Waterloo Structured BASIC For The PET

Feed Your PET Some APPLESOFT

COMPUTE.

$2.00
September/October, 1980
Vol. 1, Issue 6

The Journal for Progressive Computing™

The Resource Magazine For Apple, Atari, and Commodore

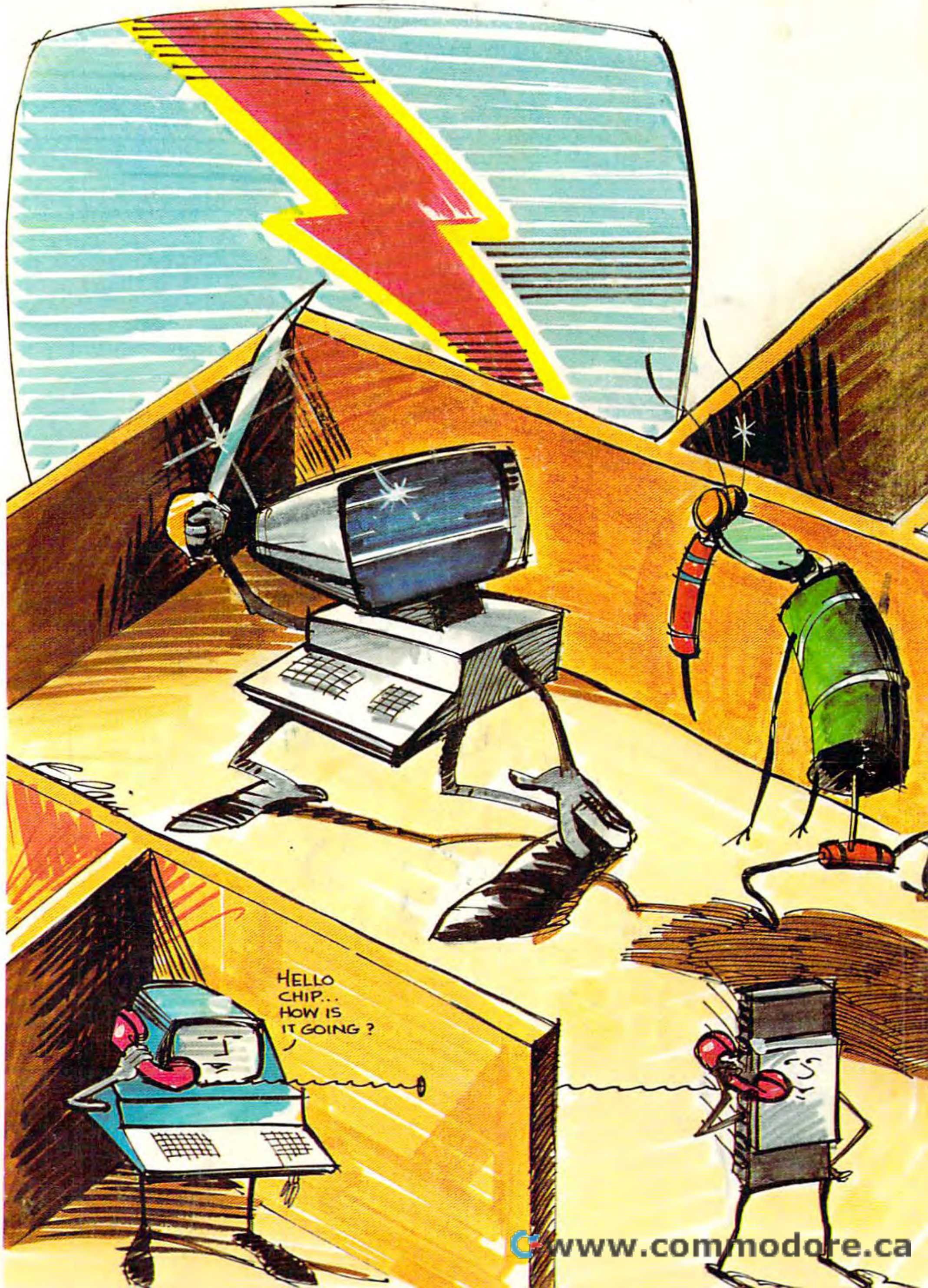
Teaching Basic Academic Skills - Can Micros Make A Difference?

Mixing Atari Graphics Modes

Thesus Versus The Minotaur - PASCAL Visits Ancient Greece

RS-232 Communications

TelePET



WordPro™

"WordPro is the most sophisticated Word Processing Software package available for the Commodore Computer line."

Solve Your Paperwork Problem . . . Let WordPro Software Do The Work

Using standard typing methods, hundreus of valuable hours are spent erasing, revising, and retyping letters and documents as you work towards a final draft copy. The second, third, or fourth drafts take just as long to type as the first!

With WordPro word processing software you can transform your Commodore computer into a "state of the art" word processing machine with sophisticated word processing features at an affordable price.

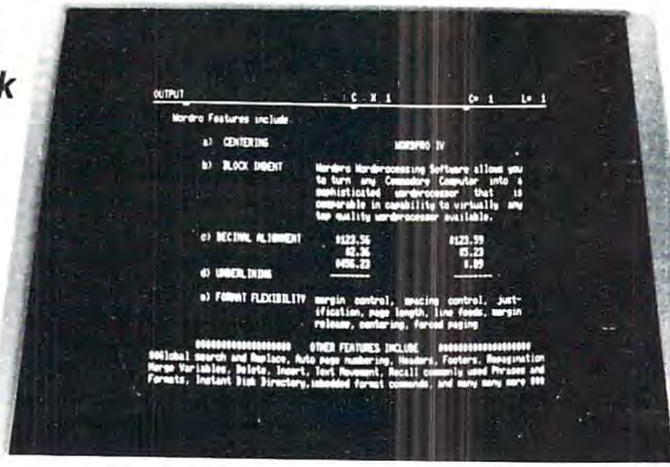
There are four versions of WordPro, ranging from the simple to the sophisticated. WordPro 1 on cassette will give computer enthusiasts a full range of text editing capabilities with cassette file storage. WordPro 2 is disk based and allows fast and easy file handling and manipulation. WordPro 3 was designed for professionals and contains the many features required in a business environment such as global search and replace, headers, footers, decimal tabulation, repagination, merging capabilities, and much, much more. WordPro 4 is our best. WordPro 4 runs on the new Commodore 8032, 80-column display computer. WordPro 4 has all the features of WordPro 3, plus additional features usually found only on the most sophisticated and expensive word processing equipment.

WordPro is a new breed of word processing software. Powerful, sophisticated, and easy to use, WordPro was field-tested by dozens of attorneys and commercial customers during 1979. WordPro is now installed and is saving its owners valuable time and money in hundreds of offices nationwide.

WordPro was designed with the user in mind. WordPro's unique "STATUS LINE" constantly interacts with the user by displaying the status of the system. Editing, storing documents, recalling letters, even the most sophisticated comands, are accomplished by a few, easy to remember, keystrokes.

You may find that WordPro alone is reason enough to own a computer.

WordPro can be found at most Commodore dealers worldwide. Call us for the number of the dealer nearest you. If you cannot locate a stocking WordPro dealer you may place an order with Professional Software via check or VISA/MasterCharge.



Actual Photograph of WordPro on CBM Model 8032

The many features of WordPro 1 - 4:

WordPro 1 - Cassette based • Status line • Text Editing • Insert/Delete • Screen Scroll Auto Repeat • String Search • Erase Functions • Link Files • Margin Controls • Tab Functions • Justification • Page Length

WordPro 2 - Most WordPro 1 Functions Plus + Disk Based • Paragraph Indent • Centering • Text Transfer • Hyphenation • Appending • Margin Release • Variable Blocks (Form Letters) • Multiple Copies • Automatic Disk Commands • Complete Disk File Handling

WordPro 3 - Commercial Disk Version for 40 Columns • WordPro 2 Functions Plus + Global Functions (Search/Replace/Copy) • Merging Disk File Linkage • 10 or 12 Pitch • Repagination • Duplicate Lines • Auto Delete Word/Sentence/Range • Numeric Mode • Underlining • Continuous Print • Headers/Footers • Auto Page Numbering • Proportional Justification • Forced Paging • Non-Print Comments • BASIC Language File Compatibility

WordPro 4 - Commercial Disk Version for 80 Columns • WordPro 3 Functions Plus + Displays and Formats Text to Screen for Review

WordPro 1 — For 8K RAM units. Requires C2N Peripheral/integrated cassette drive - **$29.95**

WordPro 2 — For 16K RAM units with 40 column screen. Requires 2040 disk drive - **$99.95**

WordPro 3 — For 32K RAM units with 40 column screen. Requires 2040 disk drive - **$199.95**

WordPro 4 — For Model 8032 with 80 column screen. Requires 2040 or 8050 disk drive - **$299.95**

All four versions of WordPro are written in 6502 machine code.

Professional Software Inc.

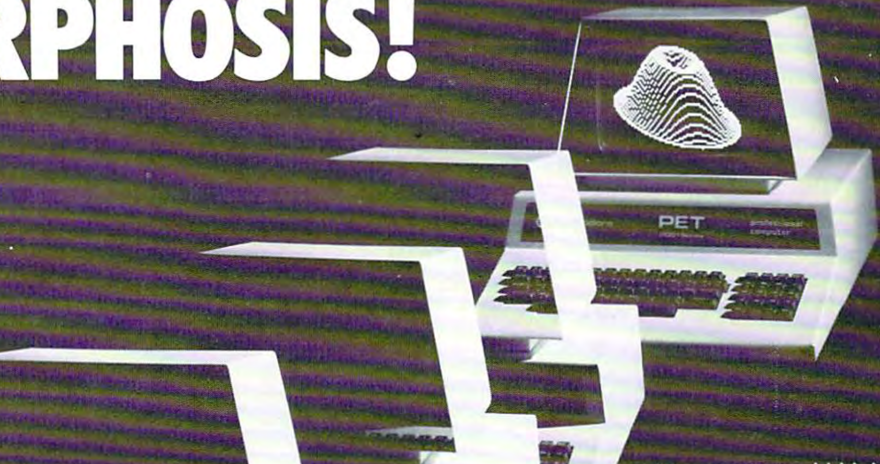
166 Crescent Rd., Needham, MA 02194

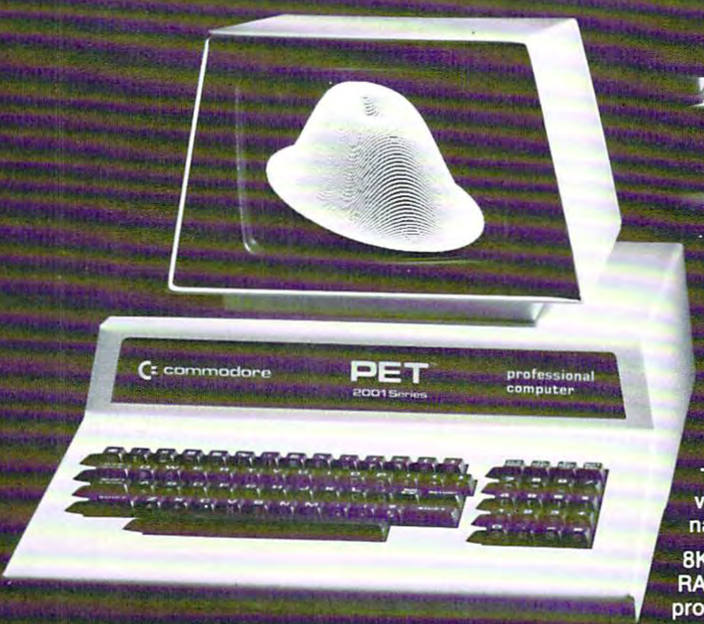
(617) 444-5224

WordPro Dealer Inquires Invited

WordPro was developed by Steve Punter of Pro-Micro Software Ltd., and is marketed exclusively by Professional Software Inc.

METAMORPHOSIS!





Now you can add high resolution graphics to your Commodore PET computer. The MTU K-1008-6 GRAPHIC INTERFACE can be used with either old, new, or business PET computers. It is simple to use, and fits inside the PET for protection.

The GRAPHIC INTERFACE gives you easy control over each dot in a matrix which is 320 wide by 200 high for a total of 64,000 dots. Because each dot can be controlled, either graphic images, text lines, or any mixture of the two can be displayed. Since each dot is controlled from software you can even design your own special character font or graphic image set (logic, chemical, architectural).

INTERFACE TO ALL PETS - With separate connector boards for each style PET (K-1007-2 for OLD PETS, K-1007-3 for NEW). The K-1008-6 can be used with either.

THREE TYPES OF VIDEO - You can select either normal PET video, graphic video, or the COMBINED image of both video signals simultaneously!

8K RAM MEMORY EXPANSION - The graphic matrix requires 8K RAM which is supplied onboard. This memory can be used for program or data storage when not being used for graphics (or see your program in binary on the display!).

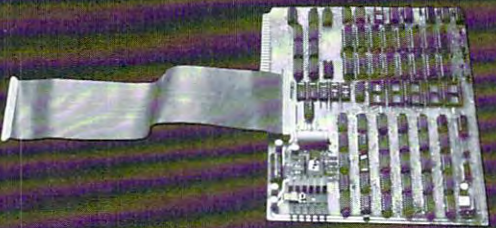
OLD OR NEW PETS CAN NOW HAVE HIGH RESOLUTION GRAPHICS

FLEXIBLY ADDRESSED ROM SOCKETS - Five ROM sockets are included on the board. They can be set *at the same* or different addresses, with you controlling which sockets are enabled at any time through software control. You also choose the sockets to be enabled when the PET is turned on.

EXTERNAL EXPANSION - This board also creates the KIM memory expansion bus supported by all MTU products. This allows insertion into our K-1005-P card file for expansion up to 4 other boards outside the PET case.

LIGHT PEN - The board has been designed to work with an optional light pen which MTU will be announcing soon.

SOFTWARE INTERFACED TO BASIC - MTU also has available machine language software to allow you to plot points, draw lines, and display characters at high speed.



Call or write for our full line catalog of products.

MICRO TECHNOLOGY UNLIMITED
P.O. Box 12106
2806 Hillsborough Street
Raleigh, N.C. 27605
(603) 627-1464

MTU
Micro Technology Unlimited
P.O. Box 12106
2806 Hillsborough Street
Raleigh, N.C. 27605
As of June 1, 1980

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

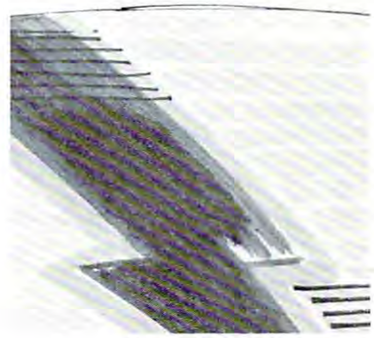
5115 MENEFEE DRIVE • DALLAS, TX 75227 • 214-381-0690

Table of Contents

September/October 1980. Volume 1. Issue 6

Page 26

Page 64



Page 71

Page 86

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

Domestic Subscriptions: 12 issues, $16.00. Send subscription orders or change of address (P.O. Form 3579) to Circulation Dept., COMPUTE. Magazine, P.O. Box 5406, Greensboro, NC 27403. Controlled circulation postage paid at Greensboro, NC 27403. ISSN 0194-357X.

Robert C. Lock, Publisher/Editor
Joretta Klepfer, Associate Editor
Carol Holmquist Lock, Circulation Manager
J. Gary Dean, Art Direction/Production Assistance
COMPUTE receives continuing editorial assistance from the following persons:
Harvey Herman, University of North Carolina at Greensboro
Jim Butterfield, Toronto, Canada
Larry Isaacs, Raleigh, NC
The following writers contribute on a regular basis as Contributing Editors:
Al Baker, 2327 S. Westminster, Wheaton, IL 60187
Gene Beals, 115 E. Stump Road, Montgomeryville, PA 18936
Len Lindsay, 1929 Northport Drive #6, Madison, WI 53704
Roy O'Brien, P.O. Box 426, Beaumont, CA 92223

Subscription Information (12 Issue Year):
COMPUTE. Circulation Dept.
P.O. Box 5406
Greensboro, NC 27403 USA
U.S. $16.00
Canada $18.00 (U.S. funds)
Europe: Surface Subscription, $20.00 (U.S. funds) if ordered direct, or available in local currency from the following distributors:
United Kingdom
Contact L. P. Enterprises,
8-11 Cambridge House
Cambridge Road
Barking, Essex
England IG1 18NT
Germany,
Switzerland,
Austria
Contact Ing. W. Hofacker GMBH
8 Munchen 75
Postfach 437
West Germany
Canadian Retail Dealers should contact:
Micron Distributing
409 Queen Street West
Toronto, Ontario M5V 2A5
(416) 363-6058

The Editor's Notes

Robert Lock, Publisher/Editor

Atari Marches On But Where Is Southern California?

It appears that the Atari machines have really picked up in sales. Southern California notwithstanding, the feedback I'm getting is that dealers ranging from the bigger mail order houses to the local corner store are seeing a great deal of *buying* interest in the hardware. Now I'm talking about US sales only, in as much as Atari's not really cranked up yet outside the US. And it honestly looks as if there's movement. Certainly makes the dealers happy, and COMPUTE also for that matter, in as much as we've been supporting the Atari since our beginning. It appears that the upsurge in buying began in mid to late June, and hasn't let up. Okay, so why all this ballyhoo from here? I'm setting the stage for some comments on Southern California:

The Background

Southern California, as we all know, has long been a focal point for the state of the art in small computing activity. There's much activity elsewhere of course, but Southern California has been active in developing what I would describe as a more advanced market. If you look at the number of major firms based out there you'll see a bit of what I mean.

The Apple Phenomena

This area enjoys an extremely active Apple market. In the LA area for example there must be dozens of dealers who are first and foremost Apple dealers.

It appears that some of the dealers have absolutely refused to carry the Atari, even to the point of occasionally calling it bad names and describing it in perjorative terms. And with an area of such tremendous Apple loyalty, that seems understandable. But on with the story.

The Feedback Cycle

Given the nature of the small computer market, all of us who are involved in any way with the activity of marketing a productr or service to users and buyers of these small computers rely on various means of marketplace feedback to develop and maintain marketing plans.

From here, I rely on numerous inputs, including those from dealers and subscribers all over the US. I've run into several advertisers in the last few weeks who have traditionally relied on their dealer contacts in that area to provide some portion of their planning feedback. In each of these cases, both advertisers had the clear and imminent opinion that the Atari machine was struggling, being clobbered by the Apple, etc., etc., and so on. Now mind you, this isn't the immediate concern. Everyone expects a new market (e.g. software or hardware for Atari) to be slow going at first. Their concern was the *future* of the machine, and by all tried and tested, locally valid, channels of feedback it appeared that Atari was in fact looking at a long up-hill struggle.

But all of this was totally inconsistent with my feedback. Not only were dealers all over the country telling me the machines were really starting to move, but our Atari subscriber base has been growing at a faster and faster rate. Clearly somebody's buying the machines, and if it wasn't the forefront, the vanguard, of Southern California, then who was it?

Aha!

What I finally decided, and I welcome some comment, is that Atari is selling to the market they've said all along they wanted to sell to. The (frequently) non-technical, new consumer of computing equipment. That's the market the machines are designed for and targeted at. The hobbyist market hotbed, Southern California with users with different needs, and dealers with different expectations, is not supplying good feedback on that market because Atari's successfully reaching the one they're aimed at. I think we may, after all, be achieving a new generation in consumer computing. ©

You May Be Expiring...

or Renewing Your Subscription To COMPUTE!

If you're an early COMPUTE! subscriber, your subscription may be running out. If your mailing label bears the code "11/80" or "12/80" then Issue #7, the November/December issue, is your last one.

Don't Miss An Issue

Renew now by sending us your check or money order for your 1980 subscription. Please follow these simple guidelines. Mail your renewal check to COMPUTE! P.O. Box 5406, Greensboro, NC 27403 USA. Mark the envelope "Attention: Subscription Renewal". Include your current mailing label. If you've thrown your envelope away, please make sure that you include your name and address (especially your address) the same way you've been receiving the magazine. Check the new price schedule and include a check, money order or Master Charge/Visa number with your renewal.

COMPUTE! Goes Monthly

First Monthly Issue is January, 1981

We're going monthly, by popular demand, and expanding the scope of COMPUTE! We're adding a special Gazette for Ohio Scientific machine owners, and a Gazette for the Single-Board AIM, KIM and SYM Owners.

We'll maintain the same high quality, the same resourceful standards, that have taken our paid circulation from less than 2,000 at the beginning of this year (Issue 2) to over 11,000 for this issue (Issue 6)!

New Pricing (COMPUTE!'s Still A Bargain)

A one-year (twelve issue) subscription to COMPUTE! is now $16.00 in the US. Canadian subscriptions are now $18.00 (in US funds). Surface mail subscriptions, to everywhere else in the world, are now $20.00 in US funds.

See The Reader's Feedback In This Issue for More Information

The Reader's Feedback

Robert Lock,
Publisher/Editor, and Readers

In case you missed it in the Editor's Notes, we're going monthly. Check there for a full timetable and information on keeping your subscription current.

Votes for **Best Article in Issue 5** indicate that lots of readers like the current range of material in COMPUTE. Jim Butterfield took the honors with Mixing Basic and Machine Language. Second place went to Plotting With the 2022, closely followed by How to Program in BASIC with the Subroutine Power of FORTRAN and Assembly Language Programming with UCSD PASCAL.

And now for the rest of the feedback ...

Author Note

A Commodore user makes this request:

You should indicate on all machine language listings the ROM version...

I agree. You should also indicate what machine you're using, e.g. keyboard, etc. We're already having review problems trying to keep machine configurations matched up with software design, so when you send software for review, please indicate what it will run on.

One More Author Plea

Please present machine language programs with fluent explanations. If one now uses "BASIC" to program, how would one enter this program into PET using machine language? Please do not be afraid to offend us with simple explanations.

On Merging Our Two Magazines

What happened to Nuts and Volts?

Include OSI in COMPUTE. My C2-4PMF has more in common with the Apple or PET than with a SYM...

First of all, Nuts and Volts moved to compute II when we established that single- board computer magazine. Secondly, I admit that compute II wasn't necessarily the place for OSI machines.

Our ability to go monthly has in part been defined by the merger of our two magazines. We announced in the August/September issue of compute II that we were merging the two magazines effective with the November/December issue of COMPUTE. In that issue, you'll find the return of the Single-Board Computer Gazette (covering the 6502 based KIM, SYM, and AIM systems), and the addition of an OSI Gazette. You OSI owners will in part determine the stability of the OSI Gazette by your submissions, so get writing!

Issue 7 of COMPUTE! (November/December) will be one united issue again, and in January you'll receive the first monthly issue of COMPUTE!

And Coming Next Issue (Ouch! Groan!)

I learned my lesson last time. Please understand that one of the advantages of waiting 'till the last minute to write my columns is keeping you as current as possible on "coming attractions". The disadvantage is that I got carried away in my enthusiasm last time round. Looking back, I must have said "And next issue we'll have..." 10 times in the first three pages. I blew it. I hereby officially announce that you should read such comments on my part as "And in a future issue we'll have...". That way if my enthusiasm gets ahead of our collective abilities here you won't be disappointed.

In a future issue, we'll have those promised business reviews. I am very pleased to report that over 50 business users have signed up to review professional software. What we're trying to do is get things rolling so that reviews will be the balanced opinion of several reviewers rather than the hasty overview of one. I apologize for my over enthusiastic promises last time.

On The Quality of COMPUTE!

I was fascinated to see that the most prevalent comment regarding our going monthly was *"Yes, do it, but only if you can maintain your current quality."* We pride ourselves on the quality of COMPUTE!, both in editorial quality and physical quality. That's been our goal since we started the magazine in the Fall of 1979, and we're committed to maintaining that quality. As always, keep me posted on our progress.

R.C.L. ©

HOWARD W. SAMS & CO., INC.
4300 West 62nd Street, P.O. Box 7092
Indianapolis, Indiana 46206
(317) 298-5400

Indicate quantity in boxes above and complete ordering information below. Return entire ad with order.

Total amount of order $______

Add local sales tax where applicable $______

GRAND TOTAL $______

☐ PAYMENT ENCLOSED (Save postage and handling costs)
☐ CHECK ☐ MONEY ORDER
☐ MASTER CHARGE ☐ VISA AD033

Expiration Date______

Interbank Number (Master Charge only)______

Account Number______

Minimum credit card purchase $10.00

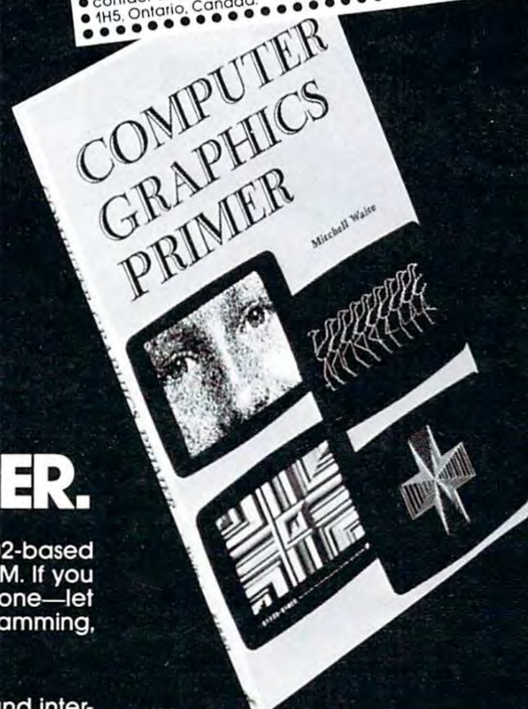
Name (print)______

Signature______

Address______

City______ State______ Zip______

Prices subject to change without notice. All books available from Sams Distributors, Bookstores and Computer Stores. Offer good in U.S. only. In Canada, contact Lenbrook Industries, Ltd., Scarborough, M1H, 1H5, Ontario, Canada.



GET THE MOST OUT OF YOUR 6502-BASED COMPUTER.

Sams has three books written especially for the popular 6502-based microcomputers like the APPLE, PET, ATARI, OSI, SYM, AIM and KIM. If you own a 6502-based computer—or are thinking about buying one—let Sams help you get the most out of it. We make graphics, programming, interfacing and software design simple.

■ **PROGRAMMING & INTERFACING THE 6502, WITH EXPERIMENTS.**
By Marvin De Jong. The more you know about programming and interfacing, the more performance you can get out of your microcomputer. This hands-on guide to 6502 presents 80 carefully graded experiments to help you get the most out of your AIM, KIM or SYM. NO. 21651. $13.95

■ **6502 SOFTWARE DESIGN**
Leo Scanlon—a leading computer expert—simplifies software design. Takes you from fundamentals into more complex topics. Get more versatility out of your computer by learning to program it yourself. INCLUDES 89 TEST PROGRAMS! NO. 21656. $10.50

■ **COMPUTER GRAPHICS PRIMER**
Mitchell Waite—one of the most popular computer authors—brings computer graphics into sharp focus. Shows you how to use a 6502-based computer to create complex drawings, plans, maps and schematics on a video screen. NO. 21650. $12.95

Sams BOOKS

Computers And Society

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

This month we want to bring you up to date on two shows we attended. One of us (BB) attended the American Society for Training and Development national convention in Anaheim, and the other (DT) attended the summer International Consumer Electronics Show (CES) in Chicago. Both of these shows had many small computer systems on display. By looking at these products at trade shows and conventions, we get to see developments before they become available at the corner computer store. We were sufficiently excited by what we saw to want to share our perspectives with you.

The following report presents Betty's view of the ASTD convention:

Does anyone remember CAI? The darling child of the late 60's and early 70's, computer assisted instruction has been struggling for its life for the past decade. Suffering from high costs as school budgets became increasingly tight, CAI never quite justified its existence or fulfilled the promises of early dreamers. In the latter part of the 70's the big guns in CAI turned to the adult education market and aimed at big business and industrial training. (Training is that skills-increasing activity engaged upon by people within a business or industry. Adult education may cover some of the same subjects, but it is conducted in a school or university.) Control Data Corporation took its successful Plato-for-schools and created Plato-for-industry. Boeing's Computer Services division offered training in all computer related subjects.

The applications still seemed limited. The CAI offered by these companies required either that the learner go to a centrally-located learning center to use a time-sharing terminal, or install such a terminal at company facilities. In my opinion, CAI, with all its promises, was just limping along.

A few months ago, I found out that CAI is indeed alive and well, and living in the personal computer industry. General applications in industrial training may be as close as tomorrow.

At the end of April I attended the national convention of the American Society for Training and Development, held in Anaheim, California. Among the more than 700 vendors of training hardware and software were several who displayed very exciting uses of personal computers in training for business and industry.

I am excited about what I saw because I wear at least two hats in this world. I am a computer enthusiast who has spent over three years in a research center watching people playing with CAI and playing with it myself. I am also a training director who is concerned with helping people learn and retain knowledge in the most efficient way. Until my April trip, I did not believe that the computer was efficient or cost effective. I may now be wrong.

What follows in this column is a brief description of what I saw and some caveats. I should point out that what I saw is not all that is available in the world for CAI for industry. Some vendors may not have attended this convention. I may have walked right by others. To all left out, my apologies. Write to me at Innovision and I'll be glad to take a look at your product and include it in a later column.

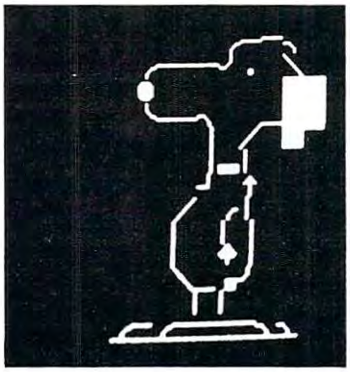
Let's start with a look at some of the hardware, because there was more of that than software. (One of the major problems I see with the use of computers in business training is that canned programs are not widely available.)

One of the exciting applications of the microprocessor was created by Videodetics (Anaheim, California). They have harnessed the technology by marrying it to videotape to create programmable video tape. Providing automatic search-out and playback of specific sections of tape, the controller-indexor system creates an interactive learning situation. The lesson creator programs in a series of questions, the answers to which lead the learner down various videotape paths. The learner is either praised or corrected (or both), as the lesson progresses. The unit makes possible such activities as reciprocating multiple-choice tests, reinforcement of correct responses, and remediation of incorrect choices.

With this product trainers can upgrade videotape equipment (if compatible) to allow learner control for a very low cost (between $550 and $700). The company is currently polishing a random access version of the controller, which should lend even greater flexibility to the system. The developers point out that the unit has also been used very successfully in point-of-purchase sales presentations, so it may serve a dual function in some businesses. In addition, the videotape visual display has the advantage of interest and color over the standard monochrome CRT normally associated with the CAI environment.

Recognizing the value of the videotape medium when compared to CRT only, Comco Creative Industries has interfaced an Apple (obtained from Bell and Howell) with a 3/4 inch video cassette to present an answer to the problem of boredom without loss of the advantages of conventional CAI. The box which accomplishes the jumping and linking of tape and computer units will cost, I am told, something around $1000.

Judge this character generator for yourself



Regular PET Graphics

HAL GRAPHICS

Compare the two Snoopys and you'll begin to see why the new HAL PCG 6500 is attracting so much attention. It's the programmable character generator for your PET that gives sharp, detailed graphics.

Important features include:

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

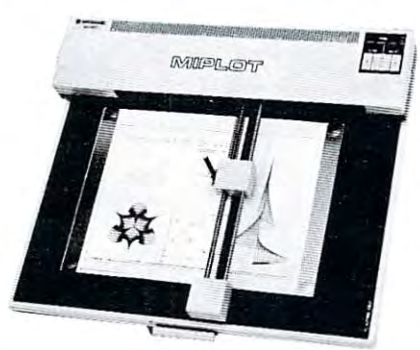
The HAL PCG 6500 is available exclusively at Systems Formulate ...

Exciting games are now available for the HAL PCG 6500 ... Jupiter Lander, Night Driver, Super Slot and more ... $19.95 each.

Only $200 (plus $5 shipping & handling)

Add $45 for 28 pin Conversion Kit

MIPLOT: the right plotter at the right price



MIPLOT by Watanabe Corporation

Designed for straight forward interface to any microcomputer that outputs the ASC11 code, MIPLOT can even be used by operators with no plotter experience.

- Incorporates pre-programmed "intelligent" functions required for producing graphs and drawings
- Solid and broken line types can be specified
- Built in character generator for letters, numbers and symbols
- Characters can be enlarged and rotated to four orientations
- Special printer mode outputs character data as-is
- Uses commonly available hard fiber-tip pens
- Maximum plot speed approximately 2 inches per second
- Built in self-test mode

Only $1,200

at Systems Formulate Corporation

(plus shipping & handling)

ADCOM Light Pen (with sound) ... Only $34.95

(plus $2 shipping & handling)

Unlike many light pens, the ADCOM interfaces with PET through the second cassette port rather than the parallel port. The result? Sound!

- Ready to plug in, no assembly necessary.
- Faster than many other light pens designed for PET.
- Built-in sensitivity control.
- Machine language and demo program included for ease of programming.

Expand with the finest in micro peripherals, systems and software from Systems Formulate.

Call today for more information or to place your phone order: (415) 326-9100

We honor Master Charge, Visa, check or money order.

(California residents add 6.5% tax)

SYSTEMS FORMULATE CORPORATION

39 Town & Country Village • Palo Alto, California 94301

The Bell and Howell system (designed around their version of the Apple II) comes with their own variety of PILOT (Mark-PILOT) as the CAI language. It contains both authority and presentation systems. Authors may use color graphics and animation with this system. Representatives at the convention were talking about a price of between $5,000 and $7,000 for the whole package.

The advantege of this system over the interactive video tape lies precisely in the greater variety of teaching techniques possible with the computer programs. While the system does require the instructor to spend some development time on-line with the lesson, it may be no more than that required for development of in-house training materials. But all such efforts are very time consuming. I remember a figure of 40 hours development time for each one hour of student program given to me several years ago by the Plato people at the University of Illinois.

Is this effort cost effective? Paper programs also take many hours, but the equipment is cheaper. I believe, however, that the potential impact upon the learner may be great enough to warrant the additional costs. Also, with this new generation of CAI equipment, costs are plummeting and the machines are becoming multi-purpose. After all, if you're going to have a personal computer in your home or office, you might as well use it to learn something.

While the personal computer based CAI units such as the Camco system are the logical competitors to the Plato and Boeing time-shared learning systems, other, more powerful, stand-alone systems are also being offered. One interesting-looking unit was shown by Regency Carroll of Champaign, Illinois (the home of Plato). It's language, USE, allows judging, help sequence branching, selective erase, and animation. They do not, I believe, have a video tape interface. My recollection of their system is that it is considerably more expensive than the Camco system, albeit more powerful. While the developers may claim an apples and oranges comparison, I believe that training people will go for the lower end machines because of the lower initial costs, especially if they are able to interface these systems with existing video tape equipment.

With all of the systems I saw, however, I felt the same frustration. After spending two years as a one-person training department, I know the enormous value of canned programs. I would much rather buy something than spend the time developing it in-house. So I look for quality programs. My search through the convention exhibit hall for quality programs for the computer systems led to very little. I found only one company, Educational Programming System, of St. Louis, who offered CompuCourse programs. These combine text and automated activities such as using the computer to set up an actual budget in a budgeting course. Their diskettes will be available for the Apple II or II + and for the TRS-80, and will be sold in retail stores in October. Their first available program will be in Personal Financial Planning, costing $95. Other titles they plan include: Managing Corporate Cash; Long Range Planning; Advertising: Strategy and Design; Writing for Successful Management; and How to Build Memory Skills.

These subjects are included in the curriculum for many companies. It will be interesting to see whether other software suppliers jump on the bandwagon to prepare materials for industrial-placed personal computers. Right now it seems to be a wide open field. Only the manufacturers of the computers themselves and a few software companies have provided any learning programs. Atari and SRA have agreed to develop software for schools, pre-school through university. But specific "training" programs are yet to come. I believe that if the personal computer is to make an impact on the world of business and industrial training, software developments must keep up with hardware developments. I'm hoping that they will. CAI will really have come of age then.

While educational and industrial applications for personal computers are growing at a rapid pace, we have yet to see the true "home computer" market open up. Most people who have computers in their homes use them for business applications, or as a hobby. The Consumer Electronics show is an interesting showcase of technology for the home. The presence of personal computers at this show indicates a feeling that someday soon the true consumer market for computers will become a reality. It is worthwhile for computer vendors to plan for this market, since, once the market develops, personal computer sales might rival those of color televisions. With the view of someone looking at incipient computer sales of several million units a year, let's look at Dave's perspectives on the CES:

The June CES is the second of two international consumer electronics shows presented in the United States every year. While exhibited products included almost any entertainment item which uses silicon, it is interesting to see how this show is becoming a showcase for Personal Computers. Since the birth of the "appliance grade" personal computer in 1977, attendees of these consumer electronics extravaganzas have had to include computers in the list of products which are capturing the hearts and minds, if not the pocketbooks, of a growing fraction of consumers all over the world.

While the time has not yet arrived for the computer to be considered a common household appliance, the incipient emergence of several well supported information utilities suggests that it will not be long before computer sales exceed 1,000,000 units per year, and the long-awaited emergence of "home" computing becomes a generally accepted phenomenon.

As in past shows, Texas Instruments continued to stress the versatility of their 99/4 computer in applications including speech output and connections to the Source and MicroNet information utilities. However, TI apparently has not received the market acceptance they had hoped for, and it will be interesting to see how effective they are at surviving in an industry which has seen several fine products withdrawn from the market. According to several cottage industry people interviewed at the show, a major frustration with the TI computer arises from the lack of a way to generate and call machine language subroutines from BASIC. From a human factors point of view, I found it distressing to see lower-case characters properly displayed on the screen (from a Source data base) while there is no provision for the entry of lower-case letters from the computer keyboard.

The APF Imagination Machine remained unchanged in the past year, with their emphasis being placed on peripherals on marketing schemes. While many of the characteristics of this 6800-based computer are quite nice (single keystroke BASIC keywords, excellent keyboard feel, etc.), the excessively large size of the computer combined with an indistinct display makes this computer less appealing than it might otherwise be.

Ohio Scientific presently provides one of the widest product lines available, ranging from a small personal computer to a conventionally packaged minicomputer. The 6502-based C-1P and C-1P/MF computers have been given a new plastic housing in apparent preparation for their presence in Montgomery Wards' stores all over the country. At a little over $1000, the C-1P/MF is probably the lowest price computer with a floppy disk.

At the other extreme in cost, the HP-85 desk-top computer with built-in 5'' CRT was well displayed by Hewlett-Packard in a booth which, to my eyes, was sparsely attended. It may be that the CES is the wrong place to show a $3250 computer whose features seem not too far removed from those of computers selling for thousands of dollars less.

The Compucolor disk-based computer system from Intelligent Systems Corp. has perhaps suffered from styling problems - especially when compared to the more expensive Intecolor computers also manufactured by ISC. However, through the miracles of modern packaging, the Compucolor computer has been given a face lift and now looks amazingly like its larger brother.

As in the past shows, the Atari 400 and 800 computers continue to draw large crowds. The use of dedicated display and sound processors serves to extend the power of the 6502B microprocessor to give these computers the finest color and sound capability shown at the CES. While much of the Atari display was devoted to their ability to connect to home information utilities and to play very sophisticated animated games, they did introduce a light-pen attachment and also demonstrated some educational software developed for Atari by SRA, a division of IBM.

Among the several new computers introduced at the June CES, one of the most interesting entries was the Sinclair ZX-80. This Z-80-based computer (which weighs only slightly more than its instruction manual (320 g vs. 250 g)) contains a full typewriter-like keyboard (membrane type), 1 KB of RAM and a 4 KB BASIC. Keywords are entered with single keystrokes, and the syntax of each line entry is continuously monitored. It is almost impossible to get the computer to accept a syntactically invalid line of code. The ZX-80 connects to the UHF input on a black and white TV and displays 24 lines of 32 characters. As an indicator of the attention paid to low-cost design, conversion of the ZX-80 from the European PAL to the U.S. NTSC TU standard is accomplished by the addition of a single diode. Since power (9 V DC) is provided from an outboard plug-mounted power supply, the ZX-80 can be used almost anywhere. Rather than sell this product through stores, the initial Sinclair marketing plan is to sell the ZX-80 from England, fully assembled, for $199. Presently, the ZX-80 only supports an integer BASIC, but an 8 KB floating point BASIC is in development. Since external RAM can be added to bring the computer to 16 KB, the ZX-80 may create a totally new market. Since new markets appear to be Mr. Sinclair's forte, this product bears watching.

An even smaller computer was introduced by Panasonic: the HHC hand held computer. The central unit (which will retail for about $400) is about the size of the Craig translator. This unit contains a 6502 microprocessor, 1 KB of RAM and slots for up to four ROM cartridges. In addition to pre-programmed functions (information terminal, language translator, etc.), ROM packs will be available for languages such as FORTH and EASIC. The main unit contains a full complement of keys (although with the wrong spacing for easy typing) and has a liquid crystal one line display (24 characters, upper and lower case dot matrix). The addition of myriad peripherals. Among the peripherals demonstrated at the show, I saw the TV adaptor which buffers and displays a screen full of information in color. A small printer, a modem/acoustic coupler, and RAM expansion units were also shown. RAM units contain their own battery backup thus allowing users to create their own ''firmware'' for this sytem. While the main unit is nicely packaged, the expanded system has an ''Erector Set'' quality to it that detracts from its overall appearance. Nonetheless, the emergence of this product along with the Sinclair ZX-80 shows that there is still room for experimentation in the personal computer market.

Commodore's exhibit stressed their watches and calculators, with one 80-column CBM computer on display. Hidden behind a smoked plastic screen, however, was the Commodore VIC - an as-yet experimental computer designed to connect to a color TV. If VIC becomes a product soon (and I hope that it will), this compact 6502-based machine is certain to capture the hearts of thousands of users. Sized only slightly larger than the Sinclair computer (and using the "old" PET keyboard), VIC is designed to sell, with 4 KB of RAM, in the $400 range. If it uses nearly the same BASIC used in the rest of the CBM world, strong cottage industry support is virtually guaranteed in advance. Through products of this type, Commodore is retaining their commitment to the low-end market while broadening their product line to compete with machines such as the IBM 5120.

In June of 1979, Casio showed their versatility as a company by introducing the Casiotone professional music synthesizer. This departure from their traditional watches and calculators was followed this year by the introduction of the FX-9000P, an 8080A-based computer whose packaging closely resembles that of the HP-85. A crisp built-in high resolution 5" CRT display (32 characters by 16 lines, 256 by 128 pixels) is capable of mixed text and graphics applications. When this computer comes to market early next year, it is expected to retail for $900 with a ROM BASIC. The built-in 8 KB RAM can be supplemented with plug-in modules. The user can choose between 16 KB dynamic RAM cartridges or 4 KB RAM cartridges with battery back-up. As with the Panasonic entry, programs can be written into removable RAM cartridges and treated like ROM-based firmware. A tape cassette interface is available along with a real-time clock with calendar and alarm. Several parallel and serial interfaces are available to allow connection to printers, disk drives and modems. In other words, the FX-9000P is a serious small computer priced to sell by the thousands. The physical resemblance of this computer to the HP-85 is striking. At a $2600 price advantage over the HP entry, the Casio FX-9000P was the recipient of much well deserved attention.

While the Mattel Intellivision has been shown with a full keyboard attachment for more than a year, there has been much speculation regarding the reasons this portion of the product has not been introduced commercially. Early plans were to not make the Intellivision user programmable. As of the June CES, a new philosophy is apparent. The Intellivision keyboard unit (designed to retail for $500) will contain a 6502 microprocessor with 16 KB of RAM and running what appears to be a full extended Microsoft BASIC. Since the display portion of this product (housed in the video game unit) contains a 16-bit GI computer and the "Teleview" information utility chip set, this new product may leverage its way into a broad share of the market.

Several companies who have personal computers on the marketplace chose to not display their wares at the CES. For example, since Radio Shack computers are not available for sale through non-Radi Shack stores, they do not display their wares at the CES.

Apple Computer, however, has used the CES as a showcase for their products. This June, Appl relied on full page advertisements in the trade dailies, and did not have an exhibit on the floor. One could conjecture that the recent introduction of the $4,000 + Apple III at the National Computer Conference (NCC) was considered to be adequate exposure, especially since this new product is probably not geared towards the type of markets addressed by the majority of the buyers who attended the CES. Judging from their advertising, however, it is clear that Apple is planning to maintain their strong position in the $1,000 personal computer marketplace.

The Exidy Sorcerer was not on display either, although this was probably due in part to the forthcoming acquisition of this product line by another company.

Another computer which was not displayed was the Sharp PC-1211 hand held computer. This $200 CMOS computer has a complete keyboard and 24 character liquid crystal display. While the product is available in Japan and Europe, it is rumored that Sharp has elected to not introduce this product in the U.S., but to wait until a later version is ready, perhaps by next year. Since the Sharp PC-1211 supports a serial I/O port, an attachment is available for storing data and programs on a conventional tape cassette. Unlike Panasonic, however, Sharp is apparently not ready to introduce the communications and printer options which are probably very important selling points for these machines. I have received one of these computers from Japan and have found it to be very nice to operate, both from a hardware and software point of view. The resident BASIC is well designed for scientific calculations, although string operations are quite limited. It will be interesting to see if another vendor picks this product up as a private label item, thus gaining income for Sharp without forcing their hand too early.

What message, if any, can one glean from all this information? For one thing we know that computer manufacturers have a long way to go before their products will appeal to the average consumer. The trends towards simpler and easier to use computers are evident. Communications (in the form of connections to information utilities such as the Source and MicroNet) are perceived as being of paramount importance to consumers, and the development of high quality software is becoming more evident. It may take a year or so, but before the end of this decade, the personal computer revolution will come home. You, as a personal computer enthusiast, have a head start on what promises to be a most exciting future. ©

Teaching Basic Academic Skills
Can Micros Make A Difference?

Tory Esbensen, Coordinator of Elementary Curriculum and Instruction
Doug Hed, Supervisor of Media Services, Edina Public Schools, Edina, Mn.55435

As microcomputers become more visible in school settings, they may be increasingly asked to present their teaching credentials. This report is a preliminary attempt to respond to that likely development.

In the fall of 1979, the Iowa Tests of Basic Skills were given to all of the 3rd and 5th graders in the Edina Public Schools. Students who scored poorly on these tests in capitalization, punctuation, and usage, were singled out to take advantage of microcomputer programs written for the PET in these academic areas by MICRO-ED, INC. (Box 24156, Minneapolis, Minnesota, 55424).

Although every elementary school in the Edina system uses microcomputers, and although microcomputer programs are readily available to any classroom teacher wishing to employ them, for the purpose of this project special instructional arrangements were made with student support centers that had been established in four of our elementary schools. Briefly, in those schools where support centers existed, 3rd and 5th grade students who scored in the bottom quartile of the Iowa Tests in capitalization, punctuation, and usage, were targeted to receive additional instruction from microcomputers. This selection procedure was based on local norms which are higher than national norms.

It is important to emphasize that no attempt was made to handle this as a pure research project. No students were used as a control group. In the four schools in which the project was formally carried out, we tried to provide microcomputer instruction to every student who seemed to need it. In those schools which had no support centers, microcomputers were also used by individual teachers to provide additional instruction to students. No attempt was made to restrict this in any way.

Above all, great care was taken to avoid giving any impression that microcomputers are somehow preferable to other modes of instruction. In our opinion, it is important to have micros viewed as the instructional allies of teachers, not as competitors.

Therefore, what this report will provide is information concerning what happened to a group of students when microcomputers were used to play a major role in furnishing certain kinds of remedial instruction. No comparison with other instructional practices or results is intended or implied.

We shall begin by considering a group of 59 fifth grade students who scored the lowest in the Iowa Tests in the area of English usage. In the fall of 1979, based on national norms, the median score for this group placed it at the 37th percentile for a grade equivalent score of 4.4.

When this group was re-tested in the spring of 1980, its median score for English usage placed it at the 58th percentile for a grade equivalent score of 6.4. Academically, this group of students gained a total of 20 months over a period of 7 months.

Next we shall look at a group of 67 fifth grade students who scored the lowest in the Iowa Tests in the area of punctuation. In the fall of 1979, based on national norms, the median score for this group placed it at the 36th percentile for a grade equivalent score of 4.5.

When this group was re-tested in the spring of 1980, its median score for punctuation placed it at the 62nd percentile for a grade equivalent score of 6.5. Academically, this group of students also gained a total of 20 months over a period of 7 months.

Then we shall consider a group of 73 fifth grade students who scored the lowest in the Iowa Tests in the area of capitalization. In the fall of 1979, based on national norms, the median score for this group placed it at the 35th percentile for a grade equivalent score of 4.5.

When this group was re-tested in the spring of 1980, its median score for capitalization placed it at the 70th percentile for a grade equivalent score of 7.0. Academically, this group of students gained a total of 25 months over a period of 7 months.

Now we shall consider a group of 43 third grade students who scored the lowest in the Iowa Tests in the area of capitalization. In the fall of 1979, based on national norms, the median score for this group placed it at the 25th percentile for a grade equivalent score of 2.4.

When this group was re-tested in the spring of 1980, its median score for capitalization placed it at the 59th percentile for a grade equivalent score of 4.2. Academically, this group of students gained a total of 18 months over a period of 7 months.

Next we shall look at a group of 35 third grade students who scored the lowest in the Iowa Test in the area of English usage. In the fall of 1979, based on national norms, the median score for this group placed it at the 33rd percentile for a grade equivalent score of 2.4.

When this group was re-tested in the spring of 1980, its median score for English usage placed it at the 72nd percentile for a grade equivalent score of 5.1. Academically, this group of students gained a total of 27 months over a period of 7 months.

Skyles Electric Works

Q. Your students are gathering around the several PET computers in your classroom. And they all are hungry for hands-on turns at the keyboards. Some students are just beginning to understand computers; others are so advanced they can help you clean up the programs at the end of the period. How do you set up a job queue, how do you keep the beginners from crashing a program, how do you let the advanced students have full access? And how do you preserve your sanity while all this is going on?

A. With the Regent.

Q. What is the Regent?

A. The ultimate in classroom multiple PET systems. A surprisingly inexpensive, simple, effective way to have students at all levels of computer capability work and learn on a system with up to 15 PETs while the instructor has complete control and receives individual progress reports.

Up to 15 PETs, one dual disk drive and as many as five printers can interface with the Regent, and do all those good things we promised. It's designed to operate with 8K, 16K, 32K PET/CBM models and with the Commodore disk drives and new DOS.

Five levels of user privilege, from the Systems Level, through Levels One and Two, Student; Levels One and Two, Operator. From only the use of system commands to complete control for the exclusive use of the instructor.

There's complete system protection against the novice user crashing the program; the instructor has total control over, and receives reports concerning, usage of all PETs.

A complete set of explanations for all user commands is stored on the disk for instant access by all users. And a printout of the record of all usage of Regent is available at the instructor's command.

The Regent includes a systems disk with 100,000-plus bytes for program storage, a ROM program module, together with a Proctor and a SUB-it . . . and complete instructor and student user manuals.

Q. SUB-it? Proctor? What are they?

A. **The SUB-it** is a single ROM chip (on an interface board in the case of the original 2001-8 models) that allows up to 15 PETs to be connected to a common disk via the standard PET-IEEE cables. The Commodore 2040, 2050 or 8050 dual disks and a printer may be used.

(The SUB-it has no system software or hardware to supervise access to the IEEE bus. The system is thus unprotected from user-created problems. Any user—even a rank novice—has full access to all commands and to the disk and bus. This situation can, of course be corrected partially by the Proctor, completely by the Regent.)

The SUB-it prevents inadvertant disruption when one unit in a system is loading and another is being used.

The Proctor takes charge of the bus and resolves multiple user conflicts. Each student can load down from the same disk but cannot inadvertently load to or wipe out the disk. Good for computer aided instruction and for library applications, offering hundreds of programs to beginning computer users.

A combination of hardware and software protects the disk from unexpected erasures and settles IEEE bus usage conflicts. Only the instructor or a delegate can send programs to the disk. Yet all the PETs in the system have access to all disk programs. Available for all PET/CBM models. SUB-it and PET intercontrol module and DLW (down-loading software) are included.

Q. How expensive are these classroom miracles?

A. We think the word is **inexpensive**. The **Regent** system is **$250** for the first PET; **$150** for each additional PET in the system. The **SUB-it** is **$40**. (Add an interface board at **$22.50** if the PET is an original 2001-8.) And the **Proctor** is $95.

There are cables available, too: 1 meter at $40 each; 2 meter, $60 each; 4 meter, $90 each.

Phone or write for information. We'll be delighted to answer any questions and to send you the complete information package.

Skyles Electric Works
231 E South Whisman Road
Mountain View, CA 94041
(415) 965-1735

Finally, we shall report on a group of 39 third grade students who scored the lowest in the Iowa Tests in the area of punctuation. In the fall of 1979, based on national norms, the median score for this group placed it at the 20th percentile for a grade equivalent score of 2.2.

When this group was re-tested in the spring of 1980, its median score for punctuation placed it at the 80th percentile for a grade equivalent score of 5.3. Academically, this group of students gained a total of 31 months over a period of 7 months.

Inasmuch as our elementary student support centers played such a central role in the shaping of this microcomputer instructional project, it would seem appropriate to explain something about the operation of these centers.

During the 1979-80 school year, our Concord, Cornelia, Creek Valley, and Wooddale elementary schools housed student support centers. By the fall of 1980, all of our elementary schools will have them. Here is how these centers function:

Each one is supervised by a paraprofessional, and instruction there is not necessarily remedial in nature. Students are scheduled into the center according to specific instructional needs as determined by their classroom teachers. A student may begin work in the center at any time during the year, and continue until a designated sequence of lessons has been completed.

Different kinds of instructional materials and equipment (kits, tape recorders, etc.) are available in the center. Nevertheless, the microcomputer has been the major engine of instruction. It is not hard to understand why.

Unlike many other machines, the microcomputer is not a special function device. The typical piece of hardware is dedicated to perform a specific function. Thus, a motion picture projector shows films, a record player plays records, and so on. Not so with the computer. Sometimes called a "smart" machine, this sophisticated device needs only to be told what to do in order to carry out a broad range of tasks. It can help manage a business enterprise, assist doctors in diagnosing illnesses, and play a strong game of chess. As our present study shows, it can help students learn effectively.

What sets the microcomputer apart from its more ponderous ancestors? The expression *computer-on-a-chip* tells the story. The ability of modern technology to miniaturize its creations means that something small can nevertheless be incredibly powerful. A microcomputer such as the PET weighs only about forty pounds, uses no more energy than a 150-watt light bulb, and can be plugged into an ordinary electrical outlet as you would a radio or phonograph. Although it costs no more than a good television set, its versatility, for all practical purposes, is limited only by the skill and imagination of those who know how to use it. Within the field of education, its capabilities are only just beginning to be explored.

What do students think of the microcomputer? Our student support center personnel are unanimous in their verdict: The students love it! Indeed, never was remedial instruction sought with such eagerness as when it was offered by way of the microcomputer.

Teachers, too, for the most part, have been supportive of this mode of instruction - increasingly so as time has gone on. Several have commented favorably on the tangible benefits they have observed as a result of their students having worked with micros.

Although parents have not been queried formally as to their views on the matter, a number of them have voluntarily expressed their enthusiasm for the use of microcomputers as an additional aid to learning.

So where do we go from here? Let us tentatively offer these concluding thoughts:

The education establishment (of which we are bona fide members) will take most kindly to microcomputers when these wonderful instruments are seen as supplemental to other forms of instruction, not as replacements for them. This means that manufacturers and publishers alike would be well advised to promote micros as being particularly useful to teachers in the areas of remediation, enrichment, special education, and homebound instruction. Implication: Any comprehensive and relatively expensive arrangement requiring full-scale classroom participation may be a difficult package to sell to educators.

Mastery learning, including competency-based teaching and testing, may very well be an idea whose time is rapidly coming if, indeed, it is not already here. Implication: It is possible that as school people generally begin to grasp some of the implications of microcomputers for education, it will be seen tht micros may be fundamental to the successful application of mastery learning on a broad scale.

No one, of course, can clearly foresee what is going to happen. But all of us who are impressed by the mighty potential of the microcomputer would do well not to repeat the mindless optimism of the 1960's when (do you remember?) teaching machines first blossomed. In those halcyon days, equipment vendors rushed to market with hardware that needed only programs in order to teach anything. Teachers, it was cheerfully assumed, would quickly fill this need by creating instructional hearts for tin woodsmen. Alas, this did not happen. Implication: We should carefully avoid making this mistake again.

Finally, this observation:

A famous educator once said, "Madam, we guarantee results - or we return the boy!"

In our dawning new age of customer-oriented education, what is more likely to be returned now is the machine. ©

Basic In A Nutshell

Name: Step-By-Step
Vendor: Program Design, Inc., 11 Idar Court, Greenwich CT 06830
Price: $49.95
Purpose: Teaches how to program a TRS-80 using BASIC
Documentation: Outstanding
Loading: OK – Level 6, not critical
Implementation: This is a case of a BASIC program that teaches BASIC programming. It starts out with the assumption that the student only knows how to turn the TRS-80 on. Three cassette tapes are mounted in the cover of a looseleaf notebook that also contains supplementary information frames. The course is divided into ten two-part lessons. From a simple PRINT "HI" through arrays and graphics to complex programs, all of the Level II commands and statements are exercised.

The instruction method consists of explanation, example, trial and testing. Commands and statements are presented and explained, examples are shown both on the screen and in the notebook, and then the student is presented with some problems to solve using the BASIC elements under discussion. If an incorrect answer is given, two more tries are allowed, and then the correct answer is displayed. Each lesson ends with a test that is administered and scored by the computer. The results are then entered into the student's progress chart. More comprehensive examinations are given at the end of Lesson 5 and at the end of the course.

Suitability: This is the kind of educational programming that personal computing needs more of. The student (my teenage son) learned much more quickly than I could have taught him, and at his own pace. However, this course isn't just for youngsters but for anyone who wants to be able to program effectively using the BASIC language. In a household where there isn't anyone to do the teaching, this course would be especially useful. I'd like to see a similar course for assembly-language programming.

Other software available from the same vendor: IQ Builders (four different kinds), Memory Builder and Story Builder.

Reprinted with permission:
80 Microcomputing, February 1980

Step by Step also available for Apple II and Pet Apple II version also available on disks for $59.95.
Available at Computerland and other fine computer dealers. Or, use the coupon below.

PDI

Program Design, Inc. 11 Idar Court Greenwich, Conn. 06830 203-661-8799

ORDER FORM

Quantity	Title	Computer	Price
______	STEP BY STEP	______	______

Please send me a list of dealers in my area ☐
Check or m.o. enclosed ☐

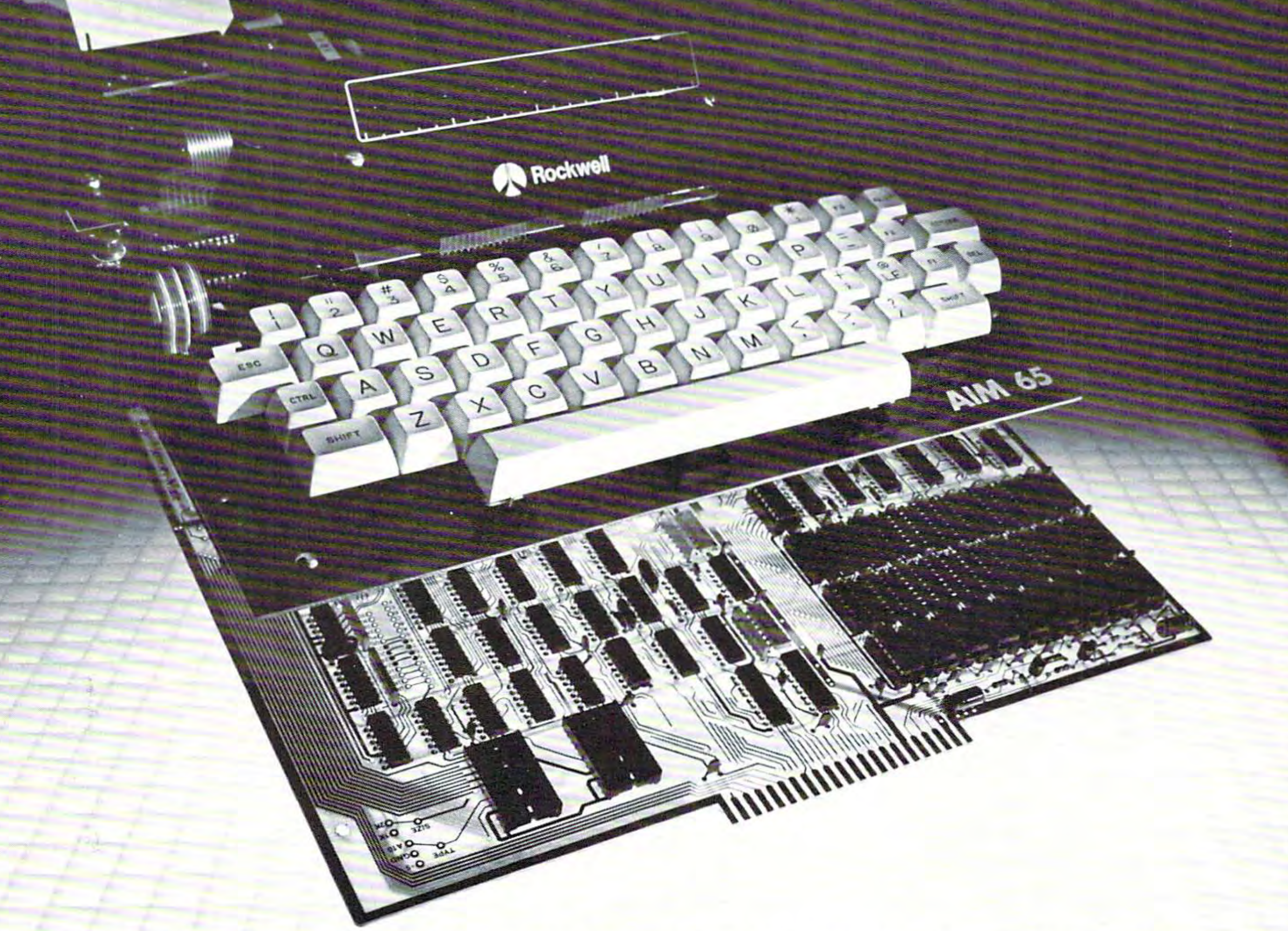
Shipping 5% ______
CT resident add 7% ______
TOTAL ______

BankAmericard/VISA # ______
Mastercharge # ______
MC 4 digits over name ______
EX date ______ Phone ______

Name: ______
Address: ______
City: ______
State & Zip: ______

*Foreign orders: U.S. funds
Add $10

Finally...Serious Expansion for the AIM-65



Introducing **Memory-Mate***, the AIM-65 expansion board that lets you spend your time on application solutions, not hardware hassles. Add Memory-Mate to your AIM-65 and make quick work of development and process control projects.

In its primary function, the Memory-Mate board provides 16-48K of RAM expansion assignable in 4K blocks anywhere in the system. Memory-Mate's parity check circuitry insures system RAM integrity (including AIM's 4K on-board RAM) for high reliability applications. The programmable write protect feature eases software development chores. This compact board, which fits directly **beneath** the AIM, also includes four programmable I/O ports, a tone generator for audible warnings, and sockets for 4K of PROM.

I/O intensive applications are accommodated with Memory-Mate's STD BUS interface option. Use off-the-shelf STD BUS cards to solve your biggest I/O problems.

The Memory-Mate with 16K RAM is priced at $475, with 16K expansion chip sets (including parity chip) costing $100 each. With 48-hour active burn-in and warranty for a full year, you won't have to worry about reliability either.

First of the complete AIM-Mate* series, Memory-Mate will be joined shortly by the Video-Mate*, Floppy-Mate* and the AIM-Mate case. For further information on the entire AIM-Mate series, write 'Attn: AIM-Mate Series' at the address below.

*TM Forethought Products

Introductory Special for Compute Readers.
16K Memory-Mate Board for $399 thru Oct. 15! Call for Details.

Forethought Products

87070 Dukhobar Rd., Eugene, OR 97402
(503) 485-8575

RS232 COMMUNICATIONS

Michael E. Day
2590 DeBok Road
West Linn, OR 97068

As more computer equipment is purchased by the small systems user, connecting this equipment together becomes a bigger problem, particularly when the equipment is made by different manufacturers.

One of the more common methods of connecting data communications equipment together is by way of the RS232 standard. However, even this has been cause for confusion, as there are various levels of implementation within the standard.

The purpose of this article is to provide sufficient information concerning the RS232 standard to allow proper implementation at the desired level.

The minimum level of RS232 consists of:
- Pin 2 TXD (Transmitted Data -OUT-)
- Pin 3 RSD (Received Data -IN-)
- Pin 7 Logic Grnd

The 2nd level consists of the minimum level plus:
- Pin 6 DSR (Data Set Ready)
- Pin 8 DCD (Data Carrier Detect)
- Pin 20 DTR (Data Terminal Ready)

The 3rd level consists of the other two levels plus:
- Pin 4 RTS (Request to Send)
- Pin 5 CTS (Clear to Send)
- Pin 22 RI (Ring Indicator)

Pin 1 Protective Ground should be used at all levels; however, it is not required for proper operation.

Level 1 is normally used with equipment tied directly to each other, such as a terminal tied directly to a computer. Level 2 is normally used where some degree of handshaking is required, and is often found on accoustic couplers. The third level is used where a more detailed control of the information flow is required. This level will usually be found with auto answer modems.

This is a generalization of what will be encountered by the small systems user, and in no way implies that all equipment will follow these rules. Some equipment will need other special signals, or not use all of the signals within a specific level. Synchronous transmission will normally require additional special lines and will be described in detail later.

There have been three standards of RS232 produced--A, B, & C. RS232A is obsolete, and equipment using this standard is almost non-existent. RS232B is also obsolete; however, there is still some old equipment around that uses this standard. RS232B is basically the same as RS232C except that the Transmit Data and Receive Data signal levels are inverted; that is, a marking condition is a positive level rather than a negative level.

The following is a description of the full RS232C standard. It is not required that all signals be provided, and it may be implemented in part or in full.

Each data set has a standard 25-pin connector (Cinch or Cannon chassis-mount, female type DB-25S). The table below has the pin number, the circuit mnemonic, and description for each signal in the RS232-C interface. Unassigned pin may have a different function in each type of data set, so check the technical manual for pin assignments for each data set.

Pin Number	Mnemonic	Description
1	AA	Protective Ground
2	BA	Transmitted Data
3	BB	Received Data
4	CA	Request to Send
5	CB	Clear to Send
6	CC	Data Set Ready
7	AB	Signal Ground (Common Return)
8	CF	Received Line Signal Detector
9	--	(Reserved for Data Set Testing)
10	--	(Reserved for Data Set Testing)
11		Unassigned
12	SCF	Sec. Rec'd. Line Sig. Detector
13	SCB	Sec. Clear to Send
14	SBA	Secondary Transmitted Data
15	DB	Transmission Signal Element Timing (DCE Source)
16	SBB	Secondary Received Data
17	DD	Receiver Signal Element Timing (DCE Source)
18		Unassigned
19	SCA	Secondary Request to Send
20	CD	Data Terminal Ready
21	CG	Signal Quality Detector
22	CE	Ring Indicator
23	CH/CI	Data Signal Rate Selector (DTE/DCE Source)
24	DA	Transmit Signal Element Timing (DTE Source)
25		Unassigned

For timing and control interchange signals, the function will be ON when the voltage is more positive than plus three volts and OFF when the voltage is more negative than minus three volts. The table below illustrates the signal function voltage relationships.

INTERCHANGE VOLTAGE

	NEGATIVE -3 to -25	POSITIVE +3 to +25
Binary State	1	0
Signal Condition	Marking	Spacing
Function	OFF	ON

HAYDEN SOFTWARE...

New!

DATA MANAGER: A Data Base Management System and Mailing List

(Lutus) Do what the big machines do with your Apple II! This all-machine language program stores up to 96,000 alphanumeric characters on just one floppy disk. Powerful cursor-based editor facilitates easy information alteration in the data base. Program permits the user to sort on any key and subfiles on any search. Retrieve data in any combination of categories from up to 32,000 characters within one-half second. Choose between screen display or serial print-out (via the game paddle connector). Ideal for mailing lists, data banks, index files, or any other facts, files, or statistics that are begging to be organized. *#04909, Apple II Disk Version, $49.95.*

New!

APPLE ™ ASSEMBLY LANGUAGE DEVELOPMENT SYSTEM

(Lutus) Enter the world of machine language programming with this brand new, winning, utility program. This very capable Assembler/Editor/Formatter allows you to write and modify your language programs quickly and easily. Features a cursor-based editor, local and global labels, and disk-based macros which permit you to incorporate frequently used subroutines into any program. Formprint program lets you print a formatted listing of source and object files. *#04609, Apple II Disk Version, $39.95.*

New!

MICROCOMPUTER CIRCUIT ANALYSIS PROGRAM

(Savon) Minimize your time calculating the frequency response of a circuit with this new program. MCAP performs a linear voltage, impedence, or transfer impedence analysis of an electronic circuit. Enter the circuit description in a systematic nodal notation and the program then calculates, lists and plots the circuit's frequency response. MCAP readily analyzes circuits with up to 15 nodes — larger circuits can be divided into subsections for individual analysis. And, the circuits analyzed can contain any or all of the six types of components: resistors, capacitors, inductors, bipolar transistors, FETs, and Op-Amps. Educators in particular will find this a strong applications program for circuit analysis. *#04504, Apple II; #04501, PET; each Cassette $24.95.*

Available at your local computer store!

Hayden Book Company, Inc.

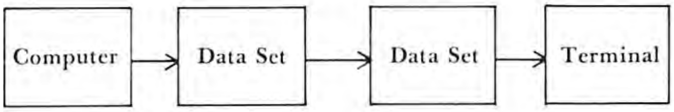
50 Essex Street, Rochelle Park, NJ 07662

or **Call Toll Free,** 24 hours a day,
(1-800-827-3777, ext. 302)*
TO CHARGE YOUR ORDER TO
Master Charge or Visa!
Minimum order is $10.00;
Customer pays postage and handling.
From Missouri, call
1-800-892-7655, ext. 302.

For those who prefer their ready-to-run programs

WELL DONE!

The figure below illustrates a 2-wire, point-to-point, half duplex, and a telephone leased line which is always available to the customer.



Assume that the computer needs to transmit a message to the terminal. The computer's software brings up the "data terminal ready line" to its data set. If the data set is "ON" it will return "Data Set Ready" (interlock it) to the computer. When the computer wants to transmit, it raises the "request to send" level which tells the data set to turn on the carrier wave. The carrier wave is sent from the computer's data set over the telephone circuit to the terminal's data set. The terminal's data set, upon detecting the carrier, will raise the "Received Line Signal Detector" level to inform the terminal, in effect, that a message is about to be received. After a fixed delay time (strappable in some data sets) and after raising "Request ToSend," the computer's data set will return, "Clear To Send." The computer upon receiving the "Clear To Send" signal, can now start transmitting the message, as marks and spaces, on the "Transmitted Data" line to its data set. The data set converts the digital signals into frequency or phase-- shifted signals for transmission over the leased line to the terminal's data set.

Most data sets contain a clamp circuit which clamps the "Received Data" line. "Received Line Signal Detector" level is not generated until after the carrier is detected. The clamp delay masks out all the possible noise on the line which occurs during the switching from either transmit to receive or receive to transmit.

If this was a synchronous operation, the clocking or synchronization of each bit would be done by the computer's data set. So that the computer knows when each bit must be placed on the "Transmitted Data" line, the data set sends clock to the computer on the "Transmission Signal Element Timing" line. This clock will be coincident with the leading adge of each data bit on the "Transmitted Data" line.

At the terminal end of the system, the computer's data set turns on its carrier; the terminal's data set detects it and sends "Carrier Detected" level to the terminals. Several milliseconds later (length determined by the "Clear To Send" delay in the transmitting data set), the first message bits arrive and are converted to a digital signal, which is passed from the data set to the terminal on the "Received Data" line. In synchronous operation, clocking for the data is generated by the receiving data set and is passed to the terminal on the "Receiver Signal Element Timing: line in order that the terminal can correctly clock the bits into its buffer or memory as they arrive. The clock pulse is timed to occur at the center of the data bit on the "Received Data" line.

The following is a list of the definitions of the RS232-C signals which are listed in order of pin number. To simplify the definitions, the transmitter of the message will be identified as the "transmitting terminal" and the receiver as the "receiving terminal."

PROTECTIVE GROUND PIN 1: This ground is electrically connected to the equipment frame. It may be connected to external grounds, as required.

TRANSMITTED DATA PIN 2: This signal is generated by the transmitting terminal and is transferred to the local transmitting data set for transmission of data to the receiving terminal. The transmitting terminal will hold "Transmitted Data" in marking condition during the intervals between characters or words, and at all times when no data are being transmitted.

In all systems, the transmitting terminal will not transmit data unless an ON condition is present on all of the following four signals:

1. Request To Send
2. Clear to Send
3. Data Set Ready
4. Data Terminal Ready

RECEIVED DATA PIN 3: This signal is generated by the receiving data set in response to data signals received from transmitting terminal via the transmitting data set. "Received Data" will be held in the binary one (marking) condition at all times when "Received Line Signal Detector" is in the OFF condition. This is called clamping the line.

On a half-duplex channel, "Received Data" signal will be held in the binary one (marking) condition when "Request To Send" is in the ON condition and for a brief interval following the ON to OFF transition of "Request To Send" signal to allow for the completion of transmission and the decay of line reflections. This is called squelch.

REQUEST TO SEND PIN 4: This signal is used to condition the data set for data transmission. On simplex channels or duplex channels, the ON condition maintains the data set in the transmit mode. The OFF condition maintains the data set in a non-transmit mode.

On a half-duplex channel, the ON condition maintains the data set in the transmit mode and inhibits the received mode. The OFF condition maintains the data set in the receive mode.

A transition from OFF to On instructs the data set to enter the transmit state which turns on the carrier. The data set responds by taking such action as may be necessary and indicates completion of such actions by turning ON "Clear To Send," thereby indicating to the terminal that data may be transferred on the interchange signal "Transmitted Data."

A transition from ON to OFF instructs the data set to complete the transmission of all data which was previously transferred on the interchange signal "Transmitted Data" and then assumes a non-transit mode or a receive mode, as appropriate. The data set responds to this instruction by turning OFF "Clear To Send" when it is prepared to again respond to a subsequent ON condition "Request To Send."

When "Request To Send" is turned OFF, it will not be turned ON again until circuit "Clear To Send" has been turned OFF by the data set.

An ON condition is required on "Request To Send" as well as on "Clear To Send," "Data Set Ready" and, where implemented, "Data Terminal Ready" whenever the transmitting terminal transfer data on the interchange signal "Transmitted Data."

It is permissible to turn "Request To Send" ON at any time when "Clear To Send" is OFF, regardless of the condition of any other interchange circuit.

CLEAR TO SEND PIN 5: A signal generated by the data set to indicate whether or not the data set is ready to transmit data.

The "Clear To Send" ON condition together with the ON condition of interchange signals "Request To Send," "Data Set Ready" and, where implemented, "Data Terminal Ready" will be transmitted to the communication.

The OFF condition is an indication to the transmitting terminal that it should not transfer data across the interface on interchange "Transmitted Data."

The ON condition of "Clear To Send" is a response to the occurrence of a simultaneous ON condition on "Data Set Ready" and "Request To Send" delayed as may be appropriate to the data set for establishing a data communication channel to a remote terminal (including the removal of the MARK HOLD clamps from the received data interchange circuit of the remote data set).

Where "Request To Send" is not implemented in the data set with transmitting capability, "Request To Send" shall be assumed to be in the ON condition at all times and "Clear To Send" will respond accordingly.

DATA SET READY PIN 6: This signal is used to indicate the status of the local data set. The ON condition of this signal is presented to indicate

SECONDARY RECEIVED DATA PIN 16: This circuit is equivalent to "Received Data" except that it is used to receive data on the secondary channel.

When the secondary channel is useable only for circuit assurance or to interrupt the flow of data in the primary channel, "Secondary Received Data" is normally not provided. See interchange "Secondary Received Line Signal Detector."

RECEIVER SIGNAL ELEMENT TIMING PIN 17: Signals on this circuit are used to provide the terminal with received signal element timing information. The transition from ON to OFF condition shall normally indicate the center of each signal element "Received Data." Timing information on "Receiver Signal Element Timing" shall be provided at all times when circuit "Received Line Signal Detector," is in the ON condition. It may, but need not, be present following the ON to OFF transition of "Received Line Signal Detector."

UNASSIGNED PIN 18: This pin may be used by the manufacturer for any purpose desired.

SECONDARY REQUEST TO SEND PIN 19: This signal is equivalent to "Request To Send" except that it requests the establishment of the secondary channel instead of requesting the establishment of the primary data channel.

Where the secondary channel is used as a backward channel, the ON condition of "Request To Send" will disable "Secondary Request To Send" and it will not be possible to condition the secondary channel transmitting data set to transmit during any time interval when the primary channel transmitting data set is so conditioned. Where system considerations dictate that one or the other of the two channels be in transmit mode at all times but never simultaneously, this can be accomplished by permanently applying an ON condition to "Secondary Request To Send" and controlling both the primary and secondary channels, in complementary fashion, by means of "Request To Send." Alternatively, in this case, "Secondary Clear To Send" need not be implemented in the interface.

When the secondary channel is useable only for circuit assurance or to interrupt the flow of data in the primary data channel, "Secondary Request To Send" will serve to turn ON the secondary channel carrier. The OFF condition of "Secondary Request To Send" will turn OFF the secondary channel carrier and thereby signal an interrupt condition at the remote end of the communication channel.

DATA TERMINAL READY PIN 20: This signal is used to control switching of the data set to the communication channel. The ON condition prepares the data set to be connected to the communication channel.

SIGNAL QUALITY DETECTOR PIN 21: Signals on this circuit are used to indicate whether or not there is a high probability of an error in the received data.

As ON condition is maintained whenever there is no reason to believe that an error has occurred.

An OFF condition indicates that there is a high probability of an error. It may, in some instances, be used to call automatically for the retransmission of the previously transmitted data signal. Preferably the response of this circuit shall be such as to permit identification of individual questionable signal elements on "Received Data."

RING INDICATOR (CE) PIN 22: The ON condition of this signal indicates that a ringing signal is being received on the communication channel.

DATA SIGNAL RATE SELECTOR PIN 23: Signals on this circuit are used to select between the two data signaling rates in the case of dual rate synchronous data sets or the two ranges of data signaling rates in the case of dual range non-synchronous data sets.

An ON condition shall select the higher data signaling rate or range of rates.

The rate of timing signals, if included in the interface, shall be controlled by this circuit as may be appropriate.

TRANSMIT SIGNAL ELEMENT PIN 24: Signals on this circuit are used to provide the transmitting data set with signal element timing information.

The ON to OFF transition shall nominally indicate the center of each signal element on "Transmitted Data." When "Transmit Signal Element Timing" is implemented in the data set, the data set shall normally provide timing information on "Transmit Signal Element Timing" whenever the data set is in a power on condition. It is permissible for the data set to withhold timing information on this signal for short periods provided "Request To Send" is in the OFF condition.

UNASSIGNED PIN 25: This pin may be used by the manufacturer for any purpose desired.

Although the "EIA" publishes an interface standard, some data set manufacturers do not conform to the standard in all cases. *CHECK* the specifications on each data set to determine which signals are on each pin. ©

Solving Equations With A Computer

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

INTRODUCTION

There is a large body of knowledge, known as "Numerical Analysis," that is used to solve problems that would be either too difficult or too inefficient to solve with either hand calculations or an electronic calculator. The problems generally attacked with numerical analysis techniques require either a computer or a programmable electronic calculator. The purpose of this article is to show how a few techniques from numerical analysis can be used to solve difficult equations. These techniques do not require any extraordinary mathematical skills; a first course in high-school algebra will suffice.

To begin, we will assume you can solve equations of the type,

$$2x + 5 = -3 \qquad (1)$$

This type of equation is solved using the rules:

RULE (1) The same number (or algebraic expression) can be added or subtracted from both sides of an equation.

RULE (2) Both sides of an equation can be multiplied or divided by any non-zero number (or algebraic expression).

Thus, in Equation (1), we would first subtract five from both sides of the equation and next both sides of the equation would be divided by two, giving $x = -4$ as the answer. Any equation of the form

$$Ax + B = C \qquad (2)$$

has a solution $x = (C - B)/A$, which is very easy to program in BASIC or FORTRAN. The program in Listing 1 does this.

Listing 1. Program to solve a linear equation.

```
10 INPUT A, B, C
20 X = (C - B)/A
30 PRINT X
40 END
```

Clearly in this case the problem could just as well have been done with pencil and paper. We are interested in more difficult problems, but RULES (1) and (2) above describe how equations may be modified to get the unknown "x" by itself on one side of the equation, and we will need these rules in what follows.

To these rules we add a third, namely

RULE (3) In certain cases both sides of an equation may be operated on by the same function and the results are still equal.

To illustrate, if $x^2 = 9$, then we may operate on both sides of this equation with the square root function (SQR in BASIC) to get $x = 3$. Note that this technique misses the answer $x = -3$, but it illustrated the fact that taking the square root of both sides of an equation (usually) yields a valid result. Likewise, one can take the logarithm (LOG in BASIC) of both sides of an equation provided we are dealing with positive numbers, and we can take the exponential function (EXP in BASIC) of both sides of an equation, using RULE (3).

The type of equations that are of interest in the present context can best be illustrated by some examples. How would you solve for x in the following equations:

$$x^2 = \cos(x) \qquad (3)$$

$$e^x - 4x = 0 \qquad (4)$$

$$\log(x) - \cos(x) = 0 \qquad (5)$$

These so-called non-linear equations cannot be solved by a simple application of the rules given so far. In fact, you may be disappointed to know that no single technique will solve all possible non-linear equations. Many people like mathematics because it seems to follow simple, hard-and-fast rules that lead to answers that are either right or wrong. On the contrary, mathematics requires creativity and the ability to view a problem from many angles. Furthermore, more often than not, the answers are only approximately correct rather than absolutely correct. In any case, let us examine two techniques that may be used to solve these difficult looking equations.

The Method Of Successive Substitutions

The method of successive substitutions is one of the simplest techniques used to solve these equations. It comes with no guarantee that it will work, but because it is simple it is frequently worth trying.

The first step is to take the equation to be solved and using the three rules given in the Introduction, put the equation in a form with x on the left-hand side and everything else on the right-hand side of the equation. For example, the equation $x^2 = \cos(x)$, Equation (3) above, becomes either $x = (\cos(x))/x$ or $x = \sqrt{\cos(x)}$. It is typical to find several possible forms. This step is usually described in textbooks by telling you to put your equation in the form

$$x = f(x) \qquad (6)$$

In our example, $f(x)$ is either $\cos(x)/x$ or $\sqrt{\cos(x)}$ depending on whether we are using $x = (\cos(x))/x$ or $x = \sqrt{\cos(x)}$. In any case, the equation is modified so that x is all by itself on one side of the equation and everything else is on the other side.

The second step is to *guess* at a value of x that will satisfy the equation. This is an important step because it may determine the success of the method. If you cannot make a reasonable guess by inspection of the equation or from some other source of information, then you can always have your computer print a table of x and f(x) to see where they are (almost) equal. For example, if you are trying to solve $x^2 = \cos(x)$ and you have completed the first step by transforming the equation to $x = \sqrt{\cos(x)}$, then use your computer to make a table of x and $\sqrt{\cos(x)}$. A few simple statements will suffice, as Listing 2 indicates. Be sure to be available to break the program because it has an infinite loop. The numbers in Table 1 were obtained with the program.

Listing 2. Program to compare x with f(x) for $x = \sqrt{\cos(x)}$.

```
10 X = 0
20 FX = SQR(COS(X))
30 PRINT X, FX
40 X = X + .1
50 GO TO 20
```

Table 1. Output of the program in Listing 2.

X	SQR(COS(X))
0.0	1.00
0.1	.99
0.2	.98
0.3	.97
0.4	.95
0.5	.93
0.6	.90
0.7	.87
0.8	.83
0.9	.78
1.0	.73

The values of SQR(COS(X)) in Table 1 have been truncated to two decimal places. Note in particular that at X = 0.8 the function f(x) = SQR(COS(X)) is 0.83 which is larger than X, while at X = 0.9 the function is .78 which is smaller than X. Thus, somewhere in between 0.9 and 0.8 the function will be equal to X, and the equation x = f(x) will be satisfied, giving us the answer. Thus, a good initial guess at a solution is either 0.8 or 0.9; either one will do.

The next step in solving the equation by the method of successive substitutions is to iterate. What this means is that we substitute our guess into f(x), getting a new value for x. If we call our first guess x_0 then our next guess is obtained from the equation

$$x_1 = f(x_0) \qquad (6)$$

and successive guesses (or approximations) are obtained from the following equations:

$$x_2 = f(x_1),$$
$$x_3 = f(x_2),$$

etc. All of this is handled in the program in Listing 3. Study this program to see how the process is done.

Listing 3. Program to iterate $x = \sqrt{\cos(x)}$.

```
10 X = 0.8
20 FX = SQR(COS(X))
30 PRINT FX
40 X = FX
50 GO TO 20
```

The results obtained from running the program in Listing 3 are given in Table 2. After three iterations (three times through the infinite loop) the answer (with a starting guess of 0.8 radians) is correct to two decimal places. After 15 times through the loop the answer is correct to six decimal places, namely 0.824132. Obviously one could build an "end" condition into the program. Suppose you want an answer correct to six decimal places. Inserting the statements:

```
35 IF ABS(X - FX) <.000001 THEN 60
:
60 END
```

would do the trick.

To illustrate the problems you can have, try solving the same equation using the form x = (cos(x))/x. Simply replace statement 20 in Listing 3 with

```
20 FX = (COS(X))/X
```

and run the program. Remember, this is the same equation that we are solving, but with a starting value of 0.8 radians you obtain the results given in Table 3. In this case, the answers do not get closer and closer to a solution, but the process diverges. Your luck has run out, but you were warned that the method does not always work. A way to tell if the method is going to work is available, but its explanation is beyond the scope of this article. Consult the reference at the end of this article.

Table 2. Results obtained with the form $x = \sqrt{\cos(x)}$.

0.8 = starting value
0.834689589
0.819394751
0.826234596
0.823194739
0.824549519
0.823946477
0.824215052
0.824095467
0.824148719
0.824125007
0.824135566
0.824130864
0.824132958
0.824132025
0.824132440
0.824132255

Table 3. Iteration results from x = (cos (x))/x

0.8 = starting value
0.870883387
0.739652527
0.998715996
0.542078343
1.58028501
-6.00432032E-03
-166.543742
5.99978489E-03
166.669642

One other illustration should suffice before we move to another technique. Consider Equation (4). It is not in the form x = f(x), but if we use RULE (1) and add 4x to both sides we get $4x = 3^x$. Using RULE (2) we divide both sides of the equation by four to get our required form, namely $x = (e^x)/4$. Replace statement 20 in Listing 3 with FX = EXP(X)/4 and pick a starting value of say X = 0. In 15 iterations you will have found a solution good to six decimal places; X = 0.3574029 (the trailing 9 may be uncertain). However, the flush of success may drain from your rosy cheeks when you realize that this equation has two answers, and the method of successive substitutions will not work to find the other answer.

How do we know that the equation has two answers? If you write a short program to print the value of e^x -4x for some values of x, you obtain the results in Table 4. Note that the function e^x - 4x is positive at x = 0.2 while it is negative at x = 0.4. That means that somewhere between 0.2 and 0.4 the function e^x - 4x went through zero, and at that point the equation was satisfied. That is the answer we found above, namely x = 0.3574029. Note also that at x = 2 the function is negative, while at x = 2.2 the function is positive, indicating that another answer is to be found between 2.0 and 2.2. Try to find this answer with successive substitutions.

Table 4. The value of e^x -4x versus x.

X	EXP(X) - 4*X
0.0	1.00
0.2	0.42
0.4	-0.11
0.6	-0.58
0.8	-0.97
1.0	-1.28
1.2	-1.45
1.4	-1.54
1.6	-1.44
1.8	-1.15
2.0	-0.61
2.2	0.22
2.4	1.42

The Method Of Interval Halving

The failure of the method of successive substitutions to converge to an answer in certain situations is reason enough to look for another method. The method of interval halving is particularly attractive because you are (almost) guaranteed that it will find an answer if you know that the answer lies between two numbers. Refer again to our problem of finding the solution to the equation e^x -4x = 0 and Table 4. Table 4 indicates that one answer is between 0.2 and 0.4 and another answer is between 2.0 and 2.2, *because the function e^x -4x changes sign on these intervals.* The first step in the interval halving method is to put the equation to be solved in the form

$$f(x) = 0 \qquad (7)$$

and to find two values of x (call them x_L and x_R, L and R for left and right) such that the function f(x) is positive for one of these values of x and it is negative for the other.

Suppose we deal with our example, e^x - 4x = 0. It already is in the form f(x) = 0. Furthermore, let us concentrate for the moment on the solution that we could not find with the method of successive substitutions. That solution we know to be between x = 2.0 and x = 2.2. Thus, x_L = 2.0 and x_R = 2.2

The second step is to try a value of x half-way between x_L and x_R. This is where the name "interval halving" originates. This value of x, call it x_M (M for middle) is given by the simple expression,

$$x_M = (x_L + x_R)/2 \qquad (8)$$

Now comes the tricky part. Suppose f_L is the value of the function when x_L is substituted (plugged in) into f(x), and suppose f_M is the value of the function when x_M is plugged into the function f(x). If the product ($f_L \cdot f_M$) is positive, then x_M lies to the left of the answer just like x_L. We know this because the product can only be positive if f_L and f_M have the same sign. In this case, we replace x_L with a new value, namely x_M. On the other hand, if the product ($f_L \cdot f_M$) is negative, then f_L and f_M have opposite signs, and x_M is to the right of the answer. In that case we replace x_R with x_M, giving a new value for x_R. In either case, we have bracketed the answer in an interval half as wide as the interval we started with.

Repeating this process allows us to bracket the answer in as small an interval as we wish, with the answer as accurate as we wish. Each time we calculate a new x_M we must test the sign of the product ($f_L \cdot f_M$) to see if x_M is to the right or left of the answer. If the answer was originally known to be in an interval of width w, where w is the difference between our first x_R and x_L, then after n iterations or repetitions of the interval halving process, the error in the answer is $w/2^n$. Thus, after 10 iterations the error in the answer is about 1/1000 of the original uncertainty in the answer.

A program to solve the equation e^x - 4x = 0 with the method of interval halving is given in Listing 4. With little modification, this program can be used for other equations as well. The variables in the program in Listing 4 are closely related to our previous discussion, so no further explanation will be given. I expect that most people can understand BASIC about as well as algebra. Table 5 shows the answers we obtain for both of the solutions to this equation. A total of 20 iterations are done in the program, giving an error of less than 0.00000096 if the distance between the original x_L and x_R were less than one. The roundoff error in many machines will exceed this.

Listing 4. Interval halving used to solve e^x - 4x. = 0.

```
10 INPUT XL, XR
20 FOR I = 1 TO 20
30 XM = (XL + XR)/2
40 FL = EXP(XL) -4*XL
50 FM = EXP(XM) = 4*XM
60 IF FL*FM<0 THEN 90
70 XR = XM
80 GO TO 100
90 XR 5 XM
100 PRINT XM
110 NEXT I
```

Are there any practical applications of these techniques for ordinary citizens? The answer is yes. Sup-

COMPUTE! HAS MOVED

Our New Telephone Number is
919 275-9809

We Are Now Located At
200 East Bessemer Avenue,
Greensboro, NC 27401

Our Mailing Address Is Unchanged
P.O. Box 5406
Greensboro, NC 27403

pose you are paying on a loan whose balance is BAL, using equal monthly payments called PMT, and you have N payments yet to make. What is the equivalent simple interest rate, called the APR, of your loan? The equation relating these quantities is

$$BAL = PMT((1 - (1 + I)^{-N})/I) \quad (9)$$

Table 5. Results of the program in Listing 4.

XL = 0.3, XR = 0.4	XL = 2.0, XR = 2.2
XM	XM
0.35	2.1
0.375	2.15
0.3625	2.175
0.35625	2.1625
0.359375	2.15625
0.3578125	2.153125
0.35703125	2.1546875
0.357421875	2.15390625
0.357226563	2.15351563
0.357324219	2.15332032
0.357373047	2.15322266
0.357397461	2.15327149
0.357409668	2.15329590
0.357403565	2.15328369
0.357400513	2.15328980
0.357402039	2.15329285
0.357402802	2.15329132
0.357403183	2.15329209
0.357402993	2.15329247
0.357402897	2.15329228

Note that in Equation (9), I is the monthly interest rate, and it must be multiplied by 1200 to convert it to an annual rate expressed in a percent form. In any case, I challenge you to solve Equation (9) by straightforward, direct techniques. Refer to the July/August issue of COMPUTE. for a solution of this equation by interval halving.

I would like to conclude this article by saying that you have only seen the tip of the iceberg as far as numerical analysis is concerned. One of the best elementary texts on this subject is Peter A. Stark's **INTRODUCTION TO NUMERICAL METHODS,** Macmillian, 1970. Note that many techniques require a knowledge of calculus. You may want to check your library for textbooks on the subject. One last plea: if you are a high school student who is planning a career in computer science, please get all of the courses in mathematics that your school offers. Although you do not have to be a mathematical genius to work in the computer field, every tool in the old toolbag will be helpful.

Appendix A.

The method of successive substitutions is guaranteed to converge to an answer if

$$|F'(x)| < 1$$

where x is any number in the interval between the first guess and the answer.

Appendix B.

The method of interval halving will not work in the somewhat unusual case of a double root to a polynomial equation. For example, if a factor of a polynomial equation is

$$(x - 1)^2 = x^2 - 2x + 1 = 0)$$

then the solution at x = 1 cannot be found with interval halving.

Appendix C.

One of the most popular iterative techniques is known as Newton's method or the Newton-Raphson method. It was not mentioned here because it requires a knowledge of calculus. The iterative formula is:

$$x_i + 1 = x_i - F(x)/F^1(i_i)$$

where it is assumed that the equation is initially in the form F(x) = 0. ©

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.

Train Your PET®
with MICRO-ED
instructional tapes

All programs work with any 8K PET, old or new. PET is the registered trademark for Commodore Business Machines, Santa Clara, CA.

MICRO-ED has educational software for the PET microcomputer. We specialize in programs with these features:

- They have been written by professional educators. Our main author is Thorwald Esbensen, named in 1980 as one of North America's top school executives by Executive Educator magazine.
- Our tapes are independent modules, each one a complete lesson in itself.
- Every instructional lesson ends with a summary of student performance.
- Our programs are attractive and motivating to students.
- We will gladly replace any tape that fails to load or run properly.

Unless otherwise specified, each tape can be purchased for $7.95

MUSIC (these tapes use sound)

- MU-1 Lines and Spaces of the Treble Clef (elementary)
- MU-2 Higher, Same, Lower (elementary)
- MU-3 Matching Rhythms (elementary)

REFERENCE SKILLS

- RS-1 Dictionary Guide Words (elementary)
- RS-2 Library Terms (elementary)
- RS-3 Making an Outline (elementary)
- RS-4 Putting Fiction Books in Alphabetical Order (elementary)

VOCABULARY

- VO-1 Synonym Series (high school and adult) 3 tapes $21.00
- VO-2 Vocabulary Series (upper elementary and high school) 24 tapes $168.00
- VO-3 Antonym Machine (elementary)
- VO-4 Homonym Machine (elementary)

You may wish to order tapes by the MICRODOZEN. Any twelve $7.95 tapes can be purchased for $84.00.

Computers and The Handicapped

Susan Semancik

Updates to Issue #5's Computers And The Handicapped Column:

Programs 2 and 3, written for the use of the Prestodigitizer Board with the handicapped, have been updated so that they will both run on either Old or New ROM PET Computers. In addition, they have both been expanded to allow punctuation to be entered, to allow the user to stop the programs without turning the computer off, to allow a space to be entered in order to separate letters or words in the messages appearing on the PET's screen, to allow individual letters to be deleted, and to allow the clearing of the entire screen - all from codes entered from the Prestodigitizer Board!

The last four modifications each required specialized codes to be developed in both Braille and Morse Code. Since program 3 uses only Level I Braille capital letters and punctuation, the four necessary codes were taken from Level II Braille and should pose no contradictions in this usage. They are illustrated below:

STOP	SPACE	DELETE	CLEAR
. ●	. .	. ●	. ●
. ●	. .	. .	. ●
. ●	. ●	. .	. .

The changes to program 3 are described below: The directions in Lines 1-8 reflect the above mentioned changes.

```
1 REM *** PROGRAM 3 - DIGITIZER BRAILLE ***
2 REM
3 REM WILL ACCEPT LETTERS, COMMA, PERIOD,
AND QUESTION MARK.
4 REM USE REGIONS 1-6 FOR THE BRAILLE CELL
INPUTS; REGION 7 TO END AN INPUT
5 REM FOR A SPACE, USE DOT 6
6 REM TO DELETE A CHARACTER, USE DOT 4
7 REM TO CLEAR THE SCREEN, USE 4 & 5
8 REM TO STOP THE PROGRAM, USE 4,5, & 6
```

Lines 9, 300, and 310 are necessary to determine the proper zero page locations for either an Old or New ROM PET.

```
9 P = PEEK(50003):Q = P*160:L = 200*P + 6
300 POKEQ,161:POKEQ + 1,3:POKEL,221:POKEL + 1,3
310 POKE863,L:POKE909,Q
```

Line 80 enters the end of the assembly language program into memory, storing the ASCII value in memory location 922 just before the character is printed.

```
80 DATA192,0,16,3,76,63,3,177,0,141,154,3,32,210,255,
96,-1
```

The disassembled listing would then be changed as follows:

```
910: STA 922
     JSR 65490
     RTS
```

By returning to the BASIC program after printing, line 410 will check to see if location 922 contains a 96, which will end the program. Since 96 represents a shifted space, this will not affect the appearance of what has been printed on the screen.

400 PRINT""
410 SYS(826):IFPEEK(922)$_{78}$96THEN410
420 END

Lines 130 and 140 contain the ASCII of space, delete, clear, and shifted space in the appropriate locations.

```
130 DATA0,65,20,67,44,66,73,70,0,69,147,68,0,72,74,71,
0,75,0,77,76,83,80,0
140 DATA79,0,78,0,82,84,81,32,0,0,0,0,0,0,0,0,96,0,46,
0,87,0,0,85,0,88,63,86
```

The four new necessary Morse codes were taken from specialized vowels that would not ordinarily be used in this type of communication program. They are listed as follows:

STOP	SPACE	DELETE	CLEAR
---.	..-..	..--	.-.-

The changes to program 2 are described below: The directions in Lines 1-8 reflect the above mentioned changes.

```
1 REM *** PROGRAM 2 - DIGITIZER MORSE ***
2 REM
3 REM WILL ACCEPT LETTERS, COMMA, PERIOD,
AND QUESTION MARK.
4 REM USE REGION 1 TO INPUT A DOT; REGION 2
FOR A DASH, REGION 7 TO END INPUT
5 REM FOR A SPACE, USE ..-..
6 REM TO DELETE A CHARACTER, USE ..--
7 REM TO CLEAR THE SCREEN, USE .-.-
8 REM TO STOP THE PROGRAM, USE ---.
```

Lines 9 and 130 are necessary to determine the proper zero page locations for either an Old or New ROM PET.

```
9 P = PEEK(50003):Q = P*160
130 POKE5254,4:POKE5255,5:POKEQ,136:POKEQ + 1,19:
POKE917,Q:POKE937,Q
```

Lines 60-80 enter the end of the assembly language program into memory, with the look-up table pointing to the character with the lowest ASCII value used, which is 20 for the Delete key.

```
60 DATA76,76,3,169,20,141,216,3,160,0,174,215,3,177 ,0,
205,215,3,240,12,200,200
70 DATA238,216,3,192,0,240,17,76,145,3,200,177,0,205,
213,3,208,237
80 DATA173,216,3,32,210,255,96,-1
```

This changes the disassembled listing:

```
906: LDA# 20
```

Since location 984 is keeping track of the ASCII value of the character pointed to in the table, it needs to be increased everytime the pointer moves up in the table. Also, adding more table values affects the table limit to be checked.

```
925: INC 984
     CPY# 0
     BEQ +17
```

Since the ASCII values have kept up with the pointer, adding is no longer needed. The ASCII value of the character is placed in the accumulator for printing.

```
943: LDA 984
     JSR 65490
     RTS
```

By returning to the BASIC program after printing, Line 140 will check to see if location 984 contains a 96, which will end the program. Since 96 represents a shifted space, the appearance of the screen has not been affected.

```
135 PRINT"  ";
140 SYS(826):IFPEEK(984)<>96THEN140
150 END
```

Lines 110 and 123 insure all the unused table values will be set to zero.

```
110 READOP:IFOP = -1THEN123
123 FORI = 5000TO5255:POKEI,0:NEXTI
```

Lines 125 and 126 set the alphabetic characters in the proper place in this expanded table.

```
125 I = 5090
126 READOP:IFOP = -1THEN128
```

Lines 128-130 set the punctuation characters and the four special characters in the proper table locations.

```
128 POKE5000,4:POKE5001,3:POKE5024,5:POKE5025,4:
POKE5048,6:POKE5049,51
129 POKE5052,6:POKE5053,21:POKE5086,6:POKE5087,
12:POKE5152,4:POKE5153,14
```

These programs were also tried out at the Marine Science Center's Communication's Workshop for the handicapped. The blind students in particular were excited about the digitizer pad and were able to communicate to the deaf through messages entered on the PET's screen. Using these programs in conjunction with some other equipment we've been experimenting with gave us some fascinating results that we hope to be able to share with you in the next issue of COMPUTE! ©

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

Let Your Pet Play Politics with HAT IN THE RING

A Presidential Election Game

Thorwald Esbensen

Here is a timely social studies game that readers of COMPUTE have permission to copy for their own personal use. The program will run on any 8K PET, old or new.

HAT IN THE RING is a two-player exercise designed to acquaint students with some of the political considerations involved in running a presidential campaign. Each player assumes the role of a presidential candidate - one for the Republicans, the other for the Democrats. Throughout the exercise, each candidate makes decisions intended to result in a successful campaign.

At the outset, each candidate has 9 units of priority resources that can be assigned as needed in order to bolster the campaign in any of the states. The overall campaign ends after each candidate has made, in alternating turns, 10 decisions.

Within each state, the outcome of the campaign hinges upon four factors:

- Media Exposure
- Personal Campaigning
- Domestic Issues
- International Issues

Although the weight of these four factors is randomly determined by the computer, the probabilities are that Media Exposure and Domestic Issues will prove to be substantially more powerful in their impact than will the factors of Personal Campaigning and International Issues. The political situation in each state keeps changing as the game progresses.

As the campaign begins, the computer randomly chooses the candidate who will have the first turn. The computer may be commanded to do one of six things:

1. Raise funds (increase resources).

2. List the states in which the Republican candidate leads.

3. List the states in which the Democratic candidate leads.

4. List the current probable electoral count for each candidate.

5. List each candidate's remaining resources.

6. Get ready to display the political situaiton in state.

The 9 resource units that each candidate has at the beginning of the game are the maximum allowed. So there is no point in asking the computer to raise more funds (resources) until some of these units have been used up. However, when the computer *does* try to raise funds, it will yield a result of from 0 to 3 additional resource units.

Let us say that the candidate chooses Command Number 6. The computer now asks the candidate for the name of a state, and then displays the political situation in that state. The number immediately following the name of a state represents the number of electoral votes that the state can cast for a presidential candidate. The candidate with the highest total of political points in a state (the combined points for Media Exposure, Personal Campaigning, Domestic Issues, and International Issues) will win all of that state's electoral votes at the end of the game.

The candidate can affect the political situation in any state by committing some resource units to that state. When resources are committed, they have a multiplier effect on the category to which they are committed. For example, if the strength of Media Exposure is a given state is 15, and if 3 resource units are then committed to that category in this state, the new Media Exposure strength in the state becomes 15 x 3, or 45. If a candidate commits more resources than the candidate has, this blunder automatically results in the election of the opponent.

The fifty states, plus the District of Columbia (abbreviated D.C.), have 538 electoral votes in all. At the end of the game, the candidate with a majority of these electoral votes (270 or more) wins the game.

Readers who do not wish to spend their time copying the following program listing may spend $9.95 for a copy of the program tape itself from MICRO-Ed, Inc., P.O.Box 24156, Minneapolis, Minnesota, 55424.

```
9 POKE59468,12
10 PRINT"ĥ↓↓↓↓↓→→→→→→→→→→COPYRIGHT 1980":
     ¬PRINT"↓→→→→→→→→→→ɼMICRO-EDȓ, INC."
12 PRINT"↓→→→→→→→→→→P.O. BOX 24156"
13 PRINT"↓→→→→→→→→→→MINNEAPOLIS, MINNESOTA ¬
     ¬55424"
30 NM=INT(RND(TI)*2)+1:IFNM=1THENN1$="REPU
     ¬BLICAN":N2$="DEMOCRATIC"
40 IFNM=2THENN1$="DEMOCRATIC":N2$="REPUBLI
     ¬CAN"
60 SP$="                                        ¬
     ¬    "
80 DIMST$(52),EL%(51),P%(51),PP%(51),
     ¬M%(51),MM%(51),D%(51),DD%(51)
```

```
85 DIMT%(51),TT%(51),I%(51),II%(51):R=9:
     ¬RR=9:FORZ=1TO2000:NEXTZ:TU=1
100 PRINT"ĥ↓↓↓"
102 PRINT"ṟ→→→****************************
     ¬****
103 PRINT"ṟ→→→* Q#P                        ¬
     ¬Q#P  *
104 PRINT"ṟ→→→*$L$:$- HAT IN THE RING ¬
     ¬-$L$:$ *
105 PRINT"ṟ→→→*                              ¬
     ¬    *
106 PRINT"ṟ→→→* A PRESIDENTIAL ELECTION ¬
     ¬GAME *
107 PRINT"ṟ→→→*                              ¬
     ¬    *
108 PRINT"ṟ→→→****************************
     ¬****
110 PRINT"ṟ↓↓→→→READ PRINTED INSTRUCTIONS ¬
     ¬FIRST."
120 GOSUB19000
130 FORZ=1TO51:READST$(Z),EL%(Z):NEXTZ:
     ¬READM$,P$,D$,I$
150 PRINT"ĥṟ↓↓↓↓↓→→→→PREPARING POLITICAL ¬
     ¬SITUATION"
200 FORZ=1TO51:M%=INT(RND(TI)*10)+1
207 M%(Z)=M%(Z)+M%
210 P%=INT(RND(TI)*5)+1
212 P%(Z)=P%(Z)+P%
220 D%=INT(RND(TI)*10)+1
222 D%(Z)=D%(Z)+D%
225 I%=INT(RND(TI)*5)+1
227 I%(Z)=I%(Z)+I%
235 MM%=INT(RND(TI)*10)+1
237 MM%(Z)=MM%(Z)+MM%
240 PP%=INT(RND(TI)*5)+1
242 PP%(Z)=PP%(Z)+PP%
245 DD%=INT(RND(TI)*10)+1
247 DD%(Z)=DD%(Z)+DD%
250 II%=INT(RND(TI)*5)+1
252 II%(Z)=II%(Z)+II%
260 T%(Z)=M%(Z)+P%(Z)+D%(Z)+I%(Z)
265 TT%(Z)=MM%(Z)+PP%(Z)+DD%(Z)+II%(Z):
     ¬NEXTZ
280 K=0:T1=0:T2=0
285 IFY1+Y2=20THENTU=10:GOTO4000
290 IFC=2THENC=0
300 C=C+1:IFC=1THENPRINT"ĥṟ";N1$:Y1=Y1+1
310 IFC=2THENPRINT"ĥṟ";N2$:Y2=Y2+1
315 TN=TN+1:IFTN=3THENTU=TU+1:TN=1
320 PRINT"ṟCANDIDATE'S TURN NO.";TU:
     ¬PRINT"ṟ(CHOOSE NUMBER BELOW)"
325 PRINT"EEEEEEEEEEEEEEEEEEEEEEEEEEEE":
     ¬PRINT"ṟ1r̂ RAISE FUNDS (INCREASE ¬
     ¬RESOURCES)"
335 PRINT"↓ṟ2r̂ LIST THE STATES IN WHICH ¬
     ¬THE":PRINT"↓→→";N1$;" CANDIDATE ¬
     ¬LEADS."
340 PRINT"↓ṟ3r̂ LIST THE STATES IN WHICH ¬
     ¬THE":PRINT"↓→→";N2$;" CANDIDATE ¬
     ¬LEADS."
345 PRINT"↓ṟ4r̂ LIST CURRENT ELECTORAL ¬
     ¬COUNT FOR":PRINT"↓→→EACH CANDIDATE."
350 PRINT"↓ṟ5r̂ LIST CANDIDATES' REMAINING ¬
     ¬RESOURCES."
355 PRINT"↓ṟ6r̂ GET READY TO DISPLAY ¬
     ¬SITUATION":PRINT"↓→→FOR A CERTAIN ¬
     ¬STATE."
357 GETG$:IFG$<>""THEN357
360 GETG$:IFVAL(G$)<1ORVAL(G$)>6THEN360
365 IFG$="1"THEN1000
370 IFG$="2"THEN2000
375 IFG$="3"THEN3000
380 IFG$="4"THEN4000
385 IFG$="5"THEN5000
390 IFG$="6"THEN6000
```

```
1000 PRINT"ĥ":NR=INT(RND(TI)*4)
1017 IFC=1THENR1=R:R=R+NR
1020 IFC=2THENR2=RR:RR=RR+NR
1030 IFR>9THENR=9:NR=R-R1
1035 IFRR>9THENRR=9:NR=RR-R2
1040 PRINT"r̲↓↓↓↓INCREASE IN YOUR RESOURCE ¬
      ¬UNITS:r̂";NR
1060 GOSUB19000:GOTO150
2000 PRINT"ĥ";N1$;" CANDIDATE LEADS:↓":
      ¬FORZ=1TO51
2020 IFT%(Z)>TT%(Z)THENPRINT"r̲";ST$(Z);EL%
      ¬(Z):CT=CT+1
2030 IFCT=20THENGOSUB19000:CT=0:PRINT"ĥ"
2040 NEXTZ
2050 CT=0:GOSUB19000:GOTO150
3000 PRINT"ĥ";N2$;" CANDIDATE LEADS:↓":
      ¬FORZ=1TO51
3020 IFT%(Z)<TT%(Z)THENPRINT"r̲";ST$(Z);EL%
      ¬(Z):CT=CT+1
3030 IFCT=20THENGOSUB19000:CT=0:PRINT"ĥ"
3040 NEXTZ
3050 CT=0:GOSUB19000:GOTO150
4000 PRINT"ĥ":FORZ=1TO51
4020 IFT%(Z)>TT%(Z)THENT1=T1+EL%(Z)
4025 IFT%(Z)<TT%(Z)THENT2=T2+EL%(Z)
4030 NEXTZ
4032 IFY1+Y2<>20THEN4040
4035 IFTU=10ANDT1>T2THENPRINT"ĥ↓↓r̲";N1$;"→
      ¬CANDIDATE WINS!!":END
4037 IFTU=10ANDT1<T2THENPRINT"ĥ↓↓r̲";N2$;"→
      ¬CANDIDATE WINS!!":END
4040 PRINT"r̲↓";N1$:PRINT"↓CURRENT ¬
      ¬ELECTORAL COUNT:";T1
4050 PRINT"r̲↓↓↓";N2$:PRINT"↓CURRENT ¬
      ¬ELECTORAL COUNT:";T2
4060 PRINT"↓↓":GOSUB19000:GOTO150
5000 PRINT"ĥ↓";N1$;" RESOURCES:";R:
      ¬PRINT"↓↓";N2$;" RESOURCES:";RR
5010 PRINT"↓↓":GOSUB19000:GOTO150
6000 PRINT"ĥWHICH STATE DO YOU WANT";
6020 INPUT"→→2←←←";ST$
6030 IFST$="2"THEN6000
6040 K=K+1
6050 IFST$(K)=ST$THEN6070
6060 IFK>51THENK=0:PRINT"↓↓NO SUCH STATE. ¬
      ¬TRY AGAIN.↓↓":GOTO6020
6065 GOTO6040
6070 PRINT"ĥ"
6080 PRINT"r̲";N1$
6085 PRINT"↓SITUATION IN"
6090 PRINT"r̲↓";ST$;EL%(K)
6100 PRINT"EEEEEEEEEEEEEEEE
6110 PRINTM$;M%(K):PRINT"↓";P$;P%(K):
      ¬PRINT"↓";D$;D%(K):PRINT"↓";I$;I%(K)
6120 TP$="r̲↓↓TOTAL POINTS "
6125 PRINTTP$;"r̂";T%(K)
6170 PRINT"h̲":PRINTTAB(20);"r̲";N2$:
      ¬PRINTTAB(20);"↓SITUATION IN":
      ¬PRINTTAB(20);
6190 PRINT"r̲↓";ST$;EL%(K)
6195 PRINTTAB(20);
6200 PRINT"EEEEEEEEEEEEEEEE
6205 PRINTTAB(20);M$;MM%(K)
6210 PRINTTAB(20);"↓";P$;PP%(K)
6220 PRINTTAB(20);"↓";D$;DD%(K)
6230 PRINTTAB(20);"↓";I$;II%(K)
6232 PRINTTAB(20);
6235 PRINTTP$;"r̂";TT%(K)
6300 PL$="h̲↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓"
6305 PRINTPL$
6310 PRINT"WILL YOU COMMIT RESOURCES? ¬
      ¬(r̲Yr̂ OR r̲Nr̂)"
6320 GETG$:IFG$<>"Y"ANDG$<>"N"THEN6320
6330 IFG$="N"THEN150
6335 PRINTPL$:PRINTSP$
6350 PRINTPL$:PRINT"TO WHICH CATEGORY? ¬
      ¬(r̲M→P→D→r̂OR r̲Ir̂)"
6360 GETC$:IFC$<>"M"ANDC$<>"P"ANDC$<>"D"AN
      ¬DC$<>"I"THEN6360
6365 PRINTPL$:PRINTSP$
6370 PRINTPL$:PRINT"HOW MANY? (r̲1r̂ TO ¬
      ¬r̲9r̂)"
6380 GETR$:IFVAL(R$)<1THEN6380
6400 IFC=1THENR=R-VAL(R$)
6410 IFR<0THEN8000
6420 IFC=2THENRR=RR-VAL(R$)
6430 IFRR<0THEN8500
6450 IFC=2THEN6700
6500 IFC$="M"THENM%(K)=M%(K)*VAL(R$)
6510 IFC$="P"THENP%(K)=P%(K)*VAL(R$)
6520 IFC$="D"THEND%(K)=D%(K)*VAL(R$)
6530 IFC$="I"THENI%(K)=I%(K)*VAL(R$)
6550 GOTO150
6700 IFC$="M"THENMM%(K)=MM%(K)*VAL(R$)
6710 IFC$="P"THENPP%(K)=PP%(K)*VAL(R$)
6720 IFC$="D"THENDD%(K)=DD%(K)*VAL(R$)
6730 IFC$="I"THENII%(K)=II%(K)*VAL(R$)
6750 GOTO150
7000 DATA ALABAMA,9,ALASKA,3,ARIZONA,6,
      ¬ARKANSAS,6
7010 DATA CALIFORNIA,45,COLORADO,7,
      ¬CONNECTICUT,8
7020 DATA DELAWARE,3,D.C.,3
7030 DATA FLORIDA,17,GEORGIA,12,HAWAII,4
7050 DATA IDAHO,4,ILLINOIS,26,INDIANA,13,
      ¬IOWA,8
7060 DATA KANSAS,7,KENTUCKY,9
7070 DATA LOUISIANA,10
7080 DATA MAINE,4,MARYLAND,10,MASSACHUSETT
      ¬S,14,MICHIGAN,21,MINNESOTA,10
7090 DATA MISSOURI,12,MISSISSIPPI,7,
      ¬MONTANA,4
7100 DATA NEBRASKA,5,NEVADA,3,NEW ¬
      ¬HAMPSHIRE,4,NEW JERSEY,17,NEW ¬
      ¬MEXICO,4
7110 DATA NEW YORK,41,NORTH CAROLINA,13,
      ¬NORTH DAKOTA,3
7120 DATA OHIO,25,OKLAHOMA,8,OREGON,6
7130 DATA PENNSYLVANIA,27
7140 DATA RHODE ISLAND,4
7150 DATA SOUTH CAROLINA,8,SOUTH DAKOTA,4
7160 DATA TENNESSEE,10,TEXAS,26
7170 DATA UTAH,4
7180 DATA VIRGINIA,12,VERMONT,3
7190 DATA WASHINGTON,9,WEST VIRGINIA,6,
      ¬WISCONSIN,11,WYOMING,3
7200 DATA "r̲Mr̂EDIA↓←←←←←EXPOSURE.....",
      ¬"r̲Pr̂ERSONAL↓←←←←←←←←CAMPAIGNING.."
7205 DATA "r̲Dr̂OMESTIC↓←←←←←←←←ISSUES......
      ¬."
7210 DATA "r̲Ir̂NTERNATIONAL↓←←←←←←←←←←←←←IS
      ¬SUES......."
8000 PRINT"ĥr̲↓↓↓→→→SORRY.  YOU HAVE ¬
      ¬OVER-COMMITTED YOUR"
8010 PRINT"r̲↓RESOURCES. THE ";N2$;" ¬
      ¬CANDIDATE WINS"
8020 PRINT"↓↓→→→→→→→→→→r̲END→OF→GAME":END
8500 PRINT"ĥr̲↓↓↓→→→SORRY.  YOU HAVE ¬
      ¬OVER-COMMITTED YOUR"
8510 PRINT"r̲↓RESOURCES. THE ";N1$;" ¬
      ¬CANDIDATE WINS"
8520 PRINT"↓↓→→→→→→→→→→r̲END→OF→GAME":END
19000 PRINT"↓→→→→→PRESS r̲SPACE→BARr̂ TO ¬
      ¬CONTINUE"
19010 GETG$:IFG$<>" "THEN19010
19020 RETURN
```

©

THE FIRST ANNUAL COMPUTER PROGRAMMING CONTEST (of Herkimer, NY)

E. Q. Carr
Planetarium Director
Herkimer BOCES Planetarium
Herkimer, NY 13350

They came from 40 miles away. Some brought their own PET's with large keyboards because they wanted no part of our little keyboards. The lone Apple II arrived and fortunately we had planned a second one for them. These kids were confident, a big short of cocky, quite determined and planned to win a timed problem solving contest. Three-and-a-half hours later they left, confident, determined to win the next contest, perhaps in college.

And during that time, the 31 contestants did warm-up exercises, engaged in a two problem contest, polished off 90 hamburgers, listened enthusiastically to lectures on Fortran and a slide shown on the history of digital computer technology, received their prizes and visited a minimicrocomputer faire set up by local computer vendors. The actual time schedule appears in Table I.

Why Programming Contests

I do not know of research which indicates the mind increases its power and capacity by competition. It is obvious however, that a system of proper instruction and training with competition produces desirable results. The programs for the athletically apt are a paradigm worthy of imitating. Athletics produces very little in the export market to aid in the balance of payments however. But computer technology and software has income value to the country in terms of billions of dollars over a long period of time.

Moreover, discipline, challenge and association with peers n a competition is a tool for self-calibration. Then, there is the exhilaration of stretching to the limit of one's inherent capabilities. In a contest, the kids grow in stature and self-esteem, and that's obvious from even a cursory observation.

The practice and training for competition is, of course, the most valuable part of it all. All the basics must be in place, techniques studied and reviewed, some of the simple algorithms FOR-NEXT loops, IF-THEN, logic statements AND, OR, NOT must be ready tools, used without hesitation.

Organization

The idea of a contest appealed to the experienced computer teachers in our areas who responded to a phone call sampling 15 schools known to have microcomputers, or terminals accessing a mainframe. A mailing for organization went out three months before the expected contest date. Six teachers and two community college students attended that first meeting. They were the nucleus for sample problems demonstrating the potential skill of contestants. A member of this group decided there should be prizes for the winners and undertook getting contributions. The college students offered to generate a range of problems for the contest. The residue of tasks consisted of the physical facilities, speakers, publicity, registration, the mini-faire organization, correspondence, refreshments, contest rules, judging, selecting a final date, arranging computers and contacting suppliers for literature.

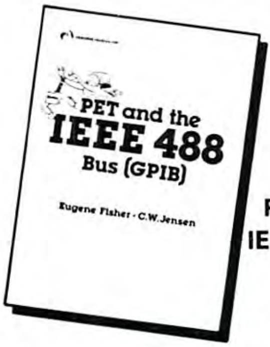
Of course, we had two months for all that.

Registration

With a date selected, a Saturday in April, there was a hope that there would be no student events in conflict. It proved an unfortunate choice. We lost a number of local schools' contestants to a track meet. Which may demonstrate something about the nature of a number of kids involved. They are also athletes. Within a week, all but 2 teams had sent the registration forms, and the final pre-contest registration was 32 students.

Sudden Death

The committee agreed that contestants would be paired as teams. There were several reasons, but fundamentally, the purpose was to assure that the maximum



PET and the IEEE 488 Bus (GPIB)
by E. 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 PET edition

Some Common BASIC Programs
by L. Poole, M. Borchers, C. Donahue

76 Programs you can use even if you don't know BASIC. This book gives you a variety of math power including personal finance, taxes and statistics as well as other programs you'll want like Recipe Cost and Check Writer. All programs can be run on a PET or CBM with 8K or more.
#40-3 $12.50

PET owners can purchase the programs ready-to-run on cassette or disk. Use the book as a manual for operating instructions and programming options.
Disk #33-0 $22.50
Cassette #25-X $15.00

Practical BASIC Programs
ed. Lon Poole

These are 40 easy to use programs that each do something useful.

Income averaging, checkbook reconciliation, statistics, factorials, temperature conversion and musical transposition are just a few. It offers a wealth of practical computing power. Includes write-ups, program notes and instructional examples to help you realize the potential uses of each program.
#38-1 $15.00

6502 Assembly Language Programming
by L. Leventhal

Increase the capabilities and performance of PET (and other 6502-based computers) by learning to program in assembly language.
#27-6 $12.50

New for your PET

from

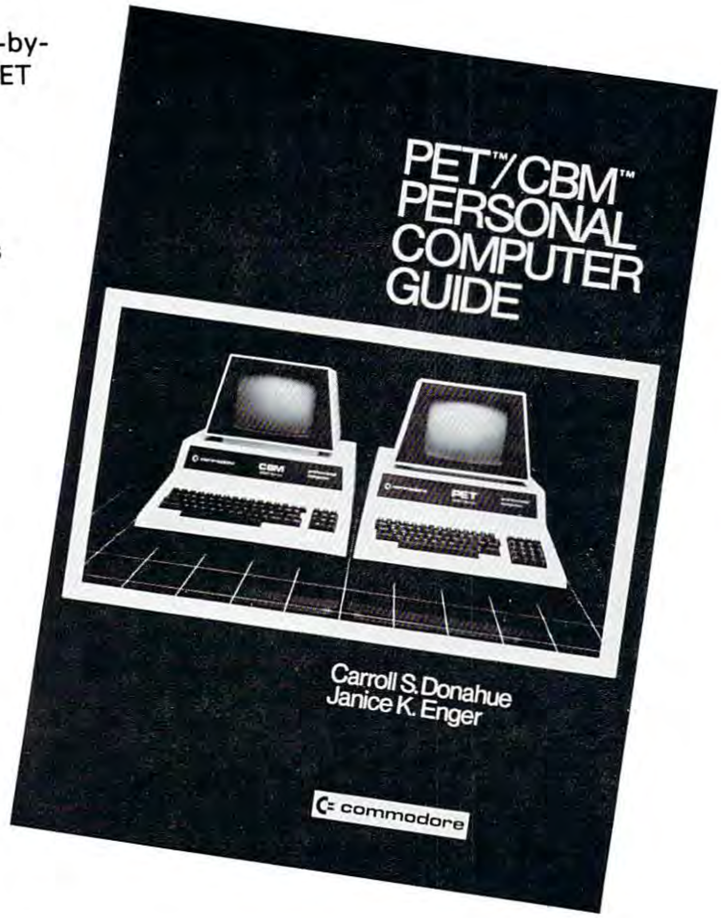
OSBORNE/McGraw-Hill

PET Personal Computer Guide
by C. Donahue and J. Enger

Everything you always wanted to know about PET/CBM computers . . . but don't.

This book is a step-by-step guide to the PET computer.

Assuming no prior knowledge of computers this PET guide contains a wealth of information that you'll need in training your PET to perform.



#30-6 $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		
40-3 Some Common BASIC Programs PET/CBM ed. (book)	$12.50		
25-X Some Common BASIC Programs PET Cassette	$15.00		
33-0 Some Common BASIC Programs PET Disk	$22.50		
38-1 Practical BASIC Programs	$15.00		

California residents add 6% sales tax — California resident tax
S.F. BART residents add 6½% sales tax
Shipping: (Shipping for large orders to be arranged) — Shipping
Total Amount Enclosed

- 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 √06
630 Bancroft Way, Dept. G3
Berkeley, California 94710
(415) 548-2805 · TWX 910-366-7277

Name: ____________________

Address: ____________________

City: ____________________

State: ____________ ZIP: ____________

Phone: ____________________

number of teams possible complete the problems, and to permit more students to participate. It was expected that the less confident students would be supported and decide to enter.

The sudden death aspects of a single problem contest prompted a two problem contest based on a winning team with the minimum accumulated solution time. Students were cautioned that while the contest was a timed power test, it was the first correct solution that drew a time mark. Incorrect answers meant a maximum time, that is, 20 minutes allotted to each problem.

The Mini-Microfaire

All four of our local computer stores agreed to participate. So did four local colleges and two large computer manufacturers' representatives. One of the vendors volunteered to contribute to the prizes. There was no charge for vendors.

How the Contest Worked

About a third of the contestants arrived early, as did all the Faire vendors who participated. Two vendors and two college representatives failed to show however. The half hour "warm-up" period at the start is more correctly a "set-up period". Still, it was interesting to see kids set up their own exercises on the machines. One brought a memory test tape and nearly created a panic when he elaborated it would take 15 minutes to fully check the machine's memory, and how sorry he was no one else could be sure their computer was working. Another set up beautiful graphics of animated rain clouds that moved across the screen. Each team was moving in different and original patterns that reinforced confidence. This alone was an exciting phase of the contest for an observer.

One of the teachers had taken on the task of Contest Commissioner whose function was to run the actual programming contest. He distributed the test problems, face down to each of the 15 computer stations and on a signal, the student hit RETURN, starting the PET's internal clock. He supervised the master clock and acted as the referee of referees.

Referees were the teachers and sponsors of the contestants. They responded to the students who indicated they had solved the contest problem. Referees determined whether a solution within the question statement and rules had actually been achieved.

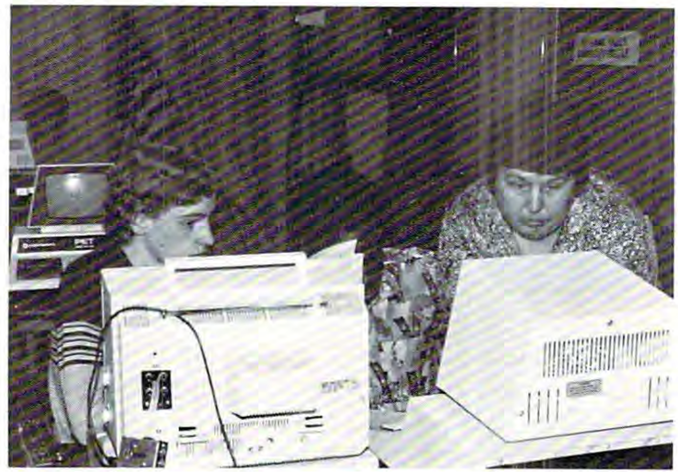
Analysis of Contest Problem Results

The first problem was selected to have a solution in approximately 12 single statement lines for an average student and a solution time of 5 minutes. The second problem was judged to be of greater difficulty. The questions, possible solution and requirements are given in Table II. The scoring times are plotted in Table III for each problem. I believe the data indicates a wide range in student skills. There may be other implications as well, but the paucity of data precludes generalizations.

The problems however met several contest design goals that included maintaining student confidence by permitting every student to complete the first problem. The first problem assumed fast students would complete the problem in 5 minutes or less. The second problem was designed for approximately a 10 minute solution time. These goals were met.

Reprise

The contest was an exciting event for students, vendors and teachers alike. The lessons we learned will make it a better contest for the kids. Indeed the Contest Committee is already at work!



FIRST ANNUAL COMPUTER PROGRAMMING CONTEST
(of Herkimer, NY)

CONTEST RULES (1980)

1. A maximum of twenty (20) teams can be accommodated on a First Registered, First Served Basis. A maximum of three (3) teams, but to accommodate the greatest number of schools we may limit a school to two (2) teams.

2. A Team consists of two (2) members. Each team will be given two (2) problems to solve. The winning team will have the lowest accumulated total time to problem solution.

3. Solutions will be checked by Referees with data entry on spearate lines. Please use line numbers spaced by tens of units (10, 20, 30 ....).

4. Contestants will have thirty (30) minutes beginning at 10 a.m. for familiarization with the PET 2001s.

5. The programs are limited to the following list of BASIC statements, commands, etc.

A. INPUT	F. DIM	K. ON	P. TI, TI$
B. READ	G. PRINT	L. ON ... GOTO	Q. AND
C. DATA	H. GOTO	M. GO SUB	R. OR
D. REM	I. IF ... THEN	N. RETURN	S. NOT
E. LET	J. FOR ... NEXT	O. END	

NO OTHER COMMANDS WILL BE ALLOWED

6. When you have reached a solution and have checked it carefully, print the statement PRINT TI$, hit RETURN and call a Referee.
Remember, once you've hit RETURN, you can no longer change the program.

7. Schools are encouraged to use local contests to select their teams.

PROBLEM #1

THE FOLLOWING SCHEDULE OF LICENSE FEES IS PROPOSED TO PERSUADE PEOPLE TO SAVE GASOLINE BY USING SMALLER ENGINES IN CARS.

HORSEPOWER	LICENSE FEE
Up to 20 HP	$ 0
More than 20 HP, but 40 HP or less	$ 50
More than 40 HP, but 60 HP or less	$ 200
More than 60 HP, but 80 HP or less	$ 800
More than 80 HP	$10,000

THE SOLUTION SHOULD BEGIN:

1. Prompt an input of AUTO HP?
2. Print an output such as THE LICENSE FEE IS $50
3. Return to the original prompt.

A POSSIBLE SOLUTION

```
3 PRINT"3"
10 INPUT"INPUT AUTO HORSE POWER";P
20 IF P<=20 THEN 200
30 IF P<=40 THEN 300
40 IF P<=60 THEN 400
50 IF P<=80 THEN 500
60 IF P>80  THEN 600
200 F=0
210 GOTO 1000
300 F=50
310 GOTO 1000
400 F=200
410 GOTO 1000
500 F=800
510 GOTO 1000
600 F=10000
1000 PRINT"LICENSE FEE IS $";F
1010 GETA$:IF A$=""THEN 1010
1020 GOTO 3
```

Solution Requirements:

Equivalent of Lines 10, 1000, and 1020 are required.
The equivalent of Line 3 is a tie breaker.
Return to Line 10 from 1020 is acceptable.

PROBLEM #2

WRITE A PROGRAM THAT WILL TAKE THE COORDINATES OF TWO POINTS IN A PLANE, COMPUTE THE DISTANCE BETWEEN THEM, GIVE THE COORDINATES OF THE MIDPOINT AND THE SLOPE OF THE LINE SEGMENT. THE PROGRAM SHOULD USE THE COORDINATES DESIGNATIONS SHOWN.

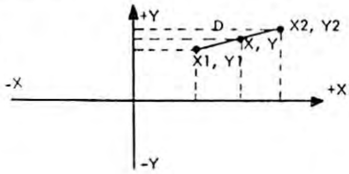
DEFINITIONS FOR LINE OF COORDINATES X1, Y1, X2, Y2 ARE:

DISTANCE $= \sqrt{(X1-X2) + (Y1-Y2)}$

MIDPOINT $= X = \frac{X1 + X2}{2}$, $Y = \frac{Y1 + Y2}{2}$

SLOPE $M = \frac{Y1-Y2}{X1-X2}$ WHERE X1 = X2

(WHEN X1 = X2, SLOPE IS UNDEFINED)



1. Use a READ X1, Y1, X2, Y2 to start. Leave lines 500 and 600 for referee data.
2. The Program should print out:

   DISTANCE =
   MIDPOINT =
   SLOPE = (0, ± M, Undefined)

3. Use Data Lines for at least 3 coordinate sets containing an Undefined Slope, a Zero Slope and a Negative Slope.

A POSSIBLE SOLUTION

```
10 READX1,Y1,X2,Y2
20 LET D=(((X1-X2)^2+(Y1-Y2)^2))^.5
30 PRINT"DISTANCE =";D
40 LET X=(X1+X2)/2
50 LET Y=(Y1+Y2)/2
60 PRINT"MIDPOINT:X=";X;"Y=";Y
70 IFX1=X2 THEN 110
80 LET M=(Y1-Y2)/(X1-X2)
90 PRINT"SLOPE=";M
100 GOTO10
110 PRINT"X1=X2,SLOPE IS NOT DEFINED"
120 GOTO10
500 DATA3,5,7,8,-3,5,4,5,-3,6,-3,-8
600 DATA1,6,9,1
```

SOLUTION REQUIREMENTS:

1. Lines 30, 60 and 90 or their equivalents are required
2. Referee must insert Data Lines 500 and 600 to get answers

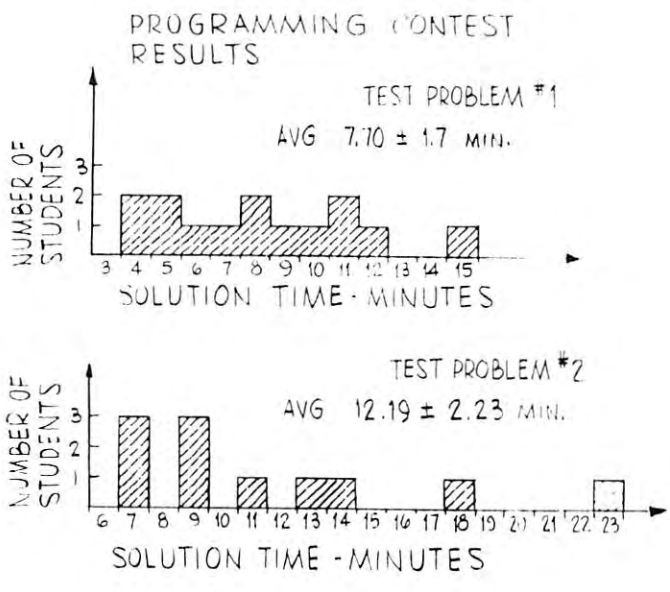


TABLE II

THE STAR MODEM

From Livermore Data Systems

PET IEEE 488 MODEM

SALE $

RS232 MODEM

SALE $139

The STAR modem from Livermore represents a significant breakthrough in the development of acoustic modems. The small, lightweight case houses a high-performance modem that competes with the highest quality standard-sized couplers available. Yet, because of its costs effective design, the STAR has become the price/performance leader in the industry.

CIRCUITRY

The switchable, four-section bandpass filter provides the user with excellent out-of-band rejection to assure accurate processing of the received carrier, even at signal levels of less than –47 dBm. Further, the proven soft limiter and phase lock loop discriminator yields data that is essentially jitter free.

The oscillator is built using highly stable, state-variable circuitry that delivers a nearly harmonic free, phase coherent sine wave to the telephone network, assuring compatibility with all other 103 type modems. Because of the pureness of the sine wave, the STAR modem exceeds even the stringent harmonic requirements of all CCITT countries.

CARRIER DETECT

To assure accurate teleprocessing connections, the carrier detect circuitry prevents the modem from attempting to operate when excessive noise would produce errors or cause marginal operation. The circuitry also has a special amplitude sensor that prevents chatter when the received signal fades.

EXCLUSIVE ACOUSTIC CHAMBERS

The exclusive triple seal of Livermore's new flat mounted cups locks the handset into the acoustic chamber yielding superior acoustic isolation and mechanical cushioning. Designed to adapt to most common handsets used throughout the world (also fits GTE handsets), the STAR offers the utmost in flexibility and transmission reliability.

SELF TEST

The self test feature on the STAR allows the user to verify total operation of the acoustic modem by using the terminal in the full duplex mode. No need for remote assistance in diagnosing terminal or modem products.

Utilizing the experience gained from building high quality couplers for over twelve years, Livermore has designed a coupler superior to any it its class for cost efficiency in industrial, commercial, business or home situations. You can see why we call it the STAR!

SPECIFICATIONS

Data Rate. 0 to 300 baud.
Compatibility. Bell 103 and 113; CCITT.
Transmit Frequencies.* Originate – 1070 Hz/Space, 1270 Hz/Mark; Answer – 2025 Hz/Space, 2225 Hz/Mark.
Receive Frequencies.* Originate – 2025 Hz/Space, 2225 Hz/Mark; Answer – 1070 Hz/Space, 1270 Hz/Mark.
Frequency Stability. ±0.3 percent.
Receiver Sensitivity. –50 dBm ON, –53 dBm OFF.
Transmit Level. –15 dBm.
Modulation. Frequency shift keyed (FSK).
Carrier Detect Delay. 1.2 seconds ON; 120 msec OFF.
EIA Terminal Interface. Compatible with RS 232 specifications.
Interface. IEEE 488.
Optional Interfaces. 20 ma;
***International (CCITT) frequencies available.**
Switches. Originate/Off/Answer; Full Duplex/Test/Half Duplex.
Indicators. Transmit Data, Receive Data, Carrier Ready, Test.
Environmental. Ambient operating temperature 5°C. to 50°C. Relative humidity 10 to 90 percent (non-condensing).
Power. Supplied by 24 VAC/150 MA UL/CSA listed wall-mount transformer. Input 115 VAC, 2.5 watts (220 VAC, 50 Hz adaptor available on request).
Dimensions. 10" x 4" x 2"
Weight. 1.7 lbs. (2.2 lbs. shipping weight including AC adaptor.)
Warranty. Two years on parts and labor.

6502	7.45	10 @	6.95	50 @	6.55	100 @	6.15
6502A	8.40	10 @	7.95	50 @	7.35	100 @	6.90
6520 PIA	5.15	10 @	4.90	50 @	4.45	100 @	4.15
6522VIA	6.90	10 @	6.50	50 @	6.10	100 @	5.70
6532	7.90	10 @	7.40	50 @	7.00	100 @	6.60
2114-L450			4.65	20 @	4.35	100 @	4.15
2114-L300			5.95	20 @	5.45	100 @	5.10
2716 EPROM			21.00	5 @	19.00	10 @	17.00
4116-200 ns RAM			7.00			8 @	6.25
6550 RAM (PET 8K)							12.70
S-100 Wire Wrap			2.65		Solder Tail		2.15

CASSETTES—AGFA PE-611 PREMIUM

High output, low noise, 5 screw housing, labels

C-10 10/5.65 50/25.00 100/48.00
C-20 10/6.45 50/29.50 100/57.00
C-30 10/7.30 50/34.00 100/66.00

All other lengths available. Write for price list

ATARI **ATARI 800 SPECIAL $799**
All Atari Modules 20% OFF

DISKS

(write for quantity prices)

SCOTCH (3M) 8"	10/3.10	50/2.85	100/2.75
SCOTCH (3M) 5"	10/3.15	50/2.95	100/2.85
Maxell 5"	10/3.65	50/3.40	100/3.15
Maxell 8" Double Dens	10/4.10	50/3.95	100/3.80
Verbatim 5"	10/2.40	50/2.35	100/2.30
BASF 5"	10/2.45	20/2.35	100/2.30
BASF 8"	10/2.50	20/2.45	100/2.35
Diskette Storage Pages			10 for 3.95
Disk Library Cases		8" - 2.85	5" - 2.15

Commodore CBM-PET SPECIALS

FREE –Up to $235 free merchandise with purchase of one of following CBM-PET items.

		FREE
8032 32K-80 column CRT	$1795	235
8016 16K-80 column CRT	1495	205
8050 Dual Disk Drive-950,000 bytes	1695	220
CBM Modem-IEEE Interface	395	50
CBM Voice Synthesizer	395	50
8N full size graphics keyboard	795	100
16K Full Size Graphics or Business Keyboard	995	145
32K Full Size Graphics or Business Keyboard	1295	205
2040 Dual Disk Drive-343,000 bytes	1295	205
2022 Tractor Feed Printer	795	100
2023 Pressure Feed Printer	695	90
C2N External Cassette Deck	95	12
Used PETs (limited quantities)	Call	

****** EDUCATIONAL DISCOUNTS ******
Buy 2 computers, get 1 FREE

VISICALC for PET (CBM/Personal Software)	$128
CBM Assembler/Editor (disk)	89
CBM General Ledger, A/P, A/R NEW!	270
CBM Full Size Graphics Keyboard	$ 74
WordPro I-for 8K PET	25
WordPro II-16 or 32K, 2040, Printer	72
WordPro III-32K, 2040, Printer	178
WordPro IV for 8032	268
PASCAL Compiler for PET	75
Programmers Toolkit-PET ROM Utilities	$ 44.90
PET Spacemaker Switch	22.90
Dust Cover for PET	7.90
IEEE-Parallel Printer Interface for PET	79.00
IEEE-RS232 Printer Interface for PET	149.00

Centronics 737 Proportional Spacing Printer	**$800**
NEC Spinwriter-parallel	2450
SYM-1 $209 with 4K RAM	$ 238
SYM BAS-1 BASIC in ROM	85
SYM RAE-1/2 Assembler in ROM	85
MDT 1000 Synertek Development System	1345
KTM-2/80 Synertek Video Board	349
KIM-1 (add $34 for power supply)	159
Seawell Motherboard-4K RAM	195
Seawell 16K Static RAM-KIM, SYM, AIM	320
S-100 Static RAM kit SALE	198
Leedex Video 100 12" Monitor	129
Zenith Z19 Terminal (factory asm.)	770
KL-4M Four Voice Music Board for PET	$34.90
Visible Music Monitor (4 Voice) for PET	29.90
SPECIAL—KL-4M with Visible Music Monitor	59.90
MICROTHELLO for PET by Michael Riley	$9.95
Machine language version—you can't win at Level 5	
PAPERMATE 60 Command PET Word Processor	$29.95
Full-featured version by Michael Riley	

A P Products 15% OFF

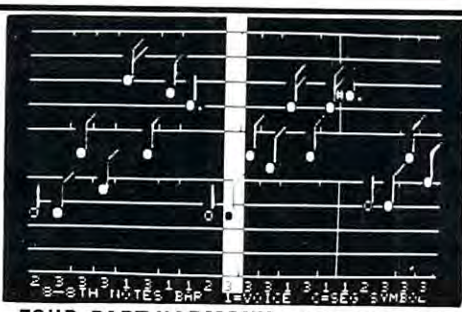
All Book and Software Prices are Discounted

PET Personal Computer Guide (Osborne)	$12.75
PET and the IEEE-488 Bus (Osborne)	12.75
6502 Assembly Language (Osborne)	9.45
Programming the 6502 (Zaks)	10.45
6502 Applications Book (Zaks)	10.45
Programming a Microcomputer 6502	7.75
6502 Software Bookbook (Scelbi)	9.45

115 E. Stump Road
Montgomeryville, PA 18936 **215-699-5826**

A B Computers

WRITE FOR CATALOG
Add $1.25 per order for shipping. We pay balance of UPS surface charges on all prepaid orders.



FOUR PART HARMONY FOR THE PET

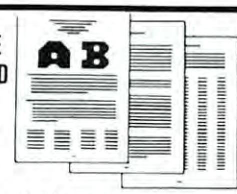
A-B Computers announces a combination system consisting of the KL-4M DAC Board and the Visible Music Monitor for Commodore PET-CBM computers. The package enables PET users to easily create and play musical compositions of up to 4 parts.

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



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**

KMMM PASCAL for PET

A subset of standard Pascal (Jensen, Wirth) for PET

Includes following standard identifiers:

Types: INTEGER, BOOLEAN, CHAR, TEXT

Procedures: RESET, READ, READLN, REWRITE, WRITE, WRITELN

+ - * = <> < > <= >=

() [] (: or (* *) := , ; '

div, mod, or, and, not , if , then, else, case, of, repeat, until, while, do, for, to, downto, begin, end, const, var, array, function, procedure.

Sequential I/O supported (both disk and tape)

Extensions:

Memory can be manipulated as an array (MEM)

% sign allows hex values in literals and I/O

Floating Point available in 4th quarter 1980 (nominal upgrade charge)

Included in package:

- Machine Language Pascal Source Editor
- Machine Language Compiler (generates P-Code)
- P-Code to machine language translator (produces optimized machine language object code —not just a P-Code interpreter).
- P-Code Interpreter (for debugging and learning)
- Run-time package
- User Manual
- Sample programs

Versions available for ROMs 2.0 (cassette only), 3.0, 4.0. Requires 16K minimum.

KMM PASCAL on tape or disk

(with User Manual) **$75**

User Manual Only **$15**

EARL FOR PET

Editor, Assembler, and Relocator/Linker Disk File Based for PET

Editor can edit files larger than memory.

Assembler features:

- 2 Pass Assembler
- Generates Relocatable Object Code
- Uses standard MOS Technology mnemonics
- Disk file input
- Listing output to screen or printer

Relocator/Linker capabilities:

- Relocates assembler output to desired memory location
- Resolves symbols declared external in assembly
- Links multiple object programs as one memory load

EARL for PET (with User Manual) .. **$65**

User Manual Only **$15**

ATARI Computer Systems

BASE UNITS	SALE
400 Computer System	$ 419
800 Computer System (with 16K RAM)	800
ACCESSORIES	
8K RAM Memory Module	95
16K RAM Memory Module	155
CX70 Light Pen	59
CX30-04 Paddle Controller Pair	16
CX40-04 Joystick Controller Pair	16
CX81 I/O Data Cord (5 feet)	12

PERIPHERALS	
410 Program Recorder	69
Disk Drive	539
815 Dual Disk Drive (Double Density)	1195
820 Printer (40 Column Impact)	399
822 Thermal Printer (40 Column)	358
825 Printer (80 Column Impact)	795
830 Acoustic Modem	159
850 Interface Module	175
PERSONAL INTEREST AND DEVELOPMENT	
CX4101 Invitation to Programming 1 (tape)	16
CXL4007 Music Composer (cartridge)	48
ENTERTAINMENT	
CXL4004 Basketball (cartridge) uses Joysticks	32
CXL4009 Computer Chess (cartridge) - Joystick	32
CX4111 Space Invaders (tape) uses Joysticks	16
CXL4011 Star Raiders (cartridge) - Joystick	48
CXL4006 Super Breakout™ uses Paddles	32
CXL4010 3-D Tic-Tac-Toe (cartridge) - Joysticks	32
CXL4005 Video Easel (cartridge) uses Joysticks	32
PROGRAMMING LANGUAGES	
CXL4003 Assembler Editor (cartridge)	48
EDUCATION-Talk & Teach™ Courseware (4 tapes per course, use with CXL4001)	
CXL4001 Education System Master (cartridge)	20
CX6001 U.S. History	24
CX6002 U.S. Government	24
CX6003 Supervisory Skills	24
CX6004 World History (Western)	24
CX6005 Basic Sociology	24
CX6006 Counseling Procedures	24
CX6007 Principles of Accounting	24
CX6008 Physics	24
CX6009 Great Classics	24
CX6010 Business Communications	24
CX6011 Basic Psychology	24
CX6012 Effective Writing	24
CX6014 Principles of Economics	24
CX6015 Spelling	24

115 E. Stump Road
Montgomeryville, PA 18936 **215-699-5826** **A B Computers** WRITE FOR COMPLETE CATALOG
Add $1.25 per order for shipping. We pay balance of UPS surface charges on all prepaid orders.

Al Baker's Programming Hints: APPLE and ATARI

More on Menu Selection

The Apple is a programmer's computer. It has many strengths and few weaknesses. However, using its strengths often requires the very best from a programmer. This issue we are going to explore effective use of the Apple's paddle controllers in menu selection. Next issue we'll continue this exploration with an even more powerful application of the paddles. But we're getting ahead of ourselves.

Last Issue: Atari

I left the Atari readers with a problem last time: condense the number selection routine as much as possible and use it in a program. If you'd like to share your solution with the rest of us, send me a listing. My solution is in Listing 1. The program is the old favorite "Guessing Game".

The routine is condensed into lines 1000 to 1050. I made a few changes in it to accomodate the game. The main change was to remove the setup of the variable "A". The rest of the program is the standard number guessing program. Lines 7 through 23 initialize the variables, including "A", and lines 30 through 80 pick out a random number and ask the player to guess it.

Line 90 calls the joystick number selection routine. If the player makes a correct guess, then lines 200 to 220 tell him so and loop back for another game. Otherwise lines 117 to 140 give him a Bronx cheer, tell him how he was wrong, and loop back for another guess.

```
1 REM          GUESS A NUMBER
2 REM
3 REM
5 REM   SET UP THE JOYSTICK DATA
6 REM
7 A=10
10 LOW=1
20 HIGH=20
21 X=17
22 Y=12
23 PLAYER=1
27 REM
28 REM       PLAY THE GAME
29 REM
30 GRAPHICS 0
40 POSITION 2,5
50 ? "I AM THINKING OF A NUMBER BETWEEN"
60 ? LOW;" AND ";HIGH;".  ";
70 ? "WHAT IS YOUR GUESS:"
80 GUESS=INT(RND(0)*20)+1
82 REM
84 REM      GET THE PLAYER'S ANSWER
86 REM
90 GOSUB 1000
100 POSITION 14,20
110 IF A=GUESS THEN 200
112 REM
114 REM       WRONG GUESS
116 REM
117 SOUND 0,200,10,15
118 FOR I=1 TO 50:NEXT I
119 SOUND 0,0,0,0
120 IF A<GUESS THEN ? "TRY HIGHER"
130 IF A>GUESS THEN ? "TRY LOWER "
140 GOTO 90
170 REM
180 REM       CORRECT GUESS
190 REM
200 ? "YOU GOT IT"
210 FOR I=1 TO 500:NEXT I
220 GOTO 30
960 REM
970 REM    JOYSTICK NUMBER SELECT
980 REM    (DISCUSSED LAST ISSUE)
990 REM
1000 POKE 752,1
1010 POSITION X,Y:? A;" ";:FOR SND=0 TO 15:SOUND 0,100-A,10,15-SND:NEXT SND
1020 IF (STICK(PLAYER-1)=11)*(A>LOW) THEN A=A-1:GOTO 1010
1030 IF (STICK(PLAYER-1)=7)*(A<HIGH) THEN A=A+1:GOTO 1010
1040 IF STRIG(PLAYER-1) THEN 1020
1050 RETURN
```

The Apple Paddle

The Apple hand controller has two inputs: a paddle and a button. The combination is called "the paddle". We are going to use the paddle to make menu selections in programs which do not otherwise use the Apple keyboard. If one or more players are playing a game which exclusively uses the paddles for game inputs, it is poor design to force them to use

$819

16K $895
32K $1195

$499

APPLE*

MICROSOFT ADVENTURE
The original of ADVENTURE written for the DEC-10 systems is now available for the APPLE. Explore Colossal Cave for treasures while avoiding the dangers hidden within its many passages. 130 different rooms. 15 treasures, and characters ranging from helpful to deadly await you within the cave. Be careful where you step, and also who's behind you!
32K disk machine language $29.95

ANDROID NIM
by Leo Christopherson
The game that made Leo Christopherson famous is now available for the APPLE! The improved graphics and color of the APPLE make the game even better. Try to be the last one to shoot the androids on the screen. If you do, you win! Also includes realistic sound effects.
24K cassette machine language . . $14.95

MAGIC PAINT BRUSH
Hi-Res graphics package plus! Draw Hi-Res pictures using all APPLE's colors. Connect any points on screen, fill areas, plot, rotate, and scale shape, or 'paint' with a set of 9 brushes. Also comes with Shape Table Designer and 2 demo programs. Slot Machine and Applesoft Invaders.
32K disk Applesoft-ROM $29.95

THREE-D
You don't have to be an engineer or scientist to have high resolution graphics for your computer! This program permits rotation, scaling, shift, distortion, and combination of three dimensional graphics on the screen. MP Software.
48K disk Applesoft-ROM $29.95

ATARI*

STAR RAIDERS
The best! A ROM cartridge holds the game. A fast-paced full-color, space battle in which you must defeat the enemy Zylon ships while protecting your home bases. Real-time action and effects make this game the best space game available. Sixty levels of rating from Garbage Scow 4th Class to Commander make for continuously exciting play.
ROM cartridge $59.95

3D GRAPHICS
by Tim Hayes
High quality graphics program for the ATARI computer allows you to rotate, distort, shrink, and combine three dimension graphic projections on the screen. With the high resolution abilities of the ATARI, one of the finest graphics packages available anywhere!
16K cassette $29.95

WALL STREET CHALLENGE 6402
A computer simulation of the Stock Exchange is easy to play and always challenging. Invest in stocks, and try to make it big!
8K and 16K version on one cassette
.... $19.95

ALL STAR BASEBALL 6401
Two players face each other, one at batand the other pitcher and outfield. Innings, balls, strikes, and a variety of plays make for an exciting game. Joysticks are optional.
8K and 16K versions on one cassette.
.... $19.95

PET*

STARFLEET ORION
Command a starfleet! 2 player game system includes rule book, battle manual, control sheets. Two programs. 22 space ship types and 12 play tested scenarios.
8K cassette $19.95

RESCUE AT RIGEL
Search the moon base and rescue Delilah Rookh from the High Tollah. Automatic Simulations.
24K cassette $19.95

MORLOC'S TOWER
Match wits with the evil wizard and try to defeat him! Automated Simulations.
32K cassette $14.95

TIME TREK
by Brad Templeton from Personal Software. Real time action. Star Trek type game with sound effects. There are no 'turns'. The action continues whether you move or not. You and the Klingons can move, steer, and fire at the same time.
8K cassette $14.95

This is only a very small sample of our product line. For a complete selection, send $1 for our catalog of hardware, software, and publications and receive a $2 credit toward your first order.

APPLE, ATARI, and PET are trademarks of Apple Computer Co., Warner Communications, and Commodore, respectively

The Software Exchange 6 SOUTH ST., MILFORD, NH 03055

To order: Call Toll-Free 1-800-258-1790 (In NH call 673-5144)

master charge VISA

The Software Exchange & Hardside (Div. of Robitaille & Sons, Enterprises, Inc.), SoftSide Publications the keyboard when making menu or other choices between or during games. Let's avoid the keyboard.

Listing 2 is a sample program that uses paddle 0 to make a menu selection. It puts up a list of five selections. Turn the paddle until the selection you want is highlighted and push the button. That selection is yours. Push the button again to make another selection. This sounds simple enough and, for the user, it is.

For the programmer the problem is anything but simple. There are several problems that must be solved. First, the paddle must "feel" right. This means that it must have fluidly -- no jerkiness. Also, pushing the button must feel like an "event" similar to pressing a keyboard key. The button was designed to feel like a continuous "state" where holding it down creates a continuous input until released.

The Program

Lines 60 to 90 initialize the sample program and define the five selections on the menu. Lines 100 to 106 clear the screen and bypass the user's input. This is done to get the menu on the screen to start with. PR contains the menu item pointed to by the paddle. The menu display routine on lines 220 through 280 print the five selections. The menu entry that matches PR is shown in inverse video.

Now let's sturdy closely the input routine on lines 140 to 185. Line 145 picks up the value from paddle 0 and converts it into a number between 0 and 5*50-1. This is done so that line 150 will always assign a value to the variable PA between 1 and 5. If line 145 allowed values of PA greater than 249 then line 150 would let PA equal as much as 6, a menu item which doesn't exist.

Now the program checks the status of the button in line 160. If PEEK(-16287) is greater than 127 then the button is pressed and the user has made his choice. Otherwise the program checks to see if the new value from the paddle is unchanged. As long as PA equals PR the program will continue to monitor the paddle and button by looping back to line 145. If they are not the same, the user has moved the paddle. The program sounds the bell in line 184 and BEGINS to change the menu display to match the new paddle position.

If line 185 was PR = PA then rapid changes of the paddle would create jerky changes in the display. Spinning the paddle from left to right might cause the menu to change from a highlighted first selection to a highlighted fifth selection, for example. This doesn't feel right to the user, especially the non-programmer. He usually thinks there is almost a mechanical linkage between what he does and what happens on the screen. Having a smooth paddle motion create jerky screen changes violates this sense and feels wrong.

Instead of setting PR equal to PA, line 185 moves PR closer to the value of PA. If PA is bigger than PR, then it adds 1 to PR. If PA is less than PR than it subtracts one from PR. Finally, lines 220 to 280 reprint the menu and go back for more user input.

```
10  REM       PADDLE MENU SELECT
20  REM
30  REM
40  REM
50  REM  DEFINE THE OPTIONS
60  DIM OP$( 5 )
70  FOR I = 1 TO 5: READ OP$( I ): NEXT
     I
80  DATA  THE MAGICIAN,THE DETECT
     IVE,THE SOLDIER
90  DATA  THE COWBOY,THE POLITICI
     AN
100 PR = 0
105  CALL  - 936
106  GOTO 220
110  REM
120  REM  WAIT FOR PADDLE MOVE
130  REM
140  CALL  - 936
145 PA =  PDL (0): IF PA > 249 THEN
     PA = 249
150 PA =  INT (PA / 50) + 1
160 BU =  PEEK ( - 16287)
170  IF BU > 127 THEN 320
180  IF PA = PR THEN 145
184  PRINT  CHR$ (7);
185 PR = PR +  SGN (PA - PR)
190  REM
200  REM  DISPLAY MENU
210  REM
220  FOR I = 1 TO 5
230  IF PR = I THEN  INVERSE
240  HTAB 10: VTAB 5 + 2 * I
250  PRINT OP$( I )
260  NORMAL
270  NEXT I
280  GOTO 145
290  REM
300  REM  SELECTION HAS BEEN MADE

310  REM
320  CALL  - 936
330  VTAB 5
340  PRINT  CHR$ (7);"YOU SELECTE
     D ";OP$( PA )
341  REM
342  REM  WAIT FOR BUTTON TO BE R
     ELEASED
344 BU =  PEEK ( - 16287)
345  IF BU > 127 THEN 344
346  REM
347  REM  WAIT FOR BUTTON TO CONT
     INUE
```

```
348  REM
350  VTAB 15
360  PRINT "PRESS BUTTON TO CONTI
     NUE";
370 BU =  PEEK ( - 16287)
380  IF BU < 128 THEN 370
385  CALL  - 936
386  PRINT  CHR$ (7);
390  REM
400  REM  WAIT FOR BUTTON TO BE R
     ELEASED
410  REM
420 BU =  PEEK ( - 16287)
430  IF BU > 127 THEN 420
440  GOTO 100
```

The remainder of the program handles the user's menu choice. Lines 320 to 340 display the choice and lines 344 and 345 wait for him to release the button. Remember that he pressed the button to make the selection. The program shouldn't reread the button until the user has let it go. Once the button is released, lines 350 and 386 request that the user press the button, wait for the button press, and sound the bell. Then lines 420 to 440 wait for the button to be released before going back to the menu.

Conclusion

We've explored one use of the Apple paddle. Next time we are going to simulate a joystick for such games as Space Invaders. The actual arcade game uses a joystick to control sun motion. Most Apple versions treat the paddle as direct input to position the sun. This tends to frustrate Space Invader fans who are used to the real thing.

Here is your problem for next time. How can you simulate a two way joystick (left<-center-> right) with the Apple paddle without needing to know at any time how far the paddle is from its center position?

©

Fun With the 6502 Atari Software Reviews

Atari 3 Dimensional Graphics

Len Lindsay

Sebrees Computing (456 Granite Ave., Monrovia, CA) is marketing a sophisticated software package that will help you create your own three dimensional pictures on your Atari screen, and even have them moving in any direction you would like. The software package costs $29.95 plus $1.50 postage and handling.

I have referred to this software as a package, rather than as a program, because you receive four separate programs on the tape. Along with the tape comes a manual explaining how to use the programs and how they work. The last pages of the manual contain complete listings of each of the programs.

I received a preliminary version of **3D Computer Graphics** for the ATARI, and was impressed. For example, the first program mixes all three types of TEXT MODES on the screen at one time which is very interesting. The final version will use more colors in Graphics mode 8 than the default maximum set by BASIC.

The programs allow you to set up the three dimensional coordinates of any object you wish drawn on the screen (using X, Y, Z axes). It takes quite some time to figure out all the coordinates and enter them into the program. (This is a serious program; plan on spending some time with it.) Once you have all the coordinates entered, you can have the object drawn on your screen. This sounds simplistic, but you can vary the place that you are viewing the object from; vary the field of view; and "vary" the viewing position you are looking at the object from. *And* you can draw the object, or erase it, all under program control. Thus you can actually create a three dimensional animated scene.

The final manual will have a complete chapter on examples. The last program on the tape is an example of animation all ready to RUN. You can watch an animated SPACE SHUTTLE. A plastic model of the Space Shuttle was used along with graph paper to identify its outline coordinates.

If you are set for some serious fun with your ATARI, and have the time to enter coordinates of your object into the program's data base, then you should enjoy this package.

The Video Easel Cartridge from Atari

This is one of ATARI's plug-in cartridges. It is an amazing cartridge, showing off some of the ATARI's amazing capabilities. It uses joysticks to control many of its functions. The more you use it the more things you find you can do.

Painting (demo mode)

There are 6 different dynamic paintings preset for your instant use. Simply hit **P** (for Painting), then hit a number 1 - 6 for the painting number, then hit RETURN and the painting begins. My favorite paintings are numbers 1, 3, and 6. You can switch from one painting to another at any time. The screen is not cleared when you transfer unless you want it to clear. To clear the screen, hit **C** (for Clear) and RETURN. The paintings are constantly changing, creating beautiful displays. And it is *FAST*. You can even control the speed if you wish. Simply use joystick #1. Push it forward and the painting speeds up. Pull it back and it slows down. Slow it down to a snails pace and watch how the video magic is performed (it is extremely interesting). You can control colors used with joystick #2. Push it forward or right or left to change the color registers. Hold the RED button down at the same time and you change the luminance levels.

Drawing

Now for the exciting news. You can create your own custom dynamic painting sets. It is very simple and provides a great sense of accomplishment and satisfaction.

Hit **D** (for Draw) and RETURN and you are in the DRAW "set-up" mode. Use joystick #1 to set up the master pattern. Your master pattern can both DRAW and ERASE lines. To set up a DRAW hold the RED button down as you move a small dot around the screen with your joystick (leaving a trail). To erase, don't hold the RED button down. When you are ready simply hit **S** (for Start) and RETURN and ZAP, the computer starts drawing your pattern over and over rapidly filling the screen with your instant masterpiece. You can control the speed and colors as for PAINTING mentioned above.

This DRAW mode can easily be used for trivial fun. But, it can just as easily be used for some serious fun. However, to create some well-designed, thoughtful, dynamic paintings, you will have to spend some time experimenting. See if you can

figure out just what the computer does when it duplicates your pattern. You may find that the screen fills up too fast, and then looks like garbage. AHA. Your master pattern should include some ERASE lines, to help keep the screen from filling up. There are other tricks to creating long-lasting dynamic art, but I won't spoil your fun by telling you what they are.

Quad Drawing

If you like DRAWing, you will have four times the fun with QUAD DRAWing. This divides the screen into four quadrants, focusing on the center. Now you draw in all four quadrants at once (symmetrically of course). This is a fast way to create interesting designs.

Who and Why

Children as young as 6 years old can control the VIDEO EASEL. It will encourage experimenting and exploration, as well as allow creative play and aid visual thinking. It can be used at very simple levels, but can be much more sophisticated for use by high school students. It was with more sophisticated use in mind that LIFE was included.

Life

This is a population simulation. Although it is referred to as the game of LIFE, there really are no opponents or winning strategies. A whole article could be devoted just to explaining the significance of this famous "game", in fact many such articles already have been published. A partial bibliography is included with the VIDEO EASEL manual. There is even a newsletter dealing with this computer simulation. It basically deals with a population of "cells" that you set up. Once you START it, it follows set rules:

1) **Law of Survival** - each cell with 2 or 3 neighbor cells survives to the next generation. Each cell with 4 or more neighbors will die from "overcrowding". Each cell with one or less neighbors will doe from "isolation".

2) **Law of Birth** - each empty space with exactly 3 neighbors will create a new cell for the next generation.

ATARI's LIFE is very fast, and has many fancy "extras"built in to make it easier to use for fun and recreation. You can put the cells on the screen one at a time if you wish, but that takes time. So ATARI gives you several options to put many cells up at one time.

a) **BIG X** - puts an X on the screen
b) **LINE** - puts lines (horizontal or vertical) on the screen of any length
c) **DIAGONAL** - puts diagonal lines on the screen
d) **IBEAM** - puts a large I on the screen

Each of the above can be used with the others. Once you have the screen set up with all your "cells" you simply hit S (for Start) and watch the generations go by. Some very interesting patterns are created. For more advanced use, you can automatically put GLIDERS and FACTORIES on the screen. Factories create Gliders. They both are fully explained in the manual.

Final Remarks

Video Easel may appear to be simply a glorified drawing program, but it is much more. I am very satisfied with it, and my daughter enjoys it as well, even though she is only three years old. In several years it should prove thought provoking for her. It can be used successfully in grade schools (for the patterns and drawing) and in high schools (for the game of LIFE simulation). ©

THE apple® GAZETTE

Randomize For The Apple II

Sherm Ostrowsky
291 Salisbury Avenue
Goleta, CA 93017

When you play games on your Apple II, do things start looking familiar after a while? Does the game get boring because you know just what is going to happen next, even though the random number generator in the program is supposed to make each event unexpected? There is a simple, one-line statement in BASIC which can remedy this problem; you can use it in your own programs, or even insert it into commercial programs after loading them.

The problem arises because the random numbers generated by the RND(1) function are part of a pseudo-random sequence which is always the same whenever you turn on the computer. You can select a different pseudo-random sequence by first entering a seed, S, and using a statement like:

```
X = RND(-S)
```

before any of the calls to RND(1). But this sequence, too, will always be the same very time you run your program with the same seed. What is really needed is a way to generate a starting seed which is different every time you run the program, and which is unknown to you.

Some versions of BASIC have a command ("RANDOMIZE") which does just this. Apple BASIC and APPLESOFT, unfortunately, do not have this command. A method which has been used by many Apple programmers to get around this difficulty is to ask for a starting seed from the user at the beginning of the program run, e.g.,

```
10 INPUT "SEED: ";S : X = RND(-S)
```

This will indeed start a different sequence of random numbers for that run, but it has some undesirable features. It may not be compatible with the ambience of the game ("Welcome to the space world of the twenty-third century' please enter a seed."). The user may not know enough about computers to understand what is wanted. And in any case, it is best if the seed is not known to the user, so each game can come as a complete surprise.

A somewhat more sophisticated approach, which I have seen used in at least one elegant program, uses a sequence such as

```
20 PRINT"HIT ANY KEY WHEN READY
 TO PROCEED";
30 X = RND(1): X = PEEK(-16384):
 IF X < 128 GOTO 30
```

This does the job: while waiting for the key to be pressed, the repeated calls to RND place the system at an unknown and unpredictable location in the random number sequence. It is, however, not necessary to program this so directly, because the Apple monitor has a built-in routine which does the same thing.

The Apple's pseudo-RANDOMIZE function works as follows: whenever the cursor is blinking at you while awaiting some kind of input, a little machine-language loop is rapidly and repeatedly incrementing a two-byte integer stored at decimal locations 78 (low byte) and 79 (high byte). No matter how fast you respond by typing some input, this loop will have gone around so many times that the number stored in those locations will be quite random and unknown. Now, in order to get your program into the computer, it must obviously have been necessary either to type it in or read it in from tape or disk; either way, the blinking cursor must have appeared for you, even if only to await the LOAD command. Therefore, there will *always* be a random number waiting there to be your seed.

A simple way to use this pseudo-RANDOMIZE function is just to put the following statement near the beginning of your program:

```
10 X = RND(-PEEK(78) - 256 * PEEK(79))
```

Thereafter, any uses of RND(1) will get numbers out of a completely unpredictable sequence of random numbers. ©

SCREENDUMP

Jeff Schmoyer

Screendump is a machine language utility program which prints the contents of an Apple II text screen to any printer. It is executed by pressing a control-Z on the keyboard in response to any input. Its uses include printing the catalog without having to specifically start the printer, getting a hardcopy printout of instructions from programs, and selectively preserving information on the screen. Screendump will run on any size Apple II computer with or without a disk drive. It will work from Applesoft, Integer BASIC, or the Monitor.

Throughout this article control characters are printed in the format control-Z. This means press the Control key, and while holding it down, type Z or whatever other character is requested. Control characters are not displayed on the Apple's screen. All the addresses shown with dollar signs ($) in front are hexadecimal addresses.

When activated, Screendump replaces the system's standard character input hooks with its own. Normally when the computer wants a character, it goes to the Monitor keyboard input routine which waits for one to be typed and then passes it on. When Screendump is on, the computer goes to it for a character. It then checks to see if the character typed was a control-Z. If not, it passes it on just like the normal routine. If the character was a control-Z, it prints out the screen.

To accomodate different types of printers and interfaces, Screendump has its own output hooks at $2FE and $2FF. These should be set to the address of the printer driver routine which prints one character. On each of Apples' and most other manufacturers printer cards resides a ROM containing a printer driver routine to make the card work. After a PR#1 (if the card is in slot 1) is executed, the computer jumps to that driver whenever it wants to print a character. It does this by setting the systems output hooks to the appropriate driver address. For an Apple parallel card in slot 1 this address is $C102. If a different card is to be used, this address may be discovered through the following procedure.

Go to the Monitor by pressing Reset on a standard Apple II or by CALL -151 on an Apple II Plus or an Apple II containing a Language System.

Type control-P control-K and Return to disconnect the DOS.

Type the slot number that the printer card is in followed by a control-P and Return. For slot 1 this would be 1 control-P Return.

Type 36.37 Return. This will display the printer address in reverse order. For $C102 it would show 36:02 C1.

These need to be placed in Screendump in reverse order also. For $C102, $2FE should get $02 and $2FF should get $C1.

One other change is required for systems without disk drives. The value at locations $2B7 needs to be changed from $4C to $60.

To *use* Screendump from the Monitor type 2AFG and Return or from either BASIC type CALL 687 and Return. Now any time control-Z is pressed, the current screen will be printed. Screendump may be used anytime up to the next Reset or IN# command.

To *save* Screendump to tape, from the Monitor type 2AF,2FFW. To save it to disk type BSAVE SCREENDUMP,A$2AF,L$51. To reload it from tape, go to the Monitor and type 2AF.2FFR. To reload it from disk enter BLOAD SCREENDUMP or alternatively BRUN SCREENDUMP. The latter would both load Screendump and activate it.

Some printer cards, such as Apples' parallel card, need to be initialized before they can be used. This initialization must be done each time the computer is turned on, or with some cards, each time Reset is pressed. For a parallel card in slot 1 the sequence would be

PR#1
control-I 40N control-I K
PR#0.

After Screendump is activated, through the execution of a CALL 687 or one of the other previously described turn-on procedures, it may be utilized by typing a control-Z in response to any input. For example if the catalog is being displayed and the computer is waiting for any key to be pressed before showing more, control-Z can be entered and what is on the screen will be printed. The catalog will not advance until some other key is pressed. As another example, assume you are playing your favorite adventure game and it is waiting for you to enter a command. Typing control-Z will print the current screenful of information describing your whereabouts for future reference.

In some cases the control-Z character may need to be used by other programs or devices such as the Micromodem II, for their own purposes. If Screendump is active, it will never let a control-Z go through he system. To make a different use of control-Z, Screendump may be deactivated through an IN#0 or Reset, or the Screendump execution character can be changed to something other than a control-Z. This is accomplished from Applesoft by typing POKE 702,CHR$("newchar"). The character in quotes, newchar, may be any character the system is not using for something else. For example, an A would not be a good character to use since everytime an A was typed the printer would start.

The operation of Screendump is as follows. SDINIT is the startup routine. It takes SCREENDUMPs address and puts it in the input hooks so that Screendump is called to get each character. If a disk is being used, the DOS is jumped to, passing the input address information along to it.

As previously outlined, when the system wants a

character it goes to SCREENDUMP which in turn looks to the Monitor routine KEYIN for a character. It then checks to see if the character entered was a control-Z. If not, it returns the character to the caller, your adventure program or whatever.

DUMPIT is the routine that actually prints the screen. First it saves the CPU registers and the current screen pointers. This is so that when it is finished, the cursor position and other information will still be intact.

Next it zeros the X register which will serve as the screen line counter. The Y register will contain the character position on that line. The X register (the line) is then transferred to the accumulator (the A register) and the Monitor routine GBASCALC is called. This routine translates the line number in the accumulator into the actual location of where that line starts in computer memory.

The forty characters for that line are now printed followed by a carriage return. The routine then goes to the next line and so on until it has done 24 lines.

After it finishes the screen, it restores the saved screen pointers and registers. Finally it goes to get a new character. It does this instead of passing the control-Z on to the system and causing a probable SYNTAX ERROR.

Screendump resides in memory at the end of the keyboard buffer area. These locations are generally not used by other programs but if a very long line is typed in, over 170 characters, Screendump will be destroyed. If it was active at the time of destruction, the computer will stop or do strange things. Hit Reset to recover.

```
:ASM
0000:                ; SCREENDUMP
0000:                ;
0000:                ;  DUMP SCREEN TO PRINTER WHENEVER
0000:                ;  CONTROL-Z IS PRESSED.
0000:                ;
0000:                ;     BY JEFF SCHMOYER  5/80
0000:                ;
0000:                ;
0000:                KSWL        EQU $38            CHAR IN HOOKS
0000:                KSWH        EQU KSWL+1
0000:                DOSSET      EQU $3EA           DOS SET HOOK ROUTINE
0000:                CH          EQU $24            CURSOR HORIZONTAL
0000:                GBASL       EQU $26            BASE LINE ADDRESS
0000:                GBASH       EQU GBASL+1
0000:                GBASCALC    EQU $F847          CALCULATE BASE ADDRESS ROUTINE
0000:                PRINT       EQU $C102          PRINTER CARD CHAR OUT ADDRESS
0000:                RDKEY       EQU $FD0C          MONITOR IN
0000:                KEYIN       EQU $FD1B          GET ONE PRESS
0000:                CR          EQU $8D            CARRIAGE RETURN
0000:                ;
02AF:                            ORG $2AF
02AF:                            OBJ $2AF
02AF:                ;
02AF:                ;  INITIALIZE THE INPUT HOOKS
02AF:                ;  TO POINT TO OUR ROUTINE.
02AF:                ;
02AF: A9 BA          SDINIT      LDA #SCREENDUMP
02B1: 85 38                      STA KSWL
02B3: A9 02                      LDA #SCREENDUMP/256
02B5: 85 39                      STA KSWH
02B7: 4C EA 03                   JMP DOSSET         MOVE SCREENDUMP INPUT HOOKS TO DOS
02BA:                ;
02BA:                ;  GET A CHAR FROM THE KEYBOARD
02BA:                ;  AND CHECK FOR CONTROL-Z.
02BA:                ;  IF NOT THEN RETURN CHAR TO CALLER.
02BA:                ;
02BA: 20 1B FD       SCREENDUMP  JSR KEYIN          GET A CHAR
02BD: C9 9A                      CMP #$9A           IS IT A CTRL-Z?
02BF: F0 01                      BEQ DUMPIT         YES DUMP SCREEN
02C1: 60                         RTS                NO, SEND BACK CHARACTER
```

```
02C2:               ;
02C2:               ;  SAVE CURRENT POINTERS AND
02C2:               ;  PRINT THE SCREEN.
02C2:               ;
02C2: 8A            DUMPIT     TXA                 SAVE REGS
02C3: 48                       PHA
02C4: 98                       TYA
02C5: 48                       PHA
02C6: A5 26                    LDA GBASL           SAVE CURRENT SCREEN POINTERS
02C8: 48                       PHA
02C9: A5 27                    LDA GBASH
02CB: 48                       PHA
02CC: A5 24                    LDA CH
02CE: 48                       PHA
02CF: A2 00                    LDX #0              LINE COUNTER
02D1: 86 24                    STX CH              ZERO CURSOR HORIZONTAL
02D3: A0 00         NEXTLINE   LDY #0              COLUMN COUNTER
02D5: 8A                       TXA                 A GETS LINE
02D6: 20 47 F8                 JSR GBASCALC        TRANSLATE IT
02D9: B1 26         NEXTCHAR   LDA (GBASL),Y       GET A CHAR
02DB: 20 FD 02                 JSR PRINTONE        OUT WITH IT
02DE: C8                       INY                 MOVE TO NEXT CHAR
02DF: C0 28                    CPY #40             LINE DONE?
02E1: D0 F6                    BNE NEXTCHAR        NO
02E3: A9 8D                    LDA #CR
02E5: 20 FD 02                 JSR PRINTONE
02E8: E8                       INX                 NEXT LINE
02E9: E0 18                    CPX #24             ALL DONE?
02EB: D0 E6                    BNE NEXTLINE        NO
02ED: 68                       PLA                 PUT OLD LINE STUFF BACK
02EE: 85 24                    STA CH
02F0: 68                       PLA
02F1: 85 27                    STA GBASH
02F3: 68                       PLA
02F4: 85 26                    STA GBASL
02F6: 68                       PLA
02F7: A8                       TAY                 RESTORE REGS
02F8: 68                       PLA
02F9: AA                       TAX
02FA: 4C 0C FD                 JMP RDKEY           GET NEW KEYPRESS
02FD:               ;
02FD:               ;  JUMP TO ACTUAL PRINTER DRIVER
02FD:               ;  CHARACTER OUTPUT ROUTINE.
02FD:               ;
02FD: 4C 02 C1      PRINTONE   JMP PRINT

    0   ERRORS IN THIS ASSEMBLY
```

©

Thesus Versus The Minotaur: PASCAL Visits Ancient Greece

Joseph H. Budge 2507 Elderwood Lane
Burlington, NC 27215

In ancient Crete there was a monster called the Minotaur who lived in an impossible maze, the Labyrinth. The Minotaur was a magical creature, half-man and half-bull. Once a year he demanded a human sacrifice. In return for the sacrifice he would protect the rest of the citizens from the evils of their enemies and nature. Appropriately enough, this was called the Minoan civilization. As time went on, the Minoans grew tired of the yearly sacrifices. After all, it was a drain on the population. Not only that, but people forgot how valuable their protection was. Eventually, the Minoans actually had to force people to sacrifice themselves. Imagine that! The victim would be thrown into the labyrinth, get lost, and eventually bump into Minotaur, with predictable consequences.

One day your basic Greek Hero type showed up, a dude named Theseus. Since it was sacrifice day, the Minoans grabbed Theseus and threw him in the Labyrinth. Being a Greek Hero and all, Theseus had his trusty battle-ax and a ball of string. He unwound the string until he found the Minotaur, slew the beast, and then followed his string back out. Through the marvels of modern science we are able to take you back in time to the very time and place of this epic event. Theseus is just stepping off his boat...

Suddenly a Minoan guard on the city wall challenges him: "Who goes there?"

"Tis I, Theseus, son of fair Hebride and mighty warrior, I come in peace."

Well, before he knows what's happening, a platoon of soldiers emerges from the city, grabs Theseus, and drags him off to the King's Palace. The King, of course, is delighted to see Theseus. So delighted, in fact, that he pulls out all the stops and orders a State Banquet be prepared in the field right out in front of the Labyrinth. But once dinner is ready, the platoon shows up again while the Great High Priest Mumbo-Jumbo explains the fate that has befallen Theseus: he's about to become bull fodder. Our Hero mutters the ancient Greek equivalent of "Sure, no sweat, baby!" before he grabs his pack and marches off through the great front doors of the Labyrinth. This is what he does:

Original maze:

```
* * * * * * * * * * * *
* . * . * * . . . . * *
* . . * M * . * * . * *
* . * * * * . * . * * *
* * * . . . * . * . * *
* * . * * * * . . * * *
. . * * * . . * * * * *
* * . * . * * * . . . *
* * * * * * . . * * * *
```

Figure 1: The Labyrinth

period = passageway
star = wall
M = Minotaur

Theseus is a jerk. The Minotaur lives.
Here is where he went:

```
* * * * * * * * * * * *
* . * . * * + + + + * *
* . . * M * + * * + * *
* . * * * * + * + * * *
* * * + + + * + * + * *
* * + * * * * + + * * *
+ + * * * + + * * * * *
* * + * + * * * + + + *
* * * * * * + + * * * *
```

Figure 2: Labyrinth after the first search plusses indicate where Theseus checked.

Theseus comes out of the maze, squinting in the sunlight. A roar from the angry crowd washes over him. The High Priest explains that the Minotaur must be either slain or fed. To emphasize his point, the Minotaur gives a big roar from inside the labyrinth! This sends the crowd scattering and leaves Our Hero quavering in his sandals. Mustering up his courage, once more he enters the labyrinth. While he wasn't looking, the Minotaur moved some of the walls around, so now he's just as lost as the first time he went in. Here's what happens this time:

Original maze:

```
* * * . * * * * * * * *
. . . . * * * * . * * .
* * * * * . * * * . . *
* . . * . * * . * . * *
. * . * * . . * * . . *
* . * * . * * * . * * *
* . . . * * . * * . * .
* * * . . M * * . . . *
* * * * * * * . * * * *
```

Figure 3: The re-arranged Labyrinth.

```
Theseus has slain the Minotaur
Here is where he went:

* * * + * * * * * * * *
+ + + + * * * * + * * +
* * * * * + * * * + + *
* T T * + * * + * + * *
T * T * * + + * * + + *
* T * * + * * * + * * *
* . T T * * . * * + * +
* * * . T W * * + + + *
* * * * * * * + * * * *

Figure 4: Theseus' trail out of the Labyrin
```

Well, the Minoans are just delighted! They make Theseus a Prince of the Realm, heap rewards on him, and throw a huge party in his honor. Many of the guests want to know how he did it, but Theseus keeps on saying "Aw, shucks, it was nothing..". Finally the King comes over; he's dying of curiosity too. So Theseus stoops over and draws the following program in the sand:

THEORY OF OPERATION

This is a maze search program written in Standard Pascal for the Apple II. It does not use any of the special Pascal functions unique to UCSD Pascal, therefore it should be portable to other Pascal machines.

The labyrinth is read into an array, the size of which is set by the constants M & N. In the array a '*' indicates wall, a '.' indicates passageway, a ' + ' means we've been on that passage before, and 'M' means Minotaur. The labyrinth is placed in an array surrounded by a circle of spaces and a circular wall. By placing Theseus inside this circle of sentinels, he can search out the entrance to the maze in the same way as he searches the maze. The search itself is straightforward. At any given square Theseus looks north, and then on around the compass. He takes the first available passageway that he hasn't taken before. If there's no passage, he back's up until he finds one or gets back to his starting point. Theseus himself is merely a stackpointer. He points to the end of his string, which is really the stored information on the status of each point he has traversed. To advance, a new node is pushed onto the stack, while retreat is performed by a pop. When the stack is empty, Theseus is back outside the maze. If Theseus finds the Minotaur he will slay him (M becomes W), and leave his string behind (a trail of 'T's).

```
program maze;

const    m = 9;                      (* rows in maze *)
         ml = 10;                    (* rows + 1 *)
         m2 = 11;                    (* rows + 2 *)
         n = 12;                     (* columns in maze *)
         nl = 13;                    (* columns + 1 *)
         n2 = 14;                    (* columns + 2 *)

type     stackptr = ^thissquare;                     (* pointer to stack nodes *)
         direction = (north, northeast, east,
                      southeast, south, southwest,
                      west, northwest);              (* legal directions *)
         thissquare = record                         (* one stack node *)
                         row : integer;              (* row in maze *)
                         column : integer;           (* column in maze *)
                         looking : direction;         (* direction looking here *)
                         string : stackptr           (* pointer to next node *)
                      end;
         map = array [-1..m2,-1..n2] of char;        (* Theseus's world *)
         vitality = (alive,dead);                    (* states of existence *)
         validmove = -1..1;                          (* a range for indexing *)
         index = array [north..northwest] of validmove;
         table = record                              (* array of indexes *)
                 hmove : index;
                 vmove : index
                 end;
         markers = set of char;                      (* used for input testing *)

var      maze : map;                                 (* the labyrinth *)
         Theseus : stackptr;                         (* top of stack = Our Hero *)
         Minotaur : vitality;                        (* how's the beast feeling? *)
         compass : table;                            (* look-up table of moves *)
         done    : boolean;                          (* flag for exit display *)
```

'*' is put in. The array is larger than the maze will be. The extra room, two squares on all sides, are the sentinels which are used to search for entrances. The compass is initialized as a look-up for moving in an indicated direction.

```
procedure ARRAYSTART (var maze:map; var compass:table);

var      i : integer;                                (* iteration variables *)
         j : integer;
         d : direction;

begin

   (* initialize the labyrinth *)
   for i := -1 to m2 do
     begin
       for j := -1 to n2 do
         begin
           maze[i,j] := '.';                         (* set to dots *)
           if (i=-1) or (j=-1) or (i=m2) or (j=n2) then maze[i,j] := '*'
                                                     (* sets walls to '*' *)
         end
     end;

   (* now set the compass *)
   with compass do
     begin
       for d := north to northwest do
         begin
           if (d=east) or (d=west) then vmove[d] := 0
           else if (d>east) and (d<west) then vmove[d] := 1
           else vmove[d] := -1;
           if (d=north) or (d=south) then hmove[d] := 0
           else if (d<south) then hmove[d] := 1
           else hmove[d] := -1
         end
     end
end;
```

```
procedure PRINTMAZE (maze:map);

var      i : integer;                    (* iteration variables *)
         j : integer;

begin
  for i := 1 to m do
    begin
      for j := 1 to n do write (maze[i,j], ' ');
      writeln
    end
end;
```

Printmaze steps through the two-dimensional array containing the maze and prints out its contents. In normal operation the sentinels are omitted, but that may be changed by placing the appropriate values into the for-loops.

```
procedure READMAZE (var maze:map);

var      i : integer;                    (* iteration variables *)
         j : integer;
         x : char;                       (* scratch for input *)
         legals : markers;               (* set of legal inputs *)

begin
  legals := ['*','.','M'];               (* what's allowed on input *)
  for i := 1 to m do
    begin
      for j := 1 to n do
        begin
          read (x);
          if x in legals then maze [i,j] := x
          else maze [i,j] := '.'
        end;
        readln
    end
end;
```

Readmaze is the general input routine which gets the maze from the keyboard. This particular version shows the deficiencies of Standard Pascal. No string hadling is allowed, therefore the data must be read in one character at a time. The lack of string handling also makes elegant user prompting difficult. If the maze is not found in the proper format (eg m lines of n chars) then the program will terminate with a run-time error. Erroneous characters in the data will be turned into passageways ('.').

This procedure Pop's a node off the stack. In effect this moves Theseus back one square. If underflow, then procedure returns with no change, it just prints an error (he's at end of the trail...

```
procedure POP (var Theseus:stackptr);

var      p : stackptr;                   (* scratch pointer *)

begin
  if Theseus = nil then writeln ('UNDERFLOW ON STACK')
  else
    begin
      p := Theseus;
      Theseus := Theseus^.string;
      dispose (p)
    end
end;
```

```
procedure PUSHON (var Theseus:stackptr; var maze:map; compass:table);

var      p : stackptr;                   (* scratch pointer *)


begin
  new (p);
  with Theseus^ do
    begin
      p^.row := row + compass.vmove [looking];
      p^.column := column + compass.hmove [looking];
    end;
  p^.looking := north;
  p^.string := Theseus;
  Theseus := p;
  maze [Theseus^.row,Theseus^.column] := '+';
end;
```

Procedure PUSHON pushes Theseus's current location onto the stack, saving his current coordinates and the direction he was looking. Then it moves Our Hero onto the next square in the direction he was looking and set's that to a ' + ' in the maze (drops a pebble) before returning to the calling routine.

```
procedure SEARCH (var Theseus:stackptr; var maze:map; var minotaur:vitality;
                  compass:table);

var     tr : integer;                                   (* temp row *)
        tc : integer;                                   (* temp column *)
        seewhat : char;                                 (* what he finds *)
        legals : markers;                               (* what he's allowed to see *)

begin
  legals := ['*','.','M','+'];                          (* what he may see *)
  with Theseus^ do
    begin
      tr := compass.vmove [looking] + row;              (* figure where he's *)
      tc := compass.hmove [looking] + column;           (* looking *)
    end;
  seewhat := maze [tr,tc];                              (* aha, he sees it! *)
  if seewhat in legals then
    begin
      if seewhat = '+' then seewhat := '*';             (* where he's been is same *)
                                                        (* as a wall: can't go there *)
      case seewhat of
        '*':    if Theseus^.looking = northwest then POP (Theseus)
                else Theseus^.looking := succ (Theseus^.looking);
                (* that was a wall or someplace he's been before *)

        '.':    PUSHON (Theseus,maze,compass);          (* a passage! *)

        'M':    begin                                   (* fight the Minotaur! *)
                  minotaur := dead;
                  maze [tr,tc] := 'W'                   (* it keels over... *)
                end
      end
    end
  else
    begin
      writeln ('What is that?');
      POP (Theseus)
    end
end;
```

Procedure Search has Theseus look around him to see what's there. Depending on what he finds, he then takes appropriate action. If he finds an illegal character he will get confused and go back out to the entrance for instructions. Otherwise he starts looking north and takes the first passageway he finds. If he finds the Minotaur he will, of course, fight. But if he finds no unexplored passage and no Minotaur, then he figure's he's at a dead-end and retreats. Once the search for this particular square is over, the procedure returns to its caller.

```
begin
  (* Initialize *)
  done := false;
  ARRAYSTART (maze,compass);
  new (Theseus);
  with Theseus^ do
    begin
      row := 0;
      column := -1;                     (* start out, 1st push moves to 0,0 *)
      looking := east;
      string := nil
    end;
  PUSHON (Theseus, maze, compass);
  READMAZE (maze);
  writeln;
  writeln ('Original maze: ');
  PRINTMAZE (maze);

  (* now go chase Minotaurs *)

  repeat
   begin
    if Minotaur = dead then with Theseus^ do
      begin
        if (row<1) or (row>m) or (column<1) or (column>n) then
          done := true;
        if not done then
          begin
            maze [row,column] := 'T'                    (* leave string *)
          end;
        POP (Theseus)
      end
    else SEARCH (Theseus, maze, Minotaur, compass)
   end
  until Theseus^.string = nil;
  writeln;
  if Minotaur = dead then writeln ('Theseus has slain the Minotaur')
  else writeln ('Theseus is a jerk. The Minotaur lives.');
  writeln ('Here is where he went:');
  writeln;
  PRINTMAZE (maze)
end.
%
```

Here's the main program...

©

Some routines from Applesoft Basic

Jim Butterfield, Toronto

Routines were identified by examining specific memory dumps. There may well be other versions of Basic; the user is urged to exercise caution.

The addresses given identify the start of the area in which the described routine lies. This may not be the proper program entry point or calling address.

DISK	ROM	Description
0800	D000	Action addresses for primary keywords
0880	D080	Action addresses for functions
08B2	D0B2	Hierarchy and action addresses for operators
08D0	D0D0	Table of Basic keywords
0A60	D260	Basic messages, mostly error messages
0B65	D365	Search the stack for FOR or GOSUB activity
0B93	D393	Open up space in memory
0BD6	D3D6	Test: stack too deep?
0BE3	D3E3	Check available memory
0C10	D410	Send canned error message, then:
0C3C	D43C	Warm start; wait for Basic command
0C5C	D45C	Handle new Basic line input
0D0F	D50F	Rebuild chaining of Basic lines
0D2E	D52E	Receive line from keyboard
0D59	D559	Crunch keywords into Basic tokens
0E1A	D61A	Search Basic for given line number
0E49	D649	Perform NEW
0E6A	D66A	Perform CLEAR
0E99	D697	Reset Basic execution to start
0EA7	D6A5	Perform LIST
0F68	D766	Perform FOR
102A	D828	Execute Basic statement
104B	D849	Perform RESTORE
1070	D86E	Perform STOP or END
1098	D896	Perform CONT
10B2	D8B0	Perform SAVE
10CB	D8C9	Perform LOAD
1114	D912	Perform RUN
1123	D921	Perform GOSUB
1140	D93E	Perform GOTO
116D	D96B	Perform RETURN/POP, then:
1197	D995	Perform DATA: skip statement
11A5	D9A3	Scan for next Basic statement
11A8	D9A6	Scan for next Basic line
11CB	D9C9	Perform IF, and perhaps:
11DE	D9DC	Perform REM: skip line
11EE	D9ED	Perform ON
120E	DA0C	Input fixed-point number
1248	DA46	Perform LET
12D1	DACF	Perform PRINT
133D	DB3A	Print string from memory
135A	DB57	Print single format character
1374	DB71	Handle bad input data
13A3	DBA0	Perform GET
13B5	DBB2	Perform INPUT
13E5	DBE2	Perform READ
14E2	DCDF	Canned Input error messages
14FC	DCF9	Perform NEXT
1558	DD55	Check type mismatch
157E	DD7B	Evaluate expression
16B5	DEB2	Evaluate expression within parentheses
16BB	DEB8	Check parenthesis, comma
16CC	DEC9	Syntax error exit
16D8	DED5	Setup for variables
1713	DF10	Set up function references
1752	DF4F	Perform OR, AND
1768	DF65	Perform comparisons
17D0	DFCD	Perform PDL
17DC	DFD6	Perform DIM
17E6	DFE3	Get variable name, location
18E6	E0ED	Setup array pointer
18FB	E102	Evaluate integer expression
1917	E11E	Find or make array
1AD7	E2DE	Perform FRE, and:
1AEB	E2F2	Convert fixed-to-floating
1AF8	E2FF	Perform POS
1AFF	E306	Check not Direct
1B0C	E313	Perform DEF
1B3A	E341	Check FNx syntax
1B4D	E354	Evaluate FNx
1BBE	E3C5	Perform STR$
1BCC	E3D5	Do string vector
1BDE	E3E7	Scan, set up string
1C49	E452	Build descriptor
1C7B	E484	Garbage collection
1D8E	E597	Concatenate
1DCB	E5D4	Store string
1DF4	E5FD	Discard unwanted string
1E2C	E635	Clean descriptor stack
1E3D	E646	Perform CHR$
1E51	E65A	Perform LEFT$
1E7D	E686	Perform RIGHT$
1E88	E691	Perform MID$
1EB0	E6B9	Pull string data
1ECD	E6D6	Perform LEN
1ED3	E6DC	Switch string to numeric
1EDC	E6E5	Perform ASC
1EEC	E6F5	Get byte parameter
1EFE	E707	Perform VAL
1F3D	E746	Get two parameters for POKE or WAIT
1F49	E752	Convert floating-to-fixed
1F5B	E764	Perform PEEK
1F72	E77B	Perform POKE
1F7B	E784	Perform WAIT
1F97	E7A0	Add 0.5
1F9E	E7A7	Perform subtraction
1FB0	E7B9	Perform addition
2095	E89E	Complement accum#1
20CC	E8D5	Overflow exit
20D1	E8DA	Multiply-a-byte
210A	E913	Constants
2138	E941	Perform LOG
2179	E982	Perform multiplication
21DA	E9E3	Unpack memory into accum#2
2205	EA0E	Test & adjust accumulators
2222	EA2B	Handle overflow and underflow
2230	EA39	Multiply by 10
2247	EA50	10 in floating binary
224C	EA55	Divide by 10
2257	EA60	Perform divide-by
225D	EA66	Perform divide-into
22F0	EAF9	Unpack memory into accum#1
2315	EB1E	Pack accum#1 into memory
234A	EB53	Move accum#2 to #1
235A	EB63	Move accum#1 to #2
2369	EB72	Round accum#1
2379	EB82	Get accum#1 sign
2387	EB90	Perform SGN
23A6	EBAF	Perform ABS
23A9	EBB2	Compare accum#1 to memory
23E9	EBF2	Floating-to-fixed
241A	EC23	Perform INT
2441	EC4A	Convert string to floating-point
24CC	ECD5	Get new ASCII digit
2501	ED0A	Constants
2510	ED19	Print IN, then:
2517	ED20	Print Basic line #
252B	ED34	Convert floating-point to ASCII
265B	EE64	Constants

2684	EE8D	Perform SQR
268E	EE97	Perform power function
26C7	EED0	Perform negation
26D2	EEDB	Constants
2700	EF09	Perform EXP
2753	EF5C	Series evaluation
279D	EFA6	RND constants
27A5	EFAE	Perform RND
27E1	EFEA	Perform COS
27E8	EFF1	Perform SIN
2831	F03A	Perform TAN
285D	F066	Constants
2895	F09E	Perform ATN
28C5	F0CE	Constants
2902	F10B	CHRGET sub for zero page
291F	F128	Basic cold start
29DC	F1D5	Perform CALL
29E5	F1DE	Perform IN#
29EC	F1E5	Perform PR#
2A2C	F225	Perform PLOT
2A39	F232	Perform HLIN
2A48	F241	Perform VLIN
2A56	F24F	Perform COLOR=
2A5D	F256	Perform VTAB
2A69	F262	Perform SPEED=
2A74	F26D	Perform TRACE, NOTRACE
2A7A	F273	Perform NORMAL, INVERSE
2A87	F280	Perform FLASH
2A8D	F286	Perform HIMEM:
2AAD	F2A6	Perform LOMEM:
2AD2	F2CB	Perform ONERR:
2B1F	F318	Perform RESUME
2B38	F331	Perform DEL
2B8C	F390	Perform GR
2B95	F399	Perform TEXT
2B9B	F39F	Perform STORE
2BB8	F3BC	Perform RECALL
2BD4	F3D8	Perform HGR2, HGR
2C0D	F411	Varous graphics subroutines
2EE5	F6E9	Perform HCOLOR=
2EFA	F6FE	Perform HPLOT
2F1A	F721	Perform ROT=
2F20	F727	Perform SCALE=
2F62	F769	Perform DRAW
2F68	F76F	Perform XDRAW
2F6E	F775	Perform SHLOAD
2FE0	F7E7	Perform HTAB

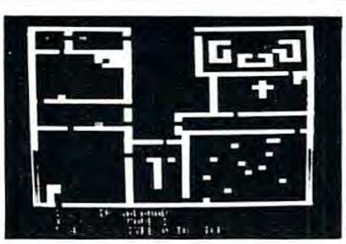
Applesoft memory map (Page 0)

Hex	Decimal	Description
000D	13	Search character
000E	14	Scan-between-quotes flag
000F	15	Input buffer pointer; # of subscripts
0010	16	Defa ult DIM flag
0011	17	Type: FF=string, 00=numeric
0012	18	Type: 80=integer, 00=floating point
0013	19	Flag: DATA scan; LIST quote; memory
0014	20	Subscript flag; FNX flag
0015	21	0=INPUT; $40=GET; $98=READ
0016	22	Comparison Evaluation flag
0024	36	Position on print line
0050-0051	80-81	Integer value (for GOTO etc)
0052-0054	82-84	Pointers for descriptor stack
0055-005D	85-93	Descriptor stack(temp strings)
005E-0061	94-97	Utility pointer area
0062-0066	98-102	Product area for multiplication
0067-0068	103-104	Pointer: Start-of-Basic
0069-006A	105-103	Pointer: Start-of-Variables
006B-006C	107-108	Pointer: Start-of-Arrays
006D-006E	109-110	Pointer: End-of-Arrays
006F-0070	111-112	Pointer: String-storage(moving down)
0071-0072	113-114	Utility string pointer
0073-0074	115-116	Pointer: Limit-of-memory
0075-0076	117-118	Current Basic line number
0077-0078	119-120	Previous Basic line number
0079-007A	121-122	Pointer: Basic statement for CONT
007B-007C	123-124	Current DATA line number
007D-007E	125-126	Current DATA address
007F-0080	127-128	Input vector
0081-0082	129-130	Current variable name
0083-0084	131-132	Current variable address
0085-0086	133-134	Variable pointer for FOR/NEXT
0087-008F	135-143	Work area, pointers, etc
0090-0092	144-146	Jump vector for functions
0093-009C	147-156	Misc numeric work area
009D	157	Accum#1: Exponent
009E-00A1	158-161	Accum#1: Mantissa
00A2	162	Accum#1: Sign
00A3	163	Series evaluation constant pointer
00A4	164	Accum#1 hi-ordeer (overflow)
00A5-00AA	165-170	Accum#2: Exponent, etc.
00AB	171	Sign comparison, Acc#1 vs #2
00AC	172	Accum#1 lo-order (rounding)
00AD-00AE	173-174	Series pointer
00B1-00C8	177-200	CHRGET subroutine; get Basic char
00B7	183	Sub entry: get prev character
00B8-00B9	184-185	Basic pointer (within subrtn)
00C9-00CD	201-205	Random number seed.
0200-02FF	512-767	Input buffer

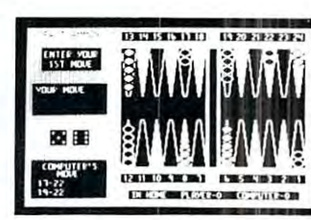
©

Exciting, entertaining software for the Apple II and Apple II Plus*

If you liked "Invaders", you'll love ASTEROIDS IN SPACE by Bruce Wallace. Your space ship is traveling in the middle of a shower of asteroids. Blast the asteroids with lasers, but beware — big asteroids fragment into small asteroids! The Apple game paddles allow you to rotate your space ship, fire its laser gun, and give it thrust to propel it through endless space. From time to time you will encounter an alien space ship whose mission is to destroy you, so you'd better destroy it first! High resolution graphics and sound effects add to the arcade-like excitement that this program generates. Runs on any Apple II with at least 32K and one disk drive.
On diskette — **$19.95**



FRACAS™ by Stuart Smith. A fantastic adventure game like no other — up to eight players can participate in FRACAS at the same time. Journey in the land of FAROPH, searching for hidden treasure while warding off all sorts of unfriendly and dangerous creatures like the Ten Foot Spider and the Headless Horseman. You and your friends can compete with each other or you can join forces and gang up on the monsters. Your location is presented graphically and sound effects enliven the battles. Save your adventure on diskette or cassette and continue it at some other time. Requires at least 32K of RAM.
Cassette: **$19.95** Diskette: **$24.95**



FASTGAMMON™ by Bob Christiansen. Sound, hi res color, and cartoons have helped maked this the most popular backgammon-playing game for the Apple II. But don't let these entertaining features fool you — FASTGAMMON plays serious backgammon. Requires at least 24K of RAM.
Cassette: **$19.95** Diskette: **$24.95**

BATTLESHIP COMMANDER™ by Erik Kilk and Matthew Jew. A game of strategy. You and the computer each start out by positioning five ships of different sizes on a ten by ten grid. Then the shooting starts. Place your volleys skillfully — a combination of logic and luck are required to beat the computer. Cartoons show the ships sinking and announce the winner. Sound effects and flashing lights also add to the enjoyment of the game. Requires at least 32K of RAM.
Cassette: **$14.95** Diskette: **$19.95**

QS QUALITY SOFTWARE

6660 Reseda Blvd., Suite 105, Reseda, CA 91335

WHERE TO GET IT: Call us at (213) 344-6599 for the name of the Quality Software dealer nearest you. If necessary, you may order directly from us. MasterCharge and Visa cardholders may place orders by telephone and we will deduct $1 from orders over $19 to compensate for phone charges. Or mail your order to the address above. California residents add 6% sales tax. SHIPPING CHARGES: Within North America orders must include $1.50 for first class shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00 — payable in U.S. currency.

*"Apple II" and "Apple II Plus" are trademarks of Apple Computer, Inc.

THE ATARI® GAZETTE

Designing Your Own Atari Graphics Modes

Craig Patchett
Program Design, Inc.
11 Idar Court
Greenwich, CT 06830

The graphics modes that Atari supplies with their 400 and 800 computers are nice, but what if you want a little more? For example, how about a large-type heading, with a smaller-type sub-heading below it, all over a graphics display? Terrific, you say, but you're not an Atari engineer? Don't worry about a thing. With this article, a little concentration, and some time in front of the keyboard, you'll have Atari graphics modes performing at the snap of your fingers.

First, a simple explanation of what we'll be doing. In a series of memory locations deep inside your Atari rests a special list of numbers that tell the computer which graphics mode it's in. Each time you change graphics modes, this list also changes. But wait a minute. Why a list of numbers instead of just one? Because there is one number for each graphics row on the screen. For example, in graphics mode 2 + 16 (no text window) there are twelve graphics rows, so there would be twelve numbers in the list. For graphics mode 7 + 16, there would be ninety six rows, or ninety six numbers. The table labeled **Modes and Screen Formats** in your Atari BASIC reference manual shows the number of rows in each graphics mode. We'll be referring to it again later.

As I said before, when you change graphics modes, using the GRAPHICS command, the list changes. It may become longer or shorter, depending on the mode, and the numbers in it will change. But the numbers will all be the same. Obviously, since they stand for the graphics mode of each row on the screen, if half of them were one number and the other half another, then half of the screen would be one mode and the other half another. This is not how Atari BASIC was designed. It is, however, what we want. So what we're going to be doing is changing the numbers in the list to make the screen behave the way we want it to. Let's take a look at exactly how it's done.

How Much of Each Mode Should I Have?

The first thing we have to do is figure out exactly how we want the screen to look. Let's take the example from the beginning of the article-a large-type heading (mode 2), with a smaller-type sub-heading below it (mode 1), all over a graphics display (mode 3). Unfortunately, we can't just decide to have, for instance, four rows of mode 2, two rows of mode 1, nd nine rows of mode 3. There's a simple rule we have to follow in deciding how many rows of each mode we're going to have.

You may already know that your television picture is made up of hundreds of little lines going across the screen from top to bottom (if you don't you know now!) If you look closely at the screen, you can probably see them. These lines are formed by a single beam of light that scans the screen very quickly (sixty times a second) to make the picture, so we'll call them scan lines. The part of the screen that your Atari lets you use for graphics has 192 of these lines.

Each graphics row is a certain number of scan lines "high". In mode 1, for example, each row is eight scan lines high. If you look at the **Table of Modes and Screen Formats** that I mentioned before, you'll see that there are twenty-four rows in mode 1 (remember, we're only interested in "full screen"). Surprise! Twenty-four rows, each eight scan lines high, means 8 x 24 = 192 scan lines in all. To figure out how many scan lines high the rows in other modes are, just look at the table and divide

192 by the number of rows in a full screen.

The reason we need to know all this is because we must make our new mode so that it has a total of 192 scan lines. No more, no less. This means you have to do a little bit of juggling around with the different modes you want to use, but it's really not all that difficult. I'll demonstrate with our example. Let's suppose we need three rows of mode 2 and two rows of mode 1. All we need to do is figure out how many rows of mode 3 we should have to make a total of 192 scan lines. We look at the table and figure out that in mode 2, each row is sixteen (192 scan lines/12 rows) scan lines high. Since we want three rows of mode 2, that makes fourty-eight scan lines so far. Similarly, we want two rows of mode 1, which uses eight (192 scan lines/24 rows) scan lines for each row. So that makes another sixteen scan lines, or sixty-four all together, which leaves us 192 - 64 = 128 scan lines still left over. We'll use these for mode 3. We look at the table again and see that mode 3 uses eight scan lines for each row also, so how many rows do we need? 128 leftover scan lines/8 scan lines per row of mode 3 = 16 rows of mode 3.

So now we know that our graphics mode is going to have three rows of mode 2, two rows of mode 1, and sixteen rows of mode 3. Let's tell the computer.

How Do I Tell The Computer?

We have to start by putting the Atari in a graphics mode it understands. Of course, we can't use just any mode, but this time the rule is a lot easier. Out of the modes you're going to be using, take the one that uses the most memory (look at the table under "RAM required"). In our example, mode 1 uses the most memory, so the first line in our program is:

10 GRAPHICS 1

The next step is to find out where the list of numbers begins. Since it isn't always in exactly the same place, we must PEEK into the computer's memory at two locations that tell us where it is. Since we'll need to use the number that tells us where the list begins later, we'll give it a name:

20 BEGIN = PEEK(500) + PEEK(561)*256 + 4

This line will always be the same no matter what modes you are going to be mixing.

The third step can be ignored if the mode you want at the top of the screen is the same as the one that uses the most memory. If not, as in our example (mode 2 is at the top of the screen, mode 1 uses the most memory), then we have to change the number in the memory location right before the beginning of the list. The table below shows what number to use for the mode at the top of the screen.

MODE	0	1	2	3	4	5	6	7	8
NUMBER	66	70	71	72	73	74	75	77	79

So, for our example, we would need:

25 POKE BEGIN-1,71

Remember, only do this step if the first graphics row is *not* the same mode as the one that uses the most memory.

Now we just have to go down the list and change the numbers that need to be changed. The numbers for the graphics mode with the most memory are already correct, since we start in that mode. Therefore, all we have to change are the other numbers. In our example, that would be the numbers for mode 2 and mode 3. To make the necessary changes, we simply POKE BEGIN + row number with the correct number for the mode we want in that row. What are the correct numbers? Just subtract sixty-four from the numbers in the table I gave above. That would mean, for example, seven for mode 2, and eight for mode 3. So we have:

30 POKE BEGIN + 2,7:POKE BEGIN + 3,7

which takes care of mode 2. Note that we didn't POKE BEGIN + 1. This was automatically taken care of when we POKEd BEGIN-1 in line 25. Remember that we also don't have to worry about the numbers for mode 1, since they are already correct. Therefore, all that's left is to change the numbers for mode 3. Since we want sixteen rows of mode 3, which means changing sixteen numbers, we'll use a FOR/NEXT loop to make life easier:

40 FOR ROW = 6 TO 21:POKE BEGIN + ROW, 8:NEXT ROW

Now the list has the correct mode numbers in it. There's still one more thing we must do. Since there may be a fewer number of rows now than there were in the mode we told the computer to start with, we have to tell the computer where the new end of the list is. We do this by POKEing the number sixty-five into the row number right after the last one we used. This tells the Atari to go back to the beginning of the list. We also tell it where the beginning is. For our example:

50 POKE BEGIN + 22,65:POKE BEGIN + 23, PEEK(5 60):POKE BEGIN + 24, PEEK(561)

And now we're done. Note that the only changes that you would need to make in line 50 when designing your own modes is in the numbers 22, 23, and 24. These are just the three row numbers after the last one you use on the screen.

How Often Do I Have To Do All This?

This whole procedure must be repeated whenever you want to use a specially designed graphics mode. You can't skip any of the steps except for the third one, and then only under the condition I already described.

So Now What Do I Do?

The last thing I'm going to cover, briefly, in this article is how to print and draw in your new mode. This only applies if the row you want to print or plot on is within the normal range for whatever mode it is. In simpler terms, if we had put the sixteen rows

of mode 3 at the top of the screen, and mode 2 at the bottom, then mode 2 would have been in rows 19,20, and 21. But mode 2 usually only has twelve rows, so if you tried to print on line 19 you would get an error message. Now, there is a way around this, but it's somewhat complicated so I'm going to leave it for a future article. For now, however, you can use the following rules as long as you stay within the normal range of the mode you're working with.

The first thing you have to do is POKE location eighty-seven with the number of the graphics mode for the row you want to PRINT or PLOT in. Next, POSITION the cursor and PRINT, or PLOT and DRAWTO. When you tell the Atari to POSITION X,Y or PLOT X,Y, the X value is still the number of spaces in from the left that you want to go. The Y value is still the number of rows down from the top that you want to go, but you may have to experiment with different values to get it exactly where you want it. Just make sure that you remember to POKE 87 with the mode number you're going to PRINT or PLOT in.

To help you understand what I just said, and to show off the example mode we've been working on, try entering these lines, as well as the other ones that are included throughout the article. When you've entered them in, just RUN the program, and BREAK it when you're done. Notice that the commands for colors are the same in the new mode; that is, you can still print different color letters and use the COLOR command for graphics points, etcetera. The one difficulty that might arise is when you mix mode 0 with other modes. Since mode 0 has a different background color (blue) than the other modes (black) you will have to use the SETCOLOR command to make the mode 0 rows invisible. Otherwise, you should have no problems whatsoever.

```
60 SETCOLOR 4,4,2:REM BACKGROUND
70 POKE 87,2:POSITION 6,0:PRINT #6;"THIS
 IS":POSITION 3,1:PRINT #6;"GRAPHICS MOD
E":POSITION 8,2:PRINT #6;"TWO"
80 POKE 87,1:POSITION 6,3:PRINT #6;"this
 is":POSITION 1,4:PRINT #6;"graphics mod
e one"
90 POKE 87,3:COLOR 3:FOR LINE=1 TO 3:PLO
T 15,LINE*5+8:DRAWTO 22,LINE*5+8:NEXT LI
NE:PLOT 22,13:DRAWTO 22,23
100 GOTO 100:REM KEEP GRAPHICS ON SCREEN
```

Look Ma, New Modes!

That's all there is to making your own graphics modes on your Atari computer. The easiest way to make sense of everything I've covered here is to **experiment.** Start off by changing the example program and watching what happens, and then try designing your own modes. Just a little practice and in no time you'll be an expert. Above all, have fun doing it; after all, the Atari works for you, not the other way around.

©

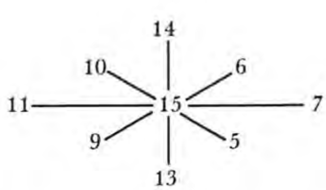
What To Do If You Don't Have Joysticks

Steven Schulman

Use of joysticks with the ATARI computer can add excitement to your programs. But what do you do if you don't have joysticks yet and aren't ready to buy them? Are you out of luck? Do you have to type in numbers to select from a menu of answers? Does it mean you can't use games like *IRIDIS'* ZAP or the latest from your computing magazines? No! There's another way.

In amongst the bits and bytes that make up the memory of your ATARI, any time you press a key on your keyboard the value of the 764th word changes. By taking a peek at what number is there you can find out which key it was. Listing I shows you how to find out what the value will be when any key is pressed. Try running it and pressing different keys, shifted and unshifted, reverse video, etc. When you finish use the break key to stop the program.

"How does this help solve my problem of not having joysticks?" you may ask. To see this you have to know what happens when you use the joysticks. If your program has a line I = STICK(1), the value of I will be one of 9 possible values depending on the position of the joystick when that line is reached. The values will be



where the value of I = 15 means that the joystick is in the upright position. In addition, J = STRIG(1) will have a value J = 0 if the fire button is pressed and a value of J = 1 if the fire button is not pressed.

Returning to what we know about the value of the last key pressed, we found that the values for the arrows were:

= 14 = 7
= 15 = 6

and the values for the shifted arrows were

Shift = 78 Shift = 71
Shift = 79 Shift = 70

Finally, the value for the space bar is 33.

We can therefore have the same results as we would get from using a joystick by using the arrows, shift arrows and space bar. The shift bar will be our firing button, the arrows will be the obvious up, down, left and right, and the shift up will be to the upper left, the shift down will be to the upper right, the shift left will be to the lower left, and the shift right will be to the lower right. Any other key or no key at all being pressed is equal to the joysticks being in an upright position.

The routine in listing II will play the part of a joystick. After calling the subroutine the value of 1 will be the same as would have been returned by I = STICK(1) and the value of J will be the same as what would have been returned by J = STRIG(1). When you do buy your joysticks, simply replace the subroutine call and remove the subroutine from your program. Happy computing!

Listing I

```
100 I = PEEK (764)
110 ? "I = ";I : REM PRINT THE VALUE OF THE KEY
PRESSED
120 POKE 764,255 : REM TELL THE COMPUTER
THAT NO KEY WAS PRESSED
130 FOR PAUSE = 1 to 500 : NEXT PAUSE : REM
SLOW DOWN THE MACHINE SO YOU CAN READ
THE RESULTS
140 GOTO 100
```

Listing II

```
100 JOYSTICK = 1000 : REM LOCATION OF
SUBROUTINE
110 GOSUB JOYSTICK : REM CHECK THE 'JOYSTICK'
120 ?"THE 'JOYSTICK' HAS VALUE = ";I
130 ?"THE 'FIRE BUTTON' HAS VALUE = ";J
140 FOR PAUSE = 1 TO 500 : NEXT PAUSE
150 GOTO 110
1000 REM JOYSTICK SUBROUTINE
1010 I = PEEK (764)
1020 J = 1
1030 POKE 764, 255
1040 IF I = 14 THEN I = 14 : RETURN
1050 IF I = 79 THEN I = 6 : RETURN
1060 IF I = 7 THEN I = 7: RETURN
1070 IF I = 70 THEN I = 5 : RETURN
1080 IF I = 15 THEN I = 13 : RETURN
1090 IF I = 71 THEN I = 9 : RETURN
1100 IF I = 6 THEN I = 11 : RETURN
1110 IF I = 78 THEN I = 10 : RETURN
1120 IF I = 33 THEN I = 15 : J = 0 : RETURN :
REM FIRE BUTTON
1130 I = 15 : RETURN
```

©

Screen Print From Machine Language On The Atari

Larry Isaacs

If you are doing machine language programming on the ATARI, it can be very advantageous to know where some of the operating system subroutines can be found. I can provide you with only one at this time, but it's one of the handier ones. This is the output subroutine for the Editor device. It accepts the full ATASCII character set, printing the displayable character on the screen, or executing the control characters. To use the routine, simply load the character into the accumulator and execute a JSR $F6A4 instruction. The only other fact needed is that the X and Y registers aren't preserved by this subroutine.

To illustrate the use of this subroutine, the DUMP program is provided. This program also illustrates one way of using machine language with BASIC. The program asks for starting and ending addresses, which should be given in hex. Then the requested memory is dumped on the screen by a machine language program executed by the USR command.

Naturally, before the machine language can be executed, it must be placed in memory. This is done by the BASIC subroutine in statements 10200-10430. This subroutine loads machine code found in DATA statements, which begin at line 20000 in this program. The first thing the subroutine does is read the number of bytes in the machine language program. It then dimensions DYM$ to length 1 and an array called STORAGE of sufficient size to hold the machine code.

The subroutine then starts reading the data as strings and POKEing the appropriate code. If the string read doesn't start with a special character (".","*","+","=", or "!") then the string is assumed to be two hex characters which are stored in the next available byte. If the string begins with a ".", then the string is assumed to be a comment and is ignored. If it begins with an "*", the subroutine assumes the rest of the string is four hex characters which form a two byte address. This address is POKE'd low byte first, then the high byte. If the string begins with a "+", the rest of the string is assumed to be four hex characters which form a two byte displacement from the beginning location of the code. This displacement is added to the beginning location of the code to form a two byte address. This address is also POKE'd low byte first, followed by high byte. If the first character is an "=", then the rest of the string is assumed to be a displacement as with "*". However, once the address is computed, the current poke location plus one is subtracted from this address to form a one byte displacement which is POKE'd into the next location. Finally, if the first character of the string is an "!", the subroutine stops loading machine code. The rest of the string is assumed to be a two byte displacement as with the "*", and the computed address is checked with the current poke location to see if it matches. If they don't match, it's likely that you've miscounted some bytes and that some of the displacements given by strings starting with the "*" or "=" character are in error.

This may seem somewhat complicated, but it really makes it fairly simple to write relocatable code. This relocability is necessary because you don't know where the code will be loaded until the program is running. Relative addresses used by branch instructions may be given as a hex byte or as an "=" followed by the displacement from the beginning of the program. Internal absolute addresses should be given with a "+" followed by the displacement. And finally, external addresses can be specified by giving two hex bytes, or by an "*" followed by the address.

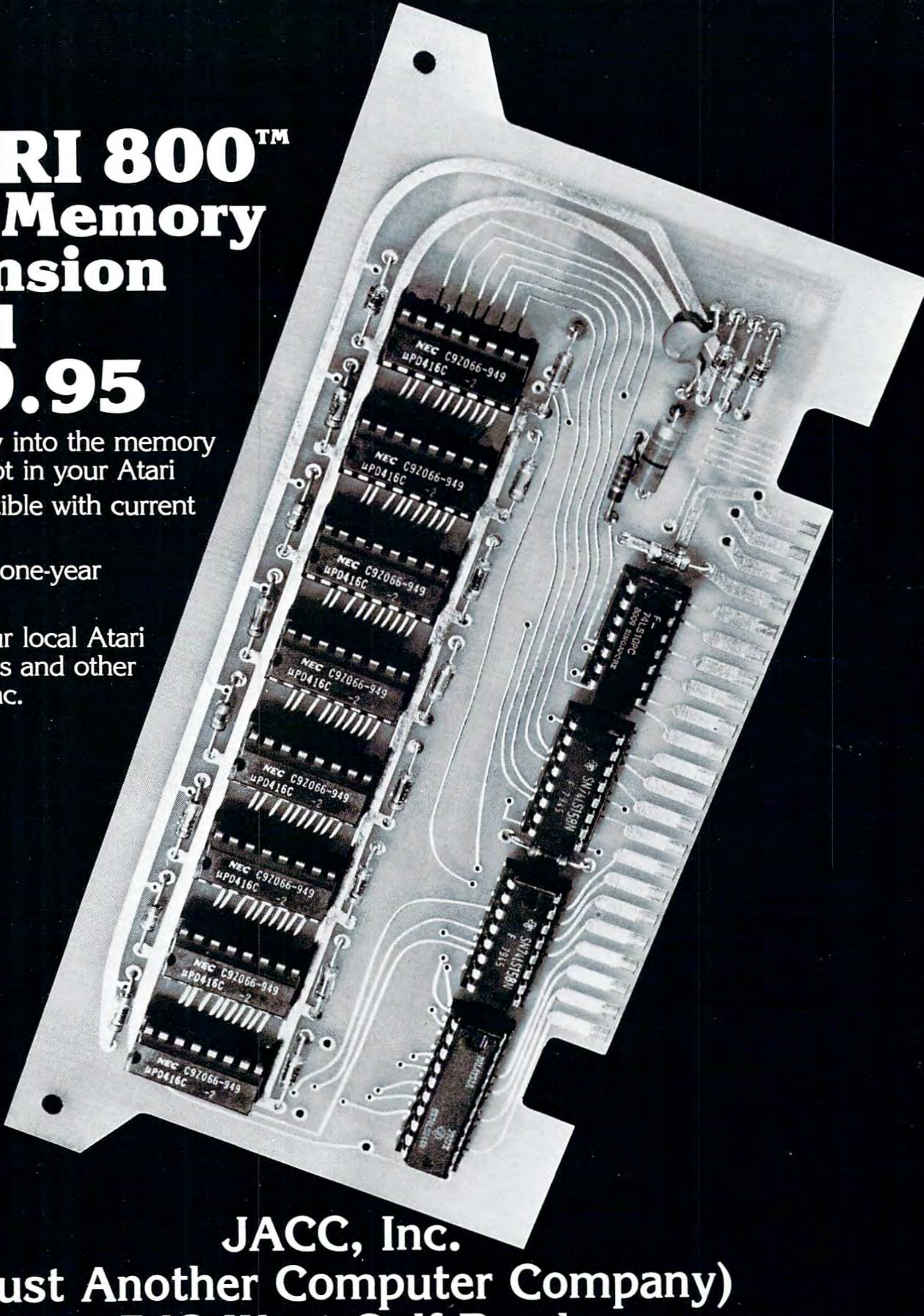
Once the code is loaded, ADR(DMY$) gives the first location. This also happens to be the entry point of the machine language dump program. Now the dump routine can be executed by calling for the USR function to be executed with ADR(DMY$) as its address. This is done on line 80 of the BASIC program.

It is important to note that the dump routine can only be executed while the BASIC program is running. Trying to execute it by a direct command will not work because the direct command gets inserted in between the end of the program and where the machine code has been poked. This will cause the machine code to be moved; and since it contained some internal absolute addressing, it will not execute properly any more. If the code contains no internal absolute addressing, it can be executed by a direct command.

The machine code is fairly simple, so you should be able to understand what it is doing. Upon entry, the machine code first checks to see if the right number of parameters are present. If not, the parameters are pulled off the stack and the program returns to BASIC. If the correct number (2) is present, the machine code will dump the requested memory, printing 8 bytes per line.

Hopefully you will find some of the techniques used in this program useful, as well as the program itself.

ATARI 800™
16-K Memory Expansion Board
$179.95



- Plugs directly into the memory expansion slot in your Atari
- Fully compatible with current hardware
- Backed by a one-year warranty
- Inquire at your local Atari Dealer for this and other fine JACC, Inc. products

JACC, Inc.
(Just Another Computer Company)
543 West Golf Road
Arlington Heights, IL 60005

Dealer/Distributor Inquiries Invited

Dealer Line: 312-364-6268

On the west coast contact Micro Distribution, Inc.
2612 Croddy Way, Santa Anna, CA 92704 714-641-0205

```
1 DIM SA$(4),EA$(4)
10 GOSUB 10200
20 PRINT "INPUT STARTING ADDRESS";
25 INPUT SA$
30 PRINT "INPUT ENDING ADDRESS";
35 INPUT EA$
40 WORD$=SA$:GOSUB 10100
50 SA=NWORD
60 WORD$=EA$:GOSUB 10100
70 EA=NWORD
80 DUMMY=USR(ADR(DMY$),SA,EA)
90 GOTO 20
10000 REM COMPUTE NBYTE FROM HEX$
10010 I=1:GOSUB 10040:NBYTE=X*16
10020 I=2:GOSUB 10040:NBYTE=NBYTE+X
10030 RETURN
10040 X=ASC(HEX$(I,I))-ASC("0")
10050 IF "0"<=HEX$(I,I) AND HEX$(I,I)<="
9" THEN RETURN
10060 IF "A"<=HEX$(I,I) AND HEX$(I,I)<="
F" THEN X=X-7:RETURN
10070 STOP :REM ERROR
10100 REM COMPUTE NWORD FROM WORD$
10110 HEX$=WORD$(1,2):GOSUB 10000:NWORD=
NBYTE*256
10120 HEX$=WORD$(3,4):GOSUB 10000:NWORD=
NWORD+NBYTE
10130 RETURN
10200 REM PUT THE CODE
10210 READ N:REM NUMBER OF BYTES
10220 DIM CODE$(40),HEX$(2),WORD$(4),DMY
$(1),STORAGE(N/6+1)
10230 PC=ADR(DMY$)
10240 READ CODE$
10245 IF CODE$(1,1)="." THEN GOTO 10240
10250 IF CODE$(1,1)="*" THEN GOTO 10300
10260 IF CODE$(1,1)="+" THEN GOTO 10310
10265 IF CODE$(1,1)="=" THEN GOTO 10350
10270 IF CODE$(1,1)="!" THEN GOTO 10410
10280 HEX$=CODE$(1,2):GOSUB 10000
10290 POKE PC,NBYTE:PC=PC+1:GOTO 10240
10300 WORD$=CODE$(2,5):GOSUB 10100:GOTO
10320
10310 WORD$=CODE$(2,5):GOSUB 10100:NWORD
=NWORD+ADR(DMY$)
10320 NBYTE=INT(NWORD/256)
10330 POKE PC,NWORD-NBYTE*256
10340 PC=PC+1:GOTO 10290
10350 WORD$=CODE$(2,5):GOSUB 10100
10360 NBYTE=ADR(DMY$)+NWORD-(PC+1)
10370 IF NBYTE>127 THEN STOP
10380 IF NBYTE<-128 THEN STOP
10390 IF NBYTE<0 THEN NBYTE=NBYTE+256
10400 GOTO 10290
10410 WORD$=CODE$(2,5):GOSUB 10100
10420 IF NWORD=PC-ADR(DMY$) THEN RETURN
10430 STOP :REM ERROR
20000 DATA 137
```

```
20010 DATA .0000,4C,+0030,.JMP START
20020 REM INCPNTR
20030 DATA .0003,E6,D4,.    INC PNTR
20040 DATA .0005,D0,=0009,.BNE @1
20050 DATA .0007,E6,D5,.    INC PNTR+1
20060 REM @1
20070 DATA .0009,60,.       RTS
20080 REM PRBYTE
20090 DATA .000A,48,.       PHA
20100 DATA .000B,4A,.       LSR A
20110 DATA .000C,4A,.       LSR A
20120 DATA .000D,4A,.       LSR A
20130 DATA .000E,4A,.       LSR A
20140 DATA .000F,20,+0015,.JSR PRNYBLE
20150 DATA .0012,68,.       PLA
20160 DATA .0013,29,0F,.    AND #$0F
20170 REM PRNYBLE
20180 DATA .0015,C9,0A,.    CMP #$0A
20190 DATA .0017,30,=001B,.BMI @2
20200 DATA .0019,69,06,.    ADC #$06
20210 REM @2
20220 DATA .001B,69,30,.    ADC #$30
20230 DATA .001D,20,*F6A4,.JSR OUTCHR
20240 DATA .0020,60,.       RTS
20250 REM TSTPNTR
20260 DATA .0021,38,.       SEC
20270 DATA .0022,AD,+002D,.LDA EA
20280 DATA .0025,E5,D4,.    SBC PNTR
20290 DATA .0027,AD,+002E,.LDA EA+1
20300 DATA .002A,E5,D5,.    SBC PNTR+1
20310 DATA .002C,60,.       RTS
20320 REM EA
20330 DATA .002D,00,00,.    .WORD
20340 REM COUNT
20350 DATA .002F,00,.       .BYTE
20360 REM START
20370 DATA .0030,68,.       PLA
20380 DATA .0031,F0,=0009,.BEQ @1
20390 DATA .0033,C9,02,.    CMP #$02
20400 DATA .0035,F0,=003E,.BEQ CONTINUE
20410 DATA .0037,AA,.       TAX
20420 REM @3
20430 DATA .0038,68,.       PLA
20440 DATA .0039,68,.       PLA
20450 DATA .003A,CA,.       DEX
20460 DATA .003B,D0,=0038,.BNE @3
20465 DATA .003D,60,.       RTS
20470 REM CONTINUE
20480 DATA .003E,68,.       PLA
20490 DATA .003F,85,D5,.    STA PNTR+1
20500 DATA .0041,68,.       PLA
20510 DATA .0042,85,D4,.    STA PNTR
20520 DATA .0044,68,.       PLA
20530 DATA .0045,8D,+002E,.STA EA+1
20540 DATA .0048,68,.       PLA
20550 DATA .0049,8D,+002D,.STA EA
20560 REM DUMP
20570 DATA .004C,A9,9B,.    LDA #EOL
```

```
20580 DATA .004E,20,%F6A4,.JSR OUTCHR
20590 DATA .0051,A9,24,.   LDA #'$
20600 DATA .0053,20,%F6A4,.JSR OUTCHR
20610 DATA .0056,A5,D5,.   LDA PNTR+1
20620 DATA .0058,20,+000A,.JSR PRBYTE
20630 DATA .005B,A5,D4,.   LDA PNTR
20640 DATA .005D,20,+000A,.JSR PRBYTE
20650 DATA .0060,A9,20,.   LDA #'
20660 DATA .0062,20,%F6A4,.JSR OUTCHR
20670 DATA .0065,A9,08,.   LDA #$08
20680 DATA .0067,8D,+002F,.STA COUNT
20690 REM LOOP
20700 DATA .006A,A9,20,.   LDA #'
20710 DATA .006C,20,%F6A4,.JSR OUTCHR
20720 DATA .006F,A0,00,.   LDY #$00
20730 DATA .0071,B1,D4,.   LDA (PNTR)Y
20740 DATA .0073,20,+000A,.JSR PRBYTE
20750 DATA .0076,20,+0003,.JSR INCPNTR
20760 DATA .0079,CE,+002F,.DEC COUNT
20770 DATA .007C,D0,=0069,.BNE LOOP
20780 DATA .007E,20,+0021,.JSR TSTPNTR
20790 DATA .0081,10,=004B,.BPL DUMP
20800 DATA .0083,A9,9B,.   LDA #EOL
20810 DATA .0085,20,%F6A4,.JSR OUTCHR
20820 DATA .0088,60,.      RTS
20830 DATA !0089
```

©

COMPUTE needs you! Let us know what interesting applications you're coming up with for your Atari.

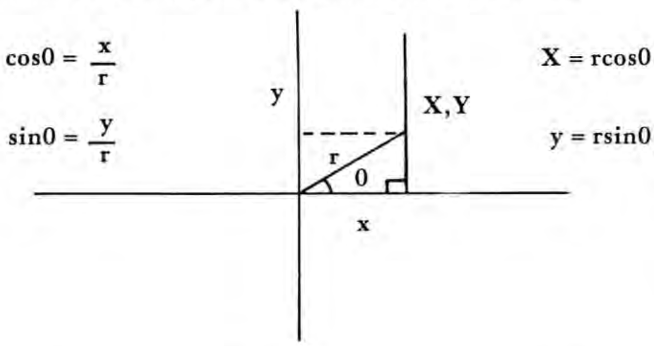
Star Fleet to All Cruisers...

Who's the current champ of Star Raiders? Send in your best score, comments and playing strategies to COMPUTE, P.O. Box 5406, Greensboro, NC 27403.
Attn: "Atari Gazette"

Graphics Of Polar Functions

Henrique Veludo
353 West 56th Str. #116
NYC NY 10019

This program will plot polar functions such as roses, spirals, polygons, on the high resolution screen of the ATARI 800, with input from the programmer. The general equations for converting the polar coordinates to rectangular coordinates are as follows:



First the program will display a function menu (line 100), then ask the user to input which function to display, together with its parameters, INCR(ement) and SC(ale). The INCR(ement) is the interval in degrees that the computer uses to "increment" the angle T from 0° to 360°. One must decide whether the speed of execution or accuracy in plotting is preferable. A small INCR(ement), e.g. 0.1, will draw a very accurate graph very slowly. A larger INCR(ement), e.g. 5.0, will draw much faster and less accurately. An INCR of 1.0 is a good compromise. The SC(ale) is included to allow the graph to fill most of the screen. Without it, some functions will appear too small, others will be too large to plot. A SC(ale) between 10 and 100 should do for most functions. Lines 220 to 226 check for a 0 input that might confuse the program and display an error message. Line 230 asks if the x-y axes are to be displayed and lines 390-395 display them. Lines 300-370 will select random colors and intensities (with enough separation to be visible). Lines 400-690 contain the calculation and plotting routines for x,y. In line 410 the variable U is included for use with the spiral function and dictates how many revolutions the spiral will have; it can be changed at line 222. Line 420 converts degrees to radians (in this context the program seems to work better with radians but it could be converted to degrees, with the DEG function, and changing the values of the functions). Line 430 will direct the program to the proper function chosen in the input. Lines 610-620 calculate the x,y coordinates. Line 630 will check for an out-of-range cursor, stop the drawing, and avoid an error message. Line 670 will activate the buzzer to signal that the plotting is over. Lines 680-690 wait for a key to be pressed to clear the screen and return to the menu. If the buzzer sounds without anything being plotted, it means that the function is too large to plot. (Decrease the SC(ale) value to continue.) I chose to use random-selected colors. They could be chosen by the user in an input statement as well (where you input the parameters after the menu display).

Here are some values for the functions that work beautifully:

R = Q:SC = 4:INCR = 60
R = 2(I-SIN(Q)):SC = 20
R = COS(2 SIN(6 (Q))):SC = 90
R = SIN(COS(IOO Q)):SC = 90
R = COS(2 SIN(2 Q)):SC = 90
R = I:INCR = 45:SC = 60 polygon

R = 2(I + COS(Q)):SC = 20
R = SIN(3(Q)):SC = 80
R = SIN(4COS(2Q)):SC = 90
R = COS(3SIN(Q)):SC = 90
R = COS(SIN(IOO Q)):SC = 90
R = I:INCR = I20:SC = 80 triangle

```
10 REM PROGRAM TO PLOT POLAR FUNCTIONS
20 REM BY HENRIQUE VELUDO FOR ATARI 800
80 DIM A$(1)
90 ? ")"
100 POSITION 7,1:? "GRAPHS OF POLAR FUNC
TIONS"
110 POSITION 2,3:? "FUNCTION MENU:":?
120 ? "      1)R=B*Q                  SPIRAL
"
130 ? "      2)R=A*(1+COS(Q))         CARDIO
ID"
140 ? "      3)R=A*(1-SIN(Q))"
150 ? "      4)R=A*SIN(B*Q)           ROSE"
160 ? "      5)R=A*COS(B*Q)"
170 ? "      6)R=COS(A*SIN(B*Q))"
180 ? "      7)R=SIN(A*COS(B*Q))"
190 ? "      8)R=A                    POLYGO
N"
200 ? :? :? "INPUT:":?
210 ? "FUNCTION #,A,B,INCR.,SC.";:INPUT
N,A,B,INCR,SC
220 IF N=0 THEN N=1
222 IF N=1 THEN U=4
224 IF A=0 THEN A=1
226 IF B=0 THEN B=1
230 ? :? :? "DO YOU WANT THE X-Y AXES DI
SPLAYED";
240 INPUT A$:IF A$(1,1)="Y" THEN W=1
300 COLOR 1:GRAPHICS 24
310 I=INT(RND(1)*16)
320 L1=INT(RND(1)*8)*2
330 L2=INT(RND(1)*8)*2
```

```
340 IF ABS(L1-L2)<4 THEN 320
350 SETCOLOR 4,I,L1
360 SETCOLOR 2,I,L1
370 SETCOLOR 1,I,L2
380 IF W<>1 THEN 410:REM --DISPLAY AXES?

390 FOR I=0 TO 319 STEP 4:PLOT I,96:NEXT
 I
395 FOR I=0 TO 191 STEP 3:PLOT 160,I:NEX
T I
400 REM ----PLOTTING CALCULATION
410 FOR T=0 TO 360*U STEP INCR
420 Q=T/57.3
430 ON N GOTO 510,520,530,540,550,560,57
0,580
500 REM ----EQUATIONS FOR R
510 R=B*Q:GOTO 610
520 R=A*(1+COS(Q)):GOTO 610
530 R=A*(1-SIN(Q)):GOTO 610
540 R=A*SIN(B*Q):GOTO 610
550 R=A*COS(B*Q):GOTO 610
560 R=COS(A*SIN(B*Q)):GOTO 610
570 R=SIN(A*COS(B*Q)):GOTO 610
580 R=A:GOTO 610
600 REM PLOTTING X,Y
610 X=INT((R*COS(Q))*SC)
620 Y=INT((R*SIN(Q))*SC)
630 IF ABS(X)>159 OR ABS(Y)>95 THEN 670
640 IF T=0 THEN PLOT 160+X,96-Y
650 DRAWTO 160+X,96-Y
660 NEXT T
670 FOR I=1 TO 75:POKE 53279,0:NEXT I
675 W=0
680 U=1:OPEN #1,4,0,"K:":GET #1,X:CLOSE
#1
690 PUT #6,125:GOTO 90
```

©

Reading the ATARI Keyboard on the fly

James L. Bruun

For most programs the normal method of using the INPUT statement to get keyboard characters into a program is perfectly satisfactory. There are times, however, when we need to get a keystroke without stopping the program to wait for a key to be struck.

The ATARI computer has all the features needed to enable the programmer to check the keyboard without waiting for an INPUT statement to get the character. Memory location 764 retains a key code for the last key pressed. Further, when the RUN command is executed, that cell is set to 255 to indicate that no key has been pressed. During the running of a program, that location can be POKEd with a 255 to indicate that we've checked it since the last key was pressed.

The following program illustrates the use of these features in a subroutine. First, initialize an I/O buffer and string variable.

```
10 OPEN #1,4,0,"K:"
20 DIM CHAR$(1)
```

Then build the subroutine. Always precede your block of subroutines with an END statement to prevent accidental execution.

```
30 PRINT "(ESC) (CLEAR)"
40 POKE 752,1
50 GOSUB 5000
60 IF CHAR = 0 THEN 50
70 POSITION 5,5
80 PRINT "CHARACTER = (";CHAR$;")"
90 GOTO 50
```

Most programs that would need this feature would perhaps be doing complex things if the keystroke has not occurred, but in this one we have chosen to 'do nothing' until a key is pressed. ©

THE PET® GAZETTE

User's Report: Waterloo Structured Basic for the PET

P. T. Spencer
7 Brightside Drive
West Hill, Ontario
Canada M1E 3Y8

Waterloo Structured Basic comes in the form of an EPROM which sits at address hex 9000. After SYS'ing to 9*4096, you have all of standard PET basic, plus the following statements: IF (without THEN or GOTO), ELSE, ENDIF, ELSEIF, IF-THEN-QUIT, LOOP, ENDLOOP, WHILE, UNTIL, PROC, ENDPROC, and CALL. You also can insert as many blanks as you wish at the beginning of each basic line.

The EPROM chip comes with a serial number, complete instructions for installation, a 161 page manual aimed at beginners to structured programming, and a purchaser's registration card. Future updates to the chip are said to be free to registered purchasers, provided the chip is returned for reburning. The list price for this package is stated to be $150, with substantial discounts for educational institutions and bulk orders from users' clubs. For example, I purchased mine as part of a group of about twenty at the Toronto PET Users' Club for $61.50.

The idea behind the Waterloo structured approach is apparently that a program should be readable to someone else, or to the programmer himself after one or two years. To aid comprehension, you are supposed to indent freely, any use of GOTO's hither and thither.

After installing my chip and reading the instruction manual, I sat down to redo a routine that I had written in standard PET Basic the week before (see figures 1 and 2). The subroutine is one for a general file management program I have.

The file management program itself allows me to create a file, with the number of fields per file record set at startup. For example, I have a house inventory with the fields set as description, replacement value, date, and insurance category. Another file is a class list with the fields as student name, marks for N tests, and average mark, where N is generally different for each different class. Numeric fields are stored as strings (saves space) until calculations, if any, need to be done.

The subroutine shown in figures 1 and 2 allows me to change output format to the printer, so that the file can be printed as a table without my having to stop the program and manually change the printer formatting line each time I want to print a different file table. There are probably better ways to do this than the one shown here, but this method illustrates the difference between Waterloo and standard basic quite nicely.

Figure 1, the standard basic version, is not incomprehensible, but neither does it go out of its way to be clear. However, it does have the advantage of being only 528 bytes long, whereas the Waterloo version is 831 bytes long. The Waterloo version, however, looks nicer and probably will be considerably esier to understand six months down the road.

The first, and most important, disadvantage of Waterloo basic that I ran into in writing the code in figure 2 is that when Waterloo Basic is enabled, you can't use Basic Aid, Brett Butler's Trace, or Programmer's Toolkit. I missed the convenience of being able to race the cursor around the screen at high speed, being able to trace execution to find bugs, being able to renumber when there was no space between lines and I had to add a line, and having the next available line number automatically appear on the screen.

Since figure 2 gives the same results as figure 1, it presumably must be a correct, if perhaps not particularly elegant, use of Waterloo basic. Lines 6000 to 6380 are the Waterloo equivalent of GOSUB-RETURN, the difference being that a procedure can have a name, the name may be as long as you wish, and thus can be much more informative.

Lines 6090 to 6130 illustrate the use of the IF-ELSE-ENDIF construction. I found it quite difficult to break out of the IF-THEN GOTO habit. I had become so used to this in the two years I have had a PET that it had become almost automatic.

The WHILE-ENDLOOP construction in lines 6190 to 6230 is handy, as the WHILE condition is evaluated first, unlike the standard FOR-NEXT construction, which goes through the loop once regardless of what value the index variable has.

Lines 6270 and 6280 could probably have been integrated as one, but I liked the symmetry with line 6220. The IF-THEN-QUIT construction is much more convenient than the standard IF-THEN-set index variable to maximum and GOTO NEXT that must be used to keep the stack clean. The short cut in lines 6120 and 6160 of figure 1 is not particularly recommended, as sooner or later it will probably cause an ?OUT OF MEMORY ERROR because of stack problems.

Lines 6040, 6330, and 6340 took some advance thinking, as I find it much easier to let it all happen at the end as an INPUT S2$: IF S2$ "Y" THEN 6050 construction, especially when in Waterloo basic I faced the prospect of having to go back and change the identation of most of what had been written. In fact it was this that first decided me to sit down and write out the Waterloo code before hacking away at the keyboard, a blessing in disguise, as the code in figure 2 worked with much less debugging that that in figure 1, which WAS composed at the keyboard.

In summary, I would recommend Waterloo basic if you usually write programs longer than 4K, if your friends call your efforts pathologically complicated, or if you have already been trained in structured programming. On the other hand, I would not recommend it if you are addicted to machine language utilities such as Toolkit, Trace, or Basic Aid, or if you have less than 16K of RAM (structured programming trades space for readability). Another consideration is that if Waterloo basic gains the same acceptance as their WATFOR and WATFIV did with Fortran, it may not be long before it becomes the industry standard, in which case you will HAVE to have it unless you plan to use only your own programs.

Figure 1: Standard Basic Coding

```
10 GOSUB 6000
20 STOP
6000 PRINT"DYNAMIC FORMATTING
     PETER SPENCER"
6010 INPUT"HOW MANY COLUMNS";CL%
6020 S2$="":EC(0)=-1
6030 FORI=1TOCL%
6040 PRINT"COLUMN";I;:INPUT"TO START
     AT SPACE";SC(I)
6050 INPUT"AND END AT SPACE";EC(I)
6055 IFSC(I)<=(EC(I-1)+1)THEN PRINT
   "NOT POSSIBLE**TRY AGAIN":GOTO6040
6060 NEXTI
6070 W$="":K=1
6090 FORJ=1TOCL%
6100 IFJ=1ANDSC(1)=1THEN6150
6110 FORI=KTO(SC(J)-1)
6120 W$=W$+" ":K=K+1:IFK>80THEN6195
6140 NEXTI
6150 FORL=SC(J)TOEC(J)
6160 W$=W$+"A":K=K+1:IFK>80THEN6195
6180 NEXTL
6190 NEXTJ
6195 PRINT"COLUMN FORMAT IS"
6200 PRINTW$
6210 INPUT"OK";S2$
6220 IFS2$<>"Y"THEN6030
6230 OPEN3,4,2:PRINT#3,W$:CLOSE3
6240 RETURN
READY.
```

Figure 2: Waterloo Basic Coding

```
10 CALL DYNAMIC FORMATTER
20 STOP
6000 PROC DYNAMIC FORMATTER
6010 PRINT"DYNAMIC FORMATTING  PETER SPENCER"
6020 INPUT"HOW MANY COLUMNS";CL%
6030 S2$="":EC(0)=-1
6040 WHILE S2$<>"Y"
6050   I=1
6060   WHILE I<=CL%
6070     PRINT"COLUMN";I;:INPUT"TO START AT SPACE";SC(I)
6080     INPUT"AND END AT SPACE";EC(I)
6090     IF SC(I)<=(EC(I-1)+1)
6100        PRINT"NOT POSSIBLE**TRY AGAIN"
6110     ELSE
6120        I=I+1
6130     ENDIF
6140   ENDLOOP
6150   W$=""
6160   I=1
6170   K=1
6180   LOOP
6190      WHILE (K<SC(I) AND K>EC(I-1))
6200         W$=W$+" "
6210         K=K+1
6220         IF K>80 THEN QUIT
6230      ENDLOOP
6240      WHILE K>=SC(I)
6250         W$=W$+"A"
6260         K=K+1
6270         IF K>80 THEN QUIT
6280      UNTIL K>EC(I)
6290      I=I+1
6300      IF I>CL% THEN QUIT
6310   ENDLOOP
6315   PRINT"COLUMN FORMAT IS"
6320   PRINT W$
6330   INPUT"OK (Y/N)";S2$
6340 ENDLOOP
6350 OPEN3,4,2
6360 PRINT#3,W$
6370 CLOSE3
6380 ENDPROC
READY.
```

©

ΩMEGA
SALES
CO.

"WHOLESALE COMPUTER PRICES" DIRECT TO THE PUBLIC

12 Meeting St., Cumberland, R.I. 02864

TRS•80
Model II - $3,500

Apple II
16K - $1049

INTERTEC SUPERBRAIN
32K RAM - $2449.00
64K RAM - $2649.00

PRODUCT SPECIAL OF THE MONTH!!

Atari 800
$749.00

Products are NOW IN STOCK AT ΩMEGA Sales Co.

For TRS•80 Apple CBM (Interface Included)

Epson TX-80 - $745

NEC Spinwriter
5510-5530 - $2449

Soroc 120 - $699

CALL TOLL FREE FOR ΩMEGA'S PRICE!

ΩMEGA OFFERS THE BEST DELIVERY AND PRICE ON:
APPLE • ATARI • TRS•80 MODEL II • INTERTEC •
T.I. 810 • HEWLETT-PACKARD-85 • SOROC •
COMMODORE • NEC • QUME • CENTRONICS

ΩMEGA sells only factory fresh, top quality merchandise to our customers.
ΩMEGA will try to match any current advertised price with similar purchase conditions.
Before you buy anywhere else - be sure to call ΩMEGA Sales Co.

ΩMEGA TOLL FREE

1-401-722-1027 or

1-800-556-7587

ΩMEGA ships via UPS, truck, or air. COD's, VISA, Mastercharge accepted.
"A member in good standing of the better business bureau."

VISA master charge

TelePET

Jim Butterfield, Toronto

This is the age of computers talking to other computers. There's no reason why your PET can't join in the conversation, too. New communications interfaces for the PET are being announced fairly often these days. What's involved in the hookup?

Most commercial offerings give you the whole package to enable you to hook up and be "on the air" fairly quickly. But since their technical approaches are different, it's worth while to look at what a communications interface needs to do.

Interface elements

There are several problems that need to be addressed in order to hook your PET to a telephone line. Starting at the telephone end, they are:

1. The telephone company gets annoyed if you wire things directly to the telephone line, unless they are "approved" devices. The small user should also worry about the dangers to his PET: some hefty voltages can come from the telephone exchange.

The easiest solution to this is an acoustic coupler. You fit your telephone handset into one of these, and it arranges to make noises into the transmitter and to listen to the earpiece with a microphone. No electrical connection - sound power does the whole job.

2. The telephone system was designed to carry voice, or sounds in a certain frequency range. The PET signal needs to be changed to an audible signal in order to be transmitted; at the other end, the sound frequencies need to be changed back into bits - the ones and zero that the PET needs.

This problem is solved by a device called a Modem. A Modem consists of two parts: a modulator, which changes bits to tone frequencies for sending; and a demodulator, which changes the tones back to bits.

3. You can normally send and/or receive only one bit at a time. PET handles eight bits at a time. Something has to take the eight bits from the PET (the "parallel" signal, since eight bits come out together) and fire them off one bit at a time (creating a "serial" signal, with one bit after the other). In the other direction, you must collect the eight bits, one at a time, pack them together and deliver them to the PET as a parallel eight-bit byte.

Tied into this problem of parallel-to-serial conversion is a related job. Much of the time PET will have nothing to send. We must distinguish between an idle connection, where nothing is being sent, and an active connection which has a character under way.

This last task is usually effected by a signal called a *start bit*. The start bit is sent before the PET's information bits; it says, "here comes a character". If you don't use a start bit, you know that the line is idle.

All of the above tasks can be performed in machine-language programs, or in a rather clever chip called a UART. Either way, you must arrange to send a start bit, then the eight data bits, one at a time, and then a brief pause (sometimes called a *stop bit*) before you start the next character. Coming the other way, the receiving PET must wait for a start bit and then collect the eight data bits into a single byte.

4. If you're communicating with a non-PET at the distant end, the other computer will probably want to receive a standard code called ASCII, and will send that code back to you. PET does not store characters in ASCII format, so that a little translation will be needed in both directions.

PET has characters that don't exist in ASCII. For example, most of the PET graphic characters don't have any corresponding ASCII characters. You'll have to give them up.

There are a few ASCII characters that don't have any counterpart in the PET. Most of these are called *control* characters. You'll probably need a few of these for a good communications interface. Most commercial packages make them available with a two-key combination from the PET. For example, the keys Reverse, semicolon often generate the character known as ESC or Escape in ASCII; this character usually tells the distant computer to stop whatever it's doing and wait for a new command from you. It's a very handy character to know when the distant computer has started to send out a massive amount of data which you realize you really don't want.

5. The physical connection at the PET is either the IEEE-488 bus or the Parallel User Port. If it's the IEEE-488 bus, the connected device will have to obey the protocols - recognizing when it's selected, receiving and delivering characters to the bus, etc.

If it's the Parallel User Port, PET will need to contain a machine language program which is called by the user's program any time it is desired to receive or send.

The IEEE-488 bus is simple to use - a normal PRINT# command will send data - but since the bus is shared with other devices, careful design is needed.

Tracing the Flow

Let's put the above together and track a character from the PET to the line, and vice-versa.

1. PET decides to send a character. If the interface is via the IEEE bus, PET might simply issue the command PRINT#7, "A"; or if the interface is via the parallel user port, the program might say, SYS 30456,"A" There are many possible variations.
2. The character- in this case, the letter A which is represented in PET text mode as hexadecimal C1 -must be translated to true ASCII. This might be

Skyles Electric Works

"If you could own only one peripheral for your PET, it should be this. It opens the whole world to your PET."

The Cat: Switches for mode selection and operation. LEDs display unit status. Acoustic self-test is standard. Compact powerpack plugs directly into wall socket.

Now $325.00* complete with membership in The Source*, Skyles six-foot cable/interface to the PET user port, together with Skyles cassette program in machine language and in BASIC.
(If bought separately: $180.00, modem; $80.00, cable/interface and program; $100.00, membership in The Source)*

The Cat and D-Cat have been specially prepared by Skyles for interfacing to the PET user port (*not* to the IEEE port) and with a special cassette program, allowing communication

...from modem to disk and disk to modem
...from modem to terminal; read on screen, save on disk
...from disk to printer through IEEE
...from disk to screen

The D-Cat: FCC-approved for handset jack connection with any modular phone, either single or multi-line. No need for adapters. Can fit under phone; installs in seconds. Mode switch to monitor voice or data transmission. Special "hold" function; complete self-test. Power pack plugs directly into wall socket.

Now $350.00* complete with membership in The Source*, Skyles six-foot cable/interface to the user port and Skyles cassette program in machine language and in BASIC.

(If bought separately: $210.00, the modem; $80.00, cable/interface and program; $100.00, membership in The Source)*

*What About the Source?

Sometimes called "The Information Utility," it's a telecomputing network that gives you, through your Cat or D-Cat modem, thousands of programs and data bases and allows you to communicate with other users interactively and through electronic mail.

**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

done in either program or in hardware; in either case, the result is hexadecimal 41.

3. The parallel to serial translation now takes place. Once again, this may be done within a program or by hardware (a UART chip). A start bit is generated followed by the eight bits of data; each is sent at the appropriate time.

4. Each bit, as it is generated, is translated by the modem into an appropriate tone frequency. One frequency represents a zero bit, another represents a one bit.

5. The tones generated by the modem are fed into a small speaker which is very close to the telephone handset transmitter. The sound from the speaker is picked up by the telephone and sent to the line. It's on the way...

At the receiving end:

6. The telephone earpiece has been making a whining sound from the tone received from the line. The sound is picked up by a small microphone close to the earpiece.

7. The signal reaches the modem which examines the tone and classifies it as either a logic zero or a logic one. It passes along the logic state - zero or one - to the serial to parallel translator.

8. The serial to parallel translator waits patiently for a start bit (logic zero) to be received. When it sees this, it carefully collects the eight data bits at the apropriate times. This might be done either in a program or in hardware (again, with a UART).

9. The eight-bit character might be placed into a buffer or might just be held for pickup by the PET. In either case, the received character will need to be translated from ASCII into PET format.

The Modem/Acoustic Coupler

The modem and acoustic coupler are invariably packaged together. Speeds up to 30 characters per second are generally available; lower speeds will work, but the highest rate of 30 cps is a virtual standard now.

The Commodore interface packages everything into the modem/coupler case: IEEE bus interface, UART, the whole thing. Other suppliers use standard commercially available modem/couplers and supply extra hardware and/or programs to complete the interface.

The commercially available modems use an interface known as RS-232. It's nice to have this interface available, since you cannect other things besides modems to it. Various types of terminals, both video or hard copy, will hook up with no problems.

Parallel/Serial interfaces and Buffering

It's economical and flexible to use a program to do your parallel/serial interface, and buffering can be provided quite easily. It does take up memory space, however, and it can keep the PET rather busy: bits move in and out at a rate of one every three milliseconds or so. Your interface from Basic will be rather more tricky, too: PRINT# or GET# won't make the connection too easily.

Hardware costs more, but helps with some of these problems. You may not be liberated from the need for special programs, though. The mighty UART chip can only catch or send one character at a time. Unless you have buffering, PET will have to wait before the next character can be sent or received.

The GPIB bus

The IEEE-488 bus is ideal for sending or receiving characters from Basic. As always, however, there's a catch or two. If the device you're sending to is busy and can't catch the character you want to send it, it will probably hang up the bus so that everything stops until it's ready. The same thing may happen if you try to INPUT or GET a character or value that hasn't arrived yet; you'll either time out or wait.

This isn't new. Many devices hold up the IEEE bus - the printer and the disk do it, for example. But with a communications interface, waiting time becomes a serious problem. You might lose a character if the bus is hung up waiting for something else to happen. It becomes more important to use the bus in a more sophisticated way.

Looking them Over

All of the above problems have been solved in a variety of ways by the various suppliers. A remarkable amount of ingenuity has been called into play, and the user has considerable choice.

Check out the units available to see which ones fit your style. How much of the package is hardware, and how much software? How easily can you interface with your own Basic programs? Can you attach devices other than a modem? Does the unit contain buffering? How is the translation to and from ASCII accomplished? Can you abandon ASCII if you choose and send directly from PET to PET, for graphics or program transfer? How much memory will you need in the PET? Will you need disk? And, of course, how much money will it all cost?

There's no single answer. Find out what suits you.

Communications interfaces are here. You'll see more of them used in the PET community. One of these days, you'll be tempted to join the network. ©

Word Pro Converter

Robert W. Baker, BAKER ENTERPRISES,
15 Windsor Drive, Atoo, NJ 08004

An ever increasing number of programs make use of Commodore's Word Pro program with its excellent editing facilities to generate files for their own use. However, disk files created by Word Pro 3 are not fully compatible with those created or used by Word pro 4 on the 8016/8032.

If you create any files on a 2001 series PET/CBM using Word Pro 3, you will have to do some editing to be able to use the same file on an 8016/8032 CBM with its 80 column screen. This simple utility program will eliminate the boring task of editing the file, and do all the necessary changes for you automatically. It will run on either a 2001 PET/CBM or an 8016/8032 CBM; using a 2040 disk. Remember, though, that the 2040 disk must have the DOS 2.0 ROMs if you are using an 8016/8032 CBM.

The Word Pro Converter program is very straight forward in operation and no fancy frills or options are included. The file to be converted must be on the diskette in Drive #0. The new file created will be written on the diskette on Drive #1 with the same name. If the file all ready exists on Drive #1, it will be deleted first. The only input to the program is the name of the file to be converted. It should be very simple to add an output file name option along with drive number selections if desired. During program execution, any disk error will be displayed and terminate the program.

In theory, the program simply copies the file byte-by-byte while counting characters and looking for a RETURN within each original 40 character line. Straight text that continues over several 40 character lines is copied as-is, creating new 80 character lines. If a RETURN is detected in any line, an extra 40 spaces are added at the end of the line whenever required to make the line 80 characters long.

Files stored by Word Pro 3 contain 40 characters per display line regardless of content. Thus, if you have a single FP command on a line, there is a 37 byte overhead with Word Pro 3. Word pro 4, on the other hand, stores 80 characters per display line regardless of content. The same FP command in Word Pro 4 will then have a 77 byte overhead! While Word Pro 4 has its advantages with the 80 column screen, the disk files created will be generally bigger than those created by Word Pro 3 for the same text. This is especially true when there are a large number of formatting commands or blank lines.

Program Variables

E	input file status, 64 = end-of-file
N	#characters in input file line, 40 max
P	#characters in output file line, 80 max.
R	RETURN character flag: 0 = no 1 = yes
B$	character (byte) being copied

```
100 REM ******************************
110 REM
120 REM    SIMPLE UTILITY PROGRAM
130 REM     TO CONVERT DISK FILES
140 REM    CREATED BY WORD PRO III
150 REM  FOR LOADING BY WORD PRO IV
160 REM
170 REM ------------------------------
180 REM
190 REM       BY: ROBERT W. BAKER
200 REM
210 REM        BAKER ENTERPRISES
220 REM 15 WINDSOR DR., ATCO, NJ 08004
230 REM
240 REM ******************************
250 :
260 :
270 PRINT"ħ  W O R D   P R O   C O N V E ¬
    ¬R T E R
280 PRINT"↓↓THE FILE TO BE CONVERTED MUST ¬
    ¬BE ON
290 PRINT"ON THE DISKETTE IN DRIVE #0↓
300 PRINT"THE NEW FILE GENERATED WILL BE ¬
    ¬WRITTEN
310 PRINT"ON THE DISKETTE IN DRIVE #1,
320 PRINT"WITH THE SAME FILE NAME.↓↓
330 INPUT"FILE NAME    .←←←";FI$
340 IF FI$="." THEN 330
350 PRINT"↓rCONVERTING FILE, PLEASE ¬
    ¬WAIT...
360 OPEN 15,8,15
370 OPEN 5,8,8,"0:"+FI$+",P,R"
380 GOSUB 560
390 PRINT#15,"S1:"+FI$
400 OPEN 6,8,9,"1:"+FI$+",P,W"
410 GOSUB 560
420 GET#5,A$,B$:GOSUB 560
430 PRINT#6,A$;B$;:GOSUB 560
440 P=0
450 N=0:R=0
460 GET#5,B$:E=ST:GOSUB 560
470 PRINT#6,B$;:GOSUB 560
480 P=P+1:IF P=80 THEN P=0
490 IF E=64 THEN PRINT"↓rDONE !!!↓":
    ¬GOTO 610
500 N=N+1
510 IF ASC(B$)=31 THEN R=1
520 IF N<40 THEN 460
530 IF R=0 OR P=0 THEN 450
540 FOR N=1 TO 40: PRINT#6," ";GOSUB 560:
    ¬NEXT
550 GOTO 440
560 INPUT#15,EN,EM$,ET$,ES$
570 IF EN=0 THEN RETURN
580 PRINT"↓rDISK ERROR !!!↓
590 PRINT EN;EM$,ET$,ES$
600 PRINT"↓OPERATION ABORTED!
610 CLOSE 5: CLOSE 6: CLOSE 15
READY.
```

©

Multitasking On Your PET? QUADRA-PET

Charles Brannon

QUADRA-PET is a machine language program that lets you partition the memory of an upgrade ROM PET or CBM into four 8K blocks. Each block is an independent program workspace. Programs existing in each 8K partition can be selected and then used and modified without affecting any of the other programs. You can jump to any other of the programs at any time.

After initialization with SYS 926, PET displays the question:

WHICH PET? [1-4]

Perhaps Mary, an avid computer-games buff, types in "1" and loads STARTREK. She plays it for a while and then leaves to eat lunch. Meanwhile, Bob goes to the PET, sees that someone is using PET #1, and switches to PET #2 to write a business program. After nearly "perfecting" it, he leaves to see what Mary is up to. Now the kids come in, and after arguing for a half-hour agree to share the PET, one using PET #3 and the other PET #4. Fortunately for Bob and Mary, nothing the kids do can harm their programs.

How To Use QUADRA-PET

1. Load or type in one of the versions of QUADRA-PET. (Basic or hex)
2. Enter **NEW**
3. **SYS 926** to initialize.
4. PET will respond with **WHICH PET? (1-4)**
5. Select the one you wish to use.
6. Before loading or typing in a program for the first time, type in **NEW.**
7. To select another PET, **SYS 826** and follow instructions 4-7.

Now comes the fun part -- how does it work? Many memory locations in zero-page (0-256) are *pointers* QUADRA-PET works with three of those pointers.

On power-up, PET determines the end of memory by writing a character to every memory location and then reading it back. PET then increments a memory location until a failure in reading that character occurs. This indicates that the end of available memory has been reached. Physically, this pointer is at location 52 decimal. ($34). The second pointer is at the start of memory, stored in location 41. Originally, this points to the actual start of user memory, 1024. The last pointer is the end of text pointer. As you write your program it changes.

QUADRA-PET partitions the memory by changing these pointers to point to successively higher memory locations, depending on which PET is in use. Since the end of text pointer changes, it must be saved before we move to a new PET and restored on return. QUADRA-PET, as it is in machine language, does all these things seemingly instantaneously.

HOW TO SAVE A PROGRAM PRODUCED WITH QUADRA-PET:

1. SYS 1024 to go to the Monitor.
2. Enter: .M 0028 002B and type RETURN.
3. You will get a display something like:
   .: 0028 01 04 3E 04 .. ..
4. We will use only the first four bytes. Write down the first pair in reverse order on paper, for example: 0401
   Do the same with the second pair. (e.g. 043E)
5. Enter: .S "PROG NAME",01,XXXX,YYYY where "PROG NAME" is the name of your program, XXXX is the first number you wrote down, and YYYY is the second. For example, to save the example program which we will call "PET #1, you would enter: .S "PET #1", 01,0401,043E
6. Press RETURN and press play and record to save your program.
7. To load *this* saved program into a space prepared by QUADRA-PET, just SYS 1024 and enter .L "PROG NAME" where "PROG NAME" is the name of your program.

HOW TO LOAD A PRE-EXISTING PROGRAM INTO A SPACE PREPARED BY QUADRA-PET:

I could tell you how to do this on the old ROM PET but quite frankly, I can't find the memory locations for this procedure in the new PET. All you PET experts -- HELP!
If you can figure it out, please send in the procedure to **COMPUTE.**

A little imagination will create many uses for QUADRA-PET.

For education, it is the perfect way to keep four students' programs in the PET at the same time. Each program can be worked on and modified in any way without affecting any other of the programs.

In business, four different business programs can exist simultaneously in PET's memory, ready to use. For the small penalty of loading the programs into the program workspaces at the start of the day, all four are within reach of a carriage return -- faster than any disk drive.

Machine language programmers can fill partitions with useful routines, leaving one or more partitions for BASIC. QUADRA-PET itself is short and easily relocatable.

I would be interested to find out what novel and useful applications for QUADRA-PET *you* can think up!

Happy QUADRA-PETing!

References

CBM User Manual 2001-32, First Edition. Commodore Business Machines, Inc., Palo Alto, CA (1979)

Havery B. Herman, "Memory Partition of BASIC Workspace", **COMPUTE**, pp. 18-20 (Jan., Feb. 1980)

Jim Butterfield, "PET in Transition (memory map) **COMPUTE**, pp. 68-70 (Fall, 1979)

```
 0 REM*********************************
 1 REM             QUADRA PET
 2 REM*********************************
 3 REM:BY CHARLES BRANNON 06/07/80
 10 FOR I =  826 TO 941
 20 READ A
 30 POKE I, A
 40 NEXT
 50 SYS926
 60 END
 1000 DATA174,126,3,165,42,157,131,3,165
 1010 DATA43,157,135,3,169,143,160,3,32
 1020 DATA28,202,32,228,255,41,15,240,249
 1030 DATA201,5,176,245,170,202,142,126,3
 1040 DATA169,1,133,40,189,127,3,133,41
 1050 DATA189,131,3,133,42,189,135,3,133
 1060 DATA43,169,0,133,52,189,139,3,133
 1070 DATA53,32,119,197,96,0,4,32,64
 1080 DATA96,3,3,3,3,4,2,64,96
 1090 DATA32,64,96,128,87,72,73,67,72
 1100 DATA32,80,69,84,63,32,40,49,45
 1110 DATA52,41,0,169,0,141,0,32,141
 1120 DATA0,64,141,0,96,76,58,3
READY.
```

©

OOPS! A Crucial Update to Disk ID Changer, Issue #5, COMPUTE

Rene W. Poirier

DISK ID CHANGER was intended to change the ID on diskettes to prevent having diskettes with duplicate ID characters. Information has surfaced to the effect that ID Changer does not accomplish its goal. Though it successfully changes the characters on Track 18 Sector O, those characters are purely cosmetic and for display in the directory only. The actual ID characters are deeply imprinted on each of the sectors of the diskette.

The bulletin mentioned in the previous article did not specify the DOS to which it was referring. It now appears that the actual reference was to the new DOS which auto-initializes a diskette when it recognizes a change in the sector ID characters. In this case, swapping diskettes with identical ID characters will fool the new DOS and it will not auto-initialize and create a new Block Availability Map.

This can cause *real problems.* DISK ID CHANGER, though intended to prevent this, fails to do so. In fact, it can compound the problem, since the true ID is lost to the user.

Thanks to Jim Butterfield for bringing this error to my attention and directing me onto a course toward a solution. The program, ID CORRECTOR checks the diskette for the actual, or original, ID characters inprinted on the sectors, and compares those characters to the cosmetic characters on Track 18 Sector 0. If they match properly, it so informs you. If they do not, it can replace the erroneous ID with the actual characters, returning the changed diskette to its original configuration. If use of the ID CORRECTOR on drive 0 is desired, change the variable DV in line 7.

This will not solve the original problem of more that one disk having identical original ID characters, such as backup diskettes. Care will have to be taken when using these diskettes. It would seem advisable to include forced initialization commands in programs to force creation of a new BAM when diskettes are changed. The real solution to the problems would lie in a command to duplicate, but with a change in ID characters on the copy diskette, but alas....

The portion of DISK ID CHANGER for changing the name on a diskette for library naming purposes is valid and safe. To prevent accidental use of the portion which changes ID characters, I have included another version called DISK NAME CHANGER which will alter only the ID NAME of a diskette. It appeared easier to build a new program than to try to extract the appropriate sections from DISK ID CHANGER.

```
1 REM *** DISK ID CORRECTOR
2 REM *** BY RENE W POIRIER
3 REM *** BERLIN, N.H.
4 :
5 REM *** WITH THE HELP OF JIM BUTTERFIELD
      ¬ ***
6 :
7 DV=1:REM SET DRIVE NUMBER (1/Ø)
8 :
9 :
1Ø OPEN9,Ø,Ø:OPEN15,8,15
2Ø PK=PEEK(59468):POKE59468,12
3Ø MD$="h":FORI=1TO2Ø:MD$=MD$+"↓":NEXT
4Ø FORI=1TO39:BL$=BL$+" ":NEXT
5Ø PØ$="ĥ↓↓             ID CHECKER/CORRECTOR
6Ø RE$="PRESS rRETURNr̂ TO CONTINUE
99 GOTO1ØØØ
1ØØ INPUT#15,ER:IFER=ØTHENRETURN
11Ø INPUT#15,ER,EM$,ET$,ES$
12Ø PRINTMD$"rDISK ERROR! r̂#"ER" "EM$"  ¬
      ¬"ET$", "ES$
13Ø END
2ØØ INPUT#9,Q$:PRINT:Q1$=LEFT$(Q$,1):
      ¬RETURN
3ØØ CLOSE15:POKE59468,PK:PRINT"ĥ":END
1ØØØ PRINTPØ$:PRINT"↓PLACE DISKETTE TO BE ¬
      ¬CHECKED IN DRIVE" DV "↓↓"
1Ø1Ø PRINTRE$:GOSUB2ØØ
1Ø2Ø AD$="":ID$=""
1Ø3Ø PRINT#15,"I"+STR$(DV):GOSUB1ØØ
1Ø4Ø OPEN2,8,2,"#Ø":GOSUB1ØØ
1Ø5Ø PRINT#15,"U1:2";DV;",18,Ø":GOSUB1ØØ
1Ø6Ø FORJ=33TO34
1Ø7Ø PRINT#15,"M-R";CHR$(J);CHR$(16):
      ¬GET#15,Z$
1Ø8Ø AD$=AD$+Z$:NEXTJ
1Ø9Ø PRINT#15,"B-P:2,162":GET#2,A$,A1$:
      ¬ID$=A$+A1$
11ØØ PRINT"↓↓ACTUAL ID RECORDED ON ¬
      ¬SECTORS IS: "AD$
111Ø PRINT"↓FILE ID IS: "ID$
112Ø IFAD$<>ID$THEN12ØØ
113Ø PRINT"↓→→→→→→→→rTHIS DISK IS OK!
114Ø CLOSE2
115Ø PRINTMD$BL$:PRINTBL$MD$"DO YOU WISH ¬
      ¬TO CHECK "
116Ø PRINT"ANOTHER DISKETTE? (Y/N) r";:
      ¬GOSUB2ØØ
117Ø IFQ1$="Y"THEN1ØØØ
118Ø IFQ1$<>"N"THEN115Ø
```

```
1190 GOTO300
1200 PRINT"↓↓ACTUAL ID AND FILE ID DO NOT ¬
     ¬MATCH!
1210 PRINTMD$"SHALL I PROCEED TO CORRECT? ¬
     ¬(Y/N) ɼ";:GOSUB200
1220 IFQ1$<>"Y"THEN1140
1230 PRINTMD$BL$MD$"BE PATIENT...
1240 PRINT#15,"B-P:2,162":GOSUB100
1250 PRINT#2,AD$;:GOSUB100
1260 PRINT#15,"U2:2";DV;",18,0":GOSUB100
1270 PRINT#15,"I"+STR$(DV):CLOSE2
1280 PRINTMD$BL$"ĥ"MD$"CHECKING DISK
1290 PRINTLEFT$(MD$,8);
1300 GOTO1020
```

```
1 REM *** DISK NAME CHANGER
2 REM *** BY RENE W. POIRIER
3 REM *** BERLIN, N.H. 03570
4 :
5 :
6 :
7 :
10 OPEN9,0,0:OPEN15,8,15
20 PK=PEEK(59468):POKE59468,12
30 MD$="ħ":FORI=1TO20:MD$=MD$+"↓":NEXT
40 FORI=1TO39:BL$=BL$+" ":NEXT
50 P0$="ĥ↓↓          DISK NAME CHANGER
60 RE$="PRESS ɼRETURNȓ TO CONTINUE
70 DATA 2,OUT OF RANGE,TOO LONG
80 READ A:DIM EM$(A):FORI=1TOA:READEM$(I):
     ¬NEXT
99 GOTO1000
100 INPUT#15,ER:IFER=0THENRETURN
110 INPUT#15,ER,EM$,ET$,ES$
120 PRINTMD$"ɼDISK ERROR!ȓ #"ER"  "EM$"   ¬
     ¬"ET$", "ES$
130 END
200 INPUT#9,Q$:PRINT:Q1$=LEFT$(Q$,1):
     ¬RETURN
300 CLOSE2:CLOSE15:POKE59468,PK:PRINT"ĥ":
     ¬END
400 PRINTLEFT$(MD$,MD)BL$:PRINTBL$
410 PRINTLEFT$(MD$,MD);:RETURN
500 MD=21:GOSUB400
510 PRINT"ɼUNACCEPTABLE ENTRY -- "EM$(EM)
520 PRINTRE$:GOSUB200:GOSUB400
530 RETURN
1000 F=0:PRINTP0$
1005 PRINT"↓ON WHICH DRIVE SHALL WE ¬
     ¬PERFORM
1010 PRINT"THE CHANGES? (0/1) ɼ";:GOSUB200
1020 DV=VAL(Q$):IFDV<0ORDV>1THENEM=1:
     ¬GOSUB500:GOTO1000
1030 IFDV=0ANDQ1$<>"0"THENEM=1:GOSUB500:
     ¬GOTO1000
1040 PRINTP0$:PRINT"↓PLACE DISKETTE IN ¬
     ¬DRIVE"DV"↓"
1050 PRINTRE$:GOSUB200
1060 PRINT#15,"I"+STR$(DV):GOSUB100
1070 OPEN2,8,2,"#":PRINT#15,"U1:2";DV;",
     ¬18,0":GOSUB100
1080 PRINT#15,"B-P:2,144":GOSUB100:DN$=""
1090 FORI=1TO16:GET#2,A$:DN$=DN$+A$:NEXT
1100 MD=10:GOSUB400:IF F THENRETURN
1110 PRINT"THE PRESENT DISK NAME IS:
1120 PRINTTAB(5)CHR$(34)DN$CHR$(34)
1130 MD=13:GOSUB400
1140 PRINT"DO YOU WISH TO CHANGE IT? ¬
     ¬(Y/N) ɼ";:GOSUB200
1150 IFQ1$="N"THEN1350
1160 IFQ1$<>"Y"THEN1130
1170 MD=16:GOSUB400
1180 PRINT"ENTER NEW DISK NAME":PRINT"LIMI
     ¬T TO 16 CHARACTERS ɼ"
1190 GOSUB200:IFLEN(Q$)>16THENEM=2:
     ¬GOSUB500:MD=18:GOSUB400:GOTO1190
1200 NDN$=LEFT$(Q$+BL$,16)
1210 MD=21:GOSUB400
1220 PRINT"SHALL I SEND ɼ"NDN$
1230 PRINT"TO THE DISKETTE ON DRIVE"DV"? ¬
     ¬(Y/N) ɼ";:GOSUB200
1240 IFQ1$="Y"THEN1270
1250 IFQ1$="N"THENGOSUB400:MD=18:GOSUB400:
     ¬GOTO1170
1260 GOTO1210
1270 MD=21:GOSUB400:PRINT"BE PATIENT...
1280 PRINT#15,"B-P:2,144":GOSUB100
1290 PRINT#2,NDN$;:GOSUB100
1300 PRINT#15,"U2:2";DV;",18,0":GOSUB100
1310 PRINT#15,"I"+STR$(DV):GOSUB100:CLOSE2
1320 F=1:PRINTP0$:GOSUB1060:F=0
1330 PRINT"THE NEW DISK ID IS:
1340 PRINTCHR$(34)DN$CHR$(34)
1350 CLOSE2:MD=21:GOSUB400:PRINT"DO YOU ¬
     ¬WISH TO DO
1360 PRINT"ANOTHER DISKETTE? (Y/N) ȓ";:
     ¬GOSUB200
1370 IFQ1$="Y"THEN1040
1380 IFQ1$="N"THEN300
1390 GOTO1350
```

©

Variable-Field-Length Random Access Files On The 2040 Disk Drive

Peter Spencer,
7 Brightside Drive,
West Hill, Ontario,
Canada M1E 3Y8

Do you have voluminous file storage needs, but hate to see a large fraction of each disk eaten up by the empty space that seems to be an inherent feature of most random access programs?

This program shows how to write variable field length random access files on the 2040 disk drive. The density of packing is truly amazing. Compare it to the density achieved by any fixed field length program you have, including the lengthy relative record program in the 2040 User's Manual.

The writing to disk is done in lines 41 to 77, and the retrieval from disk is in lines 82 to 106. The rest of the program is a driver routine patched on from a longer program of mine.

For this sample program, I have used the line number as the key for each field. You can easily use some other key, and have more than one field per key. In that case, you must change the output to the key file (lines 71-77) so that it contains the number of keys used, each key, the number of fields for that key (if variable), and the track, sector, and buffer pointers for each field within that key. Lines 88-95 would have to be similarly changed.

Yes, you read the above correctly, you can even have a variable number of fields per key! Such a variable field number, variable field length program can be of considerable use if you want to store abstracts, test questions, criterion-referenced test questions (using the criterion or instructional objective code "number" as the key), or parts inventory (you could use the machine name as the key, and each part as a field, with subfields for cost, price, onhand, backordered, and so forth).

The driver routine I have used can be considerably shorter if you wish to use regular input rather than the bullet-proof and hyphenation-proof form provided in lines 118-133. There, a line-overrun on input from the keyboard (detected in line 125) results in the entire word being removed to the next consecutive line (accomplished in lines 128-133 and 119).

```
1 CLR
2 PRINT"ĥɼVARIABLE FIELD LENGTH FILES ON ¬
      ¬THE 2040ȓ           ↓PETER SPENCER"
3 GOSUB108:MK=0:LL=80
4 DIMPA(300):DIMTA(300):DIMSA(300)
5 NLS=1:D=0:F=0:X=0:Y=0:T=0
6 SP$="                                        ¬
      ¬    "
7 M$=CHR$(13)
8 S$="":Z$="":IN$="":DIMA$(300):OPEN15,8,
      ¬15
9 REM:  PROGRAM ENTRY
10 PRINT"ĥɼSȓTART NEW FILE, OR ɼWȓORK ON ¬
      ¬OLD FILE? ";
11 GOSUB33
12 PRINT"NAME OF FILE ";:GOSUB119:
      ¬A$(1)=IN$
13 IFS$="S"THENGOTO22
14 GOTO83
15 REM:  SHOW FILE ENTRIES
16 FORK=1TONLSSTEP15:F=K:D=K+14
17 FORI=FTOD:PRINTI;TAB(6);A$(I):NEXTI
18 PRINT"h";SP$;SP$;SP$
19 PRINT"hɼSȓCROLL NEXT 15 LINES, OR ¬
      ¬ɼEȓXIT? ";:GOSUB33:IFS$="E"THENK=NLS
20 PRINT"ĥ↓↓↓";:NEXTK
21 REM:  SHOW MENU
22 PRINT"ĥ";SP$;SP$;SP$
23 PRINT"hɼRȓEAD IN, ɼOȓUTPUT, ɼTȓYPE, ";
24 PRINT"ɼSȓCROLL, ";
25 PRINT"ɼEȓXIT?";:GOSUB33
26 IFS$="E"THEN79
27 IFS$="T"GOTO110
28 IFS$="O"GOTO42
29 IFS$="R"GOTO83
30 IFS$="S"THENPRINT"ĥ↓↓↓";:GOTO16
31 GOTO22
32 REM:  GET UTILITY
33 GETS$:IFS$=""THEN33
34 PRINTS$:RETURN
35 REM:  READ ERROR CHANNEL
36 INPUT#15,EN$,EM$,ET$,ES$
37 IFEN$="00"THENRETURN
38 PRINT"ERROR ON DISK"
39 PRINTEM$;EN$,ET$,ES$
40 CLOSE6:CLOSE7:CLOSE15:END
41 REM:  OUTPUT ROUTINE
42 IFMK<>0THEN46
43 PRINT"INSERT DISK IN LEFT DRIVE & TYPE ¬
      ¬GO";:GOSUB33
44 PRINT#15,"I1"
45 OPEN6,8,6,"#":GOSUB35
46 PRINT"THERE ARE";NLS;"ENTRIES":MK=1
47 PRINT"STORE FROM ";:GOSUB119:X=VAL(IN$)
      ¬:PRINT"TO ";
48 GOSUB119:Y=VAL(IN$)
49 I=X
50 REM:  ALLOCATE 1 BLOCK
51 T=1:S=0
52 PRINT#15,"B-A";1;T;S
53 INPUT#15,EN$,EM$,ET$,ES$
54 IFEN$="00"THEN57
55 IFEN$="65"THENT=VAL(ET$):S=VAL(ES$):
      ¬GOTO52
56 GOTO38
57 BP=1
```

Computer Supplies for Small Computer Systems

Commodore Printer Ribbons

IN STOCK

Our full line of supplies includes:
Dikettes (3M, basf)
Diskette retrieval devices
BPI Business Package for the Commodore
Tractor Printer Paper
Tractor Printer Labels
Printer Ribbons for NEC, Commodore and Xymec
Dust Covers
Adapters
Uncrashers
Computer Desks
NEC Spinwriters
Print Thimbles
Xymec Intelligent Printers
"New Dealer" Kits

Zeigler Electronic Products

DEALERS:
We no longer market COMMODORE Equipment. (Jerry Zeigler is with Commodore) but Connie and I have the computer supplies you need to support Commodore System Sales.

Give us a call:
Trudie Zeigler
or
Connie Rittenberry
(404) 289-1596 or 289-2265
3661 CALUMET RD.
DECATUR, GEORGIA, 33034

MICRO COMPUTER INDUSTRIES, LTD.

INVENTORY CONTROL WITH POINT OF SALE FOR CBM AND PET 32K

DISK VERSION INCLUDES: Storage of 2500 items per diskette, Accounts Receivable, Writes Purchase Orders, Invoices, Summaries, Post Income and will Sort by 10 fields.

$100.00

INVENTORY CONTROL 8K

CASSETTE VERSION INCLUDES: Purchase Order program and Printing functions.

$ 39.00

GENERAL LEDGER

DISK VERSION INCLUDES: The total functions of the Inventory Control programs plus; Accounts Payable, Notes Payable, Purchases, Expenses, it also issues complete Reports, Statements and Summaries.

$350.00

Instruction Manual $10.00 refundable with purchase.

MICRO COMPUTER INDUSTRIES, LTD.

1520 East Mulberry Suite 240
Fort Collins, Colorado 80524
1-303-221-1955

```
58 PRINT#15,"B-P:"6;BP:GOSUB35
59 PRINT#6,A$(I);M$;:GOSUB35:PRINTI;A$(I);
     ¬T;S;BP
60 PA(I)=BP:TA(I)=T:SA(I)=S
61 BP=BP+LEN(A$(I))+1
62 IF(LEN(A$(I+1))+1+BP)>255THEN67
63 I=I+1
64 IFI<=YTHEN58
65 PRINT#15,"U2:"6;1;T;S:GOSUB35
66 CLOSE6:GOTO72
67 PRINT#15,"U2:"6;1;T;S:GOSUB35
68 I=I+1
69 IFI<=YTHEN50
70 CLOSE6
71 REM:  OUTPUT KEY FILE, OVERWRITING OLD ¬
     ¬KEY FILE IF NECESSARY
72 OPEN7,8,7,"@1:"+LEFT$(A$(1)+SP$,
     ¬10)+".KEY01,S,W":GOSUB35
73 PRINT#7,NLS;M$;:GOSUB35
74 FORI=1TONLS
75 PRINT#7,TA(I);M$;SA(I);M$;PA(I);M$;:
     ¬GOSUB35
76 NEXTI
77 CLOSE7:GOSUB35
78 REM:  EXIT PROGRAM
79 PRINT"SHUT DOWN?";:GOSUB33
80 IFS$="N"GOTO22
81 CLOSE6:CLOSE7:CLOSE15:END
82 REM:
83 PRINT"ɽREAD KEYS AND FILE FROM DISK"
84 IFMK<>0THEN87
85 PRINT"INSERT DISK IN LEFT DRIVE & TYPE ¬
     ¬GO";:GOSUB33
86 PRINT#15,"I1":MK=1
87 OPEN7,8,7,"1:"+LEFT$(A$(1)+SP$,
     ¬10)+".KEY01,S,R":GOSUB35
88 INPUT#7,NLS:RS=ST:GOSUB35
89 PRINT"↓NLS=";NLS
90 PRINT" #  TR SE BP"
91 FORI=1TONLS
92 INPUT#7,TA(I),SA(I),PA(I):RS=ST:GOSUB35
93 PRINTI;TA(I);SA(I);PA(I)
94 NEXTI
95 CLOSE7:GOSUB108
96 REM:  READ FILE
97 OPEN6,8,6,"#":GOSUB35
98 FORI=1TONLS
99 PRINT#15,"U1:"6;1;TA(I);SA(I):GOSUB35
100 PRINT#15,"B-P:"6;PA(I)
101 GOSUB35
102 INPUT#6,A$(I):GOSUB35
103 IFTA(I)=0THEN106
104 PRINTI;A$(I)
105 NEXTI
106 CLOSE6:GOSUB108
107 GOTO22
108 FORI=1TO1000:NEXTI:RETURN:REM:
     ¬ DELAY LOOP
109 REM:  TYPE ROUTINE
110 PRINT"LENGTH OF LINE (MAXIMUM=80)";:
     ¬Z9$="80":GOSUB119:LL=VAL(IN$)
111 PRINT"ħɽTYPE NEW LINES";CHR$(13);"(TYP
     ¬E 'STOP' TO STOP))":PRINTSP$
112 D=NLS:IFD>=5THENF=D-4:GOSUB135:GOTO114
113 F=1:GOSUB135
114 PRINTNLS+1;CHR$(13);"↑";TAB(4)
115 GOSUB119:IFIN$="STOP"THEN22
116 A$(NLS+1)=IN$
117 NLS=NLS+1:GOTO111
118 REM:  BULLET-PROOF INPUT
119 IN$="":IFZ9$<>""THENPRINT"? ";Z9$;:
     ¬POKE167,0:IN$=Z9$:Z9$="":GOTO121
120 PRINT"? ";:POKE167,0
121 GETZ$:IFZ$=""THEN121
122 IFZ$=CHR$(13)THENPRINT" ":POKE167,1:
     ¬RETURN
123 IFZ$=CHR$(20)THENONSGN(LEN(IN$))+1GOTO
     ¬121,127
124 PRINTZ$;:IN$=IN$+Z$
125 IFLEN(IN$)>LLTHENGOSUB128:PRINT" ":
     ¬POKE167,1:RETURN
126 GOTO121
127 PRINTZ$;:IN$=MID$(IN$,1,LEN(IN$)-1):
     ¬GOTO121
128 FORZ9=LEN(IN$)TO1STEP-1
129 IFMID$(IN$,Z9,1)<>" "THEN133
130 Z9$=RIGHT$(IN$,LEN(IN$)-Z9)
131 IN$=LEFT$(IN$,Z9-1)
132 Z9=1
133 NEXTZ9:RETURN
134 REM:  SCREEN DISPLAY
135 FORI=FTOD:PRINTI;TAB(6);A$(I):NEXTI:
     ¬RETURN
READY.
```

©

Flexible GET for the Pet

Elizabeth Deal
Malvern, Pa.

This article describes a few ways to achieve a flexible GET routine that includes a flickering cursor, possible use of cursor keys other than left or delete and disabling of the Pet's quote mode, which is useful in several other applications.

The conceptual problem of substituting GET for INPUT has been solved by Pet users, most recently by Mr. Bruey (Compute #3) and Mr. Greenberg (Compute #4). It is a good idea to read those two articles before using GET. It is essential to use input edit routines for any math application as outlined by Mr. Bruey.

The program presented here is a simple one. REM lines describe how to expand it for a more complex use. It uses cursor left to make corrections and it permits all ASCII characters that are printable to be in a string in a non-graphic form. The program does not use the delete key, for it can disturb information already on the screen. Which other cursor keys are permitted to work and which, if any, are permitted to become part of a string depend on application. In some situations it may be desirable to permit the return key and cursor down to become part of a string — for instance in a multi-line input or in input in a tabular form. Those decisions are put around line 360. The instructions in REM lines show how to avoid passing the beginning of a string with the cursor going back or up and how to signal the end of input by a return key at the logical end of the string.

The program uses several pointers that the Pet updates with each PRINT for its own use. These pointers locate the cursor on the screen. The pointers are in locations 196-198 in the new Pet. Line 760 of the program shows an untested conversion for the old Pet. It is by use of the values of these pointers that we keep the cursor within desired limits. Comparison of the starting position, GS, with the current position, GP, can be performed in various ways depending on what sort of input one needs.

The same pointer is used to flicker the cursor during the time the Pet is waiting for input. Lines 390 and 400 show how it is done. (Line 400 shows an additional poke, more about it later). EXCLUSIVE-OR operation on the contents of the position under the cursor with 128 done twice performs the necessary reversals. In case you provide no prompts to the user, a harmless PRINT"" is in this routine to flash the first position before the first character comes in. The advantage of using a method similar to the one Pet uses becomes obvious when one permits cursor controls to enter the strings. In such situation cursor-left, for instance, will ride over the string, flashing each character it encounters.

At this point we are about even with the INPUT statement. GET can do what INPUT can do. We are a bit better off in that we can go up and down with our fake cursor. We will be much better off when we disable Pet's quote mode with POKE205,0. Pet keeps track of an open quote in location 205, and causes several graphic characters to be printed when you want to quote something in reverse or use cursor keys. Pet also makes it very difficult to quote a quote. POKE 205,0 solves these problems. There is one restriction, however. The string must be printed on the screen one character at a time, as in line 290. It also must be input from tape using a GET# statement (see lines 670-700). Printing a string one character at a time solves another string work problem-that of strings containing comma or colon. At this point a string can contain anything, which INPUT cannot match.

POKE 205,0 and printing one character at a time are valuable tools in applications other than GET. For example, one can have a very decent screen image save and print back routine that duplicates every character in the least amount of tape or disk space.

```
170 REM ===INIT FOR GET ROUTINE=======
180 GX=255:G0=0:G1=1:G2=40:G7=128:G8=256:
      ¬G4=196:G5=197:G6=198:GQ=205
190 GH=19:GL=GH+G7:GD=17:GT=GD+G7:GE=29:
      ¬GV=GE+G7:GU=20:GM=GU+G7
200 GR=13:G=0:GA=0:GS=0:GP=0:GF=0
210 REM ===MAIN PRG-ILLUSTRATION======
220 PRINT"TYPE 'XX' TO QUIT OR A STRING + ¬
      ¬RETURN":PRINT:PRINT"FOR INSTANCE"
230 E$=CHR$(34)+"rRVS_IN QUOTES"+CHR$(34)
      ¬+CHR$(44)+" COLON"+CHR$(58)+" COMMA"
240 E$=E$+CHR$(44)+" QUOTE"+CHR$(34)+"**":
      ¬FORJ=1TOLEN(E$):PRINTMID$(E$,J,1);
250 POKE205,0:NEXT:PRINT
260 :
270 L=GX:GOSUB310:IFGG$="XX"THENEND
280 LL=LEN(GG$):IFLL>LTHENPRINT:PRINT"TOO ¬
      ¬LONG":GOTO270
290 PRINT:FORJ=1TOLL:PRINTMID$(GG$,J,1);:
      ¬POKE205,0:NEXT:PRINT:PRINT:GOTO270
300 REM ====GET ROUTINE==============
310 GG$="":PRINT"";:GOSUB390:GS=GP
320 GETG$:IFG$=""THENGOSUB390:GOTO320
330 GA=ASC(G$):IFGA=GRANDGP>GSGOTO380
340 IFGA=GHORGA=GDORGA=GLOR(GA=GT)ORGA=GEO
      ¬RGA=GUORGA=GMORGA=GRGOTO320
350 GP=GP-G1*(-(GA=GV)):IFGP<GSGOTO320
360 PRINTG$;:IFGA=GVTHENGG$=LEFT$(GG$,
      ¬GP-GS):GOTO320
370 GG$=GG$+G$:GOTO320
380 RETURN
390 GP=PEEK(G4)+G8*PEEK(G5)+PEEK(G6):
      ¬GOSUB400:GOSUB400:POKEGQ,G0:RETURN
400 GF=PEEK(GP):POKEGP,(G7ORGF)AND(NOT(G7A
      ¬NDGF)):RETURN
410 REM==============================
420 REM
430 REM 1.FLASHING CURSOR AND DISABLE
440 REM   OF QUOTE MODE IS IN THE LAST
```

```
450 REM   2 LINES OF THE GET ROUTINE.
460 REM 2.CURSOR LEFT CAN BE USED TO
470 REM   MAKE CORRECTIONS IN INPUT
480 REM 3.ALL OTHER CURSOR KEYS ARE
490 REM   DISABLED , BUT THE ROUTINE
500 REM   CAN BE CUSTOMIZED TO USE
510 REM   ANY DESIRED CURSOR KEYS LIS-
520 REM   TED IN THE INIT SECTION
530 REM     GS = STARTING CURSOR POS
540 REM     GP = PRESENT  CURSOR POS
550 REM 4.FOR CURSOR LEFT OR DELETE:
560 REM   STAY IN GET LOOP IF GP-1<GS
570 REM 5.FOR CURSOR UP: STAY IN GET
580 REM   LOOP IF GP-40<GS
590 REM   CURSOR MUST COME DOWN TO THE
600 REM   LAST CHARACTER ON THE LAST
610 REM   LINE IF IT WAS ABOVE IT, OR
620 REM   SOME INFO WILL BE LOST
630 REM 6.YOU MAY USE QUOTES, RVS,
640 REM   COMMA, COLON WHERE ALPHABE-
650 REM   TIC INFO IS PERMITTED
660 REM
670 REM 7.TO TAPE:PRINT#FF,GG$
680 REM   FROM TAPE:
690 REM    2 GET#FF,V$:CHECK STATUS:
700 REM    4 PRINT V$;:POKE205,0:GOTO2
710 REM
720 REM 8.FOR OLD ROMS+SMALL KEYBOARD
730 REM   SUBSTITUTE THESE VALUES IN
740 REM   THE 1ST LINE (NOT TESTED !)
750 REM
760 REM   G4=224:G5=225:G6=226:GQ=234
770 REM
780 REM==============================
```

©

ROM-antic thoughts

Jim Butterfield, Toronto

Here comes another ROM set or two from Commodore, and once again the user will need to take the decision: should he upgrade?

It's a tough question. If he does, it will cost money, and some of his programs may cease to work until they have been modified. If he doesn't, he'll be left behind and won't have access to some of the new goodies.

Basic programs will, as always, remain compatible, so long as they don't bristle with obscure PEEKs and POKEs. Machine language itself doesn't change, but programs which use routines built into the ROMs will need changing since the routines will have moved to new locations. Some commercial machine language programs will survive transfer to new ROMs, but many won't.

A more subtle problem creeps in. As the machine is enhanced, programs will start to use the new built-in features, and users may find themselves having to retro-convert these so that they will run on older systems. A command such as DOPEN is convenient and compact, but users who haven't converted up will have to translate this to the appropriate OPEN 1,8,3 ... command. New disk features will be particularly noticeable for this. New systems, for example, won't need to initialize disk and will offer very simple disk error checking; older systems will need to add extra coding to do these. Some new disk features such as APPEND or Relative files have no counterpart on the old systems.

Some terminology

Commodore are currently referring to ROM sets by means of a numbering scheme. They translate roughly as follows:

Basic 1.0 Original ROM, as fitted in the early 4K and 8K PETS. Not good for disk I/O; arrays limited to 256 elements; cassette tape files a little awkward.

Basic 2.0 Upgrade ROM, fitted on more recent machines. Garbage collection still a problem. Keyboard/disk interface rather clumsy. Built-in Machine Language Monitor. Linefeed output to IEEE a minor problem.

Basic 4.0 New ROM, currently being released. Disk commands built into Basic. Garbage collection fast, and Linefeed problem eliminated. Uses more ROM space. Available for both 40- and 80-column machines, but not for original PET 8K hardware.

Basic 5.0 Business ROM, not yet released. Rumoured to have many Basic enhancements, including high-precision decimal arithmetic.

Basic 2.0 and 4.0 have alternate versions for the two types of keyboard -- graphics or business.

Disk systems:

Dos 1.0 Original 2040 disk system. INITIALIZE command needed; RENAME sometimes doesn't work.

DOS 2.0 New system, currently being released. INITIALIZE not needed but allowed. Relative files and APPEND command implemented. Fast BACKUP command. Can be retrofitted to early 2040 units. The new 8050 disk system will have characteristics similar to DOS 2.0.

Printer ROM systems haven't settled down yet. There are two systems available, but both have minor problems; a third is rumoured.

Upgrading: the Options

Users who still have Basic 1.0 should upgrade, at least to Basic 2.0. There are too many good things available.

The original 8K machines cannot be readily upgraded beyond Basic 2.0; their hardware won't support Basic 4.0.

It's not necessary to upgrade both Basic and DOS ROMs at the same time, but it's probably a good idea. Basic 4.0 and DOS 2.0 work harmoniously together.

Switch to Basic 4.0 if you need any of the following:

--to be up to date with the latest software;
--to eliminate garbage collection delays;
--to allow inexperienced users to use the disk with more natural, English-like commands.

Switch to DOS 2.0 if you want to take advantage of the new APPEND feature or the powerful relative file structure.

Summary

If you still have original ROMs (the ones that say*** COMMODORE BASIC *** upon power-up), plan to upgrade.

It's your option as to whether you want to switch to Basic 4.0 and DOS 2.0. If you like the new features, go ahead. But you'll still have a good, serviceable system if you stay with upgrade ROM (Basic 2.0).

Upgrading the disk file can be treated as a separate question. The original unit is excellent for program saving and loading. But if you plan to do a lot of work with data files, the new features of DOS 2.0 can look very attractive.

It's your choice. ©

Converting ASCII Files to PET BASIC

Harvey B. Herman
Chemistry Department
University of North Carolina at Greensboro
Greensboro, North Carolina 27412

Recently I have been experimenting with a program (not discussed here) which makes PET into a terminal (CompuMart T/C 2001 terminal option) which can communicate with remote computers. Normally, the characters that are received by the PET, when acting as a terminal, are displayed on the screen. I modified the program to optionally save the characters (ASCII Code) to a reserved area in high memory (approximately decimal 8192 and above). Obviously, this program required memory in this area and will need to be modified for an unexpanded 8K PET. The question one might ask is, "What can I do with an ASCII file in high memory?". This article is intended to answer that question.

Commodore's PET is not the first computer I have worked with and I suspect the same may be true for many readers. I have spent many hours developing BASIC programs on remote computers for use with my research and in my teaching. It would be advantageous if I could also use these programs (suitably modified) on the PET. I have no strong desire to retype all these programs again. If I could convert the ASCII file of a program listing made by a terminal program into a PET BASIC program it would save immense amounts of work. Any minor changes could then be done with the screen editor.

The program called ASCII shown in the figure, converts ASCII files in high memory into BASIC programs. It is intended for use with expanded PETs with "old" ROMs. The POKE locations in statement number 63290, 525-527-528, need to be changed to 158-623-624 for "new" ROMs. The program uses the dynamic keyboard idea of Mike Lauder (see Best of PET Gazette). It writes two lines on the screen and puts two carriage returns in the keystroke buffer. The first line is a BASIC statement taken from part of a program listing saved in high memory by the terminal program. The second line is an immediate mode statement which restores a memory position counter and jumps back into the main program again. It is necessary to remember the current position in high memory because all variables were set to zero after the previous step. This is true whenever a new BASIC statement is added to a program, as in this case. All the new BASIC statements are added to the front of the main program which was purposely written with very large statement numbers. At the conclusion of this program, the statements belonging to the ASCII program can be deleted by hand or with The Programmer's BASIC Toolkit.

The ASCII program can be used to do a minor amount of editing "on the fly". Some of my original programs were done on a computer which uses '#' instead of '<>' and I included a conversion in statement 63180. Also '[' and ']' were used in place of '(' and ')', in some places and this conversion is done in statements 63160 and 63170. I also removed '7 (bel character) from the programs. Besides giving a syntax error later when run on a PET program, the inclusion of '7 occasionally caused lines to be over 80 characters long. This stopped the ASCII program with a syntax error which then had to be manually restarted. All these programmed editing changes saved a lot of manual editing later.

The end of my ASCII files is signified by the ASCII character 4, otherwise the program might continue indefinitely, adding unwanted BASIC statements, or garbage. This check is done in statement 63200. It should reach this point and stop if each line begins with a number, is less than 80 characters long, and the counter in statement 63070 is positioned to the beginning of the ASCII listing in high memory. Conversion to PET syntax, if required, would begin here.

I have used the ASCII program to convert very large ASCII files to PET programs. The same program should be useful when I acquire CP/M ASCII files on 5-1/4" diskettes. The disk and operating system which I am using (PEDISK and Wilserv Software) can read CP/M files, and the ASCII program discussed here, will convert them to PET BASIC programs.

```
63000 REM PROGRAM CONVERTS AN ASCII FILE ¬
      ¬IN
63010 REM HIGH MEMORY TO A PET BASIC ¬
      ¬PROGRAM
63020 REM HIGH MEMORY BEGINS AT $2017(8215
      ¬ DEC)
63030 REM
63040 REM HARVEY B. HERMAN
63050 REM
63060 REM I IS MEMORY COUNTER
63070 I=8215
63080 REM THROW AWAY FIRST LINE
63090 A=PEEK(I):IFA<>13THENI=I+1:GOTO63090
63100 PRINT"ĥ↓↓↓"
63110 I=I+1
63120 REM NEXT CHARACTER FROM ON HIGH
63130 A=PEEK(I)
63140 REM REPLACE [ & ] WITH ( & )
63150 REM REPLACE # WITH <>
```

```
63160 IF A=91 THEN A=40
63170 IF A=93 THEN A=41
63180 IF A=35 THEN PRINT"<>";:GOTO63110
63190 REM CHAR $04 AT END OF FILE
63200 IF A=4 THEN STOP
63210 REM THROW AWAY '7
63220 IF A=39 THEN IF PEEK(I+1)=55 THEN ¬
      ¬I=I+1:GOTO 63110
63230 REM PRINT BASIC LINE ON SCREEN. ¬
      ¬AFTER CR
63240 REM PRINT NECESSARY VARIABLES AND ¬
      ¬PUT CR
63250 REM IN KEYSTROKE BUFFER. END PROGRAM
63260 REM INCORPORATE LINE AND BEGIN AGAIN
63270 PRINT  CHR$(A);
63280 IF A=13 THEN PRINT"I=";I;":GOTO63100
      ¬h↓"
63290 IF A=13 THEN POKE 527,13:POKE528,13:
      ¬POKE 525,2:END
63300 GOTO63110
```

©

Compactor

Robert W. Baker,
Baker Enterprises,
15 Windsor Drive,
Atco, NJ 08004

This program is the result of several days of experimenting with BASIC program structures and the 2040 disk. In short, the program will read a BASIC program that was saved on disk and create a new, compacted copy. The program will delete all REMarks, unnecessary spaces, and leading colons. Much of this is similar to other utility programs currently available. However, this program goes one step further. It combines program lines whenever possible to eliminate the link, line number, and line-end-flag overheads normally associated with each line. It will make a program as small as possible, and most likely faster running.

While creating this program, I came across a few undocumented "quirks" of Commodore BASIC. Since many people are currently experimenting with the capabilities of having programs "write" programs on disk, this information may be of interest:

Zero Length Lines:

Normally, it is impossible to create a zero length line using the screen editor on the PET. By zero length line, I mean a line with a link, line number, and end-of-line flag; but no BASIC commands or text. If you were to type just a line number using the screen editor, you would actually delete a line instead of entering a zero length line. However, when writing a BASIC program on disk as a data file there is nothing stopping you from entering a zero length line. But if you want the program to run, you cannot have any zero length lines in the program. BASIC cannot link the program lines correctly whenever there is a zero length line in the program.

Long Lines:

At the other extreme, you cannot create a BASIC line that is longer than 255 bytes. Again, using the PET screen editor you could not create such a line. You are normally limited to a maximum of 78 bytes due to the line wrapping characteristics and at least a one digit line number. When writing a BASIC program on disk as a data file, be careful not to create a line greater than 255 bytes. Otherwise the program will usually not load from the disk. If it does load, the program will be totally destroyed and unuseable.

Printing Long Lines:

Here's a quick comment on the Commodore printers. If you list a program that contains lines longer than 80 characters, the printed listing may be incorrect. It appears that the printer occasionally switches out of listing mode and into print mode when a line exceeds 80 characters. At the start of the next line everything is ok again.

Program Description

When running the COMPACTOR program, the BASIC program to be compacted must be on the diskette inserted in drive #0. The new compacted version will be written on the diskette in drive #1 with the same filename, but with a "/C" suffix. The program will read the program to be compacted as a sequential disk data file, and the file will be read twice.

The first pass checks for line numbers within the subject program that are the targets of: GOTO, GOSUB, or IF...THEN..(line#) statements. When a target line number is found, it is saved in matrix TL if not all ready saved. A check is also made for multiple target lines in ON..GOTO and ON.. GOSUB statements. Each target line will be displayed on the PET screen in the order found. This helps give some indication of the scanning progress since it can be rather slow.

During the second pass, each line is copied, deleted, or compacted as appropriate. The line number will be displayed as each line is processed to let you know how the program is progressing. The rules followed by the COMPACTOR are as follows:

Any leading colons and/or spaces on a line are deleted.

A line that has only REMarks is deleted if it is not a target line. The remark will be replaced with a single colon if the line is a target line and must be retained. This may produce a leading colon if the next line is not a target line and is combined with this line. The line cannot be reduced to a zero length line since BASIC cannot link a program correctly with a zero length line, as mentioned earlier.

If any line contains an IF...THEN or GOTO statement, another line cannot be combined with this line. Any line combined after these BASIC commands would never be executed, thus the program would not function properly.

Any spaces within a line, not enclosed in quotes, are deleted.

Any REMarks at the end of a BASIC line are deleted to the end of the line.

Anything within quotes is copied, untouched. If an ending quote is missing from the line, one is

PET SOFTWARE

LAS VEGAS CASINO SERIES:

These four programs were developed both as a tutorial for those planning to visit a Casino and wanting to learn to play correctly, and as a means for the serious gambler to develop and thoroughly test a gambling 'system' under actual Casino conditions. All betting odds and options available in the Casino of the MGM Grand Hotel in Las Vegas have been incorporated into these programs. Full screen graphics have been used to show the cards being dealt, the Roulette Wheel spinning, and the Dice being thrown in order to increase realism and heighten enjoyment.

1. **Casino Blackjack:**
For 1 to 5 players plus the dealer. Keeps track of winnings or losses for each player plus number of hands won, lost, and tied and the number of times the dealer and each player has busted. Play with 1, 2, or 4 decks. Change decks or reshuffle at any time. Split pairs, Double Down, or place an Insurance bet. Full Casino rules and many other options .... $ 7.95

2. **Casino Roulette:**
Bet on one number, two numbers, odd, even, black, etc. Watch the numbers come up as the wheel spins. Twelve ways to bet .... $ 7.95

3. **Casino Baccarat:**
James Bond's favorite game. Two games in on. Casino style and Blackjack style. Includes special features to help in developing a winning 'system' .... $ 7.95

4. **Casino Craps:**
Bet the Dice to Pass or Fall Off. Bet the Hardway or Press with Double Odds. Ten ways to bet .... $ 7.95

5. **Casino Package:**
All four Casino Programs above $24.95

STRATEGIC GAMES:

6. **Backgammon:**
Play Backgammon against your PET. Excellent graphics and doubling option make for a fast and exciting game .... $ 9.95

7. **SP(*) INTRUDERS: (WITH SOUND)**
Written by COMMODORE-JAPAN. Performs exactly like the popular video arcade version being played all over the country. Machine language graphics and sound provide hours of entertainment .... $ 9.95
Please specify which ROM set your PET has.

8. **Checkers:**
Play against the PET. PET plays a good, fast game according to International Checker Rules. Excellent graphics show the board and all checkers. Watch your PET move his man around the board. Clock shows elapsed time for each move .... $ 7.95

9. **Qubic-4:**
This is three dimensional Tic-Tac-Toe played on four, 4 x 4 boards. PET plays a fast exciting game choosing one of three strategies for each game .... $ 7.95

10. **Go Moku:**
Ancient Chinese board game played on a 9 x 9 board. Get 5 men in a row before your PET. A different strategy for each game .... $ 7.95

11. **Othello:**
English game known as Reversi. Try to capture the PET's men before he captures yours. Play against the PET or against your friends. Fast and fun .... $ 7.95

12. **Cribbage:**
An Excellent version of this favorite card game. All cards are shown using PET's excellent graphics. The PET plays a cool logical game difficult to beat even for the best players .... $ 7.95

13. **Game Package:**
Any six of the above programs $34.95

GUARANTEE: All programs are guaranteed to be free from errors and to load on any PET. Any defective tapes will be replaced free of charge.

master charge THE INTERBANK CARD

VISA®

Order From: **CMS SOFTWARE,** 5115 Menefee Drive, Dallas, TX 75277 added if another line could be combined with this line. Therefore, if a line does not contain an IF.. THEN or GOTO statement, an ending quote is added.

When a colon is found within a BASIC line and not within quotes, the next non-space character is checked before copying the colon. If a REMark follows the colon, the colon and the rest of the line is deleted. Otherwise, the colon is copied and processing continues as normal.

At the end of each BASIC line, a check is made to see if the next line can be combined with this line. If there were no IF...THEN or GOTO commands, and the next line is not a target line, the lines are combined. When combining lines, the line and line number are discarded, a colon is written, and the next line is processed as part of the previous line.

If the next line cannot be combined with the current line, the end of line flag is copied along with a dummy link and the next line number. A dummy link is used to avoid excessive processing and working buffers necessary with calculating program links. Besides, the links are automatically corrected by PET BASIC with the RUN or CLR commands. As a standard operating procedure, the newly created program outputted by COMPACTOR should be loaded and re-linked, then re-saved onto disk. The program can be re-linked by issuing a CLR command after being loaded.

As mentioned previously, a BASIC program line cannot exceed 255 bytes in length. If it does, the program may not load from disk or it may be totally unuseable. To protect against this, the COMPACTOR program stops combining lines if more than 170 bytes have been written in a single BASIC line. Since any normal line cannot exceed 78 bytes in length, this should insure than no program generated lines are longer than the maximum length.

As an example of what this program will do, I included a listing of a compacted version of the COMPACTOR program itself. Since this program has many REMarks, compacting saves over 3000 bytes for about a 50% saving in memory space. On most programs the savings will be much smaller, depending on the programming style. A side benefit, however, is the increase in the operating speed of compacted programs. I should warn, though, that the compacting process can be rather slow. Compacting of the COMPACTOR program (a 6K program with all the REMarks) takes about 16 minutes. But all you have to do is start it off and then go get a cup of coffee while the PET does the work! And you only have to run it once for any given program!

For those too lazy to type in the program, I'll be happy to provide copies on tape at $2 each.

```
10 REM ***************************
20 REM *    C O M P A C T O R    *
30 REM *   -------------------   *
40 REM *    BY:  ROBERT BAKER    *
50 REM *                         *
60 REM *    BAKER ENTERPRISES    *
70 REM *     15 WINDSOR DR.      *
80 REM *    ATCO, N.J.  08004    *
90 REM ***************************
100 :
110 CLR : DIM TL(1000)
120 :
130 REM ****************
140 REM READY DISK FILES
150 REM ****************
160 :
170 PRINT"ĥ"SPC(15)"ɾCOMPACTOR↓↓
180 PRINT" ɾINPUTr̂ FILE IN ɾDRIVE #0↓
190 PRINT"ɾOUTPUTr̂ FILE IN ɾDRIVE #1↓↓
200 INPUT"ɾINPUT FILE NAMEr̂";FL$
210 PRINT"ĥSCANNING FILE
220 PRINT"   FOR TARGET LINES.....↓↓
230 OPEN 15,8,15 : GOSUB 2370
240 OPEN 5,8,5,"0:"+FL$+",P,R"
250 :
260 REM ***************************
270 REM READ LOAD ADR, LINK & LINE#
280 REM ***************************
290 :
300 GOSUB 2370 : GOSUB 2310
310 GOSUB 2310 : IF V+V1=0 THEN 790
320 GOSUB 2310 : LN=V1+(256*V)
330 :
340 REM *****************************
350 REM         SCAN BASIC LINES
360 REM FOR GOTO, GOSUB & THEN TOKENS
370 REM *****************************
380 :
390 GOSUB 2330
400 IF V=0 THEN 310
410 IF V=137 OR V=141 THEN 480
420 IF V<>167 THEN 390
430 :
440 REM ****************
450 REM GET TARGET LINE#
460 REM ****************
470 :
480 LT=0
490 GOSUB 2330 : IF V=32 THEN 490
500 IF V<48 OR V>57 THEN 580
510 LT=(10*LT)+VAL(C$)
520 GOSUB 2330 : GOTO 500
530 :
540 REM ***********************
550 REM CHECK IF ALL READY FOUND
560 REM ***********************
570 :
580 FOR X=0 TO N
590 IF TL(X)=LT THEN 710
600 NEXT X
610 TL(N)=LT : N=N+1
620 PRINT LT,
630 IF N<1000 THEN 710
640 PRINT"↓↓TOO MANY TARGET LINES!
650 GOTO 2430
660 :
670 REM ***************************
680 REM CHECK FOR 'ON...GOTO/GOSUB'
690 REM ***************************
700 :
710 IF V=44 THEN 480
720 IF V<>32 THEN 400
730 GOSUB 2330 : GOTO 710
740 :
```

```
750 REM *****************
760 REM SORT TARGET LINES
770 REM *****************
780 :
790 IF N<2 THEN 900
800 FOR X=0 TO N-1
810 FOR Y=0 TO N-2
820 IF TL(Y) < TL(X) THEN 840
830 V=TL(Y) : TL(Y)=TL(X) : TL(X)=V
840 NEXT Y,X
850 :
860 REM *********************
870 REM GET READY FOR COMPACT
880 REM *********************
890 :
900 PRINT "ñCOMPACTING LINES....↓↓
910 CLOSE 5
920 OPEN 5,8,5,"0:"+FL$+",P,R"
930 GOSUB 2370
940 FO$=LEFT$(FL$,14)+"/C"
950 PRINT#15,"S1:"+FO$
960 OPEN 6,8,6,"1:"+FO$+",P,W"
970 GOSUB 2370
980 :
990 REM  **************
1000 REM COPY LOAD ADR
1010 REM **************
1020 :
1030 GOSUB 2310
1040 PRINT#6,CHR$(V1);
1050 PRINT#6,CHR$(V); : R=0
1060 :
1070 REM ***********************
1080 REM COPY LINK & LINE NUMBER
1090 REM ***********************
1100 :
1110 GOSUB 2310 : K1=V1 : K2=V
1120 F=0 : IF V+V1=0 THEN 2230
1130 GOSUB 2310 : L1=V1 : L2=V
1140 LN=L1+(256*L2) : PRINT LN,
1150 GOSUB 2330
1160 IF V=32 OR V=58 THEN 1150
1170 IF V=0 THEN 1200
1180 IF V<> 143 THEN 1240
1190 GOSUB 2330 : IF V>0 THEN 1190
1200 F=1 : FOR X=0 TO N
1210 IF TL(X)<LN THEN NEXT X
1220 IF TL(X)=LN THEN 1240
1230 GOTO 1110
1240 PRINT#6,CHR$(K1);CHR$(K2);
1250 PRINT#6,CHR$(L1);CHR$(L2); : R=4
1260 IF F THEN PRINT#6,":"; : R=5
1270 F=0 : GOTO 1360
1280 :
1290 REM **************************
1300 REM ****  SCAN BASIC LINE  ***
1310 REM **** & COMPACT PROGRAM ***
1320 REM **************************
1330 :
1340 PRINT#6,C$; : R=R+1
1350 GOSUB 2330
1360 IF V=137 THEN F=1
1370 IF V=139 OR V=167 THEN F=1
1380 IF V=0 THEN 1820
1390 IF V=32 THEN 1350
1400 :
1410 REM ********************
1420 REM 'REM' TOKEN -
1430 REM DISCARD REST OF LINE
1440 REM ********************
1450 :
1460 IF V<>143 THEN 1550
1470 GOSUB 2330 : IF V>0 THEN 1470
1480 GOTO 1820
1490 :
1500 REM **************************
1510 REM QUOTE -
1520 REM COPY TILL NEXT OR LINE END
1530 REM **************************
1540 :
1550 IF V<>34 THEN 1690
1560 PRINT#6,C$; : R=R+1
1570 GOSUB 2330
1580 IF V=34 THEN 1340
1590 IF V>0 THEN 1560
1600 IF F THEN V=0 : GOTO 1050
1610 PRINT#6,CHR$(34); : R=R+1
1620 GOTO 1820
1630 :
1640 REM ************************
1650 REM IF COLON - CHK NEXT CHAR
1660 REM       ELSE COPY CHAR
1670 REM ************************
1680 :
1690 IF V<>58 THEN 1340
1700 GOSUB 2330
1710 IF V=32 OR V=58 THEN 1700
1720 IF V=143 THEN 1470
1730 IF V=0 THEN 1820
1740 PRINT#6,":"; : R=R+1
1750 GOTO 1360
1760 :
1770 REM ****************************
1780 REM END OF LINE -
1790 REM CAN WE COMPACT THESE LINES ?
1800 REM ****************************
1810 :
1820 IF F OR (R>170) THEN V=0:GOTO 1050
1830 GOSUB 2310
1840 IF V+V1=0 THEN 2230
1850 GOSUB 2310 : LN=V1+(256*V)
1860 L1=V1 : L2=V : PRINT LN,
1870 :
1880 REM ************************
1890 REM CHK IF LINE# IS A TARGET
1900 REM ************************
1910 :
1920 FOR X=0 TO N
1930 IF TL(X)<LN THEN NEXT X
1940 IF TL(X)=LN THEN 2110
1950 :
1960 REM ********************
1970 REM NOT USED -
1980 REM DISCARD LINK & LINE#
1990 REM ********************
2000 :
2010 GOSUB 2330 : IF V=143 THEN 1470
2020 IF V=32 OR V=58 THEN 2010
2030 IF V=0 THEN 1830
2040 PRINT#6,":"; : R=R+1 : GOTO 1360
2050 :
2060 REM ****************************
2070 REM LINE# NEEDED -
2080 REM WRITE LINE END, LINK & LINE#
2090 REM ****************************
2100 :
2110 PRINT#6,CHR$(0);CHR$(1);CHR$(1);
2120 PRINT#6,CHR$(L1);CHR$(L2); : R=4
2130 GOSUB 2330
2140 IF V=32 OR V=58 THEN 2130
2150 IF V=0 OR V=143 THEN PRINT#6,":";
2160 F=0 : GOTO 1360
2170 :
2180 REM ********************
2190 REM END OF COMPACT -
2200 REM WRITE END OF PROGRAM
2210 REM ********************
2220 :
```

```
   2230 PRINT#6,CHR$(Ø);CHR$(Ø);CHR$(Ø);
   2240 PRINT"ĥɼDONE↓↓"
   2250 GOTO 2430
   2260 :
   2270 REM ************************
   2280 REM *****  SUBROUTINES  *****
   2290 REM ************************
   2300 :
   2310 GOSUB 2330 : V1=V
   2320 :
   2330 GET#5,C$ : GOSUB 2370
   2340 IF C$="" THEN V=Ø : RETURN
   2350 V=ASC(C$) : RETURN
   2360 :
   2370 INPUT#15,EN,EM$,ET,ES
   2380 IF EN=Ø THEN RETURN
   2390 :
   2400 PRINT : PRINT"↓↓↓ɼDISK ERROR↓
   2410 PRINT EN;EM$;ET;ES
   2420 :
   2430 CLOSE 5 : CLOSE 6 : CLOSE 15
  READY.
```

```
 110 CLR:DIMTL(1ØØØ):PRINT"ĥ"SPC(15)"ɼCOMPA
       ¬CTOR↓↓":PRINT" ɼINPUTf̂ FILE IN ¬
       ¬ɼDRIVE #Ø↓":PRINT"ɼOUTPUTf̂ FILE IN ¬
       ¬ɼDRIVE #1↓↓":INPUT"
ɼINPUT FILE NAMEf̂";FL$:PRINT"ĥSCANNING ¬
       ¬FILE":PRINT"   FOR TARGET LINES.....
       ¬↓↓"
 230 OPEN15,8,15:GOSUB2370:OPEN5,8,5,"Ø:
       ¬"+FL$+",P,R":GOSUB2370:GOSUB2310
 310 GOSUB2310:IFV+V1=ØTHEN79Ø
 320 GOSUB2310:LN=V1+(256*V)
 390 GOSUB233Ø
 400 IFV=ØTHEN31Ø
 410 IFV=137ORV=141THEN48Ø
 420 IFV<>167THEN39Ø
 480 LT=Ø
 490 GOSUB233Ø:IFV=32THEN49Ø
 500 IFV<48ORV>57THEN58Ø
 510 LT=(1Ø*LT)+VAL(C$):GOSUB233Ø:GOTO5ØØ
 580 FORX=ØTON:IFTL(X)=LTTHEN71Ø
 600 NEXTX:TL(N)=LT:N=N+1:PRINTLT,:
       ¬IFN<1ØØØTHEN71Ø
 640 PRINT"↓↓TOO MANY TARGET LINES!":
       ¬GOTO243Ø
 710 IFV=44THEN48Ø
 720 IFV<>32THEN4ØØ
 730 GOSUB233Ø:GOTO71Ø
 790 IFN<2THEN9ØØ
 800 FORX=ØTON-1:FORY=ØTON-2:IFTL(Y)<TL(X)T
       ¬HEN84Ø
 830 V=TL(Y):TL(Y)=TL(X):TL(X)=V
 840 NEXTY,X
 900 PRINT"ĥCOMPACTING LINES....↓↓":CLOSE5:
       ¬OPEN5,8,5,"Ø:"+FL$+",P,R":GOSUB237Ø:
       ¬FO$=LEFT$(FL$,14)+"/C":PRINT#15,"S1:
       ¬"+FO$:OPEN6,8
,6,"1:"+FO$+",P,W":GOSUB237Ø:GOSUB231Ø:
       ¬PRINT#6,CHR$(V1);
 1050 PRINT#6,CHR$(V);:R=Ø
 1110 GOSUB231Ø:K1=V1:K2=V:F=Ø:IFV+V1=ØTHEN
       ¬223Ø
 1130 GOSUB231Ø:L1=V1:L2=V:LN=L1+(256*L2):
       ¬PRINTLN,
 1150 GOSUB233Ø:IFV=32ORV=58THEN115Ø
 1170 IFV=ØTHEN12ØØ
 1180 IFV<>143THEN124Ø
 1190 GOSUB233Ø:IFV>ØTHEN119Ø
 1200 F=1:FORX=ØTON:IFTL(X)<LNTHENNEXTX
 1220 IFTL(X)=LNTHEN124Ø
 1230 GOTO111Ø
 1240 PRINT#6,CHR$(K1);CHR$(K2);:PRINT#6,
       ¬CHR$(L1);CHR$(L2);:R=4:IFFTHENPRINT#
       ¬6,":";:R=5
 1270 F=Ø:GOTO136Ø
 1340 PRINT#6,C$;:R=R+1
 1350 GOSUB233Ø
 1360 IFV=137THENF=1
 1370 IFV=139ORV=167THENF=1
 1380 IFV=ØTHEN182Ø
 1390 IFV=32THEN135Ø
 1460 IFV<>143THEN155Ø
 1470 GOSUB233Ø:IFV>ØTHEN147Ø
 1480 GOTO182Ø
 1550 IFV<>34THEN169Ø
 1560 PRINT#6,C$;:R=R+1:GOSUB233Ø:IFV=34THE
       ¬N134Ø
 1590 IFV>ØTHEN156Ø
 1600 IFFTHENV=Ø:GOTO1Ø5Ø
 1610 PRINT#6,CHR$(34);:R=R+1:GOTO182Ø
 1690 IFV<>58THEN134Ø
 1700 GOSUB233Ø:IFV=32ORV=58THEN17ØØ
 1720 IFV=143THEN147Ø
 1730 IFV=ØTHEN182Ø
 1740 PRINT#6,":";:R=R+1:GOTO136Ø
 1820 IFFOR(R>17Ø)THENV=Ø:GOTO1Ø5Ø
 1830 GOSUB231Ø:IFV+V1=ØTHEN223Ø
 1850 GOSUB231Ø:LN=V1+(256*V):L1=V1:L2=V:
       ¬PRINTLN,:FORX=ØTON:IFTL(X)<LNTHENNEX
       ¬TX
 1940 IFTL(X)=LNTHEN211Ø
 2010 GOSUB233Ø:IFV=143THEN147Ø
 2020 IFV=32ORV=58THEN2Ø1Ø
 2030 IFV=ØTHEN183Ø
 2040 PRINT#6,":";:R=R+1:GOTO136Ø
 2110 PRINT#6,CHR$(Ø);CHR$(1);CHR$(1);:
       ¬PRINT#6,CHR$(L1);CHR$(L2);:R=4
 2130 GOSUB233Ø:IFV=32ORV=58THEN213Ø
 2150 IFV=ØORV=143THENPRINT#6,":";
 2160 F=Ø:GOTO136Ø
 2230 PRINT#6,CHR$(Ø);CHR$(Ø);CHR$(Ø);:
       ¬PRINT"ĥɼDONE↓↓":GOTO243Ø
 2310 GOSUB233Ø:V1=V
 2330 GET#5,C$:GOSUB237Ø:IFC$=""THENV=Ø:
       ¬RETURN
 2350 V=ASC(C$):RETURN
 2370 INPUT#15,EN,EM$,ET,ES:IFEN=ØTHENRETUR
       ¬N
 2400 PRINT:PRINT"↓↓↓ɼDISK ERROR↓":
       ¬PRINTEN;EM$;ET;ES
 2430 CLOSE5:CLOSE6:CLOSE15
READY.
```

©

BLACKJACK ANALYZER

PUT PET TO WORK

PLAYS 100,000 games in 24 minutes.
Find the most advantageous method.

User defines player method by simple keyboard entry including when to hit and double on soft and hard hands, split pairs and take insurance. Then run simulator and see resulting gain or loss in 24 minutes run time.

Uses 4 Decks. Full casino rules. Countless permutations.
- OR -
Switch to visual play and watch your system in real time.
8K. Specify old or new ROM. 6502 ML
Cassette Tape $12.00

COUNTING TUTOR

Play Blackjack and win by keeping track of high and low cards. Program displays count on request if you lose track. A real time simulator.
8K. Specify old or new ROM. 6502 ML
Cassette tape $6.00

Malco
54 Hesbeth Court
Toronto, CANADA
M4A 1M6

Oops! Our address was misspelled in the last issue. If your letter was returned, please try again.

☆ *BUSINESS* ☆ *EXPERIMENTAL* ☆ *EDUCATIONAL*
☆ *TELECOMPUTING* ☆ *PERSONAL* ☆ *RENTALS*

micro systems by
PETTED
hardware with a soft touch

SPECIALIZING IN

Commodore Business Machines

PRODUCTS BY

CmC, NOVATION, TNW CORP., BIZCOMP, MICRO, CENTURY RESEARCH, TAB, POW'R PROTECT'R, systems, BASE 2, OSBORN, Eaton, SOURCE, MACTRONICS, MicroNet, Skyles, KILOBAUD, Dr. DALEY'S, INSTANT SOFTWARE, ASCRO, MICRO SOFTWARE SYSTEMS, COMPUTER FORUM, COMPETITATIVE SOFTWARE, HAYDEN, ATARI, NEECO, COMPUTE & COMPUTE II, DYSAN, CREATIVE COMPUTING, LIVERMORE RESEARCH, ROBOTICS AGE, SYBEX, KNIGHT ENTERPRISES.

Another Last Minute News Note: Commodore has announced that the three for two educational offer is on again.

Send For Our Free Catalog

Mail Orders To:

PETTED
P. O. BOX 21851
Milwaukee, Wisconsin 53221

PETTED micro systems
is located across from the SURFIN' TURF skate board park.
Highway I-894 & Hwy 36 *4265 W. Loomis Rd.*
Milwaukee, Wisconsin

VOICE (414) 282-4181 ☆ **PBBS (414) 282-8118**

VISA® 10047,101 master charge

☆ **Vendors Display Your Products With Us.** ☆

ELCOMP

NEW Books

Care and Feeding of the Commodore PET
Eight chapters exploring PET hardware. Includes repair and interfacing information. Programming tricks and schematics.
Order No. 150 **$11.00**

8K Microsoft BASIC Reference Manual
Authoritative reference manual for the original Microsoft 4K and 8K BASIC developed for Altair and later computers including PET, TRS 80, and OSI. OSI owners please take note!
Order No. 151 **$ 9.95**

Expansion Handbook for 6502 and 6802
(S 44 Card Manual) Describes all of the 4.5 x 6.5 44 pin S 44 cards incl. RAM, ROM, dig. I/O, MUX/A to D, EPROM Prog. etc. With schematics and funct. descriptions. A must for every KIM, SYM and AIM owner.
Order No. 152 **$ 9.95**

Microcomputer Application Notes
Reprint of Intels' most important application notes, including 2708, 8085, 8255, 8251 chips. Very necessary for the hardware buff.
Order No. 153 **$ 9.95**

Complex Sound Generation
New, revised applications manual for the Texas Instruments SN 76477 Complex Sound Generator. Circuit board available ($8.95)
Order No. 154 **$ 6.95**

Small Business Programs
Complete programs for the business user. Mailing List, Inventory, Invoice Writing and much more. Introduction into Business Applications. Many listings.
Order No. 156 **$14.90**

The First Book of Ohio Scientific, Vol. I
Contains an introduction to personal computers and describes the Ohio Scientific Line. Contains explanatory diagrams, block, hook-up, expansion, tricks, hints and many interesting listings. Hardware and software information not previously available in one compact source. 192 pages.
Order No. 157 **$ 7.95**

The First Book of Ohio Scientific, Vol. II
Vol. II contains very valuable information about Ohio Scientific microcomputer systems. Introduction to OS 65D and OS65 U, networking and distributed processing, systems specifications, business applications, hard and software hints and tips.
Order No. 158 **$ 7.95**

Mailing List Program for Challenger C1/C2 8K
Order No. 2004 - Personal Version **$ 9.95**
Order No. 2005 - Business Version **$ 9.95**

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 No. 1105 **$12.00**

Important Software for CBM 16K/32K
Most powerful Editor/Assembler for Commodore CBM 16/32K on cassette. Very fast— Editor divides screen into 3 parts. Scrolling text window. 24 direct commands, 19 serial commands, status and error messages. Assembler can be started directly from the editor or from the TIM monitor. Translates in three passes. If an error is encountered, automatic return to the editor. Cassette with DEMO.
Order No. 3276 **$69.00**

ATTENTION APPLE USERS
Same as above for Apple II or Apple II plus.
Order No. 3500 **$89.00**

MONJANA/1 makes Machine Language Programming easy!
In every Commodore CBM there is a spare ROM socket waiting for its MONJANA/1. The new MONJANA/1 Machine Language Monitor in ROM offers more user guidance and debugging aids than any other monitor available today. It is indispensable for anyone intending to take full advantage of the computers features. Trace, link, disassemble, dump, relocate, line assemble and much more. Every command function has demand printout option. Price includes extensive manual.
Order No. 2001 **$98.00**

JANA-Monitor on cassette for the PET
Similar to MONJANA/1 — very powerful.
Order No. 2002 **$29.00**

ELCOMP PUBLISHING Inc.
3873-L Schaefer Ave., Chino CA 91710 (714) 591-3130

Please send me the books/software indicated below.

☐ I enclose $ ____________ send postpaid.
☐ Send COD ($5 extra)
☐ Charge my ☐ VISA ☐ Mastercharge
Acct. No. ____________
Expir. date ______ Signature ____________

Book No. Book No. Software No.

☐ 1 Year subscription to ELCOMP Newsletter $9.80

Name Phone
Address
City State Zip

CA add 6% sales tax. We also accept Eurocheck. All orders outside USA must add 15% shipping.

A few entry points, original/ upgrade/ 4.0 ROM

Jim Butterfield

Entry points seen in various programmer's machine language programs. The user is cautioned to check out the various routines carefully for proper setup before calling, registers used, etc.

ORIG	UPGR	4.0	DESCRIPTION
C357	C355	B3CD	?OUT OF MEMORY
C359	C357	B3CF	Send Basic error message
C38B	C389	B3FF	Warm start, Basic
C3AC	C3AB	B41F	Crunch & insert line
C430	C439	B4AD	Fix chaining & READY.
C433	C442	B4B6	Fix chaining
C48D	C495	B4fB	Crunch tokens
C522	C52C	B5A3	Find line in Basic
C553	C55D	B5D4	Do NEW
C567	C572	B5E9	Reset Basic and do CLR
C56A	C575	B5EC	Do CLR
C59A	C5A7	B622	Reset Basic to start
C6B5	C6C4	B74A	Continue Basic execution
C863	C873	B8F6	Get fixed-point number from Basic.
C9CE	C9DE	BADB	Send Return, LF if in screen mode
C9D2	C9E2	BADF	Send Return, Linefeed
CA27	CA1C	BB1D	Print string
CA2D	CA22	BB23	Print precomputed string
CA47	CA43	BB44	Print "?"
CA49	CA45	BB46	Print character
CE11	CDF8	BEF5	Check for comma
CE13	CDFA	BEF7	Check for specific character
CE1C	CE03	BF00	'SYNTAX ERROR'
CFD7	CFC9	C187	Find fl-pt variable, given name
D079	D069	C2B9	Bump Variable Address by 2
D0A7	D09A	C2EA	Float to Fixed conversion
D278	D26D	C4BC	Fixed to Float conversion
D679	D67B	C8D7	Get byte to X reg
D68D	D68F	C8EB	Evaluate String
D6C4	D6C6	C921	Get two parameters
D73C	D773	C99D	Add (from memory)
D8FD	D934	CB5E	Multiply by memory location
D9B4	D9EE	CC18	Multiply by ten
DA74	DAAE	CCD8	Unpack memory variable to Accum #1
DAA9	DAE3	CD0D	Copy Acc #1 to (X,Y) location
DB1B	DB55	CD7F	Completion of Fixed to Float conversion
DC9F	DCD9	CF83	Print fixed-point value
DCA9	DCE3	CF8D	Print floating-point value
DCAF	DCE9	CF93	Convert number to ASCII string
E3EA	E3D8	E202	Print a character
na	E775	D722	Output byte as 2 hex digits
na	E7A7	D754	Input 2 hex digits to A
na	E7B6	D763	Input 1 hex digit to A
E7DE	F156	F185	Print system message
F0B6	F0B6	F0D2	Send 'talk' to IEEE
F0BA	F0BA	F0D5	Send 'listen' to IEEE
F12C	F128	F143	Send Secondary Address
E7DE	F156	F185	Send canned message
F167	F16F	F19E	Send character to IEEE
F17A	F17F	F1B6	Send 'untalk'
F17E	F183	F1B9	Send 'unlisten'
F187	F18C	F1C0	Inpute from IEEE
F2C8	F2A9	F2DD	Close logical file
F2CD	F2AE	F2E2	Close logical file in A
F32A	F301	F335	Check for Stop key
F33F	F315	F349	Send message if Direct mode
na	F322	F356	LOAD subroutine
F3DB	F3E6	F425	?LOAD ERROR
F3E5	F3EF	F42E	Print READY & reset Basic to start
F3FF	F40A	F449	Print SEARCHING. . .
F411	F41D	F45C	Print file name
F43F	F447	F486	Get LOAD/SAVE type parameters
F462	F466	F4A5	Open IEEE channel for output.
F495	F494	F4D3	Find specific tape header block
F504	F4FD	F53C	Get string
F52A	F521	F560	Open logical file from input parameters
F52D	F524	F563	Open logical file
F579	F56E	F5AD	?FILE NOT FOUND, clear I/O
F57B	F570	F5AF	Send error message
F5AE	F5A6	F5E5	Find any tape header block
F64D	F63C	F67B	Get pointers for tape LOAD
F667	F656	F695	Set tape buffer start address
F67D	F66C	F6AB	Set cassette buffer pointers
F6E6	F6F0	F72F	Close IEEE channel
F78B	F770	F7AF	Set input device from logical file number
F7DC	F7BC	F7DF	Set output device from LFN.
F83B	F812	F857	PRESS PLAY..; wait
F85E	F835	F87A	Sense tape switch
F87F	F855	F89A	Read tape to buffer
F88A	F85E	F8A3	Read tape
F8B9	F886	F8CB	Write tape from buffer
F8C1	F88E	F8D3	Write tape, leader length in A
F913	F8E6	F92B	Wait for I/O complete or Stop key
FBDC	FB76	FBBB	Reset tape I/O pointer
FD1B	FC9B	FCE0	Set interrupt vector
FFC6	FFC6	FFC6	Set input device
FFC9	FFC9	FFC9	Set output device
FFCC	FFCC	FFCC	Restore default I/O devices
FFCF	FFCF	FFCF	Input character
FFD2	FFD2	FFD2	Output character
FFE4	FFE4	FFE4	Get character

©

Feed Your PET Some APPLESOFT

G. A. Campbell
36 Doubletree Road
Willowdale, Ontario M2J 324

We all know that there is no such thing as compatibility in the world of personal computers. For example, the APPLE and the PET store programs on tape quite differently. However, by using the program in Listing 1, you can load programs from an APPLE directly into a PET. To be more specific, you can load APPLESOFT programs (cassette or ROM versions) into an upgrade-ROM PET. Conversion to original-ROM PET's is trivial.

Structure of an APPLESOFT tape

One of the things which make the process fairly easy is the simple way APPLE's save programs. A bit is stored as one full cycle on tape. A short cycle is a zero-bit, one about twice as long is a one-bit, and leader is slightly longer again. A byte is simply made up of eight bits, unlike the PET, which has a start-bit and a parity-bit. The high-order bit comes first.

A program is stored as two blocks. The first is a length block. It contains four bytes:

low-order half of program length
high-order half of program length
fixed hexadecimal '55'
checksum of the above.

The checksum is formed by beginning with hexadecimal 'FF', then doing an exclusive-or on each byte of the block.

The second block contains the exact image of the program as it resides in memory. It is suffixed by two bytes, the second of which is a checksum formed the same way as for the length block. These two bytes are not counted in the program length.

Each block is preceded on tape by about ten seconds of leader (long bits) and one zero-bit, and followed by some tape which is effectively blank.

The other thing which makes the task easy is that both APPLESOFT and PET BASIC were written by Microsoft, and thus programs have exactly the same format in memory.

The APPLESOFT Loader

The program in listing 1 has many comments to point out the subtleties of how it operates. The major functions are:

Initialize everything upon entry so the program can be rerun if there is an error.

Time the cycles passing the head on the cassette.

Throw away the first 'bit'.

Wait for the 'start-bit'.

Make bytes out of the following bits.

Do the checksum on the length block, and set up to read the actual program.

Convert the statement pointers if the program was cassette APPLESOFT.

Translate the BASIC tokens.

Convert the statement pointers from beginning at hexadecimal 0801 to hexadecimal 0501.

Move the program down from 0801 to 0501. (The code to do this is at the start of the program, since part of the loader is overlaid.)

Memory Requirements

The loader reads programs into the same location as ROM-based APPLESOFT. This is hexadecimal 0801, which is just above the screen on an APPLE. However, by the time the process is completed, the program has shuffled down to 0401. Thus, on an 8K PET you can load 6K of program text. Ignoring memory differences due to conversion, you have an additional 1K available for variables. APPLESOFT is also available as a loadable program (as opposed to ROM), in which case the APPLE requires 11K more than the PET to hold the same program.

Program Operation

The steps to load an APPLESOFT program are:

From BASIC, load the 'APPLESOFT LOADER'.

Type RUN, but don't press RETURN.

Position the APPLE tape at the beginning of the tone for the program you want. For the first program on a tape, just do a rewind. Otherwise, you will need an audio cassette-player. The person who provided the APPLE tape should be able to show you how to position a tape, since they do it all the time.

Press PLAY and wait 3 to 9 seconds.

Press RETURN.

There are several possible results. The good one is that the PET displays 'OK' and 'READY.'. Stop the tape and type 0 (zero) and RETURN. This deletes line zero, which is the last remnants of the loader. this is safe even if the APPLE program has a line zero, since only the first one is deleted. The APPLE program is now available for any required conversion. (See below)

About half the time, a question-mark will print. This is followed by a 'BRK', which puts you in the machine-language monitor on the upgrade-ROM PET. Type 'X' to return to BASIC and try again. There was a checksum error on the length block. The error was possibly caused by the tape being positioned incorrectly. If you obtain the question-mark a couple of times, try changing the '3E' (decimal 62) which is stored at hexadecimal E811 by the routine named 'INIT' to '3C' (decimal 60). The APPLE is not consistent on whether a cycle is 'low-high' or 'high-low'. Since the loader only

notices one transition per cycle, catching the wrong one gives it half of that bit, and half of this bit. Garble is the result. Fortunately, the program block always seems to be consistent with the length block. The time spent establishing which way to go is slight, since the length block ends about 11 seconds in to the tape.

You may get the message 'TOO BIG' immediately after reading the length block, which means the program won't fit into available memory.

The worst result is that the PET displays 'BAD'. This means that there was a checksum error on the program block. It is necessary to reload the loader, and perhaps to reset the PET. I didn't see this result until I had succeeded in loading about 30 APPLE programs. The APPLE tends to like tapes which are a little 'quieter' than PET tapes, so you might try getting a louder copy of the program.

Now the Fun Begins

Unfortunately, cassette tape format is not the only difference between the PET and the APPLE. After deleting line 0, you have a program loaded. You can list it, change it, or save it. But will it run? The answer is maybe. It can happen. But some programs will be hopeless. The APPLE has a very fancy graphics system, and APPLESOFT supports it. All the graphics commands are translated into CMD (which the APPLE doesn't have). If there are any of these in the program you just loaded, you may have big trouble. Perhaps the person who gave you the APPLE tape can help you convert it, but it may not be worth the effort.

There are several other BASIC commands on the APPLE which are not available on the PET. The loader translates most of these into VERIFY, which is not supported by the APPLE. There are a few APPLE commands which are very easy to convert to the PET. The loader does 'phony' translations on these. And finally, there are commands which translate exactly, but do not give the same result. The worst part of trying to correct these differences is that a line of BASIC can be 239 characters long on the APPLE, versus 80 on the PET. The longer lines will run just fine, but can not easily be changed using the PET screen editor. Thus you have to split this type of line into multiple lines.

The whole process will be greatly helped if you have an extended BASIC which includes the commands FIND and RENUMBER. This allows you to FIND commands which could cause problems, and split program lines without concern about smearing existing lines.

Space does not permit a complete tutorial on converting APPLESOFT programs. However, ignoring graphics, here are some suggestions:

Commands with no PET equivalent

DEL - To delete program lines; unlikely to be imbedded in a program, since it also stops execution.

TRACE

NOTRACE - Usage obvious. Not needed in a working program.

POP - Cancel a GOSUB. This is an atrocious technique.

HIMEM - Set top of memory. Could be replaced with POKE's but is unlikely to be in a pure BASIC program which doesn't use graphics.

LOMEM - Set bottom of memory. Within a program it will probably cause the program to fail (even on the APPLE).

ONERR

RESUME - Replace with programmed editing.

SPEED = - Sets display rate. Replace with delay loops in key locations if necessary.

& - Does a jump to a machine-language routine which the user must establish. Not part of normal BASIC programs.

NORMAL

FLASH

INVERSE - Adjusts the video mode for subsequent PRINT statements. The equivalent to INVERSE is specified within the text on the PET.

Commands with phony translation

TEXT/CONT - TEXT sets the 'text window' to be the whole screen. CONT has no function within a program, so it is substituted. A program with multiple TEXT statements probably changes the size of the 'text window' with POKE statements in order to print headings once, and then change what appears under them with PRINT statements.

HTAB/NEW - NEW has no function within a program except to make it commit suicide. HTAB is like TAB, but does not appear in a PRINT statement. HTABn can be directly converted to PRINTTAB(n-1); although it can very often be moved into an adjacent PRINT statement.

HOME/OPEN - OPEN is not supported by APPLESOFT. HOME clears the 'text window', so it can usually be replaced with PRINT"clr".

VTAB/CLOSE - CLOSE is not supported by APPLESOFT. VTABn positions the cursor on line 'n'. Programs which use VTAB usually have lots of them, so at the start of the program define a string, for example DN$, containing a 'HOME' character followed by 24 'DOWN's. Then replace VTABn with PRINT LEFT$(DN$,n);

STORE/SAVE

RECALL/LOAD - It is assumed that you won't be converting programs with LOAD or SAVE in them. STORE and RECALL are used to dump matrices out to tape and read them back. Convert by putting in the appropriate OPEN, PRINT#, CLOSE or OPEN, INPUT#, CLOSE loops.

Commands which may give different results

PR#/PRINT# - Used to do I-O to devices other than the screen and keyboard. Definitely not equivalent.

CALL/SYS - Used to invoke a machine-language program. Almost certainly will require change. Note that CALL, WAIT, PEEK, and POKE on the APPLE may specify negative numbers. The address used will be 65536 minus the amount specified. This convention is a carryover from integer-BASIC, and has no equivalent on the PET. The most popular CALL's on the APPLE are:
-936 - clear the text window. Replace by printing a screen-clear.
-958 - clear the text window from the current print position. More difficult to replace.
-868 - clear from the current print position to the end of the line.

WAIT - Wait for an external event. Will require rework, since it references an actual memory location.

POKE - Sets a specific memory location to a particular value. Usually will require substantial rework.

PEEK - Returns the value stored in a specific memory location. Will also require rework.

USR - Another way to invoke a machine-language routine.

RND - On the APPLE, RND(0) repeates the previous RND, unlike the PET, where it generates a truly random number.

GET - On the APPLE, this waits for a key to be pressed. On the PET, a null string is returned if no key has been pressed. To convert, make sure it is on a line by itself, and add a test like this: nnnn GET A$: IF A$ = "" THEN nnnn
In the APPLE program there may be a PEEK at location -16384 to see if a key is being pressed which can be combined with the GET.

= - (Horrors. If you can't trust ' = ', what can you trust!) If the result of a comparison is used as a number, it will give a different result. For example, N = A = B sets N to a value depending on whether A equals B. On the APPLE, an equal condition gives a value of 1, on the PET, equal gives -1.

ASC - Usually ASC of a letter is 64 greater on the APPLE than on the PET.

LIST - Terminates program execution on the PET, but not on the APPLE.

INPUT - APPLESOFT allows INPUT of a null string. You may encounter programs which invite you to 'PRESS RETURN TO CONTINUE'. On the PET, of course, you will obtain the 'READY.' prompt and you are out of the program. Change the prompt to 'PRESS A KEY TO CONTINUE', and replace the INPUT with a GET.
- INPUT generates a question-mark prompt on the PET but not on the APPLE.

BELL - On the APPLE, you can make the speaker beep by printing a control-G. No character appears on the screen. On the PET it prints as a reverse-G.

TAB - Use one position less on the PET.

PRINT - There are a number of detail differences. For example, tab-fields (invoked with commas) are 10 characters wide on the PET versus a sequence of 16,16,8,16,16,8. . . on the APPLE. A number is preceded by a space and followed by a skip on the PET, but not on the APPLE.

The Bottom Line

Does it work? It sure does! As long as you avoid graphics, you can have a program up and running in short order. I was able to load one Adventure-style game and have it completely running in less than half an hour. It sure beat keying in 16K of program text.

Many thanks must go to Keith Falkner of Toronto, who provided the description of what an APPLE tape looks like, many tapes to test with, and access to the manuals describing APPLESOFT.

Program Availability

If you wish to obtain the program on tape, please write me. Enclose $10, and I will send you the loader in Upgrade-ROM and Original-ROM versions, as well as the source in a format suitable for Carl Moser's ASSM/TED. For 32K PETS, this will be the whole program, for 16K there is no room for comments.

```
                                .LS
               ;
               ; APPLESOFT LOADER
               ;
               ; FOR USE ON THE COMMODORE PET/CBM
               ;
               ;    COPYRIGHT (C) 1980
               ;    GORD CAMPBELL
               ;    36 DOUBLETREE ROAD
               ;    WILLOWDALE, ONTARIO
               ;                M2J 3Z4
               ;
               ; TO ASSEMBLE USING CARL MOSER'S
               ; ASSM/TED, REQUIRES 'SET' COMMAND
               ; AND A 32K MACHINE, SINCE THE SOURCE
               ; (INCLUDING COMMENTS) IS TOO LARGE
                 TO FIT
               ; INTO DEFAULT AREA, AND OBJECT
               ; GOES INTO THE DEFAULT TEXT AREA.
               ;
               WHERE      .DE 1
               ; USED FOR STORE INDIRECT
               ; THE ONLY PART OF PAGE ZERO
               ; WHICH IS SMEARED. IT DOESN'T
               ; MATTER, BECAUSE THE 'USR'
               ; VECTOR SHOULD BE SET UP BY
               ; ANY PROGRAM WHICH USES IT.
               PGMEN      .DE $2A
               ; BASIC 'END OF PROGRAM'
               ;
               ; CHANGE THIS TO $7C AND YOU
               ; ARE CONVERTED TO ORIGINAL ROM.
               ;
               PRINT      .DE $FFD2
               ; PRINT ROUTINE
                          .BA $0400
                          .OS
               ; HERE IS A BASIC PROGRAM.
0400- 00 0D 04            .BY 0 $0D 4 0 0 $9E
0403- 00 00 9E
0406- 31 30 35            .BY '1056:' $80
0409- 36 3A 80
040C- 00 00 00            .BY 0 0 0 0
040F- 00
               ;
               ; IT READS '0 SYS1056:END'
               ;
               ;
               ; -- VARIABLES --
               ;
0410- 00 00 00 LENGTH     .BY 00 00 00 00
0413- 00
```

```
                ; APPLESOFT 'LENGTH' BLOCK
                ; IS STORED HERE
0414- 00 00     STLEN       .SI 0
                ; LENGTH OF CURRENT BLOCK
0416- 00 00     STLOC       .SI 0
                ; WHERE IT GOES
0418- 00        CHAR        .BY 0
                ; CURRENT CHARACTER
0419- 00        MODE        .BY 0
                ; WHICH ACTIVITY NOW:
                ;   0 - SYNCHRONIZING
                ;   1 - LEADER
                ;   2 - DATA
041A- 00        BLOCK       .BY 0
                ; WHICH BLOCK:
                ;   0 - LENGTH BLOCK
                ;   1 - PROGRAM BLOCK
041B- 42 41 44  BAD         .BY 'BAD'
041E- 4F 4B     OK          .BY 'OK'
                ; CHECKSUM MESSAGES
                ;
                ; *** ENTRY POINT ***
                ;
                ;   MUST BE AT $0420
                ;   FOR THE 'BASIC' PROGRAM
                ;
0420- 4C 50 04              JMP INIT
                ; SKIP PAST CODE WHICH MOVES
                ; THE PROGRAM DOWN FROM $0801
                ; TO $0501. THIS CODE IS NEEDED
                ; BECAUSE WHEN LINE ZERO (THE
                ; PHONY BASIC PROGRAM) IS DELETED
                ; 'END OF PROGRAM' ETC ARE ONLY
                ; ADJUSTED BY ONE PAGE MAXIMUM.
                ;
                ; MOVE PROGRAM DOWN 3 PAGES
                ;
0423- A9 08     MOVE        LDA #8
0425- 85 02                 STA *WHERE+1
0427- A9 05                 LDA #5
0429- 85 2B                 STA *PGMEN+1
042B- A0 00                 LDY #0
042D- 84 01                 STY *WHERE
042F- 84 2A                 STY *PGMEN
0431- B1 01     MOVLP       LDA (WHERE),Y
0433- 91 2A                 STA (PGMEN),Y
0435- E6 2A                 INC *PGMEN
0437- D0 02                 BNE MOVOK
0439- E6 2B                 INC *PGMEN+1
043B- A5 2A     MOVOK       LDA *PGMEN
043D- C5 2C                 CMP *PGMEN+2
043F- D0 07                 BNE INWHERE
0441- A5 2B                 LDA *PGMEN+1
0443- C5 2D                 CMP *PGMEN+3
0445- D0 01                 BNE INWHERE
0447- 60                    RTS   ; FINISHED
0448- E6 01     INWHERE     INC *WHERE
044A- D0 E5                 BNE MOVLP
044C- E6 02                 INC *WHERE+1
044E- D0 E1                 BNE MOVLP
                ; ** ALWAYS GOES **
                ;
                ; INITIALIZATION
                ;
                ; SET UP POINTERS ETC ON ENTRY
                ; SO IF WE HAD A BAD LOAD, WE
                ; CAN TRY AGAIN BY ENTERING 'RUN'
                ;
0450- A9 04     INIT        LDA #4
0452- 8D 14 04              STA STLEN
0455- 8D 17 04              STA STLOC+1
0458- A9 10                 LDA #$10
045A- 8D 16 04              STA STLOC
045D- A9 00                 LDA #0
045F- 8D 15 04              STA STLEN+1
0462- 8D 19 04              STA MODE
0465- 8D 1A 04              STA BLOCK
0468- 78                    SEI
                ; DISABLE INTERRUPTS
0469- AD 10 E8              LDA $E810
                ; CLEAR 6520
046C- A9 3E                 LDA #$3E
046E- 8D 11 E8              STA $E811
                ; MAKE 6520 RESPOND TO
                ;  LOW TO HIGH TRANSITION
                ;
                ; FOR SOME TAPES THE '3E'
                ; ABOVE MUST READ '3C'
                ; (IE. HIGH TO LOW TRANSITION)
                ;
0471- AD 16 04              LDA STLOC
0474- 85 01                 STA *WHERE
0476- AD 17 04              LDA STLOC+1
0479- 85 02                 STA *WHERE+1
                ;
                ; END OF INITIALIZATION
                ;
047B- A0 08     INITY       LDY #8
047D- A2 00     INITX       LDX #0
047F- E8        COUNT       INX
                ; COUNT HOW MANY TIMES
                ; THROUGH THE LOOP
0480- 2C 11 E8              BIT $E811
                ; HAVE WE A TRANSITION YET?
0483- 10 FA                 BPL COUNT
                ; BRANCH BACK IF NOT YET
0485- AD 10 E8              LDA $E810
                ; RESET THE 6520
0488- AD 19 04              LDA MODE
                ; WHAT WERE WE DOING?
048B- F0 2C                 BEQ STARTUP
048D- C9 01                 CMP #1
048F- F0 2D                 BEQ STARTBIT
                ; REAL DATA NOW
0491- E0 40                 CPX #$40
0493- 30 03                 BMI ZEROBIT
0495- 38                    SEC
0496- B0 01                 BCS SETBIT
                ; ** ALWAYS GOES **
0498- 18        ZEROBIT     CLC
                ; THE CARRY BIT NOW INDICATES
                ; WHETHER WE GOT A ZERO OR ONE
0499- 2E 18 04  SETBIT      ROL CHAR
                ; ROTATE IT INTO THE CHARACTER
049C- 88                    DEY
                ; FINISHED THIS CHARACTER?
049D- D0 DE                 BNE INITX     ; NO
049F- AD 18 04              LDA CHAR
04A2- 91 01                 STA (WHERE),Y
                ; STORE THE CHARACTER
04A4- CE 14 04              DEC STLEN
                ; REDUCE CHARACTER COUNT
04A7- D0 08                 BNE NEXTCHAR
04A9- AD 15 04              LDA STLEN+1
                ; FINISHED THIS BLOCK?
04AC- F0 19                 BEQ FINMODE
04AE- CE 15 04              DEC STLEN+1
04B1- E6 01     NEXTCHAR    INC *WHERE
                ; INCREMENT DATA POINTER
04B3- D0 C6                 BNE INITY
04B5- E6 02                 INC *WHERE+1
04B7- D0 C2                 BNE INITY
                ;   ** ALWAYS GOES **
04B9- EE 19 04  STARTUP     INC MODE
                ; THROW AWAY FIRST TRANSITION
04BC- D0 BF                 BNE INITX
                ;   ** ALWAYS GOES **
04BE- E0 40     STARTBIT    CPX #$40
                ; IS IT A START BIT?
04C0- 10 BB                 BPL INITX     ; NO
04C2- EE 19 04              INC MODE
04C5- D0 B6                 BNE INITX
                ; ** ALWAYS GOES **
04C7- AD 1A 04  FINMODE     LDA BLOCK
                ; WE JUST LOADED A BLOCK.
                ; WHICH ONE WAS IT?
04CA- D0 62                 BNE LOADED
04CC- A9 FF                 LDA #$FF
04CE- 4D 10 04              EOR LENGTH
                ; CHECKSUM ON LENGTH BLOCK
04D1- 4D 11 04              EOR LENGTH+1
04D4- 4D 12 04              EOR LENGTH+2
04D7- CD 13 04              CMP LENGTH+3
04DA- F0 07                 BEQ NEXTBLK
04DC- A9 3F                 LDA #$3F
                ; BAD LOAD: PRINT QUESTION MARK
                ; AND QUIT WITH A 'BREAK'
04DE- 20 D2 FF              JSR PRINT
04E1- 58                    CLI
                ; QUIT NOW
04E2- 00                    BRK
04E3- AD 10 04  NEXTBLK     LDA LENGTH
                ; INITIALIZATION FOR PROGRAM LOAD
04E6- 8D 14 04              STA STLEN
04E9- AD 11 04              LDA LENGTH+1
04EC- 8D 15 04              STA STLEN+1
04EF- EE 14 04              INC STLEN
                ; LOAD CHECKSUM TOO
                ; MUST GO TWO BYTES PAST
                ; THE END OF THE ACTUAL PROGRAM
```

```
                  2270 ;
04F2- D0 03       2280              BNE LEN1
04F4- EE 15 04    2290              INC STLEN+1
04F7- EE 14 04    2300 LEN1         INC STLEN
04FA- D0 03       2310              BNE LENOK
04FC- EE 15 04    2320              INC STLEN+1
04FF- A9 08       2330 LENOK        LDA #$08
                  2340 ;
                  2350 ; ALWAYS LOAD AT $0801
                  2360 ; IF IT'S CASSETTE APPLESOFT
                  2370 ; CONVERT IT LATER
                  2380 ;
0501- 85 02       2390              STA *WHERE+1
0503- A9 01       2400              LDA #$01
0505- 85 01       2410              STA *WHERE
0507- A5 35       2420              LDA *PGMEN+11
0509- CD 15 04    2430              CMP STLEN+1
050C- D0 13       2440              BNE DIFFPAGE
                  2450 ; CHECKING ON WHETHER THERE IS
                  2460 ; ENOUGH MEMORY.
050E- AD 14 04    2470              LDA STLEN
0511- C5 34       2480              CMP *PGMEN+10
0513- 90 0E       2490              BCC MEMOK
0515- A2 06       2500 MEMBAD       LDX #6
0517- BD 5F 06    2510 MEMCHR       LDA TOOBIG,X
051A- 20 D2 FF    2520              JSR PRINT
051D- CA          2530              DEX
051E- 10 F7       2540              BPL MEMCHR
                  2550 ; MESSAGE DISPLAYED: QUIT NOW
0520- 60          2560              RTS
0521- 90 F2       2570 DIFFPAGE     BCC MEMBAD
0523- A9 00       2580 MEMOK        LDA #$00
0525- 8D 19 04    2590              STA MODE
0528- EE 1A 04    2600              INC BLOCK
052B- 4C 7B 04    2610              JMP INITY
052E- 58          2620 LOADED       CLI
                  2630 ; ALLOW INTERRUPTS NOW
052F- A5 01       2640              LDA *WHERE
                  2650 ;SET HIGH ADDRESS
0531- 8D 16 04    2660              STA STLOC
0534- A5 02       2670              LDA *WHERE+1
0536- 8D 17 04    2680              STA STLOC+1
                  2690 ; INITIALIZATION FOR CHECKSUM
                  2700 ; AND PROGRAM LINKAGE
                  2710 ;
0539- A9 08       2720              LDA #8
053B- 85 02       2730              STA *WHERE+1
053D- A9 05       2740              LDA #5
053F- 8D 02 04    2750              STA $0402
0542- A9 01       2760              LDA #1
0544- 85 01       2770              STA *WHERE
0546- 8D 01 04    2780              STA $0401
0549- 8D 19 04    2790              STA MODE
                  2800 ;NOW USE 'MODE' AS QUOTE-MODE FLAG
                  2810 ; VALUES ARE:
                  2820 ;    0 - CURRENTLY INSIDE QUOTES
                  2830 ;    1 - NOT IN QUOTES
                  2840 ;
054C- A9 FF       2850              LDA #$FF
054E- A0 00       2860              LDY #0
0550- 51 01       2870 CHKLOOP      EOR (WHERE),Y
                  2880 ; CHECKSUM CALCULATION
0552- E6 01       2890              INC *WHERE
0554- D0 02       2900              BNE CHKEND
0556- E6 02       2910              INC *WHERE+1
0558- A6 01       2920 CHKEND       LDX *WHERE
055A- EC 16 04    2930              CPX STLOC
055D- D0 F1       2940              BNE CHKLOOP
055F- A6 02       2950              LDX *WHERE+1
0561- EC 17 04    2960              CPX STLOC+1
0564- D0 EA       2970              BNE CHKLOOP
0566- D1 01       2980              CMP (WHERE),Y
0568- F0 15       2990              BEQ CHKOK
056A- AD 1B 04    3000              LDA BAD
                  3010 ; PRINT 'BAD'
056D- 20 D2 FF    3020              JSR PRINT
0570- AD 1C 04    3030              LDA BAD+1
0573- 20 D2 FF    3040              JSR PRINT
0576- AD 1D 04    3050              LDA BAD+2
0579- 20 D2 FF    3060              JSR PRINT
                  3070 ; DO THE REST ANYWAY
057C- 4C 8B 05    3080              JMP CASSREL
057F- AD 1E 04    3090 CHKOK        LDA OK
                  3100 ; PRINT 'OK'
0582- 20 D2 FF    3110              JSR PRINT
0585- AD 1F 04    3120              LDA OK+1
0588- 20 D2 FF    3130              JSR PRINT
058B- A9 01       3140 CASSREL      LDA #1
058D- 85 01       3150              STA *WHERE
058F- A9 08       3160              LDA #8
0591- 85 02       3170              STA *WHERE+1
0593- AD 02 08    3180              LDA $0802
0596- C9 08       3190              CMP #$08
0598- F0 16       3200              BEQ TRANS
059A- A0 01       3210 CASSLP       LDY #1
                  3220 ; IT'S CASSETTE APPLESOFT
                  3230 ; ORIGINAL ADDRESS WAS $3001
059C- B1 01       3240              LDA (WHERE),Y
059E- F0 EB       3250              BEQ CASSREL
                  3260 ; ON THE SECOND PASS, IT LOOKS
                  3270 ; LIKE ROM APPLESOFT
05A0- 38          3280              SEC
05A1- E9 28       3290              SBC #$28
05A3- 91 01       3300              STA (WHERE),Y
05A5- AA          3310              TAX
05A6- 88          3320              DEY
05A7- B1 01       3330              LDA (WHERE),Y
05A9- 85 01       3340              STA *WHERE
05AB- 86 02       3350              STX *WHERE+1
05AD- 4C 9A 05    3360              JMP CASSLP
05B0- A0 00       3370 TRANS        LDY #0
05B2- B1 01       3380              LDA (WHERE),Y
05B4- AA          3390              TAX
05B5- D0 08       3400              BNE NOTEN
05B7- C8          3410              INY
                  3420 ; LAST LINE OF TOKENS DONE?
05B8- B1 01       3430              LDA (WHERE),Y
05BA- D0 03       3440              BNE NOTEN
05BC- 4C 12 06    3450              JMP TOKDONE
05BF- A0 01       3460 NOTEN        LDY #1
05C1- B1 01       3470              LDA (WHERE),Y
05C3- 8E 16 04    3480              STX STLOC
                  3490 ; SET END OF CURRENT LINE
05C6- 8D 17 04    3500              STA STLOC+1
05C9- A0 04       3510              LDY #4
05CB- E6 01       3520 TOTXT        INC *WHERE
                  3530 ; STEP PAST POINTER
                  3540 ; AND LINE NUMBER
05CD- D0 02       3550              BNE WHOK
05CF- E6 02       3560              INC *WHERE+1
05D1- 88          3570 WHOK         DEY
05D2- D0 F7       3580              BNE TOTXT
05D4- B1 01       3590 TRLOOP       LDA (WHERE),Y
05D6- C9 22       3600              CMP #$22
                  3610 ; IS IT A QUOTE?
05D8- D0 0F       3620              BNE NOQ
05DA- AD 19 04    3630              LDA MODE
05DD- F0 05       3640              BEQ MODEON
05DF- CE 19 04    3650              DEC MODE
05E2- F0 12       3660              BEQ NXTCHAR
                  3670 ; ** ALWAYS GOES **
05E4- EE 19 04    3680 MODEON       INC MODE
05E7- D0 0D       3690              BNE NXTCHAR
                  3700 ; ** ALWAYS GOES **
05E9- AE 19 04    3710 NOQ          LDX MODE
05EC- F0 08       3720              BEQ NXTCHAR
                  3730 ; BRANCH IF WE ARE IN QUOTES
05EE- AA          3740              TAX
05EF- 10 05       3750              BPL NXTCHAR
                  3760 ; ONLY TRANSLATE TOKENS
05F1- BD 00 07    3770              LDA $0700,X
                  3780 ; TRANSLATE FROM TABLE
05F4- 91 01       3790              STA (WHERE),Y
05F6- E6 01       3800 NXTCHAR      INC *WHERE
05F8- D0 02       3810              BNE WHEOK
05FA- E6 02       3820              INC *WHERE+1
05FC- A5 01       3830 WHEOK        LDA *WHERE
05FE- CD 16 04    3840              CMP STLOC
                  3850 ; HAVE WE FINISHED THIS LINE?
0601- D0 D1       3860              BNE TRLOOP
0603- A5 02       3870              LDA *WHERE+1
0605- CD 17 04    3880              CMP STLOC+1
0608- D0 CA       3890              BNE TRLOOP
060A- A9 01       3900              LDA #1
060C- 8D 19 04    3910              STA MODE
                  3920 ; RESET QUOTE MODE FLAG
060F- 4C B0 05    3930              JMP TRANS
                  3940 ;
                  3950 ; FINISHED TOKEN TRANSLATION
0612- EE 16 04    3960 TOKDONE      INC STLOC
                  3970 ; INCLUDE THE '00 00' (END OF
                  3980 ; PROGRAM) IN THE LENGTH
                  3990 ;
0615- D0 03       4000              BNE MORLOC
0617- EE 17 04    4010              INC STLOC+1
061A- EE 16 04    4020 MORLOC       INC STLOC
061D- D0 03       4030              BNE LOCDON
061F- EE 17 04    4040              INC STLOC+1
0622- AD 16 04    4050 LOCDON       LDA STLOC
0625- 85 2C       4060              STA *PGMEN+2
```

```
0627- 85 2E                     STA *PGMEN+4
0629- AD 17 04                  LDA STLOC+1
062C- 38                        SEC
062D- E9 03                     SBC #$03
062F- 85 2D                     STA *PGMEN+3
0631- 85 2F                     STA *PGMEN+5
                 ;
                 ; SET UP PROGRAM LINKS FOR
                 ; MOVE FROM $0801 TO $0501
                 ;
0633- A9 01                     LDA #1
0635- 85 01                     STA *WHERE
0637- A9 08                     LDA #8
0639- 85 02                     STA *WHERE+1
063B- A0 00      RELLP          LDY #0
063D- B1 01                     LDA (WHERE),Y
063F- 8D 14 04                  STA STLEN
0642- C8                        INY
0643- B1 01                     LDA (WHERE),Y
0645- F0 15                     BEQ RELDONE
0647- 8D 15 04                  STA STLEN+1
064A- 38                        SEC
064B- E9 03                     SBC #3
064D- 91 01                     STA (WHERE),Y
064F- AD 14 04                  LDA STLEN
0652- 85 01                     STA *WHERE
0654- AD 15 04                  LDA STLEN+1
0657- 85 02                     STA *WHERE+1
0659- 4C 3B 06                  JMP RELLP
065C- 4C 23 04   RELDONE        JMP MOVE
065F- 47 49 42   TOOBIG         .BY 'GIB OOT'
0662- 20 4F 4F
0665- 54
                 ; MESSAGE 'TOO BIG' REVERSED
                                .BA $0780
                 ;
                 ; **** TOKEN TRANSLATION TABLE ***
                 ;
                 ; (SEE FOOTNOTES BELOW)
                 ;
0780- 80               .BY $80       ; END
0781- 81               .BY $81       ; FOR
0782- 82               .BY $82       ; NEXT
0783- 83               .BY $83       ; DATA
0784- 85               .BY $85       ; INPUT
0785- 95               .BY $95       ; *DEL
0786- 86               .BY $86       ; DIM
0787- 87               .BY $87       ; READ
0788- 9D               .BY $9D       ; *GR
0789- 9A               .BY $9A       ; *TEXT/CONT **
078A- 98               .BY $98       ; PR#/PRINT# *
078B- 84               .BY $84       ; IN#/INPUT# *
078C- 9E               .BY $9E       ; CALL/SYS   *
078D- 9D               .BY $9D       ; *PLOT
078E- 9D               .BY $9D       ; *HLIN
078F- 9D               .BY $9D       ; *VLIN
0790- 9D               .BY $9D       ; *HGR2
0791- 9D               .BY $9D       ; *HGR
0792- 9D               .BY $9D       ; *HCOLOR=
0793- 9D               .BY $9D       ; *HPLOT
0794- 9D               .BY $9D       ; *DRAW
0795- 9D               .BY $9D       ; *XDRAW
0796- A3               .BY $A3       ; *HTAB/NEW  **
0797- 9F               .BY $9F       ; *HOME/OPEN **
0798- 9D               .BY $9D       ; *ROT=
0799- 9D               .BY $9D       ; *SCALE=
079A- 9D               .BY $9D       ; *SHLOAD
079B- 95               .BY $95       ; *TRACE
079C- 95               .BY $95       ; *NOTRACE
079D- 95               .BY $95       ; *NORMAL
079E- 95               .BY $95       ; *INVERSE
079F- 95               .BY $95       ; *FLASH
07A0- 9D               .BY $9D       ; *COLOR=
07A1- 95               .BY $95       ; *POP
07A2- A0               .BY $A0       ; *VTAB/CLOSE **
07A3- 95               .BY $95       ; *HIMEM
07A4- 95               .BY $95       ; *LOMEM
07A5- 95               .BY $95       ; *ONERR
07A6- 95               .BY $95       ; *RESUME
07A7- 93               .BY $93       ; *RECALL/LOAD **
07A8- 94               .BY $94       ; *STORE/SAVE  **
07A9- 95               .BY $95       ; *SPEED=
07AA- 88               .BY $88       ; LET
07AB- 89               .BY $89       ; GOTO
07AC- 8A               .BY $8A       ; RUN
07AD- 8B               .BY $8B       ; IF
07AE- 8C               .BY $8C       ; RESTORE
07AF- 95               .BY $95       ; *&
07B0- 8D               .BY $8D       ; GOSUB
07B1- 8E               .BY $8E       ; RETURN
07B2- 8F               .BY $8F       ; REM
07B3- 90               .BY $90       ; STOP
07B4- 91               .BY $91       ; ON
07B5- 92               .BY $92       ; WAIT *
07B6- 93               .BY $93       ; LOAD
07B7- 94               .BY $94       ; SAVE
07B8- 96               .BY $96       ; DEF
07B9- 97               .BY $97       ; POKE *
07BA- 99               .BY $99       ; PRINT
07BB- 9A               .BY $9A       ; CONT
07BC- 9B               .BY $9B       ; LIST
07BD- 9C               .BY $9C       ; CLEAR
07BE- A1               .BY $A1       ; GET    *
07BF- A2               .BY $A2       ; NEW
07C0- A3               .BY $A3       ; TAB( *
07C1- A4               .BY $A4       ; TO
07C2- A5               .BY $A5       ; FN
07C3- A6               .BY $A6       ; SPC(
07C4- A7               .BY $A7       ; THEN
07C5- 9D               .BY $9D       ; *AT
07C6- A8               .BY $A8       ; NOT
07C7- A9               .BY $A9       ; STEP
07C8- AA               .BY $AA       ; +
07C9- AB               .BY $AB       ; -
07CA- AC               .BY $AC       ; * (TIMES)
07CB- AD               .BY $AD       ; /
07CC- AE               .BY $AE       ; ^ (EXPONENTIATION)
07CD- AF               .BY $AF       ; AND
07CE- B0               .BY $B0       ; OR
07CF- B1               .BY $B1       ; >
07D0- B2               .BY $B2       ; =
07D1- B3               .BY $B3       ; <
07D2- B4               .BY $B4       ; SGN
07D3- B5               .BY $B5       ; INT
07D4- B6               .BY $B6       ; ABS
07D5- B7               .BY $B7       ; USR *
07D6- B8               .BY $B8       ; FRE
07D7- 9D               .BY $9D       ; *SCRN(
07D8- 9D               .BY $9D       ; *PDL
07D9- B9               .BY $B9       ; POS
07DA- BA               .BY $BA       ; SQR
07DB- BB               .BY $BB       ; RND *
07DC- BC               .BY $BC       ; LOG
07DD- BD               .BY $BD       ; EXP
07DE- BE               .BY $BE       ; COS
07DF- BF               .BY $BF       ; SIN
07E0- C0               .BY $C0       ; TAN
07E1- C1               .BY $C1       ; ATN
07E2- C2               .BY $C2       ; PEEK *
07E3- C3               .BY $C3       ; LEN
07E4- C4               .BY $C4       ; STR$
07E5- C5               .BY $C5       ; VAL
07E6- C6               .BY $C6       ; ASC
07E7- C7               .BY $C7       ; CHR$
07E8- C8               .BY $C8       ; LEFT$
07E9- C9               .BY $C9       ; RIGHT$
07EA- CA               .BY $CA       ; MID$
                 ;
                 ; REMAINDER NOT IMPLEMENTED
                 ; SUBSTITUTE 'REM'
                 ;
07EB- 8F 8F 8F         .BY $8F $8F $8F $8F $8F $8F
07EE- 8F 8F 8F
07F1- 8F 8F 8F         .BY $8F $8F $8F $8F $8F $8F
07F4- 8F 8F 8F
07F7- 8F 8F 8F         .BY $8F $8F $8F $8F $8F $8F
07FA- 8F 8F 8F
07FD- 8F 8F 8F         .BY $8F $8F $8F
                 ;
                 ; COMMANDS WHICH ARE PRECEEDED BY
                 ; AN ASTERISK ABOVE ARE NOT
                 ; IMPLEMENTED ON THE PET. THE ONES
                 ; WHICH DEPEND ON APPLE HARDWARE
                 ; (GRAPHICS AND PDL) ARE TRANSLATED
                 ; INTO 'CMD', THE OTHERS INTO
                 ; 'VERIFY'
                 ;
                 ; COMMANDS WITH AN ASTERISK TO
                 ; THE RIGHT MAY TRANSLATE BADLY.
                 ;
                 ; COMMANDS WITH TWO ASTERISKS ARE
                 ; PHONY TRANSLATIONS FOR MANUAL
                 ; CONVERSION.
                 ;
                 ; SEE ARTICLE FOR DETAILS.
                 ;
                 ; END OF PHONY BASIC PROGRAM
0800- 00 00 00              .BY 0 0 0
                            .EN
```

APPLESOFT LOADER - TAPE CONTENTS

File 1. "APP LOAD SOURCE"

- The source for the program in the format used by Carl Moser's ASSM/TED.
- Requires a 'SET' command due to size.
- SET $4100 $6FFF will leave some room.

File 2. "APP LOAD SOURCE" - second copy

FILE 3. "APP LOAD OBJ"

- The object program.
- Can be LOADed and SAVEd from BASIC. (ie. doesn't require machine-language monitor)
- The cassette must be moving before you type RUN.

File 4. "APP LOAD OBJ" - second copy

File 5. "APP LOAD IMAGE"

- Memory-image of source program saved using machine-language monitor
- Resides in $4100 to $6800
- Allows PRINT and ASSEMBLE to function with disk version of ASSM/TED. (I think that's all that will work). Use the following sequence of commands:

SYS 4 : - get into machine-language monitor
.L "APP LOAD IMAGE" (load the source)
(load ASSM/TED)
.G 2000 - invoke the assembler
SET $4100 $6800
HARD SET
ASSEMBLE

File 6. "APP LOAD IMAGE" - second copy

©

Capute

Wherein we acknowledge recent goofs. . .

This page brought to you by Robert Lock, Editor/Publisher and our (sometimes hostile) but always active readers.

Corrections for Larry's Atari Article on Input/Output

Here are some corrections to my article in the July/August COMPUTE on Input/Output on the ATARI. First of all, the listings are numbered incorrectly. Listings 1 through 3 are numbered ok. Listing 7 should be Listing 4. And, Listings 4 through 6 should be Listings 5 through 7, respectively.

In the text concerning the XIO9 and XIO5 commands, references are made to an EOF character. These should be EOL characters instead. Also in this section, just under Listing 5, the paragraph on the XIO5 command makes comparisons to the PRINT command several times. These comparisions should be to the INPUT command instead. Thanks to the folks at *Iridis* for pointing out these last corrections. ©

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. ©

Advertiser's Index

COMPUTE.
The Journal For Progressive Computing.™

My computer is: PET _____ APPLE _____ ATARI _____ OSI _____ SYM _____ KIM _____
AIM _____ Other _____ (Specify) _____

☐ Please enter my 1 year (12 issue) subscription to COMPUTE.
☐ New subscription
☐ Renewal subscription

_____ $16.00 (U.S. Mail Order Subscription)
_____ $18.00 U.S. Funds (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, N.C. 27403 919-272-4867

COMPUTE.
The Journal For Progressive Computing.™

My computer is: PET _____ APPLE _____ ATARI _____ OSI _____ SYM _____ KIM _____
AIM _____ Other _____ (Specify) _____

☐ Please enter my 1 year (12 issue) subscription to COMPUTE.
☐ New subscription
☐ Renewal subscription

_____ $16.00 (U.S. Mail Order Subscription)
_____ $18.00 U.S. Funds (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, N.C. 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:

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 Technology Unlimited:

Please rush me the Fall 1979 catalog of 6502 expansion and application products.

I have the following computer(s):

Old PET____, New PET____, AIM-65____, KIM-1____, SYM-1____

Thank you for expediting.

Sincerely,

____

NAME: ______________________________

STREET: ______________________________

CITY/STATE/ZIP: ______________________________

COMPUTE 80

Dear MICRO-ED:
I would like further information on your educational programs

Name ______________________________

Address ______________________________

City ________________ State ________________ Zip ____

Skyles Electric Works

YES, I WANT MORE INFORMATION ABOUT:
PET PERIPHERALS

☐ 8, 16, 24 K bytes of PET Memory Expansion Systems

☐ Full sized PET Keyboard with numeric pad

☐ PAL-40 (40 column), PAL-80 (80 column) PET Printers

PET PROGRAMMING AIDS

☐ BASIC PROGRAMMERS TOOLKIT

☐ MacroTeA 6502 Software Development System

Name ______________________________

Address ______________________________

City/State/Zip ______________________________

☐ *Please send name of local Skyles Electric Works dealer*

Place Postage
Stamp Here

Micro Technology Unlimited
P.O. Box 12106
2806 Hillsborough St.
Raleigh, NC 27605

PLACE
STAMP
HERE

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

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit No. 503 Cupertino, CA

Postage will be paid by addressee:

Skyles Electric Works
10301 Stonydale Drive
P. O. Box 574
Cupertino, CA 95015