254 (decimal) says that this is an End-of-File record, in which all data bytes are zero. When you find this control code, you have read past the end of valid data.

"Hex Tape Dump" is an ordinary BASIC program in all respects, and it is self-prompting when entered exactly as shown. Try it, and then spend some time playing with the BASIC cassette functions. You will end up with an even greater appreciation of the capabilities of this machine called ATARI.

*ATARI is a registered trademark of Warner Communications Inc.

```
LISTING 1: HEX TAPE DUMP

10    GRAPHICS 0:DIM BUF (8)
15    TRUE=-1:FALSE=0:FBT=TRUE:TOP=TRUE:PAGE=FALSE
20    PRINT CHR$(125):PRINT"       >>> ATARI HEX
      TAPE DUMP <<<"
25    PRINT:PRINT:PRINT"POSITION TAPE TO START OF
      FILE, THEN"
30    PRINT"PRESS Λ PLAY Λ : Λ RETURN Λ & STAND BY...";
35    REM OPEN THE CASSETTE FOR READ
40    TRAP 1000:REC=1:OPEN #1,4,0,"C"
45    REM DSPLY EIGHT BYTES PER LINE
50    FOR I=1 TO 8
55    REM FILL LINE IF OUT OF DATA
60    IF PAGE THEN PRINT"-- ";:BUF(I)=0:GOTO 100
70    GOSUB 500:BUF(I)=BYTE:IF NOT TOP THEN 90
75    REM PRINT TOP-OF-PAGE HEADER
80    TOP=FALSE:PRINT CHR$(125):PRINT"> RECORD #";
      REC;"...";NBTS+1;" BYTES"
85    PRINT:PRINT" ----------HEX----------   -ASCII
      --":PRINT
90    GOSUB 2000:PRINT" ";
100   NEXT I:PRINT"  ";
105   REM NOW PRINT ASCII FROM BUF
110   FOR I=1 TO 8:BYTE=BUF(I)
120   IF BYTE<32 OR BYTE>122 THEN BYTE=46
130   PRINT CHR$(BYTE);:NEXT I
150   PRINT:IF NOT PAGE THEN 50
160   PRINT:PRINT"PRES Λ SPACE Λ TO CONTINUE...";
170   REC=REC+1:PAGE=FALSE:TOP=TRUE
175   REM WAIT FOR USER RESPONSE
180   IF PEEK(764)<>33 THEN 180
190   POKE 764,255:GOTO 50
500   REM READ BYTE & SET FLAGS
505   GET #1,BYTE:IF NOT FBT THEN 600
510   FBT=FALSE:NBTS=1:STAT=PEEK(1023)
520   IF STAT=250 THEN NBTS=PEEK(1151)
530   IF STAT=252 THEN NBTS=128
600   NBTS=NBTS-1:IF NBTS>0 THEN RETURN
610   FBT=TRUE:PAGE=TRUE:RETURN
1000 REM DONE OR ERROR
1005 PRINT:IF PEEK(195)=136 THEN PRINT CHR$(28);
     "*** READ PAST END-OF-FILE ***":END
1010 PRINT CHR$(28);"--TASK ABORTED...ERROR NUMBER ";
     PEEK(195):END
2000 REM PRINT ONE BYTE IN HEX
2005 NYB=INT(BYTE/16):GOSUB 2100
2010 NYB=BYTE-NYB*16
2100 IF NYB<10 THEN PRINT CHR$(NYB+48);:RETURN
2110 IF NYB<16 THEN PRINT CHR$(NYB+55);:RETURN
2120 PRINT " ! ";:RETURN
```

©

Atari Graphics: 16 Colors!

Clyde H. Spencer
Mountain View, CA

Would you like to be able to have graphics displays with more than four colors on the screen simultaneously? Would you find it useful to be able to draw dotted, colored lines or fill shapes with textured color? If your answer is yes, then read on and I will tell you how to do something that not only isn't documented, but if you could find someone at ATARI to talk about it, they would probably tell you "it can't be done without the GTIA chip." That is the creation (or at least simulation) of a variation of playfield graphics modes 9, 10 and 11.

I can almost hear you mumbling to yourself now, "I didn't know there were graphics modes higher than 8! And what is this thing called a GTIA chip?" Let's talk about the GTIA chip first. As I have been able to unscramble the history of this little wonder called George's Television Interface Adapter chip, it started out life as a custom designed prototype that would do everything that the production CTIA chip (the chip in the computer that handles the graphics) does and a little more. That little more was GRAPHICS modes 9, 10 and 11. They have a resolution of 80 pixels horizontally by 192 pixels vertically with up to 16 different colors or luminescences on the screen simultaneously, out of the 128 possible. But alas, the lost little chips were never put into production. In order to meet marketing deadlines, the simpler, less powerful CTIA chips were installed in the production model computers. That is the bad news! The good news is that the Operating System and 8K Shepardson BASIC were written with the ability to implement these higher graphics modes, if the GTIA were installed. This was presumably done either as a hedge against a last-minute marketing decision to put it in, or with a view to offering this chip sometime in the future as a model upgrade like new chrome on a "Detroit behemoth." But you needn't wait for next year's model. With a little PEEKing and POKEing and a modification of the display list (see the article by Patchett in Vol. 1, issue 6 of **COMPUTE!**) you too may have colored icing on your cake.

The trick is accomplished with a short subroutine that modifies the GRAPHICS mode 8 display list to look like what a GRAPHICS mode 10 display list should look like. This is accomplished by replacing the graphics type (instruction operation code) for GRAPHICS mode 8, in the display list, with the graphics type for GRAPHICS mode 10. Finally, it is necessary to poke a 10 into one of the appropriate Operating System "shadow" registers in RAM to make the system expect to find more than four colors to display. Table 1 is a listing of the subroutine which I call "TEN,". The subroutine starts at line number 32000, so that it can be appended easily to almost any size program. It is liberally sprinkled with remarks. All essential statements are multiples of 10; the other line numbers are REMark statements, which may be deleted.

Unfortunately, it is difficult to select any given desired combination of colors because there is an interaction between the color registers; setting a particular value may affect the others. Generally, it will require considerable experimentation. I have as yet been unable to discover how exactly to predict color selection. What I have observed is as follows: Color registers 0, 1 and 2, loaded with the SETCOLOR command, create solid colors for "DATA" values (see "PALLETTE", Table 2) of 5, 10 and 15, respectively. The other "DATA" values will create colors that may be interspersed with the background color. However, SETCOLOR 0 also effects "DATA" values 1, 4, 6, and 9. Similarly, SETCOLOR 1 effects 2, 6, 8 and 9 and SETCOLOR 2 effects 3, 7, 11, 12, 13 and 14. By appropriate choice of color and luminescence values for the color registers, more than 3 solid colors may be created. There always seems to be some duplication of colors, but you can expect to get three to nine solid colors and an additional five or six colors that will create dotted lines. To get an idea of the colors that can be created, run "PALETTE." It will draw bands of color corresponding to "DATA" values of zero to 15, from the top down. Experiment with changing the constants A, B and C and see what happens! If anyone can shed more light on this subject, I would appreciate hearing about it.

A printout of the various modes' display lists would be instructive on how the ATARI does its graphics displays. But, with 198 addresses and their contents, it would consume too much space in this article. Alternatively, I have provided you with a short program called "DISPLAYLIST" (see Table 3) that will dump the standard display lists to your printer. If you wish to examine the "special" display list also, then first merge the subroutine "TEN" with "DISPLAYLIST." To merge the subroutine, first LOAD "TEN", then use the command LIST "C: or LIST "D:TEN". Then LOAD "DISPLAYLIST" and use the direct mode command ENTER "C: or ENTER "D:TEN" for cassette or disk respectively. Then delete lines 100 through 126, and replace line 130 with a GOSUB 32000. Then RUN as usual. For a detailed explanation of the display lists and the meaning of the

operation codes, I suggest that you refer to IRIDIS #1 by The Code Works.

The illustration program which I call "SPRAY" (Table 4), draws colored lines of different length which appear to radiate from a common point in the center of the screen. The color register values used were arrived at by experimentation with "PALETTE" and seem to give the greatest variety of solid colors. The colors are changed randomly by changing the values of color register 2 (SETCOLOR 2). If you allow the program to run long enough for the attract mode to be invoked by the Operating System, then all the color registers will be changed automatically. This will then also change the background color and lend additional interest to the patterns formed.

The resolution of GRAPHICS modes 9, 10, and 11 (192v X 80h) is a departure from the usual ATARI practice of having twice as many pixels horizontally as vertically. However, that very departure from the "standard" is itself an additional advantage in that it provides flexibility in what can be done with "direct" graphics modes. One does not have to resort to "mixed-modes" to achieve special effects. "Mixed-Modes" graphics, where one custom-tailors the display list, line by line, allows virtually unlimited flexibility in designing a screen format, but I would not recommend this to the novice. However, this ready-made subroutine approach for multiple colored and dotted lines is simple and straightforward and should further your appreciation of that wonderful and versatile machine, the ATARI.

TABLE 1. Subroutine TEN.

```
32000 REM : *TEN* a Subroutine to
32001 REM : create a simulated GR.10.
32002 REM : Written by Clyde Spencer.
32003 REM
32004 REM
32005 REM : Create GR.8 mode display
32006 REM : list without text.
32010 GRAPHICS 8+16
32013 REM
32014 REM
32015 REM : Locate address of display
32016 REM : list.
32020 LET DL=PEEK(560)+256*PEEK(561)
32023 REM
32024 REM
32025 REM : Turn off the ANTIC chip.
32030 POKE 559,0
32033 REM
32034 REM
32035 REM : Place new instruction op
32036 REM : codes in display list.
32040 POKE DL+3,78:POKE DL+99,78
32043 REM
32044 REM
32045 REM : Begin 1st insertion loop.
32050 FOR INSERT=DL+6 TO DL+98
32060 POKE INSERT,14
32063 REM
32064 REM
32065 REM : Increment insertion loop.
32070 NEXT INSERT
32073 REM
32074 REM
32075 REM : Begin 2nd insertion loop.
32080 FOR INSERT=DL+102 TO DL+198
32090 POKE INSERT,14
32093 REM
32094 REM
32095 REM : Increment insertion loop.
32100 NEXT INSERT
32103 REM
32104 REM
32105 REM : **Change timing.**
32110 POKE 87,10
32113 REM
32114 REM
32115 REM : Turn on ANTIC chip.
32120 POKE 559,34
32125 REM
32130 RETURN
```

TABLE 2. PALETTE Program.

```
50 REM : *PALETTE* a demonstration of
51 REM : multiple colors in GR.10.
52 REM : Written by Clyde Spencer.
53 REM
54 REM
95 REM : Go to subroutine to create
96 REM : simulated GR.10.
100 GOSUB 32000
103 REM
104 REM
105 REM : Assign  register variables.
110 LET A=1
120 LET B=5
130 LET C=10
133 REM
134 REM
135 REM : Set color registers.
140 SETCOLOR 0,A,C
150 SETCOLOR 1,B,C
160 SETCOLOR 2,C,C
163 REM
164 REM
165 REM : Begin drawing loops.
170 FOR DATA=0 TO 15
173 REM
174 REM
175 REM : Set color value.
180 COLOR DATA
183 REM
184 REM
185 REM : Begin inner drawing loop.
190 FOR BAR=10 TO 20
193 REM
194 REM
195 REM : Draw lines.
200 PLOT 0,10*DATA+BAR
210 DRAWTO 79,10*DATA+BAR
213 REM
214 REM
215 REM : Increment inner drawing loop.
220 NEXT BAR
223 REM
```

```
224 REM
225 REM : Increment outer 'data' loop.
230 NEXT DATA
233 REM
234 REM
235 REM : Enter an infinite loop to
236 REM : keep screen from clearing.
237 REM : Hit 'BREAK' key to rerun.
240 GOTO 240
```

TABLE 3. DISPLAYLIST Program.

```
50 REM : *DISPLAYLIST* A program to
51 REM : print out the address and
52 REM : operation codes of the dis-
53 REM : playlist.
54 REM : Written by Clyde Spencer.
55 REM
56 REM
95 REM : Reset TRAP and ask for INPUT
96 REM : again if input is not numeric.
100 TRAP 100
103 REM
104 REM
105 REM : Ask for GRAPHICS modes 0-40.
110 PRINT "WHAT GRAPHICS MODE";
113 REM
114 REM
115 REM : Input GRAPHICS mode numeric
116 REM : value.
120 INPUT MODE
123 REM
124 REM
125 REM : Set GRAPHICS mode and create
126 REM : new display list.
130 GRAPHICS MODE
133 REM
134 REM
135 REM : Locate starting address of
136 REM : display list.
140 DISPLAYLIST=PEEK(560)+PEEK(561)*256
143 REM
144 REM
145 REM : Begin loop to list contents
146 REM : of display list.
150 FOR ADDRESS=0 TO 200
153 REM
154 REM
155 REM : Assign new address to
156 REM : variable called CONTENTS.
160 LET CONTENTS=DISPLAYLIST+ADDRESS
163 REM
164 REM
165 REM : Print address and op codes.
170 LPRINT " ",CONTENTS,PEEK(CONTENTS)
173 REM
174 REM
175 REM : Check for JUMP op code (end
176 REM : of display list).
180 IF PEEK(CONTENTS)=65 THEN GOTO 200
183 REM
184 REM
185 REM : Increment print loop.
190 NEXT ADDRESS
193 REM
194 REM
195 REM : Remember to POP the stack if
196 REM : you will be doing anything
197 REM : else.
200 END
```

TABLE 4. SPRAY Program.

```
50 REM : *SPRAY* a demonstration of
51 REM : multiple colors in GR.10.
52 REM : Written by Clyde Spencer.
53 REM
54 REM
95 REM : Go to subroutine to create
96 REM : simulated GR.10.
100 GOSUB 32000
103 REM
104 REM
105 REM : Generate random number
106 REM : between 3 and 5.
110 LET C=INT(RND(0)*3)+3
113 REM
114 REM
115 REM : Set color registers.
120 SETCOLOR 0,1,8
130 SETCOLOR 1,10,8
140 SETCOLOR 2,C,8
143 REM
144 REM
145 REM : Set quadrant flag.
150 LET SIGN=1
153 REM
154 REM
155 REM : Begin nested drawing loops.
160 FOR DO=1 TO 2
170 FOR DATA=1 TO 15
173 REM
174 REM
175 REM : Assign drawing color.
180 COLOR DATA
183 REM
184 REM
185 REM : Pick random X&Y coordinates.
190 LET X=INT(RND(0)*40)
200 LET Y=SIGN*INT(RND(0)*96)
203 REM
204 REM
205 REM : Plot colored lines.
210 PLOT 40-X,96-Y:DRAWTO 40+X,96+Y
213 REM
214 REM
215 REM : Increment color loop.
220 NEXT DATA
223 REM
224 REM
225 REM : Reset quadrant flag.
230 LET SIGN=-1
233 REM
234 REM
235 REM : Increment symmetry loop.
240 NEXT DO
243 REM
244 REM
245 REM : Pause to appreciate.
250 FOR DELAY=1 TO 1000:NEXT DELAY
253 REM
254 REM
255 REM : Do it all again!
260 GOTO 110
```

©

Assembler Joystick Driver

James E. Korenthal
New York, NY

Tired of coding all those IF statements to separate a value returned by the STICK function into its X and Y components? This short assembler routine will do the trick, and give you faster executing, more readable code to boot!

In order to understand how the routine works, we have to look at values returned by STICK in binary notation:

Direction	Decimal	Binary E W S N	X-val (see below)	Y-val
none	15	1 1 1 1	1	1
N	14	1 1 1 0	1	0
NE	6	0 1 1 0	2	0
E	7	0 1 1 1	2	1
SE	5	0 1 0 1	2	2
S	13	1 1 0 1	1	2
SW	9	1 0 0 1	0	2
W	11	1 0 1 1	0	1
NW	10	1 0 1 0	0	0

A glance at this table reveals that each direction is handled by one bit, with a zero value indicating that the joystick is pointed in that direction. The "E W S N" subheading in the binary column indicates the bit-direction correspondence. For example, a value of 9 is 1001 in binary, with 0's in the bits corresponding to south and west.

Now, how can we make use of this information? We can split up the east-west and north-south groups of two bits, and transform them into delta-x ("change in x") and delta-y values. Here's the assembler program:

```
STICKO  = $0278                ;address of joystick 0 value
ANSLO   = $D4                  ;low byte of answer
ANSHI   = $D5                  ;high byte of answer
;
        * =     $0600          ;dummy origin (code is
                                relocatable)
        PLA                    ;discard number of argu-
                                ments
        PLA                    ;discard high byte of
                                stick number
        PLA                    ;get stick number (0-3)
        TAX                    ;use it as an index
        PLA                    ;discard high byte of
                                direction
        LDA     STICKO,X       ;get value for appropri-
                                ate direction
        PLP                    ;get direction indicator
                                in carry
        BCS     STICKY         ;does he want x (0) or
                                y (1)?
        LSR     A              ;he wants x, so shift down
                                2 bytes
        LSR     A
STICKY  AND     #3             ;mask off high bits
        SEC                    ;set carry for subtract
        SBC     #2             ;change 2 to 0, 3 to 1, 1 to -1
        BPL     SAVEIT         ;was it 1?
        LDA     #2             ;transform to 2 if so
SAVEIT  STA     ANSLO          ;save low byte of result
        LDA     #0             ;zap high byte
        STA     ANSHI
        RTS                    ;all done
        .END
```

Given a joystick number (0-3) and a direction (0 for X or horizontal or east-west and 1 for Y or vertical or north-south), this routine returns, for the correct joystick, a value shown the X-val or Y-val column of our table. You should verify that subtracting one from these values yields appropriate deltas (-1, 0, or 1) for screen positions. If standard deltas are desired for the y-axis, reverse the value (subtract it from one instead of the other way around).

The following demonstration program shows how to incorporate the joystick driver into your own BASIC programs. It simply monitors the status of all four joysticks, displaying standard deltas in x,y format:

```
10 REM Atari 800 Joystick Driver Demonstration
   Program
20 REM James E. Korenthal, 1981
30 GOSUB 1000 :REM load machine language code
40 X=0 : Y=1 :REM direction codes (for read-
   ability)
50 POKE 201,8 :REM use narrow columns
60 FOR NSTICK=0 to 3 :REM loop on all three
   sticks
70 DELTAX = USR(JOY,NSTICK,X)-1 :REM get
   horizontal change
80 DELTAY = 1-USR(JOY,NSTICK,Y) :REM get
   vertical change
90 PRINT DELTAX;",";DELTAY, :REM print
   values
100 NEXT NSTICK :REM end loop on sticks 0-3
110 PRINT :REM skip to next line
120 GOTO 60 :REM loop until break or reset is
    pressed
1000 REM subroutine to load joystick driver machine
     code
1010 DIM JOY$(29) : JOY=ADR(JOY$) :REM set up
     code & pointer
1020 FOR J=JOY to JOY+28 :REM read and store
     29 bytes
1030 READ BYTE : POKE J,BYTE : NEXT J :
     RETURN
1040 REM machine code goes here:
1050 DATA 104,104,104,170,104,189,120,2,40,176,2,
     74,74,41
1060 DATA 3,56,233,2,16,2,169,2,133,212,169,0,133,
     213,96
```

As an interesting exercise, try incorporating the joystick driver into Larry Isaacs' speed-up of Chris Crawford's Player-Missile Demo (Listing 2 on page 108, **COMPUTE!** Vol. 3, No. 4). Make sure you allow the ship to move in all directions (as is, the program only allows north, south, east, or west), and watch out for that RESTORE in line 1140! ©

Atari 800:

Assembly Language Routine To Eliminate DOS/FMS When They Are No Longer Needed

John Elliott
New York, NY

As all users of VERSION I of DOS will know, about 9K of RAM is taken up by the DOS/FMS routines. More than 4K of this RAM is needed only when you want to talk directly to DOS through the menu selection screen.

The DOS REFERENCE MANUAL describes a method (1) of releasing the RAM used for these functions. However, the BASIC program listed in the manual will run only when the BASIC cartridge is inserted. This presents a problem for those of us who are using the ASSEMBLER/EDITOR cartridge. How can we get the same RAM savings as our BASIC counterparts? By using assembler, of course!

The short assembly language program listed here will eliminate DOS and FMS when either the BASIC or ASSEMBLER/EDITOR cartridge is inserted. You can still use all the DOS functions that are controllable with BASIC and the ASSEMBLER /EDITOR keywords.

The routine is designed to be as general-purpose as possible, so I will describe an operational procedure which is independent of the cartridge being used. In fact, once the program is saved on diskette, it can be executed with no cartridge inserted.

Saving The Program To Diskette

Our first task is to get the object program into RAM. If you have the ASSEMBLER/EDITOR cartridge, use it to assemble the program, with the object program going to RAM (this is the assembler default). If you do not have the ASSEMBLER/ EDITOR cartridge, but you want to make use of the program, then you can use BASIC to POKE the program into RAM.

Our next task is to save the object program on diskette. Go to the DOS menu selection, by typing DOS (RETURN). Then follow the "BINARY SAVE" procedure described in the DOS manual (2). The session should proceed as follows:

SELECT ITEM
K (RETURN)
SAVE—GIVE FILE, START, END
DISOUT. OBJ, 600, 626 (RETURN)
SELECT ITEM

This SELECT ITEM prompt indicates that the program has been saved. Note that the file name is arbitrary.

Executing The Program

Now that we have the permanent copy on disk, we can load it into RAM and use it as and when we need it. To load it into RAM, go to the DOS menu selection, by typing DOS (RETURN). Then follow the "BINARY LOAD" procedure described in the DOS manual (3). The session should proceed as follows:

SELECT ITEM
L (RETURN)
LOAD FROM WHAT FILE?
DOSOUT. OBJ (RETURN)
SELECT ITEM

This SELECT ITEM prompt indicates that the program has been loaded into RAM. To execute it, follow the "RUN AT ADDRESS" procedure described in the DOS manual (4). The session should proceed as follows:

SELECT ITEM
M (RETURN)
RUN FROM WHAT ADDRESS?
600 (RETURN)
READY/EDIT

The computer will respond with either READY or EDIT, depending on whether you have the BASIC or the ASSEMBLER/EDITOR cartridge inserted. DOS & FMS have now been eliminated.

I hope you find this routine useful. If you find that you rarely use DOS directly, you may like to make use of the AUTO. SYS feature to automate the routine. Good luck!

Notes On The Listing

The program follows fairly closely the steps taken by the BASIC program listed in the DOS manual (1). You may find it interesting to compare the two.

Lines 140-160 : Define the origin and initialize the stack pointer.
Lines 170-280 : Adjust DOS/OS vectors.
Line 290 : Link to DOS.
Lines 300-320 : Jump to the cartridge initialization routine.

I have used PAGE 6 to hold the object program. However, the routine is relocatable, so you may locate it somewhere else in RAM if you need locations \$600-\$626 for some other purpose.

As the comment at line 120 states, eliminating DOS & FMS releases 5200 bytes of RAM. The DOS REFERENCE MANUAL (1) states that the 5384 bytes of RAM are released, but this includes the length of the BASIC routine — 184 bytes.

References.

(1) ATARI DISK OPERATING SYSTEM REFERENCE MANUAL, C015200 rev.1, appendix C.3, page 58.
(2) Ibid. Page 36.
(3) Ibid. Page 38.

(4) Ibid. Page 38.

```
LOC  OBJECT   LINE  SOURCE STATEMENT

              0100 ;ROUTINE TO ELIMINATE
              0110 ;      DOS & FMS
              0120 ;RELEASES 5200 BYTES
              0130 ;
0000          0140       *=    $600
0600 A2FF     0150       LDX   #$FF
0602 9A       0160       TXS
0603 A923     0170       LDA   #$23
0605 850A     0180       STA   $0A
0607 A9F2     0190       LDA   #$F2
0609 850B     0200       STA   $0B
060B A988     0210       LDA   #$88
060D 850C     0220       STA   $0C
060F A907     0230       LDA   #$07
0611 850D     0240       STA   $0D
0613 A930     0250       LDA   #$30
0615 8D0C07   0260       STA   $70C
0618 A912     0270       LDA   #$12
061A 8D0D07   0280       STA   $70D
061D 208807   0290       JSR   $788
0620 A900     0300       LDA   #$00
0622 8508     0310       STA   $08
0624 6CFABF   0320       JMP   ($BFFA)
0627          0330       .END
```

©

Atari Data

Robert W. Baker
Atco, NJ

While recently converting a simple program from the PET to run on the Atari, I came across a few new quirks in Atari BASIC that I believe have not yet been documented. The problems have to do with using strings in DATA statements on the Atari. There is a vast difference in the ways Microsoft and Atari handle this.

It appears that Atari uses commas to separate DATA elements regardless of where they appear. Even if a string is enclosed in quotes, commas are still recognized and create separate data elements. Adding quotes actually creates another problem since they are not optional. Any quotes in the DATA statement will actually be read as part of the string data. Here's a simple program that will quickly illustrate how the DATA statement works on the Atari:

```
10 TRAP 70
20 DIM A$ (40)
30 DATA "TEST, WITH, COMMAS"
40 READ A$
50 PRINT A$
60 GOTO 40
70 END
```

When you run this program you'll see:

```
"TEST
WITH
COMMAS"
```

These three lines show that the commas are still recognized and actually create three separate data elements instead of one. Also, notice that the quotes are still part of the data as are the spaces after the commas. Thus, whenever placing strings within DATA statements on the Atari, you cannot have commas as part of the data. Also, there's no real reason to use quotes unless they're actually wanted in the data. You do not need quotes at all, even when there are spaces within the string constant.

While I'm at it, here's a copy of the program I converted for the Atari. It's a program that I use to record birthdays, anniversaries, and other important dates. The program allows you to display or print the recorded dates for any month, or the entire list. It has an option to supress special dates unless specifically requested.

The information for any date is stored in separate DATA statements. The first five characters are the actual date in the form of "MM/DD". This is followed by two spaces and any specific data associated with that date. Special dates are identified by an asterisk as the first character in the data for that date (see program line 1000). The last DATA entry must be the word "END" to terminate the list correctly.

For convenience, I normally use a separate line for each DATA statement constructed from the date itself. This makes the line very easy to locate and avoids duplication. Typically I make the month the thousands digits (1000-12000) and the day of the month the hundreds and tens digits (010-310). This leaves the ones digits for multiple events on the same date, up to 10 maximum of course.

Data recorded in the program is not sorted or re-ordered in any way. Dates are listed in the exact order found in reading the DATA statements, so they should be stored in the correct order. Whenever you update the program remember to save a new copy. For added convenience, you may want to also include the date of the last update as well (see program line 50). Just remember to avoid using commas in the data strings as discussed earlier!

```
10 REM ----------------------
20 REM DATE BOOK - FOR ATARI
30 REM BY: ROBERT BAKER
40 REM ----------------------
50 REM LAST UPDATE:  MM/DD/YY
60 REM ----------------------
65 OPEN #1,4,0,"K"
67 DIM R$(40),M$(2)
70 PRINT CHR$(125);"*** DATE BOOK *** IM
PORTANT DATES ***"
80 PRINT :PRINT
90 PRINT "DISPLAY MONTH"
95 PRINT "(1-12, A=ALL, S=SPCL)";
100 INPUT M$:IF M$="" THEN 800
105 IF M$<"A" THEN M=VAL(M$)
110 IF M$="A" OR M$="S" THEN 200
120 IF M<1 OR M>12 THEN 800
200 PRINT :PRINT
210 P=0:PRINT "WANT PRINTED COPY (Y/N)";
220 INPUT R$:IF R$="Y" THEN P=1
300 PRINT CHR$(125);:RESTORE :L=0
310 READ R$
320 IF R$="END" THEN GOSUB 900:GOTO 70
330 IF R$(8,8)<>"*" THEN 400
340 IF M$="S" THEN PRINT R$:L=L+1:IF P=1
 THEN LPRINT R$
350 GOTO 500
400 IF M$="A" OR VAL(R$(1,2))=M THEN PRI
NT R$:L=L+1:IF P=1 THEN LPRINT R$
500 IF L=20 THEN GOSUB 900:L=0
510 GOTO 310
800 CLOSE #1:END
900 PRINT :PRINT "PRESS ANY KEY TO CONTI
NUE";
920 GET #1,X
950 PRINT CHR$(125);:RETURN
1000 DATA 01/01  *D* GUESS WHO
12250 DATA 12/25  JANE DOE (1925)
12300 DATA 12/30  ME TWO (1950)
32000 DATA END
```

©

Memory Protection For Atari

Jim Clark
Seattle, WA

```
10 REM MEMPROT.BAS
20 REM by Jim Clark
30 REM 2415 East McGraw St.
40 REM Seattle, WA 98112
100 REM Load assembly language subroutine
110 PGMSIZ=24:DIM SUBR$(PGMSIZ)
120 FOR I=1 TO PGMSIZ
130 READ BYTE
140 SUBR$(I)=CHR$(BYTE)
150 NEXT I
200 REM Get amount of memory to protect
210 PRINT "How many bytes do you want to protect";
220 INPUT PROTECT
230 HI=INT(PROTECT/256):LOW=PROTECT-256*HI
240 SUBR$(6,6)=CHR$(LOW)
250 SUBR$(14,14)=CHR$(HI)
300 REM Reinitialize BASIC with the new low memory pointer
310 Z=USR(ADR(SUBR$))
400 REM Assembly language subroutine
410 REM MEMLO =$02E7 ;BOTTOM OF AVAILABLE USER MEMORY
420 REM WARMST=$08   ;WARM START FLAG
430 REM CARTA =$A000 ;BASIC CARTRIDGE ENTRY POINT
440 REM
450 REM ;THE PROGRAM IS COMPLETELY RELOCATABLE,
460 REM ;SO NO STARTING ADDRESS IS PROVIDED
470 REM
500 REM CLC                      ;INITIALIZE FOR ADDITION
510 DATA 24
520 REM LDA MEMLO                ;ADD LEAST-SIGNIFICANT BYTES
530 DATA 173,231,2
540 REM ADC #PROTECT&$FF
550 DATA 105,0
560 REM STA MEMLO
570 DATA 141,231,2
580 REM LDA MEMLO+1              ;ADD MOST-SIGNIFICANT BYTES
590 DATA 173,232,2
600 REM ADC #PROTECT/256
610 DATA 105,0
620 REM STA MEMLO+1
630 DATA 141,232,2
640 REM LDA #0                   ;RESET THE WARM START FLAG
650 DATA 169,0
660 REM STA WARMST
670 DATA 133,8
680 REM JMP CARTA                ;START BASIC OVER AGAIN
690 DATA 76,0,160
999 END
```

A problem arises in applications which require that a portion of memory be protected from BASIC on the Atari Computer. For example, most assembly language subroutines need protection. The problem is that BASIC is likely to use memory anywhere within available RAM, thus writing over the assembly language subroutine and destroying it.

In many computers it is possible to protect memory at the "high" end, i.e., at the highest RAM address. The Atari uses high memory for the data which is displayed on the screen. If you attempt to protect memory above the screen display by reducing the high memory value that BASIC thinks it has, then you cannot clear the screen or scroll text in any of the split-screen modes because these actions affect memory *beyond* the screen display area. These actions cause no problem when the screen display is actually the last thing in memory, because they apply to nonexistent memory. However, if you want to use memory beyond the screen display for your own purposes, then your data will be damaged by any action in your program which clears the screen or scrolls text in a text window.

Another alternative is to protect low memory. The main problem with this approach is that the memory protection must be done before BASIC gets control, since BASIC starts saving any program you enter beginning at the low memory address. The program shown here solves this problem as follows: it takes control of the Atari in an assembly language subroutine and resets the system's low memory pointer. It then reinitializes BASIC — just as if you had pressed the SYSTEM RESET key — and BASIC takes control again, blissfully unaware that it now has less RAM to work with than it did before you ran this program.

To find the address of the memory you have protected, type ?PEEK(743) + 256*PEEK(744) *before* you run this program. The number printed can be used as the origin for an assembly language subroutine, or as the destination address for whatever data you want to store in the protected area.

When you run the program, it asks how much memory you want to protect. Type in any positive number which is less than the amount of RAM you have available, as determined by typing ?FRE(0). The program reinitializes BASIC, and if you type ?PEEK(743) + 256*PEEK(744) again, the number printed will be greater than the value shown before running the program: the difference is the amount you requested to be protected. The memory area will remain protected until you turn the computer off, and the area can be used for assembly language subroutines, redefined character sets, player-missile graphic objects, or any other use you might wish. ©

Program Development In Atari Basic

Paul E. Hoffman
Cambridge, MA

The Atari microcomputers are good machines on which to program, especially in Basic. The editor is easy to use and has many nice features. Unfortunately, developing large programs on the Atari is not as easy as one would like due to the lack of dynamic program chaining.

Optimally, you should be able to make many program fragments and be able to run a linking program that would look like:

10 ENTER 'D:PROG1'
20 ENTER 'D:PROG2'
30 ENTER 'D:PROG3'

and so on, that would bring all of the programs together. The problem is that the Atari operating system stops the program right after an ENTER, so that a running program can only have one ENTER command.

The way around this restriction is to use the screen as a device to give the ENTER commands to link the large program. Of course, you can simply type the commands in each time they want to link the programs. It is easier, however, to have the Atari display the ENTER commands; all you have to do is RETURN over them.

Breaking up a longer program into modules that are about two screens long makes debugging programs much easier. Small program fragments can be listed completely without having to guess which line range you want. Also, the fragments can be broken up into logical units of the program, such as the variable definitions, screen setup, and so forth.

Developing a program using fragments has some drawbacks, but they are easy to overcome. You must be careful about using GOTO's across segments, since you may decide to renumber some lines in the segment to which you are jumping. This problem can be avoided by using as few GOTO's as possible, and by always going to REM statements that tell you which lines to change if the line you are going to needs to be renumbered.

Subroutines can be treated the same as statements that are gone to, but there is a more efficient way to deal with them. GOSUB's that point to the beginning of a program are found by the program more quickly than those that point to the end, so putting all of your subroutines in the first module of a program will make your program run faster. If all you use the first module for is to define constants and subroutines, it is unlikely that you will need to renumber much, thus avoiding the problems with GOTO's.

There are two other restrictions for using this program. The first is that you can only have up to ten modules, due to the amount of space each ENTER takes on the screen. Second, if you renumber your program using a Basic renumbering program available from some third-party vendors, you should only renumber the full program to avoid incorrect GOTO's.

The program in Listing 1 can be used to display the ENTER commands on the screen to link together 10 programs called PROG1, PROG2, ... You can change the name of the program in lines 30 and 60 to any name you want. Once the program has been run, simply hit RETURN's over each line, and your program will be linked and in memory.

Line 10 clears the screen; lines 20 through 50 display the first five ENTER's (for PROG1 through PROG5). The CHR$(34) is a quote mark; using CHR$(n) instead of the character itself in program listings makes them easier to read and reproduce.

Lines 60 and 70 write a one-line program that clears the screen and displays the next five enters (for PROG6 through PROG10). If you have five or less program fragments to link, you can eliminate lines 50 through 70. Line 80 puts the cursor over the first ENTER (after the READY prompt appears), then line 90 clears the linking program from memory.

Using the above scheme for writing large programs should make them easier to edit and develop. We can always hope that future Atari versions of Basic (as well as other programming languages) give more flexibility in program linking so that programs like this one are not necessary.

```
10 ? CHR$(125):REM CLEAR THE SCREEN
20 FOR I=1 TO 5
30 ? :? :? "ENTER ";CHR$(34);"D:PROG";I;CHR$(34)
40 NEXT I
50 ? :?
60 ? "POKE84,1:POKE85,0:FORI=6TO10:?:?:?";CHR$(34);"ENTER ";CHR$(34);";CHR$(34);
";CHR$(34);"D:PROG";CHR$(34);
70 ? ";I;CHR$(34):NEXT I:POKE 84,1:POKE 85,0"
80 POKE 84,1:POKE 85,0
90 NEW
```

©

THE OSI® GAZETTE

OSI C1P Newspaper Route Listing Program Part Two

Charles L. Stanford
Cinnaminson, NJ

Running The Program

With all the preambles and caveats out of the way, let's take a look at the program. Line 0 through whatever contain the customer list. I found it easiest to enter the first time directly as DATA Statements. We devised a form (see Listing 2), and John filled it out from his collection cards and entered the data into the program over a period of several days. Note that Line 0 contains the number of customers. We vary that to match the route, but keep the number of DATA lines at 75. The program doesn't care as long as there are more DATA's than READ's. Be very careful, however, that every line has an identical number of characters.

When printing to the screen or to an external device, Line 425 can be used to select the format of the list. If you want only the Sunday route, use a line such as IF D(X) <> 1 AND D(X) <> 3 AND D(X) <> 6 THEN GOTO 460. Note that AND and not OR is used for <> IFs.

Listing 2

```
10 INPUT"NUMBER OF CUSTOMERS";N
20 PRINTTAB(12);"NAME";
30 PRINTTAB(25);"No.";TAB(35);"ST.";
40 PRINTTAB(41);"D";TAB(45);"P"
50 FORX=1TON
60 PRINTRIGHT$("  "+STR$(X),2);
70 PRINT"DATA";
80 PRINT": : : : : : : : :,";
90 PRINT": : : :,: : :,: :,: :"
100 NEXT
OK

OK
RUN
NUMBER OF CUSTOMERS? 5
           NAME            No.       ST.   D   P
 1DATA: : : : : : : : :,: : : : :,: : :,: :,: :
 2DATA: : : : : : : : :,: : : : :,: : :,: :,: :
 3DATA: : : : : : : : :,: : : : :,: : :,: :,: :
 4DATA: : : : : : : : :,: : : : :,: : :,: :,: :
 5DATA: : : : : : : : :,: : : : :,: : :,: :,: :
OK
```

The BASIC Garbage Collector Bug

There is one more problem involved in getting this program to run successfully. That's the Garbage Collection Bug in BASIC ROM Number 3. Most OSI owners probably already know about this problem, but as an assist for those who are new to it, a short background. When the ROMs were programmed, there were two code errors within the Garbage Collector Routine starting at $B147 in ROM 3. The errors don't stop the routine from running, but they sure keep it from doing anything useful. The GC is necessary to undo the damage to memory caused by the accumulation of strings in upper RAM. When a string is concatenated, or even recreated, all versions are retained. If you say that A$ = A$ + A$, both versions stay in existence. You can see that an active routine such as the one at Line 900 of Listing 1 will soon use up all the RAM. But the GC is automatically called whenever RAM gets short, and all the redundant strings are discarded. On the OSI, not only doesn't this happen, the whole program hangs up, the screen pulses, and only a quick "BREAK" can save source code destruction.

There are several solutions. One is to buy one of the corrected PROMs available from several sources. But Rodger Olsen of Aardvark advised me that even a repaired OSI GC isn't perfect. He suggests the software fix in **The (Real) First Book of OSI**. I just ordered my copy, so I can't comment. What I have done is include a fix I devised, which puts a repaired OSI GC in Page 2 of RAM (the unused part starting at $0222). It is shown in Listing 3. You must remember that a BREAK will require that the Vector at locations 11 and 12 (Dec) be reset. The GC will not fit between $0222 and $02FF, so it

uses part of the original code as subroutines. There are several other fixes available, but one or the other must be used or you'll have to omit the names, and reset the string pointer at 129 and 130 (Dec) each time a loop of the Routine at Line 900 is run.

As is often the case, this program concept can be extended into many other areas. How about a checkbook balancer with the purpose of each check printed out along with its number and amount? A Christmas card checklist? The names can be any length consistent with your RAM; just change the "8" spaces wherever they appear. Routines to add values of numeric variables can be easily added.

Listing 1A

```
Ø DATA75
1 DATA"STANFORD",29Ø3, 1,1,1
2 DATA"JONES    ",    2,12,2,3
3 DATA"SMITH    ", 321, 2,1,6
4 REM-LINES 4 THRU 73 OMITTED
74 DATA"        ",    Ø, Ø,Ø,Ø
75 DATA"        ",    Ø, Ø,Ø,Ø
```

Table 1

```
1Ø DIMN$(2):DATA"A","BBB"
2Ø X=Ø:Y1=3:Z=487
3Ø READN$(1):READN$(2)

Ø3ØØ ØØ  START SOURCE       Ø33E 58  ASCII - X
         CODE                  3F ØØ  2ND CHAR
  Ø1 17  ADDRESS OF            4Ø ØØ  VALUE OF X
  Ø2 Ø3   NEXT LINE            41 ØØ   IN 32-BIT
  Ø3 ØA  LINE                  42 ØØ   FLOATING POINT
  Ø4 ØØ   NUMBER               43 ØØ   BINARY
  Ø5 85  TOKEN - DIM           44 59  ASCII - Y
  Ø6 4E  ASCII - N             45 31        - 1
  Ø7 2A        - $             46 82  VALUE OF Y1
  Ø8 28        - (             47 4Ø
  Ø9 32        - 2             48 ØØ
  ØA 29        - )             49 ØØ
  ØB 3A        - :             4A 5A  ASCII - Z
  ØC 83  TOKEN - DATA          4B ØØ  2ND CHAR
  ØD 22  ASCII - "             4C 89  VALUE OF Z
  ØE 41        - A             4D 73
  ØF 22        - "             4E 8Ø
  1Ø 2C        - ,             4F ØØ
  11 22        - "             5Ø 4E  ASCII - N
  12 42        - B             51 8Ø  TOKEN - $TRING
  13 42        - B             52 13  7+(DIM+1)*4
  14 42        - B             53 ØØ  ALWAYS ØØ
  15 22        - "             54 Ø1  ALWAYS Ø1
  16 ØØ  END OF LINE           55 ØØ  ALWAYS ØØ
  17 2A  ADDRESS OF            56 Ø3  DIM + 1
  18 Ø3   NEXT LINE            57 ØØ  FOUR BYTES TO
  19 14  LINE                  58 ØØ   EACH ELEMENT OF
  1A ØØ   NUMBER               59 ØØ   AN ARRAY; START
  1B 58  ASCII - X             5A ØØ   WITH ELEMENT Ø
  1C AB  TOKEN - =             5B Ø1  LEN OF $TRING
  1D 3Ø  ASCII - Ø             5C ØE  ADDRESS OF
  1E 3A        - :             5D Ø3   $TRING
  1F 59        - Y             5E ØØ  ALWAYS ØØ
  2Ø 31        - 1             5F Ø3  LEN OF $TRING
  21 AB  TOKEN - =             6Ø 12  ADDRESS OF
  22 33  ASCII - 3             61 Ø3   $TRING
  23 3A          :             62 ØØ  ALWAYS ØØ
  24 5A          Z           Ø363  UNUSED RAM
  25 AB  TOKEN - =
  26 34  ASCII - 4
  27 38        - 8
  28 37        - 7
  29 ØØ  END OF LINE
Ø32A THROUGH Ø33B
      HOLDS LINE #3Ø
Ø33C ØØ  END OF LINE
  3D ØØ  CODE FOR
  3E ØØ  "LAST LINE"
```

Sample Run-Entire List

```
                JOHN'S INQUIRER ROUTE
                ------ -------- -----

1. STANFORD   29Ø3 GEORGETOWN RD    DAILY & SUN   GARAGE
2. JONES         2 BARTON CT        DAILY ONLY    REAR DOOR
3. SMITH       321 BRANCH PIKE      DAILY & SUN   UNDER RUG
4.               Ø
5.               Ø
```

Sample Run-Sunday Customers Only

```
                JOHN'S INQUIRER ROUTE
                ------ -------- -----

1. STANFORD   29Ø3 GEORGETOWN RD    DAILY & SUN   GARAGE
3. SMITH       321 BRANCH PIKE      DAILY & SUN   UNDER RUG
```

Listing 1B

```
 2ØØ READX:N=X:DIMN$(X),A(X),S(X),D(X)
 2Ø1 DIMP(X),S1$(15)
 2Ø2 GOSUB7ØØ
 2Ø5 FORX=1TO9:PRINT:NEXT
 21Ø PRINT"Newspaper Route":PRINT
 215 PRINT"1.  Print the route to printe
r":PRINT
 22Ø PRINT"2.  Print the route to screen
":PRINT
 225 PRINT"3.  Add a customer":PRINT
 23Ø PRINT"4.  Delete a customer":PRINT
 24Ø INPUT"Enter your preference";X2
 245 ONX2GOSUB3ØØ,4ØØ,5ØØ,6ØØ
 25Ø GOTO2Ø5
 3ØØ REM-PRINT TO PRINTER
 31Ø POKE517,1
 32Ø GOSUB42Ø
 33Ø POKE517,Ø
 34Ø RETURN
 4ØØ REM-PRINT ROUTE TO SCREEN
 4Ø5 PRINT:PRINT:PRINT:PRINT
 41Ø PRINTTAB(15);"John's Inquirer Route
 415 PRINTTAB(15);"------ -------- -----
 42Ø PRINT:PRINT:FORX=1TON
 425 REM-SELECT LIST ON THIS LINE
 43Ø PRINTRIGHT$("  "+STR$(X),2);".";
 435 PRINTTAB(4);N$(X);TAB(14);RIGHT$("
   "+STR$(A(X)),4);
 44Ø PRINTTAB(19);S1$(S(X));
 445 PRINTTAB(35);D1$(D(X));
 45Ø PRINTTAB(48);P1$(P(X))
 46Ø X=USR(X)
 47Ø NEXTX
 48Ø RETURN
 5ØØ REM
 51Ø PRINT"          Add a customer":PRINT
 515 INPUT"Enter customer's code number"
;X3:PRINT
 52Ø FORX=NTOX3+1STEP-1
 525 N$(X)=N$(X-1):A(X)=A(X-1):S(X)=S(X-
1):D(X)=D(X-1):P(X)=P(X-1)
 53Ø X=USR(X):NEXTX
 535 INPUT"Enter customer's name";N$(X3)
:PRINT
 54Ø INPUT"Enter house number";A(X3):PRI
NT
 545 INPUT"Enter street name code number
";S(X3):PRINT
 55Ø INPUT"Enter delivery code number";D
(X3):PRINT
 555 INPUT"Enter special location code n
umber";P(X3):PRINT
 56Ø N$(X3)=LEFT$(N$(X3)+"        ",8)
 565 PRINTN$(X3);A(X3);S1$(S(X3));D1$(D(
X3));P1$(P(X3))
 575 X=USR(X)
 58Ø INPUT"Add another";I$
 585 IFLEFT$(I$,1)="Y"THEN5ØØ
 59Ø GOTO8ØØ
 6ØØ REM-DELETE
 61Ø PRINT:PRINT:PRINT:PRINT
 62Ø INPUT"Enter customer's code number"
;X4
 63Ø FORX=X4TON-1
 64Ø N$(X)=N$(X+1):A(X)=A(X+1):S(X)=S(X+
1):D(X)=D(X+1):P(X)=P(X+1)
 645 X=USR(X)
 65Ø NEXTX
 66Ø N$(N)="        "
 67Ø A(N)=Ø:S(N)=Ø:D(N)=Ø:P(N)=Ø
 68Ø INPUT"Delete another";I$
 685 IFLEFT$(I$,1)="Y"THEN6ØØ
 69Ø GOTO85Ø
 7ØØ REM-STARTUP SEQUENCE
 71Ø FORX=1TON
 72Ø READN$(X):READA(X):READS(X):READD(X
):READP(X)
 73Ø NEXTX
 75Ø S1$(1)="Georgetown Rd":S1$(2)="Bran
ch Pike":S1$(3)="Essex Ct"
 755 S1$(4)="Somerset Dr":S1$(5)="Bergen
Dr":S1$(6)="Salem Dr"
 76Ø S1$(7)="Brigham Ct":S1$(8)="Salem C
t":S1$(9)="Cooper Ct"
 765 S1$(1Ø)="Hunterdon Dr":S1$(11)="Riv
erton Rd":S1$(12)="Barton Ct"
 768 S1$(13)="Carriage Way":S1$(14)="Mid
dlesex Dr"
 775 D1$(1)="Daily & Sun":D1$(2)="Daily
only":D1$(3)="Sun only"
 78Ø D1$(4)="Sat only":D1$(5)="M-F only"
:D1$(6)="Sat-Sun only"
 79Ø P1$(1)="Garage":P1$(2)="Carport":P1
$(3)="Rear door"
 795 P1$(4)="In door":P1$(5)="Mail box":
P1$(6)="Under rug"
 799 RETURN
 8ØØ REM-SAVE DATA - ADD CUSTOMERS
 8Ø5 B=782
 81Ø FORX=NTO1STEP-1
 815 B=782+(X-1)*28
 82Ø GOSUB9ØØ:X=USR(X)
 825 NEXTX
 83Ø CLEAR:GOTO2ØØ
 85Ø REM-SAVE DATA - DELETE CUSTOMERS
 855 B=782
 86Ø FORX=1TON
 865 GOSUB9ØØ
 87Ø X=USR(X)
 875 B=B+6:NEXTX
 88Ø CLEAR:GOTO2ØØ
 9ØØ REM-SAVE DATA
 9Ø5 L$=LEFT$(N$(X)+"        ",8)
 91Ø Q=8:POKEB,34:B=B+1:GOSUB995:POKEB,3
4:B=B+1
 915 B=B+1:L$=RIGHT$("    "+STR$(A(X)),4
)
 92Ø Q=4:GOSUB995
 925 B=B+1:L$=RIGHT$("  "+STR$(S(X)),2)
 93Ø Q=2:GOSUB995
 935 B=B+1:L$=RIGHT$(" "+STR$(D(X)),1)
 94Ø Q=1:GOSUB995
 945 B=B+1:L$=RIGHT$(" "+STR$(P(X)),1)
 95Ø Q=1:GOSUB995
 955 RETURN
 995 FORR=1TOQ:POKEB,ASC(MID$(L$,R,1)):B
=B+1:NEXTR:RETURN
 999 END
OK
```

Listing 3

```
41ØØØ REM-GARBAGE COLLECTION FIX
```

```
41010 POKE11,34:POKE12,2
41020 FORX=0TO139
41030 Y=PEEK(45383+X):POKE546+X,Y
41040 NEXTX
41050 FORX=0TO46
41060 Y=PEEK(45596+X):POKE696+X,Y
41070 NEXTX
41080 POKE613,4:POKE699,2:POKE700,24
41090 POKE629,177:POKE630,2
41100 POKE686,76:POKE687,211:POKE688,177
41110 POKE689,166:POKE690,157
41120 POKE691,208:POKE692,3
41130 POKE693,76:POKE694,19:POKE695,178
41140 POKE743,38:POKE744,2
OK
```

©

Voracious Butterfly

John Wright
Ottawa, Canada

The name came after seeing the program run. Voracious Butterfly was originally just a first exercise in using peeks, pokes and graphics, and as a visual check on how random is RND.

Display

A 24 x 24 section of the screen is filled with G187, the mini chequerboard, and G43, +, moves around one square at a time in a random direction. Each time it lands on a new square it 'eats' the G187 and replaces it with a G32 (Blank), G42 (*) or a character. When all the characters are displayed, the end routine pokes in another word and strips out the remaining G187s. A counter at the bottom of the screen increments by 100 every 100 steps.

Program

The program has 6 modules and a main line. The subroutines at 300, 500 and 1100 are called once each and could have been written in the main line. Conversely L70 to L150 could have been another module.

SUB 250 converts from X, Y coordinates to a POKE address.

SUB 300 to 480 reads in the word which is used in the end routine, puts 32 in all locations of the MA matrix to POKE blanks, replaces some of those 32s by 42s to sprinkle stars in the top and bottom thirds of the screen, and zeros counters.

SUB 500 to 560 reads character data into MA and puts a 1 in MB corresponding to each character in MA.

L70 to 155 picks the start point for the Butterfly and POKEs two zeros for the counter.

SUB 800 to 960 takes 100 steps. On each step the contents of MA are poked to the screen location, the contents of MB are added to TT (MB is 0 unless there is a display character in which case MB (X,Y) is 1. It is then reset to 0).

L840 checks conditions for a normal exit.

L860, 870 give the next step in the walk, with equal probability of staying still or moving to any of the eight adjacent squares.

L900, 910 stop the Butterfly from going off screen. Using SGN allows it to be done with one statement instead of separate IFs for 0 and 25. If the Butterfly goes off left, it reappears right as though there is a wrap-around. Similarly for top and bottom.

SUB 1000 to 1090 adjusts the base of the random number by incrementing the original input. This

was done to disturb any patterns. It also sets up an exit if the program runs too long and the Butterfly tires, and increments the display counter.

SUB 1100 to 1180 is the end routine. It POKES the top word and clears out the remaining G187s.

L170, 470, 530 and 840 could be changed to the handwritten version to make the display part easier to change.

Changing the Randomness

The original version did not have the routine at L1000, the Q loop at L800 and instead of wrap-around at L900 and 910 it had fold-back. This can be simulated by:

```
1030 REM DUMMY LINE
 900 IF X=0 OR X=25 THEN X=X-SX
 910 IF Y=0 OR Y=25 THEN Y=Y-SY
```

This version did not always work. On one occasion it ran for about 35 minutes and left two sizeable areas of the screen untouched.

Presumably the random number generator settled into a pattern, so to disturb this the 1000 routine was introduced, changing the base after 100 steps.

Variations

The listing includes many REM statements which can be omitted, and most statements are 1 to a line so compaction is possible.

Apart from 'wrap-around X and fold-back Y' (which I have left in my archive version) or vice-versa, the variations are in the display capability.

If you are on familiar terms with him,

```
 300 IB = 5
3000 82,79,78,78,89
```

may be acceptable.

If you change the main display data, you should use the handwritten version of L170, 470, 530 and 840 or recalculate. Remember that the display goes in bottom first, top last.

There is no reason why the display should not be a picture or a pattern. In this case the 'top' word may be better placed at the bottom by:

```
1100 Y = 1
```

It would be easier to have the display data as characters and blanks. They would then be read in by string variables and converted.

```
  45 BL$ =" 24 spaces "
 300 READ A$
 305 FOR I=I TO LEN(A$):TW(I)=ASC(MID$
     (A$,I,1)):NEXT I
 505 READ A$:A$=LEFT$(A$+BL$,24)
 520 MA(X,Y)=ASC(MID$(A$,X,1))
3010 DATA " ABC etc.
```

Use the " in the data statement if there are leading blanks.

Side Benefits

The program is a good conditioner for a flabby waist. Judicious use of body English may guide the Butterfly to uneaten squares.

```
1 REM  VORACIOUS BUTTERFLY
10 REM  JOHN WRIGHT
15 DIM MA(24,24),MB(24,24)
20 REM 'RANDOM WALK' DEMONSTRATION
30 INPUT"RANDOM NUMBER";NR:RN=NR
40 OO=54116:REM THIS IS FOR 600 BOARD
50 GOSUB300
60 GOSUB500
70 REM  PICK START POINT
80 J=INT(570*RND(RN))+1
90 X=INT(J/24)+1
100 Y=J-24*(X-1)
110 GOSUB250
120 POKEZ,43
140 REM  PAUSE THEN REMOVE +
150 FOR J=1TO100:I=J:NEXTJ
155 POKEOO+6,48:POKEOO+7,48
160 GOSUB800
170 IFTT=61THEN200   IF TT=CH...
180 GOSUB1000
190 IFCT<>-1THEN160
200 GOSUB1100
210 IFCT=-1THENPRINT"EXHAUSTED BUTTERFLY"
220 END
247 REM
248 REM
249 REM  SCREEN POSITION FOR X,Y
250 Z=OO-32*Y+X
260 RETURN
296 REM
297 REM
298 REM  INPUT DISPLAY MATRIX
299 REM  TOP WORD
300 IB=6
305 FORI=1TOIB:READTW(I):NEXTI
310 REM  BLANK SCREEN,INPUT DISPLAY, SET UP
    COUNTER MATRIX
315 FORX=1TO30:PRINT:NEXTX
317 POKEOO+1,32:REM  ERASE CURSOR
320 FORX=1TO24
330 FORZ1=1TO3
340 FORZ2=1TO8
350 Y=8*(Z1-1)+Z2
360 GOSUB250
370 POKEZ,187
380 MA(X,Y)=32
390 REM  STARS AT TOP AND BOTTOM
400 IFZ1=2THEN420
410 IFRND(RN)<.15THENMA(X,Y)=42
420 MB(X,Y)=0
430 NEXTZ2,Z1,X
460 REM  ZERO COUNTERS
470 TT=0:CT=0 : CH=0
```

```
480 RETURN
497 REM
498 REM
499 REM READ IN DATA
500 FOR Y=11TO15
510 FORX=1TO23
520 READMA(X,Y)
530 IFMA(X,Y)<>32 THEN MB(X,Y)=1   1:CH=CH+1
540 NEXTX,Y
560 RETURN
797 REM
798 REM
799 REM  100 STEPS IN RANDOM WALK
800 FOR Q=1TO100
810 TT=TT+MB(X,Y)
820 MB(X,Y)=0
830 POKEZ,MA(X,Y)
840 IFTT=61THEN960    IF TT=CH...
850 REM  NEW PLACE FOR +
860 SX=INT(3*RND(RN))-1:REM  GIVES+1,0,-1
870 SY=INT(3*RND(RN))-1:REM  DITTO
880 X=X+SX
890 Y=Y+SY
900 IFX=0ORX=25THENX=X-24*SGN(SX):REM  TRY
    ...,THENX=X-SX
910 IFY=0ORY=25THENY=Y-24*SGN(SY):REM  TRY
    ...,THENY=Y-SY
920 GOSUB250
930 POKEZ,43
950 NEXTQ
960 RETURN
997 REM
998 REM
999 REM  ADJUST BASE OF RANDOM NUMBER
1000 CT=CT+1
1010 IFCT=100THENCT=-1:GOTO1070
1020 NR=NR+1
1030 RN=NR
1040 AC=INT(CT/10)+48
1050 IF AC=48THEN1070
1060 POKE00+4,AC
1070 AC=CT-10*AC+528
1080 POKE00+5,AC
1090 RETURN
1097 REM
1098 REM
1099 REM  END ROUTINE
1100 Y=17
1110 FORX=1TOIB
1120 GOSUB 250
1130 POKEZ,TW(X)
1140 NEXTX
1150 FORZ=00-776TO00
1160 IFPEEK(Z)=187THENPOKEZ,32
1170 NEXTZ
1180 RETURN
3000 DATA 82,79,78,65,76,68
3010 DATA 82,32,82,32,69,69,69,32,65,32,65,32
3020 DATA 32,71,71,32,65,32,65,32,78,32,78
3030 DATA 82,32,82,32,69,32,32,32,65,65,65,32
3040 DATA 71,32,71,32,65,65,65,32,78,78,78
3050 DATA 82,82,32,32,69,69,32,32,65,32,65,32
3060 DATA 71,32,32,32,65,32,65,32,78,78,78
3070 DATA 82,32,82,32,69,32,32,32,65,32,65,32
3080 DATA 71,32,71,32,65,32,65,32,78,78,78
3090 DATA 82,82,32,32,69,69,69,32,32,65,32,32
3100 DATA 32,71,32,32,32,65,32,32,78,32,78
OK
```

©

THE PET® GAZETTE

Saving Machine Language Programs On PET Tape Headers

Louis F. Sander
Pittsburgh, PA

This article describes a simple method of using your old ROM PET to write a brief machine language program (MLP) onto the *header* of any PET program tape, where it will automatically LOAD right along with the other program on the tape. The method described is the only way we know to use the first cassette buffer with a program loaded from TAPE #1, short of putting a program in there *after* the LOAD. The only restriction on the MLP is that it cannot exceed 171 bytes in length, but there's a way to expand this limit to 187.

After loading, the MLP will reside in the 1st cassette buffer, where it will not restrict the use of the other buffer or the user program area. Of course the MLP will be removed from memory, but not from the tape header, if the 1st cassette is subsequently used to LOAD, SAVE, or VERIFY, since those actions write into the 1st cassette buffer. For the same reasons, the MLP cannot be SAVEd again itself using normal procedures.

I have used this method to couple a joystick handler to a BASIC game program, and to load a tiny machine language monitor along with a program under development. It could easily be used in a program protection system, and the creative programmer can no doubt find many other uses for it. To see this wonder in action, first write a machine language program that is no longer than 171 bytes, and which will ultimately reside in locations 028F-0339 hex, (655-825 decimal), of your old ROM PET. Be careful, because you are writing a program in an area of memory that is wiped out whenever TAPE #1 is used. When you are ready to put your MLP onto the header of a program tape, use a machine language monitor, BASIC POKEs, or any other method to perform the steps below:

1. Put your MLP temporarily into locations starting at 034F, (847). If you run past location 03F9, (1017 decimal), your MLP is too long.

2. Put ASCII spaces into all memory locations between the end of your MLP and loaction 03F9, (1017), inclusive. The space character is a 20 hex, or a 32 decimal.

3. Using the appropriate ASCII codes for the name of the main program to be SAVEd, 16 characters maximum, put the program name into locations 033F-034E, (831-846 decimal). Fill any unused locations in this range with ASCII spaces, just as in step 2. Note that the name here is *not* the name of your new little MLP, but the name of the main program you'll be saving.

4. Using normal procedures, LOAD or key in the main program to be saved. You can change the program once it has loaded, for instance if you want to add some SYS calls to access your little MLP.

5. If your main program is in machine language, this step is required *unless* the main program was loaded from tape and still has the same starting and ending addresses: Put the main program's starting address, in lo byte, hi byte order, into 00F7-00F8 hex, and put its ending address plus one, in the same order, into 00E5-00E6.

6. In direct mode, POKE 249,63 : POKE 250,3 : POKE 238,187

7. PEEK to see that your POKES were successful and accurate.

8. Get the cassette you are going to SAVE onto, and put it into TAPE #1.

9. In direct mode, type SYS 63135 if your main program is in BASIC, or SYS 63153 if your main program is in machine language. When you hit RETURN, PET should initiate a SAVE, with the normal messages, but with some additional garbage after the program name. The garbage is your little MLP, and this is the last time you'll ever see it in this form.

10. VERIFY what you saved.

11. You can now LOAD and use this tape in the normal way. (Momentarily power down if you

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.

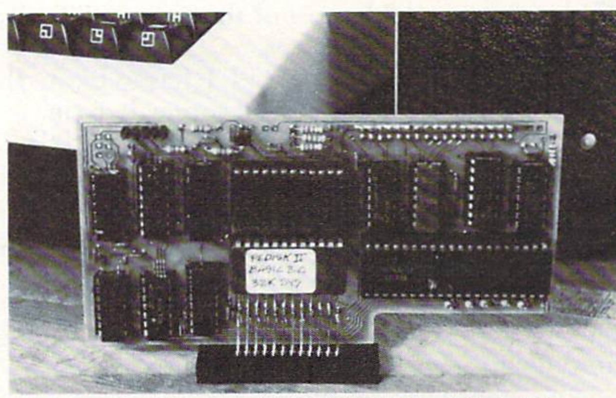
5¼″ system — 1 drive, double density (143K) $595.00
5¼″ system — 1 drive, quad density (286K) $795.00
8″ system — 1 drive, IBM 3740 business (295K) ...$1,295.00

2 drives, double density (286K) $945.00
2 drives, quad density (572K) $1,195.00
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.

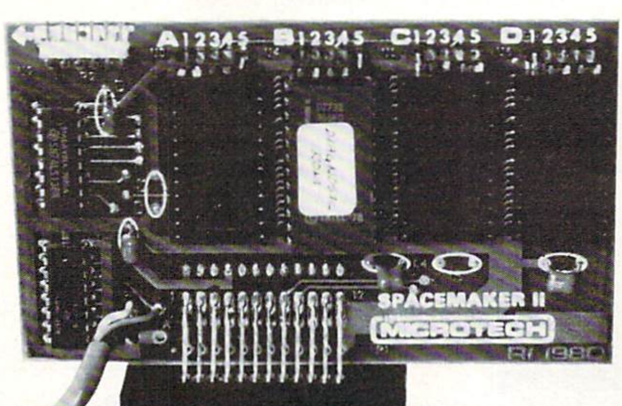
The small *PEDISK II* controller board mounts inside any 2000, 4000, or 8000 series machine. Standard systems are available with one or two disk drives. An optional third drive can also be added. The *PEDISK II* System can be operated simultaneously with any Commodore disk system for data exchange purposes.

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

*PET IS A TRADEMARK OF COMMODORE

want to try it now). Every time you LOAD it, your MLP will be sitting at 028F hex, where it can be called from BASIC by a simple SYS 655.

Neat, isn't it? Here's how it works. When a program SAVE is initiated, PET makes up a 192-byte header to record ahead of the program. The header contains the following data, from start to finish:

- One byte, a 01, which identifies the tape as a program, not a data file.
- Two bytes showing the first memory location the program is to be stored in.
- Two bytes showing the last memory location the program will take up, plus one.
- Sixteen bytes of program name that will print onto the screen during searches.
- Another 171 bytes of program name that PET will look at and compare to while searching, but which won't usually print onto the screen.

If the program name is less than 16 + 171, or 187 characters, which most of them are, PET fills out the 187 bytes with spaces.

To make up the header, PET has to locate the program name, which is stored somewhere in memory. Locations 249 and 250 decimal specify the location of the first character of the program name, and location 238 specifies its length. Normally, the contents of these locations are set during the SAVE dialogue between you and the screen, but our three POKES circumvent this. They tell PET that a 187 character "program name" will be found beginning at 033F, (831 decimal). Our previous steps have set this bogus "program name" to a sixteen-character *real* program name including trailing spaces, plus our MLP, plus the expected trailing spaces in unused bytes. We had to put our 187 bytes outside the 1st cassette buffer, because the SAVE in Step 9 clears that buffer out.

Our SYS 631xx tells PET to start recording the header, including the bogus program name, followed by the main program. Little does PET know or care that the final 171 characters in its "program name" include our MLP. (The extra steps and different SYS for the machine language main program are required because PET SAVEs BASIC and Machine Language in slightly different ways.)

When our specially prepared tape is LOADed, all the bytes from the header are deposited in the 1st cassette buffer. PET uses the first five bytes for directions on where to store the main program. It routes the next 16 bytes to the screen to tell us what program it FOUND, and it all but ignores the rest of the bytes that were read in from the header. But when we later call on them as a MLP, PET can execute them just like any other machine language code.

That's all there is to it. But what if we need a longer MLP? Where there's a will, there's a way. In the earlier detailed instructions, we limited ourselves to a 171-byte MLP. By using the method described, we retained the nicety of a 16 character *real* program name. If we are willing to give up

Little does PET know or care that the final 171 characters in its "program name" include our machine language program.

some or all of these characters, we can start our temporary storage area a little lower in the second cassette buffer, change the value of our POKEs and the SYS we use to call our MLP, and put up to 187 bytes of MLP on that header. The first 16 characters will still print on the screen, although since they make up part of a MLP rather than a "name", they'll print some strange characters. The basic idea is given here; the calculations are left up to you, as is finding the way to use this whole method with the *second* cassette buffer. ©

Commodore ROM Systems: Terminology

Jim Butterfield
Toronto, Canada

The first PET ROMs didn't seem to have a name. They were just the PET ROMs. Users perceived different types of ROM sets: some had 28 pins each and were manufactured by MOS Technology, a Commodore subsidiary; others had 24 pins and were made by outside suppliers. Then there was a bug replacement: early ROMs numbered either 6540-011 or 901447-01 were changed for corrected ROM systems.

This first ROM system was internally called Basic 1.0 by Commodore. Unfortunately, the Basic language itself is called Basic Level 2.

A year later, another ROM set arrived which once again had no name. Dealers and users just called them "new ROMs" since that's what they were at the time. Unfortunately, new ROMs aren't new any more, and calling them by that name confuses newcomers.

This second system was called Basic 2.0 by Commodore. And naturally, the Basic was called Basic Level 3. The numbers were generally used only within Commodore so that users weren't exposed to the confusion. How the Commodore people sorted it out, I can't guess.

Finally, Basic 4.0 was released, and at this point the two number systems caught up to each other. The machine prints COMMODORE BASIC 4.0 upon power-up, so that everyone knows what number belongs to this ROM set. Finally.

What happened to the missing number? Nobody seems to know. My theory is this: that Commodore internally called their first Basic "1.0 ROMs" for the obvious reason. In the meantime, Commodore sales may have become concerned by comparisons to Radio Shack Level I Basic (a fairly primitive version of Basic) and decided to start their numbering at Level II. At that point, I surmise, they were stuck, and the production 2.0 ROMs had to be called Level 3. during this period of time, the number you got depended on the department within Commodore that you were talking to. Finally, the production people must have decided that if they skipped a number everybody would be caught up. They were right: we all agree on what Basic 4.0 is, even though we're still cloudy on which one should be called Basic 2...

A proposed standard.

Let us call the first ROM, whatever its number: Original ROM. That's the ROM without a Machine Language Monitor; that can't handle IEEE disk; that has a limit of 256 elements in an array, and that has some minor tape irregularities. Many users will have upgraded their systems, but there are still a lot of the Original ROMs around, and writers for **COMPUTE!** should be specific if their programs will work only on the more recent machines.

The next ROM should be called Upgrade ROM. That's the ROM with a Machine Language Monitor and other improvements, but without the speeded garbage collection routines and without the extra disk commands such as DLOAD or SCRATCH.

Subsequent Basic ROMs may be called by their number: at the moment, Basic 4.0 is the only one out, but there's a 5.0 rumored for the near future.

Within these various styles of Basic, there are a few small variations which may or may not be significant to the reader. Some machines have graphic keyboards and others have full ASCII keyboards; there are minor changes in ROMs to accomodate the particular keyboard used. More significantly, an article which asks a user to press the TAB key may not be too helpful to a user who doesn't have such a key on his computer.

We all agree on what Basic 4.0 is, even though we're still cloudy on which one should be called Basic 2...

Similarly, there's a visible difference between 40-column and 80-column PET/CBM computers, and there are small ROM differences between the two types of machine. Some programs will run splendidly on both types of machine: but if your program doesn't, it's a good idea to specify the machine for which the program is written.

Hardware Differences

The ROM set doesn't always correspond exactly with the physical configuration of the machine. Original small-keyboard PETs can be fitted with upgrade ROM ... and can even be fitted with large keyboards, which makes them hard to recognize. Similarly, a green screen isn't always a guarantee that the machine is of more recent vintage; and subsequent PETs have gone through more than one board redesign.

The most significant hardware change seems

to be from the original PET — which can be spotted by its memory expansion edge connector, and by its screen noise when a screen POKE is performed. Subsequent models have expansion pins (not an edge connector) and a hash-free screen.

There are other architectural differences, the wide 80-column screen being the most obvious. Some of the changes are important in a negative way: for example, the "speedup" achieved on some computers by poking address 59458 is to be avoided since it can cause chip damage on certain models.

Generally, programs which behave differently in different hardware (as opposed to different ROM sets) are unusual. If there's a good reason for such behaviour, it should be documented ... the more explanations, the better.

...if (your program's) going to go elsewhere, try to spot any signs of travel sickness in advance.

Disk ROMs

We seem to have more consistency in disk ROM systems. The first 2040 ROM is usually called DOS 1.0; units fitted with it require initialization before a diskette can be used. The subsequent ROMs for the 2040 are DOS 2.0; they can be quickly spotted by the fact that the unit bumps the heads at time of power-up. The ROMs currently fitted to the Model 8050 disk are called DOS 2.5. At the present time, the 8050 has only one ROM set, but others may emerge. When this happens, it may be that most programs will work well on either ROM; but of course, any variant behaviour should be noted.

Programs intended for DOS 1.0 systems must initialize a diskette before starting to work on files; the assumption is that the data files are likely to be on a different disk from the program itself. If you are writing a program for your own DOS 2.0 or 8050 system, it's a good idea to remember to include such an initialization even though your system doesn't need it. Others may find it useful.

If you are working with REL type files, it's fairly obvious that DOS 1.0 systems are not compatible with your program. Note it anyway in your text; what's obvious to you may not be to newcomers.

It seems to be safe to assume that a 2.0 DOS, which has Append and Relative file features, will always be used with a 4.0 or later Basic. Don't be too sure. There are plenty of users who don't have the option of going to Basic 4.0. If you write your program for Basic 4.0 — and it's usually easiest for you to do it this way — note the fact. The Basic 2.0 user may want to take a shot at converting your program, but he needs to be warned that conversion is necessary.

Machine Language

It's possible to write machine language programs that will run on any PET/CBM machine. It can't be done every time, but if you can ... do it. I find it a great nuisance to have to keep copies of programs suitable for Original, Upgrade and 4.0 Basic ROMs. The trick is to use the Jump Table (Hexadecimal FFC0 to FFEA) for all input and output; it's identical in all PET ROM systems so that one program fits all machines. To repeat: you can't make your machine language program ROM-independent in every possible case; but it can be done surprisingly often.

If your program runs only on a particular machine type, document it. The same ground rules apply: if the user is cautioned, he may well take a shot at doing the conversion himself. If he's not warned, he may spend a lot of time typing in the program only to find that it doesn't work. Then he may spend hours looking for a transcription error that isn't there.

Summary

If you're writing a program that you think may be used on somebody else's machine, look it over carefully for compatibility problems. It doesn't matter whether you plan to sell the program, give it to a friend, or publish it in **COMPUTE!** — if it's going to go elsewhere, try to spot any signs of travel sickness in advance.

Of course, you don't have every model of PET and every printer and disk unit in your home. Obviously, you can't test everything yourself.

But learn a little about compatibility between machines, and you'll know where to look for potential trouble. If you're not sure, try the program on a friend's machine. If you're still not sure, add some cautionary sentences to your documentation.

PET/CBM machines are really highly compatible. Learn how to look for the few rough spots and your programs will become much more widely useable.

And if you're really not sure, appeal for help. Drop a couple of REM statements in your program asking for feedback. Many users will be glad to help ... to tell you what they needed to do to fix your program for their machine. Or better yet ... to tell you that your program worked fine on the first try. ©

Screener

Four Screen Utility Routines For The PET

Ralph Owens

The PET's cursor controls, TAB, SPC, and clear screen functions are good, but sometimes they are not good enough. Several times I have wished for a clear, but reversed, screen. At times I have longed for a scroll-right feature for some neat, action-packed cartoons.

Finally I decided to quit wishing and to write some screen routines. Listing 1 shows the results of my frustrations. These are four short, independent, relocatable, re-entrant, ROM-independent screen routines which use the 1K block of screen memory as data for a block action of some type. Each routine may be implemented alone, or all four may be placed in the PET at one time. Each routine uses one base address for indirect addressing. This is placed into addresses 0-2 which have (in all released ROM's) been reserved for the USR function vector, and are infrequently-used addresses.

The first routine moves everything on the screen one character to the right. It starts in the upper left corner and moves the entire row right. It repeats itself on the next line for 25 times. By changing the starting address, the number of columns to move (in register X) and the number of lines to move (in register Y), one can make any portion of the screen roll right.

The second routine reverses everything on the screen. The trick here is that it toggles the 7th bit of the screen character, which causes the entire screen to be the reverse of what it was. The method used does not allow easy modification to work on only a certain portion of the screen.

The third routine clears a window on the screen. The base address as shown in the listing is the memory address of the upper left corner of the window. The X register contains the height of the window. The Y register contains the width of the window. In listing 1, lines 460 to 490 show how to dynamically change the position of the window. If you change the blank to a reversed blank (160 instead of 32) then a reversed window will appear. If you also set the window to include the entire screen, then a clear, but reversed, screen will result.

The fourth routine is a scroll down routine. It moves everything on the screen down one line. By changing the value in the X register, and the value of the counter at location zero, one can get only a portion of the screen to scroll down.

Putting the two scrolling functions inside of FOR-NEXT loops will provide quite good animation. If you put the reverse screen routine in a FOR-NEXT loop, then a flashing, attention-getting display can be obtained.

These four routines are just some things that I've wanted to do with the screen. I hope that you can use them. Some other ideas that I have had are to make a scroll left function. One could grab the

```
100 PRINT"{CLR}THIS PROGRAM WILL LOAD FOUR SHORT,"
110 PRINT"RELOCATABLE, ROM INDEPENDANT, MACHINE"
120 PRINT"LANGUAGE ROUTINES INTO THE SECOND"
130 PRINT"CASSETTE BUFFER."
140 PRINT"{DOWN}THE FIRST ROUTINE WILL MOVE EVERY"
150 PRINT"CHARACTER ON THE SCREEN ONE SPACE TO"
160 PRINT"THE RIGHT."
170 PRINT"{DOWN}THE SECOND ROUTINE WILL REVERSE EVERY"
180 PRINT"CHARACTER ON THE SCREEN."
190 PRINT"{DOWN}THE THIRD ROUTINE WILL CLEAR ANY"
200 PRINT"SPECIFIED BLOCK OF THE SCREEN."
210 PRINT"{DOWN} THE FOURTH WILL SCROLL THE SCREEN"
220 PRINT"DOWN ONE CHARACHTER"
230 GOSUB2020
240 PRINT"{CLR}DO YOU WANT A DEMONSTRATION? (Y OR N)"
250 GETCH$:IFCH$=""THEN250
260 IFCH$<>"Y"THENPRINT"OH COME ON, HUMOR ME!!"
270 PRINT"{DOWN}{DOWN}{RVS}FIRST I WILL DEMONSTRATE THE SCROLL RIGHT FUNCTION"
280 PRINT"{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{RVS}BY{RVS} SHOWING{RVS} HOW{RVS} IT{RVS} MOVES{RVS} THE {RVS}WHOLE{RVS} SCREEN"
290 FORN=826TO965:READA:POKEN,A:NEXT
300
310 REM PUT ROUTINES INTO THE 2ND CASSETTE BUFFER FOR DEMO PURPOSES ONLY
320 REM THEY CAN BE RELOCATED ANYWHERE THAT IS SAFE
330
340 FORN=1TO40:SYS(826):FORM=1TO400-10*N:NEXTM,N
350 PRINT"{DOWN}{DOWN}NOTICE HOW THE SPEED INCREASED AS THE"
```

IRQ vector and view each character before the operating system gets it. Then when the top of the screen is reached, the scroll down function could be called automatically. This could also be done with the left and right scroll routines. If the screen buffer were expanded (under software) then true scrolling into and out of the non-visible portion of the buffer could be implemented. This could help word processing, some search-and-destroy games, and some advertising and attention-getting routines. Additionally, one could implement the cleared reverse by reading the reverse flag in the operating system and shunting to the reverse routine. I am sure that you can think of many other creative ideas which could be accomplished using similar techniques.

For those of you who don't wish to type in the complete listing of the program, I will supply a copy on tape for $2.50, the approximate cost of the tape and postage. Send a self-addressed envelope to:

Ralph Owens
Box 202
Enterprise, KS 67441

```
360 PRINT"TIME CONTROL LOOP COUNTER DECREASES"
370 GOSUB2020
380
390 PRINT"{CLR}{DOWN}{DOWN}{DOWN}NOW I WILL DEMONSTRATE THE SCREEN"
400 PRINT"REVERSAL PROGRAM.  I WILL TOGGLE"
410 PRINT"THE ENTIRE SCREEN 15 TIMES"
420 FORN=1TO15:SYS(863)
430 FORM=1TO200:NEXTM,N
440 GOSUB2020
450
460 PRINT"{CLR}NOW I WILL DEMONSTRATE THE CLEARING"
470 PRINT"OF A WINDOW ON THE SCREEN.  FIRST I"
480 PRINT"WILL REVERSE THE ENTIRE SCREEN, SO THAT"
490 PRINT"YOU CAN SEE WHAT HAS BEEN CLEARED{DOWN}{DOWN}{DOWN}{DOWN}"
500 SYS(863):GOSUB2020:SYS(890)
510 GOSUB2020
520
530 PRINT"{CLR}NOW I WILL CHANGE THE LOCATION OF THE WINDOW AND ITS DIMENSIONS"
540 POKE891,20:POKE895,130:REM SETS UPPER LEFT CORNER OF WINDOW
550 POKE 899,10:POKE901,20
560 SYS(863):SYS(890)
570 GOSUB2020
580
590 PRINT"{CLR}THE NEXT DEMONSTRATION IS OF THE"
600 PRINT"SCROLL DOWN ROUTINE."
610 PRINT"{DOWN}{DOWN}I WILL SCROLL THE SCREEN DOWN ONE STEP"
620 PRINT"{DOWN}{DOWN}THEN 10 STEPS
630 GOSUB 2020
640
650 SYS(926)
660 GOSUB2020
670 FORN=1TO10:SYS(926):NEXTN
680 GOSUB2020
690
700 PRINT"{CLR}THE IMPORTANT ADDRESSES IN THESE"
710 PRINT"ROUTINES ARE:"
720 PRINTTAB(10)"THEIR STARTING ADDRESSES"
730 PRINTTAB(10)"THEIR CONTROL ADDRESSES"
740 PRINT"{DOWN}{DOWN}THE STARTING ADDRESSES ARE"
750 PRINTTAB(10)"{DOWN}SCROLL RIGHT      826"
760 PRINTTAB(10)"REVERSE SCREEN   863"
770 PRINTTAB(10)"CLEAR WINDOW     890"
780 PRINTTAB(10)"SCROLL DOWN      926"
790 GOSUB2020
800
810 PRINT"{CLR}THE CONTROL POINTS ARE IDENTIFIED"
820 PRINT"IN THE COMMENTS IN THE LISTING."
830 PRINT"{DOWN}THEY CONSIST BASICALLY OF THE BASE"
840 PRINT"ADDRESS, AND THE INITIALIZING OF THE"
```

```
850 PRINT"POINTER REGISTERS X AND Y.  BY CHANGING"
860 PRINT"THESE, YOU CAN MAKE THE ROUTINES WORK"
870 PRINT"FOR ANY PORTION OF THE SCREEN."
880 PRINT"█TO SAVE THE ROUTINES ON TAPE 1,"
890 PRINT"ENTER THE MONITOR (BY SYS(1024))"
895 PRINT"AND TYPE:"
900 PRINT"█S "CHR$(34)"SCREEN ROUTINE"CHR$(34)",01,033A,03C7"
910 PRINT"█TO SAVE ON DISK DRIVE 0 TYPE:"
920 PRINT"█S "CHR$(34)"0:SCREEN ROUTINE"CHR$(34)",08,033A,03C7"
930 PRINT"█TO SAVE ON DISK DRIVE 1 TYPE:"
940 PRINT"█S "CHR$(34)"1:SCREEN ROUTINE"CHR$(34)",08,033A,03C7"
950
960 END
970
980
990 REM DATA FOR ROUTINES WITH ASSEMBLY MNEMONICS
1000
1010 REM SCROLL RIGHT ROUTINE
1020
1030 REM DECIMAL       ASSEMBLY             COMMENTS
1040 REM LISTING       NMEMONICS
1050
1060 DATA 169,0      :REM LDA #$00      :THESE FOUR LINES TELL WHERE
1070 DATA 133,1      :REM STA $01       :TO START MOVING THE SCREEN.
1080 DATA 169,128    :REM LDA #$80      :THE INDIRECT ADDRESS IS
1090 DATA 133,2      :REM STA $02       :AT LOCATIONS 1,2
1100 DATA 162,25     :REM LDX #$19      :SET X TO #LINES TO MOVE
1110 DATA 160,38     :REM LDY #$26      :SET Y TO #COLUMNS TO MOVE
1120 DATA 177,1      :REM LDA ($01),Y   :LOAD CHAR IN NEXT COLUMN
1130 DATA 200        :REM INY           :INCREMENT Y AND
1140 DATA 145,1      :REM STA ($01),Y   :STORE IN THIS COLLUMN
1150 DATA 136        :REM DEY           :DEC Y TWICE TO GO
1160 DATA 136        :REM DEY           :BACK TO NEXT COLUMN
1170 DATA 16,247     :REM BPL $F7       :BUT CHECK IF DONE FIRST
1180 DATA 165,1      :REM LDA $01       :IF SO THEN ADD 40
1190 DATA 105,40     :REM ADC #$28      :TO BASE ADDRESS
1200 DATA 133,1      :REM STA $01       :TO GO TO NEXT LINE
1210 DATA 165,2      :REM LDA $02       :ON SCREEN.
1220 DATA 105,0      :REM ADC #$00      :ADD IN CARRY TO
1230 DATA 133,2      :REM STA $02       :HI ADDRESS AND
1240 DATA 202        :REM DEX           :DEC X TO COUNT # LINES
1250 DATA 208,230    :REM BNE $E6       :IF NOT DONE THEN KEEP GOING
1260 DATA 96         :REM RTS           :ELSE RETURN TO BASIC
1270
1280 REM REVERSE SCREEN ROUTINE
1290
1300 REM DECIMAL       ASSEMBLY             COMMENTS
1310 REM LISTING       NMEMONICS
1320
1330 DATA 169,0      :REM LDA #$00      :LOAD BASE ADDRESS
1340 DATA 133,1      :REM STA $01       :INTO ADDRESS 1,2
1350 DATA 169,128    :REM LDA #$80
1360 DATA 133,2      :REM STA $02
1370 DATA 162,4      :REM LDX #$04
1380 DATA 160,0      :REM LDY #$00
1390 DATA 177,1      :REM LDA ($01),Y   :GET CURRENT CHAR
1400 DATA 73,128     :REM EOR #$80      :TOGGLE 7TH BIT
1410 DATA 145,1      :REM STA ($01),Y   :STORE NEW CHAR
1420 DATA 136        :REM DEY           :GOTO NEXT CHAR
1430 DATA 208,247    :REM BNE $F7       :IF NOT DONE
1440 DATA 230,2      :REM INC $02       :INC BASE ADDRESS
1450 DATA 202        :REM DEX           :DEC COUNTER
```

```
1460 DATA 208,240   :REM BNE $F0       :CONTINUE IF NOT DONE
1470 DATA 96        :REM RTS           :ELSE RETURN TO BASIC
1480
1490 REM CLEAR WINDOW ROUTINE
1500
1510 REM DECIMAL      ASSEMBLY             COMMENTS
1520 REM LISTING      NMEMONICS
1530
1540 DATA 169,0     :REM LDA #$00      :LOAD BASE ADDRESS
1550 DATA 133,1     :REM STA $01       :IN LOCATIONS 1,2
1560 DATA 169,128   :REM LDA #$80
1570 DATA 133,2     :REM STA $02
1580 DATA 162,5     :REM LDX #$05      :LOAD HEIGHT OF WINDOW
1590 DATA 160,16    :REM LDY #$10      :LOAD WIDTH OF WINDOW
1600 DATA 169,32    :REM LDA #$20      :LOAD A BLANK
1610 DATA 145,1     :REM STA ($01),Y   :STORE BLANK IN LOCATION
1620 DATA 136       :REM DEY           :GOTO NEXT POS
1630 DATA 16,251    :REM BPL $FB       :CHECK IF DONE
1640 DATA 165,1     :REM LDA $01       :GET LO BASE ADDRESS
1650 DATA 24        :REM CLC           :CLEAR CARRY JUST IN CASE
1660 DATA 105,40    :REM ADC #$28      :ADD 40 FOR NEXT ROW
1670 DATA 133,1     :REM STA $01       :AND REPLACE
1680 DATA 165,2     :REM LDA $02       :NOW ADD CARRY
1690 DATA 105,0     :REM ADC #$00      :TO HI BASE ADDRESS
1700 DATA 133,2     :REM STA $02       :AND REPLACE
1710 DATA 202       :REM DEX           :NOW COUNT NUMBER OF ROWS
1720 DATA 16,231    :REM BPL $E7       :IF NOT DONE, CONTINUE
1730 DATA 96        :REM RTS           :ELSE RETURN TO BASIC
1740
1750
1760 REM SCROLL DOWN ROUTINE
1770
1780 REM DECIMAL      ASSEMBLY             COMMENTS
1790 REM LISTING      NMEMONICS
1800
1810 DATA 169,191 :REM LDA #$BF      :LOAD STARTING ADDRESS LO
1820 DATA 133,1   :REM STA $01       :AND STORE IN USR VECTOR
1830 DATA 169,131 :REM LDA #$83      :LOAD STARTING ADDRESS HI
1840 DATA 133,2   :REM STA $02       :AND STORE IN USR VECTOR
1850 DATA 169,4   :REM LDA #$04      :LOAD PAGE COUNTER
1860 DATA 133,0   :REM STA $00       :AND STORE IN USR VECTOR
1870 DATA 162,240 :REM LDX #$F0      :LOAD X WITH 240 (1/4 OF 1000-40)
1880 DATA 160,0   :REM LDY #$00      :INDEX BY ZERO
1890 DATA 177,1   :REM LDA ($01),Y   :TO GET NEXT CHARACHTER
1900 DATA 160,40  :REM LDY #$28      :THEN RESET POINTER TO ONE LINE LOWER
1910 DATA 145,1   :REM STA ($01),Y   :AND STORE CHAR ONE LINE LOWER
1920 DATA 198,1   :REM DEC $01       :MOVE TO NEXT BASE ADDRESS
1930 DATA 169,255 :REM LDA #$FF      :CHECK TO SEE IF
1940 DATA 197,1   :REM CMP $01       :A PAGE BOUNDARY HAS BEEN CROSSED
1950 DATA 208,2   :REM BNE $02       :IF NOT THEN SKIP
1960 DATA 198,2   :REM DEC $02       :ELSE DEC HI ADDRESS
1970 DATA 202     :REM DEX           :NOW DEC COUNTER
1980 DATA 208,235 :REM BNE $EB       :CHECK TO SEE IF 1/4 FINISHED
1990 DATA 198,0   :REM DEC $00       :IF SO THEN DEC PAGE COUNTER
2000 DATA 208,229 :REM BNE $E5       :CHECK IF FINISHED
2010 DATA 96      :REM RTS           :IF SO THEN RETURN TO BASIC
2020 PRINT"QQPRESS ANY KEY TO CONTINUE"
2030 GETCHAR$:IFCHAR$=""THEN2030
2040 RETURN
2050 END
READY.
```

©

Machine Language: Comparison Shopping

Jim Butterfield
Toronto, Canada

One of the early things we learn in 6502 coding is how to compare numbers. Unfortunately, we are often taught wrongly. It's not hard: but some of the intuitive things we do at the start can backfire on us later.

We can compare any of our three data registers — A, X, or Y — with memory or with a fixed (immediate) value. We will usually follow this comparison with a branch: BEQ if equal, BNE if not equal, BCS (Branch Carry Set) if the register is equal or higher, BCC (Branch Carry Clear) if lower. Unless you're a very subtle programmer, don't follow a comparison with BPL (Branch Plus) or BMI (Branch Minus).

If you want to test A for less than 8, for example, it would seem natural to say CMP #$08:BMI LESS — which is wrong! Correct coding is BCC LESS. Don't feel bad if you have done this: even the Microsoft interpreter gets this one wrong occasionally.

Where does the problem arise? We're told, correctly, that a Compare instruction does a subtraction and throws away the result, leaving the flags behind to tell us how the comparison has gone. The flags affected are the same ones as for a subtract: Z, C, and N (Zero, Carry and Negative). Reading this we tend to think — wrongly! — that if we subtract a smaller number from a larger number we must get a positive result. Wrong! If we subtract a very small number, say 01, from a very large number, say hexadecimal FF, the result is of course FE and that is a negative number. If we then tested the N flag with BPL or BMI, our program would seem to tell us that 01 is not smaller than FF — and this is obviously nonsense. Don't get the impression that BPL and BMI might work if they were used on signed numbers. They don't work at all for almost any application.

The Carry flag, on the other hand, never plays us false. Carry is set if the value in the register is equal or higher.

So break yourself of the habit of using BPL or BMI after comparisons. Switch to BCS and BCC — they won't trip you up.

Address Comparisons

A common job in medium-to-large size machine language programs is to compare one address against another. You'll often be walking an indirect address through a table, and you want to know when you have reached the end.

There is no single instruction which will compare two bytes at a time for you. You'll have to make up a series of instructions to do it. There are many ways, but one of my favorites is two-byte subtraction. If you subtract one number from another, you'll notice right away which one is higher.

When using subtraction, remember not to fall into the same BPL trap we have already mentioned. We must once again make a point of using BCS and BCC.

First, a less elegant way which illustrates the methodology. Suppose we have an address at ADDR and ADDR + 1, low order first as usual; and suppose our second address is stored at TOP and TOP + 1. We can spot whether ADDR has reached or passed TOP by subtracting (low order first, of course):

```
SEC
LDA ADDR        low order
SBC TOP
LDA ADDR+1      high order
SBC TOP+1
BCS REACHED     branch if there or beyond
```

Notice that we don't care about the results of the subtraction: the flags tell us all we need. Now ... since we don't need those results, we could change the first subtraction to a comparison. This would save time and space, since the compare instructions don't need SEC:

```
LDA ADDR        low order
CMP TOP
LDA ADDR+1      high order
SBC TOP+1
BCS REACHED
```

This works in the same way as the previous program, but faster and in one less byte. It's quite common in monitors and other large programs.

It often happens that you have your working address loaded into your registers; you may have been doing arithmetic on the address. If your high-order address happens to end up in A, and the low-order, say, in X, you can then code quite elegantly:

```
CPX TOP         low order
SBC TOP+1       high order
BCS reached
```

Summary

Comparisons are quite easy to handle, once you get rid of your bad habits. The same techniques can be readily extended to compare values of greater than one byte.

After a while, the coding methods become quite automatic, and comparing methodology will be just one more tool in your bag of tricks. At that time, you can start writing programs that are beyond compare ... ©

Using TAB, SPC And LEN

Ronald L. Straley
Ft. Myers, FL

Back to the basics of programming on the PET. Let us look at two of the functions used to print with: (TAB and SPC). Also we will look at the LEN function while we demonstrate the other two.

According to COMMODORE's write-up, the TAB function will print strings in the position you specify + 1 which is great for printing to the screen in an unformatted manner. They say the SPC function prints the number you specify in blanks or spaces to move your print positions. But you still have an unformatted printout. What we will do is work up a program to demonstrate formatting on the screen and, also, when we want to, we can use the same routine to format the printer.

When using the TAB function, the PET always starts counting from the left side of the screen whenever TAB is encountered. As far as SPC it always starts counting from the last printed position and counts from there. **Example 1** is using TAB to print to the screen. **Example 2** is using SPC to print to the screen.

The problems start when we want to use the same routine for the printer. The printer looks at TAB and SPC in the same manner, always counting from the last printed position and is also unformatted.

Example 2

We can see that by using TAB or SPC by themselves, we can't use the same routine for the printer that we use for the screen. This is where the LEN function comes into play for us. Accordingly, the LEN function will count how many spaces there are in the string we want to print. So, with a combination of TAB or SPC and LEN, we should be able to format our output and use it either on the screen or on our printer.

What we want to do is space our print over so that, whatever the length of the string we want to print is, it will always line up in a formatted manner.

First we will try TAB and set up our format to space over X spaces and then format our output right justified. We will use a statement like: PRINT-TAB(X-LEN(B1$));B1$. What this will do is tab over X number of spaces and then subtract the number of spaces in our string and then start printing, except TAB starts from the right of the screen and we end up with no format again.

Example 3

But we now have our printer formatted.

Example 4

Now we are down to our last option, but the best one of all. SPC and LEN used in combination are the commands we have been looking for. On both the screen and the printer, the SPC function is used in the same manner: it starts counting from the last printed position. Let us now use a statement like: PRINTSPC(X-LEN(B1$));B1$ This will let us space over X number of spaces, but will then subtract the number of spaces in our string and then start printing from there. Since the printer and the screen treat SPC in the same manner, both printouts will be the same.

Example 4

This will work on all ROM machines, and 40 or 80 column, but if you want to use the 80 column PET, it works great the way it is since both the screen and the printer are 80. If you are using the 40 column PET and printer you may want to add 68 IFA$="P" THENX=15:Y=20 This will cause the printer to spread the lines out and you can have more room between the columns.

You can have fun and do some experimenting with the 3 commands TAB, SPC and LEN. You should now be able to fix up those troublesome print routines and only use one routine to do all your printing, whether on the screen or the printer.

```
10 PRINT"□":X=5:Y=8
20 PRINT"▓S■CREEN OR ▓P■RINTER"
30 GETA$:IFA$=""GOTO30
40 IFA$="P"THENOPEN1,4:CMD1
50 B$="RON"
60 B1$=".56"
65 B2$="1.25"
66 B3$="23.67"
70 PRINTB$;TAB(X);B1$;TAB(Y);B2$;TAB(Y);B3$
71 :
72 :
75 PRINTB$;TAB(X);B2$;TAB(Y);B3$;TAB(Y);B1$
78 PRINT:PRINT
```

```
79 :
80 PRINTB$;SPC(X);B1$;SPC(Y);B2$;SPC(Y);B3$
81 :
82 :
85 PRINTB$;SPC(X);B2$;SPC(Y);B3$;SPC(Y);B1$
88 PRINT:PRINT
89 :
90 PRINTB$;TAB(X-LEN(B1$));B1$;TAB(Y-LEN(B2$));B2$;TAB(Y-LEN(B3$));B3$
91 :
92 :
95 PRINTB$;TAB(X-LEN(B2$));B2$;TAB(Y-LEN(B3$));B3$;TAB(Y-LEN(B1$));B1$
98 PRINT:PRINT
99 :
100 PRINTB$;SPC(X-LEN(B1$));B1$;SPC(Y-LEN(B2$));B2$;SPC(Y-LEN(B3$));B3$
101 :
102 :
105 PRINTB$;SPC(X-LEN(B2$));B2$;SPC(Y-LEN(B3$));B3$;SPC(Y-LEN(B1$));B1$
110 PRINT
111 :
200 IFA$="P"THENPRINT#1:CLOSE1
```

```
EXAMPLE 1   RON   .561.2523.67
            RON   1.2523.67.56

EXAMPLE 2   RON      .56       1.25        23.67
            RON     1.25      23.67          .56
```

```
EXAMPLE 3   RON.561.2523.67
            RON1.2523.67.56

EXAMPLE 4   RON  .56    1.25   23.67
            RON 1.23   23.67     .56
```

©

Nuts And Volts #6

Gene Zumchak
Buffalo, NY

In my last column, I reviewed the hardware aspects of handshaking and also described the workings of a programmable input/output port. I suggested that I would detail the transmit and receive software for using handshaking to pass a block of data between two systems. While I still plan to do that (some day), I think a more meaningful exercise would be to consider the hardware and software necessary to connect a common peripheral to a computer system. Namely, I would like to consider the connection of a parallel style, or so-called "Centronics" style printer.

As I mentioned last time, (actually a whole line was left out during the typesetting), Centronics style handshake timing has three possible flags. The one asserted by the sender is a pulse called $\overline{\text{DATA STROBE}}$ and is usually low-true. The receiver responds with a BUSY indication, usually high-true. When BUSY falls false (unbusy), the $\overline{\text{ACK}}$ pulse (low-true) is asserted. Again as mentioned, only one of the return flags, either BUSY or $\overline{\text{ACK}}$, need be used.

I just recently took delivery of a NEC Spinwriter with a parallel interface. While it didn't quite take the day I allowed to get it running with my SYM, I did encounter a few surprises. While the connector and the pin assignments are definitely Centronics style, the flags are not true Centronics.

The product description manual gives one short paragraph on the timing. Fortunately, the accompanying timing diagram accurately describes the timing. While the printer has a flag called BUSY, it is not the Centronics BUSY, and does not take part in the handshake sequence. To avoid confusion, it might better have been called READY. When this BUSY goes low, it indicates that the printer is ready to receive data. Only $\overline{\text{ACK}}$ takes place in handshaking. An initial $\overline{\text{ACK}}$ is sent to indicate that it is ready for the first character, thus, $\overline{\text{ACK}}$ is used to prompt for characters rather than to indicate that the printer is through processing the last character. Actually, for characters beyond the first, the two descriptions mean the same thing.

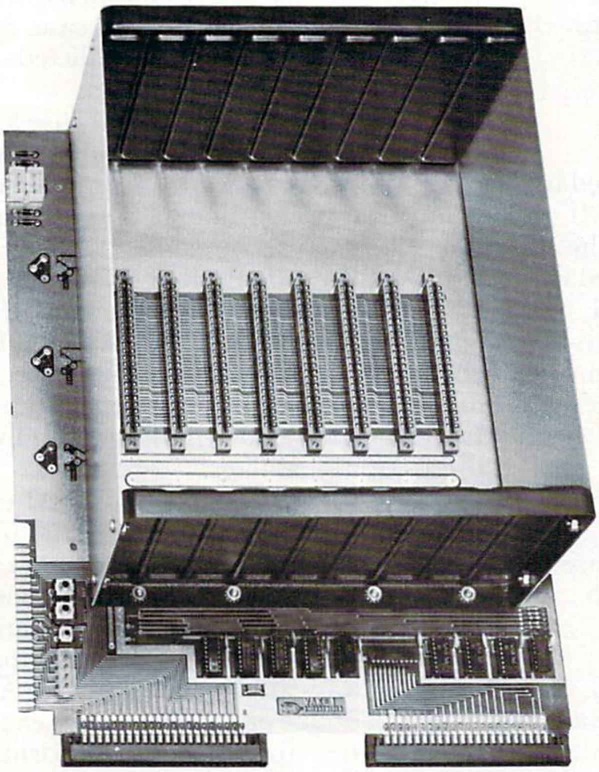
For my Receive Only model (5530), characters are accepted with the handshaking, but no characters are printed until the buffer is full, or a carriage return is received. Busy goes true when the buffer is full, and remains high while the line is being printed. This timing is shown in Fig. 1.

The easiest way to handle Centronics timing is to poll BUSY and forget about $\overline{\text{ACK}}$. However, this is not true Centronics timing, and BUSY takes no part in the handshaking. Since $\overline{\text{ACK}}$ is a narrow pulse (2.2 microseconds), it cannot be polled, but must be used to set a flip-flop. This precludes the use of a simple input port bit to handle the flag from the printer. There are two choices. We can either use a family port chip which has edge-sensitive inputs (like the 6522), or we must provide the flip-flop, which is reset by ACK and polled as a conventional BUSY flag with an input bit. DATA STROBE can be used to set the BUSY flip-flop when a character is sent. This alternative is shown in Fig. 2.

Since my SYM has no fewer than three 6522s which are available for I/O, I did not have to provide the flip-flop, but could use one of the edge-sensitive input control bits. The connection I used is shown in Fig. 3. I used the VIA chip that responds to the base address $A800. I used the low seven bits of Port A for the printer data, and bit PA7 as an input bit to poll BUSY. The CA2 output bit was used for the DATA STROBE and input control bit CB1 was used to detect the ACK edge. As can be seen from the figure, the data lines and strobe were buffered. This was necessary since the Spinwriter inputs have a 470 ohm pullup resistor. When the input is zero, there are five volts across 470 ohms and about 10 milliamps are drawn in addition to the 1.6 mA TTL input. A family port chip can usually drive only one TTL load, but certainly not 11.6 mA. I used an octal three-state gate chip for the buffer (81LS97). Any non-inverting gate, like the 74LS241 would be suitable.

A suitable program for sending a character to the printer is shown. For those not familiar with the 6522, certain defined events cause a flag in the Interrupt Flag Register (IFR) to be set. A corresponding bit in the Interrupt Enable Register (IER) can be set with software to enable the occurrence of any particular event to generate an interrupt as well. In this case, interrupts are not desirable and we will poll the IFR to see if our flag has been set. Four control bits, CA1, CA2, CB1, and CB2 can be used as input/output flags for handshaking.

VAK-1 MOTHERBOARD



We also carry:

SYM-1	**229^{00}**
AIM-65 w/1K	**389^{00}**
AIM-65 w/4K	**439^{00}**

We also do custom hardware and software for the 6502 microprocessor

PRICE: **139^{00}**

Call or write for shipping charges and our complete catalog.

The VAK-1 was specifically designed for use with the KIM-1, SYM-1 and the AIM 65 Microcomputer Systems. The VAK-1 uses the KIM-4* Bus Structure, because it is the only popular Multi-Sourced bus whose expansion boards were designed specifically for the 6502 Microprocessor.

SPECIFICATIONS:

- Complete with rigid CARD-CAGE
- Assembled (except for card-cage). Burned in and tested.
- All IC's are in sockets
- Fully buffered address and data bus
- Uses the KIM-4* Bus (both electrical Pin-out and card size) for expansion board slots
- Provides 8 slots for expansion boards on 1″ centers to allow for wire-wrap boards
- Designed for use with a Regulated Power Supply (such as our VAK-EPS) but has provisions for adding regulators for use with an unregulated power supply.
- Provides separate jacks for one audio-cassette, TTY and Power Supply.
- Board size: 14.5 in. Long x 11.5 in. Wide x 8 in. High
- Power requirements; 5V.DC @ 0.2 Amps.

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

RNB ENTERPRISES
INCORPORATED

4030 N. 27th Avenue
Suite D.
Phoenix, AZ 85017

master charge VISA

CA2 and CB2 can be outputs which are automatically asserted when data is read or written to the corresponding port. For write handshaking, for example, CA2 can be made to go low automatically when writing to Port A. CA2 is returned high by a transition on the CA1 input. Alternatively, CA2 can be pulsed low on a write to Port A, returning high without feedback after 500 ns. Finally, CA2 can be manually made low or high with software. Both CA2 and CB2 can serve as edge-sensitive inputs.

It is assumed that the character to be printed is in the accumulator when the program is called. It is good practice for an output routine to leave the registers unchanged. Since X and Y are not used in the program, they will not be affected and need not be saved. The accumulator is pushed on the stack twice; once for later use in the program, and a second time so that the program can terminate with A unchanged. This permits one output routine to follow another.

Lines 260 and 270 cause the low seven bits of Port A to be made outputs, keeping PA7 as an input bit. The PCR register, which defines the use of the CA and CB control pins, is initialized with data "$0E", called STRBOFF, which manually forces CA2 high.

The BUSY output is polled in the first loop, WAIT1. When BUSY goes false, the program falls into the second loop, WAIT2, where the IFR is read and the bit corresponding to the CB1 input flag is tested. This flag is set by a transition on the ACK line. When that condition is found, the flag is cleared by reading port B or Input Register B (IRB). Now the character to be printed is pulled from the stack and sent to port A. The DATA STROBE is exercised, by manually forcing CA2 low, then high again. This destroys the character in A, which is restored prior to the return with the second pull from the stack.

This program is by no means the only solution. There would appear to be a large number of possible connections of the port bits to accomplish the same thing, and perhaps different programs for a particular connection. However, many combinations and programs will not work. For example, why did I manually lower and raise the CA2 strobe? Why not program CA2 for the handshake pulse mode and let it pulse automatically when data is written to the A port? I confess that I tried it. Since auto pulsing will also occur for a "read" of port A as well, when an attempt was made to poll BUSY at WAIT1, data was unintentionally strobed. This caused the same character to be printed more than once. Another possible change would be to detect ACK with the CA1 input instead of CB1, and keep all functions in Port A. Again, reading Port A at WAIT1 would cause the CA1 flag to be cleared before it was recognized. The program would then fall into the loop at WAIT2 and wait for an $\overline{\text{ACK}}$ signal that would never occur since the printer would be waiting for a DATA STROBE that will never occur.

An experienced programmer will not panic when something strange happens, or worse yet, nothing at all happens. In this case, most of the unexpected results can be predicted with a careful reading of the 6522 spec' sheet.

I was initially annoyed because a legitimate BUSY flag wasn't available and also because two return flags seemed to be required. Actually, the printer has two additional output flags that I did not choose to use, one called FAULT and another for PAPER OUT. Both of these flags, however, are reflected in the BUSY flag. That is, if a fault occurs (cover not closed) or if the PAPER OUT switch is engaged, BUSY will not return false. A PAPER indicator appears on the printer panel, as well as a READY light. Thus, nothing is lost by not using these additional flags.

Is, in fact, BUSY necessary? After I had a program running successfully, I NOPed the WAIT1 loop. I then created "paper out" and "fault" conditions. The printer stopped and the READY light went out. Printing resumed as soon as the condition cleared. Thus, it appears that the printer always affects BUSY before $\overline{\text{ACK}}$, and $\overline{\text{ACK}}$ will not be asserted if an unready state exists. It would then appear that the information contained in BUSY is in fact redundant, and that only one flag, $\overline{\text{ACK}}$, and one wait loop need be used to obtain normal print operation.

The actual incorporation of the Print Character program just considered will depend upon the particular computer system which you are using, and how it handles input and output. This is, in fact, worthy of an article by itself, and I am preparing an article on console input/output, if not for this issue, then the next.

If one is attempting to interface a Spinwriter to a PET, the above program is not necessary, if the printer is interfaced by the GPIB bus. In this case, the polling and sending of flags is performed routinely internally. However, the printer will need a hardware interface that makes it look like a legitimate GPIB instrument. While building this

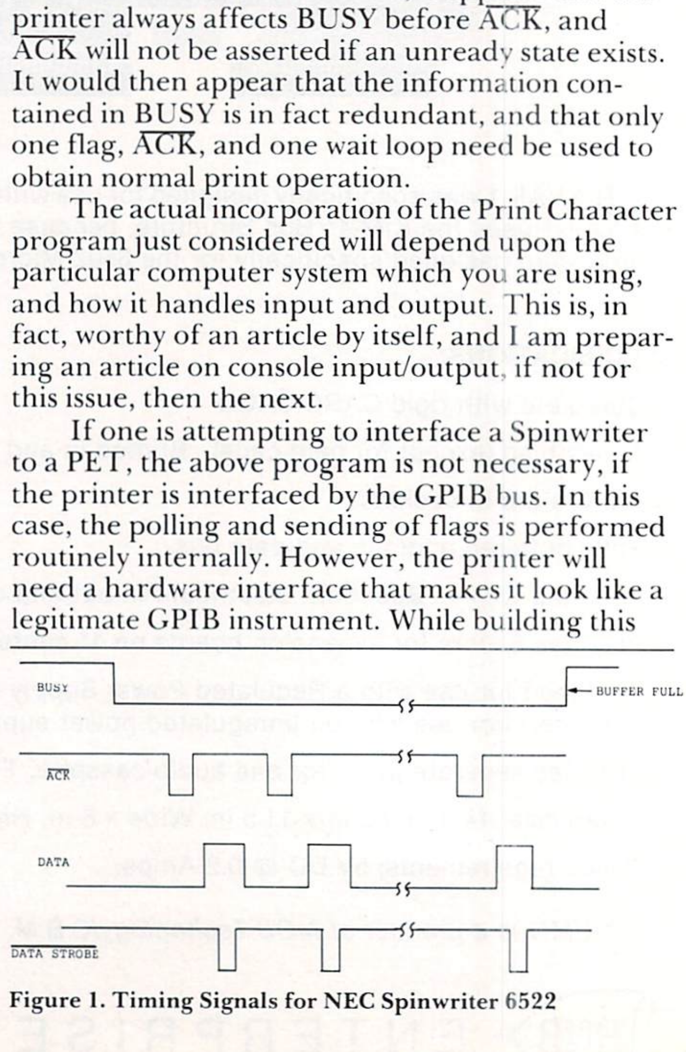


Figure 1. Timing Signals for NEC Spinwriter 6522

interface is not trivial, it would not appear to require more than a handful of gates and a flip-flop or two, perhaps $10 or less in parts (sans connectors). Since I do not have a PET, I cannot verify my gut feeling. However, I do hope to get my hands on one in the near future, and it should result in an article on building a GPIB printer interface, or perhaps building an interface for any non-standard peripheral (without using another microcomputer on the inside).

I welcome your feedback on my articles. I know that at least two people read my column.

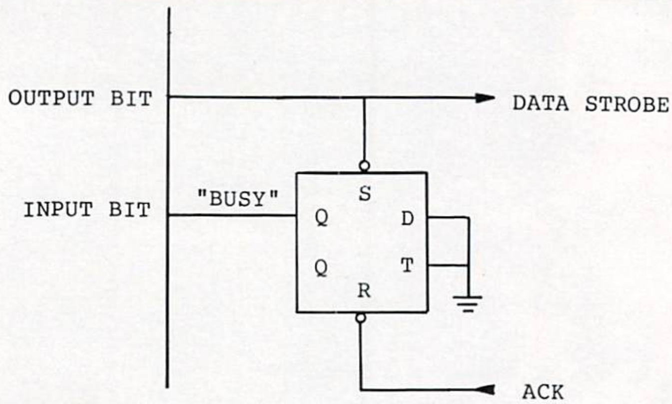


Figure 2. Flip-flop to generate "Centronics" type BUSY from ACK and DATA STROBE.

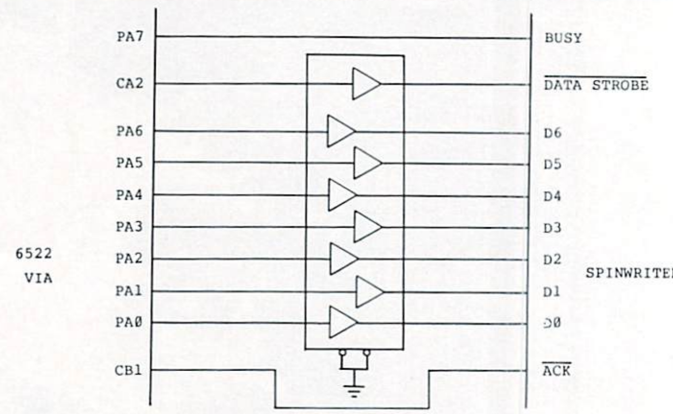


Figure 3. Connection from VIA to Spinwriter.

```
                    0010 ;BASIC HANDSHAKING WRITE ROUTINE FOR SPINWRITER
                    0020                 ;
                    0030                 ;2/25/80
                    0040                 ;
                    0050                 ; $A800 IS THE BASE ADDRESS OF 6522
                    0060                 ; ON SYM USED FOR PRINTER PORT
                    0070                 ;
                    0080 IRB             .DE $A800
                    0090 ORA             .DE $A801      ; LOW SEVEN BITS PORT A
                    0100                 ;                 IS PRINTER DATA
                    0110 IRA             .DE $A801      ; PA7 IS USED TO POLL BUSY
                    0120 DDRA            .DE $A803
                    0130 PCR             .DE $A80C      ; DETERMINES CONTROL BIT USE
                    0140                 ;                CA2 USED AS OUTPUT FOR
                    0150                 ;                PRINTER DATA STROBE
                    0160 IFR             .DE $A80D      ; BIT 4 (CB1) USED FOR ACK
                    0170 STRBOFF         .DE $0E        ; MAKES CA2 HIGH
                    0180 STRBON          .DE $0C        ; MAKES CA2 LOW
                    0190                 ;
                    0200                 .BA $300
                    0210                 .MC $300
                    0220                 .OS
                    0230                 ;
0300- 48            0240 PRINTCHAR       PHA  ; SAVE FOR POSTERITY
0301- 48            0250                 PHA
0302- A9 7F         0260                 LDA #$7F       ; MAKE LO-7 BITS PORTA OUTPUTS
0304- 8D 03 A8      0270                 STA DDRA
0307- A9 0E         0280                 LDA #STRBOFF          ; MAKE SURE CA2 HIGH
0309- 8D 0C A8      0290                 STA PCR
030C- AD 01 A8      0300 WAIT1           LDA IRA        ; WAIT FOR UNBUSY
030F- 30 FB         0310                 BMI WAIT1
0311- AD 0D A8      0320 WAIT2           LDA IFR        ; WAIT FOR ACK PULSE
0314- 29 10         0330                 AND #$10
0316- F0 F9         0340                 BEQ WAIT2
0318- AD 00 A8      0350                 LDA IRB        ; CLEAR CB1 FLAG
031B- 68            0360                 PLA '          ; GET BACK PRINT DATA
031C- 8D 01 A8      0370                 STA ORA        ; SEND TO PRINTER
031F- A9 0C         0380                 LDA #STRBON    ; PULSE DATA STROBE
0321- 8D 0C A8      0390                 STA PCR
0324- A9 0E         0400                 LDA #STRBOFF
0326- 8D 0C A8      0410                 STA PCR
0329- 68            0420                 PLA '          ; GET BACK PRINT DATA
032A- 60            0430                 RTS
                    0440                 .EN
```

DLOAD : AIM Memory Loader

Joel Swank
Rockaway, OR

The AIM 65 monitor gives you the ability to save and load non-contiguous blocks of memory on cassette tape, paper tape, or a user device. This handy feature allows you to save a program, along with any vectors or data values it needs to execute, and then to load it all with just one command. The AIM assembler uses the same format for its object files. You can assemble several programs at different locations and load them all with one load command.

But the AIM load command is missing a couple of useful features. When loading a file with the AIM load command there is no way to tell which memory locations are being loaded. After assembling a program there is no way, without a listing, to tell where the program ends. You might also want to assemble a program at one address and load it into memory at a different address, as in the case of a program that is to reside in ROM. It would also be convenient to be able to save data from one area of memory and load it back to a different area. The AIM load command cannot do this.

DLOAD is a modified version of the AIM load command that adds these two missing features. DLOAD works like the AIM load command except that it first requests an offset with the 'OFF-SET=' prompt. This hexidecimal number is input with the AIM subroutine ADDIN. ADDIN is the same routine that inputs your reply to the 'FROM=' and 'TO='

```
0000
0000           ;  DLOAD : LOAD AIM OBJECT FILE AND DISPLAY THE
0000           ;          ADDRESSES OF THE DATA LOADED. OPTIONAL
0000           ;          OFFSET LOAD.
0000           ;   AIM SUBROUTINES

0000           WHEREI =$E848                  ;OPEN INPUT
0000           INALL  =$E993                  ;INPUT A CHAR FROM AOD
0000           CLRCK  =$EB4D                  ;CLEAR CHECKSUM
0000           CHEKAR =$E54B                  ;INPUT HEX BYTE
0000           CKERR  =$E385                  ;ERROR RETURN
0000           RBYTE  =$E3FD                  ;READ OBJECT BYTE
0000           STBYTE =$E413                  ;STORE OBJECT BYTE (ADDR)
0000           DU13   =$E520                  ;CLOSE TAPE INPUT FILE
0000           ADDIN  =$EAAE                  ;INPUT ADDRESS FROM KBD
0000           RCHEK  =$E907                  ;CHECK FOR INTERRUPT
0000           WRITAZ =$E2DB                  ;DISPLAY CONTENTS OF ADDR
0000           COMIN  =$E1A1                  ;NORMAL RETURN TO AIM
0000           NUMA   =$EA46                  ;DISPLAY BYTE IN HEX
0000           OUTPUT =$E97A                  ;DISPLAY ACCUM
0000           CRLF   =$E9F0                  ;SEND CR AND LF

0000           ;   AIM RAM

0000           ADDR   =$A41C                  ;OBJECT LOAD POINTER
0000           CKSUM  =$A41E                  ;CHECKSUM STORAGE
0000           CURAD  =$A41C                  ;ADDRESS INPUT BUFFER

0000           ;   ZERO PAGE

0000           OFFL   =0                      ;OFFSET SAVE AREA
0000           OFFH   =1
0000           POINTL =2                      ;DUPLICATE LOAD POINTER
0000           POINTH =3
0000           RECLEN =4                      ;RECORD LENGTH SAVEAREA

0000                  *=$200

0200 A000      DLOAD  LDY #OFFMSG-LITS
0202 20D502           JSR KEPX                ;DISPLAY 'OFFSET='
0205 20AEEA           JSR ADDIN               ;INPUT ADDRESS
0208 B0F6             BCS DLOAD               ;ERROR - TRY AGAIN
020A AD1EA4           LDA CKSUM               ;ANY ENTERED?
020D F005             BEQ SAVOFF              ;YES, SAVE IT
020F A900             LDA #0
0211 8D1DA4           STA CURAD+1             ;NO, USE ZERO
0214 AD1CA4    SAVOFF LDA CURAD               ;COPY CURAD TO OFFSET
0217 8500             STA OFFL
0219 AD1DA4           LDA CURAD+1
021C 8501             STA OFFH

021E 2048E8    OPFIL  JSR WHEREI              ;OPEN INPUT DEVICE
0221 207C02           JSR STREC               ;START RECORD
0224 20A602           JSR PSTART              ;DISPLAY START ADDRESS
0227 4C4502           JMP BYTLUP

022A 207C02    RECLUP JSR STREC               ;START RECORD
022D A604             LDX RECLEN              ;ZERO LENGTH RECORD?
022F F037             BEQ FINISH              ;YES, END
0231 AD1CA4           LDA ADDR                ;IS NEW ADDRESS EQUAL
0234 C502             CMP POINTL              ;TO OLD ADDRESS?
0236 D007             BNE NEWLOC              ;NO, NEW BLOCK OF MEMORY
0238 AD1DA4           LDA ADDR+1
023B C503             CMP POINTH
023D F006             BEQ BYTLUP

023F 20BC02    NEWLOC JSR PEND                ;DISPLAY END OF LAST RECORD
0242 20A602           JSR PSTART              ;AND START OF THIS ONE

0245 20FDE3    BYTLUP JSR RBYTE               ;INPUT AN OBJECT BYTE
0248 2013E4           JSR STBYTE              ;STORE IT
024B E602             INC POINTL              ;BUMP POINTER
024D D002             BNE NOCY
024F E603             INC POINTH
0251 C604      NOCY   DEC RECLEN              ;COUNT BYTE
0253 D0F0             BNE BYTLUP              ;DO NEXT BYTE

0255           ; END OF RECORD
0255 20FDE3           JSR RBYTE               ;GET CHECKSUM
0258 CD1FA4           CMP CKSUM+1             ;AND COMPARE
025B D008             BNE ERROUT              ;ERROR IF NOT EQUAL
025D 20FDE3           JSR RBYTE
0260 CD1EA4           CMP CKSUM

0263 F0C5             BEQ RECLUP              ;CHECKSUM OK - NEXT RECORD

0265 4C85E3    ERROUT JMP CKERR               ;ERROR EXIT

0268 20BC02    FINISH JSR PEND                ;PRINT ADDRESS OF LAST RECORD
026B A205             LDX #5
```

prompts, so the syntax is the same. The hexadecimal number you enter is added to the starting address of each block of memory in the file. For example, a block that was saved from location $200 can be loaded back at location $1000 by replying 'E00' to the 'OFFSET=' prompt. You can calculate the proper offset by: $1000-$200=$E00. You can also load a file to a location lower in memory by adding $10000 to the desired load address before performing the calculation. A file dumped from location $B000 can be loaded back at $200 as follows: $10200-$B000=$5200. Enter '5200' in response to the 'OFFSET=' prompt. If the file contains multiple blocks, then the offset is added to the starting address of all blocks. This means you must take care when loading a file containing vectors or zero page data. These blocks will also be displaced by the offset you entered. You may load a file to its original address by entering a space or return in response to the 'OFFSET=' prompt.

DLOAD next issues the standard AIM 'IN=' prompt to open the input device. You respond as you normally would when using the AIM load command. DLOAD then displays the start and end addresses of each contiguous block of memory as it is loaded. If you are using an offset, the addresses displayed are those at which the data is being stored and not the addresses in the file. DLOAD calls the AIM RCHEK subroutine at the start of each data block so that you can stop or cancel the program. DLOAD used zero page memory locations 0-4, so be sure not to try to load anything there. Included is a listing of DLOAD assembled at location $200. DLOAD can be executed from ROM.

```
026D 20FDE3  FLUP    JSR RBYTE         ;READ END OF LAST RECORD
0270 CA              DEX
0271 D0FA            BNE FLUP
0273 2093E9          JSR INALL
0276 2020E5          JSR DU13          ;CLOSE TAPE
0279 4CA1E1          JMP COMIN         ;RETURN TO MONITOR

027C         ;   END OF MAINLINE

027C         ;   SUBROUTINES FOLLOW

027C         ;  STREC : INPUT BEGINNING OF RECORD

027C 2007E9  STREC   JSR RCHEK         ;CHECK FOR INTERRUPT
027F 2093E9          JSR INALL         ;SEARCH FOR ';'
0282 C93B            CMP #';'
0284 D0F6            BNE STREC
0286 204DEB          JSR CLRCK         ;CLEAR CHECKSUM
0289 204BE5          JSR CHEKAR        ;GET RECORD LENGTH
028C 8504            STA RECLEN        ;SAVE IT
028E 204BE5          JSR CHEKAR        ;GET RECORD ADDRESS
0291 8D1DA4          STA ADDR+1        ;AND SAVE
0294 204BE5          JSR CHEKAR
0297 18              CLC
0298 6500            ADC OFFL          ;ADD OFFSET
029A 8D1CA4          STA ADDR
029D AD1DA4          LDA ADDR+1
02A0 6501            ADC OFFH
02A2 8D1DA4          STA ADDR+1
02A5 60              RTS

02A6         ;  PSTART : DISPLAY STARTING ADDRESS OF MEMORY BLOCK

02A6 20F0E9  PSTART  JSR CRLF          ;NEW LINE
02A9 A007            LDY #STMSG-LITS
02AB 20D502          JSR KEPX          ;DISPLAY 'START='
02AE 20DBE2          JSR WRITAZ        ;DISPLAY ADDRESS
02B1 AD1CA4          LDA ADDR          ;COPY ADDR TO POINT
02B4 8502            STA POINTL
02B6 AD1DA4          LDA ADDR+1
02B9 8503            STA POINTH
02BB 60              RTS

02BC         ;  PEND : DISPLAY ENDING ADDRESS OF MEMORY BLOCK

02BC A00E    PEND    LDY #ENDMSG-LITS
02BE 20D502          JSR KEPX          ;DISPLAY ' END='
02C1 38              SEC
02C2 A502            LDA POINTL        ;DECREMENT LAST ADDRESS
02C4 E901            SBC #1
02C6 8502            STA POINTL
02C8 A503            LDA POINTH
02CA E900            SBC #0
02CC 2046EA          JSR NUMA          ;DISPLAY ADDRESS
02CF A502            LDA POINTL
02D1 2046EA          JSR NUMA
02D4 60              RTS

02D5         ;   KEPX : DISPLAY MESSAGE FROM LITERAL TABLE

02D5 B9E102  KEPX    LDA LITS,Y        ;GET A BYTE
02D8 F006            BEQ RETURN        ;QUIT ON NULL
02DA 207AE9          JSR OUTPUT        ;DISPLAY IT
02DD C8              INY               ;NEXT CHARACTER
02DE D0F5            BNE KEPX
02E0 60      RETURN  RTS

02E1         ;   LITERAL TABLE

02E1         LITS    =*

02E1 4F46    OFFMSG .BYTE 'OFFSET',0
02E3 00


02E8 5354    STMSG  .BYTE 'START=',0
02EA 00
02EF 2045    ENDMSG .BYTE ' END=',0
02F1 00

02F2                .END
02F2            ERRORS= 0000
```

©

New Products

Educational Software Exchange Announces Mail-Order Service...

SOFTSWAP is a joint project of the San Mateo County Office of Education in Redwood City, CA, and Computer-Using Educators (CUE). SOFTSWAP offers approximately 240 public domain instructional programs collected for the TRS-80, Commodore PET, Apple and Compucolor. A disk of programs for the Atari is in preparation.

Most of the SOFTSWAP programs are short, stand-alone instructional units. Many are drill & practice exercises for the elementary school level or for remedial work at the secondary level. About one-fourth are math oriented. All of the programs have been evaluated by educators and edited for factual and spelling errors, inaccurate or incomplete instructions, programming errors and other problems. Each disk contains from 5 to 28 programs for various subjects and grade levels, all for one microcomputer system.

Programs and disks may be copied without charge by visitors to the Center. Any of the 13 disks may also be purchased by mail at a cost of $10 per disk, or one SOFTSWAP disk will be sent free in exchange to any educator who contributes a disk with at least one original program for the SOFTSWAP.

Our goal is to distribute the SOFTSWAP programs as widely as possible and they may all be freely duplicated onto either disks or tapes (but may not be sold). All programs have been donated to the SOFTSWAP and we encourage educators everywhere to send their own contributions to this growing collection. New disks are listed in the CUE NEWSLETTER, or send $1 for ordering/exchange information to:

Ann Lathrop,
Library Coordinator
SOFTSWAP, San Mateo
County Office of Education
333 Main Street
Redwood City, CA 94063

Be sure to include your name, address and information on the microcomputer system(s) you have.

Copyright Notice: Every effort is made to scrupulously avoid copyright infringement and programs identified as being in violation of copyright are removed from the exchange.

The Complete Guide To Hassle-Free Pet Programming

Reston Publishing Company has announced the publication of a new computer programming book, PET BASIC by Ramon Zamora, William Scarvie and Bob Albrecht.

Perfect for the beginning PET user, this book is filled with lively examples, do-it-yourself exercises and creative explorations. You'll be confident as you experiment with your machine's many capabilities and features, and you'll create graphical images without confusion.

Experienced PET users will find this book a handy reference guide. You'll discover a variety of things your PET can do. Each new piece of information is presented logically, step-by-step with open page formats and lots of humor.

PET BASIC I can also be used to teach children to use PET (age 9 and up). You'll find games and exercises and current symbols in children's world characters.

The authors have over a dozen years of experience teaching and writing books for beginning computer users.

May, 1981, (R-5524-5) paper $12.95, 222 pp. (R-5525-2) cloth $17.95, 272 pp. To Order from Reston: call 800-336-0338.

Smart Terminal Program For The Atari® Features Autodialing

Redmond, Wa. — The MicroPeripheral Corporation has announced TSMART, a smart terminal program written for the ATARI 800®, with provision for autodialing other computers. The program is available on cassette with instructions for transferring to disk. TSMART permits transfer of BASIC programs between a remote host computer and an ATARI cassette or disk storage device. The autodial feature works in conjunction with the AUTO-MICROCONNECTION, a direct connection modem ($199.50), manufactured by the MicroPeripheral Corporation.

The program permits off-line text preparation (messages, manuscripts, letters, etc.) with a text editing or word processing program for on-line transmission. A built-in feature permits creation and storage of text, then transmission by TSMART for those who do not have a text editor.

TSMART also permits transfer of source code assembler files. The recipient can create the object code using an editor/ assembler program. A separate command is available for transferring object (hexadecimal) code files, such as ATARI Music Composer Files.

An AUTOBUF feature will open and close the memory storage buffer automatically when uploading or downloading. TSMART recognizes the automatic buffer open/close (X-on/X-off) codes transmitted by TSMART or other compatible programs. Downloading from FORUM 80 bulletin boards is also accomplished automatically. The buffer can be "toggled" on and off as many times as desired while data is being downloaded. Another feature is software selectable half or full duplex operation.

The program will also automatically send messages to bulletin boards using the standardized block mode or 16 line prompt recognition message entry.

The program was written for the ATARI 800® by Dr. James W. Clark. It can be used with any RS-232 compatible modem, although the dialer feature cannot be used with acoustic modems. TSMART is supplied in a protective binder with extensive easy-to-use operating instructions and is priced at $79.95. For additional information on TSMART or the MICRO-CONNECTION™, contact the MicroPeripheral Corporation, 2643 151st Place N.E., Redmond, WA. 98052, Telephone (206) 881-7544.

Computer Courses For Deaf Adults

Rochester Institute of Technology (RIT) will offer two computer courses for deaf adults this summer through the National Technical Institute for the Deaf (NTID). The first course, Introduction to Data Processing, will provide deaf adults with introductory technical skills applicable to job situations involving computers. Topics include: the relationship of data processing to other parts of a business and an introduction to the BASIC programming language. It will be offered from August 3–7.

The second course, Advanced Data Processing, will give experienced computer users knowledge of software applications on small computer systems. Topics include: data base, interactive programming packages, and color graphics. This course will be offered from August 10-14.

Both courses involve intensive full-day sessions and feature hands-on practice in a microcomputer center. For more information or to register, contact Donald Beil, NTID Data Processing Dept., Rochester Institute of Technology, One Lomb Memorial Drive, Rochester, NY 14623, or call (716)475-6373 (voice or TTY).

Game I/O Extender For Apple II From Vera Computing

Newbury Park, CA, June 15, 1981 — Versa Computing, Inc. has announced a new peripheral device for Apple II computers. E Z Port extends the I/O port to the outside of the computer for quick and easy changeover from game paddles to joystick to VersaWriter, etc.

E Z Port is a board which adheres to the side of the computer with a special foam adhesive strip. A 24" cable connects to the game I/O inside the Apple.

E Z Port incorporates a ZIP socket (Zero Insertion Pressure) in its design. Ordinary DIP plugs and sockets are not designed to be used over and over — eventually the sockets will not make contact and the pins will snap off the DIP plugs. ZIP sockets will help 16 pin connectors last much longer, because no pressure is exerted within the socket until the ZIP's cam lever is switched to the engage position.

At only $24.95, E Z Port is one of the most effective improvements an Apple owner can make to the computer.

E Z Port is available from your local computer retailer or from: Versa Computing, 3541 Old Conejo Road — Suite 104, Newbury Park, CA 91320. Telephone: (805) 498-1956 or 499-4800. Dealer Inquiries Welcome.

AIM 65 Enclosure

This enclosure is specially designed for the Aim 65 microcomputer. Four models are offered to hold any of the major systems on

the market for the Aim 65 including power supply. Formed out of heavy ABS plastic it features a metal card cage and sturdy metal base. The color is white with black trim. The price is $149.95 each. Contact Don-El Enterprises, 3261 Michigan Ave., Costa Mesa, CA 92626. Phone (714)546-7481.

Free PET*/CBM COMAL Compiler

The COMAL USERS GROUP has announced that it will distribute, free of charge, a COMAL Information Package that will include instructions on how Commodore PET/CBM users may obtain a FREE COMAL compiler for their computer.

COMAL is a powerful, struc-

tured language like PASCAL, yet is easy to learn like BASIC. It is destined to become quite popular in the years to come. Already it is reported to be the official programming language in DENMARK.

The COMAL Information Package contains information about User Groups, Books, Articles, and Software that both current and prospective COMAL users should know about. And it contains instructions for Commodore PET*/CBM computer users on how to get a free COMAL compiler for their computer.

To receive the COMAL Information Package, simply send a Self Addressed Stamped Envelope to: The COMAL USERS GROUP, 5501 Groveland Terrace, Madison, WI 53716. Outside the United States, please include $2.00 for Air Mail Handling, $.50 to Canada. *PET is a trademark of Commodore.

SHORTAX Program Updated For 1981 Taxes

SHORTAX, a year around tax planning program, will now compute 1981 income and social security taxes as well as 1980 and 1979. The update is based on tax laws in effect as of January 1, 1981. The program will be revised whenever the tax laws are changed.

The SHORTAX program has also been modified to run on most types of CP/M systems that use Microsoft's BASIC-80 (MBASIC) release 5.0 or later, and will run on Apple computers with the appropriate CP/M modification. SHORTAX will also operate on the Radio Shack TRSDOS systems (Models I, II or III) and on Micropolis disk operating systems using the Micropolis disk extended BASIC. Use of the program requires a cpu with at least 48,000 bytes and at least one disk drive. Under CP/M a few systems may require 56,000 bytes of cpu memory. According to the company, the program is also being converted to run on a Pertec 2000 system and the Apple disk systems.

The SHORTAX program is designed for fast, interactive calculations of before and after tax simulations. As many as 20 complex tax computations can be simulated in as little as an hour. It can be used to quickly calculate the tax impact of incorporating a business, filing an amended tax return, investing in a tax sheltered investment or transferring income producing property to a college trust fund. The program calculates the federal income tax using the tax rate schedules, the income averaging method and the optional maximum tax formula.

Where applicable, it will compute the add-on minimum tax or the alternative minimum tax. For individuals, it will compute the employee social security tax or the self employment tax. The program will compute the applicable taxes for all individual filing statuses, for corporations and for accumulation trusts.

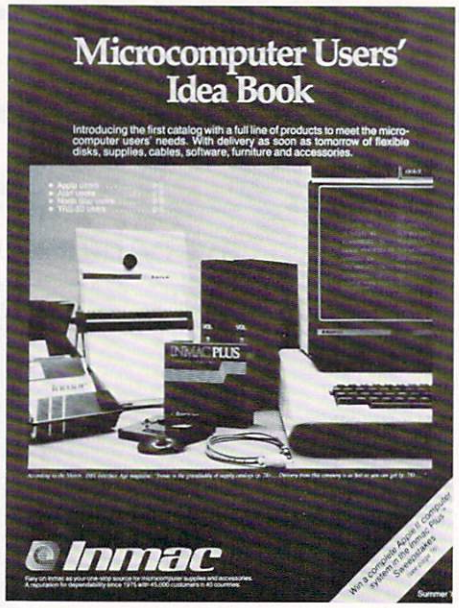
Although designed primarily for professional tax and financial advisors, it is self documented and can be used by business owners or investors who are reasonably familiar with the tax laws. The program was designed by Vernon K. Jacobs, a CPA and CLU who practices as a financial and tax consultant. He is also the editor of **Tax Angles**, a tax strategy newsletter, and is the author of **The New Taxpayer's Counterattack**.

The retail price of the program is $500 and annual updates are $300. The user manual is $15. For further details write to Syntax Corporation, Box 8137-P, Prarie Village, KS, 66208 or call at (913) 362-9967. Dealer and O.E.M. inquiries are welcome.

New Inmac Catalog For Microcomputer Users: A "Micro-Offspring"

Santa Clara, CA, June 18 — The first catalog dedicated to meeting supply, accessory, and cable needs of microcomputer users is now available from Inmac. Called **The Microcomputer User's Idea Book**, the 32-page publication lists over 1000 products — from software packages and CRTs to flexible disks, printer ribbons, and EDP-tailored furniture.

Featured for the first time by Inmac are several peripherals and software packages. These include the recently introduced high-speed Centronics 739 printer, a line of Sanyo data display monitors, and VisiCalc and DB Master software for Apple users.



The Idea Book has been designed to make product selection quick and simple. Separate sections show complete supplies, accessories, and cables for Apple, Atari, TRS-80, and Northstar. In addition, extensive cross-referencing shows compatibility with many other systems.

Other sections list products by functions — storage, preventative maintenance, safety and security, lightning, and productivity.

For more information write to:

Inmac
Department 12
2465 Augustine Drive
Santa Clara, CA 95051

Utility "Translation" For Apple Owners

Mint Software has announced the release of a utility for users of the Applewriter, Supertext and Superscribe word processing systems. Super Apple Textwriter allows the user to:

1. Convert files generated under any one of these three word producers into files accessible by the other two. For example, users wishing to convert files generated by Applewriter into files accessible by Supertext may do so with ease.

2. Convert standard text files into files accessible by either Supertext or Applewriter or Supertext files into standard text files.

This utility is of particular value to those users who wish to use their word processing system to edit information obtained from one of the communications networks (e.g. The Source), as well as those who wish to use a modem to transmit over the phone lines files created by one of the word processors. It is even possible to develop and edit a BASIC program utilizing the editing features of a word processor, and then use Super Apple Textwriter to convert the resultant file into a text file which may then be EXECed into mmemory.

Super Apple Textwriter retails for $49.95. It may be ordered from Mint Software, 6422 Peggy Drive, Baton Rouge, Louisiana 70806. Phone (504) 766-2318. Dealer inquiries are invited.

The International Microcomputer Software Directory

This directory provides all microcomputer users, professional and amateur, with a primary reference source of microcomputer software for all applications and systems.

It is drawn from a large database that is continually being updated with information collected from all parts of the world through offices in Britain and America. The directory has three main sections:

1. **System Classification** — for the user limited to a particular system. Programs compatible with each respective system are listed in subject and price order.

2. **Subject Classification** — for the user who can use any system or who has not yet purchased a system. Programs are listed under subject areas in price order giving information as to

compatible systems.

3. Software House Classification — for the user wishing to buy from a particular software house. Programs are listed in ISPN (international standard program number) order, cross-referenced with the other two sections and giving full details (where available) as to date of release, warranties, distributors, distribution medium (cassettes, disk, etc.) source code, compatible systems, special configurations needed, special features, limitations and prices.

The complete directory is available for $28.95 (plus $2.95 postage and packing). Also available at $14.95 (plus $1.95 postage and packing) are system specific directories that are extracted and cross-referenced from the main data base. These list those programs written specifically for the Apple, PET, TRS 80, and CP/M. These publications will be available in June, 1981.

An on demand search facility will be available from July 1981, which will provide up to the minute information on new software available for a particular application or system.

For more information contact:

Imprint Software
US-420 South Howes
Ft. Collins, CO 80521
(303) 482-5574
UK-16 Milton Avenue
Highgate, London N6
Tel ol-348-3998

New Autodial-Autoanswer Modem For The Atari® 400/800 Computer

Redmond Wa. — The Microperipheral Corporation has just announced a new peripheral for the ATARI® Model 400/800 Personal Computer System. The MICROCONNECTION™ is a direct connect modem which eliminates the need for acoustic coupled devices. An AUTODIAL/AUTOANSWER option permits dialing or responding to other computers automatically. The option is available for use with either the Model 400 or 800, with or without the ATARI® 850 Expansion Interface. When used with the Model 850, it is directly interchangeable with the ATARI® modem.

Applications for this new product are virtually unlimited. For example, by using the AUTODIAL feature, the unattended 400/800 can send messages. text or other data to a host computer. The ATARI® 400/800 can also be set up to act as an unattended host. The modem will automatically answer the telephone and permit the 400/800 to capture data being sent to it. Typical applications include small business bulletin boards and message centers or automatic downloading of programs and other data.

LO•BALL COMPUTERS

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 COMPUTE

LO•BALL COMPUTERS

LO•BALL COMPUTE

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

The MICROCONNECTION™ for the ATARI® is Bell 103 compatible and operates in the originate or answer mode at 300 baud. Models for use without the 850 Interface incorporate a socket (DB-25) for connection to any serial printer capable of operation at 300 baud.

Another significant feature is the provision for on-line data storage. An inexpensive, voice grade cassette recorder can be plugged into the MICROCONNECTION™ and will "transcribe" on-line communications for later playback. A special version, which is compatible with European telecommunications standards, is also available.

The unit measures 7.7 inches wide by 5.5 inches deep by 1.7 inches high and weighs less than one pound. The price, complete with autodialing terminal software, power source and connecting cable (but without options), is $199.50. The AUTODIAL/AUTOANSWER OPTION IS $79.00 extra.

For more information, contact The Microcoperipheral Corporation, 2643 151st Place N.E., Redmond, Wa. 98052, telephone (206) 881-7544.

ATARI, ATARI 400/800 and ATARI 850 are trademarks of Atari, Inc., Sunnyvale, CA, a Warner Communications Co.

Atari I/O Package

The MOSAIC I/O Package can help give the ATARI computer direct ties to the real world. The four ports on the front of the ATARI computer connect directly to a PIA for use as output as well as input ports. Now ATARI owners can build custom program controllers, interface to home control circuits, or use any hardware the imagination can devise.

The I/O package comes with 4 – nine pin connectors, 4 – twelve inch lengths of nine conductor ribbon cable, and complete instructions for their use. The documentation includes examples of home-built program controllers, how to access the ports through BASIC commands, shadow registers, or directly, and how to set up and address the ports for output. Order number H309. Price $18.00.

MOSAIC ELECTRONICS
P.O. Box 748
Oregon City, OR 97045

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.

New Literature Microcomputer Products

A new catalog from ELECTRIC SPECIALISTS presents their line

of MICROCOMPUTER interference control products. Protective devices are also included.

Descriptive sections are included which outline particular problems. Suggested solutions are given. Typical applications and uses are also outlined. Request Catalog 811.

Electronic Specialists, Inc.
171 South Main Street
Natick, MA 01760
Phone: (617) 655-1532

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
800-345-8112
In Pennsylvania call 800-662-2444

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**____________
☐ **Don't yet have one...**

☐ Please enter my 1 year (12 issue) subscription to **COMPUTE!**
☐ New subscription ☐ Renewal subscription
☐ \$20.00 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______/______/

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:

11 12

The 6502 Resource Magazine
COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-345-8112
In Pennsylvania call 800-662-2444

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**____________
☐ **Don't yet have one...**

☐ Please enter my 1 year (12 issue) subscription to **COMPUTE!**
☐ New subscription ☐ Renewal subscription
☐ \$20.00 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______/______/

11 12

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 236 BROOMALL, PA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008

Place Stamp Here

COMPUTE! Magazine
Post Office Box 5406
Greensboro, NC 27403

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 236 BROOMALL, PA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008

Place
Stamp
Here

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008
USA

The 6502 Resource Magazine
COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-345-8112
In Pennsylvania call 800-662-2444

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**____________
☐ **Don't yet have one...**

☐ Please enter my 1 year (12 issue) subscription to **COMPUTE!**
☐ New subscription ☐ Renewal subscription

☐ $20.00 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______/______/

11 12

Place
Stamp
Here

To:

YES! I would like to order The LIMITED GOLD EDITION. ☐

Please fill out the information below. We will then send you a reserved purchase authorization. Your nearest dealer will then be notified to put aside one copy for you. (Please note that the **Limited Gold Edition** is in fact, a limited edition. We can only guarantee delivery to persons filling out this card, or the reserved purchase card at their local dealer. Should the edition be over-subscribed, orders will be fulfilled on a "first come" basis only.)

YOUR ADDRESS:

Name ________________

Address ________________

City ________________

State ________ Zip ________

Telephone # ________________

YOUR DEALERS ADDRESS:

Store Name ________________

Address ________________

City ________________

State ________ Zip ________

Telephone # ________________

TYPE OF SYSTEM AND MEDIA YOU WOULD LIKE:

☐ TRS-80 Tape ☐ TRS-80 Disk ☐ APPLE Disk ☐ ATARI Tape ☐ ATARI Disk

OVER

TO ORDER THE LIMITED GOLD EDITION

1 Fill in the information on the back.
2 Detach this card and place it in an envelope.
3 Enclose a stamped self-addressed envelope.
4 Mail to:

ADVENTURE INTERNATIONAL
Limited Gold Edition
Box 3435
Longwood, Florida 32750

******* DO NOT ENCLOSE ANY PAYMENT AT THIS TIME *******

★ ★

If you would just like our free catalog of 150 software packages, fill in your name and address on the back and check here ☐

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 236 BROOMALL, PA
POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008

IF YOU CAN WAIT A MINUTE, WE CAN SAVE YOU $1,000.

With the Starwriter™ Daisy Wheel 25 cps printer from C. Itoh.

A business letter, written on a 45 cps word-processing printer, might take about two minutes to print.

With the Starwriter, it might take closer to three.

The typical 45 cps printer retails for about $3,000.

But the Starwriter 25 retails for about $1,895—thus saving you about $1,000.

And therein lies the biggest difference between the Starwriter 25 and the more expensive, daisy wheel printers.

The Starwriter 25 comes complete and ready-to-use, requiring no changes in hardware or software. It uses industry-standard ribbon cartridges, and it's "plug-in" compatible to interface with a wide variety of systems, to help lower system-integration costs.

Using a 96-character wheel, it produces excellent letter-quality printing on three sharp copies with up to 163 columns, and offers the most precise character-placement available, for outstanding print performance.

C. Itoh's warranty;

3 months on parts and labor, supported by one of the best service organizations in the industry.

$1,000 OFF

Leading Edge Products, Inc., CM-5
225 Turnpike Street,
Canton, Massachusetts 02021

Dear Leading Edge:
I'd like to know more about the Starwriter, and how spending a minute can save me a grand.
Please send me the name of my nearest dealer.

Name__________
Title__________
Company__________
Street__________
City__________
State______ Zip______
Phone: Area Code__________
Number__________

LEADING EDGE.

Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Dealers: For immediate delivery from the Leading Edge Inventory Bank™ call toll free 1-800-343-6833
In Massachusetts, call collect (617) 828-8150. Telex 951-624

The Great American Solution Machine.

Meet Commodore. The business computer that's providing solutions for more than 100,000 people all over the world. Built by one of the pioneers in office machines. With a reputation for quality that can only come from vertical integration of the total manufacturing process. Commodore builds, not assembles.

Compare Commodore's word and data processing capabilities with computers costing twice or even three times as much. You'll see why so many small businesses are turning to Commodore for solutions to problems as varied as these:

☐ A car leasing company's customers were terminating too early for account profitability. Solution: A 16K Commodore. It analyzes cash flow on over 1200 accounts, identifies those for early penalties, and even writes up lease contracts. Commodore paid for itself within weeks.

☐ A law firm needed a high quality, easy-to-use, affordable word processing system. Solution: Commodore plus its WordPro software package. At a $6,000 savings.

☐ A gasoline retailer needed to inventory, order and set prices; determine Federal and state income taxes; and comply with Federal pricing and allocation regulations. All done daily weekly, monthly and yearly. Solution: Commodore. It keeps his business on track — and Uncle Sam off his back.

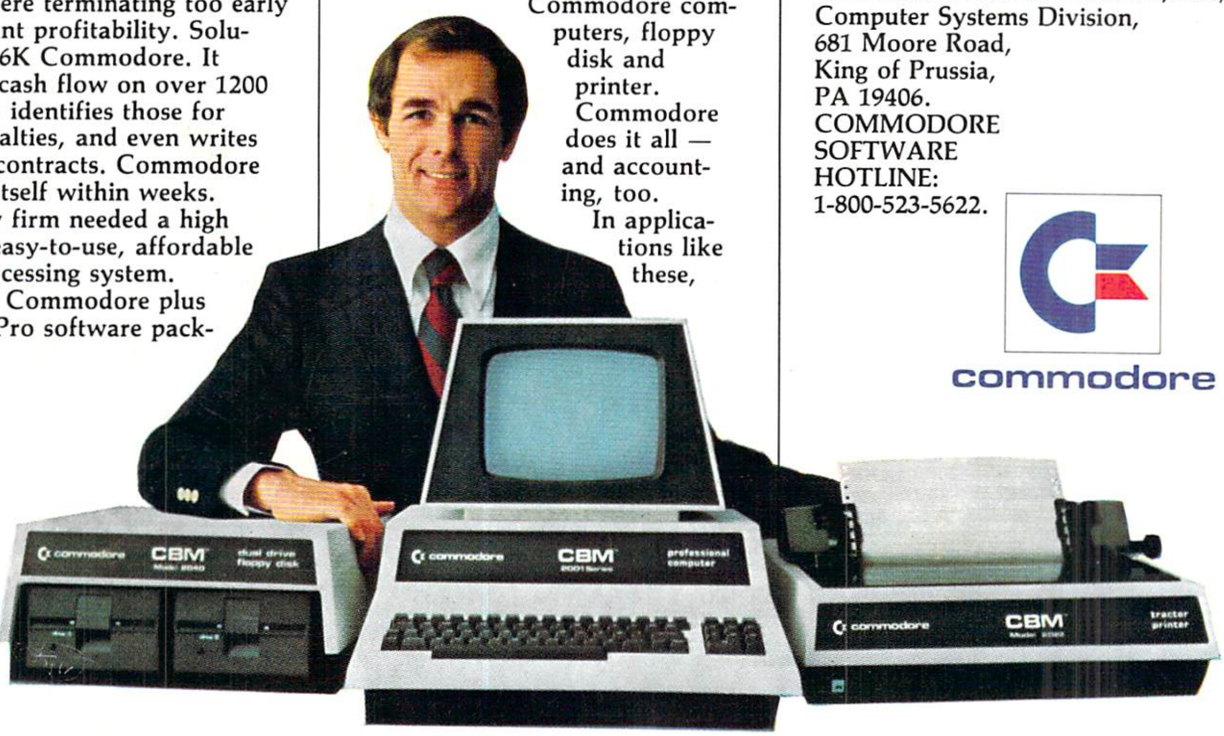
☐ A paint and wallpaper store had to inventory over 600 expensive wallpaper lines for profitability, monitor distributor sales, set and track salesmen's goals, and help the customer select the right size, pattern and quantity. Solution: Two 32K Commodore computers, floppy disk and printer. Commodore does it all — and accounting, too.

In applications like these, and many more, Commodore solves the problems that stand in the way of increased profitability. Commodore can provide the solution in your Great American business, too. Find out more by calling or writing any of Commodore's District Sales Offices.

WOBURN, MA 2 Tower Office Park 01801. (617) 938-0552.
KING OF PRUSSIA, PA 761 Fifth Avenue 19406. (215) 666-7950.
SANTA ANA, CA 1701 E. Edinger Road 92705. (714) 972-1415.
SANTA CLARA, CA 2344B Walsh Avenue 95051. (408) 727-4755.
DALLAS, TX — the new phone number is: (214) 458-1000.

Commodore Business Machines, Inc.,
Computer Systems Division,
681 Moore Road,
King of Prussia,
PA 19406.
COMMODORE SOFTWARE HOTLINE:
1-800-523-5622.

commodore



COMMODORE