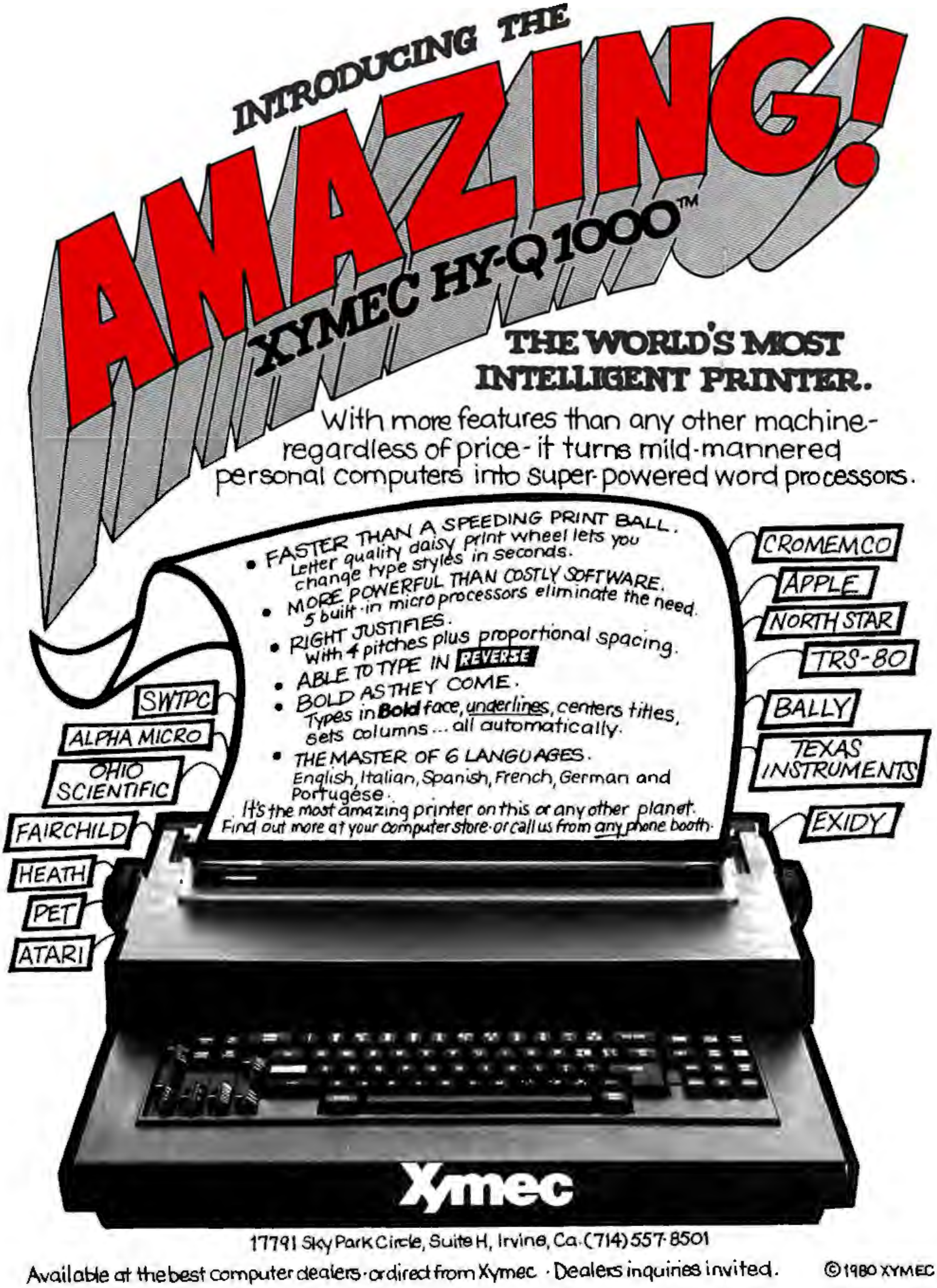
MAY/JUNE, 1980 ISSUE 4. $2.00

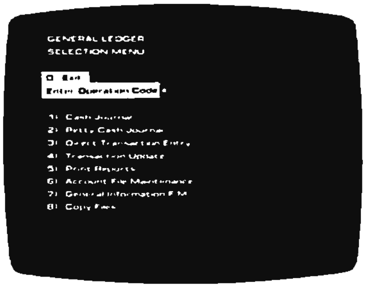
COMPUTE.

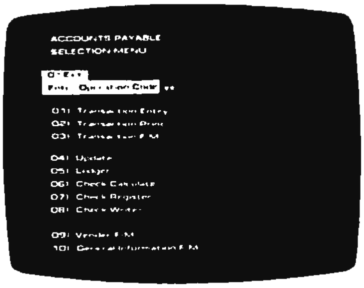
The Journal for Progressive Computing™

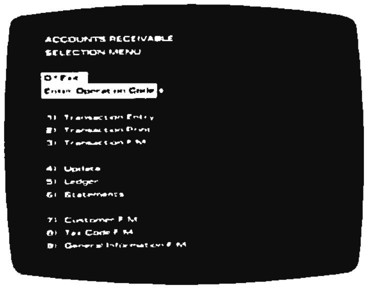

INTRODUCING THE
AMAZING!
XYMEC HY-Q 1000™
THE WORLD'S MOST INTELLIGENT PRINTER.
With more features than any other machine - regardless of price - it turns mild-mannered personal computers into super-powered word processors.
• FASTER THAN A SPEEDING PRINT BALL. Letter quality daisy print wheel lets you change type styles in seconds.
• MORE POWERFUL THAN COSTLY SOFTWARE. 5 built-in micro processors eliminate the need.
• RIGHT JUSTIFIES. With 4 pitches plus proportional spacing.
• ABLE TO TYPE IN REVERSE
• BOLD AS THEY COME. Types in Bold face, underlines, centers titles, sets columns ... all automatically.
• THE MASTER OF 6 LANGUAGES. English, Italian, Spanish, French, German and Portugése.
It's the most amazing printer on this or any other planet.
Find out more at your computer store or call us from any phone booth.
CROMEMCO
APPLE
NORTH STAR
TRS-80
BALLY
TEXAS INSTRUMENTS
EXIDY
SWTPC
ALPHA MICRO
OHIO SCIENTIFIC
FAIRCHILD
HEATH
PET
ATARI
Xymec
17791 Sky Park Circle, Suite H, Irvine, Ca. (714) 557-8501
Available at the best computer dealers - or direct from Xymec · Dealers inquiries invited.
©1980 XYMEC

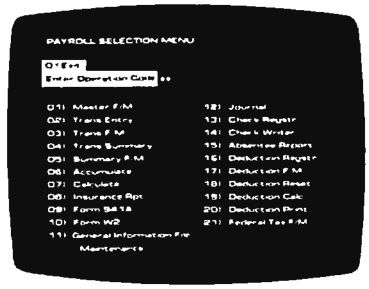
Professional Business Software

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









General Ledger

- Holds Up To 300 Accounts.
- Accepts Up To 3000 Transactions Per Month.
- Cash Disbursements Journal, Cash Receipts Journal, and Petty Cash Journal for simplified data entry.
- Maintains Account Balances For Present Month, Present Quarter, Present Year, Three Previous Quarters, And Previous Year.
- Complete Financial Reports Including Trial Balance, Balance Sheet, Profit & Loss Statement, Cash Receipts Journal, Cash Disbursements Journal, Petty Cash Journal and more.
- Accepts Postings From External Sources Such As Accounts Payable, Accounts Receivable, Payroll, Etc. **$295.00**

Accounts Payable

- Interactive Data Entry With Verified Input And Complete Operator Prompting.
- Automatic Application Of Credit And Debit Memos.
- Maintains Complete Purchase Records For Up To 200 Vendors.
- Invoice File Accepts Up To 400 Invoices.
- Random Access File Organization Allows Fast Individual Record Updating.
- Multiple Reports Provide A Complete Audit Trail.
- Check Printing With Full Invoice Detail.
- Full Invoice Aging.
- Automatic Posting To General Ledger **$195.00**

Accounts Receivable

- Maintains Invoice File For Up To 300 Invoices.
- Accomodates Full Or Partial Invoice Payments.
- Customer File Maintains Purchase Information For Up To 1000 Customers.
- Allows For Automatic Progress Billing.
- Provides For Credit And Debit Memos As Well As Invoices.
- Prints Individualized Customer Statements.
- Interactive Data Entry With FullOperator Prompting.
- Complete Data Input Verification And Formating.
- Automatic Posting To General Ledger **$195.00**

Payroll

- Maintains Monthly, Quarterly, And Yearly Cumulative Totals For Each Employee.
- Payroll Check Printing With Full Deduction And Pay Detail.
- Sixteen Different Reports Including W2 And 941.
- Interactive Data Entry With Easy Correction Of Entry Errors.
- Automatic Data Verification.
- Complete Job Costing Option With Cumulative Totals And Overhead Calculations.
- Random Access File Organization For Fast Updating Of Individual Records.
- Automatic Posting To General Ledger....**$350.00**

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

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

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

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

CMS Software Systems

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

Table of Contents

Subscription Information:
U.S. $9.00/6 issue year
CANADA $12.00/6 issue year (U.S. funds)
Europe: Air Subscription $22.50 U.S./6 issue year
Other areas: Inquire for air rates.

Address all articles, circulation questions and other inquiries to:
COMPUTE.
P.O. Box 5406
Greensboro, NC 27403
(919) 272-4867

Dealers in Canada should contact:
Micron Distributing
409 Queen Street West
Toronto, Ontario M5V 2A5
(416) 363-6058

In the United Kingdom and Europe, new subscribers and dealers should contact:

L.P. Enterprises
8-11 Cambridge House
Cambridge Road
Barking, Essex
England IG1 18NT
(United Kingdom and Northern Europe)

Ing. W. Hofacker GMBH
8 Munchen 75
Postfach 437
West Germany
(Germany, Switzerland, and Austria)

Staff of COMPUTE:
Robert C. Lock, Editor/Publisher
Carol Holmquist Lock, Circulation Manager.
Larry Isaacs, Software/Hardware Lab.
Joretta Klepfer, Editorial Assistant.
Marlene Pratto, Educational Assistant.

COMPUTE. receives continuing assistance from the following persons:
Harvey B. Herman, University of North Carolina at Greensboro, Dept. of Chemistry. Editorial Assistance.
David Gibbs, The Design Group, Greensboro, N.C., Art Direction/Design Consultation.
Jim Butterfield, Toronto. Editorial Assistance.

The following writers contribute on a regular basis as Associate Editors:
Gene Beals, 115 E. Stump Road, Montgomeryville, Pennsylvania 18936.
Len Lindsay, 1929 Northport Drive #6, Madison, Wisconsin 53704.
Roy O'Brien, Post Office Box 426, Beaumont, California 92223.
Eric Rehnke, # 61, 540 S. Ranchview Circle, Anaheim Hills, California 92807.

Application to mail at controlled circulation postage rates is pending at Greensboro, North Carolina. Postmaster: Send change of address to COMPUTE, Post Office Box 5406, Greensboro, NC 27403.
COMPUTE is published by Small System Services, Inc., 900-902 Spring Garden Street, Greensboro, North Carolina 27403. Telephone: (919) 272-4867. COMPUTE is published six times each year on a bimonthly schedule. Subscription cost for one year is $9.00. COMPUTE is available by subscription or through retail dealer sales. Subscription prices higher outside the US. (See below)

Address all manuscripts and correspondence to COMPUTE. Post Office Box 5406, Greensboro, N.C. 27403. Materials (advertising art work, hardware, etc.) should be addressed to COMPUTE., 900 Spring Garden Street, Greensboro, N.C. 27403.

ISSN 0194-357X.

Authors of manuscripts warrant that all materials submitted to COMPUTE. are original materials with full ownership rights resident in said authors. By submitting articles to COMPUTE. authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of Small System Services, Inc. Unsolicited manuscripts not accepted for publication by COMPUTE. will be returned if author provides a self-addressed, stamped envelope. Program listings should be provided in printed form as well as machine readable form. Articles should be furnished as typed copy with double spacing. Each page of your article should bear the title of the article, date and name of the author.

Advertisers Index

*DMC = Direct Mail Card

The Editor's Notes

Robert Lock

The Fifth West Coast Computer Faire San Francisco, California March 14,15 and 16

A wonderful show! I guess I enjoyed it partially because it was our first West Coast show, and we got to meet many people we'd only talked to on the phone.

The show broke it's own attendance projections with close to 19,000 people. . .most of them computer user/consumers. One interesting point: Apple, Commodore, and Atari didn't show, deciding (I assume) to hold off for the NCC in Anaheim. Instead, large regional dealers were there, frequently receiving corporate staff support. On Saturday and Sunday, for example, Commodore corporate staff were assisting the Mr. Calculator people, and Atari retail training staff and support personnel were assisting several dealers.

An Apple Breakthru

Microsoft Consumer Products (the new end-user/dealer branch of Microsoft) introduced their first piece of hardware. Called the "Z-80 Softcard", it's simply a Z-80 board that plugs into your Apple, comes with Microsoft Disk Basic and CP/M operating system, and opens up a whole new world for you Apple owners. At a package price of $349.00 (with June deliveries) they're being innundated. For the full details, read The Apple Gazette in this issue.

PET Owners Take Heart

Since the show, I've talked to both Commodore and Microsoft about the possibility of a Z-80 softcard for the PET. At this point, it hasn't been ruled out, so I'd like a show of interest. You'll find a direct mail response card in the center of the magazine to assist in the survey, and since I'm doing this on my own initiative, at my own expense, I've added a few other choices as well. I'll keep you posted on the results.

Other Highlights of the Show

Systems Formulate Corporation demonstrating their new line of "heavy-duty" business printers for the PET/CBM line.

Harry Saal of Nestar Systems demonstrating the Cluster-One System for the Apple.

Bob Skyles of Skyles Electric Works showing off his current line of PET products, including Macrotea, which started shipping a week before we sent you Issue 3's with a review indicating he still wasn't shipping. Oh well.

David Cox and Micro Technology Unlimited demonstrating their new 6502 DOS with a 20 KB transfer rate (!!!).

Atari reps trying to balance machine displays between new business software and Star Raiders, only to keep finding them mysteriously returning (in mass) to Star Raiders.

Professional business software vendors emerging, with a lot of interest in **Word Pro III** at the Commodore/Mr. Calculator booth; **Compumax** and **Grass Valley** were also exhibiting/demonstrating their respective business software offerings.

Infoworld (the new weekly newspaper of microcomputing) had a nice touch supporting their timeliness. On Sunday morning they handed out a special Faire edition covering the Faire. Unfortunately, COMPUTE. didn't win the Apple they gave away.

Programma showing off their newest software offerings for PET, Apple and Atari, complete with a wide-screen TV so we could all watch.

Enough of this... on with Issue 4. Thanks, California, for an exciting show.

On Commodore's new ROMs and the 80 Column CBM

I've received a bit of flack for treating the new ROM introduction rather matter-of-factly. What can I say? I do feel the new 80 column machine can have a substantial impact on the emerging small business market, and I like the design of the unit. It makes sense to bring it out with the new Basic 4.0 as standard (e.g. built-in DOS). For the market it's designed for (small business, etc.) a built-in DOS is ideal (if not crucial). It presents a pain for those of you software vendors who are dilligently trying to keep up with Commodore, but at the same time, we can't encourage or expect them to take a no-growth position. There have been valid reasons (improvements) for each new set of ROM releases. I do think it's time to slow things down a bit. Certainly they must continue to support all existing units as they continue to introduce new ones. I will try to keep you posted as we find out what's what.

The IEEE-488 Acoustic Coupler

COMPUTE is still coupler-less. I appreciate your calls and letters asking for preview information, but I still don't have it. Our coupler should have arrived by now but hasn't. I promise (with Commodore's help) I'll have a review for new issue.

Good News for Apple and Atari Owners

Al Baker (of Al Baker's Game Corner in Interface Age and Image Computer Products) will be joining us with a column. Al Baker's Programming Hints will begin in Issue number 5 of COMPUTE, providing helpful information to Apple and Atari owners. Look for his excellent article in this issue on Atari basic. COMPUTE grows on!

The Editor's Feedback

I mentioned earlier in this column my personal survey of your feelings on a Z-80 softcard for the PET. I've added other information to the card as well, and if you respond, will try to continue this process. The cards are a good way to get feedback to me in a concentrated form, and will help me keep the magazine on traget. I feel that many of you won't write without some easy to use method such as this one. For those of you who want to take the time to write, please do. Eventually (note the eventually), I read all letters that come through here. Even when I don't have time to respond, the message is received. Here (with name removed) is a "wish-list" that one reader sent me. It was not intended for publication, but for my own information... that's why I've removed the name of the author. I'm printing excerpts from it because it's an excellent example of mail that's extremely valuable to me:

A Reader's Shopping List

1. Devote last page to corrections checklist. Herman's fantastic article in issue 2 has a typo in line 205; it should be POKE 53, 16 and not 24.

Okay, it's done. The last page is now reserved for previous goofs. Several readers have suggested a clearly defined location for fixes. From my end, I'll work on keeping the last page blank.

2. Provide clear diagrams of any hardware type articles. RESET button is a good example. My dealer is overworked and it is a crime for me to bug him about the location of one pin.

3. A tutorial on sys and usr functions. I have similar problems to Mr. Wachtel (HELP, Issue 2), and though I have given him a solution to one of his problems, others remain. The manual is of no help whatsoever.

Watch Issue 5 for this one...

4. Provide tapes of longer programs...

This request shows up on the Editor's Feedback card. I'll be curious to see what you think. Thanks, anonomyous reader, for your input. I enjoyed the chat. Perhaps we'll call the last page **"Capute!"**

Writers

COMPUTE currently pays for unsolicited material (that we accept for publication) at a page rate of $25.00 per published page. We also pay for little things, short program hints, three-liners, etc.

We expect to raise our rates as we grow, so stick with us. When you write for COMPUTE, you can expect three things: 1. You'll help other machine owners; 2. Your material won't sit around for six months; and 3. You'll appear in a young, growing magazine that now is distributed in the US, Canada, Great Britain, Europe, Australia, Japan, and a lot of places in between. Follow the directions on the title page for submitting your material to COMPUTE. When and if we ever publish your work again, as for example in "Best of COMPUTE." or "COMPUTE's Book of Atari", you'll be paid again, the second time at our then current page rate.

Business Software

HELP! I've added a spot on the Editor's Feedback card in an attempt to entice you business user's to provide some help. I know you're out there, so write.

Product Complaints/Problems

If you have a problem or complaint that involves a product in the micro industry, I'd like to offer the following guidelines:

1. Write (or call) the manufacturer/supplier. Most will cheerfully replace a defective product, or refund your money if you're not satisfied with the performance of the product.
2. Be fair with the supplier. Hostility, while understandable, is frequently not justified. A simple letter, stating the problem, should be as effective (or more so) than a threatening one.
3. If none of this produces results, drop me a note at the address below, and I'll look into it. I've had several "chats" with manufacturers recently as a result of reader complaints.

One final point: Don't jump to the worst conclusion. I think our industry is maturing. Magazines are certainly more alert to "scams", as are their readers. With good communications between readers, magazines, and advertisers, we'll resolve some of the problems of the past.

Articles, Programming Notes, New Product Announcements.

Material submitted for possible publication in COMPUTE should be addressed to:

Robert Lock, Editor
COMPUTE
P.O. Box 5406
Greensboro, NC 27403

Program Listings

For an explanation of COMPUTE's PET/CBM program listing codes, see page 120.

Computers and Society

David D. Thornburg
and Betty J. Burr
Innovision
P.O. Box 1317, Los Altos, CA 94022

Assuming that you are a personal computer enthusiast, you have no doubt noted that your approach to computers is different from that of some of your neighbors. We have spent many hours with people who are afraid that computers are responsible for all that is wrong with the world. While there is no one cause for this attitude, the following perceived characteristics of the computer world seem to be instrumental in perpetuating a fear of technology in some people:

1. Computers are only understood by "strange" genius level people who would rather hide out with machines than interact with other humans.
2. Computer technology will cost people their jobs.
3. Increased reliance on computer technology will make our society vulnerable to collapse from technological failure. (The New York blackout of some years back is often cited as an example of future woes in this regard).
4. Any small computer which fits in the home budget is little more than a toy - a glorified video game; the "real" computers are buried in the cellars of banks and large corporations.

Despite computer-phobic statements like these, the phenomenal revolutionary growth of computer technology will soon reach out to touch us all - computer users and non-users alike. The computer will soon be like the automobile. While we do not all need to drive cars to be functioning members of society, we do need to know enough about them to not walk out in the street without looking both ways. We would like to encourage you to take part in discussions with your friends to let them know more about the realities and promise of this technology, and we would like to make some suggestions that you might find helpful.

The remainder of this column will be devoted to background material you might find interesting yourself, and that you might want to share with those friends who engage you in discussions about computers. Books and movies are the principal media for information about computers, and the ways they may be used in the future. As we promised in the last column, we will present reviews of books in this column from time to time, and we will on occasion review relevant movies as well.

These are two types of books that lend themselves to inclusion in this column. The first is the "Role of the Computer in Society" book, and the second is the "I Don't Know Anything At All About Computers But I Wish I Did" book. The first of these is typically quite philosophical in content, and is conducive to being read with furrowed brow and heightened social awareness. The second type of book is devoted to cutting through the mystery of computers and getting you to flex your fingers and start writing programs. In suggesting books of each type, we leave it to you to decide (after reading the books yourselves) which type might be appropriate for those of your friends who want to know more about this technology.

We must confess that our reason for suggesting that books be used to help you interact with your friends on this issue is based on a problem that one of us (DT) often has when giving a demonstration of a computer. As a professional in the personal computer field, he finds it all too tempting to start off at the right level and then to "show off" by running some zippy program written in half-BASIC half-machine code, all the time suggesting that the guest could generate programs twice as exciting in half the time. Trying, in the span of a few minutes, to demonstrate a complex system while trying "6 CTRL K RETURN", tossing out jargon left and right, and bringing up a super dazzling demo has turned off several people who would have much benefited from a softer approach. And, let's face it, you too really are a computer enthusiast, or you wouldn't be reading this magazine. Your enthusiasm for personal computers requires that you exercise tremendous restraint when demonstrating your system to someone who has never sat at a keyboard before. Good books can help build a buffer between your expertise and your neighbor's unfamiliarity and apprehension.

While there are many books we can write about, we thought that the following sampler should get you started towards building a library to share with your friends.

One of the better books on the social impact of computers is *Computer Power and Human Reason: From Judgment to Calculation,* by Johseph Weizenbaum (W. H. Freeman and Company, San Francisco, ISBN 0-7167-9463-3, $5.95). Dr. Weizenbaum is a Professor of Computer Science at MIT who is widely known for his development of ELIZA, a natural language processing system. This system was used to create a script that made the system behave similarly to a Rogerian psychotherapist conducting an initial interview. The user would sit at a computer terminal keyboard and take part in a "discussion" with the computer in which it appeared that the computer "understood" what was going on. In fact, the program was looking for certain words in the user's responses and was using these words in later interactions. Of concern to Weizenbaum was that several psychoanalysts looked forward to this program paving the way to automated psychotherapy - something the author neither believed in nor intended.

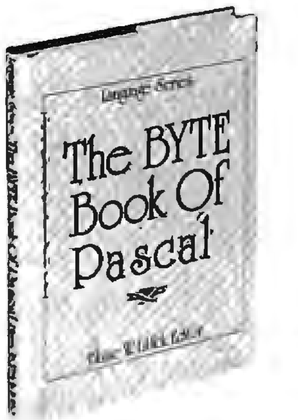
Take the mystery out of programming

with the latest from BYTE Books™

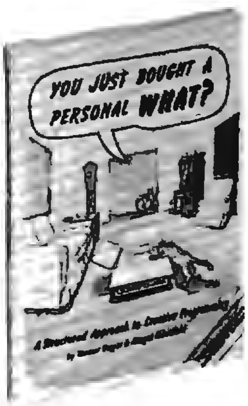
The BYTE Book of Pascal

Edited by Blaise W. Liffick

Based on the growing popularity of Pascal as a programming language, numerous articles, language forums and letters from past issues of BYTE magazine have been compiled to provide this general introduction to Pascal. In addition, this book contains several important pieces of software including two versions of a Pascal compiler - one written in BASIC and the other in 8080 assembly language; a p-code interpreter written in both Pascal and 8080 assembly languages; a chess playing program; and an APL interpreter written in Pascal.



$25.00 Hardcover pp. 342
ISBN 0-07-037823-1



YOU JUST BOUGHT A PERSONAL WHAT?

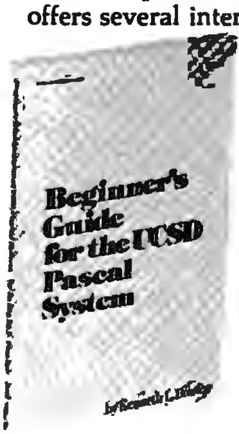
by Thomas Dwyer and Margot Critchfield

Whether you are a novice programmer or an experienced computer user, this book is filled with practical ideas for using a personal computer at home or work. It will take you through the steps necessary to write your own computer programs, and then show you how to use structured design techniques to tackle a variety of larger projects. The book contains over 60 ready-to-use programs written in Microsoft and Level II BASIC in the areas of educational games, financial record keeping, business transactions, disk-based data file and word processing. **$11.95 pp. 256 ISBN 0-07-018492-5**

Beginners Guide for the UCSD Pascal System

by Kenneth Bowles

Written by the originator of the UCSD Pascal System, this highly informative book is designed as an orientation guide for learning to use the UCSD Pascal System. For the novice, this book steps through the System bringing the user to a sophisticated level of expertise. Once familiar with the System, you will find the guide an invaluable reference tool for creating advanced applications. This book features tutorial examples of programming tasks in the form of self-study quiz programs. The UCSD Pascal Software Systems, available from SofTech Microsystems Inc, 9494 Black Mountain Road, San Diego CA 92126, is a complete general purpose software package for users of microcomputers and minicomputers. The package offers several interesting features including:



- Programs which may be run without alteration on the General Automation or DEC PDP-11 minicomputers, or an an 8080, 8085, Z80, 6502, 6800, or 9900 based microcomputers.
- Ease of use on a small, single-user computer with display screen and one or more floppy disk drives.

$11.95 ISBN 0-07-006745-7

These and other BYTE/McGraw-Hill books are available from BYTE Books or your local computer store.

Please send
☐ ______ copies of The BYTE Book of Pascal
☐ ______ copies of You Just Bought a Personal What?
☐ ______ copies of Beginner's Guide for the UCSD Pascal System
Add 75¢ per book to cover postage and handling.

Name Title Company

Street City State/Province Code

☐ Check enclosed in the amount of $______
☐ Bill Visa ☐ Bill Master Charge
Card No.______ Exp. Date______

BYTE BOOKS Division • 70 Main Street
Peterborough, NH 03458 Dept. V05

His book, then, is a discussion of the computer world from the standpoint of its potential and actual impact on society. Written to the level of the educated layman, this book shows what computers actually do, and then goes on to describe his concerns for a technology which has the power to radically change society. He suggests that the changes to come from the invention of the computer will be as profound as those which took place with the invention of the clock - that society will be irreversibly changed, for better or for worse, by computers.

When this book was published in 1976 it caused quite a stir in the computer science community. One of the topics discussed widely was Weizenbaum's description of the prototypical computer enthusiast, functioning as a "compulsive gambler" in a self-made universe: "Their rumpled clothes, their unwashed and unshaven faces, and their uncombed hair all testify that they are oblivious to their bodies and to the world in which they move. They exist, at least when so engaged, only through and for the computers. These are computers bums, compulsive programmers. They are an international 'phenomenon."

Hmmm!

While Weizenbaum's book preceded the advent of personal computers, it stands as a powerful work certain to stimulate much thought and conversation between you and your friends.

A more recent book, which concentrates on the personal computer revolution, is *Running Wild, The Next Industrial Revolution,* by Adam Osborne (Osborne/McGraw-Hill, Berkeley, CA, ISBN-0-931988-28-4). This book gives an exciting glimpse of the roots of the personal computer revolution and is concerned with the following major idea: the microprocessor was so radical an invention that established computer companies could not respond quickly to bring small distributed computing power to the public. It took new ideas and new lean companies to see the potential which lay in this technology.

Osborne goes on to suggest that if the big companies couldn't see the potential for this technology, then how can the bulk of the people in society understand and cope with its implications. His message is that the microprocessor has spawned a revolution, rather than it being part of an evolutionary development. A result of this thinking is Osborne's fear that blue collar workers will be displaced by robots soon, and that half of today's jobs will change drastically within the decade. The microelectronics revolution carries with it both promise and danger. This fast moving book is written for the layman and is fun to read.

Leaving the heavy stuff aside, we now come to a couple of books for those who want to learn to do some programming with no prior exposure to computers whatsoever.

One book we had to have for its title alone was *Computer Programming for the Complete Idiot,* by Donald McCunn (Design Enterprises of S.F., San Francisco, CA, ISBN-0-932538-04, $5.95). This book was written by someone who only recently became involved with personal computers, and who realized the need for a book to help other neophytes cope with the idiosyncracies of sending instructions to a machine. Written from the standpoint of a TRS-80 user, most of his material is of relevance to users of the various 6502-based machines as well. He carefully illustrates the importance of proper syntax in communicating with computers and then goes on to illustrate the creation of a program by carefully constructing a small payroll program step-by-step. The choice of a mundane topic like payroll may not seem as exciting to you as a program to play Space Wars, but a large number of potential computer users seem to be concerned with having these machines do something "useful"; so from their standpoint, the choice seems pretty good. This is the type of book you can leave beside your computer to let your friends explore computer programming for themselves. If you can't find this book locally, it can be ordered from the publisher (P.O. Box 27677, San Francisco, CA 94127) for $5.95 plus $1.00 for shipping and handling.

Another fairly recent book that is very well written is *Basic and the Personal Computer,* by Thomas Dwyer and Margot Critchfield (Addison-Wesley Publishing Company, Reading Massachusetts, ISBN-0-2-1-01589-7). This book is published as part of Addison-Wesley's "Joy of Computing Series" and it assumes no prior computer experience on the part of the reader. Starting off with a straightforward description of computer jargon, along with a brief overview of the components which make up a computer, the book quickly moves to its main thrust: software. Rather than use one large program to illustrate various programming techniques, Dwyer and Critchfield create a large number of small programs which not only help to get various programming ideas across, but which are of interest in their own right. This is an exceptionally readable book which is made all the more enjoyable by the whimsical illustrations sprinkled throughout the text.

We could go on and on and perhaps we should publish a bibliography next time - what do you think? Anyway, we hope that these books are of interest to you.

Next time we might talk about the relation between Goedel's Incompleteness Theorem, Star Trek - The Movie, and the proof that machines cannot "think". How about it? If this sounds interesting, let us know. We welcome suggestions for topics, and more importantly, we want you to tell us how you feel. As we said before, this is your column as much as it is ours. ©

An Introduction to Small Business Software for the PET*. II.

Can DR. DALEY's offer a better Mailing List Maintenance System?

You've seen them all! Every software supplier offers a mailing list system of some sort or another. Each of them has some advantages and some disadvantages over the others.

So when DR. DALEY's decided to offer a mailing list we felt that it had to offer some other advantages over all of the others. We have offered—and sold some—mailing list systems before, but these offer few things that makes them unique.

SERIOUS BUSINESS

When you wish to purchase a software system for any business purpose you need to give it serious and thorough consideration. What do you wish to accomplish with the software? What are your needs? How can a computer assist you in filling these needs? We have asked these questions numerous times to people who do mailings with lists in the size range of 500 to 15,000 names. The result was unanimous: everyone has different information needs. This, of course, means that everyone who buys a mailing list system, or any other business software, must find a program that comes closest to his needs. This is a time consuming, expensive task. We've talked with businessmen who have become frustrated with this process and are ready to throw in the towel. Another option is to hire a programmer to write the software for you or to write your own. This can cost more than the cost of the computer.

The last option is to find prepackaged software which each individual user can easily configure to his own needs. This would allow each business to customize its own computer maintained mailing list files to, as closely as is possible, parallel the current mailing list operation. Until now, this option has been virtually impossible to fulfill, from any software publisher.

IMPLEMENTATION

Our computerized mailing list system is designed to be easy for you, the user, to be able to easily configure your files to contain information in much the same way as you currently are doing. This means less of the pain and anguish that frequently accompanies computerization.

During the programming the author was in frequent contact with potential end users. The main thought during the development phase was to make the operation easy to understand, yet powerful enough to handle the job. Give the user as many options as is feasible, with the flexibility to make the greatest possible use of the file information. Finally, be sure that the capacity of the system is sufficient to allow most any business to make use of it.

The final version will allow records of 117 USABLE characters in length with a maximum of 15 fields within each record. It also allows reasonably large capacity with multiple diskette (maximum of 100 diskettes on a 32K PET or CBM) files and up to 1340 records per diskette.

WHAT ABOUT SORTING?

We hear this question most frequently from you. This is because sorting is the operation that divides the MAILING LIST system from any mailing list system. Why sorting? Well it is the way that the user can do such things as selective mailings to groups with common characteristics. This could include regional mailings, mailings to customers of a particular product, mailings to purchasers or to prospective customers, etc., etc. Or you might wish to make any possible combination of these categories.

Try to do this on most ordinary mailing list programs. You simply can't do it with most of the offerings on the market today.

This sorting is done by a "wild card" type of sort. This means that you can specify the contents of any portion of a field for a match and the computer will take any match for the rest of the field. This type of sort is best illustrated with the following examples:

A sort key can be: **R**1
    Matches with FORT#1
    and T4R321
    and %/R@31

Our system allows this type of sorting using up to three fields within each record. Thus you should be able to retrieve almost any conceivable subset of the files.

File organization is done using two of the fields as sort keys. This again is user selectable. You could, for example, specify that you wish the file to be in ZIP CODE sequence or in alphabetical sequence and all records within the file will be sequenced with that field. There is also a second sort field which is used to sequence the file where the first field is the same.

WHAT ABOUT LABELS?

We hear this one almost as often as the sorting. Well, here this is up to you. You can, at the time you print labels, choose the layout of the labels, you can also choose the number of labels per line. If you wish to have a four line address and printed four records wide you can do it.

WHAT ABOUT EDITING?

Editing is accomplished at several points in the program. These are at the time of entry, before saving the records to the file and from the disk file. You can easily modify any record at any of these points.

This does not really cover all of the operations on the files. Space simply does not allow a more complete description of the user oriented approach of the program.

We asked the question: Can we offer a better mailing list system? You bet we can! It's here now.

HARDWARE REQUIREMENTS

At present this requires a Commodore PET or CBM computer with a dual disk drive and a printer. It is set up to work with the Commodore printer or with most any other printer. Watch for these programs to be introduced for use with other types of popular microcomputers. The APPLE II version will be available about June 1, 1980. Watch for it!

ORDERING

At the present time many Commodore dealers do not carry our software. Thus you will most likely need to either persuade them to order for you or calling us directly at (616) 471-5514 anytime between noon and 9 p.m. Eastern time Sunday through Thursday. For only $99.95 plus four percent tax in the state of Michigan, you get this powerful, field tested, fully documented program packaged in a convenient three-ringed binder.

INVENTORY

We must add this note. There is too little space to allow us to describe the INVENTORY system adequately here. It offers the same flexibility as does the MAILING LIST described above, but we can't tell you much more. Write or call for details. It also is priced at $99.95.

Charge to your MC/VISA

master charge THE INTERBANK CARD   VISA

DR. DALEY's Software
425 Grove Ave., Berrien Springs, MI 49103
Phone (616) 471-5514
Sun.-Thurs. noon to 9 p.m., Eastern Time

*Watch for it on the APPLE II.

Formation of a COMPUTE Column for the Handicapped

Susan Semancik,
The Delmarva Computer Club

The Delmarva Computer Club is thankful for the many encouraging and enthusiastic letters we have received from people all over the country concerning our activities. We apologize for the length of time it is taking us to respond to some of the letters, but we are working on answering each letter on an individual basis. At the rate mail has been coming in lately, however, we will soon be spending more time answering letters than working on those programs for which the Club was formed!

The Editor of COMPUTE magazine has provided us with an opportunity to answer some of the common requests of these letters through his magazine and at the same time to provide a much needed "clearing-house" for information concerning help for the handicapped. In doing this, we are not trying to discourage correspondence, but are hopeful that answers to many similar requests can be quickly given through the magazine and that people needing special resources for the handicapped can be matched with those people who possess the programs, equipment, or innovative ideas to help the handicapped.

We encourage people and organizations to write to us concerning either their specific needs or their specific resources available. Even people with ideas that they themselves cannot implement should write in about them, and perhaps other people with either the needed ability or capital can join together to help make the ideas into a working reality to improve the quality of life for the handicapped. We would appreciate your including in your letters permission to publish in this column the information you give, for we will only make public those requests and offers for which permission has been granted.

We also plan, through this column to keep interested people informed of the progress we are making in specific program developments in the Club. Thank you again for your interest. We hope that you will all continue to respond, and will welcome this opportunity to openly exchange information.

Response to requests for the Manual Alphabet Tutorial Program on a PET

Our recent article on the PET Manual Alphabet Tutorial in COMPUTE's issue #3 resulted in many requests for obtaining the program. It is currently being updated and expanded, and will be available for purchase in approximately 6 weeks. We anticipate the selling price to be $49.95, with discount prices for those buying in quantity. If you wish to order this program, please advise us so we can place your name on our waiting list. We really want to attempt to make this as effective a program as possible and are considering a novel approach to encourage feedback on the usefulness of the program: If through use of this program, you are the first to make a particular suggestion to us for its improvement and we incorporate your ideas in a revised edition of the program, we will update your copy of the program accordingly at no extra charge.

Response to Delmarva Computer Club Membership and Newsletter Requests

Our membership is currently restricted in our By-Laws to the Delmarva Peninsula, and our newsletter is an "in-house" publication, not meant for outside distribution. We are currently publishing articles and announcements about the Club's activities in COMPUTE magazine, and would suggest that interested people can follow the development of specific areas of interest through COMPUTE. We would appreciate correspondence with anyone wishing to share ideas and insights with us.

Response to Requests for a Computer Program Exchange

Since we are a small club that has only recently formed, our program library is rather limited at this date. We will forward letters requesting program exchanges for particular computers to those members who have the same equipment, and an exchange on an individual basis may be possible. We would appreciate people sending us listings of programs they have available for exchange, and we will keep them informed as our library grows.

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

©

One of the more positive experiences of getting this magazine started and rolling has been the chance to plug into the "mainstream" of microcomputing activity. The mainstream, I've found, isn't by implication the most well known activity... it's simply individuals or groups working hard to develop new uses for their micros. The non-profit Delmarva group may feel geographically isolated, at times, from the "mainstream"; personally I think they're at its forefront. I'm pleased we've found each other, and especially pleased at their relationship with COMPUTE.

Robert Lock, Editor/Publisher

PRODUCT FEATURE

The CBM 8032

Larry Isaacs, COMPUTE Staff
Robert Lock, Editor

When Commodore introduced the CBM and Professional computers, we were introduced to an upgraded BASIC and an upgraded machine that was more convenient to use. The same can be said for the introduction of the new 80 column machine called the CBM 8000 Business Computer. There will be two models, the 8016 (16K of memory), and the 8032 (32K of memory). Both models come with the business style keyboard.

The CBM 8000 series comes with BASIC 4.0. This BASIC has the disk commands built in. (For a good description of these commands, see Chuck Stuart's article on BASIC 4.0 in the March/April COMPUTE.)

The major external difference is the 12 inch monitor. Case size is adjusted appropriately, and the now familiar green CRT is standard. Resolution is good, especially when combined with the TEXT and GRAPHIC functions described below. Internally, the new version of the Main Logic Board has moved the second cassette port to the right side of the board (near the rear memory expansion header). You can now plug in a second cassette without opening the case.

Since line wrap-around for the screen is no longer needed in the operating system, a number of new screen editor functions have been added. Here is a brief description of each of these.

First of all, the REPEAT, TAB, and ESC keys are functional on the CBM 8000. Holding down the REPEAT key in conjunction with another will cause that key to be entered repeatedly. Cursor controls, insert and delete, and the spacebar repeat automatically. The TAB key causes the cursor to skip to the next tab stop. Shift-TAB is used to set or clear the current cursor position as a tab stop. The ESC key functions when the screen editor is in quote mode, or there are outstanding inserts. Hitting the ESC key will exit the insert and quote modes.

In addition, there are 11 new functions available. These are BELL, TEXT, GRAPHIC, INSERT LINE, DEL LINE, SCROLL UP, SCROLL DOWN, ERASE BEGIN, ERASE END, SET TOP, and SET BOTTOM. The CBM 8000 won't have keys corresponding to these functions, but with some additional software they can be accessed from the keyboard using a sequence of keystrokes. Also, each of these functions has an assigned character value so they can be accessed under program control.

The BELL function uses CB2 sound, so you must add an external speaker if you want to use this one. Printing the appropriate character will sound the bell, as will reaching the 75th column when typing on the screen. The TEXT and GRAPHIC functions set the corresponding display mode. This involves more than selecting the character set. In text mode the lines are separated slightly to make them more readable. In graphic mode the lines are brought together so the graphics characters will touch.

The INSERT LINE, DEL LINE, SCROLL UP, and SCROLL DOWN functions operate as their name implies. When scrolling, blank lines are scrolled onto the screen, and lines scrolled off the screen are lost.

ERASE BEGIN erases, from the beginning of the line occupied by the cursor, up to and including the cursor position. The rest of the line and the cursor position are left unchanged. ERASE END erases from the cursor position to the end of the line, again leaving the rest of the line and cursor position unchanged.

SET TOP and SET BOTTOM are two functions which greatly enhance the usefulness of the 80 column screen. They allow you to define a portion of the 80 X 25 screen as a window. All screen editor and display functions operate only within this window, with screen contents outside the window left untouched. SET TOP sets the top left corner of the window to the current cursor position. SET BOTTOM sets the bottom right corner of the window to the current cursor position. After defining a window, you return to use of the full screen by hitting the HOME key twice.

These functions, when coupled with the 80 column screen, make the CBM 8000 series an aggressive entry into the small business market. Word Pro IV, the version developed for the 80 column machine, apparently includes the software necessary to interface with many of the Spinwriter special characters.

We expect commodore dealers to have units in the June/July time frame. Our word is that production is well along.

©

commodore
Authorized Dealers
APRIL, 1980

ALABAMA

ANDERSON
3156 University Dr NW
Huntsville 35805
205-539-3444

BERNEY OFFICE EQUIPMENT
1718 Vaughn Rd
Montgomery 36109
205-277-1601

COMPUTERLAND
3020 University Dr NW
Huntsville 35805
205-539-1200

5C CORPORATION
246 South Court St
Montgomery 36104
205-262-1961

THE LOGIC STORE
Lagoon Prk Shopping Vlge
Montgomery 36117
205-279-9797

RICKLES ELECTRONICS
2800 Meighan Blvd
Gadsden 35904
205-547-2534

ALASKA

KLOPF WORKS
SR 51367
Fairbanks 99701
907-452-8502

T & A SALES
2600 Denali St #101
Anchorage 99503
907-276-6911

ARIZONA

* ANCRONA
4518 E Broadway
Tucson 85711
602-881-2348

COMMERCIAL & HOME SYSTEMS
7840 E Broadway #113
Tucson 85710
602-886-6850

EVANS NEWTON
7335 E Butherus Dr #P
Scottsdale 85260
602-998-2777

INTERPORT ASSOCIATION
5801 E 22nd Street
Tucson 85711
602-747-5773

MICRO COMPUTER BROKERS
6819-P North 21st Ave
Phoenix 85015
602-242-9961

* MILLET'S ELECTRONICS
621 E Broadway
Mesa 85204
602-964-1600

ARKANSAS

ABACUS, INC
#10 Smith Bldg PO Box D
Conway 72032
501-329-9050

COMPUTERLAND
11121 Rodney Parham Rd.
Little Rock 72212
501-224-4508

* COMPUTER PRODUCTS UNLIMITED
2412 Broadway
Little Rock 72201
501-371-0449

DATA SHOPPE
236 Central Mall
Ft. Smith 72903
501-452-4948

DATA SHOPPE
1000-C E Maine
Van Buren 72956
501-474-8891

S.I.S., INC
2201 Washington Ave
Conway 72032
501-327-4471

CALIFORNIA

ACTIVE BUSINESS MCHN, INC
23 Beta Court
San Ramon 94583
415-837-1167

* ADVANCED COMPUTER PRODUCTS
1310-B E. Edinger
Santa Ana 92705
714-558-8813

* ANCRONA
11080 Jefferson Blvd.
Culver City 90230
213-390-3595

* ANCRONA
6060 Manchester Ave.
Los Angeles 900[illegible]
213-641-9322

* ANCRONA
1300 E. Edinger [illegible]
Santa Ana 927[illegible]
714-547-8424

* ANCRONA
1054 El Camino [illegible]
Sunnyvale 94[illegible]
408-243-4121

* APPLE TV
2606 S. Robertson Blvd.
Los Angeles 90034
213-559-4261

* AWARD ASSOCIATES, INC
127 First Street
Ripon 95366
209-527-0206

* BALDWIN MUSIC
1515 S Broadway
Santa Maria 93454
805-922-8265

* BUSINESS ENHANCEMENT CORP
1711 E. Valley Pkwy [illegible]
Escondido 92027
714-741-6335

BYTE SHOP
6041 Greenback Lane
Citrus Heights 95610
916-961-2983

BYTE SHOP
1122 "B" Street
Hayward 94541
415-537-2983

BYTE SHOP
16508 Hawthorne Blvd
Lawndale 90260
213-371-2421

BYTE SHOP
5453 E. Stearns St.
Long Beach 90815
213-597-7771

BYTE SHOP
11611 San Vicente Blvd
Los Angeles 90049
213-820-1524

BYTE SHOP
1415 El Camino Real
Mt. View 94040
415-969-5465

BYTE SHOP
2233 El Camino Real
Palo Alto 94306
415-327-8080

BYTE SHOP
123 Yorba Linda Blvd.
Placentia 92670
714-524-5380

BYTE SHOP
8038 Clairemont Mesa Blvd.
San Diego 92111
714-565-8008

BYTE SHOP
[illegible] Monterey St
San Luis Obispo 93401
805-543-9310

BYTE SHOP
3400 El Camino Real
Santa Clara 95051
408-249-4221

BYTE SHOP
1555 Morse Ave.
Ventura 93003
805-647-8945

BYTE SHOP
2989 N. Main St.
Walnut Creek 94596
415-933-6252

* CHANNEL DATA SYSTEMS
5960 Mandarin Ave.
Goleta 93017
805-964-8695

* CITATION SYSTEMS, INC
[illegible]
Palo Alto [illegible]
[illegible]

COMPUTER CONNECTION
214 California St
San Francisco 94104
415-781-0200

COMPUTER AGE, INC
4688 Convoy St. #105
San Diego 92111
714-565-4042

THE COMPUTER CORNER
1317 McHenry Ave.
Modesto 95351
209-529-9967

COMPUTER FORUM
14052 E. Firestone Blvd.
[illegible]
[illegible]

COMPUTERLAND
[illegible] El Camino Real
Belmont [illegible]
[illegible]

COMPUTERLAND
6743 Dublin Blvd
Dublin 94566
415-828-8090

COMPUTERLAND
2992 Navajo [illegible]
El Cajon [illegible]
[illegible]

COMPUTERLAND
11074 San Pablo Ave.
El Cerrito 94530
415-233-5010

COMPUTERLAND
22634 Foothill Blvd.
Hayward 94542
415-538-8080

COMPUTERLAND
6840 La Cienega Blvd
Inglewood 90302
213-776-8080

COMPUTERLAND
4546 El Camino Real
Los Altos 94022
415-941-8154

COMPUTERLAND
24001 Via Fabricante
Mission Viejo 92691
714-770-0131

COMPUTERLAND
81 North Lake St
Pasadena 91101
213-449-3205

COMPUTERLAND
1537 Howe Ave. #106
Sacramento 95825
916-920-[illegible]981

COMPUTERLAND
289 E. Highlands Ave
San Bernardino 92404
714-[illegible]

COMPUTERLAND
4233 Convoy St.
San Diego 92111
714-[illegible]

COMPUTERLAND
2272 [illegible] St.
San Francisco 94114
415-[illegible]

COMPUTERLAND
117 Fremont St
San Francisco 94105
415-[illegible]

COMPUTERLAND
1077 [illegible] Sunnyvale Rd.
San Jose [illegible]
408-[illegible]

COMPUTERLAND
1930 4th Street
San Rafael 94901
415-459-1767

COMPUTERLAND
223 S. Broadway
Santa Maria [illegible]
805-928-1919

COMPUTERLAND
611 Fifth St.
Santa Rosa 95404
707-528-1775

COMPUTERLAND
171 E. Thousand Oaks Blvd
Thousand Oaks 91360
[illegible]

COMPUTERLAND
104 W. [illegible]
Tustin [illegible]
[illegible]

COMPUTERLAND
[illegible] Ygnacio Valley Rd.
Walnut Creek 94598
415-935-6502

* COMPUTER PATHWAYS
12050 Nevada City Hwy [illegible]
Grass Valley [illegible]
[illegible]

COMPUTER [illegible]
[illegible] W. Sepulveda Blvd.
Torrance 90505
213-325-4754

THE COMPUTER ROOM
1500 Graham Hill Rd
Santa Cruz 95018
408-426-9473

* COMPUTER TIMESHARING CORP
5560 Ruffin Rd.
San Diego 92123
714-565-0505

* COMPUTER WORLD, INC
3808 W Verdugo Ave
Burbank 91505
213-848-5521

COMPUTER WORLD
15818 Hawthorne
Lawndale 90260
213-370-4842

* COMPUTER WORLD, INC
5848 Sepulveda Blvd
Van Nuys 91411
213-786-7411

* COMPUTER WORLD, INC
6789 Westminster Ave.
Westminster 92683
714-898-8330

* DATA EQUIPMENT SUPPLY
8315 Firestone Blvd.
Downey 90241
213-923-9361

EDUCATIONAL MICRO SYSTEMS
1527 28th St.
Sacramento 95816
916-452-3609

* GRASS VALLEY COMPUTER
18430 Jayhawk Dr.
Smartville 95977
916-272-2793

* INTL INST OF NATL HEALTH
7422 Mountjoy
Huntington Beach 92647
714-848-0774

JAY-KERN ELECTRONICS
1135 Columbus
Bakersfield 93305
805-871-5800

K-SMITH ASSOCIATES
11 Mast Court
Sacramento 95831
916-392-0317

* LANTOR, INC
8055 Manchester Ave.
Playa del Rey 90291
213-821-0642

LYON TV
19515 Village Dr.
Sonora 95370
209-532-4894

* MATTHEWS TV & STEREO
6040 Mission St
Daly City 94014
415-992-5400

* MR. CALCULATOR
160 E. El Camino Real
Mt. View 94040
415-962-0335

[illegible] CALCULATOR
55 Third St
San Francisco 94108
415-543-1541

* MR. CALCULATOR
318 Town & Country Village
San Jose 95128
[illegible]710

[illegible]PUTER WAREHOUSE
[illegible] LaHabra Way
Sacramento 95825
916-322-3290

MICRO DISTRIBUTORS
2737 Croddy Wy, Unit E
Santa Ana 92704
714-641-0205

PARADYME CONSUMER ELECTRONICS
404 Second St.
Davis 95616
916-785-5210

PASCO SCIENTIFIC
1933 Republic Ave.
San Leandro 94577
415-351-1501

PC COMPUTERS
10166 San Pablo Ave
El Cerrito 94530
415-527-6657

* PROGRAMMABLE ELECTRONICS
1748 W. Chapman Ave.
Orange 92668
714-978-6587

RADIO MART
1075 Cypress Ave.
Redding 96001
916-241-3000

* Denotes dealer buying directly from Commodore

Authorized Dealers

SYSTEMS FORMULATE CORP
39 Town & Country Village
Palo Alto 94301
415-326-9100

COLORADO

AMPTEC
2310 Providence Circle
Colorado Springs 80909
303-597-5384

AMPTEC
5975 No. Broadway
Denver 80216
303-571-0833

BYTE SHOP
3101 Walnut St
Boulder 80301
303-444-6550

BYTE SHOP
300 E. Foothills Pkwy
Ft. Collins 80525
303-223-4000

BYTE SHOP
970 So. Oneida
Denver 80224
303-399-8995

COMPUTERLAND
8749 Wadsworth Blvd.
Arvada 80005
303-420-1877

COMPUTERLAND
4543 Templeton Gap Rd.
Colorado Springs 80909
393-574-4150

COMPUTERLAND
2422 S. Colorado Blvd.
Denver 80222
303-759-4685

MICRO COMPUTER INDUSTRIES
1520 E. Mulberry, Suite 240
Ft. Collins 80524
303-221-1955

MICRO WORLD ELECTRONIX
6340 W. Mississippi
Lakewood 80226
303-936-4407

NEIGHBORHOOD COMPUTER STORE
13045 W. Alameda Pkwy.
Lakewood 80215
303-988-9140

CONNECTICUT

COLONIAL ENTERPRISES
360 Broad Street
New London 06320
203-442-2798

COMPUTERLAND
1700 Post Rd. Heritage Sq
Fairfield 06430
203-255-9252

COMPUTERLAND
55 Pratt Street
Hartford 06103
203-727-1857

CONNECTICUT MICROCOMPUTER
150 Pocono Road
Brookfield 06804
203-775-9659

HAASE OFFICE MACHINES
908 Washington Blvd
Stamford 06901
203-359-4880

MULTI BUSINESS COMPUTER SYSTEMS
28 Marlborough St.
Portland 06480
203-342-2747

DELAWARE

COMPUTERLAND
Astro Shopping Center
Newark 19771
302-738-9656

DISTRICT OF COLUMBIA

COMPUTER EMPORIUM
1990 K Street NW
Washington DC 20006
202-466-3367

FLORIDA

ACCURATE COMPUTING SERVICE
785 N. Lake Blvd.
North Palm Beach 33408
305-842-4900

* COMPUTER CENTER
6578 Central Ave.
St. Petersburg 33707
813-343-1396

COMPUTERLAND
500 E. Spanish River Blvd
Boca Raton 33432
305-368-1122

COMPUTERLAND
274 Alhambra Circle
Coral Gables 33134
305-442-4112

COMPUTERLAND
3963 N. Federal Hwy
Ft. Lauderdale 33308
305-566-0776

COMPUTERLAND
2777-6 University Blvd
Jacksonville 32217
904-731-2471

COMPUTERLAND
7374 S. Tamiami Trail
Sarasota 33581
813-921-7800

COMPUTERLAND
1520 E. Fowler Ave.
Tampa 33612
813-971-1680

COMPUTERLAND
4275 Okeechobee Blvd.
West Palm Beach 33409
305-684-3338

COMPUTER SPECIALTY
3820 Minton Road
Melbourne 32901
305-727-5147

* COMPUTERS FOR YOU
3608 W. Broward Blvd.
Ft. Lauderdale 33312
305-581-8945

[illegible]
Miami [illegible]
305-261-7866

[illegible]
Gainsville [illegible]
904-376-6066

[illegible]
305-[illegible]-1010

FOCUS SCIENTIFIC
1601 Biscayne Blvd.
Miami 33132
305-358-3948

MIDWEST CORP.
3341 NW 82nd Ave.
Miami 33122
305-592-5355

[illegible]
Clearwater [illegible]
813-443-5073

ORLANDO ELECTRONICS CO
2356 W. Oak Ridge Road
Orlando 32809
305-855-1010

PROFIT CONCEPTS, INC
3300 University Dr
Coral Springs 33065
305-753-8303

UNITED SYSTEMS & SOFTWARE
P. O. Box 117
Winter Park 32789
305-647-2251

GEORGIA

ACTION BUSINESS MACHINES
1684 N. Atwood Drive
Macon 31204
912-474-3333

* ANCRONA
3330 Piedmont Rd.
Atlanta 30305
404-261-7100

ATLANTA COMPUTER MART
5091-B Buford Hwy
Atlanta 30340
404-455-0647

COMPUTERLAND
2423 Cobb Parkway
Smyrna 30080
404-953-0406

THE LOGIC STORE
3050 Macon Road
Columbus 31906
404-568-0197

MERCHANT'S BUSINESS MACHINES
18 Executive Dr. NE
Atlanta 30329
404-633-1027

PERIMETER OFFICE SUPPLY & EQUIP, INC.
6950 Peachtree Ind. Blvd., Suite F
Norcross 30071
404-449-5832

HAWAII

COMPUTERLAND
567 S. King St.
Honolulu 96813
808-521-8002

IDAHO

* ELECTRONIC SPECIALTIES
6411 Fairview Ave.
Boise 83704
208-376-5040

* WORLD TOY & HOBBYCRAFT
7820 Fairview Ave.
Boise 83704
208-376-3581

ILLINOIS

* APPLETREE STEREO
1022 W. Lincoln Hwy
DeKalb 60115
815-758-2442

* APPLETREE STEREO
117–119 E. Beaufort St.
Normal 61761
309-452-4215

* APPLETREE STEREO
1645 N. Alpine
Rockford 61107
815-226-9826

ARPHAX CORPORATION
1422 N. E. Adams
Peoria 61603
309-676-1047

[illegible]
120 First St.
Dixon 61021
815-288-3311

BYTE SHOP
[illegible]
Champaign [illegible]
217-352-2323

BYTE SHOP
[illegible]
312-[illegible]

BYTE SHOP
[illegible] W. Dempster St.
Skokie 60076
312-673-3550

CHICAGO SYSTEMS, INC
2200 Lawrence Ave.
Chicago 60625
312-561-6137

COMPUTERLAND
[illegible]
Arlington Heights [illegible]
312-255-6488

COMPUTERLAND
136 W. Ogden Ave.
Downers Grove 60515
312-964-7762

COMPUTERLAND
1500 S. Lake St.
Mundelein 60060
312-949-1300

COMPUTERLAND
9511 Milwaukee Ave.
Niles 60648
312-967-1714

COMPUTERLAND
10935 S. Cicero Ave
Oak Lawn 60453
312-422-8080

COMPUTERLAND
4507 N. Sterling
Peoria 61614
309-688-6252

* FINANCIAL DYNAMICS
530 Park Ave.
River Forest 60305
312-771-5441

* KAPPEL'S COMPUTER STORE
125 E. Main St.
Belleville 62220
618-277-2354

* LILLIPUTE COMPUTER MART
4446 Oakton St.
Skokie 60076
312-674-1383

MICOBUS
6146 N. Lincoln Ave.
Chicago 60645
312-583-8358

* ORCUTT BUSINESS MACHINES
431 First Street
LaSalle 61301
815-224-2774

* RUHL & ASSOCIATES
24 W. Stevenson
Freeport 61032
815-235-7800

* SCHAFF SYSTEMS & ENGINEERS
Rts. 25 & 62
Algonquin 60102
312-658-5013

* STEREOTRONIC INDUSTRIES
Wadsworth Rd. & North Ave.
Zion 60099
312-336-2222

SYSTEMS, INC
2200 W. Lawrence Ave.
Chicago 60625
312-561-6137

INDIANA

* AUDIO SPECIALISTS
405 N. Michigan
South Bend 46601
219-234-5001

* COMMUNICATIONS ELECTRONICS
2204 Grand Ave.
Connersville 47331
317-825-6893

* THE COMPUTER CENTER
51591 U.S. 31 North
South Bend 46637
219-277-4655

COMPUTERLAND
[illegible] Place
[illegible] 46410
219-769-8020

* FT. WAYNE ELECTRONICS
[illegible] Ave.
Ft. Wayne 46803
219-423-3422

* GRAHAM ELECTRONIC SUPPLY
[illegible]
317-[illegible]

* [illegible] SUPPLY
133 S. Pennsylvania St.
Indianapolis [illegible]
317-634-8202

* GRAHAM ELECTRONIC SUPPLY
405 North Street
Lafayette 47902
317-423-5564

* STEWART BUSINESS MACHINES
[illegible] Broadway
[illegible]

IOWA

* THE COMPUTER CENTER
302 Commercial
Waterloo 50701
319-232-9504

CYBERIA, INC.
2330 Lincoln Way
Ames 50010
515-292-7634

THE MEMORY BANK
4128 Brady St.
Davenport 52806
319-386-3330

KANSAS

CENTRAL KANSAS OFFICE SYS
234 W. 4th St.,
Hutchinson 67501
316-662-0513

COMPUTERLAND
10049 Santa Fe Drive
Overland Park 66212
913-492-8882

COMPUTER SYSTEMS DESIGN
906 N. Main
Wichita 67214
316-265-1120

COMPUTER VIDEO ROOM
7105 W. 105th St.
Overland Park 66212
913-648-7105

MAIN ELECTRONICS
225 Ida
Wichita 67211
316-267-3581

PERSONAL COMPUTING CENTER
3819 W. 95th St.
Overland Park 66212
913-649-5942

KENTUCKY

* BARNEY MILLER'S INC
232 E. Main St.
Lexington 40507
606-252-2216

COMPUTERLAND
10414 Shelbyville Rd
Louisville 40223
502-245-8288

MICROTECH, INC
1127 S. 6th St.
Louisville 40203
502-587-8099

LOUISIANA

COMPUTER PLACE
3340 Highland Road
Baton Rouge 70812
504-387-0072

COMPUTER PLACE
1904 Pinook Rd. #202
Lafayette 70508
318-232-4097

COMPUTER SHOPPE
12-A W. Bank Expressway
Gretna 70053
504-366-0687

* COMPUTER SHOPPE
3225 Danny Park
Metairie 70002
504-454-6600

EXPANDED TECHNOLOGIES, INC
3883 Greenway Place
Shreveport 71105
318-[illegible]-5144

* FREEMAN ELECTRONICS
708 N. 7th St.
West Monroe 71291
318-388-2312

VIDEO SPECTRUM
6600 Veterans Memorial Blvd.
Metairie 70003
504-885-6527

MAINE

C.N.A
37 High St
Lewiston 04240
207-784-6961

MARYLAND

COMPUTERLAND
16065 Frederick Road
Rockville 20855
301-948-7676

COMPUTERS, ETC
13-A Allegheny Ave
Towson 21204
301-296-0520

MAC'S MERCHANDISE MART
7140 Fairbrook Rd
Baltimore 21207
301-298-0473

* THE MATH BOX, INC
2621 University Blvd. W.
Wheaton 20902
301-933-8770

YOUR OWN COMPUTER, LTD
10678 Campus Way South
Largo 20870
301-350-6680

MASSACHUSETTS

COMPUMART CORP
270 Third St
Cambridge 02142
617-491-2700

COMPUTER CITY
5 Dexter Row
Charleston 02129
617-242-3350

COMPUTERLAND
214 Worcester St
Wellesley 02181
617-235-6252

COMPUTER SHOP 11000011
288 Norfolk St
Cambridge 02139
617-661-2670

COMPUTER VILLAGE
66 Myron St
West Springfield 01089
413-739-3809

* Denotes dealer buying directly from Commodore

Authorized Dealers

* NEECO (New England Electronics)
679 Highland Ave.
Needham 02192
617-449-1760

* RETAIL COMPUTER CENTER
455 Center St.
Ludlow 01056
413-589-0106

THE SOUND COMPANY
Fairfield Mall
Chicopee 01020
413-593-5330

THE SOUND COMPANY
447 Sumner Avenue
Sprinfield 01108
413-736-3626

THE SOUND COMPANY
The Westfield Shops
423 E. Main St.
Westfield 01085
413-568-1160

MICHIGAN

THE COMPUTER CENTER
28251 Ford Rd.
Garden City 48135
313-422-2570

* COMPUTER HOUSE
1407 Clinton Road
Jackson 49202
517-783-5343

COMPUTERLAND
2927 28th St. SE
Kentwood 49508
616-942-2931

COMPUTERLAND
301 S. Livernois
Rochester 48063
313-652-9000

COMPUTERLAND
29673 Northwestern Hwy
Southfield 48034
313-356-8111

* COMPUTERMART OF ROYAL OAK
560 W. 14 Mile Road
Clawson 48017
313-288-0040

* COMPUTRONIX
423 Saginaw Rd.
Midland 48640
517-631-8060

HILTON FROST, INC.
162 Apple Ave.
Muskegon 49442
616-722-6757

INTERLINK
101 North Maplewood
Berrien Springs 49103
616-471-2144
616-471-3126

* MAIN SYSTEMS, INC
1161 N. Ballenger Hwy # 8
Flint 48504
313-232-3130

MAIN SYSTEMS, INC
1324 Lake Drive, SE
Grand Rapids 49506
616-454-0298

MAIN SYSTEMS, INC
2929 Covington Court
Lansing 48912
517-485-3910

MATRIX, INC
1241 N. Main
Ann Arbor 48104
313-663-6677

NATIONAL MICRO
1094 Third St.
Muskegon 49440
616-722-2929

* NEWMAN COMPUTER EXCHANGE
1250 N. Main Street
Ann Arbor 48107
313-994-3200

MINNESOTA

* COMPUTER DEPOT
3815 W. 70th St.
Minneapolis 55436
612-927-5601

* COMPUTER DEPOT
7625 Bush Lake Rd.
Minneapolis 55435
612-831-2088

COMPUTERLAND
8070 Morgan Circle Dr
Bloomington 55431
612-884-1474

* DIGITAL DEN
2138 Burnsville Center
Burnsville 55337
612-770-3424

* DIGITAL DEN
1009 Maplewood Mall
Maplewood 55109
612-770-3975

* MINNESOTA MICRO SYSTEMS
514 Cedar Ave. South
Minneapolis 55454
612-338-5604

* SCHAAK ELECTRONICS
1415 Mendota Heights Rd.
St. Paul 55120
612-454-6830

ZIM COMPUTERS
5717 Xeres Ave
Brooklyn Center 55429
612-560-0336

MISSISSIPPI

MISSISSIPPI MICROS INC.
Mart 51–1700 Terry Rd.
Jackson 39204
601-948-7846

MISSOURI

* COMPUTER COUNTRY
235 Dunn Road
Florissant 63031
314-921-4433

* COMPUTER COUNTRY
4479 Lemay Ferry Rd
St. Louis 63129
314-487-2033

* COMPUTER [illegible]NTRY
808 [illegible]
St. Lo[illegible]01
314-2[illegible]

COMPU[illegible]
1214 S. [illegible]
Independe[illegible]
816-461-6502

COMPUTERLAND
11990 Dorsett Road
Maryland Heights 63043
314-567-3291

* COMPUTER MART
[illegible]915 Noland Road
[illegible]endence 64055
[illegible]5005

[illegible]OOM
181[illegible]
Kansas [illegible]
816-531-105[illegible]

GATEWAY OFFICE MA[illegible]
9505 Gravois
Affton 63123
314-631-0110

PARSONS ELECTRONICS
1059 Venture Dr.
St. Charles 63301
314-723-2227

MONTANA

BYTE SHOP
1201 Grand Ave Suite 3
Billings 59102
406-252-2299

* THE COMPUTER STORE
1216 16th St. W, # 35
Billings 59102
406-245-0092

NEBRASKA

AMERICAN COMPUTERS
4442 S. 84th
Omaha 68127
402-592-1518

COMPUTERLAND
11031 Elm St.
Omaha 68144
402-391-6716

DOUBLE E ELECTRONICS
280 N. 115th St.
Omaha 68154
402-334-7870

* MIDWEST COMPUTER
8625 "I" Street
Omaha 68127
402-592-3590

NEVADA

BYTE SHOP
4104 S. Keitzke Lane
Reno 89502
702-826-8080

* HOME COMPUTERS
1775 Tropicana
Las Vegas 89109
702-736-6363

INGENUITY, INC
1562 Linda Way
Sparks 89431
702-359-6671

SUPER SOUND
1727-C E. Charleston
Las Vegas 89104
702-384-1512

NEW HAMPSHIRE

COMPUTERLAND
419 Amherst
Nashua 03060
603-889-5238

L. D. ENTERPRISES
736 Third Ave.
Berlin 03570
603-752-1942

STILLGATE COMPUTER SYSTEMS
1 West Street
Keene 03431
603-357-3981

NEW JERSEY

COMPUTERLAND
Highway E65 Route 4
Paramus 07652
201-845-9303

COMPUTERLAND
74 Elm St.
Morristown 07960
201-539-4077

COMPUTERLAND
1442 E. Route 70
Cherry Hill 08034
609-795-5900

* THE COMPUTER NOOK
Pinebrook Plaza, Rt. 46
Pinebrook 07058
[illegible]

[illegible]INDUSTRIES
12 Overlook [illegible]
[illegible]

MEDI[illegible]
[illegible] Highcrest Dr.
West Milford 07480
201-682-3341

MELAD ASSOCIATES
P. O. Box 159
Milltown 08850
201-828-3682

* S. M. I.
600 Washington Ave.
Carlstadt 07072
201-939-4800

* S.S. COMPUTER ENTERPRISES
224 E. Madison Ave.
Cresskill 07626
201-567-8076

STONEHENGE COMPUTER CO
89 Summit Avenue
Summit 07901
201-277-1020

THOR ELECTRONICS CORP.
321 Pennsylvania Ave.
Linden 07036
201-486-3300

NEW MEXICO

MICROCOMPUTER SERVICES
2201 E. 11th Street
Farmington 84701
505-327-6825

* OMEGA BUSINESS PROD. INC.
3100 Richmond NE
Albuquerque 87107
505-345-4545

NEW YORK

AMERICAN PERIPHERALS
3 Bangor Street
Lindenhurst 11747
516-226-5849

* A. S. D. OFFICE SYSTEMS
Van Wyck Plaza
Poughkeepsie 12603
914-473-9400

* ASI CELESTIAL COMPUTERS
N.Y. ASTROLOGY CENTER
127 Madison Avenue
New York City 10016
212-679-5676

* BC COMMUNICATIONS
207 Depot Road
Huntington Stn. 11746
516-549-8833

THE BINARY ORCHARD, INC
S-5854 Camp Road
Hamburg 14075
716-648-7167

BITS & BYTES
2800 Straight Rd.
Fredonia 14063
716-673-1682

BITS COMPUTERS
418 W. Dominic St.
Rome 13440
315-336-0266

COMPUCIRCUIT
222 E. Amherst St.
Buffalo 14214
716-838-2442

* THE COMPUTER FACTORY
485 Lexington Ave.
New York City 10017
212-687-5001

* CO[illegible]RAL STORE
10[illegible]ve.
Ly[illegible]563
51[illegible]0

* COM[illegible]SE, INC
721 [illegible]
Roch[illegible]
716-[illegible]

COM[illegible]
16[illegible] F[illegible]
B[illegible]
716-836-6511

COMPUTERLAND
[illegible]
Manhasset 11030
516-627-3640

[illegible]
[illegible]-1284

* COMPUTER STRATEGIES, INC
300 N. Main St.
Spring Valley 10977
914-356-7770

COMPUTER WORD, INC
422 Northern Blvd
Great Neck 11021
516-487-7830

* COMPUTER WORLD
519 Boston Post Rd.
Port Chester 10573
914-937-6662

COMTEK ELECTRONICS
2666 Coney Island
Brooklyn 11235
212-332-5933

DEJ VIDEO SERVICE, INC
840 Wellwood Avenue
North Lindenhurst 11757
516-957-8932

EXCEL COMPUTER SERVICES
3636 Hollend Ave.
Bronx 10467
212-652-4028

HONDERS COMPUTER EMPORIUM
57 North Street
Middletown 10940
914-343-4880

LOGICAL PLAYBACK
10 Grace Ave.
Great Neck 11021
212-624-9075

* MEIZNER BUSINESS MACHINES
4771 Boston Post Rd.
Pelham 10803
914-738-6000

MR. COMPUTER
Imperial Plaza
Wappingers Falls 12590
914-297-1223

* JOHN D. OWENS ASSOC
12 Schubert Ave.
Staten Island 10305
212-448-6283

SECURITY SYSTEMS INSTALLERS
86 Fair Street
Carmel 10512
914-225-5321

UNITED PHOTOCOPY CO
41 Union Square
New York City 10003
212-929-4826

* UPSTATE COMPUTER SHOP
629 French Rd., Campus Plaza
New Hartford 13413
315-733-9139

NORTH CAROLINA

BYTE SHOP
218 N. Elm St.
Greenboro 27401
919-275-2983

CAROLINA INFORMATION SYSTEMS
1213 Hillsborough St
Raleigh 27605
919-833-0210

COMPUTERLAND
3915 E. Independence Blvd.
Charlotte 28205
704-536-8500

DATA HELP
2716 Stewart Lane
Rocky Mtn. 27801
919-977-5517

NIBLOCK TYPEWRITER CO
675 W. 4th Street
Winston Salem 27101
919-722-7013

OHIO

ASTRO VIDEO ELECTRONICS
504 E. Main St.
Lancaster 43130
614-687-0629

[illegible]
[illegible]raine Rd
[illegible]ark 44126
[illegible]61

[illegible]
[illegible]00 Busch Blvd
Columbus 43229
[illegible]442

[illegible]LAND
[illegible]sch Road
Columbus 43229
614-888-2215

COMPUTERLAND
1288 Som Center Rd
Mayfield Heights 44124
216-461-1200

COMPUTERLAND
4579 Great Northern Blvd
North Olmsted 44070
216-777-1433

* GRAHAM ELECTRONIC SUPPLY
239 Northland Blvd.
Cincinnati 45215
513-772-1661

* INDUCTIVE COMPONENTS
1200 Ferris Rd.
Amelia 45102
513-752-4731

* McSHANE
123 W. Washington
Medina 44256
216-725-4568

* MICRO MINI COMPUTER WORLD
74 Robinwood Ave.
Columbus 43213
614-235-5813

OMICRON COMPUTER SYSTEMS
873 N. Monroe Drive
Xenia 45385
513-372-1612

RADIO COMMUNICATIONS
1995 Marion-Bucyrus Rd.
Marion 43302
614-382-4126

* UNIVERSAL AMATEUR RADIO
1280 Aida Dr.
Reynoldsburg 43068
614-866-4267

* Denotes dealer buying directly from Commodore

Authorized Dealers

OKLAHOMA

* AUDIO HORIZONS
3707 E. Frank Phillips
Bartlesville 74003
918-333-7748

* BRADFORD BROTHERS
507 S. Main
Tulsa 74103
918-584-4558

COMPUTERLAND
10621 North May Ave
Oklahoma City 73120
405-755-5200

COMPUTER SYSTEMS & SOFTWARE
928 N. York # 19
Muskogee 74401
918-687-3161

* TEL-COM, INC.
727 N. Brauer Ave.
Oklahoma City 73106
405-232-2035

OREGON

* ANCRONA
1125 NE 82nd Ave
Portland 97220
503-254-5541

COMPUTERLAND
12020 SW Main St.
Tigard 97223
503-620-6170

* COMPUTER PATHWAYS
831 Lancaster Dr NE
Salem 97301
503-399-0534

* COMPUTER PATHWAYS UNLTD
2151 Davcor SE
Salem 97302
503-363-8929

PENNSYLVANIA

* A.B COMPUTERS
115 E. Stump Road
Montgomeryville 18936
215-699-8386

* CARR ELECTRONICS
Main & Reliance
Telford 18969
215-723-3831

CENTURY SYSTEMS, INC
5711 Sixth Ave.
Altona 16602
814-946-4361

* CHIPS 'N DIP
4820 Anne Lane
Sharpsville 16150
412-962-9682

THE COMPUTER HOUSE
1000 Greentree Road
Pittsburgh 15220
412-921-1333

COMPUTERLAND
4644 Carlisle Pike
Mechanicsburg 17055
717-763-1116

COMPUTERLAND
81 E. Lancaster Ave.
Paoli 19301
215-296-0210

COMPUTERLAND
1457 MacArthur Rd.
Whitehall 18052
215-776-0202

GLOSSER BROS
Franklin & Locust Sts
Johnstown 15901
814-536-6633

* MARKETLINE SYST[illegible]
2337 Philmont
Huntingdon V[illegible]
215-947-6670

* MICRO COMPUTE[illegible]
806 Penn A[illegible]
Sinking Spri[illegible]
215-670-130[illegible]

* MIGHTY BYTE [illegible]
537 Easton [illegible] Pl.
Horsham [illegible]
215-348-4[illegible]

PUERTO [illegible]

* MICRO CO[illegible]
1568 Av[illegible]
Caparra [illegible]
Puerto [illegible]
809-78[illegible]

SOUTH [illegible]

BYTE SHOP
1920 Blossom [illegible] Box 1544
Columbia 29206
803-771-7824

[illegible]

[illegible]

ULTRA SYSTEMS
210 East Kemp
Watertown 57201
[illegible]

[illegible]

[illegible]
Nashv[illegible]
615-329-1001

COMMUNICATIONS MARKETING, INC
2979 Sandbrook Ave.
Memphis 38116
901-332-0242

MICROSYSTEMS, INC
[illegible]

TEXAS

* ANCRONA
2649 Richmond Ave.
Houston 77098
713-529-3489

APPLICATIONS PLUS
8585 N. Stemmons Fwy.
Suite 1136
Dallas 75247
214-637-5410

BORDER ELECTRONIC SERVICE
P. O. Box 3804
El Paso 79923
915-532-2524

COMPUTERLAND
3300 Anderson Lane
Austin 78757
512-452-5701

[illegible]

COMPUTERLAND
6439 Westheimer
Houston 77057
713-977-0909

COMPUTERLAND
17647 El Camino Real
Houston 77058
713-488-8153

COMPUTERLAND
1535 Promenade Center
Richardson 75080
214-235-1285

THE COMPUTER SHOP
917 N. Judge Ely Blvd.
Abilene [illegible]
[illegible]

[illegible]
5135 Fredericksburg Rd.
San Antonio 78229
512-341-8851

COMPUTER SOLUTIONS
[illegible]

[illegible]
Arlington 76010
817-429-7988

DATA MATRIX
7401 Gulf Freeway, # 107
[illegible]

FMI SYSTEMS
7407 Brentfield
Dallas 75248

FNS COMPUTER SERVICES, INC
1911 N. Robison Rd
Texarkana 75501
[illegible]

[illegible]
Houston 77023
713-651-7038

MICRO MART
1015 Navarro
San Antonio 78205
512-222-1426

SORT-DATA CORP
104 S. Texas Ave. Suite A
Bryan 77801
713-822-7080

UTAH

BYTE SHOP
3616 W. 2100 South
Salt Lake City 84120
801-355-1041

COMPUTER DATA SYSTEMS
695 E. 1000 North
Logan 84321
801-753-6990

[illegible]MPUTER HUB
[illegible] S. State
[illegible]ray 84107
[illegible]-262-5416

[illegible]UTERLAND
[illegible] E. 2nd South
[illegible] Lake City 84111
[illegible]364-4416

[illegible]MPUTER WORKS
[illegible] State St
[illegible] 84601
[illegible]4-0204

[illegible] SHOP
[illegible] Washington Blvd.
[illegible] 84401
[illegible]-5244

[illegible] SHOP
[illegible]aday Blvd
[illegible] City 84117
[illegible]2629

VIRGINIA

COMPUTERLAND
8411 Old Courthouse Rd
[illegible]

[illegible]
703-698-5050

RMS TECHNOLOGY, INC
706 Industry Drive
Hampton 23661

[illegible] 115
[illegible]

TY[illegible]
1984 Chain Bridge Rd
McLean 22101
703-821-8333

WASHINGTON

AMERICAN MERCANTILE CO
[illegible]

COMPUTERLAND
14340 NE 20th
Bellevue 98007
206-746-2070

COMPUTERLAND
1500 S. 336th St. # 12
Federal Way 98003
206-838-9363

COMPUTERLAND
8791 S. Tacoma Way
Tacoma 98449
206-581-0388

COMPUTER SYSTEMS DESIGN, INC
1105 W. Lincoln, P O Box 735
Yakima 98907
509-453-2956

* MICRO COMPUTER CENTER
11822 NE 8th St.
Bellevue 98005
206-455-3710

* OMEGA NORTHWEST, INC
839 106th Ave. NE
Bellevue 98004
206-455-1138

* OMEGA NORTHWEST, INC
5421 196th SW
Lynwood 98036
206-775-7585

* OMEGA NORTHWEST, INC
1032 NE 65th
Seattle 98004
206-522-0220

PERSONAL COMPUTERS, INC
South 104 Freya
Spokane 99202
509-534-3955

PRITCHARD INSURANCE
180 S. Woodland Beach Rd.
Camano Island 98292
800-426-2303

WEST VIRGINIA

WISCONSIN

BYTE SHOP
6019 W. Layton Ave.
Greenfield 53220
414-281-7004

* COLORTRON
2111 Lathrop Ave
Racine 53405
414-637-2003

COMPUTERLAND
690 S. Whitney Way
Madison 53711
608-273-2020

COMPUTERLAND
10111 W. Capitol Dr
Milwaukee 53222
414-466-8990

* MADISON COMPUTER STORE
1825 Monroe St
Madison 53711
608-255-5552

PETTED
4265 W Loomis Road
P. O. Box 21851
Milwaukee 53221
414-282-4181

WYOMING

COMPUTER CONCEPTS
1104 Logan Ave
Cheyenne 82001
307-632-9132

* Denotes dealer buying directly from Commodore

commodore

USA Sales Office

Commodore Business Machines, Inc.
Computer Sales Division
950 Rittenhouse Road
Norristown, Pennsylvania 19401
215-666-7950

District Sales Offices

Western District
2955 N. Airway Avenue
Costa Mesa, California 92626
714-979-6307

Northwest District
3330 Scott Boulevard
Santa Clara, California 95051
408-727-1130

Midwest District
790 Maple Lane
Bensenville, Illinois 60106
312-595-5990

Northeast District
950 Rittenhouse Road
Norristown, Pennsylvania 19401
215-666-7950

South Central District
4350 Beltwood Parkway South
Dallas, Texas 75234
214-387-0006

Southeast District
5360 Snapfinger Woods Drive
Decatur, Georgia 30035
404-987-3311

COMMODORE BUSINESS MACHINES, INC.

Keeping Up The Payments

Robert Lock

Basically Useful BASIC

Programs furnished by Marvin L. DeJong, School of The Ozarks

I decided to use this highly useful set of programs furnished by Dr. Marvin L. DeJong to illustrate some points to you beginning Atari/Apple/PET Owners. Much of the Basic software we provide in this magazine is easily transferred for use on your own machine, even though it may be presented by a PET owner, an Atari owner, etc. These programs, developed on (I assume) the AIM-65, will run without revision on your Apple, Atari, or PET. Your results will differ slightly, a variation caused by your particular Basic interpreter, but the program code itself is transparent. In future issues, we hope to solicit the help of you readers in maintaining the flow of software for various machines. Thus, when we print an extensive program for PET that one of you sits down and modifies to run on your Apple, please send me a copy (at least the mods), so we can update in a later issue for Apple owners. I'd like to devote a section of the magazine to "listing reruns" if the demand warrants. One option, I suppose, would be to simply cover techniques for conversion, with occasional routines that provide some equivalency across machines. Please take the time to note your thoughts on this on the Editor's Feedback card in the center of the magazine. My thanks to Marvin for the programs.

A bit of "editorial license": When I keyed in these programs for reproduction, I tried to preserve their original flavor. You should note that I took the liberty of nudging lines that were too long over into a non-existent continuation line. If you're just starting out, don't try to enter the program that way... you'll have blanks in your output (or worse).

Example: Program 2, Line 60.

My line shows:

60 PRINT "ENTER YOUR INTEREST RATE IN
 PERCENT."

When you're keying in the *program*, you should type it like this:

60 PRINT "ENTER YOUR INTEREST RATE IN PERCENT."(press carriage return)

Clear enough?

```
 10 REM PROGRAM 1- CALCULATES THE BALANCE OF
    A LOAN AFTER N PAYMENTS.
 20 PRINT "ENTER YOUR CURRENT BALANCE."
 30 INPUT BAL0
 40 PRINT "ENTER THE NUMBER OF PAYMENTS."
 50 INPUT N
 60 PRINT "ENTER YOUR MONTHLY PAYMENT."
 70 INPUT PMT
 80 PRINT "ENTER YOUR ANNUAL INTEREST RATE
    IN PERCENT."
 90 INPUT APR
100 I=APR/1200
120 DUMY=BAL0
130 FOR J=1 TO N
140 BAL0=BAL0*(1+I)-PMT
150 NEXT J
160 NT=N*PMT-(DUMY-BAL0)
170 PRINT "YOUR BALANCE AFTER ";N;" PAYMENTS
    IS $";BAL0
180 PRINT "YOU PAID $";NT;" IN INTEREST."
190 END
ENTER YOUR CURRENT BALANCE.
?3000
ENTER THE NUMBER OF PAYMENTS.
?23
ENTER YOUR MONTHLY PAYMENT.
?143.79
ENTER YOUR ANNUAL INTEREST RATE IN PERCENT.
?10.8
YOUR BALANCE AFTER 23 PAYMENTS IS $30.3651184
YOU PAID $337.535119 IN INTEREST.
```

```
 10 REM PROGRAM 2- CALCULATES THE MONTHLY
    PAYMENT TO AMORTIZE A LOAN
 20 PRINT "ENTER THE NUMBER OF PAYMENTS."
 30 INPUT N
 40 PRINT "ENTER THE BALANCE OF THE LOAN."
 50 INPUT BAL0
 60 PRINT "ENTER YOUR ANNUAL INTEREST RATE IN
    PERCENT."
 70 INPUT APR
 80 I=APR/1200
 90 PMT=BAL0*I*(1/(1-(1+I)↑(-N)))
100 PRINT "YOUR PAYMENTS ARE $";PMT
110 NT=N*PMT-BAL0
120 PRINT "YOU PAY $";NT;" IN INTEREST."
130 END

ENTER THE NUMBER OF PAYMENTS.
?23
ENTER THE BALANCE OF THE LOAN.
?3000
ENTER YOUR ANNUAL INTEREST RATE IN PERCENT.
?10.8
YOUR PAYMENTS ARE $144.984191
YOU PAY $334.636391 IN INTEREST.
```

```
 10 REM PROGRAM 3- CALCULATES THE NUMBER OF
    PAYMENTS TO AMORTIZE A LOAN
 20 PRINT "ENTER THE CURRENT BALANCE."
 30 INPUT BAL0
 40 PRINT "ENTER YOUR MONTHLY PAYMENT."
 50 INPUT PMT
 60 PRINT "ENTER YOUR ANNUAL INTEREST RATE
    IN PERCENT."
 70 INPUT APR
 80 I=APR/1200
 90 N=-(LOG(1-BAL0/PMT*I))/LOG(I+1)
100 PRINT "YOU HAVE ";N;" PAYMENTS TO MAKE."
110 NT=N*PMT-BAL0
120 PRINT "YOU WILL PAY $";NT;" IN INTEREST."
130 END

?ENTER THE CURRENT BALANCE.
?3000
ENTER YOUR MONTHLY PAYMENT.
?143.79
ENTER YOUR ANNUAL INTEREST RATE IN PERCENT.
?10.8
YOU HAVE 23.2123282 PAYMENTS TO MAKE.
YOU WILL PAY $337.700665 IN INTEREST.
```

©

6502 SOFTWARE DESIGN

Leo J. Scanlon
Rockwell International

Editor's Note: At $10.50, this 270 page new release from Sams & Co. represents the first move into the 6502 by the Blacksburg group. My thanks to Leo for permission to reprint these interesting first chapter overviews. RCL

WHY THE 6502?

To understand where the 6502 fits in the microprocessor spectrum, a brief look must be taken at the evolution of 8-bit microprocessors. The first 8-bit microprocessor to make a significant impact on the industry was the 8008 produced by Intel Corporation. Fabricated with p-channel metal-oxide semiconductor (PMOS) technology, the 8008 is considered the foremost "first-generation" 8-bit microprocessor. The 8008 was designed with a calculator-like architecture, and had six scratch-pad registers, an internal stack register, and special instructions to perform input and output. In 1973, Intel Corporation introduced a "second-generation," silicongate, NMOS version of the 8008 microprocessor, and called it the 8080.

The 8080 is essentially an improved 8008, with more addressing, more instructions, and faster instruction times. The internal organization is better too, but the overall 8008 is historically the second-generation de facto standard in microprocessors; the circuit that many people think of first when microcomputers are mentioned. Intel Corporation got a head start on the industry with the 8008, and preserved it with the 8080 through the early 1970s. Until Motorola, Inc. introduced the 6800 microprocessor in 1974, Intel Corporation had virtually no competition.

Motorola, Inc. saw the tremendous microprocessor market potential evolving, and decided to make an entry of their own. They had essentially two ways to go: (1) they could challenge Intel Corporation on their own ground, by producing a new and improved 8080 (as Zilog, Inc. did in 1976 with the Z80), or (2) they could ignore that approach and design a more advanced microprocessor. Realizing that it would be extremely difficult to establish a strong market position (not to mention a leading position) by going after Intel Corporation with a "me too" product, Motorola, Inc. decided to challenge with a superior product.

The resulting product, the 6800 microprocessor, was organized along the lines of classic computer architectural concepts, with input and output devices accessed as memory. In the 6800 microprocessor, the load and store instructions used to access memory are the same instructions used to perform input (read) and output (write) operations on peripheral devices. This technique, called memory-mapped I/O, eliminates the performance bottlenecks that are associated with having to pass all the data handling and manipulation through a working register, as in the register-based architecture of the 8080.

The preceding brief overview was necessary in order to set to stage for introducing our subject microprocessor, the 6502. The 6502 device was designed by eight ex-employees of Motorola who saw that advances in processes, coupled with a few architectural and software changes, could result in a potentially highly marketable 6800-like microprocessor. They joined a calculator-chip company called MOS Technology.

The MOS Technology design team had two objectives in mind for their next-generation[1] microprocessor—low cost and high performance. Since there is a direct correlation between the manufacturering cost and the die size (the size of the piece of silicon that contains the transistors and resistors which make up the microprocessor), they reduced the complexity of the basic 6800 design as much as possible to minimize the amount of silicon required. Other design decisions included eliminating one of the two accumulators in the 6800 and its tri-state address output buffers. They also replaced the 16-bit index register of the 6800 microprocessor with two separate 8-bit index registers, and they discarded some of the lesser-used instructions of the 6800.

The elimination of instructions opened up some instruction-decode space and permitted the designers to provide the 6502 microprocessor with 13 addressing modes, 7 more modes than the 6800 device has. These modes give the 6502 device capabilities that are normally found only in larger computers. Additionally, the design team realized that although computers are binary machines, man is inherently a decimal-thinking animal, so they added a mode-selection instruction and control bit that allows the 6502 microprocessor to operate on either binary or decimal data. This means that the programmer does not have to remember to write in "decimal adjust" instructions after addition or subtraction operations. For electrical efficiency, the design team employed the newer depletion-load technology, which gives the 6502 clean switching characteristics, low-power

dissipation (250 mW typical for the 6502 versus 600 mW typical for the 6800), and good noise immunity.

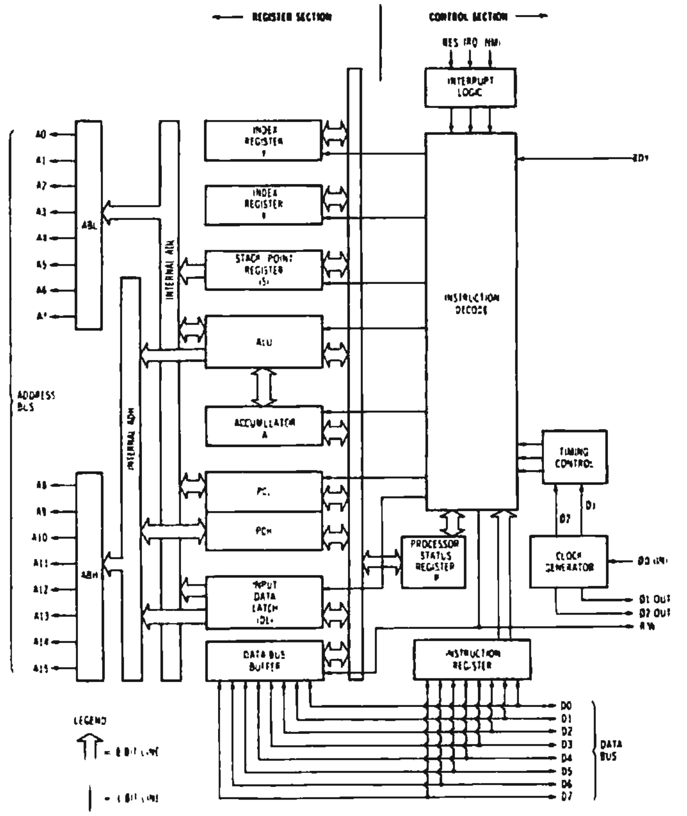
The 6502 device is one of 10 software-compatible microprocessors that MOS Technology introduced in 1975 as the 6500 Series. Through second-source agreements, the 6500 Series is also produced by Rockwell International and Synertek. All 10 microprocessors have the same instruction set and the same basic architecture, varying only in size and hardware options. The 6500 Series has been very popular since its introduction, and by the end of 1978, more 6500-Series microprocessors were being shipped than all other 8-bit microprocessors, including the 8080 and 6800.

Today, the best-selling 8-bit microprocessors are divided into two distinct families[2]—the 8080/Z80 family, with its register-oriented architecture, and the 6500/6800 family, with its memory-mapped architecture. Which architecture will be the more favored in the 1980s? It is impossible to know which at this time, but the manufacturers of the 6800 and 6500 devices are banking solidly on their opinion that the more minicomputer-like architecture holds the greatest potential for advanced circuits. Intel Corporation has not yet shown signs of sharing this opinion, but it may be significant to note that the latest 16-bit microprocessor from Zilog, Inc., the Z8000, represents a solid break with the 8080/Z80 design concept by including memory-mapped I/O.

THE 6502 MICROPROCESSOR

The 6502 microprocessor can be combined with memory and input/output integrated circuits to form a microcomputer. As the "heart" of the microcomputer, the 6502 regulates all operations of the microcomputer, based on the sequence of instructions (the program) that it is executing. The 6502 can execute 56 different types of instructions. The various combinations of addressing that are available for use by individual instruction types give the microprocessor a total of 151 executable instructions. The 6502 instruction set is described in detail in subsequent chapters of this book; for now, let us focus our attention on the internal organization (the architecture) of the 6502 and find out how it operates.

Fig. 1-1 is a block diagram of the internal architecture of the 6502. It shows the elements of the microprocessor and the buses by which they communicate with each other and with external circuits. The 6502 contains most of the control and decision-making logic, so only a few additional circuits are required to configure a small microcomputer system. One of the functions of this additional control logic is to provide the 6502 microprocessor with a clock signal that the internal clock generator will use to generate its two-phase system clock. The 6502 also requires a single +5 -volt dc power supply.



Courtesy Rockwell International

Fig. 1-1. Block diagram of the 6502 microprocessor.

All of the other inputs and outputs of the 6502 integrated circuit are compatible with standard transistor-transistor-logic (TTL).

MACHINE CODE AND ASSEMBLY LANGUAGE

For the 6502 microprocessor to perform a specific task, it must be programmed to do so. A program is nothing more than a sequence of instructions stored in sequential memory locations. The 6502 executes the program, one instruction at a time. It fetches an instruction from memory, decodes it, performs the decoded command, and then fetches the next instruction. This cycle is repeated until all instructions in the program have been executed.

What do these instructions look like? Since the 6502 microprocessor is simply a collection of electronic circuits (albeit in microscopic form), the instructions are composed of binary numbers (1's and 0's) that cause some internal electrical signals to be turned on, others to be turned off. The 6502 is an 8-bit microprocessor, so these binary instructions are comprised of multiples of 8 binary bits.

In early computers, all programming was done in the binary form, normally with switches controlling the individual bits—to set a bit to "1," turn the switch on; to reset a bit to "0," turn the switch off. But a string of 1's and 0's presents such a confusing mess that the computer industry soon realized the need for decimal loaders, which allowed the instructions to be written in decimal form. Decimal loaders were eventually replaced by hexadecmal loaders, which allowed the instructions to be written in hexadecimal form. Example 1-1 shows both binary and hexadecimal forms of typical program instructions.

Example 1-1. A Typical 6502 Program, in Binary and Hexadecimal Notation

Binary	Hexadecimal
10100101	A5
00100001	21
11000101	C5
00100000	20
10110000	B0
00101011	2B
10100110	A6
00101100	2C

Hexadecimal representation is some help to the programmer because it frees him from using all those error-prone 8-bit binary numbers. Further, hex instructions do not contain quite so many digits in them, making them somewhat easier to memorize. Unfortunately, though, a hexadecimal number gives no hint as to the function of an instruction. Does a "C5" instruction perform an addition, a subtraction, store a value in memory, or none of these? Even when using hexadecimal numbers, it is still difficult to program the 6502 microprocessor. Before you enter a hexadecimal number into the microcomputer, you would first have to find the instructions, that you want to store in the memory of the 6502, on a list provided by the microcomputer manufacturer. The appropriate hexadecimal number (op code) can then be found next to the instruction. The time spent in looking up the instructions and op codes could be very costly in developing a program, not to mention the possibilities of errors.

The next higher level of programming permits the programmer to write instructions in an abbreviated form, something closer to a human language, using abbreviations called mnemonics that can be correlated directly to the function of the instruction. A computer program can then be executed so that these mnemonics are actually converted to the sequence of 1's and 0's that the 6502 can execute. The program that converts these abbreviations into machine code (1's and 0's) is called an assembler, so this form of programming is called assembly language programming. An instruction that increments the X register by one has a hexadecimal form of E8 and an assembly language mnemonic of INX. Which do you think is easier to remember? Table 1-1 lists the assembly language mnemonics for several 6502 instructions.

How does the assembler translate instruction mnemonics to binary codes that the 6502 can execute? The assembler contains a large table (the permanent symbol table) that contains all the mnemonics (represented by strings of ASCII characters) and their binary equivalents. The assembler compares the mnemonic in your program (a string of ASCII characters) to each ASCII character string in the permanent symbol table. When a "match" occurs, the assembler fetches the binary code associated with the mnemonic, and uses this value during the assembly process. Therefore, the assembler translates the mnemonic INX (Increment X by 1) to E8, and the mnemonic CLC (Clear Carry flag) to 18. In this book, all example programs will be written using the standard mnemonics;

Table 1-1. Some 6502 Mnemonics

Mnemonic	Instruction
ADC	Add memory to accumulator with Carry
CLC	Clear Carry flag
INX	Increment index X by 1
LDA	Load accumulator with memory
TAX	Transfer accumulator to index X

these are the mnemonics that are defined in the literature of the manufacturers. ©

Thanks to these inexpensive Pet and TRS-80 memories and disk drives you can instantly increase your personal computer's capacity, capability and versatility.

Expand your Pet.

Disk Drives: Choose either double density 400k or dual head double density 800k on-line disk drive. Both with DOS operating system including 17 additional BASIC commands and random or indexed sequential access.

Exceptionally fast 5,000 bytes per second (including verify) transfer speed lets you load 20k program in just 3 seconds. Model 8k must have Expandamem™ to operate disk drives. All other models are plug compatible.

400k only $1295. 800k only $1695.

Expandamem™: Lets you expand the size and scope of your programs. 16k, 24k, and 32k internal memory boards give you 8k of assembly language subroutines accessed via the USR command.

16k only $425. 24k only $525. 32k only $615.

Software: Complete library available. Write for details.

Expand your TRS-80.

Disk Drive: Single density dual head 400k on-line disk drive is completely compatible with TRS-80 DOS and uses own controller. Only $1295.

IMMEDIATE DELIVERY! ORDER TODAY!

COMPUTHINK

965 W. Maude, Sunnyvale, CA 94086.
(408) 245-4033.

Please send me the following:

- ☐ Pet Disk Drive ☐ 400k ☐ 800k
- ☐ Pet Expandamem™ ☐ 16k ☐ 24k ☐ 32k
- ☐ Pet Software (Info only)
- ☐ TRS-80 400k Disk Drive
- ☐ Name of my nearest Computhink dealer.

Total enclosed: $________

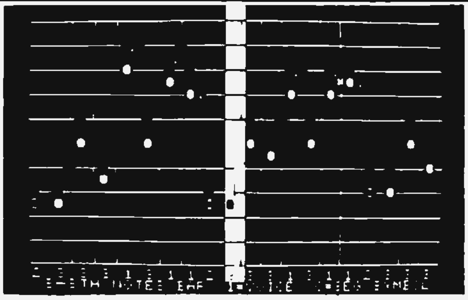
Check or Money Order only. Calif. Res. add 6% tax. Freight pre-paid in continental U.S.

Dealer Inquiries Invited.

Name ________

Address ________

City ________ State ________ Zip ________



FOUR PART HARMONY FOR THE PET

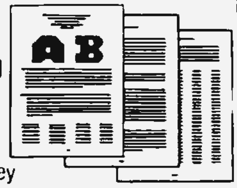
A-B Computers announces a combination system consisting of the KL-4M DAC Board and the Visible Music Monitor for Commodore PET-CBM computers The package enables PET users to easily create and play musical compositions of up to 4 parts

The KL-4M Board includes an 8-bit Digital to Analog Converter, a low pass filter to eliminate high frequency computer generated hiss, and an on-board audio amplifier. An RCA-type jack is also included for quick attachment of your speaker. Amplification of the 6522 CB2 generated single note sound is incorporated as well, so that no additional hardware (other than a speaker) is required Connection is made via the parallel and cassette ports. Both ports are extended with duplicate connectors (with keyways) so I/O capabilities are not reduced in any way Board orientation is parallel to the back of the PET so additional table space is not required The KL-4M is compatible with any of the 4 part music monitors, for which a number of precoded songs are available.

The visible Music Monitor is intended to support 4-part harmony systems such as the KL-4M Visible Music Monitor is written entirely in 6502 machine language VMM provides an easy way to enter 4-part music. The user can see the notes on the screen as they are entered, and can make changes both with the insert and delete keys, and by using cursor up and down to "move" notes on the screen. Other features include "record changer" mode to load successive songs without intervention, user definable keyboard, and entry of whole notes through 64ths including dotted and triplet notes. Additionally, you can specify or change tempo, set key signature, and transpose at any time. Wave form modification makes it possible to create new instrument sounds Voices can switch from one instrument to another or gang up on one instrument during the course of the song Music can be played either with note display (useful for debugging songs), or with no display

KL-4M Music Board & Visible Music Monitor Program **$59.90**

PAPER-MATE 60 COMMAND WORD PROCESSOR



by Michael Riley

Paper-Mate is a full-featured word processor for $29.00 by Michael Riley Paper-Mate incorporates 60 commands to give you full screen editing with graphics for all 16k or 32K PETs, all printers, and disk or tape drives It also includes most features of the CBM WordPro III, plus many additional features

For writing text, Paper-Mate has a definable keyboard so you can use either Business or Graphics machines Shift lock on letters only, or use keyboard shift lock All keys repeat

Paper-Mate text editing includes floating cursor, scroll up or down, page forward or back, and repeating insert and delete keys. Text Block handling includes transfer, delete, append, save, load, and insert.

All formatting commands are imbedded in text for complete control Commands include margin control and release, column adjust, 9 tab settings, variable line spacing, justify text, center text, and auto print form letter (variable block). Files can be linked so that one command prints an entire manuscript Auto page, page headers, page numbers, pause at end of page, and hyphenation pauses are included

Unlike most word processors, PET graphics as well as text can be used. Paper-Mate can send any ASC11 code over any secondary address to any printer

Paper-Mate works on 16K or 32K PETs with any ROM, cassette or disk, and CBM or non-CBM printers An 8K version is in the planning.

To order Paper-Mate, specify machine and ROM type.

On Tape (with manual): **$29.95**
On Disk (with manual): **$32.95**
Manual Separate: .... **$ 1.00**

BET YOU CAN'T BEAT IT! MICROTHELLO

by Mike Riley

There are five levels of play in this PET machine language program Level four is for experts only and is designed for tournament level play. So far no one has been able to beat level four Level five takes several minutes to move and is used for exhaustive analysis of specific moves.

There are several features to help in the analysis of a game. Any position on the board can be recalled and replayed. Both the level of difficulty and the position of the pieces can be changed at any time. You can play against the machine, against another person, or watch the machine play itself. You and the machine can switch sides during the game. Moves are selected with the cursor rather than by coordinates. For all PETS: **$9.95**

TUNNEL VISION & KAT AND MOUSE NOW IN MACHINE LANGUAGE

By Riley and Levinson

This program was so popular that several other versions have appeared on the market. In order to keep ahead of the competition, the program has been re-written in machine language for fast graphics.

The program includes two excellent maze games. In Tunnel Vision, you view the maze from inside in perspective. If you get lost, the program provides a map showing your trail. In Kat and Mouse, you must find your way through the maze before the hungry Kat finds you. Each maze has only one solution, and each is unique. For all PETS : **$7.95**

FREE! Up to $170 in merchandise with purchase of one of following PET-CBM items ! ! !

Axiom EX-801 PET Printer (with graphics)	$ 475 00
Axiom EX-820 PET Plotter	$ 749 00
Anderson Jacobson 841 Selectric	$1015.00
Leedex Video 100 12" Monitor	$ 119.00
Heath WH19 Terminal (factory asm.)	$ 770.00
Heath WH14 Printer (factory asm.)	$ 735.00
IEEE-RS 232 Printer Adaptor for PET	$ 88 00

BETSI PET to S-100 Interface	$ 119 00
PET Connectors-Parallel or IEEE	$ 1 95
Cassette Port	$ 1 45
Hands on Basic with a PET	$ 10 15
Programming the 6502 (Zaks)	$ 10 45
6502 Applications Book (Zaks)	$ 10 45
6500 Manuals (MOS Technology)	$ 6 50
Programming a Microcomputer: 6502	$ 7 75
6502 Assembly Language (Osborne) NEW!	$ 9 90

PET SPECIALS

		*FREE
PET 16N 16K full size graphics keyboard	$ 995	$130
PET 16B 16K full size business keyboard	$ 995	$130
PET 32N 32K full size graphics keyboard	$1295	$170
PET 32B 32K full size business keyboard	$1295	$170
PET 8N 8K full size graphics keyboard	$ 795	$100
PET 2040 DUAL DISK DRIVE-343,000 bytes	$1295	$170
PET 2022 Tractor Feed Printer	$ 795	$100
PET 2023 Pressure Feed Printer	$ 695	$ 70
PET C2N External Cassette Deck	$ 95	$ 12
Used 8K PETs (limited quantities)	$ 495	

★ **EDUCATIONAL DISCOUNTS** ★
Buy 2 PET Computers, get 1 FREE

*Free Merchandise with Purchase of PET-CBM Item

KIM-1 $159 (add $30 for power supply) SYM-1	$209
BAS-1 Microsoft ROM Basic for SYM	$ 85
Memory Plus (KIM, SYM, AIM)	$195
SYM Assembler in ROM	$ 85
KIM 16K Static RAM	$284
Seawell Motherboard - 4K RAM Space	$139
KTM-2/80 Synertek Video Board	$349
S-100 16K Static RAM Kit SALE	$198
TIS PET Workbooks - set of 6	$ 21 50
Dust Cover for PET	$ 7 45

All Book and Software Prices are Discounted

PET Personal Computer Guide (Osborne)	$ 12 75
PET and the IEEE-488 Bus (Osborne)	$ 12 75
6502 Software Cookbook (Scelbi)	$ 9.45

Cassettes (all tapes guaranteed) AGFA PE611 Premium quality, high out put lownoise in 5 screw housing with labels

C-10	10/5.65	50/25 00	100/48 00
C-30	10/6 90	50/30 00	100/57 00

2716 EPROM (5 volt)	$29 00	5 @ 27 00	10 @ 25 00
4116 RAM 16K x 1		$7 50	8 @ 7 10
2114 L 450	$5 35	20 @ 4 95	80 @ 4 45
2114 L 300	$6 10	20 @ 5 65	80 @ 5 05
6502	$8 35	10 @ 7 85	40 @ 7 35
6522	$7 15	10 @ 6 95	40 @ 6 45
6520	$5 15	10 @ 4 90	40 @ 4 45
6532	$7 90	10 @ 7 40	40 @ 6 85
6502A			$9 50
6504			$7 95
21L02			$ 90
6550 RAM (PET 8K)			$12 70
S-100 Wire Wrap		$2 85	10 @ 2 65
S-100 Solder Tail		$2 35	10 @ 2 15

ATARI — INTRODUCTORY SPECIAL

ATARI 400, Atari 800 and all Atari Modules 20% OFF.

Programmers Toolkit - PET ROM Utilities	$ 44 90
Sargon II (TRS-80 or Apple) NEW!	$ 24.90
Microchess for PET (Peter Jennings)	$ 17.90
Visible Music Monitor (4 Voice) for PET	$ 29 90
KL-4M 4 Voice Music Board (PET)	$ 34.90
CmC Word Processor (8K $25) (16K $34)	
Adventures by Scott Adams	15% Off

SALE

(write for quantity prices)

Scotch 8" Disks	10/$31 00
Scotch 5¼" Disks	10/$31 50
Verbatim 5¼" Disks	10/$26 50
Diskette Storage Pages	10/$ 3 95
Disk Library Cases 8" - $2.95	5" - $2 15
BASF 5¼" Disks	10/$28.00
BASF 8" Disks	10/$29.00

MINIMAX by COMPU/THINK

MINIMAX I (.8 Megabyte Disks) **$3395**
MINIMAX II (2.4 Megabyte Disks) **$4495**

SPECIAL — MINIMAX prices include Compu/Think PAGEMATE Database and Report Writer at no charge

115 E. Stump Road
Montgomeryville, PA 18936
(215) 699-8386
699-5826

A B Computers

WRITE FOR COMPLETE CATALOG
Add $1.00 per order for shipping.
We pay balance of UPS surface charges on all prepaid orders.

COMPUTERS Boring, Boring, Boring

Ron Kushnier
25 Wendy Way
Richboro, PA 18954

Sure, everyone knows how exciting it is to play Space Invaders or some similar game on their home computer!

But perhaps the real utility in our machines is derived from uses that are less dramatic. After all, one of the primary uses for any machine is to free us from the drudgery and mundane routine that burden us in our daily lives; so that we can go off and think lofty thoughts and delve into the esoteric.

Such an application resulted from a problem with the heating system of our new home last winter. I knew that the heating contractor and the builder were less than cooperative and that if I were to get anywhere with them I would have to produce *cold, hard facts.* I also knew that according to Murphy, on the day they would come to inspect the system, it would be sunny and warm and the heater would exhibit absolutely no sign of improper design or malfunction. I needed data- lots of data to substantiate my claim.

I turned to my Pet. Fortunately at work, I had all the peripherals I needed to turn my computer into a digital recording thermometer. I had previously ordered a DAM system (now called micro Mac) from Conn. MicroComputer Corp. This set of devices is designed to provide an analog-to-digital conversion system which interfaces to the Pet. It came with a Tempsens which has two temperature probes good to within a degree Fahrenheit. As an output device, I used an HP model 5150A Thermal Printer.

The program shown gave me a readout of the time every 15 minutes, the temperature at the floor, and the temperature at the five foot level.

I left the Pet on 24 hours a day over the next several days, annotating the data tapes by hand as required with such things as the date, average outdoor temperature, etc. The time period provided enough diverse weather conditions to manifest a fairly comprehensive variety of heating system problems.

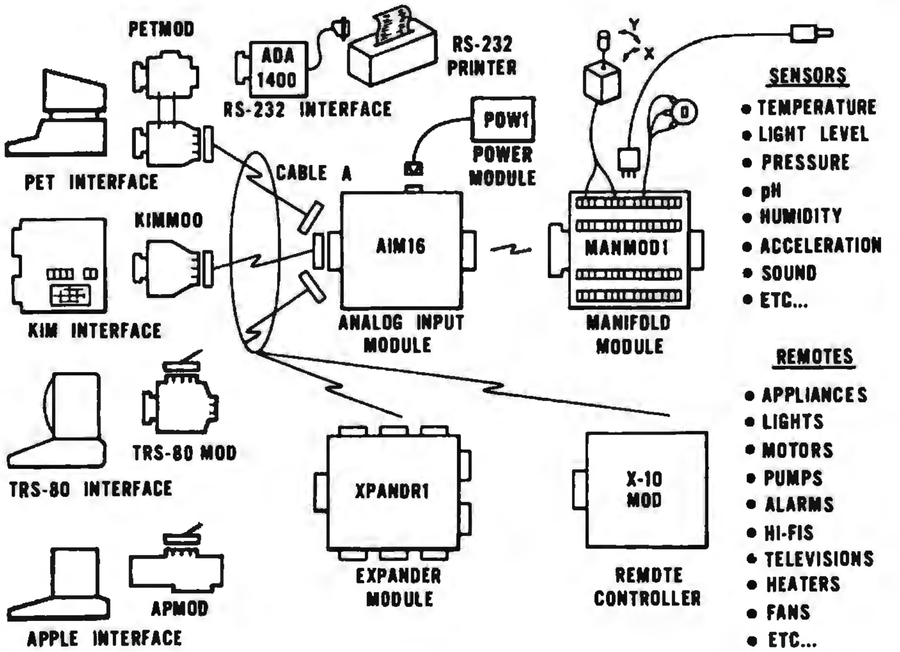
My long rolls of data tapes convinced the builder and heating contractor that I was serious. And my impressive collection of equipment could not be challenged.

The computer's function was boring-sit there hour after hour and print out numbers. This certainly could not compare with shooting down a Klingon. Yet, it provided my family and I with a needed and useful utility for which there was no substitute.

Program Listing

```
5 PRINT"ĥ
10 OPEN4,4
15 INPUT"HR,MN,SC";HR$,MN$,SC$
17 PRINT"ĥ
20 TI$=HR$+MN$+SC$
30 M$=MID$(TI$,3,2)
31 PRINT"ḫ
32 PRINT"TIME";"      5 FEET";"     ¬
     ¬FLOOR"
33 GOSUB500
35 PRINT"ḫ↓↓";TI$,HIGH,LOW
36 IFF$=M$THENGOTO30
40 IFM$="00"ORM$="15"ORM$="30"ORM$="45" ¬
     ¬THENF$=M$:GOTO90
50 GOTO30
90 GOSUB500
130 PRINT#4,TI$;HIGH;LOW
135 GOTO30
500 POKE59426,0: POKE59426,255: ¬
     ¬HIGH=PEEK(59471)-20
510 POKE59426,1: POKE59426,255: ¬
     ¬LOW=PEEK(59471)-20
520 RETURN
59468 ,12
```

MICROCOMPUTER MEASUREMENT and



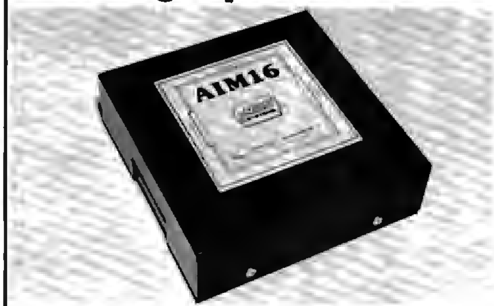
The world we live in is full of variables we want to measure. These include weight, temperature, pressure, humidity, speed and fluid level. These variables are continuous and their values may be represented by a voltage. This voltage is the analog of the physical variable. A device which converts a physical,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

Analog Input Module



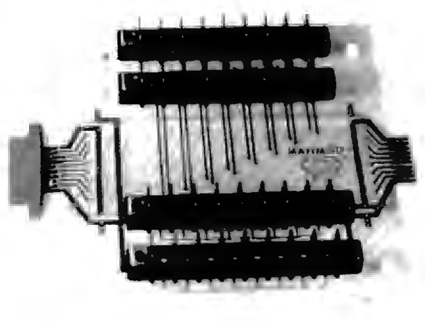
The AIM 16 is a 16 channel analog to digital converter designed to work with most microcomputers. The AIM16 is connected to the host computer through the computer's 8 bit input port and 8 bit output port, or through one of the uMAC SYSTEMS special interfaces.

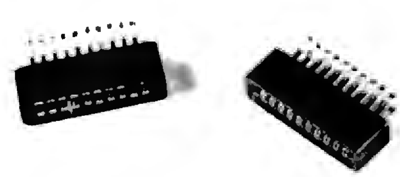
The input voltage range is 0 to 5.12 volts. The input voltage is converted to a count between 0 and 255 (00 and FF hex). Resolution is 20 millivolts per count. Accuracy is 0.5% ± 1 bit. Conversion time is less than 100 microseconds per channel. All 16 channels can be scanned in less than 1.5 milliseconds.

Power requirements are 12 volts DC at 60 ma.

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.

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.

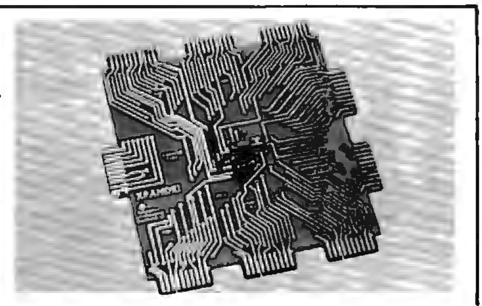
TEMPSENS

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.

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. Up to eight AIM16 modules are then connected to each of the eight ports provided using a CABLE A24 for each module. Power for the XPANDR1 is derived from the AIM16 connected to the first port.



new from...
HARDSIDE

6 SOUTH ST., MILFORD, NH 03055

ATARI®

A Warner Communications Company

HARDSIDE

TO ORDER TOLL-FREE

1-800-258-1790

(In N.H. call 673-5144)

ATARI 800 (list 1080.00) 875.00
ATARI 400 (list 630.00) 515.00
810 DISK DRIVE (699.95) 565.00
820 40-COLUMN PRINTER
(list 599.95) 470.00
410 PROGRAM RECORDER
(list 89.95) 65.00
JOYSTICK CONTROLLERS . . 18.95
PADDLE CONTROLLERS . . . 18.95
ATARI 850 INTERFACE . to be announced
ATARI 830 MODEM . . . . to be announced
ATARI 825 PRINTER . . . . to be announced

INSIDE ATARI BASIC

Larry Isaacs, COMPUTE. staff

This article will present information on how the ATARI BASIC stores programs in memory. If you are new to the field of microcomputer programming, this information should help increase your awareness of what your ATARI is doing, making its behavior a little more understandable.

The following information is based solely on what I have been able to observe while working with an ATARI 800. I believe the information to be accurate. However, it is hard to know how complete the information is.

Also for those new to microcomputer programming, the next section gives some preliminary information which should help make the rest of the article more understandable.

PRELIMINARY INFORMATION

One very important term in the field of microcomputing is the term **"byte"**. For purposes of this article, it can be considered a number which can have a value ranging from 0 to 255. The memory in your ATARI consists of groups of bytes, each byte of which can be referenced by a unique address. The part of memory which is changeable, called RAM, starts with a byte at address 0 and continues with bytes at increasing sequential addresses until the top of RAM is reached. The top of RAM is determined by the type and number of memory modules you have in your ATARI.

Bytes, or combinations of bytes, can be used to represent anything you want. Some common uses for bytes include representing memory addresses, characters, numbers, and instructions for the CPU in your ATARI. You will be exposed to several different uses for bytes in this article. Some of these uses will make reference to **"two byte binary numbers."** This is where two bytes are used to represent a number whose value ranges from 0 to 65535. The decimal value of a two byte binary number can be computed using the formula: **FIRST BYTE + (SECOND BYTE*256).**

Also in this article, reference will be made to **"page zero"**. Page zero simply refers to the first 256 bytes of memory, i.e. addresses 0 through 255. This part of memory differs from the rest of memory by the fact that these bytes can be referenced using a single byte address. The rest of memory requires two byte addresses.

THE CONVERSION

After typing in a BASIC line, hitting RETURN causes the line to be passed to the programs found in the ATARI BASIC cartridge. Here the line will undergo a certain amount of conversion before it is stored in memory. One part of this conversion involves converting all of the BASIC reserved words and symbols to a one byte number called a **token.**

Another part of the conversion involves replacing each variable name in the line with an assigned number which will range from 128 to 255. If a variable name has been previously used, it will be replaced by the number previously assigned. If it hasn't been used before, it will be assigned the lowest unused number, starting with 128 for the first variable name. Also, numbers in the BASIC line must be converted into the form which the ATARI BASIC uses before they can be stored in memory.

After the conversion is finished, the line is stored in memory. If the BASIC line does not have a line number, it will be stored after the last statement of your BASIC program, and executed immediately. If it does contain a line number, the converted line will be inserted in the proper place in your program. After the line has been executed or stored, your ATARI will wait for you to type in another line. Even though the line undergoes this conversion, the order in which the reserved words, variables, and symbols occur in the line isn't changed when it is stored in memory.

THE MEMORY FORMAT FOR A BASIC LINE

Let's begin with the general format of how a BASIC line is stored. Once a BASIC line has been converted and stored, the line number is found in the first two bytes of the memory containing the BASIC line. These bytes form a two byte binary number which has the value of the line number. The value of this number can range from 0 to 32767.

The third byte contains the total number of bytes in this BASIC line. This means you can find the first byte of the next line using the following formula: **ADDRESS OF FIRST BYTE OF NEXT LINE = ADDRESS OF FIRST BYTE OF CURRENT LINE + NUMBER IN THIRD BYTE OF CURRENT LINE.**

The fourth byte contains the number of bytes in the first statement in the line, including the first four bytes. If the BASIC line contained only one statement, the third and fourth bytes will contain the same value. If the line had more than one statement, these bytes will be different.

Next come the bytes which represent the first statement in the line. If there is more than one statement, the next byte following the first statement contains the number of bytes in the first two statements. Naturally, if there is another statement after the second one, the first byte after the end of the second statement contains the number of bytes in the first three statements, etc.

This completes the format of a BASIC line as it is found in memory. Before going on, let's put this information to use in a short program which lists out its own line numbers along with the beginning address of each line. To do this we must first find out where the first byte of the first line is found. It turns out there is a two byte binary number found in page zero which contains the beginning address of the first line. This number is contained in bytes 136 and 137. Also, we will know when we've reached the end of the program when we find a line number of 32768, which is one more than the maximum allowed by ATARI BASIC. The program to print the line numbers and their beginning addresses is shown in Listing 1.

TOKENS

In order to conserve memory, all of the BASIC reserved words, operators, and various punctuation symbols are converted into a one byte number called a token. This conversion also makes execution simpler and faster. The tokens can be divided into two groups. One group contains the tokens which occur only at the beginning of a BASIC statement, and the other group contains the tokens which occur elsewhere in a BASIC statement.

Let's first take a look at the tokens which occur at the beginning of a BASIC statement. It turns out that all statements will begin with one of these tokens. After some investigation, I found that these tokens will range in value from 0 to 54.

The procedure for listing the tokens is fairly simple, though the actual implementation is a bit more involved than the brief explanation which follows. The idea is to put "1 REM" as the first statement of the program. Then use POKEs to change the line number and token of this REM statement. By setting the line number and token to the same number, listing the line will print the token and corresponding BASIC reserved word. Fortunately the programs in the BASIC cartridge which do the listing tolerate the incomplete BASIC statements. The program for displaying these tokens is shown in Listing 2. Notice when you run this program, no reserved word is printed for token 54. This is the invisible LET token which is used for assignment statements which don't begin with LET.

A similar procedure can be used to list the other tokens as well. The main differences are to make the first statement "1 REM A", POKE 54 (the invisible LET token) into the first byte of the statement, and make the changes for the token to the second byte of the statement. The values for the tokens which occur after the beginning of a statement range from 20 to 84. The program for printing these tokens is given in Listing 3.

After running this program, you will notice there is no reserved word or symbol printed for token 22. Token 22 is the terminator token found at the end of each BASIC line, except those whose last statement is a REM or DATA statement. Also, tokens 56 and 57 didn't print a reserved word or symbol. Both of these tokens represent the "(" symbol. The "(" doesn't print because these two tokens are associated with array names, and the "(" symbol is kept with the associated variable name, as will be seen in the next section.

Of course you noticed that most of the symbols occur more than once. There is a different token for each of the different uses of the symbol. For example, the word " = " has four different tokens. Token 45 calls for an arithmetic assignment operation as in A = A + 1. Token 46 calls for a string assignment as in A$ = "ABC". Token 34 is used in arithmetic testing as in IF A = 1 THEN STOP. And finally, token 52 is the same as token 34 except that it's for testing strings.

One more token, found after the ones listed in the previous program: token 14, which indicates a constant is stored in the following 6 bytes.

```
10 REM PROGRAM TO PRINT LINE NUMBERS
20 REM AND THEIR ADDRESSES
30 REM
40 REM GET ADDRESS OF FIRST LINE
50 ADDRESS=PEEK(136)+PEEK(137)*256
60 REM GET THE LINE NUMBER
70 LNUM=PEEK(ADDRESS)+PEEK(ADDRESS+1)*256
80 REM TEST FOR END OF PROGRAM
90 IF LNUM=32768 THEN END
100 REM PRINT LINE NUMBER AND ADDRESS
110 ? "LINE # ";LNUM;
120 ? " STARTS AT ADDRESS ";ADDRESS
130 REM GET ADDRESS OF NEXT LINE
140 ADDRESS=ADDRESS+PEEK(ADDRESS+2)
150 GOTO 70
```

```
1 REM
100 REM PROGRAM TO PRINT THE TOKENS
110 REM WHICH BEGIN BASIC STATEMENTS
120 REM GET BEGINNING OF PROGRAM
130 BASE=PEEK(136)+PEEK(137)*256
140 REM CHANGE STATEMENT TERMINATOR
150 POKE BASE+5,22
160 ? CHR$(125): REM CLEAR SCREEN
170 REM PRINT TOKENS
180 FOR I=0 TO 54
190 REM CHANGE LINE NUMBER AND TOKEN
200 POKE BASE,I:POKE BASE+4,I
210 LIST I: REM PRINT TOKEN
220 REM UNDO LINE FEED IF NEEDED
230 IF I>1 THEN ? CHR$(28);
240 REM CHANGE LEFT MARGIN FOR COLUMNS
250 IF I=19 THEN POKE 82,12:POSITION 12,1
260 IF I=39 THEN POKE 82,24:POSITION 24,1
270 NEXT I
280 REM PUT PROGRAM BACK TO NORMAL
290 POKE BASE,1:POKE BASE+4,0
300 POKE BASE+5,155
310 POKE 82,2:POSITION 2,22
```

```
1 REM A
100 BASE=PEEK(136)+PEEK(137)*256
110 REM CHANGE BEGINNING TOKEN
120 POKE BASE+4,54:POKE BASE+6,22
130 REM PRINT OPERATOR AND FUNCTION TOKENS
140 ? CHR$(125)
150 FOR I=20 TO 84
160 POKE BASE,I:POKE BASE+5,I
170 LIST I
180 REM UNDO LINE FEEDS
190 ? CHR$(28);:IF I=22 THEN ? CHR$(28);
200 IF I=39 THEN POKE82,11:POSITION11,1
210 IF I=59 THEN POKE82,19:POSITION19,1
220 IF I=79 THEN POKE82,28:POSITION28,1
230 NEXT I
240 POKE BASE,1:POKE BASE+4,0
250 POKE BASE+5,65:POKE BASE+6,155
260 POKE 82,2:POSITION 2,22
```

VARIABLE NAMES AND CONSTANTS

As each new variable is encountered, it is assigned a number. These numbers begin with 128 and are assigned sequentially up to 255. Notice these numbers will fit into one byte. Also, as each new variable is encountered, the variable name is added to a variable name list, and 8 bytes of memory are reserved for that variable. In the case of undimensioned variables, these 8 bytes will contain the value of the variable. For strings and arrays, these 8 bytes will contain parameters, with the actual values and characters stored elsewhere.

This method of handling variables has some advantages. One advantage is that it keeps usage to a minimum. The variable name is only stored once, and each time that name is referenced in a BASIC statement, it occupies only one byte in the stored program. Another advantage is that the address where the value for a variable is stored can be computed from the assigned number. This isn't true of the BASIC found in some other microcomputers where values must be searched for.

There are also some disadvantages as well. First, it limits you to 128 different variable names. However, the great majority of programs won't need more than 128 variable names. One other disadvantage is that should a variable name be no longer needed, or accidentaly entered due to a typo, there is no quick way to remove that variable from the variable name list and reuse the 8 bytes reserved for it.

Apparently, the only way to get rid of unwanted variables is to LIST the program to cassette or disk. For example, LIST "C will list the program to cassette. Once the program is saved, use the NEW command to clear the old program. Then use the ENTER command to reload the program. For cassette this would be ENTER "C. Using the LIST command saves the program in character form. ENTERing the program then causes each line to be converted again as was done when you first typed it in. Now only the variables found in the program will be placed in the variable name list, and space reserved for their value. Using CSAVE and CLOAD won't do this because these save and load a copy of the memory where the program is stored. Unwanted variables are saved and loaded with the rest of the program.

Constants are stored in the BASIC statements along with the rest of the line. The constant will be preceeded by a "14" token as mentioned previously. Explaining how ATARI BASIC represents the numbers used as constants and as variable values will require some explanation about BCD (Binary Coded Decimal) numbers. I will save this information for a later article.

To give an example of using the information in this section, let's take a look at the variable name list. Fortunately bytes 130 and 131 contain the address of the beginning of the variable name list. The list will consist of a string of characters, each character occupying one byte of memory. To indicate the last character of a name, ATARI BASIC adds 128 to the value representing that character. Since the values representing the characters won't exceed 127, the new value will still fit into one byte. To indicate the end of the list, a 0 is placed in the byte following the last character of the last name. The program which prints the variable name list is given in Listing 4. Notice when you run this program, that the "(" is saved as part of a array name, and the "$" as part of a string name.

MEMORY ORGANIZATION

Finally, let's look at how the memory storage is organized for a BASIC program. The order in which the various parts of a program are found in memory is shown in Figure 1. The only part whose beginning is fixed is the variable name list which begins at address 2048. The beginning of the other parts will move appropriately, as the program grows. There are addresses in page zero which can be used to find each of the parts shown in Figure 1. These addresses, usually called pointers, are shown in Table I. This table includes the two pointers which were used in the previous programs.

Figure 1. MEMORY ORGANIZATION

Increasing
Addresses
???? End of Array Storage Area
.
???? Beginning of Array Storage Area
???? End of Program
.
???? Beginning of Program
???? End of Variable Storage Area
.
???? Beginning of Variable Storage Area
???? End of Variable Name List
.
2048 Beginning of Variable Name List

TABLE 1

ADDRESSES	NAME	CONTENTS POINT TO
130 & 131	BON	Beginning Of variable Names list
132 & 133	EON	End Of variable Name list
134 & 135	BOV	Beginning Of Variable storage area
136 & 137	BOP	Beginning Of Program
138 & 139	CEL	Beginning Of Currently Executing Line
140 & 141	BOA	Beginning Of Array storage area
142 & 143	EOA	End of Array storage area

```
100 REM PROGRAM TO PRINT THE VARIABLE NAME LIST
110 DIM ARRAYNAME(1), STRINGNAMES$(1)
120 REM GET THE BEGINNING OF THE LIST
130 ADDRESS=PEEK(130)+PEEK(131)*256
140 ? CHR$(125);"VARIABLE NAME LIST"
150 REM GET CHARACTER AND TEST FOR END
160 A=PEEK(ADDRESS):IF A=0 THEN END
170 REM PRINT CHARACTER
180 IF A<128 THEN ? CHR$(A);:GOTO 210
190 ? CHR$(A-128)
200 REM GET NEXT ADDRESS AND REPEAT
210 ADDRESS=ADDRESS+1:GOTO 160
```

APPLICATION

For those who are interested in putting this information to use, I will present one example here. I will try to give more examples in future issues of COMPUTE.

At some time you may find it useful to be able to "undimension" some arrays of strings, and reuse the memory for some other arrays and strings. It turns out that the CLR function only clears the variables found between the **BOA** (**B**eginning **O**f Variables) pointer and the **BOP** (**B**eginning **O**f **P**rogram) pointer. By temporarily changing the BOP pointer, we can keep some of the variables from being cleared. The array storage area is cleared by setting the **EOA** (**E**nd **O**f **A**rrays) pointer equal to the **BOA** (**B**eginning **O**f **A**rrays) pointer. We can save some of the array storage area by temporarily changing the BOA pointer.

The listing for this UNDIMENSION routine is shown in Listing 5. The listing also includes a small demo program to illustrate its use. Note that all of the names of variables which are to be cleared should occur in the program prior to any of the names of variables which are to be saved. This puts the storage for the variables to be cleared at the beginning of the variable storage area. Also note that a dummy string which can be cleared is needed by the UNDIMENSION routine. In your main program, this dummy string should be dimensioned just before dimensioning the strings and arrays that you will later clear, as was done in statements 120 and 150. This allows the use of the ADR function to find the end of the array area to be saved.

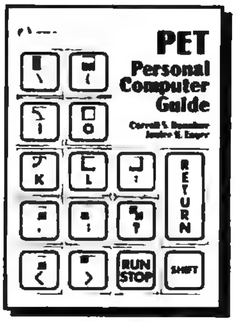
The reason the UNDIMENSION routine is not executed by a GOSUB is that the return line number is lost in the clearing process. Loop parameters will also be lost, so the routine shouldn't be executed while in a FOR..NEXT loop.

CONCLUSION

Hopefully you found the information in this article understandable, and will find it useful at some point in the future. The information does show that ATARI BASIC is fairly efficient in using memory to store programs. Also, there is very little penalty in memory usage when using long variable names. If you have any questions please send them to COMPUTE. ©

```
1 REM DIMENSION THE DUMMY STRING
2 DIM DUMMY$(1)
3 REM DIMENSION THE ARRAYS AND STRINGS
4 REM WHICH WILL NEED CLEARING
5 DIM A1(1),A2(1)
6 CLR: REM CLEAR THESE VARIABLES
7 N=3: REM N= # OF VARIABLES JUST DIMENSIONED
8 REM       INCLUDING DUMMY$
9 REM YOUR PROGRAM MAY BEGIN HERE
100 REM HERE IS AN EXAMPLE OF HOW TO
110 REM USE THE UNDIMENSION ROUTINE
120 DIM TEST$(20):TEST$="I'M STILL HERE"
130 DIM DUMMY$(1),A1(50,10)
140 A1(50,10)=1:? A1(50,10),TEST$
150 REM EXECUTE UNDIMENSION ROUTINE
160 LINE=170:GOTO 1020
170 DIM DUMMY$(1),A2(500)
180 A2(500)=2:? A2(500),TEST$
190 END
200 REM
1000 REM UNDIMENSION ROUTINE
1010 REM SAVE CURRENT POINTER VALUES
1020 S136=PEEK(136):S137=PEEK(137)
1030 S140=PEEK(140):S141=PEEK(141)
1040 REM MOVE END OF VARIABLES
1050 T1=PEEK(134)+8*N:T2=PEEK(135)
1060 IF T1>255 THEN T2=T2+1:T1=T1-256:GOTO 1060
1070 POKE136,T1:POKE137,T2
1080 REM MOVE BEGINNING OF ARRAYS
1090 T2=INT(ADR(DUMMY$)/256)
1100 T1=ADR(DUMMY$)-T2*256
1110 POKE140,T1:POKE141,T2
1120 CLR: REM CLEAR THE ARRAYS
1130 REM RESTORE POINTERS AND RETURN
1140 POKE136,S136:POKE137,S137
1150 POKE140,S140:POKE141,S141
1160 GOTO LINE
```

Something New for your PET



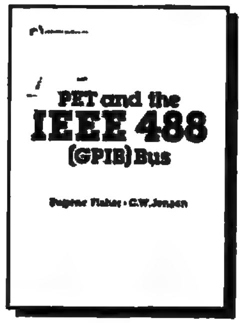
PET Personal Computer Guide

by C. Donahue and J. Enger

Everything you always wanted to know about PET but Commodore didn't tell you. A practical guide to PET programming techniques, graphics, operation, and how to cope with those*&$! PET peculiarities.

#30-6. $15 00

NEW this Winter



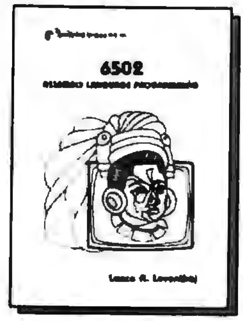
PET and the IEEE 488 (GPIB) Bus

by C. Fisher and C. W. Jensen

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

#31-4 $15 00

NEW this Winter



6502 Assembly Language Programming

by L. Leventhal

For the advanced programmer; increase the capabilities and performance of PET (and other 6502-based computers) by learning to program in assembly language.

#27-6 $12.50

Some Common BASIC Programs

By L. Poole and M. Borchers

This book was designed for people who can use a variety of practical BASIC programs — 76 programs in all that cover a wide variety of personal finance, math, statistics, and general interest topics. The documentation in the book is complete so that you can run the programs even if you aren't an experienced programmer.

#06-3. $12.50

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

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

Book/Cassette/Disk	Price	Quantity	Amount
27-6 6502 Assembly Language Programming	$12 50		
30-6 PET Personal Computer Guide	$15 00		
31-4 PET and the IEEE 488 (GPIB) Bus	$15 00		
06-3 Some Common BASIC Programs (book)	$12 50		
25-X Some Common BASIC Programs PET Cassette	$15 00		
33-0 Some Common BASIC Programs PET Disk	$22 50		

California residents add 6% sales tax. — California resident tax ____

S F BART residents add 6½% sales tax — Shipping ____

Total Amount Enclosed ____

Shipping (Shipping for large orders to be arranged)

- ☐ All foreign orders $4 00 per book for airmail
- ☐ $0 45 per book 4th class in the U S (allow 3-4 weeks)
- ☐ $0 75 per book UPS in the U S. (allow 10 days)
- ☐ $1 50 per book special rush shipment by air in the U S.

Cassettes and Disk

- ☐ No additional charge in the U S
- ☐ $1 50 each foreign airmail

OSBORNE/McGraw-Hill

Name ____

Address ____

City ____

State ____ ZIP ____

Phone ____

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

51074

Review:

PET AND THE IEEE 488 Bus (GPIB),

by Eugene Fisher and C. W. Jensen. Osborne/McGraw Hill, 233 pp. $15.00

Review by Jim Butterfield

The IEEE-488 bus may be a small part of the overall PET/CBM system, but there's enough in it to fill a book - and then some. One gets the impression that Fisher and Jensen could have gone on for another 200 pages or so without exhausting the subject.

It's a very thorough book, and it documents the IEEE-488 bus and PET's role in considerable detail: line by line, command by command. The book is primarily aimed at hardware enthusiasts and engineering types; but even the casual user who just plugs in and hopes it works will find useful information here. There are general descriptions of what goes on, with liberal sketches depicting the interaction in a somewhat whimsical form. There are extensive lists of compatible equipment and descriptions of applications. There's even a short diagnostic test which may help pinpoint trouble on the bus.

For those who need to plunge into technical detail, it's all there: from connector pin designations to signal sequences, everything is spelled out in detail. Chapter 5, "Execution and Timing Sequences", is the longest chapter in the book. It contains step-by-step outlines of everything that happens on the bus for every relevant PET Basic command.

There are a few things the authors didn't tackle. Chapter 6 deals with interfacing non-standard devices. The interface described is for a receive-only device which responds to everything the PET sends. A schematic is given, and a subsequent chapter details a similar interface to the Centronics P1 printer. But the design of a selecting interface, which can be called in as needed by the PET, is not touched upon. Even a brief outline here would have been useful.

Similarly, the authors stay away from the question of multiple PETs communicating with each other or with a common device over the GPIB bus. It's a tricky subject and laced with pitfalls; users would have been grateful for any hints they might have been given.

Even so, the book has a wealth of detail on all levels. There's hardly a page without a diagram, chart, illustration or photograph. Timing questions and logic sequences are described meticulously.

The book is remarkably free from errors. A few creep in: table 5-12 seems to have omitted the "unlisten" signal; page 72 should show an EOI on the last E of GENE; page 109 seems to show the CR and EOI coming from the PET rather than the external device. The sequence given on page 83: OPEN 5,5 "TEST" doesn't seem to work on my PET: it gives a syntax error. The authors might have mentioned the one-character delay that PET introduces in the output (so that it can append the EOI signal at the right time).

I would have liked to see more attention given to multiple devices. At first glance, it appears that since PET only works one device at a time, the single-talker, single-listener description is adequate. But all devices are on-line during the selection sequences; and it seems to be important to emphasize that any device can pull a line to true and all devices must release the line to false. That's the whole point behind the three-wire handshake; yet I couldn't find the information in the introductory section.

But what the book is missing is minor compared to what it's got. It's a gold mine of both conceptual information and hard facts. It covers hardware, software, mechanical assemblies, and standards. It lists numerous devices and gives copious references.

In short, it's virtually everything you wanted to know about the IEEE-488 bus .. or are likely to want to know in the future. ©

Review:

6502 Software Design,

Review by Jim Butterfield

by Leo J. Scanlon
Howard W. Sams & Co., Inc., 270 pp. $10.50

This is quite a pleasant book; it makes good reading. The title is misleading: the book deals with 6502 programming rather than software design. In fact, I rather missed the software design elements: planning memory, designing data structures, and testing/debugging methodology are important concepts for beginning programmers. The book expends its main efforts on the subjects of writing code and handling interrupts and input/output devices.

The book is oriented towards the Rockwell AIM 65 microcomputer, but not too heavily. Users of other systems will find it a worthwhile reference of text. The book appears ideal for a programming course textbook; it is well organized and gets into serious programming quite quickly.

Chapter 1 is partly historical, partly an overview of popular microprocessors and partly a brief plug for the AIM 65. There's a minor problem where the author describes the 6800/6502 family as characterized by novel memory-mapped architecture: in fact, all microprocessors including the 8080 and Z-80 can and do use memory mapped I/0. And I rather wish that the introduction to the 6502 had mentioned its remarkable speed due to pipelining techniques - a factor that sets it clearly ahead of the earlier 6800.

Chapter 2 gets into the 6502 instruction set. Similar op codes are grouped and discussed together, which helps to develop intuitive ideas of the machine's capabilities. Numerous examples of coding are included. The style is generally easy and straightforward, but beginners will still find it slow going: there's a lot of material to get through.

Subroutines are covered in chapter 4, mostly in terms of their mechanistic characteristics. A couple of time delay subroutines are given, and it's nice to see timing questions being worked through meticulously.

Chapters 4, 5 and 6 get heavily into coding questions: list and lookups, mathematical routines, and number-base conversion. Coding is given throughout, but the emphasis is on algorithms: the author is concerned with the methods behind a particular kind of computation. A rather unusual algorithm is given for calculating the square root of an integer; it's rather slow compared to standard methods, but readers may find it an interesting curiosity.

Chapter 7 discusses interrupts. In sixteen pages, it's hard to come to grips with the whole question, and coding examples are cursory. Chapter 8 goes into considerable detail on two input/output chips: the 6520 and 6522.

There are two appendices and an index. I found the index rather sparse: for example, "relative addressing" points you to page 40, but a more useful discussion can be found on page 55.

Coding examples are written almost entirely in assembly language. This is quite readable, but I would have preferred to see early coding examples complete with their machine language equivalents.

The overall organization of the book is good, although there are inconsistencies within chapters: some have summaries, some have references, and some have neither. Chapter 2 has a section entitled "How this chapter is arranged"; it appears on the sixth page of the chapter, which seems odd..

The coding is good, but there are a few errors. On page 68, the author suggests that when shifting signed numbers to the left, the programmer should arrange to restore the sign bit in case it's shifted out; in fact, a change in the sign bit would be an overflow situation and there would be no repairing the damaged number. Example 5-14 on page 139 should add a value of two to location 22 rather than the value of one shown; most alert readers will pick this kind of thing up. Hardware purists may wince at some of the terminology, such as use of the term "grounded" instead of "logic zero" or "low", but everybody will understand what is meant.

The author has a conversational style and avoids jargon. Occasionally, there's a lapse: I'm sure that there's an easier explanation of immediate addressing than "the operand resides in the second byte of the instruction".

The author covers a lot of material in 231 pages of text. It's not light reading, but it is well presented. 6502 programmers and beginners will find it useful. ©

BIG FILES ON A SMALL COMPUTER

Elizabeth Deal
337 W. First Ave.
Malvern, Pa.19355

The program described here demonstrates a way of reducing data storage requirements by a factor of eight. It is written in Microsoft Basic for a PET computer.

I have seen several programs that create and use cross-index files for library search, statistical surveys and similar applications. They usually require large computers, such as a 48K system with two disk drives. A very thorough file handling system has been described recently by Dr. Sanger in the November, 1979, issue of **Microcomputing.** In his article each attribute is coded as two letters and six attributes are permitted for each record. This requires twelve letters and, therefore, twelve bytes.

In the method described here, each attribute is coded as yes or no and the user can have as many attributes as he desires. If the application lends itself to such coding into a list of keys or attributes, then this system will permit the handling of large amounts of data in core at one time. It also permits the use of logical AND, OR or NOT operators in retrieval with any combination of attributes.

By way of illustration, a library search requires quick access to those entries that contain desired subject matter. Two, three, or six byte coding of each key is very core consuming, and limits the number of records that can be in core at one time.

The solution I propose is twofold: (1) set up a smart coding procedure for classification of subjects described in an article into keys that can be scored yes or no, and (2) "pack" the data for storing it in core, on tape or on disk, and then "unpack" it, one record at a time, during the search for the applicable attributes. This paper describes an efficient way to "pack" and "unpack" the data so that a larger file can be searched on a small computer without the use of accessory memory devices, such as disks. Of course, if one has a system with a disk the method described here would permit use of an even larger file. We are aware that the first part of the solution (setting up the coding procedure) is challenging. It is the real problem and the performance of the system depends on how logical and meaningful the selected keys are.

Each logical record consists of the text part and the data part. The text part must be adequate for positive identification of the articles being searched, but the length should be kept to a minimum. Name, date, and page might be enough. The data part is what we can compress. The yes-no or 1-0 codes are entered in groups of fifteen ones and zeros. These, in turn, are packed into the two byte integer variable S%.

Fifteen attributes require two bytes, thirty attributes require four bytes, and so on. A user of the system need not concern himself with what the program does with binary numbers. He only needs to know that there will be as many S% values per record as there are groups of fifteen keys. The user then needs to provide a decision for retrieving the records of interest to him. The decision is written as a statement at the beginning of a program and is immediately edited for syntax-type errors. Logical operators AND, OR, NOT, as well as arithmetic ones (=, <>, <, >) are used. The decision can be written on one or more lines leading to a combining variable TR. TR is set to one if true, and all records meeting TR condition are then displayed. Complete instructions for writing TR lines are listed in lines 2970 to 3420.

How is it done? For once those long tables of powers of two, that are a part of every book on programming, come in handy. The program is set up in such a way that the user thinks of the list of fifteen keys from left to right, 1 to 15. The program sees them as being numbered from right to left, 0 to 14. Like this:

-Key numbers B%(k)k = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

-Program sees as m = 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

-Input key values 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0

The program now takes key values and wherever it finds a "1" it raises 2 to the m-th power. The sum of all this is then stored in integer variable S%(record number, sum number)*. The bytes are used instead of at least 15. During the process of retrieval the opposite procedure takes place - the sum is "unpacked" into working storage of 15 values. The same values are, of course, reused by all records. The lines of the program that drive this system are 1470 to 1510 and 1920 to 1990 the other way. It seems like a lot of hassle, but the core saving is tremendous. The loops that do the packing and unpacking take from 0.2 second to 0.9 second, the latter representing all fifteen bits on. (These times could be reduced by rewriting these two loops as machine code subroutines.) Another way to save time would be to set up the most frequently used keys next to one another as this will leave the loop sooner. In the example shown above, the program will loop ten times. Had a "1" been in position 4 or 11 the loop would be executed five times.

The program now has two sections. One packs the data, the other unpacks it. In between, the values should be stored on tape. And at the beginning, routines for creating and updating files should be provided. As listed, the program works as if it were a file system. It can be used as a training ground in writing decision lines. It should be used as a part of a larger system.

Buy Direct From CompuMart

COMPUMART NOW OFFERS THE ENTIRE DEC LSI-11 PRODUCT LINE - CALL FOR PRICES & DELIVERY.

Buy Direct from the Largest Commodore Dealer in the Country — NCE/CompuMart.

8K-Keyboard N	$795
16K-Keyboard B	$995
16K-Keyboard N	$995
32K-Keyboard B	$1,295
32K-Keyboard N	$1,295

B — large keyboard (graphics not on keys)
N — large keyboard with graphics symbols

SUPER SAVER. Thanks to Commodore their printer prices have been drastically reduced:

Commodore Printer (tractor feed) .... was $995 **NOW $795**
Commodore Printer (friction feed) .... was $849 **NOW $695**

2 Good reasons for buying your PET™ from CompuMart:

(1) $100 IN FREE ACCESSORIES WITH 16K or 32K PET.

When you buy a 16K or 32K PET, apply $100 toward PET accessories. FREE. Choose from the accessories listed below and indicate on your order that you have reduced the cost of your accessories by $100.

(2) FREE WITH PURCHASE OF ANY PET (Offer extended until May 15, 1980)

A Texas Instruments' Hex/Octal Calculator ($60 Value), and a set of 6 PET Workbooks (Worth $36)

PET Accessories

Commodore Dual Floppy Disk Drive	$1,295.00
Second Cassette—from Commodore	$95.00
Commodore PET Service Kit	$30.00
Beeper—Tells when tape is loaded	$24.95
Petunia—Play music from PET	$29.95
Video Buffer—Attach another CRT	$29.95
Combo—Petunia abd Video Buffer	$49.95
TNW Bi-Dir RS-232 printer S-face	$229.00
KIM 1 (A Single Board Computer from Commodore)	$179.00
PET TO IEEE Cable	$39.95
IEEE TO IEEE Cable	$49.95
KIM1 & Power Supply Package Special	$200.00

We have a complete inventory of Apple computers, peripherals, & software In-Stock for immediate delivery —

$200 in FREE accessories with the purchase of a 48K Apple II reg. or Apple II plus.

16K Apple reg. or plus — $1,195 ($100 in *free* acces.)
32K Apple reg. or plus — $1,395 ($150 in *free* acces.)
48K Apple reg. or plus — $1,495 ($200 in *free* acces.)

Apple Accessories

PASCAL	$495
Micromodem	$379
VisiCalc	$150
The Controller (Business Package)	SALE! $550
The Cashier (POS System)	SALE! $200
Integer BASIC ROM Card	$200
Centronics Printer Interface	$225
Disk and Controller	$595
Parallel Printer Card	$180
Communications Card	$225
Hi-Speed Serial Card	$195
Firmware Card	$200
NEW! AppleWriter Text Editor	$75

NEW FROM EXIDY THE SORCERER II 48K COMPUTER

Z-80 Microprocessor, Full-sized keyboard • ROM PAC Programs • Microsoft BASIC • Powerful Graphics • Serial and Parallel I/O • Dual Cassette I/O Memory Internally Expandable • S-100 Expansion Options

48K Sorcerer II $1,395

Commodore's 3 for 2 is Back! SPECIAL OFFER TO EDUCATORS - GET A FREE PET COMPUTER

For a limited time only, when your school buys any 2 PET's at the regular list (see our PET™ prices listed above), Commodore will include another PET in the deal, absolutely FREE! Call CompuMart TOLL-FREE for complete details.

NOVATION CAT™ ACCOUSTIC MODEM

- Answer Originate
- Bell 108
- 300 Baud
- Low Profile Design

Looks good, works great! **$179.00**

ZENITH

EXCLUSIVE FROM COMPUMART!

COLOR VIDEO MONITOR

Beautiful Display Capabilities. Excellent for use with Apple, Atari, & Sorcerer Computers. This 13-inch monitor is Zenith's first color video display designed specifically for computers. Features include color and degaussing circuits.

Zenith Color Monitor **$499.00**

CENTRONICS PRINTERS

The 779-2 is a tractor feed printer designed for small business systems. Uses a 5x7 dot matrix to produce all 64 upper case ASCII characters. Centronics is the biggest name in printers and this is their most popular model.

List. $1,245 ...... SALE! $1,079

The New 730-1 parallel matrix printer is ideally suited for all personal and microcomputer systems. Standard Features include: 100 cps • 80 char./line • 3-way paper handling system • 7x7 dot matrix • 96 character ASCII • List $995 .... SALE! $839

To meet our Centronics' quotas we are offering these great specials on their printers!

753-9 9 x 9 Data Processing Printer. List: $2,995 SALE! $2,395
704-9 9 x 9 180 CPS. List: $1,995 SALE! $1,750
701-9 9 x 9 Matrix. List: $1,595 SALE! $1,395
730-3 Serial 730. List: $895 SALE! $745

Brand New. Centronics Model 737, the first printer in its class capable of offering print quality suitable for text processing, plus the performance & application flexibility required for data processing. Unbelievable quality of type.

737-3 Serial Word Processor Printer	**$1,045**
737-1 Parallel Word Processor Printer	**$995**

The Perfect Printer for Small Business Systems

The Paper Tiger Printer From Integral Data

ROAR!

Standard features include: 4 character 8.3 to 16.5 cpi • 56 cps at 10 char. per in. • Selectable line spacing • 8 switch-selectable form sizes.

The IDS Graphics Option for the Paper Tiger allows full dot pattern control and includes an expanded 2048-byte buffer (a 256-byte buffer is standard).

IDS Paper Tiger Printer	**$995**
IDS Graphics Paper Tiger Printer	**$1,094**

SANYO MONITORS

9-inch reg. $199 **SALE! $169**

15-inch reg. $299 **SALE! $269**

Perfect for Apple users . . .
Sanyo Cassette Recorder $55

HAZELTINE TERMINALS AT SPECTACULAR SAVINGS!

Hazeltine 1410-TTY-style keyboard, 8 baud rates from 110 to 9600, 12" screen, 24x80 display, 5x7 dot Matrix Upper Case ASCII character set, 12-key numeric keypad. List Price $895 ...... SALE! $779

Hazeltine 1500 — 8 baud rates from 110 to 19,200, and ANSI Standard Keyboard, EIA RS-232, All 94 Upper/Lower Case characters in a high resolution 7x10 matrix display. Much more.

List Price $1,145 .... SALE! $995

Hazeltine 1520 — Take advantage of this $310 savings while it lasts! The Hazeltine 1520 is a powerful multi-processor terminal

List Price $1,585 .... SALE! $1,275

THE SINGLE BOARD DEVELOPMENT SYSTEM ROCKWELL AIM 65

6502 Microprocessor • 20-character, alpha-numeric LED display • Full-size 54-key keyboard with 3 user-defined functions • Fast, on-board 20-column thermal printer • 8K Advanced Interactive Monitor program • Dual cassette interface board • On-board timer • On-board ROM expansion to 12K • 4K on-board RAM • On-board TTY interface • 16 parallel I/O lines • One serial I/O port • KIM-compatible edge connectors for even further memory or I/O expansion.

The CompuMart AIM System combines all of our options for the AIM to give you the capabilities of development systems costing 5 to 10 times as much. This system includes a 4K AIM 65 with BASIC and Assembler, an MTU power supply, a Sanyo tape recorder and an EGI Enclosure for the AIM.

CompuMart AIM System	**$785.00**
4K AIM-65	**$450.00**
Paper for the AIM	**$4.75**

CompuMart

We've Had a Reputation for Dependability Since 1971.

Dept. K , 270 THIRD ST., CAMBRIDGE, MA. 02142
To Order: 1 (800) 343-5504
In Mass: 1 (617) 491-2700

master charge VISA

Member Computer Dealers Assoc.

To customize the program for your system (1) delete lines 2770 on to reduce core from 10K to 5.2K, (2) in line 1020 insert the maximum number of records that will fit, (3) modify G if you wish in lines 1020 and 1340 (G = 1 or 2 is now permitted), and (4) insert your machine size in line 2450.

As written, the program takes about 5.2K after removal of all REM lines that are at the end. A search type program that would contain a dictionary of keys should take no more than 6K. How many records can various systems handle? If we assume that each record has 26 bytes of text plus 30 yes-no attributes, an 8K system could search 66 records, a 16K system could search 330 records, and a 32K system could search 866 records. A 100K disk would add 3,333 records and a half-megabyte disk would add 16,500 records.

* *S%(nr,g)* $= 2^5 + 2^6 + 2^7 + 2^8 + 2^{10} + 2^{14} = 18,400$

Credits: Ken Brossoie
Microcomputing, November, 1979, page 44.
Neil Harris, A-B Computers, Montgomeryville, Pa.

```
1000 C0=0:C1=1:C2=2:C3=3:C5=5:C6=6:
     ¬CA=10:CE=14:CF=15:D$=""
1010 REM ELIZABETH DEAL,MALVERN,PA
1020 R= 200:G=2:DIM HD$(R),S%(R,G),
     ¬WA$(15),V$(G),B%(15*G),QQ%(15)
1030 FORJ=C1TOCF:READWA$(J):NEXTJ
1040 FORK=0TOCE:QQ%(K)=C2^K:NEXT
1050 :
1060 :
1070 REM RENUMBER 1000,10 WHEN NEEDED
1080 :
1090 REM>>>>  R E A D   T H I S  <<<<
1100 REM        INSERT DECISION
1110 REM  KEEP LST-TR-RTRN SEQUENCE
1120 :
1130 LIST-1280
1140 DX=ABS(B%(1)ORB%(2)):
1141 DY=ABS(B%(3)ORB%(4)):
1142 DZ=ABS(NOT(B%(5)ANDB%(6))):
1143 TR=ABS(DXANDDYANDDZ):
1150 RETURN
1160 :
1170 REM>>>>> ':' MUST END EACH LINE
1180 :
1210 REM TR=ABS((B%(1)ANDNOTB%(2)) //
1220 REM TR=ABS(S%(N,2)>512)WILL DIS-
1230 REM PLAY RECS MEETING STMT TR.
1240 :
1250 REM >> TYPE 'RUN', VERIFY, THEN
1255 REM    TYPE 'GOTO1300' TO CONT.
1260 REM       (OR ~ ~  THIS LINE#)
1270 :
1280 REM ========= SECTION 1 =========
1290 :
1300 A=TI:REM //CHOSE SEC.1/2 HERE ///
1310 GOSUB2420:REM      // SUBR 1   ///
1320 PRINT"ĥ":N=1:LT=0
1330 PRINT"HOW MANY GROUPS OF 15 KEYS ¬
     ¬?":PRINTTAB(9)"ENTER 1 OR 2"
1340 INPUTG:PRINT:IFG<1ORG>2GOTO1330
1350 PRINT:PRINT:PRINT"ENTER TEXT,
     ¬ OR 'XX' TO END INPUT"
1360 PRINTTAB(2)"!";TAB(27)"!"
1370 INPUTHD$(N):IFLEFT$(HD$(N),
     ¬2)="XX"THEN1550
1380 HL=LEN(HD$(N)):E=0
1390 REM // INPUT15 BITS,FLAG ERRORS
1400 :FORJ=C1TOG:PRINT:PRINT"15 KEYS";
1410 PRINT"  !!!!!!!!!!!!!!!" :PRINTTAB(
     ¬7);:INPUTV$(J)
1420 :FORLL=C1TOCF:S$=MID$(V$(J),LL,1)
1430 IF(S$<>"0"ANDS$<>"1")THENE=E+1:
     ¬GOTO1450
1440 :NEXT
1450 :NEXTJ
1460 IFE>0THENPRINT:PRINT"ɪERROR DO ¬
     ¬AGAIN":GOTO1350
1470 TX=TI :FORJ=C1TOG:S%(N,J)=C0
1480 :FORL=C0TOCF:PQ=VAL(MID$(V$(J),
     ¬CF-L,C1)):IFPQ=0THEN1500
1490 S%(N,J)=S%(N,J)+QQ%(L)
1500 :NEXTL:PRINTTAB(14)"SUM="S%(N,J)
1510 :NEXTJ:TY=TI
1520 PRINT"(";INT((TY-TX)/C6)/CA;"SEC)";
     ¬:PRINTTAB(25)"OK";N;"OF";R
1530 N=N+C1:LT=LT+HL:IFN<=RGOTO1350
1540 PRINT:PRINT:PRINTTAB(6):PRINT"*****
     ¬** NO MORE ROOM *******"
1550 PRINT:PRINT:PRINT:PRINT"# OF ¬
     ¬RECORDS PUT IN";N-1:PRINT
1560 PRINT"# OF BYTES USED BY TEXT";LT
1570 PRINT"# OF BYTES USED BY KEYS";
     ¬2*G*(N-C1):PRINT:PRINT"BYTES ¬
     ¬LEFT";FRE(0)
1580 PRINT:PRINT:PRINT
1590 PRINT"HIT 'S' TO STOP":PRINT"ANY ¬
     ¬KEY TO CONTINUE"
1600 GETA$:IFA$=""THEN1600
1610 IF A$="S"THENSTOP: REM/CHANGE/
1620 :
1630 REM // STORE DATA ON TAPE HERE
1640 :
1650 REM ======= SECTION 2 =========
1660 REM
1670 REM >> # OF RECORDS (NR), TEXT
1680 REM  REC (HD$(NR)) AND  G-SUMS
1690 REM  S%(R,G) ARE USED IN THIS
1700 REM  SECTION; BITS ARE COMPUTED
1710 REM  AND ASSIGNED TO KEYS ARRAY
1720 REM           B%(G*15)
1730 REM >> B% OR S%  ARE CHECKED FOR
1740 REM  COMPLIANCE WITH TR STMT.
1830 REM
1840 REM =============================
1850 REM
1860 PRINT"ĥ":NR=N-1
1880 :FORN=C1TONR:K=C0:JA=TI
1890 IFKS>C0THEN2000
1910 FORK=C1TOG*CF:B%(K)=0:NEXT
1920 JS=TI:FORJ=C1TOG:TP=S%(N,J):
     ¬JJ= J-C1
1930 IFTP=C0THEN1990
1940 Q=INT(LOG(TP)/LOG(C2)):U=CF-Q
1950 :FORM=C0TOQ:BP=QQ%(Q-M)
1960 IF(TP)>=BPTHENB%(CF*JJ+U+M)=C1:
     ¬TP=TP-BP
1970 IFTP=C0THEN1990
1980 :NEXTM
1990 :NEXTJ:JE=TI
2000 FORM=1TO4:PRINT"==========";:NEXT
```

```
2005 PRINT:PRINTHD$(N)
2010 GOSUB1140:IFTR<>1THENPRINT:
     ¬PRINT"*** NO MATCH ***":GOTO2030
2020 PRINT:PRINT"THIS RECORD MATCHES:"
2030 PRINT:PRINTTR$
2040 :FORJ=C1TOG:
2050 PRINT"GROUP";J;"SUM=";S%(N,J):
     ¬NEXTJ:PRINT
2070 IFKS>C0THEN2130
2080 K=C1:PRINT:FORJ=C1TOG
2090 PRINT"BIT";(J-C1)*(K-C1)+C1;:
     ¬PRINTTAB(7);" -> ";:PRINTTAB(12);
2100 :FORK=C1TOCF:PRINTRIGHT$(STR$(B%(CF
     ¬*(J-C1)+K)),1);
2110 :NEXTK:PRINT"  <- ";J*(K-C1)
2120 :NEXTJ
2130 PRINT:PRINT:PRINT"(TIME IN BIT ¬
     ¬LOOP";INT((JE-JS)/6)/10;"SEC)"
2140 PRINT"(TOTAL TIME:";INT((TI-JA)/6)/
     ¬10;"SEC)"
2150 :FORM=1TOCA:PRINT"====";:NEXT
2160 PRINT:PRINTTAB(11)"HIT ANY KEY FOR ¬
     ¬MORE"
2170 PRINTTAB(6)"'Q' TO QUIT AND RERUN ¬
     ¬OPTIONS"
2180 GET A$:IFA$=""GOTO2180
2190 IFA$="Q"GOTO2210
2200 :NEXTN :REM //  END REC LOOP  /
2210 PRINT"ĥ↓↓"TAB(16)"OPTIONS":PRINT
2220 PRINT"1. SAME TR, NEW DATA ¬
     ¬STARTING AT REC";NR+C1
2230 PRINT"2. RERUN: SAME TR AND DATA"
2240 PRINT"3. Q U I T"
2250 GETD$:IFD$<"1"ORD$>"3"GOTO2250
2260 ONVAL(D$)GOTO2270,1880,2280
2270 N=NR+C1:GOTO1350
2280 PRINT"↓↓↓SURE ?"
2290 GETA$:IFA$="N"GOTO2210
2300 IFA$="Y"THEN END: REM /PROG END/
2310 GOTO2290
2320 REM  === DATA=FOR SUBR.1 =======
2330 :
2340 DATA"NOT","","+","-","*","/","",
     ¬"AND"
2350 DATA"OR",">","=","<","","","ABS"
2360 :
2370 REM ==========================
2380 :
2390 REM         SUBROUTINE 1
2400 REM FIND AND EDIT TR STATEMENT
2410 :
2420 TR$="":LX=0:KB=0:KS=0:P=0:J1=1:
     ¬KC=0:L1=0:R1=0:SB=0:JA=0:JZ=0
2440 PRINT"ĥFOUND 'LIST' AT";
2450 :FORJ=1350TO32000-FRE(0):IFPEEK(J)=
     ¬155THENPRINT"   ***";J: JA=J
2460 IFPEEK(J)=142THENJZ=J:PRINT"FOUND ¬
     ¬'RETURN' AT ***";JZ:GOTO2475
2470 :NEXTJ
2475 IFJA=0ORJZ=0THENPRINT"CAN'T FIND ¬
     ¬DECISIONS; SEE SUB1 LISTING ":STOP
2480 :FORJJ=JA+11TOJZ:LL=PEEK(JJ)
2490 IFLL=58THENJJ=JJ+5:LP$=CHR$(13):
     ¬GOTO2540
2500 IFLL<128THENLP$=CHR$(LL):GOTO2540
2510 :FORM=168TO182:IFLL=MTHENLP$=WA$(M-
     ¬167):LX=LX+LEN(LP$):GOTO2530
2520 :NEXTM
2530 IFM=182THENSB=SB+1
2540 LX=LX+1
2550 IFLX>255THENPRINT"STRING TOO ¬
     ¬LONG-LIMIT 255":E5=1:GOTO2700
2560 TR$=TR$+LP$
2570 :NEXTJJ
2580 :FORJJ=J1TOLX+1
2590 M$=MID$(TR$,JJ,1):KB=KB-(M$="B"):
     ¬KS=KS-(M$="S"):P=P-(M$="%")
2600 KC=KC-(M$=","):L1=L1-(M$="("):
     ¬R1=R1-(M$=")")
2610 NEXTJJ
2620 PRINT" B ";" S ";" % ";" , ";" ( ";
     ¬" ) ";"ABS";" L "
2630 KB=KB-SB:KS=KS-SB
2640 PRINTKB;KS;P;KC;L1;R1;SB;LX
2650 PRINT:PRINTTAB(5)"YOUR DECISION IS:
     ¬":PRINT:PRINTTR$:PRINT
2660 E1=(KB>0)AND(KB<>P):E2=(KS>0)AND(KS
     ¬<>P):E3=(KS>0)AND(KS<>KC):
     ¬E4=(L1<>R1)
2670 IFE1ORE2THENPRINT"* USE % -(B% OR ¬
     ¬S%) !!":PRINT"  USE NO OTHER B, S,
     ¬ %"
2680 IFE3THENPRINT"* USE S%(N,#)FOR ¬
     ¬2-DIM ARRAY"
2690 IFE4THENPRINT"* ( ) DON'T MATCH:
     ¬";L1;"LEFT, AND";R1;"RIGHT"
2700 IFE1ORE2ORE3ORE4ORE5THENPRINT:
     ¬PRINT"TYPE 'RUN' TO FIX":STOP
2710 PRINTTAB(5)"HIT ANY KEY TO ¬
     ¬CONTINUE"
2720 PRINTTAB(3)"OR 'STOP' TO CORRECT/CH
     ¬ANGE"
2730 PRINT:PRINTTAB(5)"THEN TYPE  'RUN' ¬
     ¬ TO FIX"
2740 PRINT:PRINT"(EDITING AND RE-RUN ¬
     ¬WIPE OUT DATA)"
2750 GETA$:IFA$=""THEN2750
2760 RETURN
2770 REM ==========================
2780 REM EXAMPLE OF INPUT FOR ONE
2790 REM LOGICAL RECORD WITH TEXT
2800 REM AND 2 GROUPS OF 15 ATTRI-
2810 REM BUTES EACH, STORED AS
2820 REM 25+4 BYTES.
2830 REM
2840 REM >TEXT:
2850 REM    !                        !
2860 REM    1.MAG.NAME/11-78/  /P.106
2870 REM >FIRST 15KEYS
2880 REM    111010000001011
2890 REM >SECOND 15KEYS
2900 REM    001001000100010
2910 REM >END OF DATA
2920 REM    !                        !
2930 REM    XXXX
2940 REM
2950 REM ==========================
2960 REM
2970 REM POSSIBLE USES OF TR LINE(S)
2980 REM
2990 REM 1.DECISION IS WRITTEN AT THE
3000 REM START OF A RUN AND CANNOT BE
3001 REM CHANGED DURING THE RUN.
3005 REM
```

```
3010 REM 2.QUICK LISTING OF RECORDS
3020 REM WHICH MIGHT HAVE ANY KEYS
3030 REM ON WITHIN SOME GROUP --
3040 REM TR=ABS(S%(N,#)>=512 AND
3050 REM        S%(N,#)<=4096):
3060 REM WILL DISPLAY TEXT OF RE-
3070 REM CORDS THAT HAVE SOME BITS
3080 REM FROM 3 TO 6 ON (15-12,15-9)
3085 REM
3090 REM 3.SLOWER LISTING OF RECORDS
3100 REM       (TEXT AND BITS):
3105 REM
3110 REM 3.1 ALL RECORDS
3120 REM TR=ABS(B%(1)ORB%(1)=Ø):
3125 REM
3130 REM 3.2 SOME RECORDS - USE AND
3140 REM ,OR,NOT AS YOU DO IN LOGIC
3150 REM BUT RETAIN 'ABS' IN ALL
3160 REM EXPRESSIONS. END EACH LINE
3170 REM WITH A ':'. SIMPLE EXPRES-
3180 REM SIONS WILL SHOW UP VIA
3190 REM SUB1. FOR-LOOPS AND OTHER
3200 REM TOKENS WILL NOT,BUT THE
3210 REM PROGRAM WILL STILL RUN.
3240 REM
3250 REM 4.USE ( ) CAREFULLY -
3260 REM NOT(XANDZ)<>NOTXANDNOTZ
3280 REM WORXANDYORZ<>(WORX)AND(YORZ)
3290 REM
3300 REM 5.AVOID VARIABLES WITH B,S,
3310 REM % IN THEM - ONLY ABS AND
3320 REM NAMES LISTED ABOVE ARE LEGAL
3330 REM
3340 REM 6.DECISION LINES ARE SCANNED
3350 REM FIRST FOR OBVIOUS ERRORS, SO
3360 REM WE DON'T LOOSE DATA LATER
3370 REM BY EDITING THESE LINES.
3380 REM
3390 REM 7.LOGIC OF PROGRAM DEPENDS
3400 REM ON USE OF ABS FUNCTION -
3410 REM USE FORMAT: 'VAR=ABS(  ):'
3420 REM ON ALL LINES.
3430 REM ============================
3440 REM
3450 REM PURPOSE : FOR ANY INFORMA-
3460 REM TION PROCESSING WHERE DATA
3470 REM CAN BE CODED INTO Y/N FORM.
3480 REM
3490 REM STORAGE (CORE/TAPE/DISK)
3500 REM REQUIREMENTS ARE REDUCED BY
3510 REM AT LEAST A FACTOR OF 8 AS
3520 REM COMPARED TO CONVENTIONAL
3530 REM CODING OF KEYS FOR SEARCH.
3540 REM
3550 REM THIS SYSTEM WILL STORE 15
3560 REM KEYS-ATTRIBUTES-Y/N THINGS
3570 REM IN BASIC'S 2-BYTE INTEGER
3580 REM VARIABLE.
3590 REM
3600 REM TEXT SHOULD BE LIMITED TO
3610 REM MINIMUM THAT WILL POSITI-
3620 REM VELY IDENTIFY A RECORD.
3630 REM
3640 REM ===========================
3650 REM
3660 REM ELIZABETH DEAL, MALVERN,PA
```

Using The PET's Second Cassette Buffer To Increase Available Memory Space

Chuck Stuart, President
CMS Software Systems, Inc.
5115 Menefee Drive, Dallas, TX 75227

If you have tried all the known memory saving tricks to reduce the size of your new super program to save mankind and it still won't fit into your PET's available memory, you might be able to save the day by storing part of the program in the PET's second cassette buffer.

The second cassette buffer begins at address 826 and extends through address 1023. If your program does not use the second cassette then you can use this 197 bytes for program storage. Granted, 197 bytes may not seem like a lot of extra space, but used correctly it can save you several times that much space in the PET's regular memory.

Suppose, for instance, your program requires a deck of 52 cards to be stored in memory. Such a deck would normally contain the values 1 through 52 and be stored in an integer array referenced by a name like D%. Since each element in an integer array requires two bytes of memory and there are 52 elements in our array, we can see that, stored as a normal integer array, our deck of cards would require 104 bytes of storage space plus at least another 7 bytes and probably more to properly dimension the array.

Since no array element value exceeds 255, this array could easily be stored in the second cassette buffer with each element requiring only one byte for storage. By moving our deck of cards into the second cassette buffer we have saved a minimum of 111 bytes of main memory while using only 52 bytes of the 197 available to us in the second cassette buffer. This same method works equally well for single element integer variables and strings.

If you simply use this extra space for temporary storage of information during program operation then no special problems are created. If you make full use of the second cassette buffer for permanent storage of program data such as operator prompts, suit symbols for your card deck, or other values normally stored in DATA statements, etc., then the second cassette buffer must be stored along with the main program. Since the SAVE command begins saving at location 1024, any data stored below that address in the second cassette buffer would not be saved by using the standard SAVE command.

Fortunately by using the PET's built-in ROM Monitor we can specify the exact beginning and ending addresses we want to save. If you still have the old ROM set without a Monitor don't panic. I'll show you how to save the buffer a little later.

The first thing you have to do is determine the address range you want to save in hexidecimal. As we pointed out before, the second cassette buffer begins at location 826 decimal or 033A in hex so we know that we want to start saving at location 033A hex. To determine the ending address you must first save your program onto tape using the normal SAVE command, and then load it back into memory with the LOAD command. Data stored in the second cassette buffer will not be affected by this operation. Now type PRINT PEEK(43)*256 + PEEK(42). The answer you get will be the address in decimal where your program ends and which you must now convert to hex. (If you've never taken the time to learn how to convert decimal to hex and vice versa because you figured you would never need to know then you were wrong.)

Now that you know the memory range you want to save, enter the Monitor by typing SYS1024 and pressing RETURN. To save your program including the data stored in the second cassette buffer, type the following line exactly, substituting the ending address for XXXX and filling in your own program name, and press RETURN:

S "PROGRAM NAME",01,033A,XXXX

PET will display the standard PRESS PLAY AND RECORD message and then save your program on tape in the normal manner except it will begin saving at location 826 instead of the usual location 1024 where your BASIC program begins. You can LOAD and use the program in the normal manner except that you must always use the Monitor to save the program or else you will lose any data stored in the second cassette buffer. This means that no one else will be able to copy your program unless they know the secret. There is one other thing to remember. If you make any changes to your program, don't forget to redetermine the new ending address before using the Monitor to resave the program.

If you have the old ROM set without a Monitor then you can accomplish the same result in the following manner:

A) Save the program with the SAVE command and then reload it with the LOAD command.
B) Type POKE247,58:POKE248,3 and press RETURN.
C) Type SYS63153 and press RETURN.

PET will display the standard PRESS PLAY AND RECORD message and save the program beginning

at location 826. If you want your program to have a tape name, then in addition to step B above you must execute the following steps between step A and step B.

B1) POKE the ASCII value of each letter of the name into the buffer beginning at location 826.

B2) POKE238,N where N equals the number of letters in the name and press RETURN.

B3) POKE249,58:POKE250,3 and press RETURN.

Now execute step C and your saved program will have a name. (You old ROMers got out light by not even having to perform hex/decimal conversion.)

If you stayed with me all the way through then you should have learned three new things about your PET. How to use the second cassette buffer for data storage, how to save programs using the Monitor, and how to protect your programs from being copied. There is one final hint about reducing program memory requirements I would like to pass along to those using the PROGRAMMER'S TOOLKIT. Before saving the final version RENUMBER the program in increments of 1 beginning at line 0. This will save you an average of two bytes for every line number referenced in any type of GOTO or GOSUB statement.

I'll be glad to answer any questions concerning your PET if you will include a SASE with your request. ©

ENHANCING COMMODORE'S WORD PRO III

Larry Isaacs, COMPUTE Staff

This article will deal with enhancing WordPro III, as was done for WordPro II in the previous issue of COMPUTE. The enhancement is designed for those who are using WordPro III on a CBM, and are using the NEC Spinwriter for output. Most Spinwriter print thimbles have nine characters which normally can't be printed by WordPro III. The enhancement will allow you to embed special characters in your text that will print as one of the Spinwriter characters not accessible by WordPro III alone.

The special characters are entered in your text by hitting the ESC key followed by a number from 1 to 9. If some other key is struck following the ESC key, the normal character is entered. Also, the ESC key must be struck for each special character you wish to enter. The section on ASCII output shows the ASCII data which is output for each of the nine special characters. This section also shows the characters printed for two of the available print thimbles.

This article also has a section which discusses how to make modifications. This section will cover both the enhancement presented in this article as well as the enhancement presented in the previous COMPUTE. for WordPro II. One of the areas discussed is the modification necessary to use the enhancements on PETs with graphics keyboards. These PETs don't have an ESC key (or the REPEAT key used by the WordPro II enchancements).

The Upgrade Procedure

The two machine code patches presented here are for WP3 (FEB 5), the current production version of WordPro III. These patches will be entered using the machine language monitor in the PET. Commands in this section for the machine language monitor begin with a period. This period is a prompt character printed by the monitor. To execute one of these commands, just type the rest of the command and hit RETURN.

Before entering the patches, first place the work copy of your WordPro diskette in drive 0 and initialize the diskette. Now type SYS1024 plus carriage return to get into the machine language monitor. Execute Command 1, and enter the first patch using the PET's screen editor to change the displayed memory to what is shown beneath the command. Be sure to hit a carriage return when you reach the end of each line. The carriage return causes the changes to be placed in memory. Once you have finished the first patch, enter the second patch using the same procedure for Command #2. Now execute Commands #3 and #4 to save the patches on disk.

Command #1. **M 05C0 064C**

```
.
.:  05C0 00 00 00 00 00 00 20 CC
.:  05C8 05 4C 02 06 BA BD 06 01
.:  05D0 8D C2 05 09 04 9D 06 01
.:  05D8 BD 07 01 8D C0 05 BD 08
.:  05E0 01 8D C1 05 18 68 69 01
.:  05E8 9D 07 01 68 69 00 9D 08
.:  05F0 01 4C 55 E4 AD C2 05 48
.:  05F8 AD C3 05 AE C4 05 28 6C
.:  0600 C0 05 8D C3 05 8E C4 05
.:  0608 A6 9E F0 E8 BD 6E 02 C9
.:  0610 1B F0 14 4E C5 05 90 DC
.:  0618 C9 3A B0 D8 C9 31 90 D4
.:  0620 09 80 9D 6E 02 D0 CD CA
.:  0628 A9 01 8D C5 05 86 9E D0
.:  0630 C3 78 A9 C6 85 90 A9 05
.:  0638 85 91 A9 4C 8D E3 36 A9
.:  0640 E0 8D E4 36 A9 43 8D E5
.:  0648 36 58 4C D7 14 AA AA AA
.
```

Command #2 **.M 43E0 442F**

```
.
.:  43E0 C9 5B 10 03 4C E7 36 C9
.:  43E8 71 90 37 C9 7A B0 33 A6
.:  43F0 27 F0 0C 48 A9 5F 20 3D
.:  43F8 2E A9 08 20 3D 2E 68 29
.:  4400 0F AA E0 08 10 09 BD 25
.:  4408 44 20 3D 2E 4C 35 37 A9
.:  4410 0E 20 3D 2E BD 25 44 20
.:  4418 3D 2E A9 0F 20 3D 2E 4C
.:  4420 35 37 4C F3 36 00 5E 5C
.:  4428 60 7B 7D 7C 7E 58 5A 00
.
```

Command #3 **.S "0:PATCH 1",08,05C0,064D**

Command #4 **.S "0:PATCH 2",08,43E0,4430**

This completes the upgrade procedure. If you want to return to BASIC, type "x" plus carriage return. To run an upgraded WordPro III, place the diskette containing the patches in drive 0, and execute Commands #5 through #9 in sequence. If you find that your enhanced WordPro won't come up or operates improperly, it's likely you've made an error in the first patch (the one with Command #1). If it runs and you are able to enter the special characters, but fouls up when you try to print something, you've likely made an error in the second patch.

Command #5 **OPEN1,8,15,"I0"**
Command #6 **LOAD "WP3(FEB 5)",8**
Command #7 **LOAD "PATCH 1",8**
Command #8 **LOAD "PATCH 2",8**
Command #9 **SYS1585**

The ASCII Output

Table 1 shows the ASCII data that is output for each of the special characters. Notice the Spinwriter requires a three code sequence for two of the characters on the print thimble. Also given in the table are the characters printed for two of the thimbles. One is the Courier 10, which has a standard ASCII character set. The other is the Courier Legal 10B, which has some legal symbols substituted for some of the standard ASCII characters.

Modifying The Patches

Though the machine code patches are different, the enhancements for WordPro II and III operate almost identically. To cover modification of both enhancements, I will reference WordPro III modifications directly and put the corresponding reference for WordPro II in brackets.

First let's look at what needs to be changed to use the enhancement with a graphics keyboard. At one point in the patch code the last keystroke in the keyboard input buffer is checked to see if it is $1B [$7F for WP2), which is the code for the ESC [REPEAT] key. To change the key just change the byte used in the comparison. This byte is the first byte in the line that begins at address $0610 [third byte of the line that begins at address $028A]. Changing this byte to $DE will allow you to use a shifted uparrow instead.

Next, let's look at changing the ASCII data output for the special characters. This will allow you to use a different printer and possibly change the function of the enhancement. The patch code that handles the output checks each character to be printed if it is in the range $71 to $79. Only the special characters used by the enhancement will be in this range. If a special character is found, the most significant four bits are discarded, and the least significant four bits used as an index to a table. The table begins with the sixth byte of the line with address $4420 [first byte of the line with address $3FDD]. The first byte would be output for a special character with the value $70. Currently this value isn't used so the byte is $00. The next byte would be output for a character with the value $71, and so forth up to $79. The data in this table may be changed to whatever you require.

It may be necessary to also change which special characters output the three code sequence. Once a special character has been detected and its index obtained, that index is compared with the third byte of the line with address $43F8 [second byte of the line with address $3FBD]. If the index is equal to or greater than this byte, the three code sequence is used. Changing this byte to $0A would cause all the special characters to output only the data byte in the table. To change the first and third characters used in the sequence, set the first byte of the line with address $4410 [seventh byte of line with address $3FC5] to the first character of your sequence, and the fourth byte of the line with address $4418 [second character of line with address $3FD5] to the third character of your sequence.

One other modification is to add a special character for the 0 key. To do this, change the sixth byte of the line with address $0618 [last byte of the line with address $0292] to $30, and the first byte of the line with address $43e8 [seventh byte of the line with address $3fad] to $70. Then put the byte you want to be output in the first byte of the table.

The above extension was prompted by feedback from readers wanting to adapt the WordPro II enhancement for use with a different printer. Hopefully the discussion will be sufficient for most situations. If you do make mods for another printer, please send them to COMPUTE. so we can pass along to other readers.

TABLE 1

Special Character	ASCII Characters (Hex) Output	Thimble Character Printed Courier 10	Courier Legal 10B
1	5E	^	©
2	5C	\	®
3	60	`	°
4	7C	|	¶
5	7D	{	§
6	7B	}	†
7	7E	~	™
8	0E,58,0F	¬	‗
9	0E,5A,0F	£	|

Reproduce and Attach Above Numeric Pad

7=~	8=¬	9=£
4=|	5={	6=}
1=^	2=\	3=`

7=™	8=‗	9=|
4=¶	5=§	6=†
1=©	2=®	3=°

©

Machine Language Tools

Jim Butterfield

The newcomer to machine language programming will need to provide himself with certain tools before he gets started.

It's important to have the right computer. The original ROM which came with early PETs can be used with machine language, but it's clumsy for this kind of use. If you haven't done so already, make the switch to upgrade ROM: you'll get a built-in Machine Language Monitor and the capability to dig your way out of a program crash.

Get the official reference: **6502 Programming Manual.** It may be obtained from Commodore (MOS Technology), Synertek or Rockwell, who are the three manufacturers of 6502 microprocessors. It's not a teaching book, but a reference - although many of the early chapters are set out in tutorial style.

If you see other books that seem to suit your needs, you may buy them, of course. Avoid books that deal with "somebody else's" machine - you won't learn much about the PET from books on KIM, SYM, Apple, OSI, AIM or Atari. At a later stage, you might like to buy such books in order to try your hand at converting the programs to PET - or just to get an idea as to what's in other machines.

General books on the 6502 are good. Remember that you'll have to fit what these books say into the framework of the PET architecture. Leventhal's **6502 Assembly Language Programming** (Osborne/McGraw Hill, Inc.) is full of programming examples; but they will all need to be trimmed up to fit the PET. Don't let that worry you - it's not hard, and it's good practice.

You have very little need for hardware unless you have a special project in mind. If your machine has upgrade ROM, try building or buying an un-crashing connector (see COMPUTE, issue 1, page 89). Your early programs will certainly "crash" and you'll save a lot of reloading or retyping if you can re-awaken your machine. Most non-PET machines have a Reset button that will do the same job, which is very convenient except when you hit it accidentally.

The Machine Language Monitor is very useful. It's built into upgrade ROMs. Original ROM machines can load a MLM from tape, but this is much less convenient especially since it uses up memory space that you might want for your own purposes. You may enhance the usefulness of your MLM by obtaining more elaborate versions. Many of these are available from the PET Programs On Tape Exchange (P.O. Box 516, Montgomeryville, PA 18936). *(Editor's Note: See Table of Contents. RCL)*

If you're staying with original ROM, you might like Bill Seiler's NEWMON which has many extra functions including a tiny assembler. Another useful package is Jim Russo's high monitor with machine language single-step; this is sometimes called MT6671 and sometimes DISS/STEP.

The upgrade ROM monitor is built in, but there are a couple of enhancement packages that add extra functions: SUPERMON and EXTRAMON. Both of these contain powerful extras: disassembler, tiny assembler, memory hunt, and single-stepping.

If you get deeply into machine language programming, you'll want to obtain a full assembler. These are available from commercial sources. Don't get an assembler right away, especially if you're thinking of adding memory or a disk system. Your beginning programs will be quite small, and easy to hand assemble; what's more, you'll get a better feel for how the machine works by doing it all yourself. Later, as your programs start to exceed thirty lines of code or so, you'll find that an assembler will be a big help. This is especially true when you want to make changes to a previously written program. Since few of us write perfect programs every time, the capability of making changes easily is a major advantage that assemblers give you.

The most important tool you need for machine language programming is one you've already got: your brain. Remember that each machine language instruction is simple and logical; a program gets complex only when you put many instructions together. Be prepared to change your style when you go to machine language; a lot of thinking and planning should take place before you write your first line of code.

If bits and logical operations are not familiar to you, you'll need to do some advance brushing up. Learn why you can store values only from 0 to 255 in a memory location; how a negative number is represented in binary; how to relate binary, hexadecimal and decimal numbers; and how the logical functions (AND, OR, and EOR) work. There are plenty of introductory texts around to help. ©

Algebraic Input for the PET

Harvey Davis
Department of Mathematics
Michigan State University

One of the first experiments that I made when I started using the PET was to input the character "π" as a numeric value. Of course it failed. But the possibility of inputing general expressions such as 2*π*A and SQR (2 + A*B) has always been a dream of mine. The following program allows one to do just that.

The heart of the program is the subroutine beginning at line 20. After an algebraic expression has been inputed as X$, line 30 clears the screen and prints a line such as

X = SIN(A*B +π)

at the top of the display. Lines 35 and 36 print the command CONT at the beginning of the 4th and 5th lines on the screen. If after this was done the user would STOP the program from the keyboard and then type the sequence [HOME], [RETURN], [RETURN] the computer would evaluate the expression and then continue the program where it had left off. (The second CONT is there in case the expression overflowed onto the second line.)

Many users have noticed that stopping a program causes the PET to print the contents of the key stroke buffer onto the screen. See, for example the article "Dynamic Keyboard" by Mike Louder dated July 7, 1978 and appearing on page 7 of the Pet User's Group Newsletter volume 0, number 4.

Mr. Louder writes: "The decimal addresses for the keyboard buffer are 527 through 536. The buffer counter address is 525.

"If a BASIC program is interrupted with a STOP or END the keyboard monitor searches the keyboard buffer and executes any 'ASCII' instruction that may have been typed in while the program was running."

Mr. Louder goes on to explain how this characteristic of the PET may be used to modify programs and add a computed GOTO to the user's arsenal of techniques. The following technique is similar to his computed GOTO except that it causes an expression to be evaluated and a continue to be executed.

Line 40 sets the buffer counter index at 3 which tells the computer that there are three characters in the buffer. Line 45 loads the ASCII values of the commands [HOME] (= 19) and [RETURN] (= 13) into the keyboard buffer. When the program is stopped at line 50 these commands are executed, forcing an evaluation of the expression at the top of the display and then the execution of the CONT command. The computer then continues running and all variables, including the new value for X are intact. A simple program is provided at line 100 calling this subroutine so that the reader may experiment with it.

Finally a word of caution. If the expression has a syntax error in it, the program must be run from the beginning and all variables will be lost.

As a postcript I would like to post the following questions:

1. Can the above process be done without the disconcerting writing to the screen, possibly by momentarily convincing the PET that the screen is somewhere else?
2. As mentioned above, programs can be dynamically updated. Can they be updated in such a way as to leave the varibles undisturbed? This would be quite useful in programs that work on functions such as plotters and integraters.

PROGRAM LISTING

```
 10 Q$=CHR$(34):GOTO100
 20 PRINT"[CLR][DN][DN][DN]";
 21 PRINT"X MAY BE ANY EXPRESSION"
 25 INPUT"[HM]X=";X$
 30 PRINT"[CLR]X=";X$
 35 PRINT"[HM][DN][DN][DN]CONT"
 36 PRINT"CONT"
 40 POKE525,3
 45 POKE527,19:POKE528,13:POKE529,13
 50 STOP:RETURN
 60 PRINT"[CLR][DN][DN][DN]";
 61 PRINT"A,B AND C MUST BE NUMBERS"
 65 INPUT"[HM]A,B,C=";A,B,C:RETURN
100 REM SAMPLE MAIN PROGRAM
110 GOSUB60:GOSUB20
120 PRINT"[CLR]"
125 PRINT"IF X =";X$:PRINT
130 PRINT"   AND A =";A
131 PRINT"       B =";B
132 PRINT"       C =";C:PRINT
133 PRINT"   THEN X =";X:PRINT
135 PRINT"TYPE ";Q$;"N";Q$;
136 PRINT"FOR NEW VALUES OF A,B,C"
137 PRINT"     ";Q$;"X";Q$;
138 PRINT"FOR A NEW EXPRESSION"
139 PRINT"     ";Q$;"B";Q$;
140 PRINT"FOR BOTH"
141 PRINT"     ";Q$;"E";Q$;
142 PRINT"TO END THE PROGRAM"
150 GET A$:IF A$=""GOTO150
155 IF A$<>"N"GOTO160
156 GOSUB60:GOSUB30:GOTO120
160 IF A$="X"THENGOSUB20:GOTO120
165 IF A$="B"THENGOTO100
170 IF A$="E"THEN END
190 GOTO120
```

©

PET DATA COPIER

Ron Straley
1868 Grade Ave.
Ft. Myers, Fla33901

Having written and used many programs that use data files, I know the value of having backup tapes in case of an accident.

Here is a program I wrote to solve my problem, and maybe it will help others in their line of program running save time and trouble.

It is a data tape copier for the 'PET' computer that will copy any 'PET' data file or data tape so you will have a backup data copy.

It has provisions in it to input a file name if your tape has one or else just copy a plain data file or tape.

The program has full prompts on the screen. There is also a routine to let you look at what is written on your new data tape. It can display all of your data on the screen. This is an operator selected routine.

Line 100 is start of data tape to copy load routine.
Line 430 is start of new data tape write routine.
Line 630 is start of read and display of new data tape routine.
Line 960 is cassette advance routine for space between blocks.

```
20 REM           BY
30 REM       RON STRALEY
40 REM    1868 GRACE AVE.
50 REM   FT.MYERS,FLA.33901
60 REM
100 PRINT"ĥTHIS IS A DATA TAPE COPY ¬
     ¬PROGRAM."
110 PRINT:PRINT"IT WILL COPY UP TO 450 ¬
     ¬STRINGS.":PRINT
120 PRINT"TO CHANGE, REDIMINSION'D$'IN ¬
     ¬LINE 170":PRINT:PRINT"AND'M'IN ¬
     ¬LINE190.
130 PRINT:PRINT:PRINT"****INSERT DATA ¬
     ¬TAPE TO BE COPIED****"
140 :
150 REM**READ IN DATA TAPE***
160 :
170 CLR
180 DIMD$(450)
190 T=128:F=64
200 M=450
210 PRINT:PRINT:PRINT"DOES DATA TAPE ¬
     ¬HAVE FILE NAME?(Y OR N)"
220 GETA2$:IFA2$=""GOTO220
230 IFA2$="Y"GOTO260
240 IFA2$="N"GOTO280
250 GOTO210
260 PRINT:INPUT"DATA TAPE FILE NAME?";
     ¬A1$
270 OPEN1,1,0,A1$
280 OPEN1,1,0
290 A=0:PRINT"ĥ"
300 INPUT#1,D$(A)
310 IF(ST=T)ORST=FTHEN370
320 A=A+1
330 D$(A)=A$
340 PRINT"ĥ             WORKING"
350 IFA<MTHEN300
360 :
370 REM*END OF FILE (64) OR TAPE (128)*
380 CLOSE1
390 :
400 REM**WRITE NEW DATA TAPE**
410 :
420 PRINT"ĥREMOVE TAPE TO BE COPIED."
430 PRINT:PRINT"NOW INSERT CASSETTE TO ¬
     ¬COPY TO IN DRIVE#1"
440 PRINT:PRINT"PRESS ANY KEY"
450 GETB$:IFB$=""GOTO450
460 IFA2$="Y"GOTO480
470 IFA2$="N"GOTO490
480 OPEN1,1,1,A1$:GOTO500
490 OPEN1,1,1
500 IFA=0THEN570
510 FORI=1TOA
520 A1$=D$(I)
530 :
540 GOSUB830:REM**WRITE ROUTINE**
550 :
560 NEXTI
570 CLOSE1
580 PRINT"ĥDATA TAPE NOW COPIED"
590 :
600 REM*READ NEW DATA TAPE TO VERIFY*
610 :
620 CLR:PRINT
630 PRINT"DO YOU WISH TO READ AND ¬
     ¬DISPLAY NEW DATA";:PRINT
640 PRINT"TAPE? (Y OR N)"
650 GETB$:IFB$=""GOTO650
660 IFB$="N"GOTO1020
670 IFB$<>"Y"GOTO600
680 PRINT"ĥREWIND TAPE,THEN PRESS ANY ¬
     ¬KEY"
690 GETB$:IFB$=""GOTO690
700 DIMD$(450):T=128:F=64
710 OPEN1,1,0
720 A=0
730 INPUT#1,A$
740 PRINTA$:FORJ=1TO150:NEXTJ
750 IF(ST=T)ORST=FTHEN780
760 A=A+1
770 GOTO730
780 CLOSE1
790 PRINT"THATS ALL THE DATA I CAN ¬
     ¬FIND."
800 PRINT:PRINT"PRESS ANY KEY TO COPY ¬
     ¬NEXT FILE OR DATA TAPE"
810 GETA$:IFA$=""GOTO810
820 GOTO100
830 REM**WRITE ROUTINE**
840 :
850 T1=TI:IFLEN(A1$)<190THEN900
860 PRINT#1,LEFT$(A1$,189);
870 IFTI-T1>120THENGOSUB940
880 T1=TI:PRINT#1,RIGHT$(A1$,LEN(A1$)-18
     ¬9)
890 GOTO910
900 PRINT#1,A1$
910 IFTI-T1>120THENGOSUB940
920 RETURN
930 :
940 REM**CASSETTE ADVANCE**
950 :
960 POKE59411,53:T1=TI
970 IFTI-T1<6THEN970
980 POKE59411,61:RETURN
990 :
1000 REM** ANY MORE COPIES?" **
1010 :
1020 PRINT:PRINT"ĥDOYOU WISH TO COPY ¬
      ¬ANY MORE DATA FILES"
1030 PRINT:PRINT"OR DATA TAPES? (Y OR ¬
      ¬N)"
1040 GETB$:IFB$=""GOTO1040
1050 IFB$="Y"GOTO100
1060 IFB$<>"N"GOTO1020
1070 END
```

Plus! ALL NEW *from* MICRO-ED!
MICRO-DOZEN TWO

- **AGREEMENT OF SUBJECT AND VERB (Elementary and up)**
  Problems consist of sentences with missing verbs. Two verb choices (singular and plural) are provided for each problem. The learner types the correct form of the verb.

- **RUN-ON SENTENCES (Elementary)**
  Each problem consists of a group of words that should be divided into two sentences. Using the space bar, the learner moves an arrow across the screen and places it after the word that should end the first sentence.

- **MAKE A SENTENCE (First Grade)**
  Some scrambled words appear on the screen. The learner must use some of these words (but only these words) to type a complete sentence.

- **MORE WORD DEMONS: ITS-IT'S-YOUR-YOU'RE (Elementary and up)**
  Each lesson consists of twenty problems. There are sentences with blanks to be filled by the appropriate homonyms. At the end of the lesson, the learner's performance is summarized. Among other things, this summary lists the specific homonyms that may have given the learner trouble during the lesson.

- **HIGHER, SAME, LOWER (Elementary Music)**
  Each lesson consists of ten problems in which two notes are played. Is the second note higher than, lower than, or the same as the first note? Each pair of notes is randomly selected from a bank of sixty-seven pairs.

- **MATCHING RHYTHMS (Elementary Music)**
  The computer randomly selects one of ten rhythmic patterns and plays this as it displays matching quarter notes and eighth notes on the screen. The learner attempts to match the rhythm by pressing the space bar.

- **TARGET MATH (Elementary)**
  The learner selects problems in addition, subtraction, multiplication, or division at a difficulty level of his/her choice. A target area appears at the bottom of the screen, along with a math problem. When the learner types an answer, a plane flies over and a parachute jumper floats down over the landing strip. If the jumper hits the target area, the next problem is presented.

- **WORKING WITH BASIC MULTIPLICATION FACTS (Elementary)**
  Each lesson consists of fifteen problems. If the learner wishes, he/she may practice multiplying by a certain number. Problems are posed at random and are not repeated unless they are missed.

- **WORKING WITH BASIC ADDITION FACTS (Primary Grades)**
  Each lesson consists of fifteen problems. If the learner wishes, he/she may practice adding only a certain number. Problems are posed at random and are not repeated unless they are missed.

- **ADDING OR SUBTRACTING TWO-AND-THREE-PLACE NUMBERS IN COLUMNS (Elementary)**
  This program presents addition or subtraction problems using two-and-three-place numbers in columns. The learner works with each problem beginning with the units column on the right, (as would be done with paper and pencil) and moving left as guided by an arrow displayed under the appropriate column.

- **SUBTRACTING WITH OBJECTS (Primary Grades)**
  Each lesson presents ten randomly generated subtraction problems with numbers from one to ten being subtracted from each other. Small arrows and diamond shapes represent this operation.

- **ADDING WITH OBJECTS (Primary Grades)**
  Each lesson presents ten randomly generated addition problems with sums up to twenty. Small diamond shapes representing the numbers to be added appear on the screen.

All 12 Tapes for $84.00

AVAILABLE this SUMMER

- Synonym Series
- Punctuation Series
- Reading Racer Series
- Vocabulary Series

Details concerning these new series available upon request.
We welcome suggestions for educational programs.

PET TWO-WAY RS-232 and PARALLEL OUTPUT INTERFACE

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
- PET IEEE connector for daisy chaining
- 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

Order direct or contact your local computer store.

CONNECTICUT microCOMPUTER, Inc.
150 POCONO ROAD
BROOKFIELD, CONNECTICUT 06804
TEL: (203) 775-9659 TWX: 710-456-0052

VISA AND M/C ACCEPTED - SEND ACCOUNT NUMBER, EXPIRATION DATE AND SIGN ORDER.
ADD $3 PER ORDER FOR SHIPPING & HANDLING - FOREIGN ORDERS ADD 10% FOR AIR POSTAGE.

OHIO SCIENTIFIC EXPANSION INFORMATION

Conversion of C1P (Cassette) to 52x26 display. Detailed step-by-step instructions for doubling the C1P speed and display size!
Order Bulletin 1105 — $12.00

Conversion of C1P (Minifloppy) to 52x26 display. Same as above but includes display driver for disk operating system.
Order Bulletin 1105MF — $20.00

Conversion of C2-4P or C2-8P to 8 in. or 5-1/4 in. floppy. Instructions and schematics for converting cassette-based units into floppy systems.
Order Bulletin 1106F — $20.00

OHIO SCIENTIFIC EXPANSION ACCESSORIES

We have Ohio Scientific boards, cables, schematics, and documentation. Hard-to-find items.

Most Bare Boards	$39.00
430 Super I/O Board (Bare)	$69.00
Data Separator for MPI Minifloppy	$25.00
470 Universal Disk and Real-Time Clock Board (Tested)	$295.00
Dual Mini-Flopy Breakout Cable	$34.50
CP1 to S-44E Bus Converter	$39.95
Ohio Scientific Price List	FREE

TRS-80 DISK EXPANSION

New do-it-yourself TRS-80 expansion board has floppy controllers, quiet 32K RAM, serial RS-232, parallel printer ports, active termination. Super documentation. Software compatible, by LNW Research.

Bare Board and Manual	$89.95
Manual Only	$ 6.95

New Book "TRS-80 Disk and Other Mysteries" by IJG Computer Services. Essential reference manual for TRS-80! 130 pages — $22.50

Silver Spur

ELECTRONIC COMMUNICATIONS CO.
USA:
P.O. Box 365
Chino, CA 91710
Tel.: (714) 627-9366

ELPACK

Components and Kits for Electronics and Science

ELPACK "STAR WARS" SOUND EFFECTS BOARD

A universal circuit board that uses the TI SN76477 Sound Effects Generator IC. Birds, dogs, space sounds, etc.

Bare Board and Manual	$ 8.95
Kit and Manual	$24.95
Manual Only	$ 1.00

ELCOMP Book: "Complex Sound Generation Using TISN76477," by Texas Instruments. — $6.95

ELPACK PRINTER CONTROLLER BOARD

A universal circuit board that uses the CY-480 Printer Controller IC to drive most 40 and 80-column printer mechanisms! Includes power supply. RS-232 input.

Bare Board and Manual	$69.50
Manual Only	$ 5.95

ELPACK S-BAND TV CONVERTER BOARD

Allows reception of microwave TV on ordinary TV sets. Comes with antenna construction plans.

Bare Board and Manual	$21.00
Manual Only	$ 3.95

ELPACK VHF/UHF AMPLIFIER BOARD

This board uses the popular "CA" package wideband hybrid amplifiers, including such TRW types as CA2800 through CA2880, and others. Large choice of modules to 1000 MHz and 2 watts.

Bare Board and Data	$10.95
Data Only	$ 1.00

ELPACK MPI DATA SEPARATOR BOARD

Required for OSI minifloppy systems, this data separator plugs into MPI 51/52 drives to provide read data and read clock on separate lines. Improves inner track disk performance.

Bare Board and Data	$ 6.95
Assembled Board and Data	$25.00

Europe Distributor:
(Request DM-Preis Order Form)
HOFACKER VERLAG
Tegernseerstr. 18
D-8150 Holzkirchen/Obb.
Tel.: 08024/7331

ELCOMP

Publications

Order No.	Price	Title
150	$11.00	Care and Feeding of The Commodore PET
151	$ 9.95	8K Microsoft BASIC Reference Manual
152	$ 9.95	Expansion Handbook for 6502 and 6802 (S-44 Card Manual)
153	$14.90	Microcomputer Application Notes
154	$ 6.95	Complex Sound Generation
156	$14.90	Small Business Programs

REDYSOFT

Software and Firmware for Popular Computers

"MONJANA/1 CBM Monitor ROM" — A well documented, powerful new monitor ROM that any Commodore CBM user can plug into one of the free ROM sockets.
Order Number 2001 — $98.00

NEW: Our REDYSOFT Cassette Software

Order No.	Price	Title
3475	$49.00	Assembler for CBM
3999	$34.50	BASIC with I/O for TRS-80
8094	$ 1.10	Blank Cassette (Quantity 1)
8095	$ 7.99	Blank Cassettes (Quantity 10)
8096	$69.00	Blank Cassettes (Quantity 100)

Available direct from us NOW. Soon at local dealers. Dealers please contact us for starter stock.

Payment: COD or prepaid
Check, Visa, Mastercharge
POSTPAID in USA.
California add 6% tax

Cross-Reference for the PET

Jim Butterfield

One of the handy things about the 2040 disk system is that it allows you to read programs - or write them, for that matter - as if they were data files.

The possibilities are endless: you can analyze or cross-reference programs; renumber them; repack them into the minimum number of lines, deleting spaces, comments, etc.; or even create a program-writing program that is tailor-made for a particular job.

This program does cross-referencing of a Basic program. It's written in Basic: that means that it won't run too fast (all those GET statements) but you can read what it's doing fairly easily.

There are two types of cross-references normally needed for a Basic program. One is the variable cross-reference: where do I use B$? The other is a line number cross-reference: when do I go to line 360? This program does either. An example of both types is shown - the program in this case did the cross-references of itself.

Reading a Basic Program as a File.

To read a Basic program, you must open it as a file, using type P for program. Line 170 of the cross-reference program does this.

If you read a zero character from the program (that's CHR$(0), not ASCII zero which has a binary value of 48), the GET command gives you a small problem: it will give you a null string instead of the CHR$(0) you might normally expect. You need to watch for this condition and correct it where necessary: you'll see this type of coding in lines 260, 270 and 300.

The first thing to do when you open the file is to get the first two bytes. These represent the program start address, and should be CHR$(1) and CHR$(4) for a normal Basic program starting at hexadecimal 0401. (See line 180).

Now you're ready to start work on a line of Basic. The first two bytes are the forward chain. If they are both zero (null string) we have reached the end of the Basic program; otherwise, we don't need them for this job. (See line 240).

Continuing on the Basic line: the next pair of bytes represent the line number, coded in binary. We're likely to need this, so we calculate it as L (lines 260 to 280) and also create its spring equivalent. L$. We take an extra moment to right-justify the string by putting spaces at the front so that it will sort into proper numeric order.

From this point on, we are looking at the text of the Basic line until we reach a zero which flags end-of-line. At that time, we go back and grab the next line.

Detailed syntax analysis.

When digging out variables or line numbers, we have several jobs to do. As we look through Basic text, we must find out where the variable or line number starts. For a variable, that's an alphabetic character; for a line number, it's the preceding keyword GOTO, GOSUB, THEN or RUN followed by an ASCII numeric.

Once we've "acquired" the variable or line number, we must pick up its following characters and tack them on. For line numbers, it's strictly numeric digits. For variables, things are more complex. Both alphabetic and numeric digits are allowed, but we should throw away all after the first two, since GRUMP and GROAN are the same variable (GR) in PET Basic. We must also pick up a type identifier - % for integer variables or $ for strings - if present. Finally, we have to spot the left bracket that tells us we have an array variable.

To help us do this rather complex job, we construct a character type table. Each entry in the table represents an ASCII character, and classifies it according to its type. Numeric characters are type 6. If we're looking for variables, alphabetic characters are type 5, identifiers (% and $) are type 7, and the left bracket is type 8.

To help us in scanning the Basic line, we define the end-of-line character as type 0; the quotation mark as type 2; the REM token as type 3; and the DATA token as type 4.

Every time we get a new character from Basic, we get its type from table C as variable C9. If we're looking for a new variable or line number, we see if it matches C - alphabetic for variables, numeric for line numbers. Once we find the new item, we kick C out of range and start searching based on the value of C1. This mechanism means that we can search for a variable starting with an alphabetic, and then allow the variable to continue with alphabetics, numerics or whatever.

To summarize variables in this area: A is the identity of the character we have obtained from the Basic program, and C9 is its type. If we're searching, C is the type we are looking for; otherwise it's kicked out of range, to -1 or 9. C1 tells us we're collecting characters and what types we're allowed to collect. C2 is our variables/line numbers flag; it tells us what we're looking for. M$ is the string we've assembled.

The routine from 480 to 520 scans ahead to skip over strings in quotes and DATA and REM statements.

Collecting the results.

For each line of the Basic program we are analyzing, we collect and sort any items we find, eliminating duplicates. They are staged in array A$ in lines 320 to 370. If they are line numbers, they will be left justified so that the sort will be a little odd - line 100 will come before line 20 since we use a string comparison.

When we're ready to start a new line, we add this table to our main results table, array X$, in lines 200 to 220. To save sorting time, we merge these pre-sorted values into the main table. At this point, our data has the line number stuck on the end; this way, we're handling two values within a single array.

Because the merging of the two tables must start at the top so that we can make room for the new items, the items are handled in reverse alphabetic order. We print this to the screen so that you can watch things working. At Basic speed, this program can take quite a while to run; it's nice to confirm that the computer is doing something during this period.

Final Output.

We finish the job starting at line 530. It's mostly a question of breaking the stuck-together strings apart again and then checking to see if we need to start a new line.

Do your own thing.

The size of array X$ determines how large a program you can handle. The given value of 500 is about right for 16K machines; on 32K you can raise it to 1500 or so.

If you're squeezed for space, change array C to an interger array C%. As you can see from the cross-reference listing, you'll need to change lines 100,140,150,160, and 310 - see how handy the program is?

As mentioned before, run time is slow. A machine language program - or even a Basic program with machine language inserts - would speed things up dramatically.

```
100 DIM A$(15),B$(3),X$(500),C(255)
110 PRINT"CROSS-REF    JIM BUTTERFIELD"
120 Q$=CHR$(34):S$="      ":B$(1)=Q$:
    ¬B$(3)=CHR$(58)
130 INPUT"VARIABLES OR LINES";Z$:C2=5:
    ¬IFASC(Z$)=76THENC2=6
140 FORJ=1TO255:C(J)=4:NEXTJ:FORJ=48TO57:
    ¬C(J)=6:NEXTJ
150 IFC2=5THENFORJ=65TO90:C(J)=5:NEXTJ:
    ¬FORJ=36TO38:C(J)=7:NEXTJ:C(40)=8
160 C(34)=1:C(143)=2:C(131)=3
170 INPUT"PROGRAM NAME";P$:OPEN1,8,3,"0:
    ¬"+P$+",P,R"
180 GET#1,A$,B$:IFASC(B$)<>4THENCLOSE1:
    ¬STOP
190 IFB=0GOTO240
200 PRINTL$;:K=X:FORJ=BTO1STEP-1:PRINT" ";
    ¬A$(J);:X$=A$(J)+L$
210 IFX$(K)>=X$THENX$(K+J)=X$(K):K=K-1:
    ¬GOTO210
220 X$(K+J)=X$:NEXTJ:X=X+B:PRINT:B=0
230 REM: GET NEXT LINE, TEST END
240 GET#1,A$,B$:IFLEN(A$)+LEN(B$)=0GOTO530
250 REM GET LINE NUMBER
260 GET#1,A$:L=LEN(A$):IFL=1THENL=ASC(A$)
270 GET#1,A$:A=LEN(A$):IFA=1THENA=ASC(A$)
280 C=C2:C1=-1:L=A*256+L:L$=STR$(L):
    ¬IFLEN(L$)<6THENL$=LEFT$(S$,6-LEN(L$)
    ¬)+L$
290 REM GET BASIC STUFF
300 GET#1,A$:A=LEN(A$):IFA=1THENA=ASC(A$)
310 C9=C(A):IFC9>C1GOTO380
320 K=0:IFB=0GOTO360
330 FORJ=1TOB:IFA$(J)=M$GOTO370
340 IFA$(J)<M$THENNEXTJ:K=B:GOTO360
350 FORK=BTOJSTEP-1:A$(K+1)=A$(K):NEXTK
360 B=B+1:A$(K+1)=M$
370 C=C2:C1=-1:M$=""
380 IFC2=5GOTO420
390 IFA=137ORA=138ORA=141ORA=167THENC=6:
    ¬GOTO470
400 IFA=44ORA=32GOTO470
410 IFC9<>6THENC=9:GOTO470
420 IFC9=CTHENC=-1:C1=4
430 IFC>6GOTO470
440 IFC<0ANDC9>C1ANDC9>6THENC1=C9:GOTO460
450 IFC2=5THENIFLEN(M$)>2ORC>0GOTO470
460 M$=M$+A$
470 ONC9+1GOTO190,480,480,480:GOTO300
480 B$=B$(C9):C$=""
490 GET#1,A$:IFA$=""GOTO190
500 IFA$=B$GOTO300
510 IFA$=<>Q$GOTO490
520 A$=B$:B$=C$:C$=A$:GOTO490
530 CLOSE1:INPUT"PRINTER";Z$
540 C=3:Z=6:IFASC(Z$)=89THENC=4:Z=12
550 OPEN2,C:PRINT#2:PRINT#2,"CROSS ¬
    ¬REFERENCE - PROGRAM ";P$
560 X$="":FORJ=1TOX:A$=X$(J)
570 FORK=1TOLEN(A$):IFMID$(A$,K,1)<>" ¬
    ¬"THENNEXTK:STOP
580 B$=LEFT$(A$,K-1):C$=MID$(A$,K+1):
    ¬IFX$=B$GOTO600
590 PRINT#2:Y=0:X$=B$:PRINT#2,X$;LEFT$(S$,
    ¬5-LEN(X$));
600 Y=Y+1:IFY<ZGOTO620
610 Y=1:PRINT#2:PRINT#2,S$;
620 PRINT#2,LEFT$(S$,6-LEN(C$));C$;
630 NEXTJ:PRINT#2:CLOSE2
```

©

Review

M-100 Microprinter ($390)

Digiclocks
3016 Oceanview Avenue
Orange, California 92665

David L. Bosteel
Professor of Architecture
University of Washington
Seattle, WA. 98105
Chuck A. London
Systems Engineer
Boeing Computer Services Company
Seattle, WA. 98124

The M-100 Microprinter is designed for connection to the IEEE-488 port of the Commodore PET, uses 4.75 inch aluminized paper (40 characters/line), and prints approximately 70-150 lines/minute. The unit comes fully assembled in a 10 x 7 x 5 inch high cabinet with built-in interface and power supply. A 22 inch cable is included, and the plugs are easily reversible on the cable so that the printer can be located either to right or left of the PET computer. The unit works with either "old" or "new" ROM's.

The M-100 Microprinter uses a Sharp DC-4002 electro-sensitive printing mechanism with a 8 by 8 dot matrix to print *all* the PET characters and graphic symbols, including cursor control symbols and reverse field printing, as they would appear on the PET monitor for program listings or programmed output. See sample below of printed output and commands. A "Graphics" switch sets (shifted) characters to print graphics symbols as shown on the PET keyboard. A "Line Feed" switch allows one to independently advance the paper one or more lines. The paper is friction-fed and the printer platen is gear driven so that line spacing is fixed at 5 lines/inch with a character height of .115 inches (for lower case or graphics symbols). Thus drawings or diagrams do not appear exactly as they would on the PET monitor as there is some space between each line.

The M-100 Microprinter very nicely reproduces the PET keyboard symbols, including descenders for lower case letters. The output has more nearly the size and appearance of typed text than is sometimes the case with other printers. The aluminized paper might be seen as a deterrent for record purposes to some, but, for purposes of reproduction, it can be mounted (one or two columns/page) on backing sheets and a xerox copier will provide clean copy without edges or greyness.

The principal uses of a 40 character/line printer are probably to provide listings for development or debugging of programs too long to remember as the lines scroll from the PET monitor screen, and to provide "hard copy" of program output for later use. The PET operating system has commands that make it very easy and these are summarized below as they apply to the M-100 Microprinter. In most cases monitor and printer output are identical but the "TAB" function and use of commas in PRINT# statements for printer output give different (and sometimes unpredicted) results from those seen on the monitor. The use of the "SPC" function and semi-colon separator for same-line printing may be used instead. The use of the semi-colon, when combining a sequence of operations over several lines of a program for same-line printing, appears to confuse the operating system printing firmware, or printer functioning, or maybe only this programmer. Anyhow, the results are different for monitor than for printer. Also, it is inadvisable to leave "open" files lying around in the system--appears to sometimes cause all sorts of puzzling conditions, including a "stuck" cursor (usually recoverable) and unintended programmed input/output formatting.

It is useful, in any case, to organize printer output separately from that for monitor. The PET monitor has the annoying characteristic of flashing (goes dark, then screen image returns) as each print line is output to printer. Much better practice is to organize hard copy output in a subroutine that first clears the screen of previous messages, then outputs new information to printer, and subsequently prints the same messages/information to monitor screen. The result is a stable image which minimizes the impact of printer output on monitor screen.

A brief note on hardware. The unit is attractively housed, solidly constructed, and appears to use good quality components. We say "appears" as the manufacturer recommends very careful disassembly (necessary to expose logic board and electronic components) so as not to tear the ribbon cable linking the printer base and the printing head. As a number of PET owners may be electronic buffs and could test/replace components, it might be better if the board were more readily accessible. We decided a working printer was preferrable to a more complete inspection and can only note that electronic components are soldered, not socketed. The stated nominal life for the printer mechanism (40 million characters MTBF; 1 million strokes minimum, 3 million usual) is reassuring given present usage of one roll of paper per month (130 feet = 7500 lines or approximately 200,000 characters).

The M-100 Microprinter arrived promptly (4 days) via UPS, was easily connected, and worked immediately. Some two rolls of paper later ($6/roll, Radio Shack) it continues to work. That brief period of six weeks is not a fair test of longevity, or even the warranty, but it is long enough to explore potential capabilities and to learn to use the printer effectively in programming.

The Digiclocks unit was purchased because it was/is the most inexpensive printer available that provides *all* the PET characters and graphic symbols. When the principal uses will be program listings and programmed hard copy, the Digiclocks printer would appear to be an excellent choice. For extensive text processing it is a personal view that only IBM Selectric or Daisywheel type output is satisfactory and those alternatives are beyond present means. While the M-100 Microprinter lacks some of the programmable features of the "intelligent" Commodore printer, the PET operating system makes it easy to obtain formatted printer output for most uses.

```
                    M-100 Microprinter
                    OPEN6,5:CMD6
listing-            LIST
PRINT# -            PRINT#6,'ANYTHING'
                    CLOSE6
program output-     10 OPEN6,5:CMD6
                    20 PRINT#6,'ANYTHING'
                    30 CLOSE6
GRAPHICS (OFF)= lower case(if'SHIFT'ed)
         (ON) = graphics symbol
ABCDEFGHIJKLMNOPQRSTUVWXYZ  /#$%&@?\
abcdefghijklmnopqrstuvwxyz
♠|———┤ ├ └LV┌ ▌_▜ ╱XO♣|♦
1234567890123456789012345678901234567890
```

©

In this issue, Marlene Pratto continues her presentation of the grading program shown in Issue 3's column. This time tape and disk data storage options are added. RCL

The Learning Lab

Marlene Pratto

This grading program originated on a 32K PET without a printer. Its major idea was to maintain the information in the machine in arrays for easy reviewing and recording of the results by hand. Options are now provided for storing and retrieving the partial or full term data on tape or disk (tape refers to cassette tape I/O and disk refers to the CBM 2040 disk drive I/O, while I/O refers to input/output operations).

The user is asked how the data will be entered (statements 2905-2907) and there is a new item in the menu (statements 27620-27640) allowing the data to be saved. Of the numeric data, only the raw student grades are stored. When the data is read from tape or disk the user may choose to add grades if the total number of grades will not exceed the maximum in 2150. The averages, z scores, and frequencies are then recalculated and the menu presented.

As the average are calculated, the user is asked is the average is "okay". If a change is needed and the input came from tape or disk, all of the current scores are printed to refresh the user's memory (statements 13200-13560). If the user wants to review the grades, he/she may ask to change one grade and merely input the same value. Note that a change of input method was made here in order to speed up the program (13250). If you prefer using the return key, do not make this change.

The instructions concerning grade changes have been modified. For each student the user may change all grades or one grade at a time (13600-14000).

Since the scores may come from more than one input source, two variables are needed to keep track of the number of grades from each. A new variable NN is used for the number of grades entered initially and NP is the number of grades added on a subsequent run. In the first case N = NN and in the second case N = NN + NP.

Writing the Data File

Observe statements 36010-36097. RV is a variable which points to where the program should continue if there is an I/O problem and IT refers to disk or tape. The filename and an indicator for tape or the disk drive number are requested.

The form of the OPEN statement is *OPEN parm1, parm2, parm3, parm4* where parm1 is the logical file number, parm 2 is the device number (DV), parm3 is the secondary address (SA), and parm4 is the filename and/or commands. The variables, DV and SA, are assigned differently for tape or disk (36024 & 36028). The disk filename is modified as shown in 36028. The fourth parameter consists of the drive number (either 0 or 1) followed by a : (colon), the name provided by the user and finally indicators for SEQuential file type and WRITE mode of operation.

In addition, for disk I/O, we OPEN the command error channel in order to check for disk errors. We *choose* to use file number 2 and we *must* use Secondary Address 15. Some form of initialization of the disk drive must occur. It is included in the OPEN command by using "I" + H$ as the fourth parameter where H$ is the disk drive number provided by the user. (We are assuming that the diskette has been formatted at this point.)

Tape file writing is indicated by a Secondary Address of 1. Disk file writing is indicated by file type (SEQ) and mode (WRITE) which are part of the fourth parameter in the OPEN statement. In either case, the first parameter is the file number which will later appear after the # in the PRINT# statements.

IT is checked in 36032. If the user has not responded with T or 0 or 1, the program returns to 36022 to elicit the proper resonse.

Information is written to the external medium in the following order: 1) file identifier (32 characters or less of information), 2) number of students, 3) number of grades per student, and 4) a character (D$) to indicate whether identifiers have been provided by the user or if sequence numbers will be written as identifiers.

For each student an identifier or sequence number and the student's grades are written. After the last(Jth) student's data, DONE is written. The user is informed when the file writing is completed and is returned to the menu.

Reading the Data File

The program asks how data is to be read (2905-2907). If the data comes from tape or disk, the filename is entered (2925), and the DeVice number and Secondary Address are set (2930-2935).

Tape file reading is indicated by a Secondary Address of 0. For disk, the fourth parameter is similar to that for writing, except that the mode is replaced by READ. The error channel is opened and the disk drive is initialized (this may occur twice in the program since the read and write portions are independent of each other and the user may choose any of several combinations of I/O options.) File 1 is opened for either disk or tape reading. Student data are read in 10250 and 11560.

Data as Character Strings

The data are written as character strings. On input the data are read as character strings and converted to numeric data as needed. Microsoft BASIC provides two functions for converting data from numeric to character and character to numeric. STR$(numeric variable) will convert a number to a string, while VAL(string variable) will convert a string to a number.

If the string variable does not consist of numerals, VAL returns a value of 0.

ERROR TESTING

Disk errors are checked in the subroutine at 36610 and tape errors in the subroutine at 36710. It is not possible to make the program super smart concerning error recovery, but an attempt is made rather than simply stopping the program.

Please make the following correction to the listing. 36095 CLOSE 1: IF DV = 8 THEN CLOSE 2

REFERENCES

Commodore Business Machines, **CBM Users Manual**, 1979

Commodore Business Machines, **CBM Floppy Disk User Manual**, 1979

Total Information Services, **TIS Workbook 4**, 1978


```
1450 REM K$   FILE IDENTIFIER
1470 REM E$ ERROR CHANNEL MESSAGES
1500 REM J NUMBER OF STUDENTS
1600 REM NN NUMBER OF GRADES PER
1700 REM STUDENT INITIALLY, NP GRADES
1710 REM ADDED,N=NN+NP
1720 REM MN=MAXIMUM # OF STUDENTS
1730 REM MG=MAXIMUM # OF GRADES PER ¬
     ¬STUDENT
1740 REM RV RETURN TO INPUT OR OUTPUT
1750 REM IT=1 FOR DISK,IT=2 FOR TAPE
```

```
2100 DIM N$(50),E$(4),Z(50)
```

```
2100 DIM N$(50),E$(4),Z(50)
2150 MN=50 :MG=10 :RV=1
2200 PRINT"[CLR]"
2300 PRINT"THIS PROGRAM ASSUMES THAT ";
2400 PRINT"YOU DO NOT HAVE MORE THAN ";
2500 PRINT"50 STUDENTS AND 10 ";
2600 PRINT"GRADES PER STUDENT. IF   ";
2700 PRINT"YOU DO, CHANGE THE DIM ";
2800 PRINT"STATEMENTS AND 2150."
2900 PRINT
2905 PRINT"INDICATE ENTRY OF DATA BY ¬
     ¬ɼKȓEYBOARD, ɼTȓAPE, DRIVE ɼ0ȓ "
2907 PRINT" OR DRIVE ɼ1ȓ."
2910 INPUT M$:M$=LEFT$(M$,1)
2920 IF M$="K" THEN DV=0:NP=0:GOTO 3000
2925 INPUT"ENTER FILENAME ";F$
2930 IF M$="T" THEN DV=1:SA=0:IT=2:
     ¬GOTO2950
2935 IF M$="0" OR M$="1" THEN DV=8:SA=5:
     ¬IT=1: F$=M$+":"+F$+",SEQ,READ"
2937 IF IT=1 THEN OPEN 2,DV,15,"I"+M$:
     ¬GOSUB 36610:GOTO 2950
2940 PRINT"ENTER K,T,0, OR 1.":GOTO 2910
2950 OPEN 1,DV,SA,F$:IF IT=1 THEN GOSUB ¬
     ¬36610
2952 INPUT#1,K$:ON IT GOSUB 36610,36710
2954 PRINT"FILE IDENTIFIER IS ";K$
2956 INPUT#1,H$:ON IT GOSUB 36610,36710
2958 JT=VAL(H$):IF JT=0 THEN PRINT"# OF ¬
     ¬STUDENTS IS 0":GOTO 2905
2960 INPUT#1,H$: ON IT GOSUB 36610,36710
2962 NN=VAL(H$):IF NN=0THEN PRINT"# OF ¬
     ¬GRADES IS 0":GOTO 2905
2964 INPUT#1,D$: ON IT GOSUB 36610,
     ¬36710:D$=LEFT$(D$,1)
2975 PRINT"NUMBER OF STUDENTS IS ";JT
2980 PRINT"NUMBER OF GRADES IS ";NN
2985 IF D$="Y"THEN PRINT"THERE ARE ¬
     ¬IDENTIFIERS."
2990 IF N<MG THEN INPUT"HOW MANY GRADES ¬
     ¬WILL YOU BE ADDING ";NP
2995 IF NN+NP >MG THEN PRINT"TOO MANY":
     ¬GOTO 2990
2999 N=NN+NP
```

```
4550 IF DV>0 THEN 6200
```

```
10200 IF D$="N" AND DV=0THEN 10700
10250 IFDV>0THENINPUT#1,N$(J+1):
      ¬ONITGOSUB36610,36710:PRINTN$(J+1):
      ¬GOTO10500
```

```
11300 IF DV=0 THEN PRINT"ENTER";N;
      ¬"GRADES FOR ";
11350 IF DV>0 THEN PRINT "ENTER";NP;
      ¬" GRADES FOR ";
11400 PRINT"STUDENT ";J
11500 FOR I=1TON
11520 IF DV=0 THEN 11600
11540 IF I>NN THEN 11600
11560 INPUT#1,H$:ON IT GOSUB36610,36710
11580 STUDENT(J,I)=VAL(H$):GOTO11800
11600 PRINT "GRADE #";I;
```

```
13200 PRINT"OKAY?";
13250 GET C$:IF C$="" THEN 13250
13400 PRINT C$
13500 IF C$="Y"THEN10200
13530 IF DV=0 THEN 13600
13540 PRINT"CURRENT GRADES:"
13550 FOR P=1TON:PRINT STUDENT(J,P);
13560 NEXT P:PRINT
13600 PRINT"WHICH GRADE (IF MORE THAN ";
13700 PRINT"ONE GRADE NEEDS CHANGING, ";
13800 PRINT"YOU MAY CHANGE ONE AT ";
13900 PRINT"A TIME OR CHANGE ALL BY ";
14000 PRINT"ENTERING ";
```

```
16699 REM IF OPEN FILES, THEN CLOSE
16700 IF DV=1 THEN CLOSE 1
16710 IF DV=8 THEN CLOSE 1:CLOSE 2
16720 PRINT:PRINT"WHEN YOU ARE READY ";
```

```
23200 IF Q<10 ORC$="N"THEN GOTO 23700
```

```
27620 PRINT CHR$(18);"7";CHR$(146);
27640 PRINT"WRITE FILE"
```

```
36010 REM WRITE FILE OF PERTINENT INFO
36012 REM AND RAW STUDENT GRADES
36020 RV=2:IT=0
36021 INPUT"FILENAME_";F$
36022 INPUT"ɼTȓAPE OR THE DRIVE NUMBER ¬
      ¬FOR DISK ";H$:H$=LEFT$(H$,1)
36024 IF H$="T" THEN IT=2:DV=1:SA=1:
      ¬GOTO36050
36028 IF H$="0" OR H$="1" THEN IT=1:
      ¬DV=8:SA=6:F$=H$+":"+F$+",SEQ,
      ¬WRITE"
36030 IF IT=1 THEN OPEN 2,DV,15,"I"+H$:
      ¬GOSUB 36610
36032 IF IT=0 THEN 36022
36050 OPEN 1,DV,SA,F$ :IF IT=1 THEN ¬
      ¬GOSUB 36610
```

```
36054 INPUT"ENTER AN IDENTIFIER FOR ¬
      ¬THIS SET OF STUDENTS ";H$
36058 PRINT#1,H$:IF IT=1THEN GOSUB 36610
36062 H$=STR$(J):PRINT#1,H$:IF IT=1 ¬
      ¬THEN GOSUB 36610
36066 H$=STR$(N):PRINT#1,H$:IF IT=1 ¬
      ¬THEN GOSUB 36610
36070 PRINT#1,D$:IF IT=1 THEN GOSUB36610
36080 FOR H=1TOJ
36081 IF D$="Y" THEN PRINT#1,N$(H):
      ¬ GOTO 36084
36082 REM ELSE PRINT SEQUENCE #
36083 PRINT#1,STR$(H)
36084 IF IT=1THEN GOSUB 36610
36086 FOR L=1 TO N
36088 H$=STR$(STUDENT(H,L)):PRINT#1,H$
36089 IF IT=1 THEN GOSUB 36610
36090 NEXT L: NEXT H
36092 PRINT#1,"DONE":IF IT=1 THEN GOSUB ¬
      ¬36610
36095 CLOSE 1:IF IT=2 THEN CLOSE 2
36097 PRINT"FILE WRITTEN. RETURN TO ¬
      ¬MENU.": RETURN
36610 REM ERROR CHECKING ROUTINE FOR ¬
      ¬DISK IO
36620 INPUT#2,E$(1),E$(2),E$(3),E$(4):
      ¬IF E$(1)="00" THEN RETURN
36630 REM ELSE THERE IS AN ERROR
36640 FOR P=1TO4: PRINT E$(P): NEXT P
36650 PRINT"CORRECT THE ERROR, IF ¬
      ¬POSSIBLE. TO CONTINUE THE PROGRAM"
36660 PRINT"PRESS ANY KEY.":CLOSE 1:
      ¬CLOSE 2
36670 GET Y$: IF Y$="" THEN 36670
36680 ON RV GOTO 2905, 36020
36690 RETURN
36700 REM ERROR ROUTINE FOR TAPE
36710 IF ST=0 THEN RETURN
36720 IF ST<64 THEN PRINT"STATUS IS ";
      ¬ST;". TRY AGAIN.":GOTO 37740
36730 IF ST=64 THEN PRINT"EOF ON TAPE."
36740 PRINT"PRESS ANY KEY TO CONTINUE."
36750 GET H$:IF H$=""THEN 36750
36760 RETURN
```

©

STRUCTURED GAMING: Play and Work in High School Computer Science

J. M. Moshell, G. W. Amann (The University of Tennessee)
W. E. Baird (West High School) Knoxville, Tennessee

Prologue:

Question 1: When is a computer game not a game? When is it "ok" class-activity?

Question 2: So what's wrong with games, anyway?

Exasperated answer to 2:

Students won't work on programs when they have access to games. Games are fun and programs are work.

Reflective answer to 2:

The "usual" computer games are either hand-eye (with occasionally some small amount of brain) - coordination contests, such as "Lunar Lander", or fantasy-land interactive do-it-yourself storybooks such as "Dungeons and Dragons". These activities partake of the allure of broadcast television: namely, they involve the student kinetically and emotionally, but they do not have a "cumulative" component. You can walk in on television, (or computer games like PONG or SPACE WAR) anytime; there are no prerequisites, no logical-deductive skills are required. (For an excellent exploration of this theme, see (Postman, 1979).)

We cannot call this kind of attention "passive" - observe any kid watching an action TV show, or playing a video game. Nevertheless the interaction is "non-analytical" - it has more in common with baseball than with reading; more of recess to it than of curriculum. No wonder teachers of computing have "game trouble" whenever interactive terminals or mini-micros become available.

The problem this paper explores is the development of an introductory computing curriculum built around a kind of "structured gaming". The computing community has begun to understand that carefully-chosen programming language features can guide our thought in ways that make code "work across time": remain adaptable, comprehensible, repairable. We propose that a similar choice of **gaming**-features can foster the development of logical problem-solving skills, while retaining the kinetic/esthetic motivational structure of video games. (We have all known "programming hacks" who have made the game/program connection.) We want to use microcomputer color graphics to make computing more like color-crayons, and less like arithmetic.

Having said that much, we will answer Question 1, and then flesh out our answer with a description of the curriculum we are developing, based on our answer.

Question 1: When is a "computer game" an "ok" class activity?

Answer:

1) The game must be designed with a set of concepts and skills in mind, and a plan for how the games teaches them;
2) The things learned in the activity must contribute toward a cumulative body of knowledge, a "toolkit" that the student can perceive and make use of, as toys and toy-making tools; and
3) The game must be superceded by a more interesting, more interactive game, chosen with extreme care to be unplayable unless the student has mastered the skills taught in the previous lesson/game.

Challenge:

Our basic mission, in the University of Tennessee/NSF High School Computer Science Curriculum Project (HSCS), is to make computer skills available to "average" students. Computers may indeed become as ubiquitous as telephones and televisions, and we consider that the introduction of another technology as soporific and captivating and, well, anti-thinking as television could be a major social disaster. We hope, rather, that computers will become "convivial tools" like the telephone; "convivial" means that their use is determined by the user, not by some central least-common-denominator such as a broadcaster. We haven't really got utopian ideas as to what future generations will do with computers (who could have predicted in 1915 what we'd do with automobiles?). We do, however, have a strong feeling that the question of whether individuals will be able to program their computers, or merely buy programs, is an open and important question. The challenge is this, then: to give every citizen who can dial the telephone, some ability to program a computer.

Method:

This section will be brief; we have published elsewhere (Aiken, Hughes, Moshell, 1980) the "nuts and bolts" description of HSCS. We are using a cartoon-animation software system called RASCAL (Moshell & Hughes, 1980), which runs as part of UCSD PASCAL on the APPLE microcomputer. The basic installation costs about $3200, including a single floppy disk, color television and 100 character-per-second printer. Each lesson in a one-semester (eighteen week) course consists of approximately a week of work, divided into these parts:

Introductory activities;
Exploration project;
Skill-building projects:
Buttoning-up activities;

A class consists of about fifteen students per computer (our collaborating schools have only one APPLE each; we hope to try the curriculum in multi-computer classes later.) Five groups of three students alternate computer use with planning work using graph paper and marker pens. The off-line students are planning their strategies, doing hand-simulations, and observing the online students; for to graduate to the next activity, a group must successfully predict the outcome of an assigned "seed" (geometric pattern, algorithm, program, etc.) The activities develop during eighteen weeks, from a non-linguistic color-pattern process called "quilting" through immediate-mode and straight-line-code entry of TURTLEGRAPHICS (Papert, 1970) commands, the introduction of PASCAL control structures such as REPEAT. . .UNTIL and IF. . .THEN, to the creation of cartoon characters and their animation with complex programs using the RASCAL animation system. The output is always color graphics and music; the curriculum steadily increases its "interactivity" as students learn how to use joystick input to control various types of motion. There is always an underlying lesson about how programs work. All code is in a completely structured language (PASCAL), and is taught "from the inside out" - only at later stages do "environmental details" such as declarations become of concern. A PASCAL interpreter is used which scrolls the source-program being executed on the bottom of the screen (at a controllable rate) while the program produces its output on the top part of the screen. A working system will be on exhibit at NECC/80.

The cognitive style of the HSCS Curriculum

The central common fact that unifies our program/lessons Quilting, Turtlegraphics and more traditional interactive games is that they sell themselves. No one has to compel students to do their assignments.

The point at which our curriculum diverges from "closed" games is in the fact that the only real opponent in traditional games is a pseudo-random number generator, or perhaps another human. In a cognitive game the opponent is the rich structure of our own ignorance. . . The excitement of being able to create pattern and order is as old as the wall-paintings in the caves of France. It is an essentially human activity, one at which all players can win. It is also a 'meta-game', in which an infinite number of specific games and traditional games is analogous to that between a set of blocks and a preassembled toy. A different order of learning becomes possible.

The design style of the HSCS Curriculum

The fundamental design principle we have followed is to attempt to make each lesson augment the student's skills in three areas: discovery, control, and design. We allow students to "play" with the system as each new feature is introduced, but they have "discovery questions" whose answers they seek as they "mess around", in more or less structured ways. They need to find the answers to be allowed access to the next level of the system. Students develop discovery skills by experimentally answering questions like "what does this command do?"

We ask students to undertake a specific "challenge", such as the "shoot-the-dot" game, to develop their ability to control the computer by selecting the correct command and providing correct values for its operands. Their understanding of the system is built by simulation exercises, which allow them to predict the behavior of a command, thus to choose the right command.

Later in the semester, students will begin writing programs; but even at early stages there is the impetus to design input sequences to produce the desired pattern. Students must be able to produce a sequence of commands which produces the predicted output on first submission, to graduate to the next level of the system.

Another principle we have followed can be summed up in the phrase "design from the first experience". We believe that computer science (or anything else) should be taught "from the inside out". That is, first experiences must incorporate the heart of the matter at hand, with as little extraneous matter as possible. For instance, quilting teaches the fundamental core of the computing experience: in repetition of a controlled process, there is great power. The Quilting lesson is taught without introducing a word of jargon, previous assignments, or complex command sequences. Quilting, and its fundamental message, can be taught to illiterates. The second lesson similarly teaches the relationship between operands, operators and results. Only after students have firm operational skill with a given tool, do we introduce terminology, written reference materials and the ultimately necessary environmental details such as data declarations and control statements. We are excited by the prospect of transforming gaming, a traditional problem area for computing teachers, into one of their primary tools.

The authors acknowledge and appreciate the assistance of their collaborators: R. M. Aiken, C. E. Hughes, C. R. Gregory and J. A. Ross (University of Tennessee); L. Demarotta (H. C. Maynard High School); E. Miner (Alcoa High School).

References

Aiken, R. M.; Hughes, C. E.; and Moshell, J. M., "Computer Science Curriculum For High School Students", Proceedings ACM/SIGCSE Conference, Kansas City, Montana, February 25, 1980

Moshell, J. M. and Hughes, C. E. "RASCAL: Microcomputer cartoon animation with PASCAL", Proceedings ACM/SIGGRAPH Conference, Seattle, Washington, August 1980

Papert, S., "Teaching Children Thinking", Proceedings IFIP World Congress on Computers and Education, Amsterdam, 1970.

Postman, Neil, "The First Curriculum: Comparing School and Television", Phi Delta Kappan, 61:3, November 1979.

This work is partially supported by NSF Grant SED-79-18991 ©

A PET "ANSWER BOX" PROGRAM FOR TEACHERS

THORWALD ESBENSEN
Coordinator of Elementary Curriculum and Instruction, Edina Public Schools
Edina, Minnesota 55435

Lack of appropriate software has been a persistent problem for educators interested in using microcomputers to assist classroom instruction. This difficulty is one that is not likely to go away soon.

A great many companies are currently rushing in to fill the gap. But it will probably take several years before supply catches up with demand in this burgeoning new area of educational technology. Meanwhile, those of us who are promoting the use of micros for instruction can help teachers integrate their regular classroom materials with computer hardware in ways that will be immediately useful to students and teachers alike. This article briefly describes one such approach.

The public schools of Edina, Minnesota, make considerable use of microcomputers for instruction, particularly at the elementary level. Indeed, demand for these machines has been so great that by the fall of 1980 more than eighty computers (most of them PET micros) will be helping students improve their skills in the areas of capitalization, punctuation, spelling, language usage, and mathematics.

To assist in the accomplishment of this, the Edina system regularly offers its teachers opportunities to become participants in microcomputer workshops designed to integrate the use of printed instructional materials with micros. In particular, the author, who serves as an instructor at these workshops, has written a simple program called ANSWER BOX which has turned out to be both popular and effective.

ANSWER BOX assumes that students will bring printed instructional materials to the micro and type in their answers to problems at the computer keyboard. The computer then becomes a sophisticated response device, providing immediate reinforcement, keeping track of student progress, and summarizing learning results at the end of each lesson, including time on task.

In order to prepare ANSWER BOX programs, teachers need to have only the ability to type answers on data lines. No questions of copyright are involved. Nothing in the original printed material is copied or altered in any way. The answers to problems are simply entered into the computer on the appropriate data lines. The computer does the rest.

Let's see how this works. Look at Figures 1 and 2 below.

Figure 1 is a sample instructional exercise. It consists of fifteen problems. Each problem presents a group of words. Is the group of words a complete sentence? If so, the student types **YES** (and presses the **RETURN** key). If not, the student types **NO** (and presses the **RETURN** key). If the student's answer is right, a smiling ANSWER BOX face appears while the student's name flashes on and off underneath it. If the student's answer is wrong, a frowning ANSWER BOX face appears with the word WRONG. Meanwhile, underneath the ANSWER BOX, the right answer flashes on and off. The student must then type the right answer before the computer will go to the next problem.

If the student does not know the answer to a problem, he/she may type the question mark (and press the RETURN key). The computer will then print the right answer. However, the student must also type the right answer before the computer will go to the next problem.

IDENTIFYING COMPLETE SENTENCES

Lesson 23

For each group of words, answer YES if the group forms a complete sentence. Answer NO if the group does not form a complete sentence.

1. Beyond the highest mountain that we could see
2. If there is the slightest doubt in your mind when you think about it
3. I will see him tomorrow
4. Last night under a yellow moon
5. She has seen enough trouble already
6. Although I did not know them well
7. In case you are ever interested in the results
8. Plan to go with me to the store
9. That sounds sensible
10. There is no point in talking about it
11. Overhead in the bright sunshine
12. When the last car raced across the finish line
13. After he received the mysterious letter in the mail
14. Don't tell me what to do
15. After she ate, she felt better

FIGURE 1

Figure 2 is a listing of the ANSWER BOX program. Let's examine it. The REM statements on Lines 10-60 explain how the answers to a printed exercise are to be entered on the appropriate data lines. Note that Lines 7000-7998 are to be used to enter the data.

It should be emphasized that a single ANSWER BOX program can be used to accommodate a multiplicity of printed exercises. Indeed, the author has more than once entered into one program the answers to all the exercises in a given workbook. As the REM statements indicate, the various groups of exercise answers must be separated by lesson numbers also entered on the appropriate data lines.

If several lessons are to be entered on the data lines, you can occasionally check to see how many free bytes of memory your PET still has for you to use. Typing the direct command ?FRE(0) will give you this information. If you have an 8K PET, ANSWER BOX will allow you to enter about 3000 bytes of data before running out of memory. However, don't push your luck down to the last few bytes. Some bytes are consumed when a program runs. Allow for this.

Lines 100-104 are used to print the name of a particular ANSWER BOX program. In the present instance, we are calling it IDENTIFYING COMPLETE SENTENCES.

Look at Line 7000. The first data entry is 23, which is the number of the lesson (see Figure 1). The next fifteen entries are the right answers for this lesson. Finally, the arrow pointing up tells the PET that the lesson is over and that the results should be summarized. Lines 9000-9999 summarize the results of the lesson.

Readers may fee free to copy the listing of ANSWER BOX and to use it as they think best. Readers who would rather just buy a copy of the tape itself may do so for $7.95 from *MICRO-ED, INC., P.O. Box 24156, Minneapolis, Minnesota, 55424.*

The author would be glad to respond to questions and comments from interested readers.

```
5 REM BY T. ESBENSEN FOR MICRO-ED,
     ¬ INC.P.O. BOX 24156, MINNEAPOLIS,
     ¬ MN. 55424
10 REM USE LINES 7000-7998 FOR DATA.
20 REM PAGE OR LESSON NUMBER SHOULD BE ¬
     ¬FIRST DATA ENTRY.
30 REM THEN ENTER ANSWERS.  ALL ENTRIES ¬
     ¬ SHOULD BE SEPARATED BY COMMAS.
40 REM END EACH LESSON WITH ^
50 REM EXAMPLES: 7000 DATA 1,A,B,C,D,^
60 REM            7010 DATA 2,WAS,IS,^
70 DIMWR$(100)
95 TI$="000000"
99 PRINT"ĥ↓↓↓↓"
100 PRINT"ɾ>>*****************************
     ¬*******
101 PRINT"ɾ>>*                            ¬
     ¬     *
102 PRINT"ɾ>>* IDENTIFYING COMPLETE ¬
     ¬SENTENCES *
103 PRINT"ɾ>>*                            ¬
     ¬     *
104 PRINT"ɾ>>*****************************
     ¬*******
115 PRINT"↓":INPUT"STUDENT'S FIRST ¬
     ¬NAME>>?<<<";NA$
120 IFNA$="?"THEN90
130 PRINT"↓":INPUT"LESSON NUMBER>>?<<<";
     ¬LE$
140 PRINT"ĥ":RESTORE
150 READD$
155 PRINT"h↓↓↓↓↓↓↓↓>>>>>>>>>>>>>>>ɾSEARC
     ¬HING"
160 IFD$="^"THENPRINT"↓SORRY.  LESSON ¬
     ¬NUMBER NOT FOUND.":PRINT:PRINT"TRY
     ¬ AGAIN.":GOTO130
170 IFD$=LE$THEN190
180 GOTO150
190 PRINT"ĥ"
200 READA$
210 IFA$="^"THEN9000
220 P=P+1
225 IFP<10THENSP=2
227 IFP>9THENSP=1
300 PRINT"ĥ"
301 PRINT"          9999999999999999999
302 PRINT"          5                 6
303 PRINT"          5    ANSWER BOX   6
304 PRINT"          5999999999999999996
305 PRINT"  999995                    6
306 PRINT"  5    5                    6
307 PRINT"  5    5
308 PRINT"  5    5                    6
309 PRINT"  888885                    6
310 PRINT"       5                    6
311 PRINT"       5                    6
312 PRINT"       5                    6
313 PRINT"       8888888888888888888
320 PRINT"                  O#####P
321 PRINT"          ERRORS  $     '
322 PRINT"                  L$$$$$:
326 PRINT"                  O##PO##P
327 PRINT"    TIME SPENT    $  '$  '
328 PRINT"                  L$$:L$$:
329 PRINT"                  MIN. SEC.
360 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓>>>>>>>>>>>>>
     ¬>>";E
370 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓>>>>>>>>>>>
     ¬>>>>";
375 PRINTMID$(TI$,3,2);">>";MID$(TI$,5,
     ¬2)
380 PRINT"h↓↓↓↓↓↓↓>>";SPC(SP);P;
382 INPUT">>>?<<<";R$
384 IFR$="?"THEN300
385 IFR$="?"THEN800
400 IFR$=A$THEN700
410 IFR$<>A$THEN800
700 PRINT"ĥ"
701 PRINT"          9999999999999999999
702 PRINT"          5                 6
703 PRINT"          5)~)~)~)~)~)~)~)~)~6
704 PRINT"          5999999999999999996
705 PRINT"         95                 69
706 PRINT"         55    OP      OP   66
707 PRINT"         55    L:      L:   66
708 PRINT"         55        :.       66
709 PRINT"         85    M        N   68
710 PRINT"          5     M$$$$$$N    6
711 PRINT"          5                 6
712 PRINT"          5                 6
713 PRINT"          8888888888888888888
720 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓>>>>>>>>>>>>>>
     ¬>>>";
730 FORZ=1TOLEN(NA$)/2
740 PRINT"<";
745 NEXTZ
750 PRINT"ɾ";NA$;
752 KK=KK+1
755 FORZ=1TO200:NEXTZ
760 FORZ=1TOLEN(NA$)
```

```
765 PRINT"←";
770 NEXTZ
780 PRINT"f̂";NA$;
782 FORZ=1TOLEN(NA$):PRINT"←";:NEXTZ
783 FORZ=1TO200:NEXTZ
784 IFKK=5THENKK=0:GOTO190
786 GOTO750
800 PRINT"ĥ":E=E+1:IFR$="?"THENE=E-1:
    ¬A=A+1:GOTO820
801 PRINT"        99999999999999999999
802 PRINT"        5                  6
803 PRINT"        5NNNNNNNNNNMMMMMMMM6
804 PRINT"        599999999999999999996
805 PRINT"       95       NNNNN       69
806 PRINT"       55    NM        NM   66
807 PRINT"       55    M@   3+   @N   66
808 PRINT"       55         11        66
809 PRINT"       85       WRONG!      68
810 PRINT"        5   N  $$$$$$  M    6
811 PRINT"        5   %  N??????M  '  6
812 PRINT"        5      ########     6
813 PRINT"        88888888888888888888
820 PRINT"h↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓→→→→→→→→→→→→→→
    ¬→→→";
830 FORZ=1TOLEN(A$)/2
840 PRINT"←";
845 NEXTZ
850 PRINT"r";A$;
852 KK=KK+1
855 FORZ=1TO200:NEXTZ
860 FORZ=1TOLEN(A$)
865 PRINT"←";
870 NEXTZ
880 PRINT"f̂";A$;
882 FORZ=1TOLEN(A$):PRINT"←";:NEXTZ
883 FORZ=1TO200:NEXTZ
884 GOSUB900
885 IFKK=5THENKK=0:GOTO300
886 GOTO850
900 WR$(P)=STR$(P)
910 RETURN
7000 DATA 23,NO,NO,YES,NO,YES,NO,NO,YES,
     ¬YES,YES,NO,NO,NO,YES,YES,^
7999 DATA^^
9000 PRINT"ĥ↓SUMMARY OF RESULTS FOR ¬
     ¬LESSON ";LE$
9010 PRINT"88888888888888888888888888888
     ¬888"
9015 PRINT"↓STUDENT'S FIRST NAME: ";NA$
9020 PRINT"↓NUMBER OF PROBLEMS:";P
9030 PRINT"↓NUMBER OF ERRORS:";E
9040 PRINT"↓COMPUTER GAVE ANSWER:";A
9050 PRINT"↓TIME USED: ";MID$(TI$,3,2);
     ¬" MINUTE(S) AND ";MID$(TI$,5,2);
     ¬" SECOND(S)"
9060 PRINT"↓SPECIFIC PROBLEMS MISSED OR"
9062 PRINT"###########################"
9065 PRINT"ANSWERED BY THE COMPUTER"
9070 PRINT"########################"
9090 FORZ=1TOP
9100 PRINT"r";WR$(Z);"→";
9105 IFWR$(Z)=""THENPRINT"←";:C=C-1
9110 C=C+1:IFC=10THENC=0:PRINT"↓"
9120 NEXTZ
9125 PRINT
9130 PRINT"9999999999999999999999999999
     ¬99999999"
9999 END
```

©

ATARI And SRA Announce Educational Agreement

Science Research Associates, Inc. (SRA) and Atari Inc. have announced the signing of a multi-faceted agreement concerning the development of educational computer courseware and sale of Atari Personal Computer Systems to the educational community. Under the agreement, SRA will develop educational computer courseware programs in the areas of reading, language arts, mathematics, science and social studies for use in the home on Atari personal computers. Atari will have the right to market SRA-developed courseware programs intended for home use.

SRA will also produce and market to schools a broad range of educational computer programs designed for use in the classroom with personal computers.

Atari has appointed SRA as the organization with primary responsibility for the sales of Atari's personal computer products and services to the educational community, which includes all types of educational institutions, public and private from pre-school through university level.

For more information, contact Atari® Consumer division, 1265 Borregas Avenue, Sunnyvale, CA 94086©

PET GET WITH FLASHING CURSOR

Gary Greenberg
35-63 80th St.
Jackson Hts., NY 11372

There are many reasons why some programmers prefer to use the GET statement for keyboard input rather than the INPUT statement. One reason of interest to many beginners is that the GET statement can help prevent falling out of the program when the wrong response is made. For example, pressing RETURN as a response to INPUT causes you to fall out of the program. Similarly if you call for a numeric variable and type an alphabetic character, again you are in trouble.

The GET statement works by checking the keyboard input buffer for the next character. However, without the benefit of appropriate software routines, the GET statement collects only one character, does not provide a cursor for a prompt, does not print the character on the screen and does not permit you to erase the result before entering the data. Additionally, the statement is executed so rapidly you will probably not be able to enter it in time to get it accepted. You do not use a carriage return to signal the end of the entry.

A number of magazines have provided software pieces to enable you to use the GET but they do not permit you to delete and correct the entry before entering it into the program. The following program permits you to use the GET statement instead of the INPUT statement. It will permit you to have a flashing cursor and it will permit you to delete and correct entries before committing them to the computer. You use the carriage return to signal the end of your entry. You will not fall out of your program if you respond with a carriage return.

Program Listing

```
100 PRINT"WHAT IS ANSWER?";:GOSUB 1000
110 PRINTZ1$:Z1 = VAL(Z1$):PRINTZ1
1000 REM GET WITH FLASHING CURSOR
1010 Z$ = "":Z1$ = ""
1020 PRINT"R rcl";:FORI = 1 TO 35:NEXTI
1030 PRINT"cl";:FORI = 1 TO 35:NEXTI
1040 GETZ$:IFZ$ = ""THEN1020
1050 REM ERASE ENTRY CHECK
1060 IFZ$ <> CHR$(20)THEN 1110
1070 IFZ$ = ""THEN 1020
1080 ZZ = LEN(Z1$):IF ZZ < 1 THEN 1020
1090 Z1$ = LEFT$(Z1$,ZZ-1):PRINT"cl";
1100 GOTO 1020
1110 IFZ$ = CHR$(13) OR Z$ = CHR$(141) THEN 1150
1120 PRINT Z$;
1130 Z1$ = Z1$ + Z$
1140 GOTO 1020
1150 GETZ$:IF Z$ = ""THEN 1150
1160 RETURN
```

The program is a subroutine that can be called in place of any INPUT statements. Here's how it works.

LINE 100: The print message is used as a prompt. GET does not provide a question mark and you will have to put it into the PRINT statement if you want it. After the prompt is printed the program jumps to the subroutine.

LINE 110: This line is the line after the subroutine is executed. You can use any program statements you want. The one here is just an illustration. The PRINT Z1$ is redundant since it is printed out in the subroutine. The VAL statement is used to convert the STRING to a NUMERIC. You should note that the subroutine leaves the cursor immediately after the entry. If you want a linefeed you will have to insert a "PRINT:" in line 110.

LINE 1000: The subroutine starts here.

LINE 1010: Z$ will be used to collect the data typed in. Z1$ will be used to build a string out of the data entered. This line initializes both strings as Nulls (empty).

LINES 1020-1030: These lines simulate the cursor. The two FOR-NEXT routines are time delays. To speed up the flashing cursor, shorten the loops. To slow down the cursor, increase the number of loops. The PRINT statements produce a REVERSE FIELD, SPACE (becomes a solid block in Reverse field), REVERSE OFF, CURSOR LEFT. This is followed by the delay and then PRINT SPACE and CURSOR LEFT again followed by a delay. By using different characters instead of the space character you can create some interesting cursors. This produces an interesting wig-wag effect.

LINE 1040: The GET checks the keyboard buffer. If it is empty (null string) it starts the cursor flashing again while it waits for data. Once a character has been entered the program moves on to the next line where it begins to examine the nature of the character entered.

LINE 1050-1060: These lines check for a "delete".

LINE 1070: If "delete" is entered and there is no data entered the program returns to the flashing cursor.

LINE 1080: If data has been entered this line checks to see how long the data is. This check is to make sure that the "delete" does not go past the beginning entry point on the screen.

LINE 1090: The last character in the string is deleted. You can continue to delete characters until you have erased the whole string, and the program keeps checking for the delete every time you make an entry.

LINE 1100: You return to the flashing cursor waiting for the next entry.

LINES 1110-1140: These lines first check to see if you have hit "carriage return" or shifted "carriage return". If not, then the program adds Z$ to Z1$ to keep building the string entry. After each character is added to Z1$ the program returns to the flashing cursor. When the carriage return is pressed it is the signal that the entry is complete and you branch to 1150.

LINE 1150: This line is used to empty the buffer. Sometimes, when you enter data rapidly, there is information retained in the buffer that you haven't accessed. If you don't clear it out then the next time you call the routine, the old unused data will be entered in response to the GET command before you have a chance to use your own input.

LINE 1160: This returns control to the main body of the program at line 110. ©

Two "Mini" Reviews:
Robert Lock, Editor

ATARI BASIC CARDS 24 pages, $4.95
Micronotions
1929 Northport Drive #6
Madison, Wisconsin 53704

Len Lindsay's Atari Basic Cards are simply that. . .a comprehensive heavy-paper booklet covering the commands and syntax of Atari BASIC. The preliminary copy I read was a handy size, 4¼" by 11" with four commands per page. The back of each page is left blank so the user may jot in his or her own helpful hints. The command sections are identical in size and are arranged so you can cut them out, if you prefer, and use a small card file for reference.

Each command is presented with syntax, an example of use and brief definition. Best of all for beginners— and aren't we all at this point— is the "see also" cross reference presented with most of the commands.

Although the booklet isn't typeset, the reproduced printer output is extremely clear and easy to read. At $4.95 (approximately) Atari Basic Cards is a recommended addition to the small (but growing) amount of Atari information available.

A Pre-review Review. . .

Mailing List $99.95
Dr. Daley's Software
425 Grove Ave.,
Berrien Springs, MI 49103

I haven't time this issue to give a real review of Mailing List, but felt it was important enough to comment on. Richard Daley is adopting a style of programming that I think is important. His mailing list program is designed so we, the users, have a great deal to say in structuring our files and defining how we want our data base to look. You can define your own sort keys, pull out subsets, etc. We are currently using it, with some in-house output modifications, to maintain our whole dealer data base on a 32K CBM with attached disk drives and NEC Spinwriter.

I asked Joretta Klepfer to look over my comments and here are hers:

> You might add that it is easy to do editing, updating, maintenance and back-up of data files.
>
> You can store an output format file on disk.
>
> Even more, you can create your own format of the data for printing.

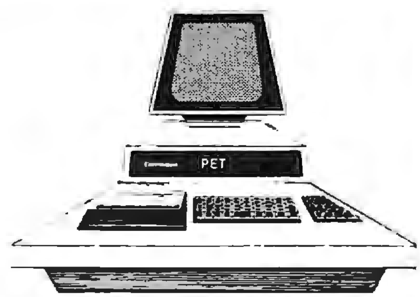
Okay. Thanks, Jo. We'll look forward to your review next time. ©

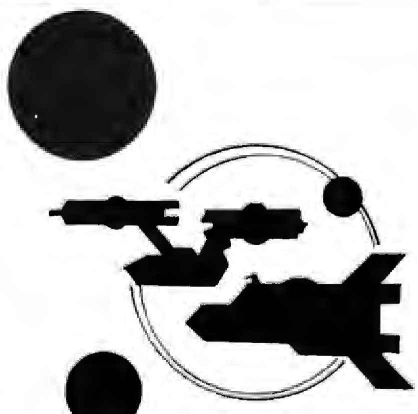
Get a load of this . . .

Instant Software™

Ask for Instant Software at a computer store near you.

PET UTILITY I You're working under a serious handicap if you can't write programs in machine language. The PET Utility I package gives you the tools you need:
•**Monitr**—The Monitr program lets you write, edit, save, and verify any machine language and/or BASIC program. Just load and run the Monitr program and then load the program you want to edit.
•**Programmer's Calculator**—This program will convert numbers into the binary, octal, decimal, and hexadecimal systems and function as a floating point calculator. It will also display all four numbering systems simultaneously and allow you to handle large numbers limited only by the size of your screen.
For the 8K PET. **Order No. 0105P. $9.95.**



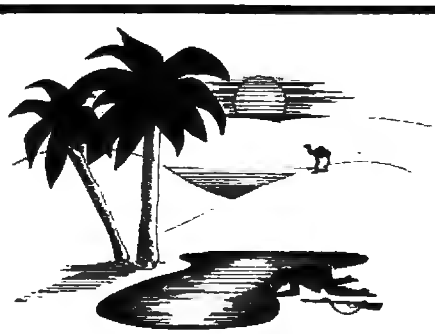


SPACE WARS You must protect your planet against a wide variety of alien attacks. Included are:
•**Space War**—One or two players can pilot their saucers and duel with laser beams or shoot stars.
•**Star Ship Attack**—Your mission is to protect the food station satellites from destruction by the enemy star ship. You must capture, destroy, or drive off the attacking ship.
•**Battlefield**—Guess the location of the four enemy divisions and destroy them before your forces are wiped out.
Engaging in battle requires the Apple 8K and Integer BASIC. **Order No. 0096A $7.95.**

ACCOUNTING ASSISTANT This package will help any businessman solve many of those day-to-day financial problems. Included are:
•**Loan Amortization Schedule**—This program will give you a complete breakdown of any loan or investment.
•**Depreciation Schedule**—You can get a depreciation schedule using any one of the following methods: straight line, sum of years-digits, declining balance, units of production, or machine hours.
This package is available for both the PET and Apple. It requires the Apple 16K and Applesoft II BASIC. **Order No. 0088A $7.95** or the PET 8K. **Order no. 0048P $7.95.**

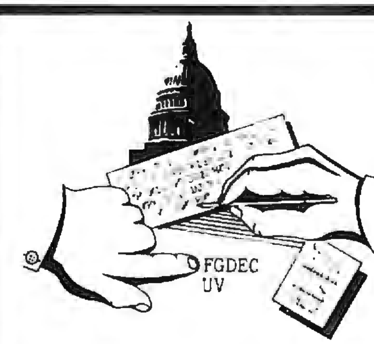
MIMIC Test your memory and reflexes with the five different versions of this game. You must match the sequence and location of signals displayed by your Apple. You'll need an Apple with 24K and Integer BASIC. **Order No. 0025A $7.95.**

HAM PACKAGE I This versatile package lets you solve many of the problems commonly encountered in electronics design. With your 8K PET, you have a choice of:
•**Basic Electronics with Voltage Divider**—Solve problems involving Ohm's Law, voltage dividers, and RC time constants.
•**Dipole and Yagi Antennas**—Design antennas easily, without tedious calculations.
This is the perfect package for any ham or technician. **Order No. 0054P $7.95.**



SAHARA WARRIORS Enjoy all the gritty realism of desert warfare:
•**Commando**—You must send your commandoes to trap a German general and cut him off from his troops.
•**French Foreign Legion**—The battalions of the French Foreign Legion are in a race with the Arabs. Which side will get its battalions into the oasis in the shortest time? You and a friend can find out.
Both programs in this package require an Apple 8K and Integer BASIC. **Order No. 0080A $7.95.**

MORTGAGE WITH PREPAYMENT OPTION/FINANCIER These two programs will more than pay for themselves if you mortgage a home or make investments:
•**Mortgage with Prepayment Option**—Calculate mortgage payment schedules and save money with prepayments.
•**Financier**—Calculate which investment will pay you the most, figure annual depreciation, and compute the cost of borrowing, *easily and quickly*.
All you need to become a financial wizard with a 16K Apple and Integer BASIC. **Order No. 0094A $7.95.**



CODE NAME:CIPHER Enjoy that same feeling of intrigue and discovery with the Code Name: Cipher package. Included are:
•**Memory Game**—Would you like to match your memory against the computer's? You can with the Memory Game.
•**Codemaster**—One player types in a word, phrase, or sentence, and the PET translates that message into a cryptogram. The other player must break the code and solve the cryptogram in the shortest time possible.
•**Deceitful Mindmaster**—This isn't your ordinary Mastermind-type game. You must guess the five letters in the hidden code word.
•**Code Breaker**—Cracking this code won't be as easy as cracking walnuts. You'll need to flex your mental muscles to win this game.
If you want a mental challenge, then Code Name: Cipher is for you. For the 8K PET. **Order No. 0112P. $7.95.**

TO ORDER:

Look for these programs at the dealer nearest you (See the list on the next page). If your store does not stock Instant Software send your order with payment to:

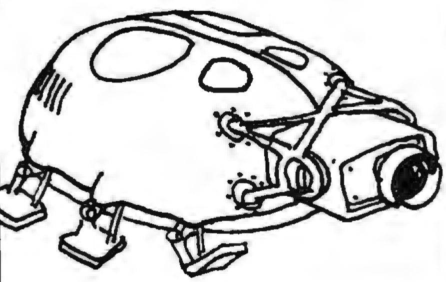
Instant Software Inc.
Order Dept.
Peterborough, N.H. 03458
(Add $1.00 for handling)

or call toll-free 1-800-258-5473 (VISA, Master Charge and American Express accepted).

Ask your local dealer for the latest Instant Software catalog or write: Instant Software Catalog Dept., Peterborough, N.H. 03458

Instant Software™ Inc. Peterborough, New Hampshire 03458 603-924-7296

Have You Been Bitten By The Computer Bug?



Creative Computing Magazine

Creative Computing has long been Number 1 in applications and software for micros, minis, and time-sharing systems for homes, schools and small businesses. Loads of applications every issue: text editing, graphics, communications, artificial intelligence, simulations, data base and file systems, music synthesis, analog control. Complete programs with sample runs. Programming techniques: sort algorithms, file structures, shuffling, etc. Coverage of electronic and video games and other related consumer electronics products, too.

Just getting started? Then turn to our technology tutorials, learning activities, short programs, and problem solving pages. No-nonsense book reviews, too. Even some fiction and foolishness.

Subscriptions: 1 year $15, 3 years $40. Foreign, add $9/year surface postage, $26/year air.

The Best of Creative Computing

The first two years of **Creative Computing** magazine have been edited into two big blockbuster books. **American Vocational Journal** said of Volume 1, "This book is the 'Whole Earth Catalog' of computers." [6A] Volume 2 continues in the same tradition. "Non-technical in approach, its pages are filled with information, articles, games and activities. Fun layout." —**American Libraries.** [6B] Each volume $8.95.

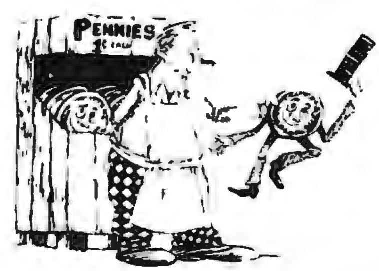
Be A Computer Literate

A first introduction to the computer world for children age 10 to 14. Full color diagrams, drawings, photos and large type make this book easy to read and understand. Written by Marion J. Ball and Sylvia Charp. This informative 62-page book is used in many school systems. Softbound [6H] $3.95.

Basic Computer Games

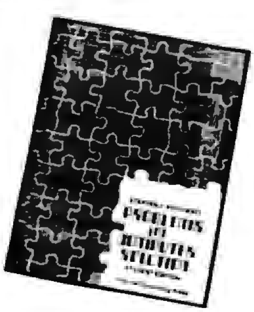
Edited by David Ahl, this book contains 101 imaginative and challenging games for one, two, or more players — Basketball, Craps, Gomoko, Blackjack, Even Wins, Super Star Trek, Bombs Away, Horserace. Simulate lunar landings. Play the stock market. Write poetry. Draw pictures.

All programs are complete with listing in Microsoft Basic, sample run and description. Basic conversion table included. 125,000 copies in print. 192 pages softbound. [6C] $7.50.



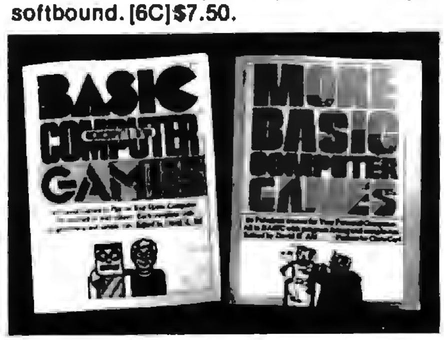
Computer Coin Games

Computer Coin Games by Joe Weisbecker aids newcomers to the field of computers by simplifying the concepts of computer circuitry through games which can be played with a few pennies and full sized playing boards in the book. Enhanced by outrageous cartoons, teachers, students and self-learners of all ages will enjoy this 96 page softbound book. [10R] $3.95.



Problems for Computer Solution

Problems for Computer Solution by Stephen J. Rogowski is an excellent source of exercises in research and problem solving for students and self-learners. Problems like the Faulty Speedometer Spotter make learning fun and easy. 104 pages, softbound, [9Z] $4.95.



More Basic Computer Games

Contains 84 fascinating and entertaining games for solo and group play — evade a man-eating rabbit, crack a safe, tame a wild horse, become a millionaire, race your Ferrari, joust with a knight, trek across the desert on your camel, navigate in deep space.

All games come complete with program listing in Microsoft Basic, sample run and description. 192 pages softbound. [6C2] $7.50.

How To Order

Send order and payment to **Creative Computing**, P.O. Box 789-M, Morristown, NJ 07960. Add $1.00 shipping and handling per order (foreign, $2.50) N.J. residents add 5% sales tax. Visa, MasterCharge and American Express orders welcome. For faster service, call in your bank card order toll free to: **800-631-8112.** (In NJ, call (201) 540-0445.)

creative computing

THE CONSUMER COMPUTER

Len Lindsay
1929 Northport Drive #6
Madison, WI 53704

Summer is almost upon us. Watch for a burst of activity in the software and computer accessories markets. Read the new ads & announcements with care.

A review will help put a perspective on the product, but the best thing is to see it operating before you buy it.That's one good reason to buy from your local computer store.

IRIDIS #1

The first issue of IRIDIS has now been shipped by THE CODE WORKS (Box 550, Goleta, CA 93017). I have a copy and enjoyed it very much. IRIDIS is a set of four programs on tape (or disk) accompanied by a booklet containing the complete listings for the programs as well as a line by line explanation of how the programs work. The best way to learn how to write your own programs is to look at other good programs. IRIDIS is an excellent way to do this, since the programs are well done, and fully explained. The IRIDIS booklet also features short articles explaining various aspects of your ATARI computer.

The first four programs may not be earth shattering, but they show some of the things that can be accomplished with an ATARI computer. The first program, **CLOCK,** illustrates how to use ATARI timer functions. It makes use of sound and fine graphics to create a chiming clock. The time is not shown with digital letters, but with the old fashioned hour, minute, and second hands.

LOGO illustrates how to use ATARI graphic commands to plot, draw and fill. It shows you how fast the color of any area can change due to ATARI's method of using Registers to store color information.

POLYGONS draws some very beautiful and intricate patterns on your screen. You watch it as it draws the pattern, then a few seconds after it is complete, it is erased and a new and different pattern is drawn. This is a good demonstration of high resolution graphics.

ZAP is the only program that allows participation. You use a joystick in this game to control a worm which constantly keeps moving around the screen. Target squares appear randomly. You try to manuveur your worm to hit the targets before they disappear. You get points for each target you "eat", and your worm grows longer as well. If your worm ever bumps into itself in its travels, the round is ended. The game is over after 5 rounds.

That's IRIDIS #1. It is available for $9.95 on cassette or $12.95 on diskette. It is a good way to learn about your computer, and add 4 good programs to your program library.

ATARI BASIC CARDS

Another aid to ATARI users who are hoping to learn how to program in BASIC is ATARI BASIC CARDS. Since I was the author of this booklet, I can't give you an unbiased review, but COMPUTE was given a copy for review. It is available from MicroNotions, (1929 Northport Drive, Room 6, Madison, WI 53704) for $5.

PROGRAMMA Emerges

PROGRAMMA INTERNATIONAL (3400 Wilshire Blvd, Los Angeles, CA 90010) is one of the largest personal computer software houses. They strive for quality programs and customer satisfaction. They now are supposed to have ATARI programs ready for release. Plus some very exciting programs for the PET/CBM. This includes "TALKING CALCULATOR" for the PET. PET users may be in for some more excitement very soon.

AUTOMATED SIMULATIONS

Automated Simulations (1902 Old Middlefield Way, Suite 15, Mountain View, CA 94043) has some new additions to its excellent line-up of programs for the PET and APPLE. COMPUTE #1 provided a full page review of their DUNJONQUEST program Temple of Apshai. Now they are offering LOWER REACHES OF APSHAI, a data tape (disk) for use with the Temple of Apshai program. Computer Role Playing addicts will rejoice with this extension of a great simulation.

If you aren't familiar with Role Playing, and want to ease into it (instead of taking on the full blown Temple of Apshai) I have reviewed their two MICROQUEST games, Datestones of Ryn and Morlocs Tower. They now are almost finished with a science fiction adventure, Rescue at Rigel. I will keep you posted about these sophisticated simulation games.

The Word Is Out

Dymax is coordinating a series of more than 10 books for PET, ATARI, APPLE and RADIO SHACK. The books are user oriented and should be a good source of information. Each book is written by a TEAM of authors, usually 3 or 4. This provides feedback for each co-author as well as several perspectives on the books content. I am familiar with this because I am a co-author for the **ATARI GAMES AND RECREATIONS** book and lead author for the **PET GAMES AND RECREATIONS** book. Both books should be available this winter, along with some of the other Dymax books. *(Editors Note: Watch COMPUTE for preliminary reviews. RCL)*

ATARI CHESS

The ATARI CHESS cartridge for the ATARI 400 or 800 computers is now available, and I have used one in my system at home. The chess board is a green and gold checkerboard with a maroon border. The pieces are black and white, and make use of ATARI's high resolution graphics for easily identifiable pieces. A blue box or "cursor" is used to show where on the board you are. That makes 6 different colors displayed at once on the screen - quite amazing, considering that there are only 5 color registers. (NOTE: exact colors vary from TV to TV).

You use a joystick to move your pieces. This may sound weird, but it is very easy, and reliable. (Just sit back in a comfortable chair with the joystick, and challenge the ATARI.) I have played the other chess games, and communicating grid locations to the computer is a chore. The ATARI method is so simple, even a young child can play. And there are 8 levels of difficulty, to satisfy all ranges of players, even those just learning to play. As the level goes up, the ATARI thinks longer about its move. According to the documentation, the lowest level (level 0) averages about 15 seconds per move. The next level (level 1) averages 30 seconds per move. Level two averages 45 seconds, level 3 - 2½ minutes, level 4 - 3 minutes, and level 5 about 10 minutes. The two remaining levels take quite a bit of thinking time, and would be only for the very serious player. Level 6 takes about 9 hours per move, and level 7 about 10 hours per move. The skill level can be changed at any time during the game if desired. In my games at level 1 it took about 10 seconds per move - at level 2 about 15 seconds - about 3 times faster than stated in the documentation.

Moving a piece is a very simple matter. You simply move the "cursor" to the top of the piece you wish to move. The cursor moves box by box, controlled by a joystick. When the cursor is on top of any piece it blinks, alternating box and piece to allow you to see both the cursor and the piece on the square. The cursor will continue moving from square to square until you hit the red controller button. If the cursor is over one of your pieces, the piece will go into blink mode. You now move the piece by joystick control while the cursor remains in the original location for reference. When the piece is where you want it, simply hit the red controller button again and the move is made, unless it is an illegal move. The ATARI will not allow you to make an illegal move, but beeps to tell you to try again.

ATARI CHESS includes many polishing touches that make playing enjoyable, and less frustrating. If you are put into check, your king begins flipping around, inside its square, softly beeping (you will not be able to miss it). Anytime you move a piece that would leave your king in check the word CHECK appears in the top corner advising you of this situation.

Perhaps the nicest feature of ATARI CHESS is that you can leave to get a sandwich while the ATARI is thinking about its move, and when you come back you will know just what piece it moved. It puts the "cursor" in the piece's original location and blinks the piece in its new location. The cursor and blinking remain on until you move the joystick (then the piece remains in its new location and the cursor follows your control).

You can play the white or black pieces, with white going first. If you wish to set up a particular chess situation, simply hit the "S" key on the keyboard, and place or delete pieces anywhere on the board, again under joystick control. Hit any other key when the set-up is as you like it. This is nice, if you would like to redo a move (after you saw how ATARI clobbered you) - simply put ATARIs piece back, and replace yours. Castling and En Passant capturing are allowed and explained in the documentation.

ATARI CHESS plays a fairly good game in its lower levels. I leave testing the top levels to someone with more time, since it takes so long to make its moves. At the easiest levels it would be a good way to learn how to play chess, for it will not let you make an illegal move, and it doesn't think too long about its own moves.

ATARI CHESS vs SARGON 2

I took my ATARI computer (and ATARI CHESS) over to a friends and challenged his TRS-80 SARGON 2. We played two games. The first game was SARGON 2 at level 0 and ATARI CHESS at level 1. This seemed to be an even match - both took about 6 seconds per move, until the end when ATARI took about 20 seconds each for a few moves. After 17 moves, each side had 13 pieces left. Move 18 was very bad for ATARI - it missed taking SARGON's queen with a pawn (and no threat afterwards either). After 68 moves the winner was ATARI CHESS.

That called for a rematch. SARGON 2 was set to level 1 and ATARI CHESS at level 2. This time the difference in thinking times was noticable. The first 20 moves ATARI CHESS took only about 7 seconds per move, while SARGON took about 26 seconds. The rest of the game, SARGON generally took a bit longer thinking time. The extra thinking time must have paid off, since SARGON 2 won after 60 moves. They traded queens on the eighth move.

This match pointed out some of the cosmetic advantages of ATARI CHESS. Its graphics were far superior, and easy to follow. My friend would use my ATARI CHESS moves as input for his move against SARGON 2. I in turn would take SARGON's move as input to ATARI CHESS. The ease of entering moves with a joystick for ATARI CHESS was quite evident - I was relaxing on the floor 6 feet from the 25 inch color TV with my joystick, while my friend sat next to the TRS-80 continually calculating the pieces starting and ending grid positions. ©

D & R CASSETTE SYSTEM

Robert W. Baker,
15 Windsor Drive,
Atco, NJ 08004

Here's something that should be of interest to anyone considering adding another cassette drive to their PET system. *D&R Creative Systems, PO Box 402B, St. Clair Shores, MI 48080* is now producing a cassette drive with a built in counter and several other features at a selling price that is less than the Commodore C2N cassette drive. The D&R S2545 Cassette System consists of a self contained, epoxy potted interface module and a Sanyo M2545A recorder. The recorder is specially modified to sense when the recorder buttons are pressed and operates identically to the PET cassette drive with full program control. The cassette recorder is still fully warranted by Sanyo, even with the slight modification.

The interface module plugs directly into the cassette port at the rear of any model PET and fits flat against the back panel. Because of the physical size of the interface module, it cannot be used with the cassette interface that is located inside the PET. There are four separate cables from the interface module which connect to the recorder REM, EAR, MIC, and added sense connectors. The sense connector is located inside the unused battery compartment which requires removing the battery compartment cover in order to make the connection.

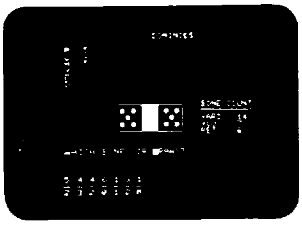
Besides the digital counter for normal program indexing, the D&R Cassette system can locate programs audibly using the Fast Forward Cueing feature of the Sanyo recorder. While the recorder is in the Play mode, the Fast Forward button can be depressed and you can hear the programs pass by the recording head at the Fast Forward Cueing speed. It is very easy to hear the 10 second leader at the start of each program and simply count the programs as they pass by to get to the desired program.

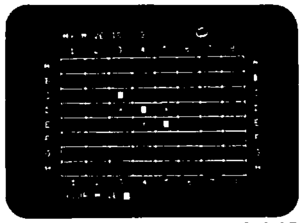
I recently tested a sample D&R Cassette system on an older 8K PET and a new 32K PET. I found no compatibility problems with tapes switched between the D&R Cassette system and three different Commodore C2N drives. D&R Creative systems does, however, warn of possible incompatibilities due to head alignments of the Commodore drives since Commodore has changed their cassette unit several times since the PET was introduced. Another point of interest is that the D&R Cassette system operates with the volume and tone controls of the Sanyo recorder all the way up and does not require any intermediate, special settings.

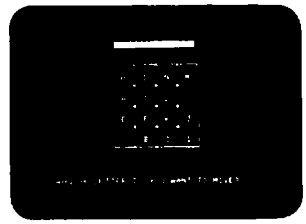
The only disadvantage of the D&R Cassette system is that the MIC and EAR cables cannot both be connected to the recorder at the same time. You must switch the cables to the Sanyo recorder whenever you switch between reading or writing a tape. This slight inconvenience seems a small price to pay when one considering the digital counter and Fast Forward Cueing features provided. Also, the Sanyo recorder can still be used as a standard recorder with a built in microphone and automatic level control for recording. Thus, the recorder could be used to take notes while working on a new program. All in all, the unit is very reliable and should be quite attractive considering the features provided at the current price of $83 plus $3 shipping. ©

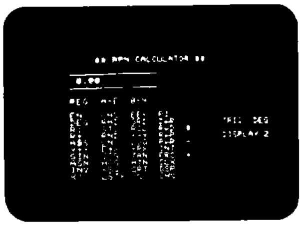
Software for the PET

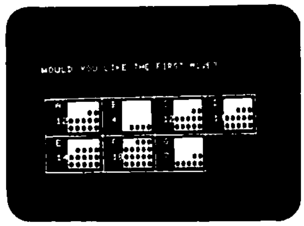
PROGRAMMA Software Program Products

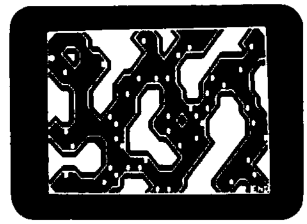

DOMINOES $ 6.95

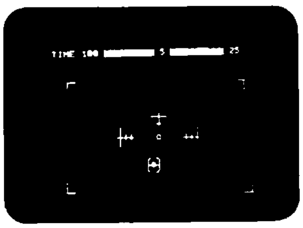

OTHELLO $ 9.95

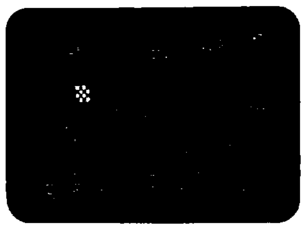

LETTER SQUARES $ 6.95

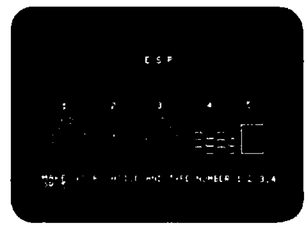

RPN MATHPACK $19.95

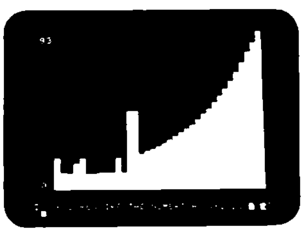

SUPER NIM $ 6.95

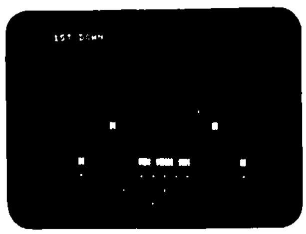

DIR/REF $ 6.95

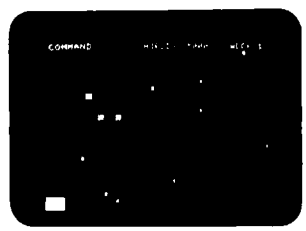

SPACE WARS $ 9.95

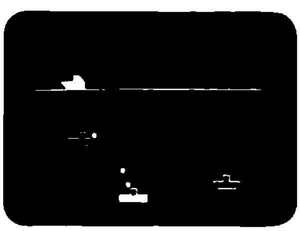

GRAND PRIX $ 6.95

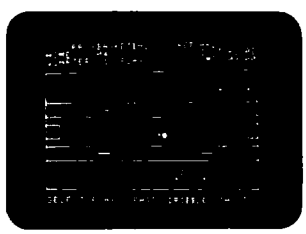

E.S.P. $ 9.95

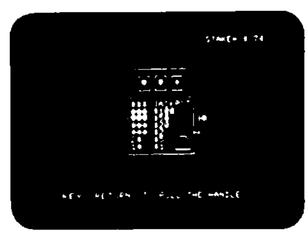

FORECAST $ 9.95

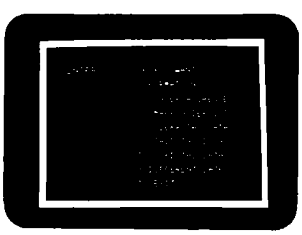

$ 6.95


BLOCKADE $ 9.95


DEPTH CHARGE $ 9.95

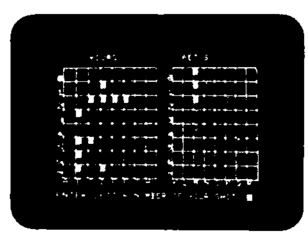

BASKETBALL $ 9.95

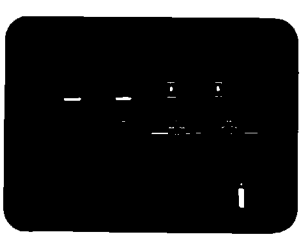

SLOT MACHINE $ 6.95


HOME ACCOUNTING $ 9.95

All orders include 3% postage and handling with a minimum of $1.00. California residents include 6% Sales Tax.

VISA MASTERCHARGE

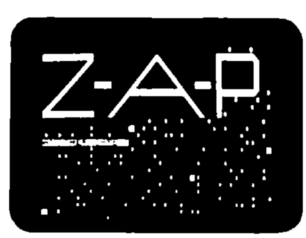

BATTLE SHIP $ 9.95


SHOOTING GALLERY $ 9.95

PET IS A TRADEMARK OF COMMODORE BUSINESS MACHINES, INC.

PROGRAMMA INTERNATIONAL, Inc.
3400 Wilshire Blvd.
Los Angeles, CA 90010
(213) 384-0579
384-1116
384-1117

Dealer Inquiries Invited



ZAP $ 6.95

For your convenience in making use of the following article, I've reprinted a portion of Chuck Johnson's excellent article in Best of the PET Gazette, p.42. RCL

PET ting WITH A JOYSTICK

Harvey B. Herman
Chemistry Department
University of North Carolina
at Greensboro
Greensboro, N.C. 27412

My older style PET keyboard gets banged around quite a bit when my kids play games which use the number pad. A recent PET-Pourri column in Kilobaud (1) prompted me to install a joystick on my PET in order to save my keytops from further wear. This article is intended to share my experiences with this project and to encourage other fumblethumbs like myself to try it.

I purchased the Atari-joysticks from Sears as suggested in the above column (catalog #6C99835) for $9.95 each. Since I could not find a mating connector, I cut off each end and attached them to a User Port connector as per the instructions (see also refs 2 and 3). The latter connector can be purchased from any number of companies (e.g., AB Computers). Four signal diodes (1N662), whose specifications I believe are not critical, are also used in this super simple interface circuit which can be constructed in about 1/2 hour. Check to make sure that the diode cathodes are connected as shown in the circuit diagram (1) and the color coded wires of the joystick are connected to the proper pins of the User Port connector. Otherwise no special precautions are necessary. I did it right the first time (yes brag!)

This neat hardware would, of course, be useless without software to work it. Cursor magazine (4) has supplied several programs which have a joystick option. I tried these first (Demon, Canyon, Pickup and Nab) with happy success. These programs are written to work with various model joysticks wired and oriented in different ways. Since my configuration was fixed I modified the joystick subroutine in each Cursor program to skip the test step. That procedure can be tedious if a program is run repeatedly. The following changes in the Cursor joystick subroutine should work for all Atari-type joysticks wired according to the circuit diagram in reference 1:

```
61030 PRINT: FOR I = 0 to 5: READ T, P:
              GOSUB 61120:
              T (I) = T: J(T) = P: NEXT I
61120 T = INT (T/16) AND T: RETURN
61250 DATA 255, 5, 223, 4, 239, 6 127, 8, 191, 2,
      63, 0.7, 1, 9, 3
```

I ran the original program once to find the values in the T array and used data statements in the modified program in order to skip the test step. This considerably speeds up the beginning of a game. I have deliberately ignored the rational behind the bit manipulations in statement 61120. It is not necessary to understand that in order to use joysticks. I emphasize this point because I hope it will encourage PET users who may strain at bits to attempt projects such as described here. In a future article I may try my hand at a tutorial for those who wish to delve into this mystery further.

If a program was not written with a joystick in mind another modification procedure must be used to convert it away from number pad use. As an example I modified the program Obstacle (5) which utlitzes the full keyboard as two pseudo-joysticks. Each player manipulates his piece (screen character) with the now standard keyboard patterns, "W, X, A and D" and "8, 2, 4, 6". The object is to keep from running into the screen trail left by the other player - the first to do so loses. As with many games it is easier to use than to describe. The following statements in the original program are used to sense a keypress by a player and change direction if necessary:

```
260 GET R$
265 IF R$ = "W" THEN AD = 1
        •
        •
        •
300 IF R$ = "8" THEN BD = 1
        •
        •
        •
```

If W is pressed the direction of the player on the left's piece is changed to up. If 8 is pressed the direction of the player on the right's piece is also changed to up. Player and direction are determined by the above keyboard pattern which can be learned quickly by new players.

Converting a program like this to joystick use is very easy. The following statements will do this:

```
260 M = PEEK (59471)
265 IF (M OR 240) = 247 THEN AD = 1
        •
        •
        •
300 IF (M OR 15) = 127 THEN BD = 1
        •
        •
        •
```

The elipsis can be fleshed out with the help of the table below. A peek at the User Port gives a unique value for each position of the joystick as long as only one is being used. It is necessary to "mask" with 15 or 240 if the possibility exists of both being used at the same time. If only one joystick is used and it doesn't matter which, a further operation with the peeked value generates simpler numbers (shown in the last column of the table). This "trick" was used in statement 61120 in the cursor subroutine and the resulting values used in the main body of the program.

In summary, I feel this is an excellent project for a beginner. The interface is quickly and easily put together. It is not difficult to modify existing programs for joystick use. "Old" PET owners with disappearing keytops will appreciate the saving on wear and tear. To be fair I should say that every program is not suitable for conversion. When fine control of movement is required joysticks may be difficult to use. Some players with poor hand-eye coordination may still prefer the keyboard. As for me it seems quite natural to chase and catch some seemingly elusive demon with my movements under reflex control by a joystick.

REFERENCES

1. **Kilobaud Microcomputing,** Robert W. Baker, January 1980, p.14
2. **PET User's Group Newsletter,** Vol. 0, No. 3. p.6, 1978
3. **Best of PET Gazette,** Chuck Johnston, p.42, 1979
4. **Cursor Magazine,** P.O. Box 550, Goleta, CA 93017
5. **Dilithium Press,** P.O. Box 92, Forest Grove, OR 97116

TABLE

T = PEEK (59471)

POSITION	Joystick 1	Joystick 2	T = INT (T/16) AND T
center	255	255	15
left	223	253	13
right	239	254	14
up	127	247	7
down	191	251	11
button	63	243	3
left up	95	245	5
right up	111	246	6
left down	159	249	9
right down	175	250	10
mask	T OR 15	T OR 240	

©

From: PET AND THE DUAL JOYSTICKS

Chuck Johnson

JOYSTICK ATTACHMENT STANDARD

The Atari Joystick terminates with a DB-9 connector; six of the nine circuits are used as follows:

```
5 4 3 2 1
. . . . .
 . . . .
 9 8 7 6
```

DB-9PIN	FUNCTION
1	Up Switch
2	Down Switch
3	Left Switch
4	Right Switch
5	Not Used
6	Push-Button Switch
7	Not Used
8	Ground (common)
9	Not Used

Pin numbers for the DB-25P are marked on the connector. User Port pins are on the bottom row of a 12 position edge connector. Proper polarization of the edge connector is strongly recommended.

Note that the buttons are wired through diodes to both the UP and DOWN functions of their joysticks (pins E and F and pins K & L). When a button is pressed, the PET data lines react as though the joysticks were pushed up and down simultaneously (an impossible condition for the position switches to generate). The decoding standard will, as we shall see, interpret this condition as a button movement. The diodes act as "one-way streets" for current flow and prevent the real UP and DOWN switches from closing both circuits.

Orientation of the diodes is very important. We want current to flow from the data lines to ground. When soldering the diodes in place, orient them with their cathodes ("banded" ends) connected to the joystick suttons (pins 14 and 22 on the DB-25P).

If the user is not interested in using the buttons, the button lines may be left unconnected and the diodes omitted.

A DB-25P connector will accept two joystick connectors; the DB-25 may then be wired to a User Port edge connector, according to the following table:

JOYSTICK	DB-9S PIN	DB-25P PIN	USER PORT PIN	USER PORT FUNCTION
1. UP	1	9	F	**PA3**
DOWN	2	10	E	**PA2**
LEFT	3	11	D	**PA1**
RIGHT	4	12	C	**PA0**
BUTTON	6	22	Diodes to E and F	**PA2 and PA3**
GROUND	8	24	A	**GROUND**
2. UP	1	1	L	**PA7**
DOWN	2	2	K	**PA6**
LEFT	3	3	J	**PA5**
RIGHT	4	4	H	**PA4**
BUTTON	6	14	Diodes to K and L	**PA6 and PA7**
GROUND	8	16	N	**GROUND**

©

THE apple® GAZETTE

APPLETIVITES AT THE WEST COAST COMPUTER FAIRE

Joe Budge
2507 Elderwood Lane
Burlington, N.C. 27215

The West Coast Computer Faire, held March 14-16 this year, was the focal point for many Apple Computer dvelopments. Several companies introduced significant new software and hardware. These new introductions should greatly enhance the Apple's capabilities. User groups from all over the world convened at the Faire to start an international user's group. The group then sponsored a full day of seminars on the Apple, with subjects ranging from the Apple's invention to its application in music and foreign language instruction.

On March 13 representatives of 60 member clubs met in San Francisco to formally initiate the International Apple Core. Directors were elected, lengthy discussions were held, and policies were worked out. Directors and officers met for the following two days arranging quite a few important details.

The International Apple Core (IAC) will be a non-profit organization dedicated to the exchange of information among Apple Computer users. This will be broadly interpreted to include technical information, software, programming information, and anything else which can benefit Apple users throughout the world. In addition the IAC will provide for communications between members and the various product manufacturers. The International Apple Core will not be affiliated with Apple Computer or any other manufacturer. Specific areas of IAC activity will include publication of the "Orchard", maintenance of a public-domain software library, support of special interest groups, support of software and hardware standards, technical support, promulgation of ethics, and organization of annual Apple - faires.

The IAC does not intend to have individual memberships. Only non-profit Apple user groups may be members. Educational institutions and other interested non-profit organizations may become associate members. They will be entitled to all the free printed information the IAC provides its members, but can not vote for directors. Commercial enterprises may become sponsors, entitled to information and participation in standards establishment. They, too, would be non-voting, but would receive preferential advertising treatment in IAC publications. Dues for members and sponsors will be $50 and $200, respectively. They will be collected every January First. Associate members pay no dues.

The "Apple Orchard" will be the IAC's official magazine. It will be published quarterly, starting September First. The "Apple Orchard" will replace future editions of "Contact." "Contact" had been sent by Apple Computer to all registered users. Unlike "Contact," the "Orchard" will be available either by subscription or by sale through clubs or stores. Apple Computer Company intends to purchase some "Orchards" from the IAC. One copy will be sent to each newly - registered Apple owner to tell them about user groups.

A number of committees and special interest groups were established to deal with specific subjects. The IAC will try to establish a hot-line for technical and software problems. Until permanent arrangements can be made, Apple owners will have to continue using current lines provided by various clubs and Apple Computer, Inc. Neil Lipson's software committee will be collecting, documenting, and distributing a diskette a month to the member clubs. These will contain public domain software and will be distributed free of charge. The IAC will also distribute to members application notes from Apple Computer and other suppliers. Special interest support includes educational and legal application groups, a handicapped usage group, and a ham radio net. Anyone wishing to contact the International Apple Core may write them at P.O. Box 976, Daly City, Ca. 94017.

The International Apple Core sponsored a series of Apple seminars during the second day of the Faire. The subjects covered a variety of fronts, from the workings of graphics and the disk operating system to applications of different languages such as Forth and Pascal. The most interest was shown in a pair of lectures by Steve Jobs and Steve Wozniak, the inventors of the Apple. They told how the Apple started as a video terminal used to play games on telecommunications networks. Being members of the Homebrew Computer Club, they became interested in microprocessors. With that, Jobs and Wozniak started to add processing and memory to the video terminal, added a hand - assembled Basic interpreter, and so on. Before they knew it, they had a microcomputer on their hands. The new machine was so popular at the meetings that Jobs and Wozniak decided to print up a circuit board. That way their friends could build the new micro too, while they were freed from supervising everyone's assembly and debugging. To recoup costs the two began selling computer boards with a set of instructions and a parts list. That worked fine for a few months. Then one store owner decided to take 50 boards, but only pre-assembled and tested. Thus was born Apple Computer Company.

Apple Computer has, obviously, grown considerably from those meagre beginnings in 1975. Upon realizing the advantage of the single-board computer over the buss machines, Apple made a few improvements to smooth out the rough edges. They added I/O decoding to the peripheral slots, a cassette interface, dynamic RAM, and better video. This created the Apple II. The Apple II has been so successful and so versatile that Apple has determined not to obsolete it. New equipment will allow Apple to easily specialize in certain areas. For example rumors abound of a new business machine similar to Radio Shack's Model II, as well as of a new Pascal teaching machine. Whether these rumors are true remains anyone's guess. Nevertheless Apple is trying to solve the problem of making a computer that is at once sophisticated and yet doesn't require high intelligence to operate.

Many fascinating new products introduced at the Faire demonstrate the Apple II's potential and versatility. The show - stopper was Microsoft's new Z-80 board. This board, which plugs into one of the Apple's peripheral slots, can supress the 6502 microprocessor and turn on its own Z-80. It uses the regular Apple memory and I/O devices. Priced at $350, the board comes complete with two diskettes containing CP/M and Advanced Microsoft Basic. Production and distribution should start in May. Fortran, Cobol, and other advanced languages should become available for the Z-80 card during the summer. On the applications side of the fence an equally significant development was Programma International's unveiling of a general accounting package. Priced around $200, it appears every bit as capable as IBM's package for the 5110. IBM's software sells at 10 times the price. Three manufacturers introduced eighty - column video boards for the Apple's text display. These are for telecommunications, word processing, and use with the Pascal editor. The boards were priced from $200 to $400, and should be available in most retail outlets at press time. With all these new developments, the next year looks to be extremely interesting for Apple users everywhere. ©

The APPLE PI Trading Library

Terry N. Taylor
Club Librarian

We have well over 3500 public domain programs on 148 disc sides in our Apple PI library. Copies of our November catalogue (showing 2780 programs) are still available for $1 plus 50 cents postage from our newsletter editor:

Rodney Nelsen
9711 Josephine St.
Thornton, CO 80229 Ph. (303) 451-7577

Later arrivals are shown in our monthly newsletters, available at $1.00 each.

While most of the approximately 800 plus programs on our Apple PI slices #1 to 20 and 178 to 200 are either original efforts by our members (both local and out-of-state) or have been modified for the Apple II from a magazine listing, there are probably a few donations that originally came from other sources. Copyrighted programs are not supposed to be in our trading library out of respect to the commercial copyright holders. So if a copyrighted version of a program appears in error, please let us know so that it can be removed. Also, since many of our members belong to several Apple user groups, if they have accidentally donated someone else's program that they did not have permission to donate to our group; and we listed it unknowingly, we are sorry again. Please inform us. The approximately 2400 programs that we have received in trade from 21 other Apple user groups are listed on our Apple PI slices #21 to 126.

Our programs were originally cataloged on a 32K Apple without the Applesoft ROM card. All of the programs on our disc slices (i.e. from 1 to 20 and 179 to 200) should have been converted to the A.S. ROM card and should now work on a 48K Apple. (In fact, some programs require 48K to operate.) To convert an Applesoft program for non-ROM use; load the program, call 3314. Then either save it or run it. To convert a program from a non-ROM card source to ROM, call 54514. This is not needed if you have either DOS 3. 2. or 3.2.1.

The Hello programs on our disc sides (i.e., Apple PI slice XXX (YYYYYYYY)) identify the type of programs on the disc (i.e. the word 'Games', 'Finance', 'Utilities', etc. will appear where the Y's are shown in brackets). The Hello program on the disk slices that we have received from other clubs is formatted the same way, except the club name or location is shown in place of the Y's when the Hello program is booted. Their name is also shown on the screen so that the donating groups receive credit for their programs.

If you have a bombout, or see possible improvements, please either note the circumstances (including the configuration of your computer) of the bombout on a card to us so that we can fix the program if needed--or improve it yourself and resubmit it as your donation. Either way, someone else will then not have to wrestle with the same bugs you did. The reason we need your configuration is that not all programs will run on every Apple (you might not have a needed accessory). As a side note, if the hires page from a previous program is still on, then (1) press CTRL-C to stop the program, (2) Poke -16298,0 to turn off the hires page, and (3) type run to start the program again. That should fix that problem. All lores Graphic programs should automatically turn off the hires page as one of their first statements. But some don't.

TO TRADE (FINALLY!!!)

FOR NON-LOCAL MEMBERS. Other Apple user groups, or individuals who just want to trade. Please either send your discs or cassettes of programs to me:

Terry N. Taylor
12319 E. Bates Circle
Aurora, Colo 80014 Ph. (303) 750-5813

I will copy your programs and return your choice of our slices on your discs and mail them right back to you. If you are a brand-new Apple user group, we will trade you two sides for one to help you get started, or even advance some programs on the understanding that you will later send some of your own programs back to us. Fair enough?

When you donate programs, please take a minute to jot down on a 3 by 5 card the following:

1) Name of the program (exactly as it will be on the disk).
2) What language the program is in.
3) What type of program it is (i.e., Music, Finance, Game, Hires, Etc.)
4) Author's name and, if different, who actually listed the program into the Apple.
5) Source of program (i.e., original program, or the Title, Page, and Issue # of the magazine copied from)
6) Length of program (if known) Also the starting address if it is a Binary program.
7) Whether the program calls up any other programs, and, if so, their names
8) What accessories, if any, are needed. (i.e., programmers aid, 48K, type of printer, disc or cassette based, etc.)
9) Finally, a short description (say 25 words or so) of what the program does.

If you don't know one of the areas, just leave it blank. Unfortunately, with over 3000 programs, not too many people can remember exactly what each program does just by looking at the title; so any help here is deeply appreciated. Still, don't let it stop you from donating your program if you feel it is a hindrance. It won't be the only one that needs some documentation.
Thanks for reading this far and happy programming.

Editor's Note: In future issues, we'll present interviews with Marketing personnel from Atari and Commodore.

An Interview With Taylor Pohlman Apple's Product Marketing Manager

Michael S. Tomczyk

Recently, I had an opportunity to talk with Taylor Pohlman, Apple Computer's Product Marketing Manager. Taylor is 33 years old and came to Apple less than a year ago from Hewlett Packard, where he was educational marketing manager. During our interview, he discussed two important areas -- computer dealers and consumers.

He described two basic types of dealers, noting that dealers are first of all businessmen and secondly, tend to operate in those markets they are most comfortable or experienced in.

The first type of dealer is the *hobbyist who becomes a businessman*. For these dealers, technology is the key. Since the first-round hobbyist market is pretty much saturated, Pohlman feels these dealers have to start moving from hardware -- their original orientation -- to software, and the application problems of non-hobbyist users.

The second type of dealer is the *retail businessman*. To them, selling computers is much like selling hi-fi stereo systems. A fair amount of retail audio-visual and electronics outlets are personal computer dealers, he said, and many have already developed the expertise needed to reach the computer market. A-V stores may already be dealing with schools, for example, and the educational market is a natural for them. Small stores also relate well to small businessmen because the dealers, as businessmen, have themselves encountered problems in general ledger keeping, manufacturing, accounting, etc.

"The personal computer market is emphasizing solution-oriented merchandizing as opposed to hardware-oriented merchandizing," Pohlman said, and this poses important challenges for all types of computer dealers. He said Apple has 600 to 700 computer dealers, all with different interests, sales and facilities. Some sell a single line and others a complete line, from small games to minicomputers.

"The question they all have to ask is, who are those hungry people outside the store who have problems to be solved by a personal computer? At Apple, we try to turn the dealers on to those markets -- and solutions -- whatever the dealer's 'focus'. By 'focus' I mean hobbyist, businessman, educator, home entertainment user. . . .but whatever its focus, a good Apple store doesn't confuse the customer.

"For example, the businessman doesn't want to know this microprocessor runs 20 percent faster than that one. He wants to know will it solve his problem? On the other hand, the store that just has a bunch of hardware sitting around may respond well to the hobbyist by making the hardware available on the floor and letting him look inside." (Incidentally, Taylor Pohlman's definition of a computer hobbyist is, "the individual who's interested in the man-machine interface.")

He kept emphasizing that the key to the overall personal computer market is problem-solving. In this regard, Apple's magazines and other literature are designed to trigger people's problem-solving approaches and stimulate their imagination. He also called attention to the company's seminar program, which encourages dealers to get out and give seminars where the people are -- at Rotary Clubs or real estate groups, for example. Apple provides instructional and advertising materials to dealers for this purpose.

"For the dealer to survive in the new marketplace, he's also going to have to provide service and support -- not just sales," he said. Consequently, Apple has a modular design and can be field-repaired at over 500 dealer-based repair centers, in 24 hours. If the dealer can't repair it, he can simply replace the failed part and send it to Apple for repair. This is especially important to the businessman, who can't afford to have his general ledger or other system go down for a week.

Turning his attention to consumers, he guessed that there are "tens of thousands" of Apples in homes, schools, and companies. "We're the one personal computer vendor that has achieved a truly disk-based population -- as opposed to those who are still out there hyping cassettes. Disk-based software is more sophisticated and makes the Apple more useful and powerful as a problem-solving tool.

"What I, as a consumer, want is a computer that allows me to define a problem and allows the machine to solve it in language and terms that I can understand. If the problem is solved I could care less what the machine is doing." He added that the level of computer awareness is very high but the level of computer *literacy* is not nearly high enough to create the "home computer revolution," mostly because right now using a computer means you have to:

- define a problem
- create an algorhythm to solve the problem
- write programs to express the algorhythm
- put the program in the computer
- run and debug the program

"A lot of time is wasted trying to *translate* information, and this wasted time is directly related to the reluctance -- or lack of reluctance -- of customers to get involved with a personal computer," he said. He went on to say that the home computer revolution will not truly arrive until you can more effectively separate the user from the intricacies of hardware and software, so he doesn't have to understand the inside of the machine or the inside of the software in order to use it.

He also spoke of a "vast dumping ground" where a lot of people who buy computers encounter so many stumbling blocks to using them (language, interface, hardware) that they wind up juking them, and said Apple is working hard to solve this and other problems not only through product development and dealer support, but also by encouraging such vocal forums as the International Apple Core.

He concluded by saying that he doesn't think Apple II is a mature product after two years and indicated that future Apples will maintain compatibility, as opposed to some companies which have come out with new machines which weren't compatible with earlier versions. He said, "If people think Apple is going to somehow change it's product or it's way of doing business, they're wrong." ©

THE ATARI® GAZETTE

MADE IN THE SHADE: AN INTRODUCTION TO "THREE-DIMENSIONAL" GRAPHICS ON THE ATARI COMPUTERS

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

If you know anything at all about the Atari 400 and 800, you probably know that these machines give you access to 256 colors. What you may *not* realize is that these colors are specified with two independent parameters which allow you to create "three-dimensional" objects on the display.

The two parameters of interest are *hue* and *luminosity.* Atari gives you access to sixteen colors (the hues), each of which can be darkened or lightened by setting the luminance to one of eight levels. Traditionally, computers that offer limited colors (sixteen total, for example) pre-select different hues *and* luminosities for each color so that inter-color contrast is always apparent, even when the picture is shown on a black and white display. If two colors have the same luminosity, you will not see any difference between the colors when they are shown on a black and white display - a phenomenon you should demonstrate to yourself sometime.

The beauty of the Atari scheme is that the wide range of available colors leads to the ability to create some pretty pictures, even though only four colors can be displayed at one time. The program presented here illustrates a common graphics task - the representation of a solid three-dimensional object through shading.

Since we can display three colors plus the background in a moderate resolution graphics mode, this lets us represent a shaded cube. After all, you can only see a maximum of three faces of a cube at any given time. The function of the program, then, is to create a two-dimensional representation of a cube in which the "realism" results from the control of the shading on each visible face.

In case you are not familiar with Atari graphics, a short digression is in order. Displayed colors are established by the **SETCOLOR** command which takes the form **SETCOLOR A, B, C** in which **A** is the color register (0-4), **B** is the hue (0-15) and **C** is the luminosity (an even number from 0 to 14). (I don't know why luminosity isn't set with numbers between 0 and 7, but the use of even numbers doesn't present too much of a problem, as you will see.) The hues (see Table I) are the various basic colors you can use to draw your pictures, and the luminosities control the brightness from 0 (very dark) to 14 (almost white). Once you have set the color registers, you need to indicate which register should be used for the various plotting commands. This function is performed with the **COLOR** statement. This statement has the form **COLOR D** in which **D** refers to the color register in which the desired color is located. Now for reasons that I don't understand, the value of **D** is generally larger than the color register number by one. In other words, D = A + 1.

PROGRAM LISTING

```
10 REM ** SHADING DEMO
20 GRAPHICS 23
30 OPEN #1,4,0,"K:"
40 FOR I=0 TO 4: SETCOLOR I,9,4 : NEXT I
50 X0=48 : Y0=36
60 COLOR 1
70 FOR I=0 TO 40
80 PLOT X0,Y0+I : DRAWTO X0+40,Y0+I
90 NEXT I
100 COLOR 2
110 FOR I=1 TO 24
120 PLOT X0+I,Y0-I : DRAWTO X0+I+40,Y0-I
130 NEXT I
140 COLOR 3
150 FOR I=1 TO 24
160 PLOT X0+40+I,Y0-I : DRAWTO X0+40+I,Y0+40-1
170 NEXT I
180 FOR I=0 TO 2
190 GET #1,A
200 IF A<48 THEN A=48
210 SETCOLOR I,1,2*(A-48)
220 NEXT I
230 GOTO 180
```

Now that these tips on Atari color are under your belt, it is time to try out the program.

The listing starts out by setting a moderately high resolution full-frame graphics mode in statement **20**. This mode allows the display of four colors and contains 160 x 96 picture elements - plenty for our need. The **OPEN** statement lets us use a **GET** statement to receive data from the keyboard without having to press **RETURN.** Note that the Atari version of **GET** is *very* different from the version you may be accustomed to from Microsoft BASIC. Next, the color registers are all set at the same color value so that when the cube is first drawn you cannot see it. The front face of the cube is drawn in **COLOR 1** (from **SETCOLOR** register 0) in lines **70-90**, and the other two faces are drawn in **COLOR 2** and **COLOR 3** in lines **110-130** and **150-170** respectively. At this point the computer waits in line **190** until a key is typed. (Note that in Microsoft BASIC the program would not stay at a **GET** command, but would look once and be on its way.) Since I expect to be **GET**-ting a keystroke from keys 0 through 7 (which have the Atari-ASCII values 48 through 55), lines **200** and **210** convert the keystroke to an even number between 0 and 14 for use in the **SETCOLOR** command. This program looks for three keystrokes - one for each face of the cube. As each key is typed (try 5, 6 and 7, for example) a cube face will become visible. The result is that a "three-dimensional" representation of a cube is now nicely displayed on your TV screen.

If you want to change the shadings, type three more numbers between 0 and 7 and see what happens. Next, for some more excitement, type J, K and L. Once again you will see the shaded cube, but the color will have changed from gold to more of a magenta. As you can see, luminance values greater than 14 cause the hue to change.

Now that you know about shading, you should be able to make some truly beautiful pictures on the Atari computers!

Table 1. Hue Values For The Atari Computers

COLOR	HUE VALUE
GRAY	0
LIGHT ORANGE	1
ORANGE	2
RED-ORANGE	3
PINK	4
PURPLE	5
PURPLE-BLUE	6
BLUE	7
ANOTHER BLUE	8
LIGHT BLUE	9
TURQUOISE	10
GREEN-BLUE	11
GREEN	12
YELLOW-GREEN	13
ORANGE-GREEN	14
LIGHT ORANGE	15

©

Atari Tape Data Files
A Consumer Oriented Approach

Al Baker
2327 S. Westminster
Wheaton, IL 60187

Introduction

This article is based on a major axiom of consumer computing:

Easier is Better

The specific corollary when writing a program which saves data between program runs is:

> *Use only one tape. Program and data should be on the same tape. They should, in fact, be the same thing.*

A consumer should be able to load his program, run it to update his checkbook and balance his budget, and then save the program on tape when done. The next day, he can load his program and all data changes from the previous day should be there.

"Impossible", you say? Well, perhaps. It is certainly impossible on some of the computers on the market. But it is not impossible on the Atari. The trick is to fool Atari Basic into saving all dimensioned variables when a program is saved to tape. We won't try to save the simple variables. Since I am not a revered expert, I won't make the mistake of saying this is impossible. (But, I think it's impossible.) Saving the dimensioned variables with a program is relatively easy.

Write your program

Listing 1 is a simple program. Nothing tricky. But notice that I print the dimensioned variables in Lines 70-130 and then assign values to them in Lines 140-190. I am assuming the variables have valid contents before changing them! The only important restriction here is to type the line containing the DIM statement first. It doesn't have to be the first line in the program. Just make sure it is the first line *typed.*

The Atari Basic variable symbol table is constructed when each line is typed in, not when the program is run. Later we will need to find the locations of the string variables in the table. This is easier if they are the first variables present. For a more complete discussion of the symbol table, see the text in the box.

```
50 DIM A$(10),B(2,3)
70 ? A$
80 FOR I=0 TO 2
90 FOR J=0 TO 3
100 PRINT B(I,J),
110 NEXT J
120 PRINT
130 NEXT I
140 ? "STRING=";:INPUT A$
150 ? "I=";:INPUT I
160 IF I=9 THEN 200
170 ? "J=";:INPUT A:B(I,J)=A
190 GOTO 150
200 END
```

Suppose the program is already written and you didn't type the DIM statement first. Write your program to tape using the command LIST"C". Type NEW. Now type the DIM statement from your program with the string variables first. Finally, reload the program from tape with the command ENTER "C". Now the string variables are at the beginning of the variable tables.

Protect the Dimensioned Variables

The next step is to fool Basic into treating the dimensioned variables as part of the program. Also, you have to add the code to let the program save itself to tape. In an application, saving the program to tape will be the final program option selected by the user. In Listing 2 this is added to the program in lines 200 through 230.

```
50 DIM A$(10),B(2,3)
70 ? A$
80 FOR I=0 TO 2
90 FOR J=0 TO 3
100 PRINT B(I,J),
110 NEXT J
120 PRINT
130 NEXT I
140 ? "STRING=";:INPUT A$
150 ? "I=";:INPUT I
160 IF I=9 THEN 200
170 ? "J=";:INPUT A:B(I,J)=A
190 GOTO 150
200 A=PEEK(140)+PEEK(141)*256
210 A=A+82
220 POKE 141,INT(A/256):POKE 140,A-PEEK(141)*256
230 CSAVE
```

Locations 140 and 141 contain the address of the end of the computer program. Program line 200 places this address in the variable A. In line 210 we add the size of the dimensioned variables. Each string variable contains as many bytes as its dimension. Each numeric array contains 6 times the number of members of the array. The B array is $6x(2+1)x(3+1) = 6x3x4 = 72$ bytes. Thus we had to add $10 + 72$ or 82 to the end of the program in the example.

Now run the program and let the internal CSAVE create a tape. Turn the computer off and then on. Now reload the newly created program from tape. For some reason this step is important. (I don't know why.) If you do not use the new tape, this procedure won't work.

Finish the program

We now have a program in memory which has an invalid program - end pointer. See the third listing. Add lines 10 through 40 to your program. Make sure that you use the correct number instead of "-82" in line 10. Remember that this number is the size of your dimensioned variables.

Refer to Table 1. Locations 140 and 141 form the program-end address. Locations 142 and 143 form the stack address and locations 144 and 145 form the pointer to the end of memory used by the program. The RUN command sets all of them equal to the incorrect end-of-program pointer. Lines 10 through 40 correct them.

Here comes the only hard part. You are going to have to peek around in memory. The RUN command sets the length of all strings to zero. You must repair their lengths if you want to save string data.

Table 1.

These two byte addresses point to important areas used by Atari Basic.

Use this	To get the location of this
PEEK(130) + PEEK(131)*256	Variable name table
PEEK(134) + PEEK(135)*256	Variable value table
PEEK(136) + PEEK(137)*256	Beginning of program
Use these only when program running	
PEEK(140) + PEEK(141)*256	End of program and beginning of dimensioned variables
PEEK(142) + PEEK(143)*256	End of dimensioned variables and beginning of stack
PEEK(144) + PEEK(145)*256	End of memory used by program

Look at Table 3. The third entry in the variable value table is the string ALPHA$. Its current length is 5 + 0*256 or 5. These two bytes must be set to the correct length of the string. Type the command: PRINT PEEK(134) + PEEK(135)*256. Now you know where the variable value table is. If you have been writing the program in the listings, you should get the answer 2056. Assume the string is the first entry in the table. The locations of the length is

Table 2

The variable name table: Entry lengths are different. Box symbolizes that 128 is added to ASCII value of last character to show the name's end.

Variable	Variable name	
AB1	AB [1]	3 character number name
AR(3,4)	AR [(]	2 character array name
CDOG(17)	CDOG [(]	4 character array name
ALPHA$ (10)	ALPHA [$]	6 character string name
E	[E]	1 character number name
FIG	FI [G]	3 character number name

Note: Variable names can be up to 120 characters long and are completely unique. Variable ABC is different from variable ABCD. Variable names DO NOT appear in the program in memory. Only a 1 byte pointer to the variable name in the variable name table appears.

2060 and 2061. Since the length of the string of data being saved in the example is 10, I set location 2060 to a 10 in line 60 of the program.

Try it out

The program is complete. Save it. Now Run it. You will probably get garbage in the print out. Put a 10 character string in the string variable. Now put numbers in various entries in the B array. Typing a 9 for the I subscript will end the program with a CSAVE. Do this CSAVE onto a new tape. Turn the computer off and on. Now load this new copy of the program and run it. Viola! The data is still there! Now load this new copy of the program and run it. Viola! The data is still there! Now just imagine that this was your budget information, address book or other files. You have a no-hassle one-tape system.

Conclusion

I have provided more information about this internals of the Atari than is really necessary to solve this problem. If you are interested in this kind of information, study it. If not, skip it. If you have any questions, I would be glad to answer them. One warning. Do not press break while the program is running and then type RUN. Always use the CONT

Table 3

The variable value table: Each entry is eight bytes.

Variable	Contents	Table Entry 1 2 3 4 5 6 7 8	Meaning
AB1	5	0/ 0/64/5,0, 0,0, 0	First byte is 0: this is a number. Second byte is 0: this is the first entry. 64 is the exponent. 5 is the binary coded decimal value.
AR(3,4)	doesn't matter	64 + 1/1/0, 0/4, 0/5,0	64 makes this an array. + 1 means that it has been dimensioned. 0 + 0*256 is the displacement into the array area. 4 + 0*256 is the size of the first dimension and 5 + 0*256 is the size of the second dimension.
CDOG(17)	doesn't matter	64 + 1/2/120,0/18,0/1, 0	This array is displaced 120 bytes into the array area, and it is dimensioned 18 + 0*256 by 1 + 0*256.
ALPHA$(10)	"12345"	128 + 1/3/228,0/5, 0/10, 0	128 makes this a string. + 1 means that it has been dimensioned. It starts 228 + 0*256 bytes into the array area. The current length of the string is 5 + 0*256. The maximum size of the string is 10 + 0*256.
E	.05	0/ 4/63/5,0, 0,0, 0	This is a number. The exponent is now 63 so the number is only 1/100 of its integer value, or .05.
FIG	-5	0/ 5/64 + 128/5, 0,0, 0	This is a minus number (+ 128 on exponent)

command after pressing BREAK. Otherwise the statements in lines 10-40 will destroy the program data. This can be prevented if you know what the correct value of A should be in line 10. Replace line 10 with 10A = n, where n is this number. Do this for your finished product.

```
10 A=PEEK(140)+PEEK(141)*256-82
20 POKE 141,INT(A/256):POKE 140,A-PEEK(141)*256
30 POKE 143,INT(A/256):POKE 142,A-PEEK(143)*256
40 POKE 145,INT(A/256):POKE 144,A-PEEK(145)*256
50 DIM A$(10),B(2,3)
70 ? A$
80 FOR I=0 TO 2
90 FOR J=0 TO 3
100 PRINT B(I,J),
110 NEXT J
120 PRINT
130 NEXT I
140 ? "STRING=";:INPUT A$
150 ? "I=";:INPUT I
160 IF I=9 THEN 200
170 ? "J=";:INPUT A:B(I,J)=A
190 GOTO 150
200 A=PEEK(140)+PEEK(141)*256
210 A=A+82
220 POKE 141,INT(A/256):POKE 140,A-PEEK(141)*256
230 CSAVE
```

The Atari Basic Symbol Table

Most Basic interpreters assign values to the symbol table as the program is run. Not true with the Atari. New variables are placed in the symbol table when the program line they are contained in is first typed.

If you later change variable names, the old variable names are not removed from the table. They stay forever! Even the CLR command does not remove them. They continue to take up room. How much room? 8 bytes plus the length of the name. Add another byte if the variable is an array.

Fortunately, it is possible to clean up the variable table. Write the program to cassette using the command LIST"C", type NEW, and then reload the program from tape with the command ENTER"C".

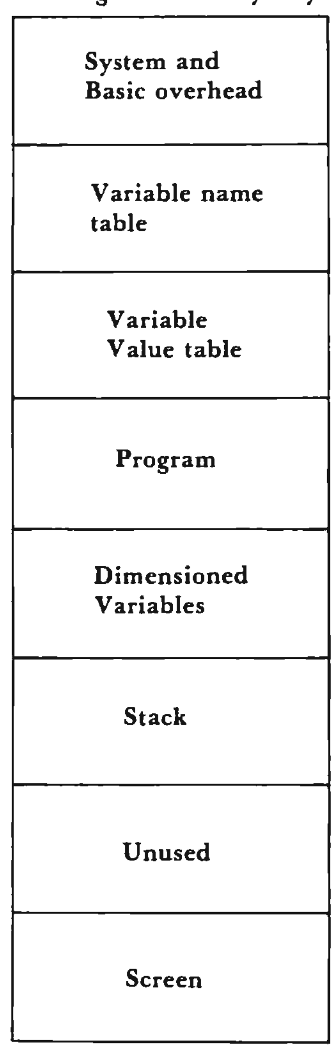
A program can often be made to run faster by placing selected variable at the beginning of the variable table. This decreases the time it takes to find variables which are used in time-critical routines.

To place these variables at the beginning of the variable table, write the program to cassette using the command LIST"C" and then type NEW. Now use those variables. For example, if the variable A must be the first variable in the table, type A = 0. If the string B$ must be used, type DIM B$(1). You are "ordering" the variable table. When you have finished placing as many variables in their correct order as you want, load the program you saved to tape with the command ENTER"C". This does not interfere with the contents of the variable table.

Changing Atari program to save the dimensioned variables.

- Get the program working.
- Place the string variables at the beginning of the variable table.
- Change the program so that it internally points the program-end address past the dimensioned variables and then saves itself to tape.
- Run the program, creating a copy on tape.
- Turn the computer off, on, and then reload the program.
- Add the statements to the beginning of the program to correct the program-end pointer, stack pointer, and end-of-memory pointer.
- Add the code to restore the actual string variable lengths to the variable value table.
- Save your finished program to tape.

Figure 1:
Basic Program Memory Layout



©

"ENTER" WITH ATARI

Len Lindsay

The power to insert or append program segments is built right into your ATARI computer, even though it is not mentioned in the ATARI documentation.

Here is an example of how this capability can be utilized. Let's say you just finished writing a program called FASTTRAP. You have it saved either on tape (CLOAD) or an disk (SAVE"D:FASTTRAP"). Now you are developing another program called NEWPROG. There are several routines in FASTTRAP that you would like to include in NEWPROG. The hard way to do this, of course, is to type them in again. The easy way is to enter them from FASTTRAP, and here is how to do it.

Any program sections that you wish to merge into another program should be recorded on tape or disk using the LIST command. The format for tape is:

LIST"C:",sssss,eeeee

The format for disk is:

LIST"D:FILENAME",sssss,eeeee

The sssss should be replaced by the starting line number of the section, and the eeeee should be replaced by the ending line number. If you wish all the lines recorded you may leave off the line numbers.

There are two routines in FASTTRAP that you would like to use in NEWPROG. Lines 2000 to 20099 are a routine that allows the ATARI to determine which plug your joystick is attached to. With the program FASTTRAP in memory save the routine to tape like this:

LIST"C:",20000,20099

or to disk like this:

LIST"D:IDENTIJST.SUB",20000,20099

The other routine in FASTTRAP you would like to reuse is one that allows the user to easily choose what color his piece will be. It is in lines 20300 to 20399. Save this routine to tape like this:

LIST"C:",20300,20399

or to disk like this:

LIST"D:COLORSET.SUB",20300,20399

You still have your program FASTTRAP saved on tape or disk. In addition you now have two other tapes or two other disk files, as created above. Now type NEW and enter your new program, NEWPROG, as usual. At any time during your programming you may append the routines previously recorded. Here is how you would insert the routine to set the piece color from tape:

ENTER"C:"

or from disk:

ENTER"D:COLORSET.SUB"

Now you might as well insert your other routine like this (from tape):

ENTER"C:"

or from disk:

ENTER"D:IDENTJST.SUB"

After you have inserted these, do a LIST and you will see that your program does indeed now include the two routines. Lines **ENTERed** in this manner will automatically be placed in the appropriate section of the program (by line number). If an identical line number is already used in your program, it will be replaced by the one you are **ENTERing.** You can insert or append program sections in any order you wish. If you accumulate a library of subroutines in this manner, writing new ATARI programs will be easier, since some of the program is all ready to go, ready to be **ENTERed** whenever you need it. ©

Atari Program Saving-Part II

Len Lindsay

Last issue I listed some methods of SAVING a program to tape. Here are the updates for disk.

To save a program to disk in the normal way (tokenized form) enter this:

SAVE"D:FILENAME.EXT" [RETURN]

If you have more than one disk, you can save to disk number two by using D2 in place of the D in all examples.

A program saved in this normal manner can be loaded back by this command:

LOAD"D:FILENAME.EXT" [RETURN]

You also can save your program onto disk in its untokenized form like this:

LIST"D:FILENAME.EXT" [RETURN]

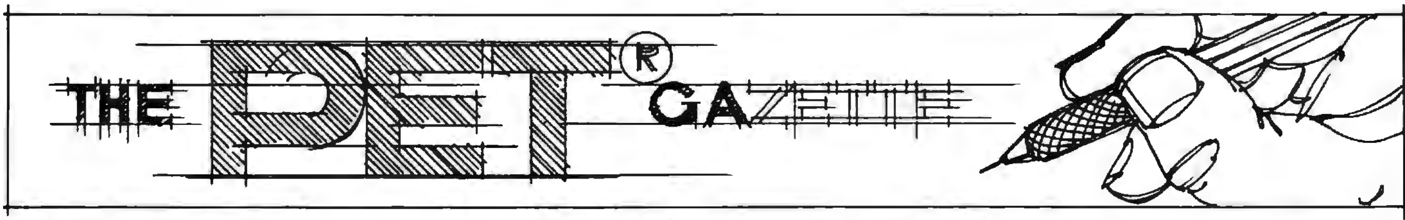
It then can be read back using this command:

ENTER"D:FILENAME.EXT" [RETURN]

Using LIST, you can save the whole program (as above) or just a section of it. To save any section, simply specify the starting and ending line numbers of that section like this:

LIST"D:FILENAME.EXT",100-900 [RETURN]

This will save just lines 100 through 900. ©



Block Access Method Map for a Commodore 2040 Disk Drive

Tom Conrad
7167 Via Lomas
San Jose, Calif. 95136

OVERVIEW

The Block Access Method (BAM) map program will allow you to see where your files are allocated. You can save and delete files and observe the allocation technique.

DESCRIPTION

The purpose of the BAM is to protect allocated files so they are not written over and therefore destroyed. The BAM map resides on the directory track 18. The BAM is in the first half of sector 0. The layout looks like this:

BAM Dump

TRACK 18 SECTOR 0

```
                                          Track
     a          b                        NUMBER

 00 :[12 01]01 00[15 FF FF 1F]              1

 08 : 15 FF FF 1F 15 FF FF 1F              2  3
 10 : 15 FF FF 1F 15 FF FF 1F              4  5
 18 : 15 FF FF 1F 15 FF FF 1F              6  7
 20 : 15 FF FF 1F 15 FF FF 1F              8  9
 28 : 15 FF FF 1F 15 FF FF 1F              10 11
 30 : 15 FF FF 1F 15 FF FF 1F              12 13
 38 : 15 FF FF 1F 15 FF FF 1F              14 15
 40 : 15 FF FF 1F 15 FF FF 1F              16 17
      ----------------------------
 48 : 12 FC FF 0F 14 FF FF 0F              18 19
 50 : 14 FF FF 0F 14 FF FF 0F              20 21
 58 : 14 FF FF 0F 14 FF FF 0F              22 23
 60 : 14 FF FF 0F -----------              24
      -------------
                  12 FF FF 03              25
 68 : 12 FF FF 03 12 FF FF 03              26 27
 70 : 12 FF FF 03 12 FF FF 03              28 29
 78 : 12 FF FF 03 -------------            30
      -------------
                  11 FF FF 01              31
 80 : 11 FF FF 01 11 FF FF 01              32 33
 82 : 11 FF FF 01 11 FF FF 01              34 35
```

a - Address of the next sector which is where the directory begins.
b - The start of the BAM map for track 1.

Detail Explanation

```
                   a  b  c  d
00 : 12 01 01 00 [15|FF|FF|1F]
```

a - Total free sectors for track 1. In this case it is hex 15 or decimal 21. Since track 1 has a maximum of 21 sectors, track 1 is totally empty.

b - The bit configurations for sectors 0 thru 7. Bit on means empty sector and bit off means allocated sector.

```
hex FF = bits "1111 1111"
               |||| ||||
               |||| |||0th sector
               |||| ||1th sector
               |||| |2th sector
               |||| 3th sector
               |||4th sector
               ||5th sector
               |6th sector
               7th sector
```

Therefore all sectors are empty.

c - bit configurations for sectors 8 thru 15

```
hex FF = bits "1111 1111"
               |||| ||||
               |||| |||8th sector
               |||| ||9th sector
               |||| |10th sector
               |||| 11th sector
               |||12th sector
               ||13th sector
               |14th sector
               15th sector
```

d - bit configurations for sectors 16 thru 20.

```
hex 1F = bits "0001 1111"
                  | ||||
                  | |||16th sector
                  | ||17th sector
                  | |18th sector
                  | 19th sector
                  20th sector
```

In any empty disk, the 'd' byte changes from 1F, 0F, 03, 01 to compensate for varying number of sectors per track.

hex 1F is the pattern where there are 21 sectors as in tracks 1 thru 17.

hex 0F is the pattern where there are 20 sectors as in tracks 18 thru 24.

hex 1F is the pattern where there are 18 sectors as in tracks 25 thru 30.

hex 1F is the pattern where there are 17 sectors as in tracks 31 thru 35.

OBSERVATIONS USING THE BAM MAP PROGRAM

The BAM turns off the bits when it allocates a sector. The BAM Map Program looks at these bits and if the bit is on (meaning it is free and has not been allocated) it will print either a "▒" or a white sqare. By looking at the map you can determine how full or empty the disk is.

Varying numbers of sectors.

The reason for the varying number of sectors per track is to pack more data on the disk. Using the worse case which is 17 sectors per track and propagate throughout the disk would decrease the number of sectors per track by 95 sectors or 24k.

Sectors not contiguous

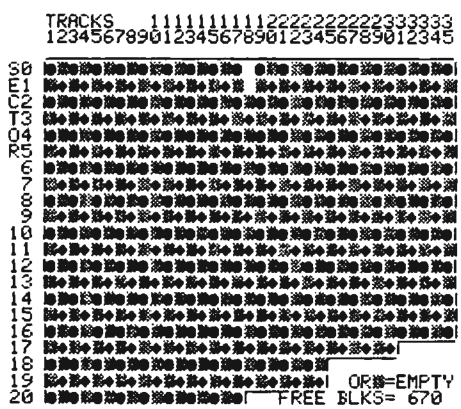
The sectors are in 255 byte blocks. A program file, which is stored in 255 bytes, is not written on the disk contiguously but written approximately one-half track apart. Using the BAM program, you can see when you save a program on a empty disk, that DOS will save the first 255 bytes on track 17 sector 0, the second 255 bytes on sector 10, the third 255 bytes on sector 3, and so on. The purpose of these gaps is to speed up the processing by not waiting for a full rotation of the disk. If the program was written contiguously after each write, DOS would have to wait an entire rotation of the disk to write the next sequental sector. Thus, when one is looking at the BAM Map, one will see where alternating sectors are allocated.

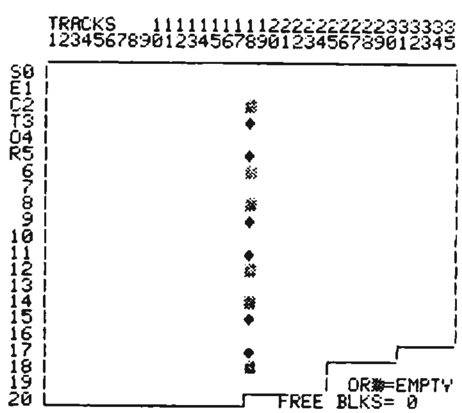
Allocation of sectors

DOS allocates disk space very efficiently. Sectors are allocated around the directory (track 18). This reduces the read/write head movement because it reads the directory first, then reads the file. By having the file close to the directory, head movement is reduced.

Where sectors are allocated

When you delete the first program on a full disk, the BAM Map will show free sectors near the directory. When you save a new program, it will start by allocating those free sectors nearest the directory and will start filling in where you deleted the old file. If the new program is larger than the old program, it will try to allocate sectors further and further from the directory. By using this allocation technique, the need for a disk compress is eliminated.





References:

Parsons, James C., "Display Track and Sector", Commodore Newsletter Vol. 1 Number 8, January 1980.

Commodore Business Machines, Commodore CBM Dual Floppy Disk Model 2040 User Manual, July 1979.

PROGRAM EXPLANATION

100-170 Initialization
180-190 Which drive?
200-430 Prints the BAM Map outline.

```
100 REM* BLOCK ACCESS METHOD DUMP    *
110 REM* WRITTEN BY   TOM CONRAD     *
120 REM*
130 REM*
140 REM*      INITIALIZATION         *
150 DIM A(4)
160 NL$=CHR$(0)
170 T=0: REM TOTAL FREE BLOCKS
175 REM* WHICH DRIVE    *
180 PRINT"ĥ↓↓↓↓↓DRIVE?"
190 GET D$: IF D$="" GOTO 190
195 REM* PRINTS THE BAM MAP OUTLINE *
200 PRINT"ĥ   ɪTRACKSf̂   111111111122222
    ¬22222333333"
210 PRINT"   123456789012345678901234567
    ¬89012345"
220 PRINT"$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$
    ¬$$$$$$$$"
230 PRINT"ɪSf̂0!                                  ¬
    ¬              $"
240 PRINT"ɪEf̂1!                                  ¬
    ¬              $"
250 PRINT"ɪCf̂2!                                  ¬
    ¬              $"
260 PRINT"ɪTf̂3!                                  ¬
    ¬              $"
270 PRINT"ɪOf̂4!                                  ¬
    ¬              $"
280 PRINT"ɪRf̂5!                                  ¬
    ¬              $"
290 PRINT" 6!                                    ¬
    ¬            $"
300 PRINT" 7!                                    ¬
    ¬            $"
```

440-460 Sets up S$ with 25 cursor downs. Sets up T$ with 40 cursor rights.

470 Initializes the requested drive & checks for a disk error.

480 Allocates buffer 0 to channel 2 for block commands that follow.

490 User command that does a block-read. It reads from the requested disk, track 18, sector 0 into the disk buffer and checks for a disk error.

500 Set the channel 2 pointer to the 5th byte in the buffer where the BAM Map starts.

510 Memory-Read Command sets up the byte pointed to by the address 1700.

530-690 Read the BAM and look for empty sectors.

550 The GET# will receive one byte from the buffer via channel 15

560 The byte is in character data and if it is null needs to be changed to CHR$(0) otherwise statement 570 will abend in the conversion to numeric

570 Conversion to numeric

580 This calculates the total free sectors available. A(1) is the total free sectors for that particular track. The total is calculated by summing all the A(1)'s except Track 18 which is reserved only for the directory and can not be allocated for any files.

610 Prints in the upper left corner a alternating ▒ and white squares to show when the program is running.

620 If sector is full (all bits are off) go to next byte.

630-670 Decodes the decimal number into bit pattern and check if bit is on.

640 same as 610

650 If the number minus the powers of 2 is greater or equal to zero then the bit is on and go to 790 to print ▒ on the screen.

700 Program is finished running now clear the square in the upper left corner

710 Print the total free blocks

720 Keep map on screen until any key is typed.

730 Close the files

740 Start program again.

750-780 Subroutine to check for disk error

790-850 Prints ▒ or white square for the empty sector.

840 The first LEFT$ is calculating how many cursor down (sector no.) are needed and the second one is calculating how many cursor LEFT$(track no.) are needed.

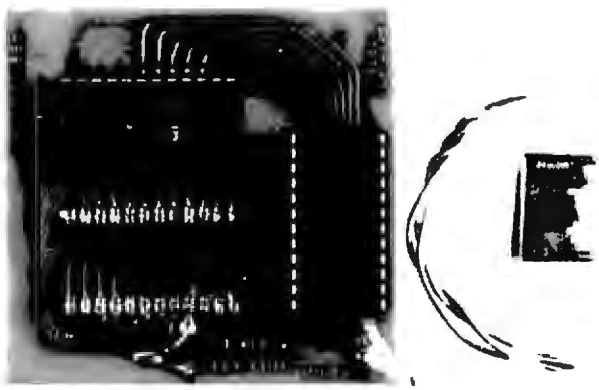
```
310 PRINT" 8'                                        ¬
    ¬              &"
320 PRINT" 9'                                        ¬
    ¬              &"
330 PRINT"10'                                        ¬
    ¬              &"
340 PRINT"11'                                        ¬
    ¬              &"
350 PRINT"12'                                        ¬
    ¬              &"
360 PRINT"13'                                        ¬
    ¬              &"
370 PRINT"14'                                        ¬
    ¬              &"
380 PRINT"15'                                        ¬
    ¬              &"
390 PRINT"16'                                        ¬
    ¬              &"
400 PRINT"17'                                        ¬
    ¬    O####"
410 PRINT"18'                                   ¬
    ¬O#####"
420 PRINT"19'                                  &r ¬
    ¬rOR&=EMPTY"           "
430 PRINT"20'$$$$$$$$$$$$$$$O######"
440 S$="": T$=""
450 W$="↓": FOR I=1 TO 25: S$=S$+W$:NEXT
460 V$=">": FOR I=1 TO 40: T$=T$+V$:NEXT
465 REM* INIT DRIVE AND CK FOR ERROR *
470 OPEN 15,8,15,"I"+D$:GOSUB 760
475 REM* ALLOC BUFFER 0 TO CHANNEL 2 *
480 OPEN 2,8,2,"#"+"0": GOSUB 760
485 REM* BLOCK-READ INTO BUFFER       *
490 PRINT#15,"U1:2,"D$,18,0: GOSUB 760
495 REM* SET BUFFER POINTER           *
500 PRINT#15,"B-P:2,4"
505 REM* MEMORY READ                  *
510 PRINT#15,"M-R"CHR$(0)CHR$(17)
520 REM* SEARCH FOR EMPTY SECTORS *
530 FOR I=1 TO 35
540 :FOR L=1 TO 4
545 ::REM*  GETS A BYTE FROM BUFFER    *
550 ::GET#2,A$
560 ::IF A$="" THEN A$=NL$
565 ::REM*CONVERSION FROM CHAR TO ASCI
570 ::A(L)= ASC(A$)
580 ::IF L=1 AND I<>18 THEN T=T+A(1)
590 :NEXT L
600 :FOR J=2 TO 4
605 ::REM* PRINTS ALTERNATING SQUARES  *
610 ::PRINT "h&":IF INT(J/2)=J/2 THEN ¬
    ¬PRINT "hr "
620 ::IF A(J)=ASC(CHR$(0)) THEN GOTO  ¬
    ¬680:REM* SECT FULL *
630 ::FOR K=7 TO 0 STEP -1
635 :::REM* PRINTS ALTERNATING SQUARES  ¬
    ¬*
640 :::PRINT "h&":IF INT(K/2)=K/2 THEN ¬
    ¬PRINT "hr "
645 :::REM* DECODES DECIMAL TO BIT    *
650 :::IF (A(J)-2^K)<0 GOTO 670
660 :::A(J)=A(J)-2^K:GOSUB 790
670 ::NEXT K
680 :NEXT J
690 NEXT I
700 PRINT"h ":REM* CLEAR SQUARE *
705 REM* PRINTS TOTAL FREE BLOCKS      *
710 PRINT LEFT$(S$,22)LEFT$(T$,23)"FREE ¬
    ¬BLKS="T"↑↑↑↑"
715 REM* MAP ON SCREEN UNTIL KEY IS HIT*
720 GET Z$:IF Z$="" GOTO 720
730 CLOSE 2:CLOSE 15
735 REM* START PROGRAM AGAIN  *
740 GOTO 170
750 REM* CHECK FOR DISK ERROR *
760 INPUT#15,EN$,EM$,ET,ES: IF EN$="00" ¬
    ¬THEN RETURN
770 PRINT "rDISK ERROR:r " EM$ " " EN$,
    ¬ET "," ES
780 END
790 REM* PRINT ALTERNATING PATTERN *
800 IF INT(I/2)= I/2 AND INT(K/2)= K/2 ¬
    ¬THEN C$="&"
810 IF INT(I/2)<>I/2 AND INT(K/2)= K/2 ¬
    ¬THEN C$="r "
820 IF INT(I/2)= I/2 AND INT(K/2)<>K/2 ¬
    ¬THEN C$="r "
830 IF INT(I/2)<>I/2 AND INT(K/2)<>K/2 ¬
    ¬THEN C$="r&"
840 PRINT "h" LEFT$(S$,3+((J-2)*8)+K) ¬
    ¬LEFT$(T$,2+I) C$
850 RETURN
```

©

Said the Toolkit to the Word Processor: "You're in My Space!" Said the Word Processor to the Toolkit: "Let's Share...here's

Socket 2 Me™!"

From the original producer of peripherals exclusively for PET lovers everywhere . . . the device that allows you to select between the BASIC Programmer's Toolkit and the Commodore Word Processor II while they occupy the same address space.



The **Socket 2 ME . . . doubles** your memory expansion in a single **socket**. It's a 2.5" x 2.75" board that fits neatly into the Toolkit/Word Processor socket on the main logic board of all *new* PETs. Then both the Word Processor and the Toolkit plug into the **Socket 2 ME**.

A miniature slide switch — part of the kit — mounts with double-stick tape (supplied) to the front part of the right side of the PET base, almost hidden by the overhang of the top of the PET cabinet. The slide switch is connected to the **Socket 2 ME** by a special cable (also supplied) . . . and you're up and running.

Up and running; installation took only a minute or so. Flip the switch from Toolkit to Word Processor. And back. No need to open the PET.

Complete with the first-rate installation and operating instructions you've come to expect from all Skyles documentation.

YOU HAVE AN ORIGINAL 2001-8 PET?

No problem. The **Socket 2 ME** interfaces with the BASIC Programmer's Toolkit model TK 160E or TK 160S connector board, the Word Processor II interfaces with the **Socket 2 ME**, the slide switch is placed on the PET base. Then, as long as the PET 2001-8 has at least 8K of memory expansion, the system is up and running.

YOU HAVE A COMPUTHINK DISK AND YOU WANT A BASIC PROGRAMMER'S TOOLKIT?

How would you like to switch between the Computhink and the Toolkit with a single **SYS** *command?*

Just add two small jumpers to the Computhink system and a short program to the DOS diskette. Plug in the BASIC Toolkit **TK 80E**, enter the **SYS** command and your system is up and running.

NO ROOM ON THE *PET 2 COMPUTHINK* DISK BOARD?

All your sockets are booked? Fret not; Skyles comes to the rescue. **Skyles Electric Works** now has a modified EPROM board available with sockets for the Toolkit and Computhink ROM chips. Plug in the ROMs, add a jumper (supplied) to the PET 2 Computhink disk board, plug the new EPROM board into the Computhink disk board. Power up and enter a short switching program into the DOS diskette. Switch between the Computhink disk and the Toolkit with a single **SYS** command.

ORDER NOW — with Skyles' 10-day money-back guarantee:

Socket 2 ME: $$22.50*
Commodore Word Processor II: $100.00*
Commodore Word Processor III: $200.00*

BASIC Programmer's Toolkit
Model TK 80ED $85.00*
Model TK 160ED $95.00*†

**Add $2.50 to each for shipping and handling.*

†Note: If Computhink EPROM board is returned, after purchase of TK 160ED, Skyles will refund $20.00.

"Socket 2 ME" is the trademark of Skyles Electric Works.

California residents: please add 6% or 6.5% sales tax as required

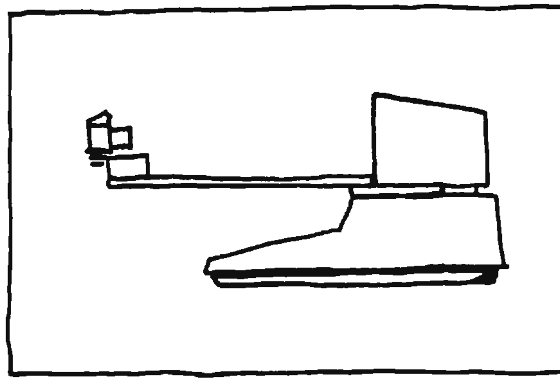
VISA, MASTERCHARGE ORDERS CALL (800) 538-3083 (except California residents)
CALIFORNIA ORDERS PLEASE CALL (408) 257-9140

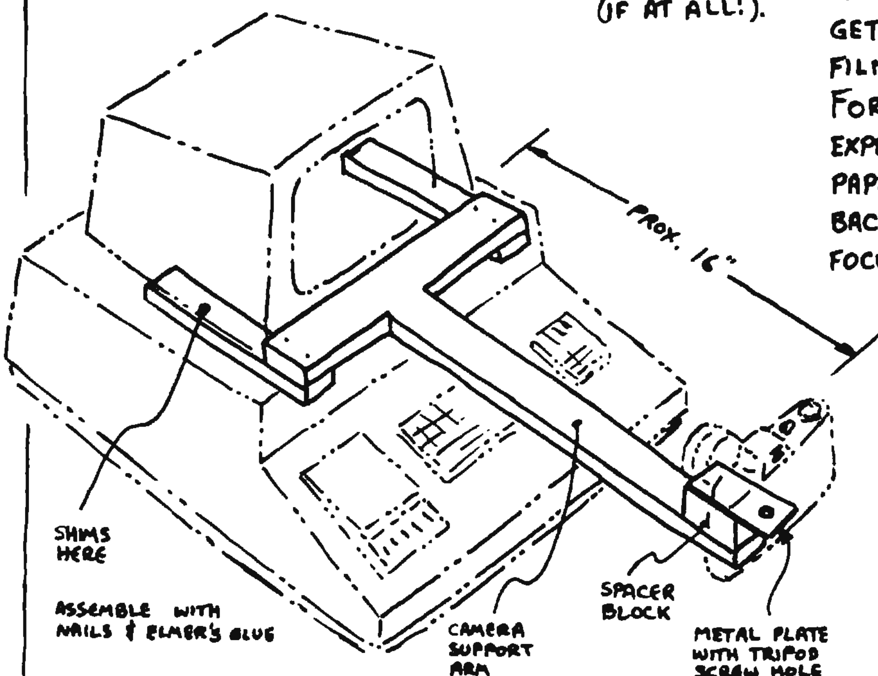
Skyles Electric Works
231 E South Whisman Road
Mountain View, CA 94041
(415) 965-1735

RAMBLIN'

HARD COPY WITHOUT A PRINTER...

EVEN IF YOU'RE NOT A PHOTOGRAPHY BUFF, YOU CAN MAKE COPIES OF THE SCREEN CHEAPLY & EASILY WITH A CAMERA, AND IT'S NOT AS IMPRACTICAL AS IT SEEMS AT FIRST. IT COSTS ABOUT A BUCK TO GET A 24- OR 36-EXPOSURE ROLL OF NEGATIVES DEVELOPED, (LOOK FOR A PLACE WITH NEXT-DAY SERVICE), AND YOU CAN READ 'EM VIA A SLIDE PROJECTOR (IF YOU HAVE ONE), OR A HAND VIEWER (ANOTHER BUCK). IT'S CONVENIENT TO MOUNT THE NEGATIVES IN 2×2 SLIDE MOUNTS (AVAILABLE AT ANY PHOTO STORE). WHAT YOU SEES IS WHAT YOU GETS; A GREAT ADVANTAGE OVER MOST PRINTERS, WHICH DON'T HANDLE THE GRAPHICS & CURSOR CONTROLS WELL- (IF AT ALL!).





ALMOST ANY CAMERA CAN BE MADE TO WORK, BUT A 35 MM SINGLE-LENS-REFLEX (SLR) MAKES THINGS VASTLY EASIER. A NO. 2 CLOSEUP LENS WILL GET YOU CLOSE ENOUGH TO FILL THE FILM AREA WITH THE SCREEN IMAGE. FOR SIMPLE CAMERAS & POLAROID, EXPERIMENT WITH A PIECE OF TRACING PAPER TAPED TO THE OPEN CAMERA BACK TO GET THE RIGHT DISTANCE & FOCUS.

ONCE YOU'VE GOT YOUR DIMENSIONS WORKED OUT, MAKE A FRAME (5/8" PLYWOOD IS FINE), WHICH SLIPS ON & OFF EASILY. IF 5/8" IS TOO SLOPPY, GLUE CARDBOARD &/OR FELT SHIMS TO THE ARMS WHICH GO BETWEEN THE TV & COMPUTER CABINETS. THE SPACER BLOCK HEIGHT WILL DEPEND ON YOUR CAMERA, BUT SHOULD PUT THE LENS ABOUT 3½" ABOVE THE CAMERA SUPPORT ARM.

EXPOSURE– SET THE SCREEN BRIGHTNESS CONTROL AS DIM AS POSSIBLE (FOR MAXIMUM SHARPNESS), AND USE A SLOW SHUTTER SPEED TO AVOID RASTER SYNCH PROBLEMS. IF YOUR CAMERA HAS AN INTERNAL LIGHT METER, IT WILL OVEREXPOSE, SINCE THE METER TRIES FOR AN AVERAGE DENSITY, OR 50%, AND MOST SCREEN IMAGES ARE 95+% BLACK. USING PLUS-X FILM (A.S.A. 125), TRY ONE SECOND AT f/11; USE A CABLE RELEASE. IF THE CHARACTERS TURN OUT DENSE & FUZZY, THEY'RE OVEREXPOSED; ADJUST ACCORDINGLY. TO USE AN INTERNAL METER, RUN THIS PROGRAM (MY FAVORITE ONE-LINER):

```
1 POKE 32768+1000*RND(1),256*RND(1): GO TO 1
```

THE SCREEN WILL FILL TO 50% BRIGHTNESS; TAKE YOUR READING, AND USE THIS EXPOSURE FOR ALL FUTURE SHOTS.

AN EARLIER COLUMN ABOUT THE CB2 LINE GENERATED SOME MAIL ABOUT MAKING MUSIC ON THE PET. SEND A SASE TO THE ADDRESS BELOW FOR A LISTING OF THE NEIL HARRIS-JIM BUTTERFIELD PROGRAM AND KEYBOARD OVERLAY WHICH WILL TURN PET INTO AN ELECTRONIC ORGAN WITH RECORD/PLAYBACK CAPABILITY.

ROY O'BRIEN, BOX 426, BEAUMONT, CA 92223

CHEEP PRINT Hard Copy For Soft Cost

C A McCarthy
1359 W Idaho Ave
St Paul, MN 55108

CHEEP PRINT SOFTWARE

Cheep Print for PET is written mostly in BASIC; a little bit of machine code is used to insure accuracy in timing the output signals. I have chosen to put the machine code in the second cassette buffer, but the code is completely relocatable and may, for instance, be placed in high memory if you change the top-of-memory pointer and command CLR before trying to run the program (neglecting CLR causes strings to overwrite the machine code; you can guess how I found this out.)

The BASIC part of Cheep Print is designed to live at the beginning of some other program, which I will refer to as the host program, and serves both to list the host program while it is under development and as data output software after the program has been completed. The host program is assumed to have line numbers of 100 and up. I have tried to structure Cheep Print so that it contains subroutines that can be called by the host program. If you look over the listing of Cheep Print, which was produced by Cheep Print itself -- the comments were added afterwards for documentation -- you will see that most of the program is taken up with formatting the output for the printer, and with the details of listing a program.

To use Cheep Print, start by making a master copy of the program that contains all the options that you might ever need; this copy should be saved for future use. Then make a sub-master that has the various parameters and options for the printer that you will be using most. You can simply load the master and delete those options that aren't needed, incidentally gaining more space for the host program. You will probably want to use Cheep Print to list itself a few times, just to make sure that the formatting is satisfactory, and to gain familiarity with the sequence of operations required. When you are about to start developing a new program, enter the sub-master copy of Cheep Print and then key in the host program. As you save your host program, Cheep Print will come along with it. Line 1 of Cheep print is a GOTO to the entry point of your host program, so that a simple RUN command will execute the host program. If you want a listing of any part of the program, simply command RUN 10; you will be prompted to check that your cassette recorder is properly set up -- cables connected and running in record mode -- and then you will be asked which lines you want listed. The options are the same as for the screen listing on PET: hit RETURN only, and get everything; enter A-B (a hyphen must be used as a separator here) and you get lines A through B inclusive; A- gives everything from A onwards, B- gives everything up through B; and A alone, with no hyphen, lists the single line A. As the program listing proceeds, the line that is currently being processed for the Cheep Print output is listed at the top of the screen. When the message "BREAK IN 80" appears, the listing is done and you can turn off the recorder.

One of the Cheep Print listing options is to put each statement on its own private print line, thus leaving much welcome space for comments. I have used this style for the listing of Cheep Print; I hope that the comments on this listing suffice to explain what is going on, and how to structure the various options for your specific needs. There is, however, one point that must be emphasized. The LIST instruction of line 2 needs to have precisely five characters as argument. The first time that you enter Cheep Print from the keyboard, key in something like 2 LISTLINES or 2 LIST12345. This is important because eventually line numbers are going to be POKED here by the commands of line 19; we need to leave enough room for the longest possible line numbers of five digits.

The other principal subroutines of Cheep Print, which the host program may find useful, are:

3-4 contains identification information such as the name of the program, date, revision, etc.; I have found this useful also when SAVEing or VERIFYing a program with PET's internal cassette recorder. The command LIST 3 puts this data on the screen, then I type the word "SAVE" and two spaces over the line number and S$-; hit RETURN, and the program gets saved with a title, without the fuss of entering the title by hand.

7 outputs the string P$, and 8 outputs the single character which has ASCII code P.

39 effects a carriage return and linefeed, and updates the variables PX and PY which give the current print position at the printer.

40-41 checks whether it's time for the printer to go to the next page, and if so, it executes 42-44.

42 linefeeds to the end of the current page,

43 is an option, useful for continuous roll paper; if the initial "RETURN" is deleted, a line of hyphens is printed at regular intervals, so your paper can be cut neatly into single sheets of uniform size. With fan-fold paper, line 43 may be ignored.

44 linefeeds to the top margin of where you want to start printing on the paper.

45-49 format the string S$ for output: space from the left edge of the paper to the left margin, print S$ if it fits on the current line; if it doesn't fit, output as much as does fit, and then repeat the process with the portion of S$ that's left over.

70 is a general purpose start-up routine. It calls

50 to start outputting MARKS, pokes the lower case mode for PET screen output, and then calls 52 for the prompt to check that your cassette is running and 54 to initialize variables and the details of paper size and format that you wish to use.

71 simply outputs the current S$, then carriage returns and, if required, linefeeds to the next page.

80 cleans up: linefeed to a blank page so that garbage created by turning off the audio doesn't spoil the beautiful page that Cheep Print has just produced, then ding the bell on the printer to announce that it's done, followed by a wait loop for a little trailer on the tape. Then the shift register at 59467 is reset so that the PET tape read and write will operate properly and the graphics mode for screen output is restored.

98 converts the hex string Q(uery)$ into the equivalent number A(nswer).

99 loads (hex machine) C(ode)$ into memory locations beginning at MA.

Lines 5 and 6 serve only as an index to the program lines that structure Cheep Print for the printer you use. I have included the most common options in the listing.

Line 23: with the REM, we list the program with extra space for comments. Delete the REM, and the program is listed single spaced, with the print output strung across the whole page. This is useful during development; not only does it execute more quickly, but it is an advantage to have as much as possible of the program in view at once, provided that the details of what is supposed to happen are fresh in your mind.

Lines 50 and 63-66 are involved in setting the baud rate and whether originate or answer mode. In line 50, only the mode needs to be determined, and the two possible choices are snuck in following the REM. Lines 63-66 contain that portion of the machine code which depends upon the baud rate and the mode. The four options given are receive and originate mode at 110 and 300 baud (10 and 30 characters per second), and which one you use is determined by which of the initial REM's isn't there. Table 2 gives the details of getting output at different frequencies and baud rates. The timing parameters MF and SF determine the mark frequency and the space frequency appropriately for you to be in originate or answer mode; the decimal equivalent of MF also needs to be reflected in the initialization at line 50. Usually you will want to be in answer mode when talking to a terminal and in originate mode when talking to a mainframe. The baud rate is determined by the timing loop parameters XR and YR, and the duration of the stop bits by SB. I did some experimenting with these parameters and found that the baud rate could be off by 7 or 8 percent and the mark and space frequencies off by 4 percent and still get solid copy; SB only needs to be sufficiently long. The output can be structured in any way you wish, of course; for instance, you could generate tapes to Kansas City standards if you wish. The most important thing is that your cassette recorder should run at the same speed at home as it does at your printer, but it doesn't matter what the speed is. If you are forced to use battery operation at your printer, then you should use battery operation at home or else experiment with the timing parameters to compensate for the speed difference that many recorders exhibit between battery and AC line operation.

Lines 85-86 contain the formatting data for the two styles of listing, and lines 87-88 contain two types of carriage return- line feed sequences. These are implemented by LISTing 85-88 on the screen, deleting the line number and REM from the data you wish to use, and then hitting RETURN; the PET screen editor picks up the line number 55 or 56 that remains and inserts these statements into the program. Lines 85-88 remain unchanged for future use. The carriage return-line feed sequence that you need will depend on the printer you use. I've used a DECwriter and a thermal printer in which carriage return (Control-M) does not also give a line feed (Control-J), and a Selectric which includes the line feed with the carriage return whether you like it or not. Further, the DECwriter requires no nulls to waste time while the print head returns to the left side of the paper, the thermal printer sometimes requires as many as four nulls, and the Selectric wants a goodly number of nulls. The Anderson-Jackson Selectric has an 800 character buffer to deal with transmission faster than the print mechanism can keep up with, and the nulls plus the program execution time seem to be enough to keep the buffer from overflowing. If you have a Selectric, one solution is to make every other character a null, which is included in the REM of line 8; if you use these nulls, declarations of P will have to go inside the FOR-NEXT loops of lines 44, 47, 53, 22, and 17. I doubt that the line 8 nulls are a good solution for a Selectric, because there is a mechanical clutch that continually engages and disengages, and probably wears more than it needs to. This clutch is also the reason that the listing output is generated line by line, rather than simply printing individual characters when ready;

As an example of using Cheep Print to serve a host program, I wrote a quick 6502 disassembler in Basic so that Cheep Print could list its own machine code for commenting. The disassembler isn't included here because there's nothing particularly exciting or instructive about it. The disassembly shows that the machine code is only a simple software timing loop, performed with interrupts inhibited (this is important!) so that the timing doesn't get messed up. The machine code outputs the start bit, and then the contents of memory location 1023/03FF low order bit first. All the printers I know ignore the highest order data bit, so I let programming convenience override generality, and output the highest order bit always as a mark, followed by sufficient marks to act as stop bits and marks until the next data come along.

Table 2 Output parameters

A. The frequency output by the 6522 shift register is given by the formula

$$\text{Frequency} = \frac{\text{System clock frequency}}{4 * (\text{contents of T2 low} + 2)}$$

assuming that the shift register contains 01010101 or 10101010.

The PET system clock appears to be spot on 1 MHz. and we calculate

For frequency	use T2 low
2225 Hz	6E hex = 110 decimal answer mark
2025 Hz	79 hex = 121 decimal answer space
1270 Hz	C3 hex = 195 decimal originate mark
1070 Hz	E8 hex = 232 decimal originate space

If desired, frequencies may be halved by using 00110011 in the shift register, and divided by four by using 00001111.

Depending on the equipment used, a certain amount of inaccuracy can be tolerated. I found no equipment that would give errors in the following ranges:

69 to 71 instead of 6E	B5 to D3 instead of C3
76 to 80 instead of 79	DA to F8 instead of E8

B. The timing loop as given by the present machine code yields

$$\text{Bit time} = 1284\ XR + YR - 1257$$

clock cycles between successive loads of 6522 T2 low, give or take one cycle due to the difference in execution times between mark and space caused by the branch at 0347. With the PET's one microsecond clock period, we have

$$XR = \text{INT}\left(\frac{\text{Bit time} + 1257}{1284}\right),$$

$$YR = \text{Bit time} + 1257 - 1284\ XR$$

For baud rate	one data bit time	XR	YR	SB
110	9091 us.	08	OF (= 15)	16 (= 22)
300	3333 us.	03	93 (= 147)	07

4% error seems to be tolerable. For 110 baud, we can range from XR,YR = 07,C0 to 08,60; for 300 baud we can range from 03,70 to 03,B8.

```
1 GOTO 1000
2 LIST   99
3 S$="8.03 #14: CHEEP PRINT MASTER WITH ¬
      ¬DISASSEMBLER"
4 RETURN
5 LINE FUNCTION: 23 LIST STYLE: 43 ROLL ¬
      ¬PAPER:50,63-66 BAUD RATE
6 LINE FUNCTION: 55 PAGE FORMAT FROM ¬
      ¬85-86: 56 CRLF FROM 87-88
7 P=LEN(P$):IF P THEN FOR P0=1 TO P:
      ¬P=ASC(MID$(P$,P0,1)):GOSUB8:
      ¬NEXT P0:RETURN
8 POKE 1023,P:SYS(M0):RETURN:REM POKE ¬
      ¬1023,0:SYS(M0):RETURN:REM FILL ¬
      ¬NULLS
10 GOSUB 70:GOSUB 3:PRINT S$:INPUT ¬
      ¬"LIST___<<<";A$:GOSUB 58:GOSUB 60:
      ¬GOSUB 68
11 PY=PV-5:GOSUB41:GOSUB3:GOSUB71:
      ¬GOSUB39:S$="LIST "+A$:GOSUB71:
      ¬GOSUB39
12 A$=" "+A$+" ":A=0:B=0:F=1:FOR J=2 TO ¬
      ¬LEN(A$)
13 IF ASC(MID$(A$,J))=45 THEN A=VAL(LEFT
      ¬$(A$,J-1)):B=VAL(MID$(A$,J+1)):F=0
14 NEXT:IF F THEN A=VAL(A$):B=A
15 B=B-63999*(B=0):K=1025:K$="GOTO21"+CH
      ¬R$(13)
16 N=FNK(K+2):K=FNK(K):IF (K=0)OR(N>B) ¬
      ¬GOTO 80
17 IF N<A GOTO 16
```



```
18 GOSUB 40:S$=RIGHT$("    "+STR$(N),5):
      ¬FOR J=1 TO 5
19 POKE FNK(1025)+4+J,ASC(MID$(S$,J)):
      ¬NEXT:PRINT"ñ"
20 FOR J=1 TO LEN(K$):POKE 526+J,
      ¬ASC(MID$(K$,J)):NEXT:POKE 525,
      ¬LEN(K$):GOTO 2
21 Y=32848+LEN(STR$(N)):Z=32647+40*(PEEK
      ¬(245)-(FNK(K)=0))
22 IF PEEK(Z)=32 THEN Z=Z-1:GOTO 22
23 REM FOR J=Y TO Z:Q=PEEK(J):GOSUB 32:
      ¬S$=S$+A$:NEXT:GOSUB 71:GOTO 16
24 F=1:C=0:FOR J=Y TO Z:Q=PEEK(J):
      ¬F=(F+(Q=34))AND1:C=C-(F*Q=58):NEXT
25 IF PY+C>PV-PB THEN GOSUB 41
26 F=1:FOR J=Y TO Z:Q=PEEK(J):F=(F+(Q=34
      ¬))AND1
27 IF Q*F=58 THEN GOSUB 45:GOSUB 46:
      ¬GOSUB 39:S$="     :":GOTO 30
28 GOSUB 32:IF PL+LEN(S$)+LEN(A$) >= ¬
      ¬PH-PR THEN GOSUB 45:GOSUB 46:
      ¬GOSUB 39:S$="      "
29 S$=S$+A$
30 NEXT J:GOSUB 45:GOSUB 46:GOSUB 39:
      ¬GOSUB 39:GOSUB 40:GOTO 16
32 A$="":IF Q>127 THEN A$=CHR$(114):
      ¬Q=Q-128
33 IF Q>63 THEN A$=A$+CHR$(115):Q=Q-64
34 A$=A$+CHR$(Q-64*(Q<32)):RETURN
39 P$=R$:GOSUB 7:PX=1:PY=PY+1:RETURN
40 IF PY<PV-PB THEN RETURN
41 GOSUB 42:GOSUB 43:GOTO 44
42 FOR PY=PY-PV TO 1:P$=R$:GOSUB 7:NEXT:
      ¬PX=1:RETURN
43 RETURN:P=45:FOR PX=PX TO PH:GOSUB 8:
      ¬NEXT:RETURN:REM PAGE SEPARATOR
44 FOR PY=PY TO PT:P$=R$:GOSUB 7:NEXT:
      ¬PX=1:RETURN
45 IF PX<PL THEN FOR PX=PX TO PL:P=32:
      ¬GOSUB 8:NEXT:RETURN
46 PQ=PH-PR-PX:IF LEN(S$) <= PQ THEN ¬
      ¬PX=PX+LEN(S$):P$=S$:GOTO 7
47 IF PQ=0 THEN GOSUB 39:GOSUB 45:
      ¬GOTO 46
48 F=PQ:FOR J1=1 TO PQ:IF 32=ASC(MID$(S$
      ¬,J1)) THEN F=J1
49 S1$=MID$(S$,F+1):S$=LEFT$(S$,F):
      ¬GOSUB 46:GOSUB 39:S$=S1$:GOSUB 45:
      ¬GOTO 46
```

```
50 POKE59467,16:POKE59464,110:POKE59466,
     ¬85:RETURN:REM 59464¯195=ORG;
     ¬110=ANS
52 PRINT "↓↓↓WHEN READY, HIT THE RETURN ¬
     ¬KEY"
53 INPUT "↑↑↑CHECK!  DEVICE READY    ←←←"
     ¬;A$:PRINT "↓↓OK":RETURN
54 S$="":J=0:P=0:P$="":Q=0:Q$="":A=0
55 PX=0:PY=1:PH=79:PV=66:PL=12:PR= 3:
     ¬PT=6:PB=6:REM FOR COMPACT LIST
56 R$=CHR$(0):FOR J=1 TO 4:R$=R$+R$:
     ¬NEXT:R$=CHR$(13)+R$:RETURN:REM IBM
58 IF ASC(A$)=160 THEN A$=MID$(A$+" ",
     ¬2):GOTO 58
59 RETURN
60 DEFFND(X)=X-48+7*(X>64)
61 DEFFNK(X)=256*PEEK(X+1)+PEEK(X)
62 MA=826:M0=MA:C$="780EFF0338B014A2":
     ¬GOSUB99
63 REM:C$="08A00FA9E89002A9C38D48E888D0F
     ¬DCAD0FA186EFF03D0E79004A216":
     ¬REM OR110
64 REM:C$="08A00FA9799002A96E8D48E888D0F
     ¬DCAD0FA186EFF03D0E79004A216":
     ¬REM AN110
65 REM:C$="03A093A9E89002A9C38D48E888D0F
     ¬DCAD0FA186EFF03D0E79004A207":
     ¬REM OR300
66 C$="03A093A9799002A96E8D48E888D0FDCAD
     ¬0FA186EFF03D0E79004A207":REM AN300
67 GOSUB99:C$="D0E95860":GOSUB99:RETURN
68 FOR J=1 TO 8000:NEXT J:RETURN
70 GOSUB 50:POKE 59468,14:GOSUB 52:
     ¬GOTO 54
71 GOSUB 45:GOSUB 46:GOSUB 39:GOTO 40
80 GOSUB 41:P=7:GOSUB 8:GOSUB 68:
     ¬POKE 59467,0:POKE 59468,12:STOP
85 REM 55 PX=0:PY=1:PH=79:PV=66:PL=12:
     ¬PR=41:PT=8:PB=8:REM FOR COMMENTED ¬
     ¬LIST
86 REM 55 PX=0:PY=1:PH=79:PV=66:PL=12:
     ¬PR= 3:PT=6:PB=6:REM FOR COMPACT ¬
     ¬LIST
87 REM 56 R$=CHR$(0):R$=CHR$(13)+CHR$(10
     ¬)+R$+R$+R$+R$:RETURN:REM DEC CRLF
88 REM 56 R$=CHR$(0):FOR J=1 TO 4:
     ¬R$=R$+R$:NEXT:R$=CHR$(13)+R$:
     ¬RETURN:REM IBM
98 A=0:FOR J=1 TO LEN(Q$):A=16*A+FND(ASC
     ¬(MID$(Q$,J))):NEXT:RETURN
99 FOR I=1 TO LEN(C$) STEP 2:Q$=MID$(C$,
     ¬I,2):GOSUB 98:POKE MA,A:MA=MA+1:
     ¬NEXT I:RETURN
```

©

Relocate PET Monitor Almost Anywhere

Roy Busdiecker
Micro Software Systems
P.O. Box 1442
Woodbridge, VA 22193

The Monitor program provided on tape to owners of PETs with the old ROMs occupies locations (in hex) $0400 to $076B, right where the PET stores BASIC programs! If you're only interested in working with 6502 machine language routines, that location for the Monitor poses no problem.

Unfortunately for me, I wanted to have the Monitor available at the same time as BASIC programs. Despite a variety of attempts, I was unable to find a combination that would allow me to enter and run BASIC programs while having the Monitor available in its original location.

On the other hand, if the Monitor could be moved somewhere very near the highest available memory location with which the PET is equipped, then the end-of-memory pointer could be reset to protect the Monitor. Then a BASIC program could be run, and it would not interfere with the Monitor.

My problem appeared to be solved when I found an article by Professor Harvey B. Herman of the University of North Carolina ("Move It", MICRO, September 1979). Although a critical figure seemed to be missing from the article, it was not too difficult to puzzle out, since the technique was based on an article by Jim Butterfield in the PET User Notes (Issue 1, Vol. 2). Incidentally, Jim has been a key figure in unravelling the secrets of the PET, both publishing information himself and pointing others in the right direction.

Although a relocation technique is summarized in this article, the discovery that got me excited was a simple but powerful extension to make it do many relocations, after the first one was completed.

Why would anyone want to relocate the Monitor to many different locations? The answer lies in the earlier observation that it would be desirable to locate it near the top end of available memory. With a 4k PET, a good location for the Monitor would be starting at the 3k location. On an 8k PET, you would want to locate it from 7k up. If a 4k PET had an 8k memory extension attached, giving a total of 12k, the Monitor should be located at the 11k point. Other combinations of PETs and external memories would lead us to want Monitors located at 15k, 19k, 23k, 27k and 31k.

With the approach described here, all of those Monitors can be created in little more time than it takes to SAVE each one.

First, we must create one working Monitor relocated to a position where it does not conflict with the original. Here's one way to do it:

1. LOAD the Monitor (Print STATUS; PEEK (630) to insure that the LOAD was good ... PET should print 0 0). Rewind the tape.
2. Type SYS62894, then press RETURN. You will get the message PRESS PLAY ON CASSETTE #1. Insert the Monitor tape tape and press PLAY. The tape will run until the FOUND MONITOR message appears, then it will report READY. At this point the tape header has been found and loaded into locations 634 through 638, but the Monitor itself has not started loading. Here is the meaning of bytes 635 through 638.

635	636	637	638
lo byte	hi byte	lo byte	hi byte
First location that program tape will load into.		Last location that program tape will load into.	

3. By changing the values in these locations, we can make this second copy load into a different area of memory. For this exercise, let's put it into locations 3k (decimal 3072, hex 0C00) and up. To do that,
   POKE 636, 12:POKE 638,15:SYS 62403
   The tape will run and PET will report LOADING until the Monitor is in. Rewind the tape and set it aside. At this point, you will have two identical copies of the Monitor. One is located from $04000 to $076A ($ indicates a hex value), as shown in Figure 1. The other copy is in locations $0C00 to $0C6A (see Figure 2 for the first 128 locations). The second copy, of course, will not work because its addresses point to the wrong locations.
4. In order to make the $0C00-0C6A copy of the Monitor usable, you must alter (patch) the contents of 84 memory locations. There are several ways to do this, but the straight-forward approach I used (outlined below) is about as easy as any other.
   a. Type SYS 1039 and press RETURN. This transfers control to the $0400-$076B Monitor, which will display
      PC SR AC XY YR SP
      C6ED 30 33 39 00 FE
   b. Consult Figure 1 to see which values are to be changed (the circled numbers). A circled value in line 0410 calls for a change in the corresponding position in line 0C10. Line 0500 corresponds to 0D10, 0600 to 0E00, and 0700 to

0F00. If the circled number is 04 in the original Monitor, that value in the copy should be changed to 0C. Likewise, 05 changes to 0D, 06 to 0E, and 07 to 0F. Figure 3 shows the first 128 locations in the copy after changes have been made.

c. To make the changes, list a section of copy with the M command, as in

M 0C00 0C7F

which prints the first 128 bytes in 16 rows. Next, use the cursor controls to position the cursor over the value to be changed, type the corrected value, and press RETURN. Even if several values in a line are changed, you only need to press RETURN once per line. Repeat the procedure until all 84 changes are made.

d. For insurance, save a copy of your new 3k Monitor before testing it.

S 01,3K MONITOR 3087,0C00,0F6B

is the command to use. To be on the safe side, make several copies on the tape. Rewind this tape, mark it "3k Monitor", and set it aside.

5. Now test your 3k Monitor, to make sure it works correctly. Type X, then RETURN to get out of the original Monitor. Type SYS 3087, then RETURN, to transfer control to the 3k Monitor. Try out the features described for the Monitor in your manual.

When you are confident that you have created a working Monitor at the 3k point, you are ready to make as many working copies in other locations as you have memory to accommodate. If you have a full 32k, you may use all the procedures below exactly as written. Otherwise, just delete (or ignore) references to locations beyond the end of your RAM. Here is the procedure:

1. LOAD the 3k Monitor. Rewind and remove the cassette.
2. Put the original Monitor tape in the cassette unit, and do step 2 of the earlier process.
3. POKE 636,28:POKE 638,31:SYS 62403. This puts a copy of the original Monitor in memory starting at 7168 ($1C00).
4. POKE 135,12. This tells your PET that end-of-memory is just below 3k, so the BASIC program described below will not mess up your Monitor.
5. Type NEW and press RETURN. This is very important!
6. Type in and run the following program, which will create working copies of the Monitor at the high end of every 4k increment of memory.

MONITOR MULTIPLE RELOCATOR

```
10 REM   MONITOR MULTIPLE RELOCATOR
20 REM   COPYRIGHT NOV 1979 BY
30 REM      ROY BUSDIECKER
40 REM      MICRO SOFTWARE SYSTEMS
50 REM      P.O. BOX 1442
60 REM      WOODBRIDGE, VA 22193
70 REM   ALL RIGHTS RESERVED
80
90 X=6
100 K=1024:M1=3072:M2=7168:K4=4*K
110 FORI=0TO875
120 C1=PEEK(M1+I):C2=PEEK(M2+I)
130 MM=0:IFC1<>C2THENMM=1:NC=NC+1
140 FORJ=0TOX:POKEM2+I+J*K4,C1+MM*(J+1)*16:NEXTJ,I
150 PRINT"THERE WERE"NC"CHANGES."
```

In line 90, X = 6 sets the program for 32k memory. For each 4k less than 32k in your machine, subtract 1 (for a 16k machine, line 90 should be X = 2). After the program has finished, you will have X + 1 new working copies of the Monitor. My PET took just under two minutes to do the computations for seven copies. Figure 4 shows the corresponding line from several copies of the Monitor before the run, while figure 5 shows them afterwards. Note the changes made by the program!

All you have to do now is to SAVE the new Monitors the same way you saved the 3k Monitor (step 4d of the first procedure). The table below gives the values to use in the SAVE command (use SYS 3087 to get into the Monitor), and the SYS numbers to use the other copies of the Monitor.

MONITOR LOCATIONS (for SAVE)

VERSION	START	END	SYS	POKE 135,
3k	0C00	0f6B	3087	12
7k	1C00	1F6B	7183	28
11k	2C00	2F6B	11279	44
15K	3C00	3F6B	15375	60
19k	4C00	4F6B	19471	76
23k	5C00	5F6B	23567	92
27k	6C00	6F6B	27663	108
31k	7C00	7F6B	31759	124

When you LOAD any one of the Monitors later on to use with BASIC programs, remember to POKE 135,__ with the number shown in the table, then do a NEW before LOADing or typing in your program.

```
      0  1  2  3  4  5  6  7
0400 00 0D 04 0A 00 9E 28 31
0408 30 33 39 29 00 00 00 A9
0410 27 8D 1B 02 A9 04 8D 1C
0418 02 A9 07 85 7D A9 EB 85
0420 7C A2 43 38 21 D0 12 A9
0428 42 85 21 D8 4A 68 85 1E
0430 68 85 1D 68 85 1C 68 85
0438 1B 68 69 FF 85 19 68 69
0440 FF 85 1A BA 86 1F 58 20
0448 F2 04 A6 21 A0 2A 20 22
0450 05 A9 52 85 0D D0 2B A9
0458 00 85 CA 85 0D 85 0A 20
0460 F2 04 A9 2E 20 D2 FF A6
0468 20 E0 02 F0 04 E0 03 D0
0470 05 20 3A 06 20 37 06 20
0478 90 05 C9 2E F0 F9 C9 20

0480 F0 F5 A2 07 DD 02 05 D0
0488 0F A5 20 85 0E 8E 20 BD
0490 0A 05 48 BD 12 05 48 60
0498 CA 10 E9 A9 3F 20 D2 FF
04A0 4C 57 04 38 A5 13 E5 11
04A8 85 0B A5 14 E5 12 A8 05
04B0 0B 60 A5 11 85 19 A5 12
04B8 85 1A 60 85 21 A0 00 20
04C0 3A 06 B1 11 20 13 06 20
04C8 F7 04 C6 21 D0 F1 60 20
04D0 5E 06 90 0D A2 00 81 11
04D8 C1 11 F0 05 68 68 4C 9B
04E0 04 20 F7 04 C6 21 60 A9
04E8 1B 85 11 A9 00 85 12 A9
04F0 05 60 A9 0D 4C D2 FF E6
04F8 11 D0 06 E6 12 D0 02 E6
```

Figure 1a. Monitor in original location, showing bytes to be changed (part1)

```
      0  1  2  3  4  5  6  7
0500 0A 60 3A 3B 52 4D 47 53
0508 4C 53[05][05][05][05][05][05]
0510[05][05]C1 B1 2C 5E D7 FD
0518 9E 9E 20 50 43 20 20 53
0520 52 20 41 43 20 58 52 20
0528 59 52 20 53 50 A5 0D D0
0530 06 20 F2[04]20 37[05]20
0538 37[06]A2 00 BD 1A[05]20
0540 D2 FF E8 E0 13 D0 F5 20
0548 F2[04]A2 2E A9 3B 20 22
0550[06]20 37[06]20 08[06]20
0558 E7[04]20 EB[04]F0 4D 20
0560 90[05]20 4F[06]90 48 20
0568 3F[06]20 90[06]20 4F[06]
0570 90 3D 20 3F[06]A0 00 B9
0578 4A[07]30 06 20 D2 FF C8
0580 D0 F5 29 7F 20 D2 FF 20
0588 2A F3 F0 20 A6 0A D0 1C
0590 20 A3[04]90 17 20 F2[04]
0598 A2 2E A9 3A 20 22[05]20
05A0 37[06]20 04[06]A9 08 20
05A8 BB[04]F0 DB 4C 57[04]4C
05B0 9B[04]29 5E[06]20 4F[06]
05B8 90 03 20 E2[04]20 E7[04]
05C0 D0 0A 20 5E[06]20 4F[06]
05C8 90 E5 A9 08 85 21 20 90
05D0[06]20 CF[04]D0 F8 F0 D4
05D8 20 CF FF C9 0D F0 0C C9
05E0 20 D0 CC 20 4F[06]90 03
05E8 20 B2[04]A6 1F 9A A5 1A
05F0 48 A5 19 48 A5 1B 48 A5
05F8 1F A6 1D A4 1E 40 A6 1F
```

Figure 1b. Monitor in original location, showing bytes to be changed (part 2)

```
      0  1  2  3  4  5  6  7
0600 9A 4C 8B C3 A2 01 D0 02
0608 A2 09 B5 10 48 B5 11 20
0610 13[06]68 48 4A 4A 4A 4A
0618 20[2B][06]AA 68 29 0F 20
0620 2B[06]48 8A 20 D2 FF 68
0628 4C D2 FF 18 69 06 69 F0
0630 90 02 69 06 69 3A 60 20
0638 3A[06]A9 20 4C D2 FF A2
0640 02 B5 10 48 B5 12 95 10
0648 68 95 12 CA D0 F3 60 20
0650 5E[06]90 02 85 12 20 5E
0658[06]90 02 85 11 60 A9 00
0660 85 0F 20 90[06]C9 20 D0
0668 09 20 90[06]C9 20 D0 0E
0670 18 60 20 85[06]0A 0A 0A
0678 0A 85 0F 20 90[06]20 85
0680[06]05 0F 38 60 C9 3A 08
0688 29 0F 28 90 02 69 08 60
0690 20 CF FF C9 0D D0 F8 68
0698 68 4C 57[04]4C 9B[04]20
06A0 90[06]A9 00 85 EE 85 FA
06A8 A9 23 85 F9 20 5E[06]29
06B0 0F 85 F1 20 90[06]A2 00
06B8 20 CF FF C9 2C F0 55 C9
06C0 0D F0 0B E0 10 F0 F1 95
06C8 23 E6 EE E8 D0 EA A5 20
06D0 C9 06 D0 C8 A2 00 8E 0B
06D8 02 A5 F1 D0 03 4C 9B[04]
06E0 C9 03 B0 F9 20 67 F6 20
06E8 3B F8 20 FF F3 A5 EE F0
06F0 08 20 95 F4 D0 08 4C 9B
06F8[04]20 AE F5 F0 F8 20 4D
```

Figure 1c. Monitor in original location, showing bytes to be changed (part 3)

```
      0  1  2  3  4  5  6  7
0700 F6 20 22 F4 20 9A F8 20
0708 13 F9 AD 0C 02 29 10 D0
0710 E5 4C 57[04]20 4F[06]A5
0718 11 85 F7 A5 12 85 F8 20
0720 CF FF C9 20 F0 F9 C9 0D
0728 F0 A4 C9 2C F0 03 4C 9C
0730[06]20 4F[06]A5 11 85 E5
0738 A5 12 85 E6 A5 20 C9 06
0740 F0 92 A2 00 20 B1 F6 4C
0748 57[04]0D 20 20 20 20 20
0750 20 20 20 20 20 30 20 20
0758 31 20 20 32 20 20 33 20
0760 20 34 20 20 35 20 20 36
0768 20 20 B7 24 24 24 24 24
```

Figure 1d. Monitor in original location, showing bytes to be changed (part 4)

```
      0  1  2  3  4  5  6  7
0C00 00 0D 04 0A 00 9E 28 31
0C08 30 33 39 29 00 00 00 A9
0C10 27 8D 1B 02 A9[04]8D 1C
0C18 02 A9[07]85 7D A9 6B 85
0C20 7C A9 43 85 21 D0 12 A9
0C28 42 85 21 D8 4A 68 85 1E
0C30 68 85 1D 68 85 1C 68 85
0C38 1B 68 69 FF 85 19 68 69
0C40 FF 85 1A BA 86 1F 58 20
0C48 F2[04]A6 21 A9 2A 20 22
0C50[06]A9 52 85 0D D0 2B A9
0C58 00 85 CA 85 0D 85 0A 20
0C60 F2[04]A9 2E 20 D2 FF A6
0C68 20 E0 02 F0 04 E0 03 D0
0C70 06 20 3A[06]20 37[06]20
0C78 90[06]C9 2E F0 F9 C9 20
0C80 F0 F5 A2 07 DD 02[05]D0
0C88 0F A5 20 85 0E 86 20 BD
0C90 0A[05]48 BD 12[05]48 60
0C98 CA 10 E9 A9 3F 20 D2 FF
0CA0 4C 57[04]38 A5 13 E5 11
0CA8 85 0B A5 14 E5 12 A8 05
0CB0 0B 60 A5 11 85 19 A5 12
0CB8 85 1A 60 85 21 A0 00 20
0CC0 3A[06]B1 11 20 13[06]20
0CC8 F7[04]C6 21 D0 F1 60 20
0CD0 5E[06]90 0D A2 00 81 11
0CD8 C1 11 F0 05 68 68 4C 9B
0CE0[04]20 F7[04]C6 21 60 A9
0CE8 1B 85 11 A9 00 85 12 A9
0CF0 05 60 A9 0D 4C D2 FF E6
0CF8 11 D0 06 E6 12 D0 02 E6
```

Figure 2. Copy of original monitor, moved to locations $0C00 - $0F6B

```
      0  1  2  3  4  5  6  7
0C00 00 0D 04 0A 00 9E 28 31
0C08 30 33 39 29 00 00 00 A9
0C10 27 8D 1B 02 A9[0C]8D 1C
0C18 02 A9[0F]85 7D A9 6B 85
0C20 7C A9 43 85 21 D0 12 A9
0C28 42 85 21 D8 4A 68 85 1E
0C30 68 85 1D 68 85 1C 68 85
0C38 1B 68 69 FF 85 19 68 69
0C40 FF 85 1A BA 86 1F 58 20
0C48 F2[0C]A6 21 A9 2A 20 22
0C50[0E]A9 52 85 0D D0 2B A9
0C58 00 85 CA 85 0D 85 0A 20
0C60 F2[0C]A9 2E 20 D2 FF A6
0C68 20 E0 02 F0 04 E0 03 D0
0C70 06 20 3A[0E]20 37[0E]20
0C78 90[0E]C9 2E F0 F9 C9 20
0C80 F0 F5 A2 07 DD 02[0D]D0
0C88 0F A5 20 85 0E 86 20 BD
0C90 0A[0D]48 BD 12[0D]48 60
0C98 CA 10 E9 A9 3F 20 D2 FF
0CA0 4C 57[0C]38 A5 13 E5 11
0CA8 85 0B A5 14 E5 12 A8 05
0CB0 0B 60 A5 11 85 19 A5 12
0CB8 85 1A 60 85 21 A0 00 20
0CC0 3A[0E]B1 11 20 13[0E]20
0CC8 F7[0C]C6 21 D0 F1 60 20
0CD0 5E[0E]90 0D A2 00 81 11
0CD8 C1 11 F0 05 68 68 4C 9B
0CE0[0C]20 F7[0C]C6 21 60 A9
0CE8 1B 85 11 A9 00 85 12 A9
0CF0 05 60 A9 0D 4C D2 FF E6
0CF8 11 D0 06 E6 12 D0 02 E6
```

Figure 3. Monitor in $0C00 - $0F6B, after corrections have been made.

```
      0  1  2  3  4  5  6  7
0F10 E5 4C 57[0C]20 4F[0E]A5
      0  1  2  3  4  5  6  7
1F10 E5 4C 57[04]20 4F[06]A5
      0  1  2  3  4  5  6  7
2F10 24 24 24 24 24 24 24 24
      0  1  2  3  4  5  6  7
3F10 24 24 24 24 24 24 24 24
      0  1  2  3  4  5  6  7
4F10 24 24 24 24 24 24 24 24
      0  1  2  3  4  5  6  7
5F10 24 24 24 24 24 24 24 24
```

Figure 4. Contents of selected locations before Monitor Multiple Relocator program is run.

```
      0  1  2  3  4  5  6  7
0F10 E5 4C 57[0C]20 4F[0E]A5
      0  1  2  3  4  5  6  7
1F10 E5 4C 57[1C]20 4F[1E]A5
      0  1  2  3  4  5  6  7
2F10 E5 4C 57[2C]20 4F[2E]A5
      0  1  2  3  4  5  6  7
3F10 E5 4C 57[3C]20 4F[3E]A5
      0  1  2  3  4  5  6  7
4F10 E5 4C 57[4C]20 4F[4E]A5
      0  1  2  3  4  5  6  7
5F10 E5 4C 57[5C]20 4F[5E]A5
```

Figure 5. Contents of selected locations after Monitor Multiple Relocator has been run.

```
ADDRESSES WHICH MUST BE MODIFIED

0C15 0C1A 0C49 0C50 0C61 0C73 0C76 0C79 0C86 0C91 0C95 0CA2 0CC1 0CC6 0CC9 0CD1
0CE0 0CE3 0D0A 0D0B 0D0C 0D0D 0D0E 0D0F 0D10 0D11 0D33 0D36 0D39 0D3E 0D49 0D50
0D53 0D56 0D59 0D5C 0D61 0D64 0D69 0D6C 0D6F 0D74 0D79 0D92 0D97 0D9E 0DA1 0DA4
0DA9 0DAE 0DB1 0DB4 0DB7 0DBC 0DBF 0DC4 0DC7 0DD0 0DD3 0DE5 0DEA 0E11 0E1A 0E21
0E39 0E51 0E58 0E64 0E6B 0E74 0E7D 0E80 0E9B 0E9E 0EA1 0EAE 0EB5 0EDF 0EF8 0F13
0F16 0F30 0F33 0F49

THEPE ARE A TOTAL OF 84 ADDRESSES TO CHANGE
```

Figure 6. Addresses of individual bytes which must be altered to correct first copy of Monitor in $0C00 - $0F6B.

©

PET PROGRAMS ON TAPE EXCHANGE

Gene Beals
Box 516
Montgomeryville, PA 18936

The "programs on tape" service and exchange functions as a low cost method for software distribution. The copying cost is $1 per program. We can put up to 4 programs (8K) on a C-10 tape, or 12 on a C-30 tape. Please add $1 per tape (either C-10 or C-30) to cover tape cost and postage within U.S. or Canada. If any written documentation is available, a copy will be included with the associated program.

If you have a program to contribute to the list (one which we don't have, or an updated or enhanced version of an existing program), please send it on tape. We will save it for the exchange and return a program of your choice.

Please write to me (Box 516, Montgomeryville, PA 18936) to request or exchange programs, or if you have any comments on the way the exchange is being conducted.

SPACE RACE - Matt Ganis -- guide your spaceship to the top of the box through randomly moving stars.
ACEY DEUCY - Matt Ganis -- you are dealt two cards and then bet whether your next card will be between the first two cards.
BOMBER - Matt Ganis -- destroy the city by dropping bombs on it. Lose points for each bomb and accumulate points for each hit.
MATH DRILL - William C. Anderson -- drill on addition, subtraction, multiplication, and division for numbers 1 through 12.
OPTI-STICKS -- combines optical illusions and graphics demo of sticks rolling down hill - nice demo.
TRACE-OLD ROM - Brett Butler -- self-relocating version
TRACE-NEW ROM - Brett Butler -- self-relocating version
LIFE 64*64 V.2 - Frank Covitz -- maintains symmetry through wrap around. Set for either growth or decay. Excellent machine language program.

Some Exchange guidelines:

1. *The Exchange is intended to promote the sharing of user generated software.*
2. *Be very careful that you submit only your own, original work to the exchange. Matters of copyright remain the sole responsibility of the individuals submitting the program. We accept no liability, express or implied. Do not submit modified (or unmodified) commercial software to the Exchange. It makes their job much harder. We make every effort to screen software; we will appreciate our readers' efforts to do the same.*
3. *We do not intend to promote the Exchange as a competitor to commercially available software. You'll find that commercial software is generally much more polished and documented than what you'll receive from us.* *Robert Lock*

FORCES - Everett Lumpkin -- Calculate resultant of vector forces
FORMAT - Elizabeth Deal -- PRINT USING subroutines w/rounding
HISTORY QUIZ - Dave Ray -- Simple CAI program
HORIZON COORD - R. Grokett Jr. -- Calculate altitudes/ azimuths of celestial objects
INDEX BOTH - R. Wagner & Wilcox -- Index program modified for all roms
LAYOUT TECHNIQUE - Jill Johnson -- CAI teaches yearbook design
NERVES - Ralph Bressler -- Game of guessing random time intervals
NEWMON ISNSTR-OLD - for 2.0ROMs Instructions for newmon program
PATTERNS - R. Grockett -- Screen flashes & cb2 beeps
PARTS OF SPEECH - Laurie Birtalan -- for 2.0 ROMs CAI grammar quiz-WAIT statement used
ACCOUNTING CAI - Janice Klages -- Simple quiz/tutor
BASEBALL TRIVIA - Amy Druschel -- Simple team name-city matching CAI
BIOCOMPAT - Joe Cannata -- Tests bi)orhythm compatibility
BLACK IV - Andy Finkel -- Updated blackjack-Vegas rules Uses up to 4 decks
CASCADE - Ralph Bressler -- Nice graphic bead dropping demo.
CIRCLE - Solves Josephus circle problems
COMPUTER HISTORY - Dave Ray -- CAI quiz on the history of computers
DIAMOND LANE - Brent Weaver -- Great video game-don't hit the maniac.
DIGIT SPAN - Ralph Bressler -- Number sequence memory test.
FLASH - Ralph Bressler -- High-speed word memory test
INDEX - David Wilcox -- for 2.0 ROMs Tape program high-speed indexing program
PARTS OF SPEECH - Laurie Birtalan -- CAI grammar quic
PCKBK - Vanderpool -- Very simple checkbook balancing program
READING TEST - Mariann McElwain -- CAI test of your comprehension
RETURN - Andy Finkel -- Keyboard dexterity reduced to simplest level.
SCREENPRINT PR40 - John Lemkelde -- Screendump for SWTPC PR40 printer
SEA QUIZ - Jackie Bober -- CAI teaches you about the sea
SUPER MASTERMIND - David C. Swaim II -- New improved version of logic game
SUPERMON REL. - Wonziak, Baum, Butterfield -- 3.0 ROMs Relocatable advanced machine language monitor
SUPERMON INSTR'S - Wozniak, Baum, Butterfield -- for 3.0 ROMs How to run supermon
TRACE-NEW - for 3.0 ROMs Trace routine for new ROMs
UNIV. ROM TEST - Jim Butterfield -- Tests out the PET's ROM chips
VOCABULARY - Everett Lumpkin -- Have the PET teach you Latin words
INTELLECTUS - General knowledge college level quiz
MP MATH - Does multiple precision arithmetic operations up to approximately 40 digits or more. Uses machine language
FN MACHINE - Number enters function machine, levers turn, lights flash, a new number appears. Next time enter new number and predict what will appear by determining mathamatical function
PIZZA-GRADE 3 - modified by Elizabeth Deal -- Well written quiz program for pre-schoolers. Must give directions over cb radio to pizza truck

Capute

Wherein we acknowledge recent goofs. . .

This page brought to you by Robert Lock, Editor/Publisher and our (sometimes hostile) but always active readers.

It seems appropriate, on this sackcloth and ashes page, to start with a new resolution: I promise to try to avoid continuation articles of the type used last issue. Charlie McCarthy's CHEEP PRINT software is in this issue. Phew. Your comments and letters are hereby officially acknowledged. Ouch. I won't do it anymore. Groan.

Perhaps the biggest blooper in Issue 3 was the accidental deletion of the last part of an OSI software listing. I can report, with some conviction, that the article was read by OSI owners. The corrected listing was printed in Issue #1 of compute II.

Corrections to Issue 3:

Utinsel, page 36.
The program line 780 printed as
780 POKE J,BL:J =J + 1
should read
780 POKE J,B1:J =J + 1

The Learning Lab, pages 52-53.
The copyright symbol shows up in various places in the listing. It should be the up-arrow (exponentiation) symbol instead.

We'll keep you posted in the unlikely event that more errors surface. ©

Program Listings for COMPUTE

Cursor control characters will appear in source listings as shown below:

```
h=HOME            , ĥ=CLEAR SCREEN
↓=DOWN CURSOR     , ↑=UP CURSOR
→=RIGHT CURSOR,    ←=LEFT CURSOR
r=REVERSE         , r̂=REVERSE OFF
```

Graphics (i.e. shifted) characters will appear as the unshifted alphanumeric character with an underline. This does not apply to the cursor control characters. The Spinwriter thimble doesn't have a backarrow symbol, so a "⁓" is used instead.

The "¬" is used to indicate the beginning of a continuation line. It is also used to indicate the end of a line which ends with a space. This prevents any spaces from being hidden. ©

COMPUTE.
The Journal For Progressive Computing.™

compute II.
The Single-Board COMPUTE.™

☐ Please enter my 1 year (6 issue) subscription to COMPUTE.
☐ Please enter my 1 year (6 issue) subscription to compute II.

My computer is: PET _____ APPLE _____ ATARI _____ OSI _____ SYM _____ KIM _____
AIM _____ Other _____ (Specify) ______________________________

Check One:
_____ $9.00 (U.S. Mail Order Subscription)
_____ $12.00 U.S. (Canadian Subscription)

☐ I want Both!
_____ $15.00 (U.S. Mail Order Subscription)
_____ $18.00 U.S. (Canadian Subscription)

My Name, Address: ______________________________
____
____
____

Charge my Visa __________ MC __________
Number ______________________ Expires _____/_____/_____

COMPUTE and compute II are publications of Small System Services Inc. 900-902 Spring Garden St
Greensboro, NC 27403 919-272-4867

The Editor's Feedback:

My computer is _____ PET/CBM _____ ATARI _____ APPLE _____ OTHER
My application is (check all that apply):
_____ Home/Personal _____ Business _____ Educational
_____ Industrial _____ Other
Comments:
New Product - Z-80 Softcard ______________________________
Others? ______________________________
Content:
Best Article This Issue (page #, title)
____
I'd like to see more ______________________________
articles in COMPUTE.
I'd like to see fewer ______________________________
articles in COMPUTE.
Other suggestions:

Place
Stamp
Here

COMPUTE./compute II
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE./compute II
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE./compute II
Post Office Box 5406
Greensboro, NC 27403

Dear MICRO-ED:
I would like further information on your educational programs

Name ______________________________

Address ____________________________

City ______________ State ______________ Zip ____

PLACE
STAMP
HERE

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

The Great American Solution Machine.

More than 50,000 students, teachers and administrators solve problems with this reliable Commodore computer.

You're looking at the Number One computer in education today.

In fact, you've probably already used it.

The Commodore.

You know it teaches. Guides. Challenges. Analyzes. Organizes. Simplifies.

But did you know it has capabilities that are far beyond its price range?

You can accomplish tasks with The Commodore at a price/performance ratio that leads the field.

You can also count on The Commodore showing up for class every day.

It's a remarkably sophisticated, remarkably reliable machine. Around the world in schools—and businesses too—there are more than 100,000 Commodore computers now at work.

If you sense a snag in the flow of knowledge in your classes, we think you should challenge The Commodore.

Compare it against any computer in—or above—its field.

See if it won't raise the level of interest and accomplishment among your students.

And simplify the complex in your administrative duties.

All at a price that makes it stand alone.

For the name of your nearest authorized Commodore dealer, just write to: Commodore Business Machines, Inc., 3330 Scott Blvd., Santa Clara, CA 95051

Call now toll-free. Ask for operator 973:

800-824-7888

(In Calif., 800-852-7777)
(In Alaska and Hawaii, 800-824-7919)

commodore

COMMODORE