THE PAPER

VOLUME IV, Issue 4 | A CSD PUBLICATION | $3.00

Table of Contents

++

Business, Correspondence, Letter Quality

It seems that the prices of dot matrix printers are continuing to drop while the features and print quality are improving. Commodore has reduced the price of its low end 4022 while offering a faster fuller featured 8023 printer. Epson prices have steadily decreased while the list of features gets larger. Friction and tractor will now be standard and Graftrax Plus is a new option. New printers to watch are the Microline series by Okidata and the Prowriters by C. Itoh. These seem to include fewer type fonts but may have some features the Epson lacks. Make sure that when you buy a printer you get a combination of low price, a long list of features and compatibilty with your hardware and software.

Prices have also been coming down on the letter quality printers but so have the list of features. The Smith Corona TP-1, Brother HR-1 and Olivetti DY211 are examples of these low priced entries. These models have decent print quality but may lack certain features and be less rugged than your application requires. C. Itoh's Printmaster F-10 is the equivalent of the Spinwriter for a little over half the cost. You won't lose any features or ruggedness when you buy this model. Still lower in price is the new Daisywriter by Computers International. It claims to emulate several established printers and to have ALL the features of a true word processing printer including the needed durabilty. Only a cable is required to connect it to your CBM system.

General Information

The PAPER is published 6 times per year by Centerbrook Software Designs at Pearl St., Livingston Manor, NY 12758. Telephone: (914) 439-3591.

New subscribers will receive all issues of the current volume. Single copy price is $4 and the subscription price is $20 for all the issues of the current volume. Subscription orders should be mailed to The PAPER, Pearl St., Livingston Manor, NY 12758

Third class postage is paid at Livingston Manor, NY 12758 (Permit #14). POSTMASTER: Mail all address changes to the address above.

Subscription Rates:

USA third class: $20/volume
Canada first class: $25/volume
Foreign surface: $30/volume
airmail: $40/volume

Payment in check or money order in US funds must accompany all orders. Only prepaid purchase orders will be accepted. All checks should be made out to The PAPER. Sorry, we cannot accept bank or credit cards.

Advertising

Advertising rates are $25 per quarter page per issue. Copy must be camera ready or there will be an additional charge. Special rates may be negotiated.

Circulation

Volume 4 Issue 3

Printed: 700 USA: 491 Canada: 34 Foreign: 26

Software

Software published in The PAPER is believed to be free of copyright restrictions. It is meant to work on the machine indicated. Many programs were originally designed to work with only one ROM set but efforts have been made to convert them to work on all present ROM releases.

Staff:

Publisher: Ralph Bressler
Editor: Doug Haluza
Assoc. Ed: Roy Busdiecker
Vic Santa Lucia

Staff writers: Bill Batcher
Jim Fowler
Jerry Key
Cindy Bressler

This issue has been prepared using the new **Superscript** word processor and a C. Itoh **Prowriter 8510AP**. An elite typeface was used with the titles of most articles expanded.

Death and Rebirth

This is the last issue of The PAPER that will be published. As of October 1 The PAPER and the Midnite Software Gazzette will be combining to form a new publication with a new name. Paid up subscribers to The PAPER will receive the first two issues of the new publication without additional payment. New subscribers will be charged according to the old PAPER schedule and advertising costs will remain the same. One big change is that issues will be sent by first class mail which should greatly improve the delivery speed. This is really a merger, not a "take over", since the staff of both publications will continue to work on the new newsletter. The Gazzette people will be primarily responsible for reviews, commentary, and short tips. Doug and I will continue to write articles and edit these for the new publication. Jim Strasma has also organized people to put the final copy together and to perform other important duties such as maintaining a mailing list and contacting advertisers. We feel that although this will reduce the number of publications for the Commodore line it will increase the quality of both publications. It will also increase the coverage of the VIC, 64, SuperPET and other machines since more people with varying interests will be involved.

I would now like to thank the many people, both subscribers and advertisers, who have supported us through an irregular publishing schedule. Our quality has been high but or quantity very low. Although I enjoy writing and responding to the many questions of people around the country, the newsletter was becoming a real drag on my time. Not only was I just breaking even but it was taking time away from my own programming and software business. The problems of administration far outweighed the problems of writing and editing. This is compounded by the fact that I am less than well organized and tend to put off the more onerous tasks. However, the worst part of the whole operation was the fact that despite all efforts we were simply unable to keep a regular and dependable publication schedule. This certainly discouraged some readers and potential subscribers but it discouraged me even more. It is strange that this merger comes just as I have begun to "get my act together". Of absolute necessity, I have begun to develop a few standards and practices which speed various operations. I hope this will carry through to my work on the new newsletter.

I would also like to thank all those people who have written articles once or on a regular basis. I hope these people will continue to write. I also owe a debt to many people who have helped edit and put together issues from proofreading copy to pasting on labels. I do not want to mention names since the list is too long and I am sure I would forget several people. Many issues were composed at American Peripherals, a Commodore dealer on Long Island, who has been very helpful in allowing us to use his facilities and equipment and who has supported us with ads and loans when we needed them. Lastly, I would like to thank my wife, Cindy, who has put up with my late hours, constant talk about computers and who has done much proofreading.

Well, that's enough of the "death" part. Let's not forget that a new, bigger, and better publication is in the works. Jim Strasma deserves the credit for getting the ball rolling and contacting me first. Without his organization and suggestion that we merge I might have let everything drop at the end of volume 4. He and his group are taking on the major tasks but I feel I will also have a part to play. Not having to administer mailing lists and bill advertisers frees me for more writing and editing and some programming. I think I'll need this time since a bigger, better newsletter will have more articles to edit and more authors ready to contribute. I am looking forward to an excellent Commodore-only magazine to help beginner and experienced programmer alike. We'll need this kind of publication as Commodore continues its push to become the leader in all fields of microcomputer use.

Reader I/O

The letters and notes presented here are in no special order. The responses are usually based solely on the knowledge of the publisher and/or editor. One of the problems of having such a small staff is that we cannot possibly know everything or have tried every piece of software and hardware. We try to research problems as much as we can. This is why we rely so heavily on the input of the readers.

On page 33 of the Nov/Dec issue in the article "Techniques for Better BASIC", you have:

```
9030 IF I$=N$ THEN RETURN
9040 NEXT S
9050 RETURN
```

Enough runs of 9030 will cause an ?OUT OF MEMORY ERROR as the stack pointer does not get reset in the FOR-NEXT loop. A preferable way is:

```
9030 IF I$=N$ THEN S=NI+2
```

This sets the index variable (S) to a value greater than the upper limit of the loop (NI) and:

(1) forces you out of the loop
(2) allows S to be used as a flag when you RETURN
(3) properly closes the FOR...NEXT loop, avoiding OUT OF MEM

Line 3040 would then be IF S>NI+1 THEN ?"...." . The >NI+1 is, of course, necessary because the NEXT ends with a value 1 greater than the upper limit of the loop. - Glenn Fisher

Glenn - I have heard this arguement many times and it is true. Apparently the key phrase is "Enough runs of...". I tried a similar situation and it ran more than 1000 before I got tired of watching. When I inspected the stack pointer it did indeed show that the stack was filling. However, I remain unconvinced that this technique alone will cause problems. - Ralph

I am thinking of buying a Commodore SuperPet but haven't decided yet. Commodore seems to have a bad name in these parts! Any advice would be welcome. - Al Thompson, 14 Power Ave, Shrewsbury, MA 01545

Al - As I indicated in our phone conversation, I think that the Commodore products are the best value around and that there is a product for almost every need. Right now I have a PET 2001 with BASIC 4.0 and 32K RAM, a 4022 printer, 4040 disk, SuperPET, 8050 disk, VIC-20, Commodore-64 and 1541 disk, so my money is where my writing is. I consider the dual personality of the SuperPET one of the best ideas around. I can word process and data manage in the 8032 mode and then switch to learning some APL. Buy one by all means! - Ralph

First let me say that we are most delighted with The PAPER and look forward to the next issue. It is so informative. We feel that Commodore's "Education Marketing Resources" for $25 is an excellent purchase and very helpful to educators. This 3" thick looseleaf has information about all types of software plus 4 disks full of public domain educational software. However, overall, we totally agree with you. It is so frustrating to see most of the popular computer magazines pushing nothing but Apple and TRS-80. This makes it difficult to sell Commodore although we think they have far superior hardware. - Jayne Schiek, 34 West Side Square, Macomb, IL 61455

Jayne - It's good to hear about another valuable resource. As far as the other issue goes, with the introduction of the new line of machines maybe some of those magazines will find space for Commodore. - Ralph

I have been an 8032 user for about six months and am interested in useful programs and products. I would like some information on clubs and software exchanges so I can expand my software library. I have VisiCalc so ideas on its uses would be helpful. Perhaps contact with someone in a club or a list of other publications would be useful in expanding the usefulness of my system. - Robert G Sackreiter, Rt 2 Box 90, Milbank, SD 57252

Robert - It's almost impossible to list all of the resources, user's groups and publications in this small space. A list of active users groups was a project that we started but never completed. The Commodore magazine has a list but it is not at all up to date. For useful programs contact the two groups below for their current arrangements:

ATUG
c/o Brent Anderson
200 S. Century
Rantoul, IL 61866

Chris Bennett
TPUG Secretary
381 Lawrence Ave.
Toronto, Ont M5M 1B9

These two groups have literally hundreds of useful programs at pennies per program. Both groups also publish newsletters. The Midnite Software Gazette published by ATUG and The PAPER will be combining as mentioned elsewhere. The TPUG publication is called TORPET. Other than these two the only other publication that comes to mind is The Transactor which is now back after a short absence. Write them at:

The Transactor
Canadian Micro Distributors
500 Steeles Ave
Milton, Ont L9T 3P7
CANADA

Visicorp has apparently decided not to update the PET version of VisiCalc and only reluctantly decided to "release" the 8096 version. The original authors, Software Arts, publish a newsletter called SATN which I DO NOT recommend. It costs $30 for 6 issues and constantly uses functions not available in the PET version.

I have just ordered a CBM 8032, 8050 disk drive and an Epson MX-100 printer. I am interested in hearing about geneology software that might be available. - Doris Mello, 923 Peach St., San Luis Obispo, CA 93401

Doris - This is one for the readers. Aside from some magazine articles I have not seen much in this area. - Ralph

Five months ago I got a SuperPET, 4040 disk and Diablo printer. Life, professional and personal, will never be the same. I got this system primarily for Word processing and office programs. But there is so much out there, and so much proliferation, that who knows what it will be doing for me. I practice alone, and have not had a secretary for the last few months. My secretary felt threatened by the SuperPET and did not seem interested in learning something new. I manage well by myself, but will soon find a secretary more compatible with my equipment and the future. - William A Bason

William - Your experience with your secretary is sad but not unusual. People in offices must face up to the fact that word processing and computers are here to stay. These machines will only replace them if the people refuse to update their skills to match the changes. - Ralph

Here are two problems for the Reader I/O column in the next issue:

(1) The Commodore printer uses an 8 x 6 dot matrix. This is easily confirmed by typing the shift & and counting the dots. And yet, both the user's manual (page 31) and the excellent Osborne guide (page 330) maintain it's a 7 x 6 dot matrix. Why ? It is true that most of the characters use only the top 7 lines of dots. The lowest line is used primarily for the "tails" on some letters. But certainly for user defined characters, the user should know that he can use all 8 rows. Of course, the top row is numbered 128.

(2) A jiffy is a jiffy is a jiffy. Right? Well, apparently not. Take two PET's and set them side by side. Enter a simple test program like this one into both:

```
10 TI$="000000"
20 GET K$: IF K$="" THEN 20
30 PRINT TI
```

Now type RUN on both machines and try to press RETURN simultaneously on both. After some minutes have passed, go over to the PETs and press a key on each. The numbers should match but it seems some PETs run fast and some run slow. Why?

One day I was sitting in my classroom looking at a dozen PETs and it occurred to me it would look cool to have a picture move from one PET to the next, on down the line. I whipped up a program on PET #1 that showed an archer releasing an arrow. The program repeated every 20 seconds or 1200 jiffies. The program on PET #12 showed the arrow landing on target every 1200 jiffies. The program in PETs 2 to 11 showed an arrow streaking across the screen every 1200 jiffies. After the pets were loaded and the word RUN typed on each, I just walked down the line and pressed return on each and the effect worked. But if I let the PETs run for even a few minutes the effect was lost. the arrows would be streaking across PET 5 before they reached PET 4. They would land on target while they were still on the previous screen. They would be enroute before the archer released them. Most disconcerting. - Bill Batcher, Box 300, Lake Grove, NY 11755

Bill - That's a really creative idea. I know that the timing differs from PET to PET but I don't know exactly why. Can anyone give a complete and understandable explanation? - Ralph

I have just purchased a ribbon re-inker for the Epson 80 (or Epson 100) and it has proved to be a very useful piece of hardware. I have re-inked 5 ribbons locally, and at about $4 per replacement ribbon, it will pay for itself shortly.

I have also constructed a device which I connect across the IEEE buss. It indicates the status of the buss by means of an inverter (7407) and a red LED connected to each lead. After using it for a time, you can tell if the disk and printer are working properly by watching the LEDs as they operate. The reason it was built is that I have 6 units connected across the buss. - John J Schueler, 405 Coyote Rd., Sedona, AZ 86336

John - The price of 4022 ribbons is really straining my tight budget to the limit. The cost per ribbon is high and they only seem to last about a week. This device sounds like a good deal if it is easy to use. - Ralph

After searching many different sources, I have been unable to find either of the following:

(1) A print using routine similar to Command-O's but without the limitation of 79 characters and the forced carriage return and line feed. This command could be called via 'sys x;a$:b$,t,a' and would allow for a format string of 200+ characters for 136 column printers. It would also have automatic rounding, right justification of numbers, and the output would be true ASCII characters for use on an ASCII printer.

(2) A softrom which can be used on the 8096.

Please let me know if you or any of your readers have such items and where they can be obtained. - Lee H Crisler, 4848 Clinton Blvd., Jackson, MS 39209

Lee - I must admit that every time I format output I recreate my own messy, BASIC formatting routine. A utility like you mention would be very handy but I know of none. I'll also have to appeal to the readers on the second issue since I do not yet have an 8096. - Ralph

In Volume 4, issue 1 on page 5 the code for identifying BASIC versions is discussed. At location 50003 my PET has 130. This can only be seen in a monitor dump since I get a 0 when I PEEK at this location. What version is that? My PET is one of the original small keyboard versions. How many PET BASIC versions are there? I've often wondered if there was a way to identify them all. - Stan Logue, 4330 Mt Abernathy Ave, San Diego, CA 92117

Stan - As you pointed out in a letter to me, BASIC 1.0 would return a 0 for a PEEK at any address from 49152 to 57599. Later BASICs (Upgrade or 3.0 and 4.0) were changed to remove this protection. For all pratical purposes there are three versions of PET BASIC as mentioned above. PEEKing at 50003 will return 0, 1, and 160 respectively for the three BASICs. In this way a programmer can make his program "smarter". But the situation gets worse when you consider at least three different screens; 9" 40 column, 12" 40 column and 12" 80 column. And still worse when you realize the number of different keyboards; original calculator style, and two version of the graphics and business keyboards. All of the keyboards are decoded a little differently. Don't forget the case reversal between the original PETs and all others! Couple this with around seven different disk drives, numerous printers, the VIC, 64, and even newer models and you have a software designers nightmare. - Ralph

Yes, this issue of The PAPER is late, but its arrival today is certainly timely, at least for me. Today was the first time that the particular combination of hardware and software I was using caused the version of Keyprint I had been using to let me down. And along comes The PAPER with your Universal Keyprint! However, whatever utility you are using to create your listings for The PAPER must contain a bug. The tip was that the three references to the SGN function (lines 190, 260, 360) did not contain the required parentheses. The program works when I replace the references with).

I can offer these two partial answers to Marg Farland's inquiry about interfacing a "FAT-40" to a video monitor. The people at Virginia Micro Systems distribute and interface for the 9" screen and hoped several months ago to have one for the 12" screen. Another partial answer comes from page 34 of the June/July issue of the Commodore magazine. According to this reference, POKE59520,12:POKE59521,0 should allow the old interfaces to work. - John Best, RD 3 Box 123, Jackson, NJ 08527

John - Thanks for the information. The only way many reader questions can be answered is by other readers taking the time to do so. Thank you! Another interface which I have seen working is from Batteries Included in Toronto. However, as it comes it only works on Canadian TV's. You might ask American Peripherals, 122 Bangor St., Lindenhurst, NY 12757, about this problem since they seem to have the code to make the interface work. - Ralph

Irate Epson Owners Reply

As I have said before, it is impossible for me to check out every question, complaint, problem and article we get. If I limited articles to those I could check on the equipment at hand, the subjects would be somewhat limited. A few readers have shared A.H. McCann problems with the Epson printer but many more have had no problems at all. In an effort to correct any misconceptions left by that article, here are a sample of the responses from readers. Please accept my apologies for not verifying all articles before publication.

I have an Epson printer connected to my 4032 and SP9000 and have had no trouble using the IEEE-488 interface and my homemade word processor. To get the program to work I had to change only one number in the program. - John Schueler

Recently I installed an 8032 and Epson MX-80FT for one of the other departments in my school system. I had some of the same bad experiences reported by Tim Amodemo and A. H. McCann. I happen to like the Commodore 4022 very much, and with the lack of clear instructions for adjusting the Epson, I was cursing whoever had recommended a non-Commodore printer! Since the system had been purchased from AB Computers I called them. They advised me to return the dip switch settings to those recommended in the instructions, to use a printer file numbered higher than 128, and finally to include "lf1" among my formatting commands when using WordPro. This proved to be perfect advice. Since using OPEN130,4:CMD4 is transparent to the Commodore printer I now use the larger number no matter what printer I am using. I must admit that I was so impressed by the added capabilities of the Epson over my Commodore 4022 when I finally got it working that I traded the 4022 in on an MX-100. - John Best

I'm writing to give a necessary rebuttal to the article "Beware the Epson Jaberwock", in which author A.H. McCann concludes, "...don't buy the Epson IEEE-488 I/F board if you want to use a word-processing program." The fault is in his word processor, WordPro II, not in the interface board. The board is designed to handle an IEEE-488 input of true ASCII characters, and is quite competent to do so. However, the PET's output is neither true IEEE or ASCII. The problem he is experiencing is with the non-ASCII code. It's too bad he spent $129 for an interface to solve something which should have been solved by his word-processing program, but then WordPro II is obsolete and not supported by Commodore. Any proper word-processor which is intended to be used with different printers will have a startup questionnaire designed to elicit from th user the necessary information to determine how to handle the carriage return and IEEE compatibility problems, and will then make the necessary corrections automatically. In fact, even WordPro II, as bad as it was, had some hoked-up provision, requiring the loading of a program called "asc editor" from the disk before preceding. A check of the instruction book will probably show a reference under "Output Options". As for Gary Brooks, who is apparently making money out of other people's ignorance, I don't blame him: waht else is new? In conclusion, I will say that it is ridiculous for Epson not to explain the situation in their literature, but except for that, they are not at fault.

For experimenters and auto-flagellants, the following algorithm will correctly translate all PET screen codes for ASCII characters to approved ASCII code and will handle all the shifted keys except pi:

```
A=(63 AND P)+ 2*(64 AND P)+2*((NOT P) AND 32)
```

Of course, one wants to do this in machine language, where it is simply a bit juggling act. - Frank Chambers, Rock House, Ballycroy Westport, Co Mayo, Ireland

I really do not take exception to much of Mr. Chambers letter. His comments are correct for his wordprocessor, "Papermate", and his Commodore equipment. It is true that some new word processing programs now have the capability of making conversions for ASCII printers. My situation and comments were based on my desire to use my 2023 printer and the Epson MX-80 FT printer simultaneously. I also wanted to continue usinng the WordPro II which I had installed in my CBM 2001-32B. The WordPro was meeting my needs for word processinng and I was satisfied with it.

The main advantage of the interface I purchased from Gary Brooks is that I can have two printers connected simultaneously, the Commodore printer as device #4 and the Epson as device #5. My greatest frustration was with the Epson User's Manual which gave no useful instructions for installing and operating their printer with a Commodore computer system. - A. Hews McCann, 3430 Frankfort Ct., Oxnard, CA 93033

In response to the question from Stan Spence on blanking the screen of the 8032. Try POKE59520,1:POKE59521,0 to blank the screen and POKE59520,1:POKE59521,40 to restore it to normal. All the screen functions on the 12" screens are handled by the 6845 controller chip, which I would like to say I understand fully but don't. However I recommend that EVERYONE buy a copy of PROGRAMMING THE PET/CBM by RAETO WEST.

I also have source code for Basic Aid and can make a version for any model at any location. This has the CRT or screen dump in it plus lots, lots more. Send a disk in a reusable mailer with return postage and label. Be sure to specify CBM printer or ASCII. Disks will be 4040 format. I can make a version of Micromon the same way. Be sure you state where you want it located.

Daniel Condon should get in touch with Clark Stewart, 104 Henrietta St, Ravenswood WV 26164 (304/273-4680). Clark has been on the air for sometime with his PET and has two different nets up on weekends.

Having trouble with VIC programs ? Especially those that run only in the original 5K ? Make friends with a PET owner. The extra memory and the resident monitor make a good place to look them over. You won't be able to run them there but can use Toolkit, Micromon, etc., to work them over.

I will be glad to take a shot at any 8032 questions or problems. - Jerry Key 141 Flint Ridge Dr., Columbus OH 43230 (614) 475-6060

Jerry - Thanks for all the information. Here is another reader and author who takes the time to let others in on what he knows. - Ralph

The Trace program by Brett Butler found in COMPUTE's First Book of PET/CBM (p. 153) and in PET Subroutines by Nick Hampshire doesn't work on BASIC 4.0. Make the following changes in DATA statements for BASIC 4.0 use:

line #11 (COMPUTE) or #20 (Hampshire) - 249, 224 to 153, 211
line #14 (COMPUTE) or #50 (Hampshire) - 121, 197 to 240, 181
line #25 (COMPUTE) or #160 (Hampshire) - 145, 192 to 177, 176
line #26 (COMPUTE) or #170 (Hampshire) - 145, 192 to 177, 176

For a quick conversion of Charles Brannin's Quadra-PET in the same COMPUTE anthology (p. 163), make the following changes. - Kenneth Stein

line #1020 - 28, 202 to 29, 187
line #1070 - 119, 197 to 238,181

Kenneth - Another note of thanks to a reader. These are two programs I would never have even thought to look at since I use neither. - Ralph

Background/Foreground

This section includes announcements, short notes and comments that have been picked up since the last issue.

New Word Processor

A new word processor from England called SuperScript will soon be available in this country. The program is completely compatible with WordPro but offers some interesting additional features. The disk is uncopiable but no protection chip is needed and one version of the program will work on BASIC 3.0 or 4.0, 16K or 32K and 40 or 80 column machines. Editting and formatting are much the same as WordPro with some added flexibility. Six printers are "officially" supported but others work very nicely and the company seems willing to include others. The ability to load and save ASCII files, send true escape codes, and have at least 1/2 again as many lines of text are but three of the attractive features of this package. The price for two disks and a professional tutorial and reference manual will be $250 with substantial discounts for educational institutions. A large dealer/distributor network will provide customer support and replacement disks. A data base management program to use with SuperScript will be available in the first quarter of 1983. Call or write Centerbrook Software, Box 460, Livingston Manor, NY 12758, (914) 439-3591 for more information.

New York Educational Conference

The New York State Association for Educational Data Systems will hold its 17th annual conference November 7 - 9 at the Americana Inn in Albany. The theme of the conference is "Moving Ahead with Educational Computing". There will be five different sessions covering the major microcomputers used in the classroom, how computers are used and the software needed to make them operate. In addition there will be two keynote speakers and an all day vendor display on Monday, November 8. A special feature will be Jim Butterfield's all day machine language workshop on Sunday, November 7. For more information contact Don Ross, Ardsley HS, 300 Farm Rd., Ardsley, NY 10502

Does Your PET Talk?

Learning Tree Software, Box 246, Kings Park, NY 11754 has developed several complete series of "naturally voiced" programs aimed primarily at the lower elementary grades. What this company has done is to design a tape deck which will load PET programs and playback a recorded voice. After the program has been loaded, commands within the software control when the tape deck is turned on and off to play the prerecorded speech. The programs are highly graphic and many make use of a light pen. The demonstrations I saw were very impressive and worked well. The price of the tape deck is $175, the light pen is $38 and a 20 program series goes for $400. Programs may be purchased separately for $25.

Public Domain

We all know what this term means but it is also the name of an Ohio group which proposes to take on the giant task of organizing ALL public domain PET software. The concept is simple. The group is asking any users group or individual with programs to send them their disks. Programs will be evaluated and cataloged onto various disks. The group will pay for any copying costs involved when you send them your disks. Once the disks are organized they will

be available for $10 each to cover postage and handling.

This is a necessary task but one that has been undertaken without much success before. The goals are admirable and the service is needed if done correctly. Only the best of a particular type of program should be included and careful culling must be done. For more information contact William Munch, 5025 S. Rangeline Rd., West Miltom, OH 45383

VIC Plotting Program

Anyone with a VIC-20 or Commodore-64 and 1515 printer should appreciate this new "High Resolution Plotting Routine". It will plot the results of user supplied functions, programs or data logging routines. Size of the plot and automatic scaling with number labels and tic marks are user adjustable. The program uses 72 dots/inch horizontally and 63 vertically for the highest resolution available. The sample plot sent me was very nice. The price is $10 from Scientific Software, Dept. E, 525 Lohnes Dr., Fairborn, OH 45324

New from CGRS Microtech

The makers of the fastest PET disk, PEDISK, have several new products to make note of. Color Chart is a color video generator board which allows the PET to produce 8 colors, alphanumerics and hi-res graphics on an external color monitor. The board plugs into a 4K ROM socket and appears as 4K of RAM. Several different modes of operation include a 32 by 16 alphanumeric display and a 128 by 192 full graphics mode. The price is $139.95. Watch for more developments in this area.

Portmaker is a small dual serial port I/O board that plugs into the ROM socket of a PET to add serial RS232 capability. A ROM socket on board allows use of all but the top 16 bytes of the original ROM. Available along with this hardware is the Standard Terminal Communications Package for Eastern House Software that converts the PET into a data management center. Disk files can be serially transmitted and received data can be recorded on disk or printer with complete control of all peripherals. A real time clock with display and alarm is included. the Portmaker hardware is available for $69.95. The complete COMPACK package including Portmaker, cable and the STCP software is $129.95. Contact CGRS, PO Box 102, Langhorne, PA 19047 for a package that may solve most of your communications problems.

Educational Activities

These are something that should happen in all schools and they are made easier by a company with the same name. This company has a large selection of computer software, books and filmstrips developed by teachers in the field. I know many of the authors and their materials are some of the best around and are summarized in a new catalog. Send for it to Educational Activities, Dept 83A, PO Box 392, Freeport, NY 11520.

Directory of Educational Computing

This is not a new job but a publication of Classroom Computer News. The Directory is filled with over 200 pages of descriptions of periodicals, professional associations, on-going projects, funding, and many sources of ideas, information and materials. A group of concerned educators has compiled over 1000 listings arranged for easy reference. Guidelines are provided for

hardware and software evaluation and selection, and a complete yellow pages lists computer services and products. A local and regional catalog of organizations using the same brand of computer is also featured. For pricing and where to obtain your copy contact Classroom Computer News, 341 Mt. Auburn St., Watertown, MA 02172

InfoAge

This is a new Canadian computer magazine which has disappointed me from the first issue. It is slick and does contain general information and techniques but nothing really specific. Those that favor this approach will find InfoAge interesting but the treatment has left me flat. Right now it is being published bimonthly and costs only $20 for 12 issues. Send to InfoAge, 211 Consumers Road Suite 302, Willowdale, Ont CANADA M2J 9Z9 for more information.

The Jefferies Report

Ron Jefferies had a real winner in CURSOR but I'm afraid his latest venture falls short of this mark. The newsletter is published monthly, is mailed first class for $30 a year which may be somewhat steep for six pages per issue. The weakest part of the newletter is that most of the news has already been reported in equal depth in other publications like InfoWorld. Ron does get some "hot" inside information at times and I like his writing style. As far as I know, Ron is the only writer which was his intent from the start. There are no ads to take up space or make the publication dependent on anyone which also contributes to the higher cost. The newsletter does not concentrate on the PET/CBM but treats it fairly and after two months seems to have no bias. My suggestion is to try 6 months and see how you like it. Contact the Jefferies Report, Box 6838, Santa Barbara, CA 93111.

Transactor Resurrected

After a short absence which caused sleepless nights for many users Carl Hildon has reformed The Transactor uder the auspices of Canadian Micro Distributors. Carl believes the new sponsorship will in no way affect the excellent quality of this technical newsletter and he has already planned bigger and better things. After leaving Commodore, Carl will be able to devote more time to a new glossier and reference oriented publication. A subscription to Volume 4 is only $15. Send to The Transactor, Canadian Micro Distributors, 500 Steeles Ave, Milton, Ontario, CANADA L9T 3P7

COMAL is VERY Alive and Well

Although COMAL has been slow to catch on in the US, it is already the standard language taught in both Ireland and Denmark with possibly England and Sweden to follow soon. As of May 1982, there is also an official COMAL standard called The COMAL KERNAL which will make versions of COMAL running on different machines more compatible. There are also several newsletters and books including a book published by Reston and authored by Len Lindsay (300-400 pages with 100 sample programs). The new COMAL version 0.12 will soon be available as well as 1.02 which will contain the complete KERNAL but only run on the 8096. The COMAL Catalyst is a most valuable newsletter with news, tips and programming information including tutorials by the original author of COMAL. For information write COMAL CATALYST, 5501 Groveland Terrace, Madison, WI 53716.

Machine Language for Beginners: Part V

by Ralph Bressler

This is the fifth and definitel y the last in a series of articles dealing with programming and techniques in machine language. Lately, I have had to borrow too many of my examples from others or contrive ones which I really don't use. This comes from the fact that I am trying to explain and illustrate techniques which I have just learned and seldom use.

As I mentioned in the last article, everyone who is serious about programming in machine language should have a copy of one of the monitor extensions such as Supermon. The mini-assembler in these packages makes entering code much easier and the disassembler allows you to see the assembly code for any machine code in any location. As programs get longer, it is almost imperative that you also have an assembler. Even small and simple assemblers allow you to save your source code for later inspection and modification. They allow comments to remind you of what you were doing and permit you to place your code where you like with few restrictions. So many assemblers are now available that choosing one can be difficult. For beginners, the primary criteria is that the assembler should be inexpensive but easy to use. If you become a dedicated ML hacker then you can buy a more expensive and more versatile program. In any case, there are many topics still to cover so let's get started.

First, I have two "assignments" to give the answers to. One problem was getting our addition program from last time to handle higher math like 9 + 3 = 12. You may remember that answers greater than 9 were a problem. We still will only attempt to add single digits but we should be able to handle double digit answers. Since the change in the code is minor, I will give only the part of the program where the changes begin. This is shown below:

```
line addr  code       label  mnemonic          comment

 432 03BD  C9 0A             CMP #$0A          ;COMPARE TO 10
 434 03BF  90 09             BCC SKIP          ;IF LESS THEN GO ON
 436 03C1  E9 0A             SBC #$0A          ;ELSE SUBTRACT 10
 438 03C3  AA                TAX               ;STORE RESULT
 440 03C4  A9 31             LDA #$31          ;LOAD A W/ 1 AND
 442 03C6  20 D2 FF          JSR WRT           ;...PRINT ON SCREEN
 444 03C9  8A                TXA               ;RECALL PREVIOUS RESULT
 446 03CA  09 03       SKIP  ORA #$30          ;CHANGE TO ASCII
```

Everything before and after this is the same as in the original program. What we have done here is really quite simple, after doing the addition we compare the answer to 10 in line 432. If the answer is less than 10 (the carry is clear), we do exactly what we did before. When the number is 10 or greater, we subtract 10 (line 436), store the result (line 438), print a 1 (lines 440 and 442), recall the previous result (line 444) and convert it to ASCII and print it as before. Again, this technique is not meant to be universal but works for this situation.

Another problem we mentioned last time was the creation of a cursor positioning routine. Let me emphasize that this is an exercise to try to do in machine language what is easy to do in BASIC. Several articles have appeared elsewhere lately giving elaborate cursor positioning routines to use in BASIC programs. This is nonsense since BASIC can do it much easier. Some of you may know that there are locations which may be POKEd in BASIC to position the cursor. These should not be used in BASIC or machine language programs as they may change from one version of BASIC to another. The routine below uses two simple loops to get the cursor where we want it.

```
line addr  code       label  mnemonic          comment

 100                  ;CURSOR POSITIONING
 110                  ;
 120                  WRT=$FFD2 ; WRITE CHARACTER TO SCREEN
 122                  OVER=$BA ; STORE # OF SPACES TO GO TO RIGHT
 124                  DOWN=$B9 ; STORE # OF LINES TO GO DOWN
 130                  ;
 140                  *=$0375 ; BEGINNING ADDRESS OF ROUTINE
 145                  ; USE JSR $0375 OR SYS 885
 150                  ;
 160 0375  A9 13            LDA #$13          ;LOAD ACC WITH "HOME"
 170 0377  20 D2 FF         JSR WRT           ;...AND PRINT IT
 180 037A  A6 BA            LDX $BA           ;LOAD X FROM "OVER" LOC
 190 037C  A9 1D            LDA #$1D          ;LOAD A W/ RIGHT CURSOR
 200 037E  20 D2 FF  LOOP1  JSR WRT           ;...AND SET LOOP TO PRINT
 210 0381  CA               DEX               ;...IT UNTIL
 230 0382  D0 FA            BNE LOOP1         ;...CURSOR IS POSITIONED
 240 0384  A6 B9            LDX $B9           ;LOAD X FROM "DOWN" LOC
 250 0386  A9 11            LDA #$11          ;LOAD A W/ DOWN CURSOR
 260 0388  20 D2 FF  LOOP2  JSR WRT           ;...AND SET LOOP TO PRINT
 270 038B  CA               DEX               ;...IT UNTIL
 290 038C  D0 FA            BNE LOOP2         ;...CURSOR IS POSITIONED
 520 038E  60               RTS               ;RETURN TO CALLER
```

This routine is self explanatory and well commented. Notice that is uses only the ROM-based WRT subroutine which has stayed the same on all Commodore products so far. Location $BA is used to store the amount we want to move horizontally while $B9 stores the vertical movement. Any free locations could be used.

Last time we introduced and made extensive use of subroutines. Some routines are ROM-based and make our jobs easier. We also create and use our own routines particularly as our programs grow larger. An explanation of how the PET handles subroutine calls was given last time and you should remember that the address to which the PET will return when done with the routine is stored on the stack. Actually, the address minus one is stored so that when the address is "popped" off the stack and placed in the program counter, the counter may be incremented as usual to get the "real" address. Keeping this in mind, we should be able to alter this "return address" if we wanted by using push and pull commands. In this way we could jump to almost any location we wanted. The example below is contrived but illustrates the method. In this short program we use our own subroutine which checks to see if we press a numeric key. If we do, the routine returns normally and clears the screen. If we press anything but a number the return address on the stack is altered to avoid the clearing of the screen.

```
line addr  code       label  mnemonic          comment

 100                  ;PROGRAM TO ILLUSTRATE THE STACK
 110                  ;AND RETURN ADDRESSES FOR ROUTINES
 120                  ;
 130                  *=$0390
 140                  ;
 150 0390  20 99 03         JSR PSHNUM        ;ROUTINE CHECKS NUMBERS
 160 0393  A9 93            LDA #$93          ;LOAD A WITH CLEAR SCREEN
 170 0395  20 D2 FF         JSR $FFD2         ;...AND PRINT TO SCREEN
 180 0398  60               RTS               ;RETURN TO BASIC
 190                  ;
 200 0399  20 E4 FF  PSHNUM JSR $FFE4         ;GET A CHAR FROM KEYBOARD
 210 039C  F0 FB            BEQ PSHNUM        ;LOOP UNTIL KEY PRESSED
```

```
220 039E  C9 30          CMP #$30          ;CHECK TO SEE IF KEY WAS
230 03A0  90 05          BCC OVER          ;...LESS THAN OR
240 03A2  C9 3A          CMP #$3A          ;...GREATER THAN A NUMBER
250 03A4  B0 01          BCS OVER          ;...AND IF IT IS NOT THEN
260 03A6  60             RTS               ;...RETURN IMMEDIATELY
270 03A7  68       OVER  PLA               ;IF NOT A NUMBER THEN
280 03A8  68             PLA               ;...PULL OFF THE RETURN
290 03A9  A9 03          LDA #$03          ;...ADDRESS AND PUSH
300 03AB  48             PHA               ;...ON A NEW ADDRESS
310 03AC  A9 97          LDA #$97          ;...AVOIDING THE CLEAR
320 03AE  48             PHA               ;...SCREEN OPERATION
330 03AF  60             RTS               ;...COMPLETELY
```

To try this out use SYS 932 and experiment with pushing numbers and non-numerics. Notice that if we do not press a number we want to return to $0398. To do this we push $03 and $97 on the stack so that they end up in typical high byte-low byte form. When RTS is encountered this address is placed in the program counter and incremented to $0398.

When coding machine language programs it is best to keep in mind that you or some other programmer may eventually want to move the program to another location. This is called relocation and the fewer the number of changes you have to make the better. This means that, in many cases but not all, the simplest, most direct way is best. Try to avoid unnecessary subroutines and jumps to locations within the program and be careful of using too many "cute" tricks. Attempt to use mostly branch commands even if they take a little more code. Here is an example of what I mean:

```
033A  LDY #$00            033A  LDY #$00
033C  TYA                 033C  TYA
033D  LDX #$00            033D  LDX #$00
033F  STA $8000,X         033F  STA $8000,X
0342  INX                 0342  INX
0343  BNE $033F          0343  BNE $033F
0345  INY                 0345  INY
0346  BNE $033E          0346  BNE $033E
0348  JMP $033A          0348  BEQ $033A
```

The program on the left uses a JuMP instruction to set up an infinite loop much as one would in BASIC. But this technique ties it to location $033A. The same program on the right uses a branch instruction to obtain the same result but is not tied to any specific location. The program on the right is "completely relocatable". This seems trivial but too many "mistakes" like this make relocation unnecessarily difficult.

One of the real problems that faces a machine language programmer is where to put his program. If you are working only in machine language, then you can put the program almost anywhere. In fact, to allow the most space and to make it easy to save and run you would probably put it where BASIC normally resides. You could start at $0400 and work up to the memory size of your machine. If you do start at $0400, then the user will have to remember to type SYS 1024 every time he uses the program. Many machine language programs start with a SYS and then all the user has to do is type RUN. Enter 10 SYS(1039) as a BASIC line exactly as it appears. Now enter the machine language program below starting at $040F.

```
100 040F  A9 93          LDA #$93          ;LOADS A WITH CLEAR SCREEN
110 0411  20 D2 FF       JSR $FFD2         ;WRITE CLEAR TO SCREEN
120 0414  60             RTS               ;RETURN TO CALLER
```

Now when we save the screen we use the monitor to save the BASIC line and the machine language. Remember to save one byte beyond the end of the program. It would look like this for tape #1: .S "CLEAR",01,0400,0415 or this for disk drive 0: .S "0:CLEAR",08,0400,0415 . Now when you load this program again the BASIC line and machine language will load normally and you can even resave it without the monitor.

This is fine for machine language only but most programmers combine machine code with BASIC. When you do this you may store the machine code in three different places, each with its own advantages and disadvantages. Until just lately many useful routines were stored in the cassette buffers. Each of these is 192 bytes long which is good for short, useful routines. If you were using a disk drive you could use both cassette buffers or if your system was tape based you could use the second cassette buffer. This was a problem since you could not be sure what kind of system another user who might want to employ your routine had. With the coming of BASIC 4.0 and the FAT 40 even the second cassette buffer became crowded. Besides all those useful rotuines all located in one area of memory, this buffer is now busy keeping track of DOS commands and tab stops.

Many people have now taken to storing their machine language routines at the top of memory. This allows quite a bit of space even in 8K and is not affected by operations from BASIC. This area is good for routines you want present from the time you turn the machine on. Several problems plague this method also. First, as more people switch to this method, the more crowded even this area will become. It is possible to load one routine after another and keep moving down in memory providing the programmer took the care to make this possible. A bigger problem is that the top of memory varies with the memory size of the machine. It is not too easy to write a machine language program that automatically relocates itself to the top of any memory. These routines must also be loaded carefully and the top of memory pointer must be set down each time to protect the area from BASIC. Of course, lengthy routines will make the space for BASIC programs and variable storage smaller.

Another place to put your machine language routines is right after your BASIC program before the storage of variables. This is perhaps the most difficult method to implement since it requires very careful set up. You will, of course, have to call your routine with a SYS from your BASIC program. Any small change in that program will cause the machine language program to shift and your call via SYS may no longer be to the correct address. Also, as the program shifts any references to addresses within the program may be incorrect. This method does allow normal loading and saving of both the BASIC and machine language parts. It also means that the program will load into any size memory providing it is big enough with no special provisions for relocation or changes in any pointers. Our next program will use this technique but before we attempt it we need to introduce two more topics.

There are four commands which we have not yet dealt with which can prove useful. These commands rotate or shift the bits in a number left or right. The four commands are:

ASL - arithmetic shift left	LSR - logical shift right
ROL - rotate left	ROR - rotate right

The differences in these commands are best shown using examples. These commands can act on the contents of the accumulator or any selected byte of memory. All of these instructions affect the carry, sign and zero flags of the status register.

ASL shifts all bits one place to the left, places the contents of bit 7 in the carry, and places a 0 in bit 0. An example might be as follows:

```
before:  C=0   0 1 0 0 1 1 0 1  =  77
after :  C=0   1 0 0 1 1 0 1 0  = 154
```

Notice that the effect of one ASL is to multiply the number by two. Also, the sign bit would be set, zero bit reset and the carry bit reset as shown.

LSR shifts all bits one place to the right, places a 0 in bit 7, and places the contents of bit 0 into the carry. An example might be as follows:

```
before:  C=0   0 0 1 1 0 1 0 0  =  52
after :  C=0   0 0 0 1 1 0 1 0  =  26
```

Notice that the effect of one LSR under some conditions is to divide the number by two. Also, the sign bit is always set to 0, zero is set to 0 and carry is set to 0 as shown.

ROL is similar to ASL except that the contents of the carry are rotated into bit 0 instead of a 0 as with ASL. An example might look like this:

```
before:  C=1   0 1 1 1 1 0 1 0  =  122
after :  C=0   1 1 1 1 0 1 0 1  =  250
```

Here, the sign bit is set to 1, zero is set to 0 and carry is set to 0 as shown.

ROR is similar to LSR except that the contents of the carry are rotated into bit 7 instead of a 0 as with LSR. An example might look like this:

```
before:  C=1   0 1 1 1 1 0 1 0  =  122
after :  C=0   1 0 1 1 1 1 0 1  =  201
```

Here, the sign bit is set to 1, zero is set to 0 and carry is set to 0 as shown.

A final topic must be presented before we create our next sample program. If we truly want to interface BASIC and machine language we should be able to pass variable values from BASIC to our ML program, process those values and retrieve them for use again by our BASIC program. To do this we must know a little about the storage of variables in the PET. All variables are stored as 7 bytes no matter what type or what their value is. The first two bytes are the name of the variable modified suitably so the PET will know if it is a floating point, integer or string variable. Floating point (numbers with fractional parts are stored in floating point notation which is not easy to comprehend. What is stored for strings is not their value but a pointer to where they are located in high memory. This leaves integer variables which are easiest to understand. After the first two bytes which are the variable name, the next two bytes are its value with the final three bytes remaining unused. The value is stored in two's complement form in high-byte, low-byte order so that bit 7 of the third byte indicates the sign of the number. A pointer at 002A and 002B (007C, 007D in BASIC 1.0) indicates the start of variable storage. This pointer points to the first character in the name of the first variable stored. The pointer plus one points to the second character, plus two to the high byte of the value and plus three to the low byte. If we store the variable we want to work with before any other and as an integer, then we will be able to easily pass the value to a machine language routine. The routine can then process the value and store the result somewhere or even change the value of the variable by storing it back where it came from. We are now ready to start our program which will allow the user to enter a number from BASIC, call a machine language routine to multiply this number by 8 and then go back to BASIC to retrieve and finally print the result. The machine language program will be stored after our BASIC program starting at $0490 and will use ASL and ROL to do the multiply. First, let's write the simple BASIC program:

```
10 X%=0
20 INPUT "NUMBER";X%
30 SYS(1168)
40 PRINT "...TIMES 8 IS";X%
50 GOTO 10
```

Line 10 makes sure that our variable X% gets stored as the very first variable. We will now write our machine language program at $0490. Remember that once this is in place we cannot change our BASIC program or we will also have to change the SYS call in line 30. Our machine language program is well commented and here it is:

```
100                ;GETS A INTEGER VARIABLE STORED BY
110                ;BASIC AND MULTIPLIES IT BY EIGHT.
120                ;
130                TEMPL=$033A ;LOCATION FOR STORAGE
140                TEMPH=$033B ;LOCATION FOR STORAGE
150                *=$0490
160                ;
170 0490  A0 03          LDY #$03          ;LOAD Y TO USE AS INDEX
180 0492  B1 2A          LDA ($2A),Y       ;GET LO BYTE OF VARIABLE
190 0494  8D 3A 03       STA TEMPL         ;STORE FOR USE LATER
200 0497  88             DEY               ;DECREASE Y
210 0498  B1 2A          LDA ($2A),Y       ;GET HI BYTE
220 049A  8D 3B 03       STA TEMPH         ;STORE IT NEXT TO LO BYTE
230 049D  A2 03          LDX #$03          ;SET X AS COUNTER
240 049F  0E 3A 03  LOOP ASL TEMPL         ;SHIFT LO BYTE LEFT
250 04A2  2E 3B 03       ROL TEMPH         ;ROTATE HI BYTE LEFT
260 04A5  CA             DEX               ;DECREASE X
270 04A6  D0 F7          BNE LOOP          ;IF NOT DONE THEN LOOP
280 04A8  A0 03          LDY #$03          ;IF DONE SET Y AS COUNTER
290 04AA  AD 3A 03       LDA TEMPL         ;GET PROCESSED LO BYTE
300 04AD  91 2A          STA ($2A),Y       ;STORE BACK AT ORIGIN
310 04AF  88             DEY               ;DECREASE Y
320 04B0  AD 3B 03       LDA TEMPH         ;GET PROCESSED HI BYTE
330 04B3  91 2A          STA ($2A),Y       ;STORE BACK AT ORIGIN
340 04B5  60             RTS               ;RETURN TO BASIC
```

Here, in lines 170 to 230 we get the low byte which is at pointer plus 3 and store it in a tempoarary location. When we decrement Y we can then get the high byte from pointer plus two and store it temporarily. The loop from 240 to 270 shifts the low byte left and then rotates the high byte left for a total of four times. This accomplishes the multiplication by eight. We rotate the high byte because we want to bring in the carry which was affected by the shift of the low byte. Finally, we recall the processed low and high bytes and store them back where they came from and return to BASIC.

A careful study of this program reveals that we have used many new techniques and ideas. As always, I suggest you make some changes and see what happens to the BASIC and machine language programs and the results obtained. First, try adding a line 5 with just a REM statement. Now use SUPERMON or the PET's built-in monitor to see what happened to the machine language program. Can you see how it moves up some in memory? Remove the line you added and look at the machine language portion again. What happened to it? Is it back where it started? It is relatively easy to multiply a number by a multiple of two. All you have to do is shorten or lengthen the loop in lines 240 to 270. Change the program to multiply the number by 10. To do this you multiply by two, store the result, do the multiplication by eight and add the two for the final result. Accessing a variable other than the first is easy unless you get fancy.

All this talk about moving programs around and storing them in different locations makes me want to reissue a warning. Try to program so that your code can be easily moved from one set of locations to another. This means that you should try to avoid jumps and use branches wherever possible. Jumps tie your program to one set of locations while branches with their relative addressing do not change when the program is moved. This, of course, is not always possible since branches have limited range. Try to steer away from subroutines in your program unless they increase the speed or performance significantly. Try not to use "cute" techniques like interrupts or wedges when others will do. If you stick to these suggestions, you will make your programs easier to transport from one location to another. Some short routines can be made totally relocatable with little effort.

There are two other topics which I would like to cover; interrupts and "wedges". Both of these have been covered before in this newsletter but I feel they should be mentioned here. Both of these techniques allow you to add functions to your PET and its BASIC but both need careful planning. Sixty times each second the PET interrupts what it is doing to do many important functions like checking the keyboard and updating the clocks. We can intercept this interruption and make the PET perform a function we want it to before it does its own work. This is called an interrupt driven program. The wedge technique gets its name from the fact that we shoehorn it into several locations in page zero. Whenever you hit RETURN to execute an immediate command or when a program is running a part of code called CHRGET (character get) is executed to get the next character from the line being executed. Eventually a command gets performed when it is recognized or a ?SYNTAX ERROR is issued if it is not. We can again intercept this function and direct the PET to scan our code first to see if the command is part of our program. If it is then our command is performed and then the PET goes on with its own checking. I am fairly well acquainted with the operation of interrupts but have only a passing knowledge of "wedges". Both of these techniques are advanced and may never be used by beginning programmers. However, when analyzing some of the programs which are published you will find them and it might be nice to at least recognize them.

The following program allows you to constantly display any page (256 locations) of memory at the top of the screen. This can be interesting especially if you use page 0 since you can see many important locations change as you do various things. There is a pointer at locations $90 and $91 that point to the normal interrupt routine at $E455 ($E26E in BASIC 2.0). We must change this pointer to point to our program and, then, at the end of our program we must jump to this routine so the PET can carry out its normal operations. When setting the pointers to our routine we must stop the normal interrupt procedure. If we do not do this the interrupt routine may strike before we complete the change and jump to some unknown location. Here is the program:

```
110                ;DISPLAYS ANY PAGE OF MEMORY
120                ;CONTINUOUSLY ON THE SCREEN.
130                ;TYPING LOAD WILL DISABLE PRG
140                ;USE POKE 927,X TO CHANGE PAGES
150                ;WHERE X IS THE PAGE NUMBER.
160                ;
170                *=$0390
180                ;
190 0390  78             SEI              ;SET INTERRUPT DISABLE
200 0391  A9 9B          LDA #$9B         ;...AND SET UP VECTOR
210 0393  85 90          STA $90          ;...TO POINT TO OUR
220 0395  A9 03          LDA #$03         ;...SHORT PROGRAM AND
230 0397  85 91          STA $91          ;...DO IT FIRST
240 0399  58             CLI              ;ENABLE INTERRUPTS AND
250 039A  60             RTS              ;...RETURN TO BASIC
```

```
270 039B  A2 00          LDX #$00          ;ZERO XR AS COUNTER
280 039D  BD 00 01 LOOP  LDA $0100,X       ;LOAD FROM PROPER PAGE
290 03A0  9D 00 80       STA $8000,X       ;STORE ON SCREEN
300 03A3  E8             INX               ;BUMP XR BY ONE
310 03A4  D0 F7          BNE LOOP          ;...LOOP IF NOT DONE
320 03A6  4C 55 E4       JMP $E455         ;DO NORMAL INTERRUPT
```

To activate this program use SYS912. Try clearing the screen while this routine is running. The screen WILL clear but will immediately fill again with the page of memory you are displaying.

Our final technique deals with a wedge. Wedges can be incredibly complex and add many different commands to BASIC. Our wedge will be very simple. When the @ sign is pressed and RETURN is hit the PET will go to the machine language monitor. To set up the wedge we insert a jump to our program at the beginning of the CHRGET routine on page zero. This routine begins at $70 and we use the first three locations. All wedge programs do this and like all the rest we must replace what was there when we write our code. As soon as this program is activated it will wait until a @ is hit and then go into action. Here is the program:

```
110                ;BREAKS TO MONITOR WHEN @ IS HIT.
120                ;ACTIVATE WITH SYS912
130                ;
140                *=$0390
150                ;
160 0390  A9 4C          LDA #$4C          ;SET UP WEDGE JUMP
170 0392  85 70          STA $70           ;...BY REPLACING CHRGET
180 0394  A9 9D          LDA #$9D          ;...CODE WITH JUMP
190 0396  85 71          STA $71           ;...TO OUR PROGRAM.
200 0398  A9 03          LDA #$03          ;...WEDGE WILL THEN WORK
210 039A  85 72          STA $72           ;...WHENEVER @ IS FOUND.
220 039C  60             RTS
240 039D  E6 77          INC $77           ;REPLACE CHRGET CODE
250 039F  D0 02          BNE OVER          ;...DESTROYED ABOVE WITH
260 03A1  E6 78          INC $78           ;...ORIGINAL CODE
270 03A3  8C 7A 02 OVER  STY $027A         ;SAVE Y
280 03A6  A0 00          LDY #$00          ;ZERO Y FOR COUNTER
290 03A8  B1 77          LDA ($77),Y       ;LOAD A USING Y AS INDEX
300 03AA  AC 7A 02       LDY $027A         ;BRING Y BACK
310 03AD  C9 40          CMP #$40          ;CHECK FOR @ SIGN
320 03AF  F0 03          BEQ BREAK         ;...IF IT WAS THEN BREAK
330 03B1  4C 76 00       JMP $0076         ;...IF NOT CONTINUE
340 03B4  00       BREAK BRK               ;BREAK TO MONITOR
```

This program is, perhaps, the simplest version of an extremely versatile structure. Whole lists of commands like AUTO, REN and DEL can be created each with its own distinct function. Of course, since BASIC has to search this whole new command list things may slow down some but there are even ways to get around this problem. Doug Haluza gave a more complete discusion of this technique with a slightly more complex example in the last issue. Try reading this and then start looking at some listing or disassemblies of programs that use the technique.

This has been a long series drawn out over too long a time by our constant problem with publishing on schedule. Many people have said this series has helped them become a little less scared and a little more willing to work with machine language. Writing it has helped me to gain new insights and to learn a lot.

Observations on Volume 4, Issue 3

by Roy Busdiecker

WOW! It's hard to believe how much information can be packed into one publication! it's reached the point where doing this column in a reasonable length of time has become a challenge. Thanks to Ralph for his extra effort ... my copy of this issue was an advance version, straight out of his printer.

Long Live Commodore

Apparently, I wasn't the only one whose feathers were ruffled by 'The Late Great Commodore' editorial two issues ago. Ralph has done penance admirably with his follow-up piece. Wish we could have gotten the same kind of turn-around from the folks at The Code Works, who publish Cursor Magazine for the PET/CBM ... they've gone off the deep end and terminated that worthwhile resource with issue #30. Their rationale was that a) they didn't charge enough at $4.95/issue, to pay decent royalties to the software designers; b) it's too hard to create programs for all the differnet versions of Commodore PET/CBM computers; c) the Commodore-64 is where all the action will be.

My answers to those arguments are that a) with the excellent quality of CURSOR, a price of $15 to $20 would not be prohibitive; b) it may be more of a challenge to acommodate 40 or 80 columns, or basic 3.0 and 4.0, but if you know that at the beginning of program design, it's not that difficult (don't bother retrofitting the old ones!); and c) it's going to take a while for the population of 64's to build up ... but there are a lot of PET/CBM's in the field owned by quite a few loyal CURSOR customers!

It's good to see the response to Commodore's TV advertising. If they had been this aggressive four years ago, I'm convinced PET/CBM would be the #1 seller in the US right now. Wish we saw some more emphasis on the larger machines in the ads, too ... VIC is good for many users, but others really need the "full-size" models. Watching the stock market this past week, I saw Commodore stock go up 10% in an othwise listless market. Someone believes!

Responding to the remark about "one good word processor", check this: the March 82 issue of Popular Computing ran a comparison of word processors ... all the popular machines, including CP/M ... guess which came out looking best!? WordPro 4+ by Professional Software, Inc., running on the CBM 8032.

Background/Foreground

The Mail-Order Menace note hit close to home. Having the experience of being a Commodore dealer has given me a new perspective. If you want the covenience and service of a dealer, you've got to be willing to pay for it. The reason a mail order house can sell cheaply is because it avoids the expenses of showrooms, clerks and customer education. If you don't need to see what you're going to buy, if you don't need to ask questions or be helped in deciding on the best system, then go mail order. Don't, however, use up the dealer's time getting educated, then buy mail order, and expect your dealer to be waiting with open arms when you show up for warranty repairs a week after the machine arrives.

Are you serious about patenting software? Copyright may be "next to useless", but it beats waiting seven years for a patent. How many programs would qualify for patent? In either case, the only "protection" you get is a little more advantage when you take the violator to court!

Reader I/O

For Stan Spence, a question and a comment. When you ordered Supermon, did you specify that you had an 8032? AB Computers is one of the best mail-order suppliers ... Gene Beals knows his stuff, and his company is reputable. Have you called them? I'm confident they'll satisfy you.

Tim, regarding your problem with the MX-80... it's either switch settings in the MX-80, or commands in WP3. I'm using and 8032 with wp4+ with no problems ... will try the other combination and let you know how it works.

Please John Nuttall ... give us the price! I've been waiting for someone to put out a PET graphics chip for the MX-80 ... don't make us wait for another issue. Ralph, I'll even pay the $2 if you get the info and publish it!

Daniel Condon: the hardware/software package to read (or transmit) RTTY (or CW) is made by an outfit called MACROTRONIX ....

Bravo to Jim Strasma for his fervor and zeal in defending Commodore against the slings and arrows of outrageous Bressler! I was very much tempted, Jim, to write such a letter myself ... but I decided that the "Late Great Commodore" editorial was so offensive to so many that it would ellicit scores of indignant responses. One question: where can I buy a legal copy of 96K Visicalc? Yes, I know it exists, but I'm told that VisiCorp (ex Personal Software) denies knowledge of it! (Pub. Note: Jim now says that VisiCorp has graciously acknowledged that it exists and will distribute it. Contact them!)

It was good to see Gary Stone's suggestion regarding the name, "The PAPER". My suggestion would be to call it "The PET/VIC Paper" with a smaller caption line covering "all Commodore computer products". It's disappointing to learn that mighty COMPUTE considers the tiny PAPER to be a rival ... I've always considered newsletters and magazines to be complimentary.

Marilyn Nicholson - don't throw away your old issues of The PAPER! I've had many beginners report great success when they go back 3 to 6 months later and re-read the articles they didn't understand earlier. The beginner's problem has been described as being like trying to get a sip of water from a fire hydrant ... ther's just too much, coming too fast, to be able to get what you want!

Observations

Oops! Here's a typo I hope got fixed! Couldn't remember what and "inflation loop" was ... turns out it had said "infinite loop" at first. Oh, well!

Oops 2! Haven't yet written the follow-up article. Too busy with income tax, college aid forms, etc., etc., etc.!

Oops 3! Be careful what you plug into MTU sockets. ROMs like Visicalc are OK, but EPROMS (many others) will be destroyed if plugged into that board. (Pub Note: I have plugged several 2716 and 2532 EPROMs into this board for a short time to record the code and have had no troubles. MTU does give a socket modification if you ask them about the problem.)

New Products

The exciting news from Commodore at this moment (April 82) is the Commodore-64, the 8250 disk drive, and the D9090 and D9060 Winchester drives.

At $595, the 64 should be a big winner. According to the Commodore fact sheet, it has 64K memory, CP/M software option, RS-232, and enhanced audio and hi-res capabilities, as well as the best PET/VIC features.

The 8250 is a double sided 8050, doubling storage to 2.1 megabytes (on two diskettes) for only $2195, a cost increase of only 22%! More exciting, it will allow relative files over a megabyte in size (versus the 8050's 183 kilobytes).

The D9090 provides 7.5Mb hard-disk storage for $3495, while the D9060 gives 5.0Mb for $2995. Each claims an unlimited number of file names, and relative files almost as long (98%) as the total disk capacity.

No Jabberwock!

It would be interesting to know where A H McCann lives, and whether he has a Commodore dealer nearby. Being forced to buy mail order is one thing ... but if he chose to mail-order versus dealer purchase because of price difference, then his is a self-inflicted wound.

In either case, his diatribe is uncalled for! The combination of 2001-32B, 2040, MX-80 F/T, and Word Pro 2 (or 3) works just fine! Why Mr McCann, did you call Commodore and Epson but not Professional Software, manufacturers of Word Pro? It's their software that makes the system work! Word Pro 2 has two main programs .. Editor (for CBM printers) and ASC Editor (for all others). Very simply, you were using the wrong program! EPSON's IEEE board works fine, too. I would be willing to wager that there are more MX-80's being used with Commodore computers and word processors than any other.

Monitor This

Better article this time, Jerry ... just one or two comments.

Repeated use of Break (versus Call) entry does make a difference. Do it often enough and you'll run out of stack storage space, and get an "OUT OF MEMORY" error. Get in the habit of using SYS 54386 (BASIC 4.0) or SYS 64785 (BASIC 3.0) all the time.

A nitpick: when you do a SAVE from the monitor, the SAVE COMMAND specifies one location past the end of your program, but the data actually saved does not include that extra byte.

Several other cautions: the first location of a machine-language routine is not always the SYS address to activate it. Also, combinations of BASIC and ML can be saved from BASIC only if the combination starts at $0400. If the ML is in the 2nd cassette buffer, you'll have to use the monitor to save the combination.

Armor Plate, No Less!

"Histogram", by Doug Haluza, might be more accurately titled "BASIC Programming Techniques" ... if you failed to read the article, go back and do it now. It's got a lot of good material in it.

Doug points out that a histogram is easier to interpret than a list of numbers. The same can be said for most graphic displays ... including the simple act of printing ANYTHING during a long computation, just to show that the computer hasn't quit!

Providing a default input is an easy addition that makes the program a lot "friendlier". Please, however, join my crusade to get instruction writers to say PRESS return rather than HIT return. Novices tend to take instructions literally!

Forgiving mistakes is another essential ingredient of quality software ... but it adds a LOT to the time it takes to build a program.

I'll argue with Doug's assertion that it's impossible to completely bulletproof a program. It just takes a lot of testing, by experts. An example of the technique is "Cheque-Check", marketed by MicroSoftware Systems, Inc. (I designed and implemented the program.) My feeling is that total bulletproofing is worthwhile only for programs intended for use by novices. Some limited protection should be applied to any program, though.

Regarding Device #3 (screen) and Device #0 (keyboard), caveat all remarks with "most times" or "usually". Using the file PRINT# and INPUT# invokes routines that don't always work exactly the way you want them to. The technique is good, but be prepared for some surprises.

Random Thoughts

Mike Todd's (where have I heard that name before?) "final word" on the RND function was a good article, and quite interesting. Final it is not! Encore, Mike! Now that you've explained the "what" (rather briefly), explain the "why" of those funny numbers. They must have some esoteric significance!

Relatively Accurate

Without detracting from the obvious value of Glenn Davidson's "Relative Files: Part I", there are several things to be noted.

Sequential files will continue to be used, for several reasons.

a. There are still 2040's around.

b. A set of short sequential files can be accessed almost as quickly as a long relative file.

c. Sequential files do not have a maximum size limitation, as do relative files (the new 8250 will effectively do away with that limitation).

Item #4 in the "Keep in mind" list is incorrect. PRINT# does indeed move the record pointer ahead one place each time it is used. INPUT# does so ONLY if there are no CHR$(13)'s embedded in the record. Otherwise, the record pointer is not advanced until all the fields in the record have been retrieved.

Say It In Machine Language

Hate to see the end of a good series but I hope Bressler's articles will be followed up by some well-documented machine language programs.

Ralph, you keep tossing out fascinating hints, but failing to provide details! What are the dozen good assemblers, who sells them, and how do they differ? I've been using the ones from Eastern House Software, and been very pleased with them. They're fast and offer a lot of features.

It would be helpful if you also presented each routine in its Monitor listing form. That makes it a lot easier to enter, if one does not have an assembler.

$FFD2 is not the only "routine" that doesn't move around from one version PET to another. Actually, $FFD2 contains a pointer (or vector) to the real screen output routine, which DOES change locations. There are at least 8 or 10 other vectors in the same area ... all provided so there will be unchanging reference points for the more popular routines.

In Reflection

It has taken (I can't believe it!) over six weeks to write this column! Perhaps I really do have too many irons in the fire. Better late than never, I suppose.

SuperPET still seems to pose the most challenges ... even if just in figuring out how to do what we already knew how to handle on the other PET/CBM machines. It will be interesting to see what's new (and what's old) with the Commodore 64.

More exciting times are ahead if we can just keep up with them.

Graphics and Animation in BASIC: Part II

by Ralph Bressler

In the first installment in this series, we examined how to create pictures using simple print statements. That article also illustrated how to animate graphics or get them to move horizontally and vertically. In this part we will see how to create a game using PRINT statements. The use of POKE and PEEK will also be introduced.

Many games we have seen consist of moving an object around the screen to avoid something chasing you and to capture something else. To begin the second part of this article I would like to present a method of doing this with PRINT statements and an X,Y coordinate system. Let's say that the upper left corner of the screen is 1,1 while the lower right is 38,24. In most of these games we use the numeric keypad to move. Hitting 2 indicates we want to go down a line, 8 up a line, 4 to the left one space and 6 to the right. For now we will ignore the diagonals and simply produce a routine which will keep us on the screen and enable us to move about. What follows is such a short program:

```
100 REM WALK-A-BOUT
110 :
120 PRINT"(clr)"
130 DN$="(home 24down)"
140 :
150 REM START US AT CENTER OF SCREEN
160 X=20:Y=12
170 PRINTLEFT$(DN$,Y)TAB(X)"*"
180 :
190 REM START IT AT RANDOM SPOT
200 TX=ABS(RND(1)*38)+1
210 TY=ABS(RND(1)*24)+1
220 PRINTLEFT$(DN$,TY)TAB(TX)"+"
230 :
240 REM SET BEGINNING TIME
250 BT=TI
260 :
270 REM MOVE USING KEYPAD
280 GET M$
290 IF M$="2" THEN CY=1 : CX=0
300 IF M$="4" THEN CX=-1 : CY =0
310 IF M$="6" THEN CX=1 : CY=0
320 IF M$="8" THEN CY=-1 : CX=0
330 :
340 REM CHECK TO SEE STILL ON SCREEN
350 IF Y+CY<1 OR Y+C>24 THEN 280
360 IF X+CX<1 OR X+CX>38 THEN 280
370 :
380 REM UPDATE POSITION
390 PRINTLEFT$(DN$,Y)TAB(X)" "
400 X=X+CX : Y=Y+CY
410 PRINTLEFT$(DN$,Y)TAB(X)"*"
420 :
430 REM CHECK IF HIT TARGET
440 IF X<>TX OR Y<>TY THEN 270
450 :
460 REM IF HIT SAY SO AND PRINT TIME
470 PRINT"(home)GOT 'EM IN"ABS((TI-BT)/60)"SECONDS"
```

This program is not very exciting since all you have to do is hit one target but it may stimulate some ideas. We use the by now familiar DN$ for vertical movement. Line 250 uses something called the jiffy clock to record the beginning time. A jiffy is 1/60 of a second so this clock is useful for measuring short periods of time. If we record the time at the beginning and the time at the end we can subtract and divide by 60 to get the elapsed time in seconds. Line 280 uses a GET instead of an INPUT to take in our move. INPUT prints a flashing cursor, allows us to enter up to 80 characters at a time and waits until we hit return. GET does not print any cursor, will only accept one character at a time and does not require a RETURN. GET is the better choice for our application. Lines 290 to 320 check to see what direction we wanted to move and set the change in X (CX) and the change in Y (CY). If we are going to go off the screen lines 350 and 360 prevent that movement. If either our X or Y position is different from the target, line 440 sends us back to move again. If we have hit the target line 470 says so and prints out the time.

There are several problems inherent in the method used above and most stem from the fact that we cannot "see" where we are going. The number of differnt targets is limited since we would need two different variables for each to record their position. Checking all these positions is slow. We can only move over clear terrain since we always leave behind a space. Moving on a grid or similar playing field would be hard. Most games that are based on this idea use a pair of statements called POKE and PEEK.

POKE is a BASIC statement which allows a programmer to place a value into a given memory location. PEEK allows us to "see" what is being stored in such a location. The PET has 65,535 different memory locations which are divided among several different functions. Some of these locations are Read-Only Memory (ROM) which stores BASIC and the PET operating system. This is permanent memory and we can PEEK at it but we cannot change what is stored there. The rest of the memory is Random-Access Memory (RAM) where the PET stores its own important information and where we store our programs. Think of these memory locations as a large post office with many boxes. POKE allows us to put mail in the boxes while PEEK allows us to see what is there. The boxes are very small and, therefore, may contain only one piece of mail or one numeric value at a time. These values may only be from 0 to 255. Trying to place a higher or lower value in a location will result in an error. One thousand of these boxes make up the screen memory and they are labeled from 32768 in the upper left corner to 33767 in the lower right. When we put something in a location elsewhere in memory we may or may not be able to see the result. However, when we change the contents of a screen memory location we can see the change. Try typing these commands starting on the second line of the screen:

```
POKE 32768,1:POKE 32769,2:POKE 32770,3
```

This should produce the first three letters of the alphabet starting in the upper left hand corner. You see, each character that the PET can generate, all 256, has a unique code which runs from 0 to 255. We have seen that 1 is the code for A, 2 for B and so on. It may seem that you now have 256 more numbers to memorize but this is not so. Clear the screen and place a "heart" (shifted A) in the upper left hand corner. Hit RETURN and ignore any message that may be generated. Now type: PRINT PEEK(32768). This looks at the contents of the upper left hand corner of the screen and prints the code for what is there. In this case you should have gotten a 65, the code for a heart. So, you need not memorize all 256 codes or even carry around a chart, the PET will tell you any of the codes you need to know. Practice with these commands some before reading further. See if you can POKE your name into the lower right hand corner.

Let's try some simple programs which use POKEs and PEEKs to manipulate the screen. The first program fills each space on the screen with the same character.

```
100 REM FILL SCREEN
110 FOR I=32768 TO 33767
120 POKE I,160
130 NEXT
140 GET SP$: IF SP$="" THEN140
```

In this program you could try changing the character code (160) in line 120. This would fill the entire screen with some other character. You could also change the range of the FOR...NEXT loop in line 110. This would control how much and what part of the screen would be filled. The second program fills each screen position with a random character.

```
100 REM RANDOM FILL
110 FOR I=32768 TO 33767
120 POKE I,INT(RND(1)*256)
130 NEXT
140 GOTO 110
```

The expression after the comma in line 120 chooses a random number from 0 to 255. As we said before, this is the range of numbers which represents the entire PET character set. This program must be stopped by hitting STOP since line 140 makes the process happen again and again. If we wanted to, we could choose both the screen position and the character randomly. This is done in the third program.

```
100 REM RANDOM FILL
110 PRINT"(clr)"
120 CH=INT(RND(1)*256)
130 SP=(INT(RND(1)*1000)+1)+32767
140 POKE SP,CH
150 GOTO120
```

In this version CH is the character code between 0 and 255. The screen position to be filled is chosen in line 130. A random number bewteen 1 and 1000 is added to 32767 to give a number between 32768 and 33767. All this may be fun and allows us to play with the new statements but it's not very useful. The next program puts a border around the screen in a clockwise direction. Some interesting changes can also be made in this program.

```
100 REM SCREEN BORDER
110 FOR B=32768 TO 32807: POKE B,86: NEXT
120 FOR B=32807 TO 33767STEP40: POKE B,86: NEXT
130 FOR B=33767 TO 33728STEP-1: POKE B,86: NEXT
140 FOR B=33728 TO 32768STEP-40: POKE B,86: NEXT
150 GETSP$:IFSP$=""THEN150
```

Since we need to make four lines for the four sides of the screen, we need four FOR...NEXT loops. The loop in line 110 places the top border. The next loop, in line 120, takes care of the right side. Notice that the STEP here must be 40 since we are going down a line or 40 spaces at a time. The STEP in line 130 which puts the bottom on the border is -1 since it starts at the right and goes left. Finally, when we do the left hand side we use STEP -40 since we are going up an entire line at a time. The 86 in the POKE statements is the character code and can be changed to anything you like. For practice you might want to try making the border go the other way by changing the limits and STEPs in the loops. An even simpler change would be to rearrange the loops so that they get done in different orders.

Now we can begin to talk about using POKEs in a program. It is probably true that POKEs should not be used when PRINT statements would be just as good. In general, PRINTs are easier to use and will make your program easier to move from machine to machine or from BASIC to BASIC. We saw at the beginning of this article that creating a "crash-into-the-targets" game is hard to do with PRINTs since a large array is needed to map out what is on the screen and updating this is slow. POKEs work well to accomplish this task after a few basic principles are mastered. When we move we must remember that moving "down" one line is really equivalent to moving 40 spaces or adding 40 to our current screen position. Thus, if we are at 33200 and move down a line we are at 33240. As before, we imagine that we are at the 5 on the numeric keypad and the other keys move us in the directions we want to go. The chart below shows the key, the direction we move, and how much this changes our position.

Key	Direction	Change
1	down and left	+39
2	down	+40
3	down and right	+41
4	left	-1
6	right	+1
7	up and left	-41
8	up	-40
9	up and right	-39

The game we will construct has the "player" trying to hit a target on the screen. Only one target appears at a time but the player must avoid hitting the walls and an ever increasing number of mines. A target is worth 100 points but hitting the wall subtracts 200. A mine is instantly fatal. A total time of two minutes is allowed. A constant display of time and score is provided. All of these rules are abitrary and all could be enhanced but this provides an easy introduction. After we present and explain the entire program we will suggest some changes which may make the game more interesting.

I had considered presenting various parts of the program one at a time and giving explanations as I went. This approach fragments the program, so here it is in its entirety followed by an explanation of each routine.

```
100 REM ***CATCH 'EM***
110 :
120 PRINT"(clr)"
130 REM PLACE BORDER
140 FORB=32768TO32807:POKEB,102:NEXT
150 FORB=32807TO33767STEP40:POKEB,102:NEXT
160 FORB=33767TO33728STEP-1:POKEB,102:NEXT
170 FORB=33728TO32768STEP-40:POKEB,102:NEXT
180 :
190 REM PLACE PLAYER
200 PP=(INT(RND(1)*1000)+1)+32767
210 IF PEEK(PP)<>32 THEN 200
220 POKEPP,81
280 :
290 BT=TI:REM SET BEGINNING TIME
300 :
370 REM MOVEMENT
375 IF TF=0 THEN GOSUB 3000
380 GETM$:GOSUB2000
390 IF M$="2" THEN CH=40
400 IF M$="4" THEN CH=-1
```

```
410 IF M$="6" THEN CH=1
420 IF M$="8" THEN CH=-40
430 IF PEEK(PP+CH) = 87 THEN SC=SC+100:TF=0
440 IF PEEK(PP+CH) = 102 THEN SC=SC-200:GOTO 370
445 IF PEEK(PP+CH) > 147 THEN SC=SC-200:GOTO 370
450 IF PEEK(PP+CH) = 42 THEN 500
460 POKEPP,32:PP=PP+CH:POKEPP,81
470 GOSUB1000
480 GOTO370
485 :
500 REM BLOW UP
510 ET=TI:PRINT"(clr)(2down)YOU HIT A MINE!(down)
520 PRINT"WITH"INT((ET-BT)/60)"SEC. LEFT(2down)":GOTO620
600 REM TIME UP
610 PRINT"(clr)(2down)YOUR TIME IS UP!(2down)
620 PRINT"YOU SCORED "SC"POINTS"
625 FORI=1TO10:GETSP$:NEXT
630 END
640 :
1000 REM PLACE MINES
1010 IF RND(1)>.1 THEN RETURN
1020 MP=(INT(RND(1)*1000)+1)+32767
1030 IF PEEK(MP)<>32 THEN 1020
1040 POKE MP,42: RETURN
1050 :
2000 REM DISPLAY TIME AND SCORE
2010 IF(TI-BT)/60>120THEN600
2020 TM=INT((TI-BT)/60):TM$=MID$(STR$(TM),2)
2030 SC$=MID$(STR$(SC),2)
2035 IFSC<0THENSC$="-"+SC$
2040 PRINT"(home)(3right)(rvs)SCORE = (off)(5shift&)(5left)"SC$;
2045 PRINT"TAB(28)"TIME = "Tm$
2050 RETURN
2060 :
3000 REM PLACE TARGET
3010 TP=(INT(RND(1)*1000)+1)+32767
3020 IF PEEK(TP)<>32 AND PEEK(TP)<>42 THEN 3010
3030 POKETP,87:TF=1
3040 RETURN
```

In lines 120 to 170, we clear the screen and use the border routine we developed above to define our playing area. Line 200 chooses a random place to put our player on the screen. If that place is not a free space (code 32), line 210 causes a new place to be chosen. Otherwise, line 220 places the player on the screen. Line 290 records the beginning time by "capturing" the value of the jiffy clock at the beginning of the game.

Lines 370 to 480 are the main loop of the program from which different routines are called. In line 375 we check to see if the target flag (TF) is zero, which would indicate no target is present. When there is no target, we go to the routine at 3000, which first chooses a random place for the target. Line 3020 checks to see that this place is occupied and causes another to be picked if it is. If it is empty, the target is placed and the target flag is set. Back at line 380 we GET the player's move and then access the subroutine at 2000. This routine first checks to see if the time is up by comparing the time at present (TI) to the time when we began (BT). If this quantity is greater than 120 seconds, then the game is over. Otherwise, line 2020 calculates the time elapsed and converts it to a string, which is easier to control when printing. Lines 2040 and 2045 print the score and time on the top line of the screen.

When we RETURN to lines 390 to 420, we check to see what move the player entered and set CH to the proper value. Notice that CH is never set to zero so that if no new move is entered CH will retain its old value and the player will continue to move in the same direction. This makes the game a little harder particularly when more mines show up. To make the player press a key for each space he moves, you could change line 380 to:

```
380 GETM$:GOSUB2000:IFM$=""THEN380
```

LIne 430 to 450 use the PEEK command to "look ahead" to see what is in the space we wish to occupy. Since line 460 does the actual move, we don't change position until PEEK checks out the territory. Line 430 checks to see if we will hit a target and gives us the points we deserve and also sets the target flag to zero. When TF=0 a new target will be placed when subroutine 3000 is called next time. If it's not a target, lines 440 and 445 check to see if it is a wall and punish us as necessary. The GOTO at the end of these lines prevents us from actually entering a wall. Line 450 checks for mines and immediately sends us to line 500 to indicate our demise if we should hit one. If everything goes OK, line 460 blanks our old position, changes our position and then places us there. When line 470 calls the mines routine, a mine is placed about 10% of the time. Line 1010 takes care of this. The rest of the routine chooses a random space, sees if it is free and places the mine or chooses a new position. Line 480 causes this loop to repeat.

The game ends when we hit a mine or time runs out. The lines from 500 to 630 end the game with 510 and 520 informing us of our ill fortune at being blown to bits and the time we had remaining. Lines 600 to 620 report that time has expired but we have not and gives us a score. Line 625 is important and should be included in all games of this type. This line clears the keyboard buffer of any keys we have pressed in the frenetic course of the game. Without this line, fours and sixes would appear after the game has ended and cause problems.

I am not suggesting that this is easy or even completely understandable the first time around. Perhaps the biggest problem is when to choose PRINT statements and when to use POKEs. Our program could be enhanced by adding sounds to the movements, the capturing of targets and the explosion of mines. The sounds can be added in subroutines quite easily. First a little background on sound. If you have a new Fat-40 or 8032 machine you may have noticed that it chirps when it is turned on which means you have a built speaker. Those of you with older machines will have to connect a speaker and amplifier to the CB2 line of the user port. This is explained in many books and magazines and set ups are available at most dealers for a modest price. Once things are connected POKEs are used to turn on the sound and generate the notes. Here is a table of the POKEs which are needed:

```
POKE 59467,16      - turns sound on
POKE 59466,81      - sets "pitch" of sound; any number can be used
                     but 15 and 81 sound best
POKE 59464,X       - chooses individual note where X is any number
                     between 0 and 255
```

All sorts of interesting sounds can be created by experimenting with loops and time delays and random notes. One caution: turn the sound off at the end of the program with POKE 59467,0 or your tape deck will not operate properly. To add a beep for each space we move in the program the following additions would be sufficient.

```
125 POKE 59467,16: POKE 59466,18
465 POKE 59464,0
```

Just as the screen can be randomly filled, music or noise can be produced randomly. Here is the rawest example of strictly random noise:

```
100 POKE 59467,16: POKE 59466,81
110 N=INT(RND(1)*256)
120 POKE 59464,N
130 GET SP$: IF SP$="" THEN 110
140 POKE 59467,0
```

This will continue until you press a key. Each note will have the same duration since there is no time delay. To add a random delay type in these lines:

```
125 TD=INT(RND(1)*100)+1
126 FOR I=1 TO TD: NEXT
```

Of course, variations on random noise are possible. These include controlling the range of noise, modifying the random number, or setting the duration of the effect. Here is such an example:

```
50000 REM VARIATION
50005 POKE59467,16: POKE59466,15
50010 FORL=1TO30:POKE59464,10+100*RND(1)
50030 FORI=1TO6:NEXT:NEXT:POKE59467,0:RETURN
```

Notice that I have used large line numbers and a RETURN at the end. This effect could be used as a subroutine within another program. To run it as is, just take out the RETURN.

Through experimentation, people invent sounds that resemble certain noises or are just handy. Here's a simple one that sounds ominous:

```
20000 REM OMINOUS
20010 POKE59467,16:POKE59466,10
20020 FORB=1TO100:POKE59464,255:POKE59464,200:
20030 NEXT:POKE59467,0:POKE59466,0:RETURN
```

Continued experimentation can produce more and more complex effects like this:

```
40000 REM COMPLEX
40010 POKE59467,16:POKE59466,85
40020 FORL=1TO10
40030 FORL1=1TO14:POKE59464,L1*16:NEXT
40040 NEXT:POKE59467,0:RETURN
```

Most of the interesting sound effects that you may have heard coming from a PET are the result of trying out certain things. You should experiment also to see what you come up with.

Patience and a little musical ability can even produce some respectable songs. These, of course, are played a single note at a time. In their simplest form these songs are just data from which notes are read and then played. With a little practice time delays can be added to make the overall effect even better. Here is a simple "song":

```
100 POKE59467,16:POKE59466,15
110 READ N: IF N=-1 THEN POKE59467,0:END
120 POKE 59464,N: GOTO 110
130 DATA 100,45,210,95,100,74,25,12,250,-1
```

If you play this you will notice it doesn't resemble any tune you know. It

shouldn't, anyway! The program below is considerably more elaborate with three pieces of data for each sound. This data includes the note, its duration and how long to wait before the next note.

```
30000 REM SONG
30005 RESTORE
30010 POKE59467,16:POKE59464,0:POKE59466,104
30020 READ C,T,Z
30030 T=T*20:Z=Z*20
30040 POKE59464,C:FORI=1TOT:NEXTI
30050 POKE59464,0:FORI=1TOZ:NEXTI
30060 IFZ>0THEN30020
30070 POKE59466,0:POKE59467,0:RETURN
30080 DATA237,10,1,237,5,1,177,20,5,237,10,1
30085 DATA177,5,1,140,20,5
30090 DATA237,10,1,177,5,1,140,15,2,237,10,1
30095 DATA177,5,1,140,15,2
30100 DATA237,10,1,177,5,1,140,25,5,177,10,1
30100 DATA140,5,1,118,25,1
30110 DATA140,20,1,177,15,1,237,25,1,237,20,1
30110 DATA237,10,1,177,25,0
```

Notice, again, that this has been coded as a subroutine for inclusion in another program. C is the note, T its duration, and Z the time before the next note.

We could use several of these routines in our target program. Load or type in the target program first. Type in OMINOUS, COMPLEX and SONG using the line numbers they already have. Now add this line:

```
515 GOSUB 30000: REM SONG
```

Also add GOSUB 40000 to the end of line 430. To lines 440 and 450 add GOSUB 20000 before the GOTO 370 at the end. You may begin to see how a very complex program can be built up a little at a time.

I realize this is an article about BASIC graphics but there are some things that BASIC cannot do becuase it is so slow. Games which require many things to move within a short period of time are impossible to do in BASIC. Once you have tried the examples in these articles and have begun to write your own programs you may want to attempt the inclusion of some machine language rotuines in your programs. For example, I wanted to fill every blank space in a maze with dots for a PACMAN-like game. A very short machine language routine did the trick. Here is that simple routine as DATA in a BASIC program.

```
10000 FORI=912TO942:READN:POKEI,N:NEXT:RETURN
10010 DATA169,128,133,2,162,0,134,1,160,0,177,1,201,32
10020 DATA208,4,169,46,145,1,200,208,243,230,2,232,224,4,208,234,96
```

Anytime you want to fill the blank spaces on the screen with dots just give the command SYS912. Another time I wanted to switch very quickly between two full screen images. PRINTing and even POKing in BASIC were too slow so I programmed SWAP-PUT, a series of short machine language routines in the last issue. You do not have to know how to write your own machine language routines since many are available. Several packages have even been printed in magazines to allow VERY quick screen scrolling in almost any direction and other interesting capabilities.

In closing, let me say again that there is no substitute for practice and experimentation. Don't be afraid to try things out and ask questions. Everyone started some time and most people will be glad to help.

The MONITOR

by Jerry Keys

This column will attempt to give relatively new users of the Machine Language Monitor a small understanding of what we can do with the MLM even if we won't normally get into machine language programming. The column will normally deal with Basic 4.0 but every attempt will be made to show the equivalent Basic 2 (03 ROMs) in parentheses.

Roy Busdiecker's point on moving too fast is well taken so I will try to concentrate on a little less in this issue.

Let's try and take a look at identifying the end of basic a little more closely this time. Along with it we'll do a little more identification of things like the end of a basic line.

So clear your machine and type in the following:

```
10 REM THIS IS A TEST FOR END OF BASIC PROGRAM
20 REM ONE MORE LINE TO SHOW DIVISION OF BASIC LINES
```

Now type SYS54386(SYS64785)
You should see the following display on the screen:

```
C*
PC  IRQ  SR AC XR YR SP
.;  B780 924D 34 33 38 36 FA (if not exactly same don't worry)
.
```

Now type .M 002A 002A(rtn)
This should be your display:

```
.:  002A 65 04 65 04 65 04 00 80
         -----
           ! this is the pointer to the end of the basic
             program. Remember, it's low byte first(65)
             then high(04) so this location 0465.
```

Then type this:

```
.M 0400 0465 (start of basic to end of program)
```

In the following display of what you will see, we are going to break up the lines for discussion purposes.

```
.:  0400 00 2F 04 0A 00 8F 20 54
    ---- -- -- -- -- -- -- -- --
      !   !  !  !  !  !  !  !  !=T
      !   !  !  !  !  !  !  !=space in hexidecimal
      !   !  !  !  !  !  !=rem
      !   !  !  !  !  !=always a space after line number
      !   !  !  !  !=number 10
      !   !  !  !=hi byte for next basic line 042f
      !   !  !=Low byte for next line of basic
      !   !=0400 is always 00
      !=Starting location number
```

So you can see from this line where the basic has started. The following lines should look like this:

```
.:  0408 48 49 53 20 49 53 20 41
.:  0410 20 54 45 53 54 20 46 4F
.:  0418 52 20 45 4E 44 20 4F 46
.:  0420 20 42 41 53 49 43 20 50
.:  0428 52 4F 47 52 41 4D 00 63
                           --
                            !=00 always breaks basic lines.
                              In this case, look up above and
                              you will see the next location
                              (042F) referred to.
```

Now, using the last byte of the line shown above and the first byte of the next line, we will have our pointer to the next line of basic.

```
.:  0430 04 14 00 8F 20 4F 4E 45
```

So the pointer is 63 04 or 0463. So look at the rest of the dump:

```
.:  0438 20 4D 4F 52 45 20 4C 49
.:  0440 4E 45 20 54 4F 20 53 48
.:  0448 4F 57 20 44 49 56 49 53
.:  0450 49 4F 4E 20 4F 46 20 42
.:  0458 41 53 49 43 20 4C 49 4E
.:  0460 45 53 00 00 00 AA AA AA
    ---- -- -- -- ----- --------
      !   !  !  !   !       !=normal code from power up test
      !   !  !  !   !=two 00 bytes for end of basic
      !   !  !  !=end of line marker
      !   !  !=S
      !   !=E
      !=start of line location.
```

What can be determined here is that basic (in the monitor) will always start each line with a pointer to the next line to be executed, even before the line number. Then the line number, then always a 00 space (even if you don't want it, try and get rid of it), next, it finally covers the text with another 00 at the end of the line. Then, if it is the end of basic, a 00 00 is appended so to find the end of basic, search for a 00 00 00 (the first set) and most of the time you will have found the end of basic.

It would be worth the exercise to translate each byte from hexidecimal to decimal to compare the basic lines with the decimal dump. You may find that you can find those hidden goodies that are tacked onto the front of some programs by this method. It may be slow and painstaking but chances are you need the practice.Also remember, if you are looking for the end of basic in a program that you think has machine language attached, make sure that the 00 00 00 series you find is the FIRST occurrence of it.

A small update on the last issue. You can load the machine language portion first when attaching ML and Basic, just make sure you type new before loading the Basic or you may get some funny happenings.

Just for fun, type SYS64790(SYS64721). Can you think of a use for that ?

Relative Files: Part II

by Ralph Bressler

First, let me apologize to Glenn Davidson for not saying that I extensively editted the first part of this article which he wrote. I hope my effort made his material more understandable and did not introduce any errors or misconceptions. I felt that this topic needed a second part to explain some of the finer points of relative files which make them very useful.

The word "record" is tossed around quite a bit when we speak of files and is sometimes used to mean different things. In a data base, a record is all the information about a certain item or person and usually includes several fields containing different pieces of data. The information for a mailing list would include a person's name, street address, city, state and zip code. All of this would be a record with each individual piece of data being considered a field. When we begin to write a program to handle a "data base record", we run into the term record as used by a programmer. A programmer considers a record to be, in the case of the CBM system, a set of up to 255 contiguous bytes of storage. This means that each "physical disk record" can contain up to 255 characters of information. Of course, as we saw last time, the length of a physical disk record can be set using a parameter like L50 when opening a relative file. In any case, most programmers try to keep the physical record number on disk the same as the data base record number. In other words, the first person in the data base has record #1 on disk, the second #2 and so on.

The last few sentences above describe the ideal situation which is not always possible. Sometimes a data base record may require more than 255 bytes of information. This might be the case when we have a lot of information to include such as a long comment. Let's suppose that we can fit the person's name and address in the first physical disk record and the comment in a second. This means that the first person takes up records one and two while the second person occupies records three and four. It is relatively simple, then, to remember to "offset" the record pointer each time we want to access the information for a person. Suppose we have written a file in such a manner for 100 people. The program below would access the information:

```
100 DOPEN#1,"PEOPLE",D0
110 INPUT"PERSON #: ";PN
120 IF PN<=0 OR PN>100 THEN PRINT"GOODBYE!": DCLOSE: END
130 RP = 2*PN-1
140 RECORD#1,(RP)
150 INPUT#1,NA$,AD$,CY$,SA$,ZC$
160 INPUT#1,CM$
170 PRINT"(clr)PERSON #"PN: PRINT
180 PRINT"NAME    : "NA$
190 PRINT"COMMENT : "CM$
200 PRINT: GOTO110
```

Numerous problems can arise here and this technique can be wasteful. All the initial address information MUST fit into the first record and cannot overflow into the next. The bigger problem is that the size of each record must be constant. This means that you are wasting space when a short address or comment is used and cramping your style when longer ones are needed. For this reasons most data bases are limited to record lengths of 255 characters.

Before we go on another point should be cleared up. When PRINT# is used it advances the record pointer in the file for each separate occurrence. Each physical record on the disk is terminated with a carriage return, although carriage returns may be inserted by the programmer to divide the record into fields. If you try to write 86 characters into a record set for a length of 85,

85 characters will be written and a carriage return will NOT show up at the end of the record. This will also generate a "record overflow" error. In this case, you will still be able to read the information back correctly but the data will be truncated to the record length that was set. Look at the following partial program:

```
200 RECORD#2,12
210 PRINT#2,NA$CR$AD$CR$CY$
220 PRINT#2,CM$
```

The name, address and city would be printed in record 12 and would be separated into fields by carriage returns and a return would follow the city name if the total length is less than the record length. The comment would be printed into record 13 since the PRINT# in line 210 advanced the record pointer once. The comment would be terminated by a carriage return if it is less than 255 characters and anything printed after line 220 would be entered in record 14 unless the pointer was reset.

INPUT# works a little differently since it only advances the record pointer when it comes to the end of a physical disk record. Carriage returns within a record terminate a certain field but DO NOT advance the record pointer. Look at the following part of a program:

```
410 RECORD#1,15
420 INPUT#1,NA$,AD$,CY$,ZC$
430 INPUT#1,CM$
```

Line 420 will attempt to read four fields of information from record 15 since the pointer was set to that record in line 410. If all four pieces of information are found in record 15 and the end of the record is reached then the pointer will advance by one and CM$ will be read from record 16. If the first two pieces of information (NA$ and AD$) are found in record 15 and the end of the record is reached, then the pointer will advance to 16 and input will procede from there. If all four pieces of information in line 420 are input and the end of the record is not reached then the pointer will NOT advance and CM$ will be looked for in record 15. If these concepts seem confusing, read them again because I feel it is important to understand these before going on to more applications.

The fields within a record may be written and read in several different ways. If you look back at the programs we set up last time, you will see we typically set a record length of 85. We then used a PRINT# statement like the one below to write our information into the file:

```
110 PRINT#1, MN$CR$NA$CR$AD$CR$CY$CR$SA$CR$ZC$
```

This would write the various fields of each data base record into one physical disk record and delimit them using carriage returns. Each field could vary in length from record to record as long as the total length (don't forget carriage returns) did not exceed 85. By the way, on the very first page of the first part of this article I made an error regarding record length. I added the lengths of the fields we required, got a total of 82 and decided a record length of 85 would be sufficient. In fact, if you used up the full 82 characters AND the carriage returns between each field, the length would be 87 and an overflow error would result. This would only be obvious with records that used the full 82 characters. In any case, be sure to count the returns since the system does! To get the information back we just depended on the fact that a carriage return terminates an INPUT so that the following line does the job:

```
310 INPUT#1, MN$,NA$,AD$,CY$,SA$,ZC$
```

There are other ways to do this which may, in some applications, be beneficial. We may want each field to be a fixed length no matter what the length of the actual data. For example, the name field should always be 20 characters even if the name is only nine. The additional characters are filled in by spaces. Borrowing a program for writing a simple mailing list from last time, we might change it to look like this:

```
10 SP$="(25 spaces)"
15 DOPEN #1,"MAILING LIST",D1,L85
20 CR$ = CHR$(13)
30 INPUT "NAME";NA$
40 IF NA$="END" THEN CLOSE 1: END
45 NA$ = LEFT$(NA$+SP$,20)
50 INPUT "STREET ADDRESS";AD$
55 AD$ = LEFT$(AD$+SP$,25
60 INPUT "CITY";CY$
65 CY$ = LEFT$(CY$+SP$,20)
70 INPUT "STATE";SA$
75 IF LEN(SA$)<>2 THEN 70
80 INPUT "ZIP CODE";ZC$
85 IF LEN(ZC$)<>5 THEN 80
90 RN = RN+1: MN$=STR$(RN)
45 MN$ = LEFT$(MN$+SP$,5)
100 RECORD#1, (RN)
110 PRINT#1, MN$CR$NA$CR$AD$CR$CY$CR$SA$CR$ZC$
120 GOTO 30
```

Now each field has a fixed length but can still be accessed as we did before using the INPUT# statement.

From the way we set up our file above we know that each field will not only have a fixed length but will start at a specific byte in each record in the file. The chart below shows this layout:

Bytes	Variable	Length
1 - 5	MN$	5
7 -26	NA$	20
27-51	AD$	25
53-72	CY$	20
74-75	SA$	2
77-81	ZC$	5

Note that one byte is "skipped" between each field since a carriage return is used to separate the fields. Now all this would be quite futile if we had no way to set the record pointer to a specific byte. The CBM relative file system allows us to not only set the pointer to a record but to a specific byte within that record. We may then read from or write to that byte. In this case, let's assume we only want to read zip codes, so we want to read starting at byte 77. The following program reads and prints the zip codes from the file we created above:

```
10 DOPEN #1,"MAILING LIST",D1
20 RP=RP+1
30 RECORD#1,(RP),77
40 INPUT#1,ZC$
50 IF LEN(ZC$)<>5 THEN DCLOSE: END
60 PRINT ZC$: GOTO 20
```

Notice how the second parameter in the RECORD statement in line 30 sets the "byte pointer". If this number were not constant, it too would have to be in parentheses. Now that I have presented this, I must say that it offers only a fraction of a second's advantage over reading all the fields of the record using INPUT#. It also adds an extra complication which may not be necessary.

If reading from a specific byte offers questionable advantage, writing to a specific byte may offer less. It might seem at first glance that writing to a record starting at a specific byte would offer something since you would avoid writing the entire record. This is not true! Writing to byte 20 in a record DESTROYS everything after what you write. In our example, trying to just change a person's street address would wipe out their city, state and zip code! The only choice then is to read all the fields of a record, change the one you are interested in and then write them all back. It seems to me that you may as well forget reading and writing using this byte method since they offer little advantage and introduce some complexity. Many books, including the Commodore manual, emphasize this and I wish someone would suggest an application where it is a real advantage.

I would now like to turn my attention to a "real" problem with the mailing list or data base program. We will probably want to keep our mailing list in some order like alphabetically by last name or in zip code order. To do this we do not want to have to enter the names in that specific order nor do we want to type in the same list twice in different orders. What we want to do is enter the list once and be able to print it out in either order. Every time we enter a name then we must find its position in the list, insert it in the correct place and push all the other data down. This takes longer and longer as the list grows, does not allow for different orders and makes deletion difficult.

The solution to this problem is to create a "key file" with numbers that tell the order of the records in the main "data file". When data is added to the main file it is simply appended to the end. Only the numbers in the "key file" are shifted making the operation faster and allowing for several different "orders". A key file might look like this:

```
1 3 6 2 4 5
```

What this says is: when outputing the data, print record one first, record three second, record six third and so on. When we add record seven, we search through the main data file and find that record seven should be printed third. Our key file then looks like this:

```
1 3 7 6 2 4 5
```

The "simple" program presented below allows adding of the data in this manner. First, we search the main data file to find where the new data belongs. We begin our search with the first record in our key file and keep searching until we find the correct position. We then "open up" a space in the key file by pusing down all the other numbers. Finally we write the current record number into the key file and append the new data to the main data file.

Deleting a name from the main data file can be done quite easily. The simplest way is to search for the name to be deleted and change its key number to a zero. Now, whenever we print names we ignore any records with zero keys. As we add and delete more and more records we begin to waste a lot of space since we never truly delete the record. The best method is to reuse the space once reserved for the records we no longer want. This is fairly easy to do but difficult to explain. The program below uses the most straightforward, easiest to explain approach. However, it may be the most ambitious attempt I have ever made in explaining a programming technique. It is unlikely I can answer all your questions so feel free to drop me a note if anything is unclear.

```
100 DOPEN#1,"TEACH LIST",D0,L50
105 DOPEN#2,"TEACH KEY",D0,L5
110 CR$=CHR$(13)
115 PRINT"(clr)MAILING LIST(down)"
120 INPUT"(rvs)S(off)TART A NEW FILE OR (rvs)O(off)PEN AN OLD ONE";SP$
125 IF SP$="S" THEN 175
130 INPUT#2,MK
135 REM MK IS MAXIMUM KEY NUMBERS
140 FORI=2TOMK+1:RECORD#2,(I)
150 INPUT#2,T:IFT<>0THENAK=AK+1
160 NEXT
165 REM AK IS ACTUAL KEY NUMBERS
170 :
175 REM MAIN MENU
180 PRINT"(clr)OPTIONS": PRINT
190 PRINT"ADD
200 PRINT"DELETE
210 PRINT"PRINT
220 PRINT"QUIT
230 GET OP$:IFOP$=""THEN230
240 IFOP$="Q"THENRECORD#2,1:PRINT#2,MK:DCLOSE:END
250 IFOP$="A"THENGOSUB1000
260 IFOP$="D"THENGOSUB6000
270 IFOP$="P"THENGOSUB2000
280 GOTO180
290 :
1000 REM ADD NAMES
1010 PRINT"(clr)"
1020 MK=MK+1:AK=AK+1
1030 PRINT"(down)(rvs)ADDING #(off)"AK
1040 INPUT"NAME       :";NA$
1050 IFNA$="END"THEN1120
1060 INPUT"ADDRESS    :";AD$
1070 INPUT"CITY       :";CY$
1080 INPUT"STATE      :";SA$
1090 INPUT"ZIP CODE   :";ZC$
1100 RECORD#1,(MK):PRINT#1,NA$CR$AD$CR$CY$CR$SA$CR$ZC$
1110 PRINT:GOSUB3000:GOTO1020
1120 MK=MK-1:AK=AK-1:RETURN
1130 :
2000 REM PRINT
2010 PRINT"(clr)":IFMK=0THENPRINT"NO NAMES":GOTO2130
2020 FORI=1TOMK
2030 RECORD#2,(I+1):INPUT#2,KN
2040 IFKN=0THEN2140
2050 RECORD#1,(KN)
2060 INPUT#1,NA$,AD$,CY$,SA$,ZC$
2070 PRINT"NAME       :"NA$
2080 PRINT"ADDRESS    :"AD$
2090 PRINT"CITY       :"CY$
2100 PRINT"STATE      :"SA$
2110 PRINT"ZIP CODE   :"ZC$
2120 GETSP$:IFSP$=""THEN2120
2130 PRINT
2140 NEXT
2150 RETURN
2160 :
```

```
3000 REM SEARCH POSITION
3010 RECORD#2,2
3020 IF MK=1 THEN PRINT#2,1:RETURN
3030 KD$=NA$
3040 FORI=1TOMK-1
3050 RECORD#2,(I+1)
3060 INPUT#2,KF
3070 RECORD#1,(KF)
3080 INPUT#1,NA$,AD$,CY$,SA$,ZC$
3090 KF$=NA$
3100 IF KF$<KD$ THEN  3120
3110 KP=I:GOSUB5000:RETURN
3120 NEXT:RECORD#2,(MK+1):PRINT#2,(MK):RETURN
3140 :
5000 REM INSERTION MOVE
5010 NK=MK
5020 RECORD#2,(NK-1+1):INPUT#2,OK:PRINT#2,OK
5030 NK=NK-1
5040 IF NK>KP THEN 5020
5050 RECORD#2,(NK+1):PRINT#2,MK
5060 RETURN
5070 :
6000 REM DELETE NAMES
6010 INPUT"(clr)NAME TO DELETE";ND$
6020 FORI=2TOMK
6030 RECORD#2,(I):INPUT#2,KF
6040 IFKF=0THEN6080
6050 RECORD#1,(KF)
6060 INPUT#1,NA$,AD$,CY$,SA$,ZC$
6070 IFNA$=ND$THEN6120
6080 NEXTI
6090 PRINT"(down)THAT NAME WAS NOT FOUND!"
6100 GETSP$:IFSP$=""THEN6100
6110 RETURN
6120 RECORD#1,(KF):PRINT#1,"**DELETED**"
6130 RECORD#2,(I):PRINT#2,0
6140 AK=AK-1:RETURN
```

In lines 100 and 105 we OPEN the main data file and the key file as relative files with the appropriate lengths. Remember that relative files are always open to both read and write. In writing files the carraige return is used so often that we give it its own variables (CR$) in line 110. Trying to INPUT from a file which has just been created will give us a "?FILE DATA ERROR" since no data has been recorded in the file. To avoid this we ask whether this is a new file or an old file in line 120. If it is new, then we skip the next section which deals with files which have already been created.

In line 130 we get the maximum number of keys from the key file. This number includes all active keys and those that have been set to zero because of a deletion. In lines 140 to 160, we determine the actual number of active keys, which is also equal to the number of people in our main data file. Notice that we search our key file beginning at the second record and it is important you understand the reason. Remember that the first entry in the key file is NOT a key but the number of keys present.

The main part of our program is contained in lines 175 to 280 and this part calls various subroutines based on the menu choice we make. When we choose Q for quit, the program writes the maximum number of keys (MK) to the first record in the key file and then closes all disk files which are open. Let me try to explain the other three menu choices as separate entities.

The "add" subroutine starts at line 1000 and calls a "search position" routine which then calls a "insertion" routine. The add routine itself simply increments the maximum and actual number of keys and tells us the number of the name we are actually adding by printing the value of the actual keys (AK). If we enter WND for the name then both MK and AK are decremented in line 1120 and we RETURN to the menu. When we actually type in a new name and other information the record pointer is set to the next position in the main data file by line 1100 and the information is written to the file by the same line. As explained before, the data in the main file is in the order it is entered. Now we must insert the main file position of this record in the key file so that when it is printed later it will be in the right order. Line 1110 calls the "search" routine which determines the correct position of this record in the key file.

First we set the pointer to the second record in the key file since this is where the keys actually begin. Line 3020 avoids a search if this is the first record we are entering. In this case, a 1 gets written into the file and we immediately RETURN. For all other records we must search and line 3030 sets the search string equal to the name we just entered since we want the file to be kept in alphabetical order by name. Next we search the file beginning with the first key and ending at one less than the maximum number of keys. This is done to avoid comparing the data we just entered to itself. In lines 3050 and 3060, we set the record pointer to the correct position in the key file (Remember the first record in this file is NOT a key!) and then input a key. Now, we may be inputing the third key from the file but it may be the number 8. This means that in alphabetical order the data which occupies record eight in the main data file will be printed third. Lines 3070 and 3080 set the data pointer to the correct position in the main file and input the data found in that record. Next we set the key field (KF$) to the name and check to see that this name is less than the name we entered as data. If it is, we go to the NEXT statement in line 3120 and go through the process again. If, on the other hand, the name we entered is equal to or greater than the one just read from the file, we set the "Key position" (KP) to the position we are at and go to the "insertion" subroutine. I'll explain that in just a minute! If we get to the end of the FOR-NEXT loop and, thus, the end of the key file without finding a position for our record, it must mean our information should go at the end of the key file. In line 3120, we set the record pointer and write our key number to the file and RETURN to the add routine.

If we must add our key number to the key file in the middle instead of at the end, we have a slight problem. We must move all the numbers after the point of insertion down one to accomodate the new number. To do this we must start at the END of the file and move everything or else we will lose are data as it is overwritten in the file. So, we start by setting a new variable (NK) equal to our maximum key (MK). We do this to preserve the value of MK for later use. In line 5020 we set the pointer in the key file, input a key number, and then print it back to the file. Remember that after the number is INPUT the record pointer increments and we write the number to the NEXT record. In lines 5030 and 5040, we decrement NK and then check to see if we have moved everything up to the position where we want to insert our new key. If we aren't there yet, we do it again. Otherwise, we set the pointer and print the data into the file in line 5050 after which we RETURN to the "search" routine and then back to the original "add" routine.

Now let's try to delete a name using the D choice on the menu. Deleting is actually easier than adding since we chose a simplistic method as explained before. When we find the record we wish to delete, we will write a 0 to its position in the key file and replace its record in the main file with the word DELETED. The subroutine at 6000 first asks us for the name we want to delete. We start searching the key file using the FOR-NEXT loop. After setting the record pointer in the key file we input a key and check to see if it is 0. If the key is 0, that record has already been deleted and we skip to the next key. A

non-zero key causes us to set the record pointer in the main file to the record indicated by the key. We input the record and check to see if the name we want to delete (ND$) is equal to the name we just read from the file (NA$). If it is not, we go on to the next key and next name. If we get through the entire key file without finding our name, then line 6090 tells us the name does not exist and we RETURN to the menu by pressing any key. If we do find the name we want, the program branches to line 6120, sets the pointer to the proper place in the main file and writes "**DELETED**". The pointer in the key file is set and a 0 is written after which the actual number of keys (and names) is decremented and we RETURN.

The option to print the information from the files is the easiest to explain. In line 2010 we check to see if there have been any names added yet and print an error message and RETURN if none exist. If there are names present, a FOR-NEXT loop is used to run through the key file starting at the second record in this file which is the first key. Line 2030 gets the key number and line 2040 checks to see if it is a 0 indicating a deleted record. If the record has been deleted, we immediately go to the next one. If the record exists, the pointer in the main file is set and the data INPUT and printed. This continues until the key file ends.

Many improvements on this program and its methods are possible. I feel it contains just the bare minimums, which were fairly easy to explain. More than this would have been too complex and I wonder if all the information presented here was clear. To change this program for zip code order or any other for that matter, change the NA$ in lines 3030, 3090, and 6070 to the variable name for whatever fields you want. The wording in line 6010 should also be changed. Further changes would include reusing the space of deleted records and allowing multiple key files. Both of these are included in other programs I've written which are too long and complex to describe here.

Two PET Bugs

by Ev Bowyer

Before BASIC 4.0 the only way to write random access disk files was by allocating and writing blocks of data using direct commands. You have to keep track of where each block is located in your BASIC program. This method allows you to use both disks as one large data base. The PET keeps track of the blocks (sectors) used in a table which is art of the directory called BAM. The BAM is read into the disk drive's memory when you initialize the drive. The problem occurred when I end my program. I CLOSEd the random access channel to the disk, but the PET DOS only wrote the BAM of the last disk I accessed back on the diskette. Therefore I lost all the blocks (sectors) I allocated on the other drive. I called Commodore Tech Support, but they don't consider this a problem, so it still works the same in BASIC 4.0. The solution is to issue an OPEN, GET and CLOSE to both drives before ending the program.

I use overlay structure in my business programs. I found that I was losing data from some of my arrays when I went from one overlay to another. The data I was losing was always a default value I plugged in from one of the earlier overlays. The data was a character string. I thought that when you created a character string in a program using something like A$="0.00", the data was placed in the string storage area in high memory. This is NOT true. The data is left in your program area and is pointed to by the variable or the array tables. This is also true for data statements. This means that the next overlay in my program destroyed the data in my arrays. The way around this problem is to code character data as a concatenated string like A$="0"+".00". This will force the string to be placed in high memory.

Auto Operating Cost Study

by Hugh Greenup

In one of Galbraith's vintage years he says you must disregard published statistics this way: the more summative a number the more the likelihood it's wrong. Ten years ago when I began my own study of auto operating cost you could hardly get useful comments about gasoline mileage; it seems nobody really knew how many miles per gallon they were experiencing. The Feds were just about ready to begin their nonsense publication of how many miles a gallon you would never get out of brand X. Then as now, Hertz and also American Automobile Association were regularly releasing colored annual vehicle operating expense numbers and the IRS had an allowable amount per mile you could deduct IF. What's happened since 1972 is that a few hundred thousand computers got into the hands of you, me, and other citizens and we can collect our own data. We can arrange our own summative statistics and draw our own conclusions all for ourselves.

First we've got the problem of collecting the numbers and believing them. A little bound book in the glove compartment and a pen can solve this one. At a year's end you know how many gallons, average price, and how many miles you went. This is a pretty important part but it is just the beginning. Immediately you reach this question: What do you mean annual operating expense? My choice is: Segregate vehicle costs into two parts. Name the first part Fixed and include all costs attributable to vehicle ownership but seemingly independent of whether or not the thing gets driven or not. That other part is per mile costs, those that increment if you drive. If in doubt about a cost, is it really a vehicle cost, ask the question; What if I was trapped in Manhattan, had no car, and took taxis or roller skates everywhere?

There are two components in vehicle operating costs that throw your average economist off his or her feed: depreciation and inflation. You and I haven't this digestive trouble at all; depreciation and inflation are very real costs out of our pockets and we must figure them into our vehicle operation costing. They're fixed costs. Depreciation is a sinking fund, so to speak a savings account preparing for a new vehicle to be bought at a future time. Inflation, as all of us know from recent dreadful experiences, requires a savings account too, since the future new vehicle is doubtless going to cost a LOT more than the one that will be replaced. Economists do recognize opportunity cost even though AAA ignores it and Hertz often treats it wrong: If you were taxi bound, owned no vehicle, you might put the same money on deposit and earn interest with it; by buying a vehicle you lost the opportunity to earn a profit on the money tied up in the vehicle. Opportunity is a fixed cost. Some of the other costs you have to include are ignored by Hertz and AAA, no doubt so they can get lower answers. In LA we have a good police force and you DO NOT bribe your way out of a traffic ticket; if you didn't drive you oughtn't ever to have this cost.

Some decisions you have to make,such as whether to put a specific cost item in the figures or not, seem to have no good answer. Whereas I am familiar with the customary statements about valuation of one's own personal time, as when repairing one's own vehicle, or doing bookkeeping so as to find out how much the thing really costs to run,or washing it by hand, etc. I have never felt satisfied including X dollars per hour. Still, I worry when I omit evaluating and including my own time. I leave my own time's value out of these computations but don't feel quite right about that. A main point here is, you do the deciding yourself about each of those matters. You do not accept the "expert's" advice since the expert may really be trying to coerce you into leasing a car or buying stock in some taxi venture or not noticing how much vehicle ownership really costs. To get a useful answer you have to do it yourself and that includes deciding what to do.

So here's what I do. I'm most interested in how much "one more" mile would cost; i.e., should I drive or fly to Reno? That's the marginal mile my program

prints out below. As I did this run this year I noticed it was the tenth year. I counted six different computer versions of my program which had been run on 4 micros and 6 maxies. My CBM 8032's easiest and best by far. None of the programs I used come from somewhere else; none are structured or top-down or for publication, and they're not for the teaching of programming. I document internally as much as I can and write code to fill up the screen and use little paper. I generally don't care how long it takes to run but I care a lot how long it takes to yield answers I believe. All my programs keep their data within and are constantly changed. This program came to life long before I could buy Command-O so you'll see the half-hearted columnar formatting code segments; since it only runs once a year I didn't feel like putting in the time to replace them with the excellent PRINT USING statement Bob Skyles neat chip permits.

What it gives for the 1982 answer looking backwards at 1981 is 22 1/2 cents per marginal mile. Data it uses each contribute unit cost and miles to consume one unit. Lastly it prints out a reminder for each vehicle cost we recognize but exclude from the marginal mile. So's not to leave any reader in suspense, if every cost were included in a total expense per mile for three years (then buy a new one) it'd come to something like $1.40 per mile omitting my own time's value. Sound high? It's right! Just try the same sort of unbiased cost analysis on your friendly local home microcomputer's costs, using say, words of output that get used for something, and you'll get $4 a word or whatever.

A considerable portion of my vehicle operating cost is deductible since it is used for business purposes. I do not use any average cost per mile for the deduction; instead I itemize. One copy of the program's output goes to the CPA's file just for the heck of it. Maybe it proves I'm intrepid or something. Here's another copy below, for you, along with the program "that done it."

MARGMILE1/18/82

	$	MILES	$/MI.	%
GASOLINE UNLEADED 1/17/82	1.289	9.6	.1342	60
REPAIR CONTRACT	420	20000	.0210	9.3
PARKING	100	6500	.0153	6.8
BRAKE JOB	250	25000	.0100	4.4
4 TIRES	255.4	30000	.0085	3.8
INSURANCE DEDUCTIBLE	250	30000	.0083	3.7
WASHES	40	6500	.0061	2.7
TUNE AND LUBE	75	12000	.0062	2.7
BATTERY	63.6	20000	.0031	1.4
PARTS	50	15000	.0033	1.4
REPAIR CONTRACT DEDUCTIBLE	50	20000	.0025	1.1
BELTS,BLADES,AIRFILT&FLUIDS	30	15000	.0020	.8
CITATION	80	50000	.0015	.7
5 QTS OIL & FILTER	9	7500	.0011	.5
MARGINAL MILE COST =			.2237	99.3

EXCLUDED FROM THE MARGINAL MILE

DEPRECIATION (AGAINST HISTORIC COST) = SINKING FUND IN CONSTANT $
PERSONAL LABOR OF ALL SORTS: REPAIR, ACCOUNTING ,WASH; ETC.
SINKING FUND AGAINST INFLATION
BI, PD, COMP & COLL INSURANCE
AUTO CLUB
OPPORTUNITY COST
VEHICLE LICENSE
SHOP TOOLS

```
100 PRINT"(clr)":POKE59468,12
110 GOSUB590:GOSUB640:PRINT#5,SF$
120 L$="                                    $  MILES $/MI.   %":PRINT#5,L$
130 READ ITEM$:IF ITEM$="3030"THENRESTORE:GOTO150
140 READ PRICE,MILES:MARGCPM=MARGCPM+(PRICE/MILES):GOTO130
150 READ ITEMS$:IF ITEM$="3030" THEN 300
160 READ PRICE,MILES:L$=ITEM$
170 X$=STR$(PRICE):L=LEN(X$)-1:X$=RIGHT$(X$,L):FORJ=1TO6-L
180 L$=L$+" ":NEXTJ:L$=L$+X$
190 X$=STR$(MILES):L=LEN(X$)-1:X$=RIGHT$(X$,L):FORJ=1TO6-L
200 L$=L$+" ":NEXTJ:L$=L$+X$
210 CPM=1000*(PRICE/MILES):X$=STR$(INT(CPM*10000)):ALL=ALL+CPM
220 L=LEN(X$)-1:IFL>7ORL<4THENPRINT"ERROR ":GOSUB220
230 IFL=7THENL$=L$+" ."+MID$(X$,2,4)
240 IFL=6THENL$=L$+" .0"+MID$(X$,2,3)
250 IFL=5THENL$=L$+" .00"+MID$(X$,2,2)
260 IFL=4THENL$=L$+" .000"+MID$(X$,2,1)
270 PCTCST=INT(CPM/MARGCPM)/10:IFPCTCST<10THENL$=L$+" "
280 IFPCTCST<1THENL$=L$+" "
290 L$=L$+STR$(PCTCST):TLPCT=TLPCT+PCTCST:PRINT#5,L$:GOTO150
300 L$="                                          ----- ----":PRINT#5,L$
310 L$="MARGINAL MILE COST =                     "
320 X$=STR$(INT(10*ALL)/10000):L$=L$+X$
330 X$=STR$(INT(10*TLPCT)/10):L$=L$+X$:PRINT#5,L$
340 DATA"GASOLINE UNLEADED 1/17/82   ",1.289 ,9.6
350 DATA"REPAIR CONTRACT             "   ,420      ,20000
360 DATA"PARKING                     ",100    ,6500
370 DATA"BRAKE JOB                   ",250    ,25000
380 DATA"4 TIRES                     ",255.40,30000
390 DATA"INSURANCE DEDUCTIBLE        ",250    ,30000
400 DATA"WASHES                      ",0040   ,06500
410 DATA"TUNE AND LUBE               ",75     ,12000
420 DATA"BATTERY                     ",63.60  ,20000
430 DATA"PARTS                       ",50     ,15000
440 DATA"REPAIR CONTRACT DEDUCTIBLE  ",050    ,20000
450 DATA"BELTS,BLADES,AIRFILT&FLUIDS ", 30    ,15000
460 DATA"CITATION                    ",80     ,50000
470 DATA"5 QTS OIL & FILTER          ",9.0    ,7500
480 DATA"3030"
490 PRINT#5:PRINT#5,"EXCLUDED FROM THE MARGINAL MILE":PRINT#5
500 READ EX$:IFEX$)"3030"THENPRINT#5,EX$:GOTO500
510 PRINT#5:CLOSE5:END
520 DATA"DEPRECIATION (AGAINST HISTORIC COST) = SINKING FUND IN CONSTANT $"
530 DATA"PERSONAL LABOR OF ALL SORTS: REPAIR, ACCOUNTING ,WASH; ETC."
540 DATA"SINKING FUND AGAINST INFLATION","BI, PD, COMP & COLL INSURANCE"
550 DATA"AUTO CLUB","OPPORTUNITY COST","VEHICLE LICENSE"
560 DATA"SHOP TOOLS","3030"
570 GETA$:IFA$=""THEN570
580 RETURN
590 RR=3:INPUT"DEVICE   3 (4left)";A$:IFA$="4"THENRR=4
600 CLOSE5:OPEN5,RR:RETURN
610 CLOSE5:OPEN5,4:GOSUB640:CMD5,SF$:LIST:PRINT#5:CLOSE5:RETURN
620 GOSUB640:GOSUB630:DSAVE("/"+SF$),D(DR):VERIFYSF$,8:PRINTDS$" "SF$:RETURN
630 INPUT"DRIVE   0 (4left)";DR:RETURN
640 SF$="MARGMILE1/18/82":RETURN
```

THE EPSON AND I
A Happy Saga.

by Jim Fowler

In issue #3 there were a number of questions raised (and partly answered) about the use of EPSON printers with COMMODORE equipment. I've lived through the same experience as A. H. McCann ("Beware the Epson Jabberwock"), but my outcome seems to have been a bit happier. At least I continue to use my IEEE interface without any problems. However, I have had to modify some of my programs.

There are two kinds of incompatability between EPSON and COMMODORE. First, COMMODORE does not use true ASCII to send characters out on the IEEE data bus. Well, I bought COMMODORE equipment in the first place because it seemed to have the better operating system with BASIC resident, and because it had a larger "vocabulary" of graphics. The outside world has a smaller vocabulary and includes a lot of stuff not used much (at least by micros) like signals for shift-in, shift-out, start-of-heading, synchronous-idle, and a dozen others. So you just have to write a little translator in those programs that send text to the printer. Second, there is no agreement about whether a carriage return (CR) also means a line feed (LF). The printer uses a motor to advance the paper which is not the one that moves the printhead. To turn on that motor for a specific length of time takes a different signal than the one that moves the printhead to the left end of the line. Here's a real problem because one of the neat features of an EPSON printer (and I don't mean the CBM modified EPSON MX-70) is that it is bi-directional. That saves time and speeds the work, but it also means that a CR signal must not be executed without first considering whether the printer ought to print material on the way back to the left margin. This problem is not peculiar to COMMODORE-EPSON combinations.

Well, enough of that. What to do? Here's what I did: I have an EPSON MX-80 (I couldn't get the F/T model at the time and now I'm glad I saved the money and spent it on GRAFTRAX-80). I use WORDPRO 4, and a number of other programs which must interface with the printer. WORDPRO 4 gives you an option of printers: CBM, ASCII, or SPINWRITER. If you set it to ASCII there is no problem about upper and lower case. Otherwise, if the printer takes output from the 8032 which is <u>all lower case</u>, it will print neatly in capital letters. This is fine for listings (unless you have a capital letter in quotes). Capital letters in the output turn into TRS-80 graphics. (If you want to print PET graphics you'll have to make other changes, but that's another story.) If you want upper and lower case to agree between the screen and the printer you must make a translation somewhere. If the output comes through the program that is no big deal. The following BASIC routine does it rapidly and you can attach it as a subroutine to any program:

Assume X is the ASC value of the character to be translated:

```
IF (X > 192) AND (X < 219) THEN X = X AND 127
IF (X >  64) AND (X < 91) THEN X = X AND 223: X = X OR 32
```

Now CHR$(X) can be sent to the printer. Or you can concatenate a line and send it to the printer as one long string (which saves time - the printer works as you gather together the next line).

The CR/LF problem must be solved in the printer hardware. Here are the switch settings that work with my EPSON MX-80 and CBM 8032 with the IEEE bus interface:

Switch #1: 1, 2 ON; 3, 4 OFF.
Switch #2: 1 unused; 2, 3 OFF (fixes LF when in last col or when CR received from computer); 4, 5 OFF; 6 ON; 7 OFF; 8 ON (maybe 7,8 have no effect with interface).
Interface switches: Sw #1: 3 ON, rest OFF (= device 4)
Sw #2: "JPET1" ON, rest OFF (agrees with above on CR/LF settings).

If you add the GRAFTRAX-80 chip the interface settings remain the same but the others should become:

Switch #1: 1 OFF (80 cols); 2, 3 ON (fixes CR/LF); 4, 5 OFF; 6, 7, 8 ON
Switch #2: 1, 2 unused; 3 ON, 4 OFF.
The instructions with the chip explain most of these.

I recommend you add the GRAFTRAX-80 chip to get your money's worth from the MX-80. If Ralph will permit this text to be printed in THE PAPER without being retyped you can see the cute things one can do:

You can expand to 5 letters/inch.
You can print at 10 per inch in *ITALICS* *which looks pretty cool or return in the middle of a line* to normal. You can get emphasis in headings or topic sentences by going to 8 per inch:
Which leaves more space between letters - OK for headings.
I have set all of this in Emphasized mode so the it will be dark enough to make good offset plates but you can get it even darker:
LIKE THIS: it's called the WALLBANGER mode because each dot is done 4 times in a small square overlapping its neighbors.
Then there is the condensed mode which gives 16.5 to the inch.
It is good for listing - you have half the page for notes!

Besides all the type faces (and these are only some of them) you can print any combination of up to 8x8 microdots in a column. You can also move the paper by as little as 1/216th inch, less than a dot-width. Thus you have a slow but precise plotter! Now, to make a picture of Snoopy ...

SYSRES

Type: Software by Ralph Bressler
Model PET: Any PET w/ BASIC 2.0 or greater and at least 16K of RAM
Any PET/CBM disk drive
Printer recommended
Source: Solidus International
#204, 4202 Guide Meridian
Bellingham, WA 98226
Price: $95

The ad for this programming aid makes some big promises and I wanted to see how close the real thing came to the claims which were made. This review supposes that you are interested in a programming aid and that you know a little about some of the commands available. I have been using SYSRES now for about two weeks in developing educational software that I sell. I cannot possibly explain in any detail all of the commands and options available. However, if what I say sounds interesting you might want to try SYSRES. Solidus offers a 30-day, money back guarantee so you have nothing to lose. Don't order it, though, unless you can afford to buy it because I think once you use it you won't be able to let it go.

SYSRES comes as a single master diskette in a nicely done, padded binder which also includes a well-written manual. More about the manual later. The master disk will create three working copies of SYSRES but no more and the working disks cannot be copied. I once said that I would never buy a protected program but that was before the new sophisticated methods of protection came about. If you somehow manage to garble all three disks, Solidus promises to replace them for free. They have also promised to offer any upgrades or new versions of SYSRES to registered users for a reasonable fee. Keep in mind that the master will produce copies for any PET/CBM model and this is a big plus. When you load SYSRES you may choose to put it at the top of 16 or 32K of RAM, in 8K of RAM starting at $9000 or split it between the top of memory and 4K of RAM at $9000.

The clearly written manual contains some 86 pages of detailed explanation of each command and many examples of useful ways to use them. There were a few points which I did not understand but this did not seem to effect my ability to use SYSRES effectively. The commands are presented in alphabetical order so finding things is easy. A detailed table of contents is present and quick reference charts are provided.

SYSRES is meant to aid programmers not program users. None of the SYSRES commands are to be included in the final version of a program. In fact, SYSRES automatically disconnects itself when a program runs. SYSRES provides auto-repeat on ALL keys and has a smooth scrolling feature which can move through a program in either direction. A screen dump is available at any time.

SYSRES has all the old familiar "wedge" commands but enhances them greatly. All the wedge commands now operate directly on the directory. This means that after listing the directory a / in front of a program name will load it while an up arrow will load and run the file. To scratch a file you just type @S1: before the name in the directory and that file is scratched from drive 1. Typing @C0: in front of a name will copy that file from drive 1 to drive 0. Of course, these commands can be used in the more standard way also. By typing @L in front of a name, we cause that file to be listed on the screen without disturbing what is in memory. The listing may be paused, as most SYSRES functions can be, by pressing the space bar. Placing a * in front of this command or any other that lists things will direct the output to the printer. The listing of these disk files or those listed from memory using *LIST may be formatted so that only one command will appear on a line and nested loops will be indented. A nice touch

even if it isn't used all the time. These command save a lot of time and prevent many mistakes in the handling of disk files.

SYSRES has an APPEND command and a true MERGE. Append adds one program to the end of another and is considerably faster than merge which interleaves line numbers. The AUTO command not only feeds line numbers but will also give up to 127 characters of text. I have used this in the generation of DATA lines and in "capturing" a picture drawn on the screen. DELETE functions in a similar manner as it always has by deleting a range of lines. MON provides a quick way to call the TIM monitor. Typing OLD after accidentally NEWing a program will recover it even if you have also entered a variable name before. WHY or WHY? provides help when a mistake is encountered and does a good job of pointing out errors. TRACE displays the line being executed or the values of variables or both. These are displayed in windows in the upper part of the screen. TRACE can be delayed until a certain point in the program at which time a POKE will turn it on. I never use this command so I have a hard time getting excited about it.

RENUMBER will renumber all or part of a program. It will also rearrange line numbers so that the order of lines and routines can be switched around. This is useful and cannot be done with any other aid I know of. DUMP lists the value of all scalar variables to the screen or printer but will not dump arrays. Many times I want to see the values in an array so this seems like a minor inconvenience. However, SYSRES has the ability to define any shifted key to perform a function. SYSRES comes with many default values for shifted keys which can be turned on using the KEYS command. Default values and keys you define may be turned on and off without destroying their values. I have defined keys to dump various array and to advance my printer one line at a time. I do not use this function in POWER since it requires you to set up special REM statements at the beginning of the program. These functions in SYSRES are very handy and easy to use.

SYSRES supports the most extensive CHANGE and FIND commands I have ever seen. The author boasts of over 700 different combinations for CHANGE and I believe it! You can direct change and find to only work if a match is found at the end or beginning of a line. You can look for EXACT variable matches so that changing X to A does not effect XX. Pattern matching is available also. Finally, you can change or find only in the command area, which eliminates finding the tokens for some commands embedded in other words in the program. These functions are very useful and extensive explanations are given in the manual.

SYSRES provides a PUT and GET command to save and load sequential files. You can actually create files to be read by other programs such as assemblers. You can also enter your most used KEY functions to be recalled later using EXEC. EXEC calls a sequential file and then enters what it finds in the file as if the information were being typed at the keyboard. One creative use is to DUMP variable values to a disk file and then load them again using EXEC.

SYSRES is the most complete and comprehensive programmer's aid I have seen. I cannot say it is the best for you since the requirements of various programmers will differ. SYSRES takes up 8K of memory no matter how you look at it. I feel that you should have a 32K machine to use it or have the extra RAM starting at $9000. This also means it would be hard if not impossible to burn it into a chip. SYSRES requires a disk drive since it comes on disk and many of its functions are oriented toward one. ROMs like POWER can be used without disk but they too are oriented in function toward drives. ROMS, of course, take up none of your programming space but they quickly crowd the available sockets. Some of the many commands and options in SYSRES will not be used by many programmers at all and take up space but that is the price to pay for such a complete package.

The bottom line is that those who have the need for a powerful programming aid and the RAM and disk drive to support it should look at SYSRES. Quite frankly, SYSRES is FUN to use and I feel I have just scratched the surface. Many good programmers produce excellent software without any programming aid. How much more could they produce with a package like SYSRES?

Millipede
Wallbanger

Type: Software
Model PET: Any PET or VIC (different versions)
Source: On Line Software
PO Box 2044
Orcutt, CA 93455
Price: $15 each

by Ralph Bressler

It is now 1 A.M. and my arms and fingers are aching and about to fall off. Still, I just had to write now and tell you about two of the worst game programs I have seen for the PET. This company must have no ethics or scrupples what-so-ever to distribute these programs at this price. All I can say is that they must be trying to lure you in with the low price. Quite frankly these are two of the most captivating, aggravating, challenging games I have seen on the PET. They make me play until I drop! I see millipedes in my sleep until I have to either take a pill or get up and play some more! I ignore work long overdue and must limit myself to only one game until the work is done. Shame on you, On Line!

Both programs are fast since they are a combination of BASIC and machine language. Both come with three pages or so of documentation giving an overview of the game, the controls and scoring and some handy strategy tips. Both also include sound to reward and to warn the player. My favorite is Millipede which is much like another arcade game of similar name. I have never played the arcade version but I think this is a close copy. There are 52 separate levels but I'll have to take the programmer's word since I have only scored 4089 and was somewhere around level 6. The game is complete with an ever expanding millipede, a bouncing spider, fleas leaving mushrooms in their wake and the deadly scorpion spreading poison mushrooms. All I can say is that when I finally get the board set up the way I want it along comes a spider to squash me or a scorpion to send the millipede straight down on top of me. Instructions are given to immediately set the game in high speed mode or level 27 and maybe someday I'll try that. For now, the game is fast and the movement smooth. At first, I found the controls a little ackward but I quickly became used to them. To move Z and C used for left and right, * for up, + for down and 2 to fire. They machine language movement routtines are easy enough to modify so that other keys may be used.

Wallbanger might have a chance of me playing it if Millipede wasn't around. In this game you control a ship which you can rotate with Z and C, move backward with *, forward with + and fire with 2. The screen is bounded by walls which close in as you progress from level to level. To begin you must choose the number of balls which will be in play. In the first mode these balls bounce randomly leaving blocks where they hit the wall which you may shoot for points. You must avoid the balls or else your ship explodes. Shooting the balls in this first mode will make the remaining ones quite vengeful. When the balls reach top speed the mode changes to one where both balls and blocks are fair game. Finally, if you fail to destroy all the balls, they head for your ship in one great Kamikaze raid. Again, any contact ends the game. Should you survive, (I did once!) the walls close in and you start all over again.

I sit here hoping that other people will buy these programs so they will know how afflicted I am. I also hope people will buy them so that On Line can show a profit and decide to make more. A company that starts this way has a good future. I, for one, would buy their next two games sight unseen just based on this effort! These opinions are reflected by several students in my school who are dedicated arcade game players and find these games interesting and challenging.

Multiplication of Fractions
Equations

Type: Software
Model PET: Any 40 column PET
Source: Microcomputer Workshop
10 Elizabeth Place
Armonk, NY
Price: $25 each

by Ralph Bressler

These programs represent just two of the many pieces of educational software produced by an experienced math educator. Not only does Don Ross have math experience but he also is quite experienced in the use of microcomputers in the classroom. This means that Don acutallu uses his own programs and is constantly revising them based on his own experiences or the feedback from others.

The fraction program divides the screen into three sections. The problem which is randomly generated is displayed in the top third and it never changes so that the student can always see the original. The second section of the screen is the work area where the fractions will be changed according to the operations the students apply. The bottom third is reserved for soliciting student input and the error message displays. Unlike some other programs which simply ask for the correct answer and then mark it right or wrong, Don's program guides the student through the problem. The student may choose at any step to either cancel or reduce the fraction or multiply. If they choose to cancel or reduce they must give the correct numbers to start with, how they will be changed and the final result. After the problem is in simplest form, the student can multiply. The error messages are concise and to the point. After completing a problem, the program reports the number of procedural and computational errors and asks if the student would like to continue or stop.

When the student is asked for input it is virtually impossible to drop out of the program. However, the stop key is not disabled as I feel it should be. A student can choose to reduce or cancel first but the program always asks for numbers to cancel. This is a minor wording error. I feel that the numbers are a little large to begin with and I see no indication of them getting any harder as the program progresses. The errors made in cancelling are not recorded anywhere as they might be. Also, I was able to make the same mistake over and over again without the program giving up and showing the correct answer. When you decide you've had enough the program simply ends with no final report. After all of this you may think I don't like the program. This is not true. Most of my objections are based on opinion and it is up to the individual to decide on the presentation of the material. I do feel disabling the stop key and showing the correct procedure after a certain number of incorrect responses would be a good idea.

Equations has much the same approach as the fractions program but is more complex. The student is presented with an equation of the form AX + B = C and is asked to solve for X using the correct procedure. Again, the student is guided through the process by being asked to choose from a set of rules. At any stage the student may add, subtract, multiply or divide on both sides of the equation or they may choose to simplify. The instructions give a sample problem and its complete solution which is very helpful. The error messages are clear and instruct the student as to where they went wrong. The program nicely accounts for odd solutions such as subtracting -5 rather than adding 5.

I feel that the program moves very slowly but I suspect this is because I already know how to do these problems. For someone who is just learning, the step by step method would be very effective. When answering questions the program is almost impossible to drop out of but, again, the stop key is active. The equations given start out pretty hard and don't seem to get easier if you

make many errors. I continue to make precisely the same errors at some places and the program never gave up and told me the correct answer. When the student decides to stop, the program ends with no final report. As I said before, many of my objections are personal opinions which some would agree with and others would not.

I feel that these two programs have what I consider to be some minor flaws but are well worth adding to your library and using with your students. In fact, I hope Don continues this series with other types of problems. There are many programs which play cute games to teach students or simply ask for an answer but there are few which show the step by step problem solving method. Once students know the method they will be able to apply it to many different problems.

Programming the PET/CBM

Type: Reference
Model: Every PET/CBM Model
Source: COMPUTE! Books
P.O. Box 5406
Greensboro, NC 27403
Price: $24.95

by Jerry Key

It's extremely difficult to say anything more about this book except GO GET IT! This has to be THE one reference book that we have always dreamed about but never expected to see. Raeto West has not only stripped the PET/CBM down to the very last piece, he told us what was there.

In a nutshell, this book not only goes into a detailed description of all the commands available in Basic, it also shows how to implement many that aren't but we wish they were. For instance, he tells you how to implement such routines as CRUNCH, DELETE, MERGE, PRINT USING and gives you seven(!) different sort routines. Differences between ROMs are detailed including the FAT 40. There is at least a complete page on every Basic command, a complete reference to ML OPCODES, a detailed and explained memory map and much, much more.

The leaning of the book is toward machine language programming but there is something here for everyone. Jim Strasma pointed out that it gets into machine language half way down the first page. Still, there has never been so much about the PET/CBM in one place!

I have been using the book daily, and I mean daily, for about three months and it is beginning to show it. I have only found one verifiable error to this time. That says a lot by itself. The mysteries of the keyboard lookup tables, what they mean and where they are have been revealed. Both on the 40 and 80 column machines. My answer to Stan Spence in I/O came from here. There are routines for a special LIST routine (note JS, it is in Basic with the ML in data statements). There are routines for repeat keys, defining your own keys and sections on sound and graphics.

When I first asked Jim Strasma his opinion of the book, he said "I would almost give up my disk drive for it". What else could be said ? My only complaint is the version from COMPUTE! is of a poor quality by comparison to the version from England. The pages seem to have a yellow tint and it is about one fourth thicker because of the type paper. The glaring blue cover doesn't do much for me but who am I to say. This book is a must but I recommend the version from England although there is no difference in content. You will have to check with the below address for the price over there:

LEVEL LIMITED
P.O. Box 438
Hampstead, London
NW3 1BH

HESCAT

Type: Software by Ralph Bressler
Model PET: BASIC 2.0 or 4.0 w/ 16K, dual disk, printer optional
Source: Human Engineered Software
3748 Inglewood Blvd. Room 11
Los Angeles, CA 90066
Price: $29.95

HESCAT is a collection of five main BASIC programs each with its own function. Some programs have machine language subroutines to speed up certain functions. HESCAT allows you to keep a file of all the programs on all your disks. Once this file is created you can search for a program to find what disks it may be on. You can also get information about all your disks or just certain ones. HESCAT will catalog about 131 disks and 3300 to 6000 filenames on a 4040 disk drive. One the 8050 drive the numbers are 214 disks and 10,000 to 20,000 filenames.

When the program is running you are presented with a main menu and sub-menus as you make various choices. Default responses are always present so that you can always return safely to a menu. To add a disk to the catalog you choose the CATALOG option and place your disk in drive 1. HESCAT will then ask for an "external ID" for that disk. This may be the same as the ID actually on the disk or it may be a different one. This prevents problems if backup disks have the same ID. HESCAT then reads the information it needs from the disk such as filenames, internal ID, number of files and free space. This summary information is added to a file called HEADERS and another file with the same name as the external ID is created. This last file contains all the detailed information about the disk being cataloged. If an ID is reused the old files are scratched and the new information replaces them. This whole process takes well under a minute per disk. HESCAT also allows you to easily "uncatalog" a disk by just giving its external ID.

After a cataloging session it is suggested that you choose the SORT NAMES option. This reads all the information about newly cataloged disks, adds it to the ALPHA NAMES file and sorts that file alphabetically by file name. This file is used by the other options which provide information about the disks. This program runs without intervention and automatically chooses the better of two sort methods.

The PRINT option allows you to obtain hardcopy information in three ways. First you choose whether you want information about all the disks, a range of IDs or just one disk. One choice is to print file names in an alphabetical list. You may also choose to print the disk directories which will give you summary information about that disk and all the filenames. Choosing the HEADERS option allows you to get just the summary information about all disks. This is useful to locate nearly empty disks and to see what IDs have been used.

Another option is one called DISPLAY which allows you to easily browse through any sequential data file. The program allows you to scroll through the file using the "D" and "U" keys for down and up or "F" and "L" to get to the first or last record in memory. Files which are very long will automatically load just enough to fit in memory and will load more as you move through them. This program is meant to be used on the HESCAT files but could be used for many other purposes.

The LOCATE option searches the ALPHA NAMES file to find a program or search string you specify. This file can be searched on disk or can be loaded into memory and then searched. It can search the 1200 to 1700 filenames which will load into 32K in around two seconds! Even if ALPHA NAMES is very long you can load segments into memory and search them one at a time. A "don't care" character is provided to help in your search. When a match is found the filename

will be displayed on the screen along with the type of file and the external ID of the disk it is on.

The package even allows you to add a user-defined option by adding code into the HESCAT program or adding another program which will be called by HESCAT. I wanted to be able to locate programs by type of file and within a range of disk IDs. I wrote the program and within a short time had it working properly with the other HESCAT programs.

With a complex package like this one, it is easy to forget how to use some of the features, particularly if they are not used often. The proper procedure can be looked up in the instructions but that requires searching for the manual. HES has included a short HELP section in the main menu program to explain the vital parts of all procedures. You just press H and then the number of the option you want help with. You can even add a "HELP" section for any option you design.

As usual the manual provided is very complete and helpful. It is nice to see a company set a high standard and then maintain it. The manual explains fully the operation of the programs and gives detailed information on how to recover from errors with minimum loss of data. A short explanation is even given on how to change the PRINT program to use different printers. The instructions also give program lists, illustrate file structure and show you how to modify the programs. The variables and routines in each program are laid out and carefully explained so that you can see exactly how the programs work.

If you want to quickly catalog your disks with a minimum amount of problems, this package is one to look at. It is versatile but easy to use because of the extensive instructions in the programs and the manual.

Prowriter 8510AP

Type: Hardware
Model PET: any with BASIC 2.0 or 4.0
Source: Distributor: Leading Edge
225 Turnpike St.
Canton, MA 02021
Dealers: many local and mail order
Price: $795

by Ralph Bressler

I have had the Prowriter connected to my PET system with a CmC ADA 1600 interface for almost two months now. It is an excellent printer and a good value particularly when you keep in mind that it is being heavily discounted and can be obtained for under $600. It works well with the SuperScript word processor which can send all the control codes needed to make the Prowriter perform properly. I have succesfully used it with programs of my own design and with commercial programs such as Flex File. A complete list of its features would consume three pages but here are some points.

The printer has pica, elite, compressed and proportional type faces which can be software selected. It can do true underlining, boldfacing, and expanded printing. Of course, it is bidirectional and logic seeking which makes it quite fast. The friction and tractor feeds work very well and the tractor is bidirectional which allows reverse line feeds. Bit image graphics are also included. DIP switch settings allow the choice of many different features such as the selection of the characters needed for several different foreign languages. A Greek character set contains many symbols needed for math and science. Variable line spacing makes superscripts and subscripts possible.

Some people feel that the print quality is not as good as the Epson but I see little difference. It is true that the Epson is better supported and more widely used right now but I feel the Prowriter is a good value and will present some advantages and few limitations.

"Probably the best-documented programs I've seen for PET/CBM."
Robert Baker
Microcomputing
September 1981

PET/CBM & VIC OWNERS!

"The strongest points of this system are its unsurpassed documentation and its human engineering."
Ralph Bressler, The Paper
Nov/Dec. 1981

Utilities & Games

GAMES FOR VIC

Skier Thrill to downhill skiing, using your joystick to hit flags and avoid obstacles. Great graphics. 3 levels of difficulty. $17.95

Maze of Mikor Adventure-like game with stunning graphics challenges you to steal the Warlock's gold as you evade the demon. $17.95

Tank Wars Match your wits against the evasive enemy, maneuver around obstacles and avoid mines. $17.95

Victrek Graphics and sound add to the excitement as you scan galactic maps, maneuver through star bases, and battle klingons. Enhanced version included for 8K VIC. $17.95

Pinball Score points with flippers through bumpers and alleys. This game is the real thing. $15.95

Simon Four squares light and sound at random. Then you imitate the sequence. It gets tougher as you get better. $15.95

Fuel Pirates Protect your stock of atomic fuel from raiding pirates using your particle cannon. $15.95

Lazer Blitz Terrific graphics as you destroy enemy aircraft from your flying saucer. $17.95

Pak Bomber is dropping bombs that you must catch. Great challenge for eye-to-hand coordination. $15.95

NEW FOR 5K VIC 20!

Tank Trap You're challenged to protect your citizens. 4 exciting levels, each tougher than the one before. $17.95.

Concentration Test your recall skills, when you try to remember what you saw beneath the block, and match it. $15.95.

Dam Bomber You must break the dam while under cannon fire. $15.95.

VIC FORTH Full FigFORTH implementation with compiler, interpreter and complete editor. Runs on standard VIC 20 with 5K. $59.95 on cartridge.

HESMON Machine language monitor. Contains 50% more commands than Commodore's. $39.95 on cartridge.

Turtle Graphics Based on LOGO. Perfect for learning computer programming. Great for kids. Very versatile. $39.95 on cartridge.

HES Writer Word processing. $39.95 on cartridge.

SPECIAL ANNOUNCEMENT TO OUR CUSTOMERS AND DEALERS

HES has relocated to the San Francisco Bay Area and is now a division of USI International. We now have greater resources to provide you with excellent software on cartridge, cassette, or diskette in superior quality packaging.

Watch for more exciting products from HES.

UTILITIES FOR PET & VIC

6502 ASSEMBLER PACKAGE

HESBAL is a 1- or 2-pass Assembler using standard MOS mnemonics and operand formats, has pseudo-opcodes and over 25 error messages. HESEDIT is a full screen text editor for use with HESBAL or alone. Assembler package runs on PET or VIC with 1 cassette and minimum 8K, (specify PET or VIC). $23.95 on cassette, $26.95 on diskette.

HESCOM transfers data and programs bidirectionally between PETs, VICs, or a PET and VIC at 3 times the speed of the disk. Set up VIC as a terminal to PET and create games for 2 players. Or use VIC as a peripheral to PET for hi-res graphics and sound. Only $49.95 on cassette, $52.95 on diskette.

HESCOUNT monitors BASIC program's execution and accumulates data. Essential for debugging and optimization. Discover how many times your program looped, and when IF statements were true or false. Fast execution. Runs on PET or VIC. On cassette $23.95. On diskette $26.95.

HESCAT Complete hi-speed diskette cataloging system. Five programs let you sort names, print reports 3 ways, and locate file names in memory or on disk, and much more. Works with any PET/CBM, 16K and dual drives. $39.95.

HESLISTER takes complex BASIC programs and prints (to screen or printer) in an easily understood manner. Lets you analyze BASIC programs to alter or debug code. Works on any PET/CBM and 1 disk drive. $23.95.

HESPLOT Very fast hi-res graphics subroutines for VIC. Includes line drawing routines. With 8K VIC plot within field of 176 x 160. On cassette $17.95

All products available at your dealer or directly from HES.
Add $2 postage. Calif. res. add 6% sales tax.
We accept VISA and MasterCard. Dealer inquiries invited.

PET, CBM, and VIC are trademarks of Commodore.

HES Human Engineered Software
71 Park Lane • Brisbane, CA 94005
(415) 468-4110

Send today for your **FREE CATALOG** of VIC and PET/CBM Software

Name_____

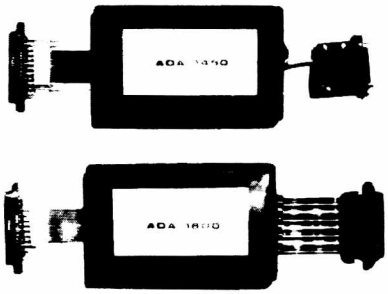
Street_____

City_____

State_____ Zip_____

Mail to **Human Engineered Software**
71 Park Lane • Brisbane, CA 94005

CBM/PET INTERFACES



RS-232 SERIAL PRINTER INTERFACE – addressable – baud rates to 9600 – switch selectable upper lower, lower upper case – works with WORDPRO, BASIC and other software – includes case and power supply.

MODEL – ADA1450 149.00

CENTRONICS NEC PARALLEL INTERFACE - addressable – high speed – switch selectable upper lower, lower upper case – works with WORDPRO, BASIC and other software – has Centronics 36 pin ribbon connector at end of cable.

MODEL – ADA1600 129.00

CENTRONICS 730/737 PARALLEL INTERFACE – as above but with Centronics card edge connector at end of cable.

MODEL – ADA730 129.00

COMMUNICATIONS INTERFACE WITH SERIAL AND PARALLEL PORTS – addressable – software driven – true ASCII conversion – selectable reversal of upper-lower case – baud rates to 9600 – half or full duplex – X-ON, X-OFF – selectable carriage return delay – 32 character buffer – centronics compatible — much more.

MODEL – SADI 295.00

ANALOG TO DIGITAL CONVERTER – 16 channels – 0 to 5.12 volt input voltage range – resolution is 20 millivolts per count – conversion time is less than 100 microseconds per channel.

MODEL – PETSET1 295.00

REMOTE CONTROLLER WITH CLOCK/CALENDAR – controls up to 256 devices using the BSR X10 remote control receivers – 8 digital inputs, TTL levels or switch closure — 8 digital outputs, TTL levels.

MODEL – PETSET2 295.00

All prices are in US dollars for 120VAC.
Prices on 220 VAC slightly higher.
Allow $5.00 shipping & handling, foreign orders add 10% for AIR postage.
Connecticut residents add 7½% sales tax.
All prices and specifications subject to change without notice.
Our 30 day money back trial period applies.
MASTER CHARGE VISA accepted.
MENTION THIS MAGAZINE WITH YOUR ORDER AND DEDUCT 5% FROM TOTAL.

IN CANADA order from: Batteries Included, Ltd., 71 McCaul Street, F6 Toronto, Canada M5T2X1. (416)596-1405.
IN THE USA order from your local dealer or direct: Connecticut microComputer, Inc., 34 Del Mar Drive, Brookfield, CT 06804, (203)775-4595.

Dealer inquiries invited.

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

NEW PET/CBM SOFTWARE

Let Computer Mat turn your Pet into a Home Arcade!

ASTEROIDZ — Its your ship vs. a swarm of killer gammaroidz. You are on a collision course and must destroy them before they blast you into the next galaxy. Four levels of play. Has hyperspace keys that move you around. Arcade style entertainment at its finest. Great graphics and sound.

Cass. 8K $9.95

MUNCHMAN — How many dots can you cover? It's you against the computer munchers ZIP and ZAP. Can you clear the maze first or will they get you? Number keys move you up, down, right and left. GREAT GRAPHICS AND SOUND.

Cass. 8K $9.95

TARGET COMMAND — Its you against a barrage of enemy lazers that are aimed at your ammo dumps. Sight in on the targets and score as many hits as you dare. As your skill increases so does the the difficulty — (5 levels to select). This is an arcade-style game with great graphics and sound effects. A must for your PET/CBM.

Cass. 8K $9.95

ALL OUR SOFTWARE RUNS IN 8K
OLD-NEW ROM — 40 CHR. SCREEN
WRITE FOR FREE CATALOG OF VIC/PET SOFTWARE
PLEASE ADD $1.00 PER ORDER FOR SHIPPING
COMPUTER MAT • BOX 1664M • LAKE HAVASU CITY, AZ. 86403

TNW'S PTERM and XPTERM software converts Commodore's PET/CBM computers to full ASCII terminals to access such remote systems as The Source, Telenet, Bulletin Boards, others. PTERM provides a flexible basic terminal capability; XPTERM adds disk file transfer capabilities. Versions available for TNW'S auto-answer/auto-dial 103 Modem, CBM-8010 coupler, and modems attached by TNW'S serial interfaces. Also: PTWX converts your PET/CBM into a TWX terminal. From $19-99. Details from George Masters:

SO
SOF
SOFT
SOFT
SOFTWA
SOFT SOFTWARE

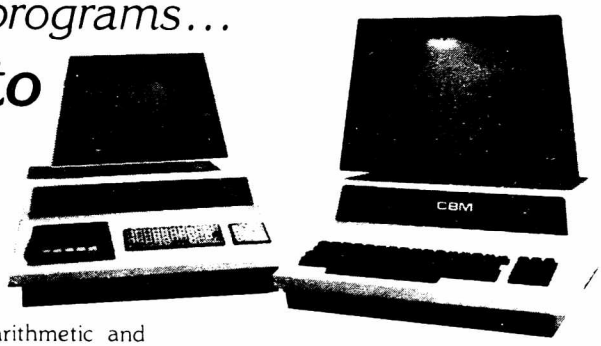
FROM TNW CORPORATION

Dept. P, 3444 Hancock St., San Diego, CA 92110
(714) 296-2115 • TWX 910-335-1194
VISA/MasterCard • Dealer Inquiries Welcome.

for fast development of fast, tight programs...

step beyond FORTH, to

RPL



High speed, low memory requirements, and user-friendly development tools are no longer mutually exclusive. **R**everse **P**olish **L**anguage, a FORTH-like language now available for the PET and CBM computers, is faster than FORTH, easier to debug than BASIC, and more space-efficient than any other language known, including assembly language. Here's what **Loren Wright,** MICRO magazine's PET Vet, says about it:

> **"RPL is generally faster and more conservative of memory than FORTH . . . RPL will serve well the need for a language that is faster than BASIC yet easier to program than assembly language. The package is well-thought-out and well-documented."**

RPL uses the ordinary Commodore BASIC screen editor for program entry and editing. And the full power of BASIC, in both immediate and program modes, remains available to the user throughout a development session. The RPL Compiler and Symbolic Debugger reside in the top 8K of memory, ready to be invoked at any time, directly from BASIC, via the commands "compile" and "debug". RPL source code is saved to disk or cassette just like BASIC source, and is compiled memory-to-memory for quick compilation turnaround and instant source accessibility. RPL supports separate compilation of program modules through the use of the compiler's "global symbol" features, which also permit the development of true "subroutine libraries".

The language itself is concise and straightforward, making it much easier to learn and master than most other computer languages. A total of only 47 special keywords and symbols provide the following capabilities:

- Nestable, multi-line IF . . . THEN . . . ELSE constructs.
- Nestable FOR . . . NEXT loops.
- Named subroutines and functions of arbitrary length.
- Compile-time constants and code ORGability.
- Full 16-bit integer arithmetic and logical manipulations.
- Built-in character-string handling.
- Stack-management directives including n-index, n-rotate.
- GET, INPUT, and PRINT operators
- Forward and backward symbolic references, including GOTO.
- Easy access to machine language.
- Predefined arrays with numeric and/or string contents.
- Local and global symbols.

. . . and *much* more. The 60-page RPL manual is clear and well-organized, making the language easy to learn and easy to use: **Loren Wright** says that **"the documentation is about the best I have ever seen."**

The Samurai RPL Symbolic Debugger is a screen-oriented, object-level debug facility using a soft-key-driven command syntax for ultra-ease of use. Features included are:

- Full visibility into both stacks at all times.
- Single-stepping, with source-level next-step display.
- Breakpointing in both auto-single-step and "go" modes.
- Address specification using expressions with symbols.
- Stack-edit capability on both stacks.
- Debugger video usage is transparent to target program.
- Extra run-time error-checking during debugging only.

. . . and, of course, much more. Here's what **Robert Baker,** author of the PET-pourri column in Kilobaud Microcomputing, says about it:

> **"RPL offers an unbeatable combination of speed, memory space efficiency, and ease of use. It is well-designed, well-implemented, and well-documented, and it deserves the serious consideration of every PET/CBM programmer. The Samurai RPL Symbolic Debugger, in particular, must be seen to be believed."**

The compiler includes a special option making it very easy for you to create "execute-only" object modules from which all development-utility software and memory allocations have been excluded. The price you pay for the compiler also includes an unlimited license to resell the RPL "run-time library" (*not* the compiler) in conjunction with "execute-only" application object modules of your own.

The Samurai RPL Compiler is now available at the special introductory price of $49.95, which includes the manual in a nice 3-ring binder and First Class postage within the continental U.S. Media supplied is of top quality, and is *not* copy-protected (this permits you to make backups for yourself without hassles). Compiler and debugger together are **$80.91, complete.** Manuals are available separately at $10.00 and $4.00, respectively, and will be credited toward software purchase. Please specify machine type, memory size, ROM version, and media type (cassette, 4040, or 8050 diskette) when ordering.

Order anytime, day or night, 7 days a week

Outside Florida:

800-327-8965

(ask for ext. 2)

Within Florida: **305-782-9985**

VISA and Master Charge accepted

All orders shipped within 2 days of receipt

(For technical inquiries, please phone 305-782-9985)

For more information, or to order by check or money order, please write:

SAMURAI SOFTWARE
P.O. Box 2902
Pompano Beach, FL 33062

MICRO SOFTWARE SYSTEMS

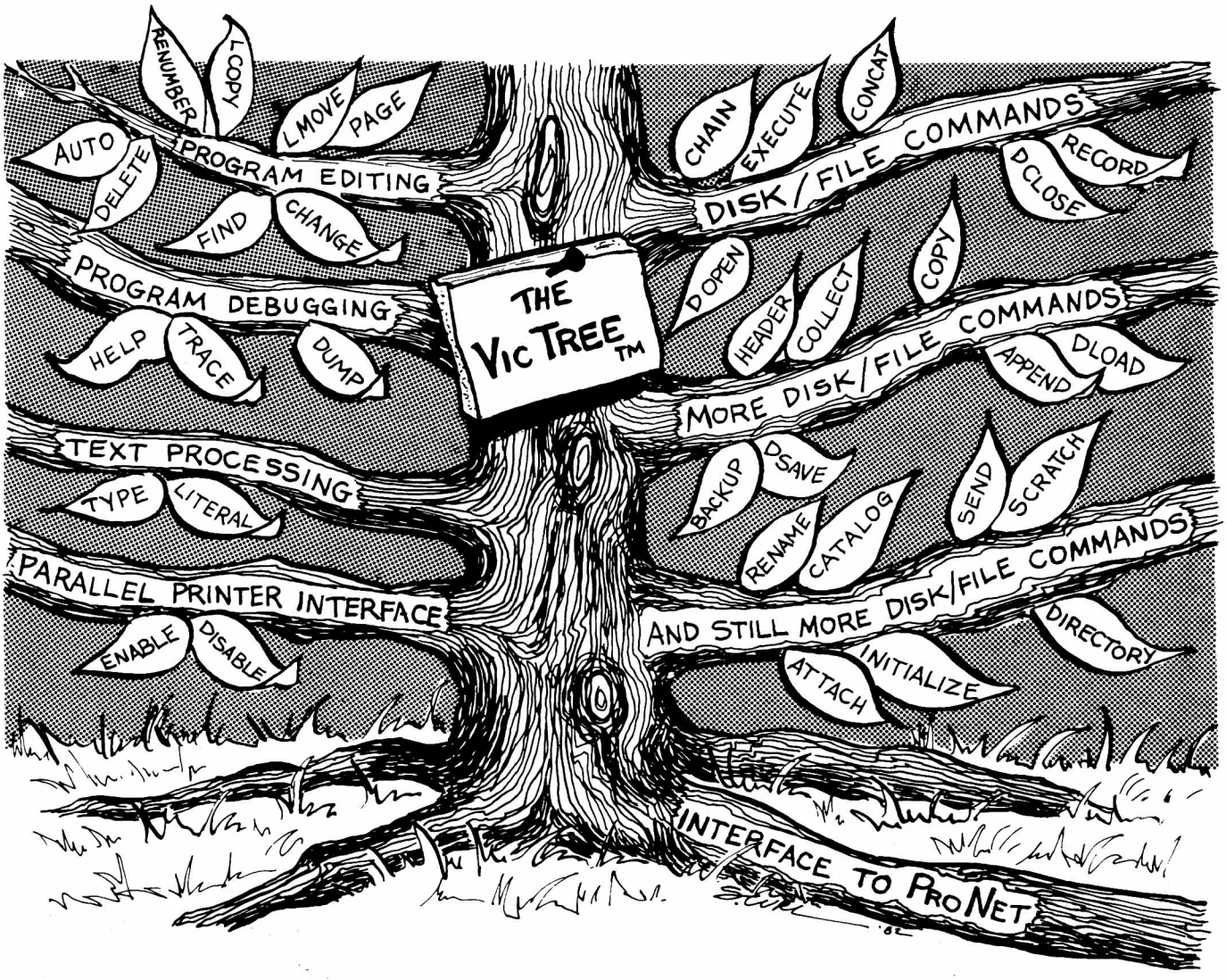
P.O. Box 1442, Woodbridge VA 22193

HARDWARE	LIST	MSS
SuperPET	1995	1676
CBM 8096 Upgrade to 8032	500	420
CBM 8032 Computer	1495	1256
PET 4032 Computer	1295	1089
PET 4016 Computer	995	836
VIC-20 Computer with Modulator	325	289
CBM 8050 Disk Drive	1795	1495
CBM 4040 Disk Drive	1295	1088
CBM 2031 Disk Drive	695	584
C2N Cassette Drive	75	69
CBM 4022 Printer	795	669
EPSON MX-80 Printer	649	499
GRAFTRAX 80 graphics ROMs ...	95	75
EPSON MX-80 F/T Printer	749	649
EPSON MX-100 Printer	995	799
IEEE Interface	59	51
ITOH STARWRITER Printer	2074	1695
XYMEC HY-Q 1000 Printer	2885	2499
NEC SPINWRITER 7730 Printer	3195	2695
IEEE Interface & Cable	149	125
TALLY 8024 Printer	1995	1689
DIGI-PLOT PLOTTER	1495	1295
DIGI-PLOT 6-COLOR PLOTTER	1995	1676
CBM 8010 MODEM	279	234
DC HAYES SMARTMODEM	299	259
SMARTMODEM + McTERM software	494	399
ESCON IEEE/Selectric IF & Cable	695	635
ESCON IEEE/Selectric 50/60/70 ..	595	499

SOFTWARE	LIST PRICE		MSS PRICE	
TITLE	CASS	DISK	CASS	DISK
Billboard (8032)	39.95	44.95	33.56	37.76
Disk Librarian	29.95	34.95	25.16	29.36
Word Pro 4 Plus		450.00		378.00
Word Check		195.00		164.00
Visi Calc (CBM)		199.00		168.00
Ozz		395.00		332.00
File Cabinet		69.95		59.00
Create-A-Base		295.00		248.00
JINSAM SYSTEM #8.		795.00		668.00
JINSAM SYSTEM #4.		695.00		584.00
JINSAM SYSTEM #1.		395.00		332.00
MAGIS Business Pkg		2495.00		1995.00
PASCAL (TCL)		295.00		248.00
PASCAL (WI)		75.00		63.00
fullFORTH+		55.00		46.20
Program Toolkit		39.95		35.60
Command-O		79.95		67.20
Matrix Sort ROM		54.95		49.20
Spacemaker II		39.95		35.60

MIN ORDER $25. FREE CATALOG. ADD 5% S&H. VA ADD 4%. VISA/MC OK

Skyles Electric Works Presents



The VicTree™

...Leaves your new Vic (or CBM 64) with 35 additional commands.
...Branches out to most BASIC 4.0 programs.
...Roots into most printers.

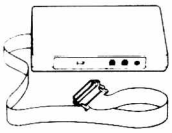
New from Skyles: the VicTree, a coordinated hardware and software package that allows your Vic to branch out in unbelievable directions and makes it easier than ever to do BASIC programming, debugging and to access your disk. And the new VicTree provides routines to interface the Vic to the powerful ProNet local network. 8kb of ROM — 4kb for the BASIC commands, 4kb for disk commands and interfacing to ProNet — plus 4kb of RAM for miscellaneous storage. Perfect not only for the new Vic but also for the Commodore 64. Unbelievably simple to use and to install, the VicTree gives you all the additional BASIC 4.0 commands to allow most BASIC 4.0 programs to work on your new Vic or CBM 64.

Now only $89.95...or **$99.95** complete with Centronics standard printer cable. (Cable alone **$19.95.)** Available now from your local dealer or order through your Visa or MasterCard toll free: (800) 227-9998 (California, Canada, Alaska, Hawaii: (415) 965-1735) or send check or money order directly to:

Skyles Electric Works

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

SIGNALMAN MARK I DIRECT CONNECT MODEM – $89.50



Standard 300-baud, full duplex, answer/originate. Powered by long lasting 9-volt battery (not included). Cable and RS-232 connector included.

EPROMS – HIGH QUALITY, NOT JUNK

Use with PET, APPLE, ATARI, SYM, AIM, etc. 450 ns. $6.50 for 2716, $12.50 for 2532. We sell EPROM programmers for PET and ATARI

5¼ INCH SOFT SECTORED DISKETTES

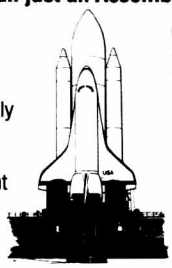
Highest quality. We use them on our PETs, APPLEs, ATARIs, and other computers. $22.50/10 or $44.50/20

NEW! C. ITOH STARWRITER F-10 DAISY WHEEL PRINTER

Letter quality, flawless copy at 40 char/sec. Bidirectional printing, 15-inch carriage, uses standard Diablo ribbons and print wheels.

PARALLEL – $1495, RS-232 – $1680, TRACTORS – $210

More than just an Assembler/Editor!



It's a Professionally Designed Software Development System

MAE

for PET APPLE ATARI $169.95

Blast off with the software used on the space shuttle project!

- Designed to improve Programmer Productivity.
- Similar syntax and commands – No need to relearn peculiar syntaxes and commands when you go from PET to APPLE to ATARI.
- Coresident Assembler/Editor – No need to load the Editor then the Assembler then the Editor, etc.
- Also includes Word Processor, Relocating Loader, and much more.
- Options: EPROM Programmer, unimplemented opcode circuitry
- STILL NOT CONVINCED: Send for free spec sheet!

ATARI AND PET EPROM PROGRAMMER

Programs 2716 and 2532 EPROMs. Includes hardware and software. PET = $75.00 –

Commodore Computer owners:
Are you tired of long waits to load and save on Cassette? Like to have a disk but cannot affort it. Then try the next best thing to disk - announcing

The Rom Rabbit

and a cassette version for VIC at $29.95

"Your Wish.... ...Is My Command"

Easy to install ROM!

Loads and saves an 8K program in about 30 seconds. Try it - your Pet normally takes 3 minutes!

Pet Owners Granted Wishes:

1. Much faster cassette load
2. Auto-repeat on all keys
3. Memory test
4. 12 commands
5. Works with or without toolkit

Visa and M.C.

ROM and Manual $49.95

Specify 3.0 (2001 PET) or 4.0 (4001 or 8032)

C. Itoh Prowriter Printer. Better than MX80. We use constantly with our Apple and PET. Can be used on IBM, Atari, TRS-80, etc. 120 cps, friction and tractor feeds, hi resolution dot graphics, nice looking, high quality construction. Parallel – $499.00, with IEEE interface for commodore – $599.00, RS232 – $660.00

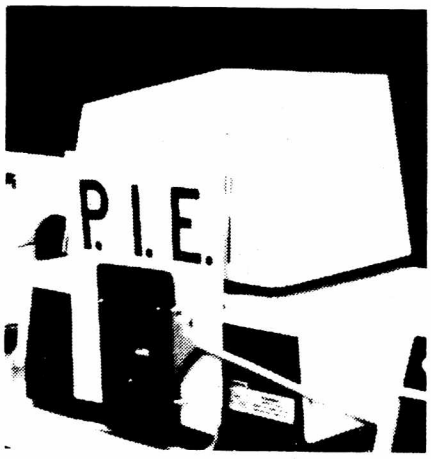
Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!

VISA MasterCard

PIE-C

PET/CBM* IEEE-488 TO PARALLEL PRINTERS By LemData Products



P.I.E.-C MEANS—Professional design, Indispensible features, Excellent quality and Cost effectiveness. You can't buy a better parallel interface for your PET/CBM.

Our P.I.E.-C will interface your PET/CBM through the IEEE-488 bus to . . . . . .

the NEC Spinwriter, the C. Itoh Starwriter, printers by Centronics, Epson, Anadex, Escon Products, the Paper Tigers by IDS, the MIPLOT by Watanabe, the DIP printers, the AJ-841, the OKIDATA printers, plus ALL OTHER parallel ASCII printers.

Assembled with custom case, CBM-TO-ASCII code converter and appropriate cable, the **P.I.E.-C** is only 129.95 (+$5 S&H). Md. Res. +5% tax. Specify printer and CBM models.

LemData Products, P.O. Box 1080, Columbia, Md. 21044 Phone (301) 730-3257

*PET/CBM are trademarks of Commodore Business Machines

Briley Software

BUYERS! - 90 Day Load; 1 Year Bus Warranty
AUTHORS! - Send one-page abstracts
DEALERS! - Send for terms, Demo Disk $30
SOFTWARE HOUSES! - Be a Co-op part of us!

PUBLISHER OF APPLICATIONAL SOFTWARE

- Since 1979 -

PROGRAMS FOR COMMODORE PET/CBM

C001P Business Researcher/$50/16K/simplx decision tool/algb,rqrd

RNAV3 Navigator Series (VOR/DME & VOR-VOR trip planning):
C002P Pacific/$25/8K/6 states
C003P Western/$30/16K/11 states
C004P Northeast/$30/16K/17 states

Bowling Secretarial System - Rvwd KILOBAUD(6/82-Baker):
D010P LeagueBowl-24/$145/32K/2040-4040
D010C LeagueBowl-24/$160/32K/8050
C011P ArchiveBowl/$40/32K
C012P TournamentBowl/$40/16K

Data Management Series (In-Memory File Handling):	Per File
C101P Deluxe Addresser/$40/16K/1 & 2-up labels/4 line	215(32K)
C102P Home Address Book/$25/8K/use cass only/NO label	312(32K)
C103P Home Inventory/$20/8K/use cass only	374(32K)
C104P Price Shopper/$20/8K/use cass only	375(32K)

Kitchen Software (Data Stmts): Rvwd MIDNITE#5/PERS.COMP.(Nov81)
C105P Grocery Mart/$15/8K and COMP.MERCHANDISING(May82)
C106P Dinner's On!/$15/8K

Educational Fun & Games /8K each/use cass only:

Deductive Detective Series/$15ea/Elem/JHS/Rvwd MIDNITE#4
C201P Mansion! C202P Museum! C203P Pentagon!

Deductive Explorer Series/$15ea/Elem/JHS
C204P High Seas C205P Fur Trapper

Test/Drill/Answer Sampler/$15/High School
C301P Ed.Pack (Quad.Equ./Volumes & Areas/Gas Law Equ.)

PROGRAMS FOR COMMODORE VIC-20
C103V Home Inventory/12K/$20
C105V Grocery Mart/5K/$15
C106V Dinner's On!/5K/$12.95
C210V BlackJack Tutor/8K/$15
...& MORE ARE COMING!!

TO ORDER: Send Check(US Funds), add tax if Calif., $.75/cass for shipping. Give address, program title(s) & code(s).

We can ship COD(U.S.), $3.50 Admin Fee PLUS req COD fees.

BRILEY SOFTWARE Box 2913 Livermore,CA 94550-0291 415/455-9139

VIC 20/PET/CBM OWNERS

WALLBANGER - Blast your way through the dodge'm, blast'm, and attack modes. If you destroy the bouncing balls before they destroy you, the walls close in for the next round. Wallbanger is written in machine language, has great sound, and encourages complex strategies.

CASS/5K/VIC 20/CBM 8032
CASS/8K/40 COL SCREEN/OLD-NEW ROMS/FAT FORTY . . . $15.00
(CALIF. RES. ADD 6% SALES TAX)

MILLIPEDE - Exterminate the oncoming millipedes and fleas as they descend through the mushroom patch. Blast giant bouncing spiders before they pounce on you. Shoot a millipede in the body and suddenly two millipedes descend toward your ship. Millipede is written in machine language, has excellent graphics, and great sound.

CASS/5K/VIC 20/CBM 8032
CASS/8K/40 COL SCREEN/OLD-NEW ROMS/FAT FORTY . . . $15.00
(CALIF. RES. ADD 6% SALES TAX)

Write for FREE game details:

ON LINE SOFTWARE
P.O.BOX 2044
ORCUTT, CA 93455

WARNING! These games cause high panic levels!

VIC 20/PET/CBM OWNERS

80 x 25

PET/CBM™

2000/3000/4000 Series

not using a CRT, or display controller chip

$275.00*

Select either
80 x **25** or **40** x **25**

On The Built-in Display

From the keyboard or program

Displays the full, original character set

Available from your local dealer or:

EXECOM CORP.
1901 Polaris Ave.
Racine, WI 53404
Ph. 414-632-1004

*Plus installation charge of $75.00

Available only for Basic 3.0 & Basic 4.0

PET& CBM™a
trademark of Commodore Business Machines

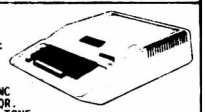
VIGIL

Exciting, new games interactive language.
* Easy to learn with 60+ powerful commands
* Double density graphics, large number display
* LOADing and SAVEing of VIGIL programs to cassette or diskette
* Nine complete programs included - Breakout, SpaceWar, AntiAircraft, U.F.O., SpaceBattle, Concentration, Maze, Kaleidoscope & FortuneTeller.
* Comprehensive 50+ page manual
* For OLD, NEW or 4.0 ROMS with 8K of memory

	U.S. & CANADA	FOREIGN
VIGIL for PET/CBM on cassette or diskette w/9 programs	$35	$40
VIGIL for VIC on cassette (requires 3K memory expander)	$35	$40
VIGIL User's Manual (refundable with software order)	$10	$12
VIGIL Interpreter Listing (6502 Assembler Language)	$25	$30

PET

PET & APPLE II USERS TINY PASCAL

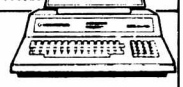


Structured language alternative to BASIC for PET or APPLE II includes:
* LINE EDITOR - creates, modifies and maintains source language.
* COMPILER - converts your source to an executable P-code format.
* INTERPRETER - executes compiled P-code. Features built-in TRACE.
* CASE-OF,WHILE-DO,IF-THEN-ELSE,REPEAT-UNTIL,FOR-TO/DOWNTO,PROC,FUNC
* Graphics version has more: GRAPHICS,PLOT,POINT,TEXT,INKEY,ABS,SQR. APPLE II has lores & hires-COLOR, HGRAPHICS,HCOLOR,HPLOT,PDL and TONE

	U.S. & CANADA	FOREIGN
TINY Pascal PLUS+ GRAPHICS PET 32K NEW/4.0 ROMS diskette	$50	$60
TINY Pascal PLUS+ GRAPHICS PET 32K NEW/4.0 ROMS cassette	$55	$65
TINY Pascal PLUS+ GRAPHICS APPLE II 48K and DOS 3.2/3.3	$50	$60
TINY Pascal NON-GRAPHICS PET 16K/32K NEW/4.0 ROMS diskette	$35	$45
TINY Pascal NON-GRAPHICS PET 16K/32K NEW/4.0 ROMS cassette	$40	$50
TINY Pascal NON-GRAPHICS APPLE II 32K/48K and DOS 3.2/3.3	$35	$45
TINY Pascal User's Manual (refundable with software order)	$10	$12
TINY Pascal 6502 Interpreter Listing-GRAPHICS version	$25	$30
TINY Pascal 6502 Interpreter Listing-NON-GRAPHICS version	$15	$20

Plus + GRAPHICS

TINY BASIC COMPILER - PET



A true compiler that turns your BASIC program into fast machine code
* Subset of PET BASIC compiles to 6502 machine code.
* Has full floating point capabilities and functions.
* Compiler listing optional with 16K version (included).
* Can load compiled machine code anywhere in memory.

	U.S. & CANADA	FOREIGN
TINY Basic Compiler-OLD/NEW/4.0 ROMS min. 8K-casette/diskette	$25	$30
TINY Basic User's Manual (refundable with software order)	$10	$12

PET MACHINE LANGUAGE GUIDE

Now in its ninth printing. Learn the hidden talents of your OLD, NEW or 4.0 ROM PET/CBM with the easy to follow manual. Details 30 of the PET's built-in routines.

	U.S. & CANADA	FOREIGN
PET MACHINE LANGUAGE GUIDE for OLD, NEW or 4.0 ROMS	$9	$11

ABACUS SOFTWARE
P. O. Box 7211
Grand Rapids, Michigan 49510

master charge
616 / 241-5510
VISA

Prices include post
Orders must be prepaid via check, money order or bank-card. Foreign orders may be paid for via international money order or bankcard: (Access,Eurocard,Barclaycard)

SYSRES™

THE ULTIMATE RESIDENT PROGRAM MANIPULATION SYSTEM FOR PET™/CBM™ MICROCOMPUTERS

SO COMPLETE, EVEN THE BEST OF THE COMPETITION DOESN'T COMPARE!

EXTENDED DOS SUPPORT

@ (type "N" keyboard)
← (type "B" keyboard)
! (original keyboard)
> (for 'wedge' users)

These commands may be used interchangably, to perform the following dos support functions.

Command	Function
@	Display disk status / send command
@N	Format (header) a new diskette
@I	Force initialize diskette
@V	Validate diskette (collect)
@D	Duplicate diskette
@C	Copy or concatenate disk file(s)*
@R	Rename file
@S	Scratch file(s)*
@$	List directory*
@U:	Reset disk drive
@L	List disk file or BASIC program*

* Added/enhanced disk command.

EXTENDED EDITOR

Command	Function
/	Quick load from disk
↑	Quick load from disk with auto run
APPEND	Append from disk to end of current program
AUTO	Auto line number (allows header)
BLOAD	Load machine language (binary) file
BRUN	Load and execute machine language program
CHANGE	Change pattern to another pattern
CLOSE	Close one or all files
CMD	Set output to file (does not send "READY.")
DELETE	Delete a range of lines from program
DUMP	Dump all scalar variables to screen or file
EXEC	Execute a file as keyboard commands
FIND	Find occurances of a pattern
GET	Read a sequential file into editor
KEY	Define a key as a special function
KEYS	Turn key functions on
KILL	Disable SYSRES™
KILL*	Disable SYSRES™ and unreserve memory
LIST	Improved BASIC LIST command
LOAD	Defaults to disk drive
MERGE	Merge from disk into current program
MON	Break to current machine language monitor
OLD	Restore program after "NEW"
PUT	Send program to disk as text file
RENUMBER	Renumber all or part of program
RUN	Run current program, ignores screen garbage
SAVE	Defaults to disk drive, allows replace
SETD	Set disk device #, allows multiple drives
SETP	Set printer channel, format mode, paging
TRACE	Select 1 of 3 trace/step modes and speed
VERIFY	Compare current program against disk/tape
WHY	Print position of last error
WHY?	List line of break or error
*	Send output to printer
#	Display current version of SYSRES™

COMPARE SPECIFICATIONS!

	SYSRES™	POWER™
Number of ADDED commands	33	13
Number of IMPROVED BASIC commands	7	none
Number of DOS SUPPORT commands	11	none
Approximate added syntax options	1200	60
Instruction manual length	86 pages	75 pages
Instruction manual style	structured	conversational
Re-loactable?	yes	no
Use on more than one (any) PET/CBM™	yes	no
Upgradable	yes	no

COMPARE FEATURES!

	SYSRES™	POWER™
Automatic printer output?	yes	no
Selectable ASCII conversion?	yes	no
List programs without loading them?	yes	no
Formatted program listings?	yes	no
Dump SEQuential/RELative files?	yes	no
Edit data files?	yes	no
True program merge?	yes	no
Auto number with **AUTO TEXT**?	yes	no
Load machine language programs?	yes	no
Auto-execute machine language programs?	yes	no
Directory (menu) file commands?	yes	no

COMPARE "EQUIVALENT" FUNCTIONS!

Function: Change occurances of one pattern to another.

Feature	SYSRES™	POWER™
Command word	CHANGE	@
'Wild cards' in search string?	yes	yes
'Wild cards' in replace string?	yes	no
Selectable range?	yes	yes
Match in entire text?	yes	yes
Match in commands only?	yes	no
Match exact variable names?	yes	no

Function: Define special one-key functions.

Feature	SYSRES™	POWER™
Command word	KEY	REM"
Requires BASIC program changes?	no	yes
Destroys variables?	no	yes
Re-define any key?	yes	no
Maximum string length	255	73
Quotes and carriage-return allowed	yes	no
Re-define any token key?	yes	no
Retain user keys from program to program?	yes	no

JUST A FEW OF THE FEATURES OF SYSRES™

* Fast up/down scrolling which works!
* Advanced repeat-key routine!
* Re-define any or all keys as any keyword (full or short form) or as any string up to 255 charactors long!
* Auto line numbering which can feed a string of up to 127 charactors as well!
* Extended DOS support (requires DOS 2A or greater)!
* Never enter another file name! All file commands work from the directory!
* Supports multiple disk drives!
* List BASIC programs, sequential and relative files without loading them into memory!
* TRUE PROGRAM MERGE (overlay). Supports subroutine libraries!
* Load and run machine language programs with parameter passing!
* Supports multiple printers!
* Automatic printer output with paging plus formatted listings with full ASCII code conversion including cursor control and special charactors for non-CBM™ printers!
* Edit text files and assembler source code without leaving BASIC!
* Renumber part of a program or even change the order of lines!
* Over 700 FIND/CHANGE commands including variable names ("A$" will not match "BA$"), pattern matching with "wild-cards", and even commands to remove spaces and REM's!
* Three TRACE modes including trace variables!
* Does not affect BASIC program operation!
* One AUTO-BOOT DISKETTE works for ALL PET™ or CBM™ computers (BASIC 2.0 or greater with at least 16k of RAM.). SYSRES™ requires NO ROM SPACE or extra boards, so you can take it with you if you want to use another computer. It may be put above the screen if you have RAM there. It boots automatically without disturbing any program in RAM!
* If, **for any reason**, you are not satisfied with the SYSRES™ system, you may return it along with any back-up disks (within 30 days) for a **full refund.** Your disks will be erased and returned to you.
* Diskette and Extensive Manual – only **$95** (Please specify disk drive model when ordering.)

CALL US FOR THE NAME OF YOUR NEAREST DEALER

We are pleased to announce the aquisition of the author, Don Lekei, and the rights to SYSRES™. Don is now hard at work producing versions of our **STOCKFILE™** series of integrated **INVENTORY CONTROL, POINT OF SALE, ORDER ENTRY,** and **BILL OF MATERIALS** packages for the PET/CBM™ computers. The best inventory control system for the APPLE][™ will soon be available for the CBM™. See your local dealer for details!

CANADA
Tel: (604) 984-0477
#6, 144 West 15th St.
North Vancouver, B.C.
Canada V7M 1R5

UNITED STATES
Tel: (206) 734-3744
#204, 4202 Guide Meridian
Bellingham, WA
U.S.A. 98226

SOLIDUS INTERNATIONAL CORPORATION

ISLAND SOFTWARE

Announces

PROGRAMS FOR THE GIFTED AND TALENTED

THE MINDSTRETCHER SERIES — — A unique set of microcomputer programs specifically designed for gifted and talented students in grades 3 through 9.

PROGRAMS WILL OPERATE ON ANY 8K PET

TAPE

MS 1. *Jigsaw* - - - Four programs, with a total of 16 picture puzzles to assemble, ranging from a view of New York City to Whistler's Mother $20.00

MS 2. *Traffic Jam / Chain Reaction* - - - Two programs. Both of these provide exercise in strategy, as you try to force your opponent into a vulnerable situation $20.00

MS 3. *Rubik / Candles* - - - Two programs. Both of these increase in difficulty to challenge the student as he develops his problem-solving skills $20.00

MS 4. *Black / Kayles* - - - Two programs. Deceivingly simple rules, but the strategy in these two contests makes use of advanced mathematical theory $20.00

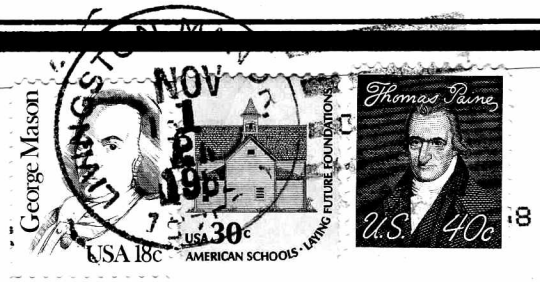
MS 5. *Jinx / Welter* - - - Two programs. Two unique diversions to develop deductive reasoning and insight into the structure of mathematical abstractions $20.00

Every program is packaged with a teacher's guide sheet which describes the history of that MINDSTRETCHER and many specific teaching suggestions based upon actual classroom use.

Please send your order with a check or purchase order to:

ISLAND SOFTWARE
Box 300
Lake Grove, N.Y. 11755

The PAPER
Box 460
Livingston Manor, NY 12758



FIRST CLASS