THE PAPER

VOLUME IV, Issue 3 | A CSD PUBLICATION | $3.00

Table of Contents

++

The Rumor Mill

Rumors in the computer industry are commonplace and stories about Commodore products are no exception. The stories about Commodore are not discouraged by the company and are, at times, encouraged by their silence. If you hear any rumors pass them along and we'll publish the latest and the best.

Acccording to some dealers Commodore plans to introduce a new business machine which will be one of the most attractive on the market considering its capabilities, price and options. The CBM II, as it is being called, will sport an 80 column swivel mounted monitor, a more contoured keyboard, and space for two built in disk drives. In addition, it will have 128K of internal memory made possible by the new 6510 processor. It seems that this new machine will have an IEEE-488 port and a true RS-232 interface as well. It will be CP/M compatible which means it can run standard MBASIC and many other packages. The capability to connect several machines to one set of peripherals will also be built in. Just how much of this will be included for the $1995 base price is anyone's guess. Exactly when and if this machine will become available is also a matter of speculation.

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. POSTMASTER: Mail all address changes to the address above.

All readers are encouraged to submit articles of general interest to PET users. Materials submitted must be free of copyright restrictions. The contents of The PAPER are not copyrighted. All articles remain the property of the author and may be reprinted with their permission. When reprinting please include a note stating that the article was originally published in The PAPER.

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 2

Printed:	700	USA:	520	Canada:	42	Foreign:	15

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:

Long Live Commodore

The last comment that appeared in this space has generated more response than anything I have written in two years. Some of these responses are printed in the Reader I/O section. The title of that piece was 'The Late Great Commodore' and in it I stated that I felt Commodore would disappear unless it became more responsive to its users, potential users, and dealers. I still maintain that this is true but I also notice some changes at Commodore. I must also agree that an electronics firm that has been in business as long as Commodore, weathering the 'Great Calculator' wars, is probably a survivor. Also, it should be clear that no one hopes more than I do that Commodore does remain and grow stronger.

Recently, Commodore has announced many new products including computers and peripherals. They did it with a confidence that has been lacking in the past. Of course, Commodore has announced before and never delivered, witness the 8-inch disk, but this seems different. The first products included several different versions of hard disk drives with up to 9.6 Megabytes of storage. Presumably to back these up Commodore also presented a 30+ Megabyte tape drive system. The only flaw I can see is the production dates which implied you might actually be able to get some of this by the summer. I doubt seriously whether your local dealer will have these before the end of this year. Commodore has further announced what seems to be three new computers at a recent west coast show. The exact specs for these are unclear but the ideas seem exciting. Ultimax is a color game machine with a membrane style keyboard and full control of color and sound. I have heard stories from several sources on this machine. Some say the basic machine is NOT programmable in BASIC while others say it is. Commodore envisions it as a competitor with Atrai's VCS and the other TV game consoles. According to some, it will be produced as a fully capable computer from the start, unlike Mattel's Intellivision, and will retail for around $150. Others say it will start out as a game machine only. In any case, Commodore is predicting the sale of 100,000 per month. The people at Commodore also seemed confident about another machine they call the Commodore-64. This computer is VIC's big brother and is housed in the same packaging. It too will connect to a TV but that's where the similarities end. It sports a full-sized keyboard, 64K of memory, and an extended BASIC to provide color and sound control. It is said to have more graphics power than Atari and a truly amazing three voice sound generator. An optional Z-80 cartridge is said to be in the works to provide the facility to run CP/M. Best of all the rumor is you will get this for just under $600. This is obviously and attempt to meet Apple head-on and will be a serious challenge if everything goes as planned. The biggest problem here is trying to combat the oceans of software now available for The Apple. A somewhat less firm announcement concerned the Emulator, a machine, that with appropriate attachments, would run software from other computers, primarily Apple and TRS-80. I am not sure whether this is a good approach or not but the prospect should make some companies a little uneasy. All-in-all Commodore seems very active in prototyping and announcing products which, if priced and supported correctly, will make it number one in the industry in the US. There is also the SuperPET which, although not brand new, seems to be catching on with many people. I feel it was exceedingly smart of Commodore to produce a machine that acts just like an 8032 but with the flip of a switch changes personalities entirely. The ability to program in all those different languages on one machine for under $2000 is remarkable.

Other factors seem to indicate Commodore is here to stay and one of these is the increasing volume of good software. CBM/PET has always had to fight the statement "There's no software for this machine." I have heard this comment over and over again, many times issued by well respected names in the industry. It is true that the game programmers seem to shy away from the PET because of the lack of color and high resolution graphics. However, the business software for the Commodore line is impressive. There may not be the volume but look at the

quality. Remember, one good word processor or data base replaces one dozen inferior products. Word processors like WordPro and Wordcraft, business packages from BPI and the CMS, specialized products for legal and medical accounting, data base managers like The Manager, Flexfile and Jinsam, programmer's utilities like the MAE assembler, HES products, POWER, and language systems like fullFORTH+, Comal, TCL Pascal and RPL outshine many products for the competitor's machines. And the software keeps coming in the educational field also. It gets harder and harder for the detractors but they still persist. Companies like Cimmarron have announced new poweful packages like their random access file system and a CP/M box will soon be on its way from England. The list grows daily.

The attitude at Commodore seems different also. Perhaps the recent immigration of Kit Spencer has made a difference but I think I detected a change even before that. Commodore first contacted me for samples of The PAPER to put in a large dealer mailing and then again for information for a school district information folder. This constituted a major step in support for Commodore which had been noted for its lack of support. Recently they have offered to provide The PAPER with more current information for publication. As a software vendor, Commodore has contacted me in regard to displaying my educational programs at various shows. I don't know if this will help either of us but I don't see how it could hurt! I hope this change in attitude continues and grows.

There are still some negative points which Commodore must overcome. It is still hard to get helpful technical information when you call Commodore. I think the problem is more in finding the person who knows rather than an actual unwillingness to inform. I know this is a problem for both indivduals and dealers alike. Commodore is also seriously backordered on some equipment and the resulting wait puts off many buyers and dealers. I hope Commodore is preparing for a possible increase in volume and production capabilities as their new products hit the market.

In summary, I did not mean at all to count Commodore out but to indicate a problem I saw as potentially crippling. There are still problems but many are working themselves out. I really can't say that Commodore has risen like the Phoenix from ashes since things have never been that serious. Let's hope they never have to.

Background/Foreground

This column has evoloved into a combination of announcements of meetings and products. It also contains observations and questions about today's microcomputer world. If you have an announcement or a short observation this might be the place to get it in print.

Technology and Special Ed Conference

The New York State Association for Educational Data Systems will sponsor a regional conference dealing with "Computer Applications in Special Education" on Saturday, May 22, 1982 at the Mill Neck Manor School in Mill Neck, NY. The conference is for teachers, parents and administrators and will run from 8:30 AM to 2:30 PM. There will be nine workshops showing how computers can be an important part of the learning experience for handicapped and nonhandicapped students alike. Each presentation will be repeated twice and exhibits will be on display. Pre-registration is $20 while late registration is $30. Contact Dr. Dolores Shanahan, Commack Public Schools, Indian Hollow Computer Lab, Kings Park Rd., Commack, NY 11725 for pre-registration or conference information. Her phone number is (516) 493-3573.

Christian Computers ?

No, it's not really a new cult or even a splinter group for religious micros. However, a new group is being formed to organize churches who use computers or who would like to. The founders of the group believe that religious groups should be organized like the business and scientific community. If you are interested in more information conatct: Christian Computer/Based Communications, Alderwood United Church, 44 Delma Dr., Toronto, Ontario, M8W 4N6.

Mail-Order Menace

A recent court decision has upheld Apple's right to tell their dealers how they can sell computers. Specifically, Apple has banned all mail order sales of their products. The announced reason is the ability of mail order firms to undercut the local dealer and their inability or unwillingness to provide service. The possibility exists that Commodore could do the same thing as several local dealers have suggested. How do you feel? Is the very low cost of mail order purchases worth the lack of service? Should we protect the consumer against himself? What about the experienced user who just wants a good deal? How about the person who is 100 miles from the nearest dealer?

Piracy and Intimidation

As of this writing the latest injunction in the "Great Pac-man Wars" belongs to Atari and Warner Brothers. The ruling essentially says that Atari owns the home video game and computer rights to anything that even resembles the munching of dots. The programming is not a factor. All it has to do is LOOK like Pac-man! Atari will not even license anyone to produce a version even though it has helped sell their machine. The only version which they have ready is the one for the VCS. Even if they do produce an Atari version, what about all the owners of other machines? How long can the smaller companies keep up the court battles against a giant like Warner? If Atari wins, what implications does this have on the field of computer software? The issue of copyrights and piracy is far from clear and the fights are far from over.

Speaking of Copies...

When will Commodore get its own Hong Kong look alike. Radio Shack had the PMC which was completely compatible and sold at a much lower cost. Tandy sued and I haven't seen many PMC ads lately. Now a company called Franklin is producing the Ace 100, an Apple act-alike for somewhat under Apple's price. Indications are that care has been taken to avoid any Apple patents particularly on the hires graphics which are missing in the initial model. What about a Commodore copy with all the PET features plus a few more for a smaller price tag? You've got to be kidding! Even Hong Kong can't do that!

Patent Your Software

Let's face it a copyright on software is next to useless. The laws are so unclear that a few changes in code and presentation may qualify the result as a new program which can be sold without any payment to the original author. A patent is a different story since it affords more protection and has different financial implications. However, it is a lot harder to get. It took S. Pal Asija seven hard fought years to patent his high speed information retrieval system called SwiftAnswer. According to Asija software must be "new, useful and unobvious" to be considered for a patent. Unobvious means that the techniques used must not be obvious to the average programmer. Visicalc might qualify under these criteria but many programs although "new and useful" would not.

NYSAEDS Annual Conference

The seventeenth annual conference of the New York State Association for Educational Data Systems will be held November 7-9, 1982 at the Americana Hotel in Albany. The theme of this conference is "Moving Ahead with Instructional Computing". The four major strands to be addressed are administrative uses, curricular issues, hardware, and programming.

Administrative uses will include library applications, networking, and using Visicalc. Major curricular issues to be discussed are computer modifications for the handicapped and problem solving with the gifted and talented. In depth analyses of the major microcomputers, including the new IBM, are being planned. Presentations on LOGO and PASCAL are being developed.

Conference fees will be $200 including registration, two night's lodging, banquets both evenings and Monday lunch.

For more information, contact Gary Bruce, Program Chairperson, 55 School St., Delevan, NY 14042

ABACUS Announces...

ABACUS Software has just announced several new products for the VIC computers. These are not games but serious software packages for programmers. The first package is a version of the popular VIGIL graphics language. This includes all the features of the PET version but also has commands to use all of the features of the VIC. Commands exist to control color, sound, joysticks and a light pen. The code produced as VIGIL runs much faster than BASIC and is ideally suited for games. An 80 page manual and demonstration tape with 9 programs are included. The price is $35.

VIC HiRes/Multicolor is a set of two utilities to help the VIC programmer easily work with high resolution and multicolor graphics. HiRes allows you to plot 104 by 152 points without any additional hardware. Points can be plotted and lines drawn. Multicolor reduces the resolution to 52 by 76 but gives an additional color. Both programs add commands directly to VIC BASIC and come with a manual and sample programs. The price is $25.

VIC PIPER allows you to compose, save, recall and playback music on the VIC with no extra hardware. Notes are entered as alpha notation with rests and durations easily entered. You can vary volume and tempo, play harmony and automatically load and run additional compositions. A manual and sample compositions are included in the $25 price.

See their ad in this issue.

Baker Software Special Offer

Bob Baker is offering alomost ALL of his excellent programs and utilities packed on one floppy disk. A second disk contains all the documentation for these programs. This package includes programs like Disk Master, XREF, Compactor, Uncompactor, Assembler/Editor, Disassembler and Word Pro Converter. In addition games like Black Friday, Bowling, and Word Hunt are included. These programs are written in BASIC but are very useful and entertaining. Compactor takes a BASIC program and deletes all REMs and spaces and then combines lines to shorten programs. This extremely useful when a program gets a little too large for your memory and is worth the price by itself. The two disks are just $29 postpaid in 4040 or 8050 format. Mastercard and VISA are accepted.

When I saw your update to the "Keyprint" program, I thought I finally had a good screen dump program but, alas, my Epson printer does not recognize the lack of carriage returns. Do any of your readers have any ideas on making the program work with an Epson MX-80?

Also, is there any method of blanking out the screen during graphics updates on the 8032? The older machines allowed POKE59409,52 but this has apparently been changed. The same is true of the control register to turn off/on the cassette motor. Can anyone help?

I have attempted for some time to obtain a copy of an extended monitor for use on my 8032. I received a version of Supermon from AB Computers, but it would not run. Does anyone have such an animal on disk or tape?

Finally, I would like to contact other PET users in my area to establish a users group. - Stan Spence, 5147 S 37th St., Lincoln, NE 68516

Stan - I don't have an 8032 so it is hard for me to research the problem. For the same reason I have not run across or had to solve these problems. Maybe some of the readers can help. As far as I know there is no way to blank the screen except by using machine language routines. This probably doesn't belong in an answer to a letter so I have included a short article entitled "SWAP-PUT" in this issue. This program automatically relocates itself to your high memory and allows you to create a copy of the screen in high memory. You can then send the screen to its copy or vice versa. You may also swap the screen and the copy or simply clear the copy. Please read it carefully and let me know if there are still any questions on this matter. Again, since I don't have an 8032 I don't know whether my copy of Supermon would run. It does work on BASIC 4.0 and I'll try to check it out. There are no other subscribers in the 685 zip area. If you get a group started or meet any other users, I sure would like them to subscribe. - Ralph

No doubt you are wondering when you will receive the first issue of the DENSPET (International MTU Hi-res Users Group) Newsletter. We had planned to wait a month or two to see if any more responses to our various magazine announcements would arrive. There has been a total of only five replies counting yours and several magazines have never printed the notice. This being the case, I have regretfully decided not to try to continue - Franks Chambers

Frank - I am sorry to here about the lack of response although I can truly say I know how you feel. Any reader who owns an MTU board should get in contact with Frank immediately. He has some interesting programs and could use the support.
Write: Frank Chambers
Rock House
Ballycroy, Westport
Co Mayo, Ireland

Regarding your comment "Copy This If You Can", I think the solution for schools is to teach programming. Students can then write programs to the specs supplied by teachers. Most high school students should be able to develop drill programs for the lower grades since this type of program is not hard to develop. Many programs of this type have been published and would be helpful in developing others. It would be an incentive to the student to see his program used in the system. I have found that most commercial programs do not do what they are supposed to and must be modified. A program which cannot be copied or listed is of little value.

I have a PET with 32K and a Computhink disk. When I bought the "Toolkit" nothing was said about where it was located which happens to be right where the Computhink disk system sits. It was impossible to use the two together and the

company would not help me with a ROM in another spot. The relocation program had been withdrawn from the market so I had to learn how to relocate it myself. - Jack Clark

Jack - I can't disagree with you on many points. I would say that good drill programs are not always that easy to write. Providing assistance when a student gets something wrong is the real challenge. Also, making programs student proof takes some time. However, I think you are correct when you suggest that high school programmers could do this. It is the only way to ensure that programs will do the job the way the teachers want it done. - Ralph

It didn't help much to have an error on page 24 of issue 2 in the "Machine Language..." article. I guess I learned more trying to figure out what was wrong and noticing the correct listing on another article on page 22. It seems that in the hex dump on page 24 the 8D's should all be changed to 9D's. Are there any other errors? - Jack Clark

Jack - It's letters like this that let me know that someone is reading what I write. You are indeed correct and I apologize to you and even more so to those that didn't realize there was an error. I was even more embarassed when one of my 8th grade students caught the same mistake. For those that would like to know, we were supposed to be using indexed absolute addressing to store something on the screen. The op code for that is 9D but instead I used 8D, the op code for simple absolute addressing. The result was that four locations on the screen each got filled 256 times. Read the article again and make the changes Jack suggests. - Ralph

I now have an MX-80 printer and Wordpro 3 hooked to my PET and disk. I have been unable to find any documentation on a PET hooked up with an MX-80 printer. I have had no success getting vertical spacing. I have tried the "ln" command and just hitting carriage returns but neither works. If you or the readers know of any solutions or have any suggestions, please let me know. An article or a book would be helpful. - Tim Amodemo, 4 Briarcliff Rd, Shoreham, NY 11786

Tim - I don't have the equipment mentioned but I have heard of some people having problems. I know that the Epson handles carriage return-line feed different than the PET printer but I don't know the solution. Maybe the article in this issue will help. Meanwhile, how about it readers? - Ralph

I strayed on a copy of The PAPER last week and congratulate you on a first-rate publication. I am an Englishman studying for a year at Notre Dame and I am also a contributing editor to PRINTOUT. Your readers may be interested to know that I have programmed a character generator for the Epson MX-80 printer to enable it to do PET graphics. It is a single chip which replaces the old one and can be fitted in about three minutes. All the usual features are available plus TRS-80 block graphics. However, no cursor controls are printed and reverse field characters appear normal. The chip comes with full instructions and has been fully tested. Further details may be obtained by sending $2 to:

Micro Electronics
32 Orchard Ave
Worthing
Sussex, ENGLAND

A firm in England is offering this chip at a substantial discount. It is a pirated sample we sent and contains several bugs. - John Nuttall

John - I have read your material in PRINTOUT and am glad to hear from you. Thanks for the information. I'm sure some readers will be interested. Your last comment is heard all too often and is a blot on that company's name and the industry in general. - Ralph

I'm looking for a software/hardware package to "read" RTTY (teletype) transmissions from my SW radio directly on my PET. Can you help me? - Daniel Condon, 11723 SW 102 Ct., Miami, FL 33176

Dan - This is another one for the readers. Send your info to Dan but pass it along to me since others may be interested. - Ralph

Our district is very much interested in interfacing our Fat-40's to a video monitor. We use a buffer to make this connection on older PETs, but this doesn't work on the new ones. Do you know where such an interface can be obtained? Since we need 30, cost and ease of use are factors. - Marg Farrand, Ann Arobor PS, 344 Gralake, Ann Arbor, MI 48103

Marg - Many people, myself included, have asked the same question. All the ads I see exclude the new Fat-40's. Many people would appreciate knowing if an interface is commercially available. - Ralph

About the "Late Great Commodore" editorial in the last PAPER...Commodore isn't half as late as the latest issue of The PAPER! I still feel that Commodore will be around for a long while after many of its fasionable competitors have folded their tents and disappeared. Any 22 year old electronics company has to be doing something right! Our new dealer in Springfield is having trouble keeping equipment in stock, and VICs, at least, are selling like hotcakes all over the world. IBM may win big in the end, but it will be a while before your local dealer has any to sell, and a while after that before you can get the dollar value out of an IBM that you do now out of a Commodore.

The fact that nearly everything Commodore promised to bring out this year is in fact working and being delivered, with manuals, says a lot. That the equipment can do more than anything from a competitor that is even close to its price range says more. And finally, there is now a wide range of top-quality software available directly through Commodore. Some of this, notably the 8096 version of VisiCalc and the DTL Compiler is better than anything comparable available for Apple, Atari, etc.

What do you need in the way of subscribers or donations to keep The PAPER alive? - Jim Strasma

Jim - Your comment about the late issue really hurts particularly since it is correct and you once warned me that six issues a year is hard to do. Also, because this issue is again late. Material keeps piling up but getting it typed and printed is painful. Don't get me wrong, I'll live. Much of what you say is echoed in my editorial this time. I agree that Commodore will be around for a while to come. If IBM or any other machine takes a large share of the market it will be more because of service offered than the technical excellence of the machine. I am finding that many business people are put off by the idea of taking an ailing computer to the local dealer. I try to combat this by pointing out, as you do, the technical and economic superiority of Commodore and the excellent programs available. Nobody should fall into the trap of believing that a technically superior machine will necessarily rule the market. Service and customer support have a lot to with sales and Commodore has improved some in this area but has a way to go. At this point, I doubt if anything can keep The PAPER alive for another volume. This may just be the winter blues but the facts say otherwise. We now have 475 people that get the newsletter. About 425 are paid and 50 promise to. I think that the demise of this newsletter would be

too bad but, obviously, only 475 other people think so. I cannot spend money on expensive advertising since I must be sure to have enough left to meet my commitment to the present subscribers. Any thoughts or suggestions would be gratefully accepted. - Ralph

Its good to see Volume 4! I hope you will take the following suggestion seriuosly and change the name to The PAPER: For PET/CBM/VIC. I think part of your circulation problem, a large part, comes from your lack of identification with the Commodore line. You are one of the few publications dedicated solely to Commodore. COMPUTE! is good but is being diluted by space provided to APPLE, Atari and even the Radio Shack Color Computer. The new Commodore magazine is taking an approach which indicates they now see alternate publications as synergistic rather than competitive. I tell people about The PAPER and show them copies but the word would spread faster with a name that proclaims your committment to PET/CBM/VIC. Advertise in COMPUTE! and Commodore! - Gary Stone

Gary - It's letters like this that make me want to eat the responses like the one I gave Jim above. Only this kind of support will keep us going. As I write this I think I will take your suggestion and change the masthead including a new machine since I think it could only help. As soon as I finish this, I am going to write a letter to the Commodore editor and send him some issues. A mention in their publication would help. I want everyone to know that I tried advertising in COMPUTE! I sent in my copy and my money with the blessing of the ad agency and then boasted to friends and advertisers about the upcoming ad. I even planned on increasing my subscriber list to over 1000. Imagine the frustration and indignation I felt when both ad and check were returned a month later. It is COMPUTE's policy not to accept ads from rival publications. The suggestion that they let their ad agency know this fell on deaf ears. I have been accused of various things by COMPUTE which only indicates that they view The PAPER as a competitor. They have mentioned that they will be running an article covering all the little guys soon. - Ralph

I am an intermediate school librarian with only a year and a half experience on the PET. I have just read the first issues of The PAPER cover to cover and understood about 10%. But I know they are full of information I want to learn and it looks like I made a good investment. I'm teaching beginning BASIC to kids before school and spend all my free hours nights and weekends learning programming skills and writing simple programs. I have six computers in my library and am trying to get other teachers to use them. The teachers look to me for answers to their questions. For example, if a BASIC program includes sound and is run, it will not save to another tape. If the same program is loaded but not run it will save. Why? Please include some articles for beginners or direct me to some books that can give me help. I find the PET manual very confusing. - Marilyn Nicholson

Marilyn - Thanks for reminding me that there are many subscribers who are just beginning. Those of us who have been at it a while welcome you and will try to help. I find it hard to write beginners articles but I will try. Let me know what you think of the first part of my BASIC graphics article. Your question on sound also reminds me that repeating information is no sin. Programs with sound contain a POKE59467,16 somewhere. As far as we are concerned this turns on the sound and disables the cassette deck. Things will appear to save correctly but will not and a subsequent load may bomb the machine. Be sure to POKE59467,0 before recording any program which may have sound or just don't run it until you record. The PET manual from Commodore is hopeless for a beginner. You should have gotten a book published by Osborne/McGraw-Hill with the PET which is quite a bit better. I hear the third edition of this book is even better. - Ralph

The MTU graphics board will give a 320 x 200 dot resolution on a Commodore 16K or 32K computer. Do you have any information on dumping the hi-res screen to a dot graphics printer such as the Epson MX-70/MX-80 or the new Commodore CBM 8023P which just came on the market? - Joseph Baker, 28048 E Broadway, Walbridge, OH 43465

Joseph - I have been able to dump hi-res to my 4022 but know nothing about the 8023P. An article in the next issue concerning the dumping of hi-res to the Epson may help. Also you might want to contact Frank Chambers whose letter and address are printed above. - Ralph

It should be possible to use two utilities which both use the CHRGET or wedge method. Initialize both utilities from power on and look at $0079 to $007D. You will most likely find a 4C followed by an address. This address will be located somewhere in the utilities' address space. This is a jump (JMP) to some code that checks to see if what the user entered is interesting to the utility. If it does not concern that utility, execution passes on to some code that duplicates what the JMP displaced. If the utility sidetracks normal execution by replacing compare statements with JMPs at $0079, why can't the same compare code within the first utility be replaced in the same way with jumps to the second utility? Working this out would be a good way to learn more about machine language.

It should be easier to chain together interrupt driven utilities. Locations $0090 and $0091 usually contain $E62E for BASIC 3.0 and $E455 for BASIC 4.0. Sixty times a second execution goes to the address in $90/$91 and the keyboard is checked, clock updated and so on. A utility puts its own address in $90/$91 and ,so, gets called 60 times a second. When through, it sends execution on with a jump to the E address. Use the H (hunt) command in Supermon to find that jump and replace that address with the address the other utility wants in $90/$91. - Jim Yost, PO Box 556, Somerville, MA 02143

Jim - Sounds very possible in both cases to me although I don't know that much. It certainly seems like a lot of work that can and should only be undertaken by someone knowledgable. You are probably right that it is an excellent way to learn but it is not a satisfactory solution for the user who just wants two utilities to work at the same time. Other problems crop up also. Some utilities are meant to reside in the same memory space while others use the same place in memory for work space. - Ralph

I have an original 8K PET which I recently converted to 32K and upgrade ROMs. I've converted all my programs to work on the new system except for SWORDQUEST and ESCAPE from the DEATH PLANET from Fantasy Games Software, originally in Madison, Wisconsin. The problem with trying to convert them is that both have lines at the beginning which won't list but contain BASIC commands and ML subroutines. The company is now gone and I can't get any information. I would appreciate it if anyone could send me fixes or complete working programs. - Rudolph Lauer, 11 High St., Nutley, NJ 07110

Rudolph - This supports my position that software protection is wrong and violates the consumer's rights in many ways. Companies will not be around forever and when they go they seldom leave anything behind. I do not have the programs myself but would be willing to bet someone has them for the upgrade ROMs. Can anyone help? - Ralph

OBSERVATIONS on Vol. 4, Issue 2

by Roy Busdiecker

When the package arrived, I could hardly believe its weight! The last issue of The PAPER had 48 pages of articles, and twelve pages of ads... and the quality was better than ever. Anyone who was ever participated in the production of a "newsletter" has got to be impressed. Actually, it doesn't seem accurate any more to refer to The PAPER as a newsletter. It seems more like a magazine these days.

Machine Language

From the number of articles and support products showing up, it's obvious that machine (assembly) language is of growing interest to a great number of PET/CBM owners. This is a natural and healthy development, and provides a significant economic benefit as well. For little or no additional investment , a PET/CBM owner essentially gets a new computer... the 6502 assembly/machine language computer, as opposed to the BASIC-speaking machine with which he is already familiar.

My nominee for best article of the issue is "Adding Commands to BASIC Using CHRGET", by Doug Haluza. It addresses a very interesting subject in the most understandable manner I have seen yet. Doug's presentation of the routines themselves includes complete listings, showing both assembly "source" code and hexadecimal "object" versions, along with the addresses of the memory locations in which the object code is stored. To simplify entry of the code, the "no-frills" monitor listing was provided, too.

In the text, Doug explained the situation, and why the CHRGET routine is important. Then he described the operation of his utility routine, which is a genuinely useful tool. He even went one step further and provided different versions for the various versions of BASIC.

One of my few complaints is that Doug failed to provide the SYS command to activate the routine (unless I overlooked it). Looks to me like the 1.0 and 3.0 versions call for SYS 826. The 4.0 takes a SYS 634, which was tested on my 8032.

It would have been nice, too, to give a little more explanation of the CHPCH routine, which had no comments in the Assembly Listing. If you try to figure out the routine by taking the commands at face value, you might conclude that the section from 320 to 370 would cause and "inflation loop" if you press any key except "backarrow". (BEQ ARROW does nothing unless the last key pressed was "backarrow"... the next command is JMP CHPCH - go back two commands. There's no way out!)

The Solution

The mystery unravels when you consider that the code in lines 320-370 gets transferred to locations $0070-0086 by the SWAP routine, while the corresponding section of CHRGOT (p37) is placed where CHPCH used to be. Once the transfer has occurred, the JMP CHPCH in line 340 no longer points to the code in line 320, but to the code residing in location $0355. After the swap, that would be the line following the CHRGOT on p37! Instead of creating an infinite loop, the JMP CHPCH really transfers control to the CHRGOT routine in its new location. Neat, but not intuitively obvious.

All things considered, Doug's article provides a good model for future articles in machine language routines. Several additional items I would like to see in such articles include highlighting the differences in code for the different versions of BASIC (either in the assembly listing or monitor listing), and to blank out meaningless code (like the last three bytes of each monitor listing on page 40). A handy way to do that blanking out, by the way, is to use

the screen print routine.

The routine works nicely, and is a handy one to have. After the usual amount of frustration that one expects when modifying someone else's machine language routines, I've managed to modify the program to peek or poke a series of sixteen locations starting at the one specified... for my purposes, that makes it even more useful. For example, to examine all the BASIC program/variable pointers, I simply enter (left arrow) 0028, and see the whole array. I'll put this together in a follow-up article.

Other Machine Language Articles

Jerry Key's article, "The Monitor", on page 17, addresses a subject on which many questions are asked by beginners. By all means, Jerry, continue. Do take it more slowly, though, and explain each step...the article covers too much, too fast, for someone who doesn't already understand. There was also and article on the same subject in Commodore's "Microcomputer Magazine" (formerly "Interface", formerly...) bearing a cover date of October 1981.

"Machine Language for the Beginner" reflects the perspective and quality that we've come to expect from Ralph Bressler. My only suggestion, Ralph, is that you include the assembled "object code" with your assembly listing. It makes understanding things a lot easier.

Essential tools for machine language programmers include a good reference on the 6502 itself, a good assembler (program), and a good extended monitor. My choices for those items are, respectively: "6502 Assembly Language Programming" by Lance Leventhal, published by Osborne/McGraw-Hill; MAE (Macro Assembler/Editor) from Eastern House Software; Extramon, Supermon, Micromon all in the public domain with the last two published in COMPUTE!

I/O from/to Readers/Editors

The I/O section has always been one of my favorites, dating back to the early "Dear Terry" letters. The most recent column was no exception, providing several "gems" (one or two gems, that may take up only a few lines of text, are often worth the whole price of the publication).

Getting KEYPRINT to work of the 8032 is something I've been looking foward to ... seems a lot more useful than the 40 column versions, because the 80 column screen fills the width of a "normal" (10 pitch) typewriter page. Ed Steinfield's fixes worked as advertised. Thanks for your initiative, Ed! If you use the escape key for other purposes, you may want to continue to use the backslash key to activate the routine. If so, the value to be POKEd in location $036A should be $DC rather than $9B.

Since the monitor can be used to store screen memory as well as programs, the combination of KEYPRINT, screen editor (built in), and monitor makes an interesting "pauper's word processor". With a little practice, I think that might be a very satisfactory solution for occasional users. To solve the 4.0 DOS and Fat Forty problems, I relocated the routine to the high end of memory.

MTU Visible Memory

A note to Larry Riley and other MTU Visible Memory Board owners: while it's possible to use the MTU memory board for storage of some protection ROMs, it's also convenient to plug the others (like VisiCalc) in the sockets provided on the board. Sockets U37, U66, and U75 are used for the ROMs addressed in the $9000 block, while U46 and U57 are used for $A000 addresses. In order to activate a particular ROM, just POKE the appropriate value in location 48895. For U37, the command takes a value of 128, while U46 takes 16 and U57 requires 64.

Our demonstration unit at Virginia Micro Systems is used alot for VisiCalc,

so we implemented the wiring change to make that ROM active when the computer is turned on. To use MTU, we must POKE 48895,6 to enable the visible memory.

The other POKE location to note is 49151. A value of 0 blanks the screen, without losing the information displayed there. A 1 restores the normal PET/CBM video, while a 2 enables the visible memory video. An interesting effect can be obtained by using a value of 3, which overlays the normal and high resolution displays (they are logically OR'ed).

Relocatability

Second the motion! I whole-heartedly endorse Vanderbilt Foster's proposal to keep machine language utilities out of the second cassette buffer. Put them at the top of memory. To avoid conflict, and solve the 8/16/32K computer problem, take the time not only to make them relocatable but self-relocating! For the folks smart enough to do the routines in the first place, there's not a lot of additional effort required to write a BASIC program to relocate the code based on the current values of the memory pointers, then reset those pointers and issue a NEW command. That way, we could load several utilities in succession, and they need not conflict with each other (at least in the address space in which they reside).

Ralph, you might want to go into the details of that approach in your series on machine language.

New Products

The blockbuster is SuperPET! The more I use this machine, the better I think it is. Extended BASIC gives back the functions we've done without on all the small computers (not just Commodore). The PASCAL is a good implementation, and the APL (also a good version according to our local "experts") does not seem to be available on any other reasonably priced machine. Flip a switch, and it's an 8032, ready to run WordPro 4+, VisiCalc, or other valuable commercial programs. It's a SuperBARGAIN!

While 8050 is the recommended disk drive, SuperPET will work with the 4040, or even the 2031, giving you all those language capabilities for around $2700. I've used that configuration, and it's really not inconvenient to work with the single drive. Rather than simply enter "p" for PASCAL, for example, you must enter "disk8/0.p", and so on.

To copy the system diskette from 8050 to 2031 (or 4040), you cannot use the UNIT TO UNIT utility provided on the CBM demo disk, because the programs are too long (try it, and see what happens!). Instead, use the routine provided with the 2031 to change its device number to 9, then put the SuperPET in 6809 mode, load EDIT, and enter: COPY DISK8/1.PASCAL,PRG DISK9/0.PASCAL,PRG. Of course, both drives must be connected.

****

40 to 80 Column Conversion

Competitive Software will soon be offering a kit which will allow owners of the Fat-40 to switch select between 40 and 80 columns. The kit will contain all the parts needed with detailed instructions for installation which requires soldering. The price will be $99.95 + $2 shipping. This will, of course, void any warentee. This means that a user could purchase a 4016 for about $795, add 16K of memory for around $25, and the conversion kit for $99.95 and have an 8032 for well under $1000. It will also switch between 40 and 80 columns which no Commodore machine can do.

Beware the Epson Jabberwock

by A H McCann

My system includes a CBM 2001-32B computer, a 2040 dual disk, a 2023 friction printer and a cassette drive. A nice system which I use constantly for my engineering calculations, accounting, word processing and some games. The 2023 printer was chosen so I could insert my letterheads to print my correspondence. It was never completely satisfactory for a variety of reasons; i.e. poor print quality, no descenders for lower case letters, uncontrollable registration for printing mailing labels, meandering of roll paper and so forth.

When the Epson MX-80 FT came out with both friction and tractor drive, at a reasonable price, it looked like the answer to all my problems. It even included an IEEE-488 interface for the PET at a price of only $55. Little did I know the frustration I was to encounter.

I ordered a printer and the interface from the Omega Sales office in Newberry Park, CA and waited impatiently for 5 weeks until UPS delivered an impressive blue and white carton. Inside I found my printer, the IEEE-488 interface board, a nicely bound, 107-page User's manual for the MX-80 printer, a 10-page User's manual for the interface board and a special 21-page Operation manual for the MX-80 FT printer. This last manual had several pages of instructions on unpacking and installing the printer, and on insertion of various configurations of paper. There was one page of MX-80 FT control codes with no instructions for their use and no examples. Not much of an Operation Manual.

I sat down with the manuals and read each one carefully from cover to cover. The User's Manual for the printer, by David Lien, was obviously written for the TRS-80. On page 14 there is an illustration of two sets of DIP switches on th main PC board of the printer, with instructions for the proper setting (on or off) of 8 rocker switches on SW-2 and 4 switches on SW-1. However, these settings are only for the TRS-80. For PET or APPLE you are instructed to ignore the illustrated switch settings and go directly to appendix D.

Appendix D gives no instructions for the PET and "specifically EXCLUDES hardware considerations, such as how to set the twelve internal switches" and refers you to the IEEE-488 User's Manual. This manual has good instructions for installing the interface board, which plugs into the top of the parallel board and covers the DIP switches mentioned in the big MX-80 manual. Another similar set of DIP switches is installed on the interface board. It is explained that the six switches on SW-1 are used only for setting the device number. Only one mandatory setting of the four switches on SW-2 is given. There is a cautionary note on page 6 stating, "once the IEEE-488 I/F Board is installed, the parallel interface of the printer cannot by used". There is no further reference to the DIP switches on the parallel interface board.

I inferred from all this that the original DIP switches were bypassed by the circuitry of the IEEE-488 board and should be ignored. Well, I put the thing together and plugged it into my PET. I tried the printer test and got all the characters in the printer, including the TRS-80 graphics. But, when I tried to use programs with PRINT# statements there was no line feed and when I tried WordPro 2, the result was graphics characters instead of capital letters.

I reread the manuals and reinstalled the interface board to no avail. I still got graphics instead of caps and caps instead of lower case letters. Finally I called the Epson office in Torrance, CA and talked to one of their software people. He explained that the original DIP switches were not bypassed and gave me the proper settings for a line feed. However, he said that the graphics/caps abd caps/lower case problem was due to the non-standard ASCII used by the PET and the conversion was beyond the scope of the Epson IEEE-488 interface board. He had no solution to offer to this problem.

I called the CBM office in Santa Clara, CA and talked to someone in the customer service department. He was aware of the deficiency of the Epson I/F

board, which is useless for word processing applications. CBM has no interface for sale, but I was referred to Gary Brooks of Brooks Systems, Pope Valley, CA, (707) 965-7269. Gary has a very nice interface for $129 which connects to the parallel connector on the MX-80 FT printer.

This means, of course, that the installation of the Brooks interface starts with the removal of the Epson IEEE-488 board, which becomes a $55 piece of junk. The Brooks interface accepts device numbers 4 or 5 from the computer. If 4 is given a CBM printer is assumed by the interface and PET ASCII is passed through unchanged. If device 5 is used the interface assumes a non-CBM printer and makes the necessary conversions to standard ASCII. Let JOY abound throughout the land! The WordPro works at last!

There was one problem with the Brooks interface. It has 7 DIP switches mounted on the face of the case. Four of these switches must be set properly for the MX-80 FT. When the interface is plugged into the IEEE-488 connector on the PET the switch handles can contact the PET case if you push hard on the interface to seat its connector. This can reverse some of the switches, which results in no response from the printer. SO!!! don't push it in that hard.

The MORAL of this sad tale (with a happy ending) is --- don't buy the Epson IEEE-488 I/F board if you want to use a word processing program. It's OK for PRINT# statements and for CMD output, but not for correspondence. The happy ending is that the MX-80 FT is truly a versatile printer when the proper interface is used. The Brooks interface works all the magic needed to utilize all of the capabilities of the printer. An ancillary advantage of this interface is that I can attach both my 2023 printer and Epson to the PET simultaneously. Whichever one is turned on does the printing.

Checking The Keyboard

by Jerry Key

Since the inception of the 8032 and the Fat-40, lots of keyboard interactive programs have been known to do rather strange things, not the least of which is to not work! To find out what's going wrong, check the program listing for a PEEK(151) or converting of keyboard input to ASCII (A$=ASC(A$)). Chances are if it's there, it is being used to obtain the value for things like controlling the movement of characters or objects on the screen. The problem is that many of the keys on the Fat-40 and 8032 have been assigned new values to make them more compatible with standard ASCII. That's what they tell us! However, on the 8032, the numeric keypad keys are assigned a different value than the numbers at the top of the keyboard! Some future use? Anyway, there is a way to check these values. Type in and run the following short program.

```
100 PRINT "(clr)"
110 PRINT " PRESS A KEY"
120 GET A$:IF A$="" THEN 120
130 PRINT A$, PEEK(151)
140 PRINT "(1down)":GOTO110
```

To spice it up a little, change line 130 to:

```
130 PRINT A$, ASC(A$), PEEK(151)
```

Perhaps this little routine will help you find out how to convert your standard 40 column programs to the Fat-40 and 8032. For anyone that hasn't seen the numeric conversion tables to the Fat-40 check COMMODORE Dec 81 or COMPUTE! Oct 81. For the ASCII value of the numeric keypad on the 8032, add 127 to the standard Fat-40 values.

Graphics and Animation in BASIC: Part I

by Ralph Bressler

I have always felt that using the pages of this, or any, newsletter to teach basic BASIC was wasteful. Several good books exist to teach beginners how a PRINT statement or an IF...THEN.. functions. However, few books explain exactly how to combine BASIC statements to produce a graphic display or how to animate that display. This is quite easily done in BASIC as I hope to show you. Some more advanced users may argue that effective graphics animation can only be done in machine language. While it is true that machine language adds speed and versatility it is not the only way to produce displays. In fact, the principles of producing good graphics is best done in BASIC where we can easily see our program and the results it produces at a reasonable speed. After these principles are learned some people may want to experiment with other means of creating displays.

To me graphics includes anything that is not text to read. This includes the simplest picture created with PRINT statements to the most complex animation. Let's start at the beginning and use print statements to create a picture. The best way to do this is to create the picture first and then make it into a program. We can use any of the PET's special graphics and its excellent cursor controls. To begin, type POKE59468,12 to make sure the PET is in graphics mode and then clear the screen. Draw whatever picture you like on the screen exactly as you want your program to print it when finished. For reasons which may become clear later place the picture a little to the right of where you actually want it to appear. As you do this avoid hitting the RETURN key which might lead to a ?SYNTAX ERROR. Also steer clear of the CLR key unless you really want to start over. When you have finished your masterpiece, HOME the cursor and move it down the left side of the screen until it is on the same line as the first characters in your picture. Now simply type a line number followed by a question mark, a quote and a RETURN. Continue to do this until the last line of the picture. You now have a program or, at least, part of a program which will reproduce your picture each time it is run and can be saved for future use. As mentioned before you will notice that the picture prints farther to the left than it did when first created. This is due to the fact that you took up some of the space on the left with the line number, question mark and quote. If you desire to be exact, position your picture to the right two spaces plus the length of your line number. Let's say you want your program to print the picture 15 spaces over and you wanted the first line number to be 1000. You should offset your picture six spaces to the right when drawing it. This is fine for large, stationary pictures. Another technique is more useful for diagrams which are smaller or may be moved from place to place.

Small pictures may be defined as string variables and printed by simply saying PRINT A$. This is very handy and can account for some fairly large graphics. It does take some thought but is really worth it. Suppose we wanted to create a set of dice to be defined as D$( ). When we want to print a certain die we need only say PRINT D$(5) and the graphic representation of a five will appear. The key here is consistency. Each die will be considered to have 9 possible positions for spots. Of course, not all will be filled at any one time. A six and a five would look like this:

```
* *     * *
* *      *
* *     * *
```

For a six we type star-space-star, go back three and down one and repeat this three times. A five is little different except that the second line is space-star-space. Even so we go back three and down one to get to the next line.

For a one we type three spaces, space-star-space and then three spaces. Still, in each case, we go back three and down one. Our entire set of dice might look like this: (Remember that I cannot show the cursor controls so I type things like (2 down) instead.)

```
D$(6)="* *(3left 1down)* *(3left 1down)* *"
D$(5)="* *(3left 1down) * (3left 1down)* *"
D$(4)="* *(3left 1down)   (3left 1down)* *"
D$(3)="*  (3left 1down) * (3left 1down)  *"
D$(2)="*  (3left 1down)   (3left 1down)  *"
D$(1)="   (3left 1down) * (3left 1down)   "
```

This is actually one of the easier tasks to be done this way. Doing all the dice the same way makes things easier and also means printing one die on top of another will completely erase the first. A small racing car might look something like this:

```
   .
  +=+
 O---O
```

We might represent it as:

```
C$=" O---O (7left 1up)  +=+  (7left 1up)   .   "
```

Notice the inclusion of an extra space in front and to the rear of the car. If we were to move this car across the screen in increments of one space, it would appear to move smoothly and leave no characters behind.

This brings us to simple horizontal motion. PRINT TAB( ) will easily accomplish this task. The following program will move our car across the screen.

```
10 PRINT"(clr)"
20 C$=" O---O (7left 1up)  +=+  (7left 1up)   .   "
30 FOR I=1 TO 30
40 PRINT"(home 3down)TAB(I)C$
50 FOR J=1 TO 100:NEXT
60 NEXT I
```

Line 40 will print the car one more space to the right each time through the loop. The 'home 3down' sequence insures that the car will always be printed on the same screen line. The way we set up the car with a trailing space will make the motion seem relatively smooth and will erase any unwanted parts of the car. Line 50 was added as a time delay. The program below shows a more complete program using horizontal motion. It has two car on a short track with random 'speeds'. I chose to use simple an X and an O for the cars but more elaborate graphics could be used as we did above.

```
100 REM CAR RACE
110 :
120 REM RANDOMIZE RANDOM NUMBERS
130 X=RND(-TI)
140 REM PRINT TRACK
150 PRINT"(clr)S              PET PRIX              F"
160 PRINT"-------------------------------------"
170 FORI=1TO5:PRINT"/                                    /"
180 PRINT"-------------------------------------"
200 :
210 REM SET INITIAL CAR POSITIONS
```

```
220 P1=1:P2=1
230 REM PRINT CARS IN POSITION
240 PRINT"(home 3down)"TAB(P1)" O"
250 PRINT"(1down)"TAB(P2)" X"
260 FORI=1TO50:NEXT I: REM DELAY
300 REM DECIDE WHO MOVES THIS ROUND
310 IFRND(1)>RND(1)THENP1=P1+1
320 IFRND(1)<RND(1)THENP2=P2+1
330 REM WAS THERE A WINNER ?
340 IFP1<38ORP2<38THEN240
350 :
390 REM ANNOUNCE LUCKY DRIVER
400 PRINT"(home 12down)"
410 IFP1=P2THENPRINT"IT'S A TIE!!
420 IFP1<P2THENPRINT"DRIVER #2 WINS!":W2=W2+1
430 IFP1>P2THENPRINT"DRIVER #1 WINS!":W1=W1+1
440 GOTO150
```

The program is commented and is quite straightforward. Notice that in lines 240 and 250 a space precedes the 'car' graphics so that, when it moves, it gets erased. The variables P1 and P2 keep track of the progress of each car. Lines 310 and 320 decide which car(s) will move on each round by picking two random numbers for each car. Line 340 tests to see if either of the cars has reached the finish line at position 38 on the screen. The lines from 390 on simply decide who won and print the result.

Vertical motion with something moving up the screen is just as easy. You should know by now that printing anything at the bottom of the screen will cause everything on the screen to move up one line or scroll. The following program causes a rocket to blast off and disappear off the screen.

```
100 REM ROCKET TO THE MOON
110 :
120 PRINT"(clr)"
130 REM PRINT ROCKET AT SCREEN BOTTOM
140 PRINT"(home 24down)"TAB(20)"!"
150 :
160 REM COUNT BACKWARDS FOR COUNTDOWN
170 FORI=5TO0STEP-1
180 PRINT"(home)"I
190 REM DELAY ONE SECOND
200 FORJ=1TO1000:NEXT J
210 NEXT I
220 PRINT"(home)BLASTOFF!"
230 PRINT"(22down)"
240 :
250 REM SCROLL SCREEN 24 TIMES
260 FOR I=1 TO 24
270 PRINT
280 REM MAKE ROCKET TRAVEL FASTER AS IT
290 REM MOVES HIGHER
300 FOR J=1 TO 500-I*25:NEXT J
310 NEXT
```

Again, this program requires little more explanation than that contained in the REM statements. The rocket is printed at the bottom of the screen by line 140. Line 300 is a time dealy which gets shorter as the rocket moves higher giving the impression of acceleration. Again the idea can be expanded and more complex graphics can be employed. This technique is not useful if you do not want the

entire screen to scroll or if you want to move things down the screen.

The following program combines upward movement without screen scrolling with the single string variable graphic technique to produce a short 'game'. Each part of the program is commented but some things will need explanation especially for real beginners. In this game a balloon is released from the ground and rises toward a row of spikes which will break it. The speed is random and at some point the balloon becomes invisible. The object is to stop the balloon as close to the spikes as possible without breaking it. Here is the program:

```
100 REM DON'T BREAK THE BALLOON
110 :
120 REM VERTICAL TAB
130 DN$="(home 24down)"
140 REM RANDOMIZE
150 X=RND(-TI)
160 :
170 PRINT"(clr)"
180 REM TURN ON NOISE
190 POKE59466,81:POKE59467,16
200 :
210 REM PRINT SPIKES
220 PRINT"(home 40shifted 2's)"
230 PRINT"(40shifted right brackets)"
240 :
250 REM BR$ IS THE BALLOON
260 BR$="(down) (up left) (up left)O(down left)!"
270 :
280 PRINT"(home)"
290 BC=23:REM LINE COUNTER TO BOTTOM SCREEN
300 TD=ABS(RND(1)*9)+1:TD=TD*10: REM CHOOSE TIME DELAY
310 SC=0:BD=0:REM ZERO SCORE AND DISAPPEAR FLAG
320 PD=ABS(RND(1)*13)+8:REM CHOOSE POSITION FOR DISAPPEARANCE
330 :
340 REM MAIN LOOP
350 :
360 PRINTLEFT$(DN$,BC)TAB(18)BR$: REM PRINT BALLOON
370 :
380 REM DECREASE COUNTER AND CHECK FOR BREAKAGE
390 BC=BC-1:IFBC<3THEN530
400 SC=SC+1:REM INCREASE SCORE
410 REM TIME FOR DISAPPEARANCE
420 IFBD=0ANDBC=PDTHENBD=1:PRINTTAB(18)"Q ":BR$=" "
430 :
440 REM MAKE NOISE AND WAIT
450 POKE59464,80
460 FORJ=1TOTD:NEXT
470 POKE59464,0
480 :
490 REM READY TO QUIT
500 GETSP$:IFSP$<>""THEN580
510 GOTO360
520 :
530 REM WAITED TOO LONG
540 PRINTTAB(18)"(up)*"
550 PRINTLEFT$(DN$,20)"YOU BROKE THE BALLOON
560 PRINT"YOU LOST 100 POINTS":TP=TP-100:GOTO620
570 :
```

```
580 REM STOPPED IN TIME
590 PRINTTAB(18)"(2up)O(down left)!"
600 PRINTLEFT$(DN$,20)"YOU STOPPED IN TIME"
610 PRINT"YOU GET"4*SC:TP=TP+4*SC
620 NT=NT+1:PRINT"AFTER"NT"TIMES YOUR AVERAGE IS"ABS(TP/NT)
630 :
640 REM CLEAR BUFFER
650 FORI=1TO10:GETSP$:NEXT
660 :
670 REM NEW ROUND
680 PRINT"GO AGAIN (Y/N)"
690 GETSP$:IFSP$=""THEN690
700 IFSP$="N"THENEND
710 GOTO170
```

We have already seen how to tab horizontally by simply using the TAB command. Since the PET has no vertical tab we must take steps to simulate one. Line 130 sets up a string variable to equal a home and 24 downs. If we were to print just the first seven characters of this string we would be on the sixth screen line. Twenty characters would leave us on the nineteenth line and so on. The command which allows us to take any number of the 25 characters in DN$ is LEFT$. LEFT$ starts at the left end of a string and takes as many characters as you tell it to. If A$="RALPH" the PRINT LEFT$(A$,3) would print RAL. Using this technique helps to set up our vertical tab.

Line 190 turns on the sound and sets a particular tone. The procedure for this can be found in several books. Line 260 sets up a ´balloon´ with a ´string´. Line 300 chooses a number between 10 and 90 which will be used in line 460 as a time delay. This controls how fast the balloon rises and the smaller the number the quicker the assent. In line 310 the ´disappearance flag´ is set to 0. A flag is a variable which usually has only a few values and indicates something has or has not yet occurred in a program. In this case when BD is 0 the balloon is still visible. When BD is set to 1 the balloon has disappeared. In line 320 the position for the disappearance of the balloon is set for between lines 8 and 21 on the screen. Line 360 uses a constant horizontal tab and a variable vertical tab to print the balloon as it rises. The way we defined the balloon in line 260 insures it will erase itself as it goes up. Line 390 decreases the balloon´s position by one line for each round and jumps to the ´waited too long´ section when the balloon meets the spikes. Line 420 says that if the balloon has not yet disappeared (BD=0) and it is at the line where it should disappear (BC=PD) then make it do so. We make the balloon disappear by printing spaces on top of it, redefining BR$ as blank spaces and setting BD to one. Line 500 checks to see if we have hit any key indicating we want to quit. If we have not we go back to line 360. If we have hit a key, we know we have not crashed and go to line 580 to make the balloon reappear, calculate and print the score. Line 620 keeps track of the number of times (NT) we have played and calculates an average score. We then get a chance to stop or play again. The purpose of line 650 is to get rid of any excess keys we may have hit in our effort to stop the balloon.

Many games we have seen consist of moving an object around the screen to avoid something chasing you and to capture something else. The next part of this series will show how this can be done with both PRINT statements and the POKE and PEEK commands. Please write if there are any questions about what has appeared here. Let me know if the articles move too fast or too slow and if they are unclear on any point. An indication of how useful the information is to you would also be appreciated.

THE MONITOR

by Jerry Key

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.

In the last issue I recommended the use of SYS54386 (SYS64785) to get to the monitor. I have had a question or two on this. To illustrate the difference between these and SYS1024 or SYS4, when using SYS54386 notice that the indicator is *C but when using SYS1024 or SYS4 it is *B. The difference is that one is a Call (*C) to the monitor and the others are Break(*B) to the monitor. Try each one repeatedly with the SYS and then eXit using X(rtn). As you continue to Break to the monitor you will notice that the number under SP in the last column displayed will decrement but with repeated use of the Call it will not. In most uses it will not make any difference but it could and is not good practice.

Anyhow, on to other things. This time, let's create a routine in machine language using the monitor and do different things with it. First, make sure that you power down. Then after power up enter the following BASIC program:

```
10 POKE59468,14:SYS 7658(rtn)
```

Then call the monitor with SYS54386. Your call should look like this:

```
C*
     PC  IRQ  SR AC XR YR SP
.;  B780 E455 34 33 38 36 FA
.
```

Then type M 1DEA 1E3B(rtn). It should look like this:

```
.M 1DEA 1E3B
.:  1DEA AA AA AA AA AA AA AA AA
.:  1DF2 AA AA AA AA AA AA AA AA
.:  1DFA AA AA AA AA AA AA AA AA
.:  1E02 AA AA AA AA AA AA AA AA
.:  1E0A AA AA AA AA AA AA AA AA
.:  1E12 AA AA AA AA AA AA AA AA
.:  1E1A AA AA AA AA AA AA AA AA
.:  1E22 AA AA AA AA AA AA AA AA
.:  1E2A AA AA AA AA AA AA AA AA
.:  1E32 AA AA AA AA AA AA AA AA
.:  1E3A AA AA AA AA AA AA AA AA
.
```

We now have memory locations 1DEA through 1E41 displayed and are ready to change them for our use. Now place the cursor on the first AA displayed after 1DEA and change the AAs to look like the display below. Be sure to type it exactly!

```
.  1DEA A9 93 A8 20 D2 FF A2 27(rtn)
.  1DF2 A9 66 9D 00 80 CA 30 03(rtn)
.  1DFA 4C F4 1D A2 27 A9 66 9D(rtn)
.  1E02 C0 83 CA 30 03 4C 01 1E(rtn)
.  1E0A A2 15 BD 25 1E 9D 79 83(rtn)
.  1E12 CA 30 03 4C 0C 1E A9 11(rtn)
.  1E1A A8 20 D2 FF 20 D2 FF 20(rtn)
.  1E22 D2 FF 60 A0 A0 A0 A0 CA(rtn)
```

. 1E2A C5 D2 D2 D9 A7 D3 A0 CD(rtn)
. 1E32 CF CE C9 D4 CF D2 A0 A0(rtn)
. 1E3A A0 AA AA AA AA AA AA AA(rtn)

Now we want to save this before we do anything else so type the following:

.S"0:MON ROUTINE",08,1DEA,1E3B(rtn)

Remember, if you are using tape, omit the 0: and change the 08 to 01. We are saving the code from the start (1DEA) to ONE LOCATION PAST the end of the code.

Now clear your machine and reload the program as you would a BASIC program. Then SYS54386(rtn). Then enter M 002A 002A(rtn). Your monitor should look like this:

.M 002A 002A
.: 002A 3B 1E 3B 1E 3B 1E 00 00

Now we can find the ending address of our routine by reversing the first two bytes after the location 002A. We then have 1E3B which was the ending location in our save. Now enter an X(rtn) to leave the monitor and get a READY. By converting the start location of our program to decimal, i.e., 1DEA=7658 we can call this routine with SYS7658. But before we call it, let's link it to the BASIC program that we entered. Type the following:

.S"0:BASIC MON",08,0400,1E3B(rtn)

This should save the entire BASIC program and the machine language code together so that we can load and save it like a BASIC program. Now exit the monitor with X(rtn) and type RUN in BASIC. If you entered it properly you should have a border at the top and bottom of the screen (40 column machines) and a highlighted name near the bottom. The key coding for the name is between locations 3E25 and 3E3A. Get out your Osborne manual and look up the codes in the table and make your own label!

The interesting part of this is not what the routine does but the method we used to attach a machine language routine to a BASIC program. Remember, enter your BASIC code first. You can always check to see where the end of BASIC is at locations 002A and 002B to make sure that it won't interfere with the machine language. The machine language will load in over the top of BASIC and then everything can be saved together from the monitor. Once saved the entire program can be loaded and saved like a BASIC program. BE CAREFUL when using the TOOLKIT or monitor extensions with programs that have machine language attached or it may be detached from the BASIC. This can be a useful tool in some cases but you have to know when it's coming. Also, the rules can be bent a bit when putting ML in the cassette buffers but more on that later.

****

The INDEX

Have you ever searched for a particular article that you just know appeared in some recent magazine or newsletter? I have done this many times and spent hours becoming more and more frustrated. The INDEX covers over 12,000 articles from over 900 issues of computer magazines written in the last six years. It arranges by subject over 30,000 entries by key words in the article title. There are separate indices for Apple, Atari, Commodore and over 10 others plus a general index. The publisher has promised to update the current volume and to add new articles and magazines. For information contact Missouri Indexing, PO Box 301, St. Ann, MO 63074

Histogram

by Doug Haluza

A simple bar graph can often show someone a great deal. A histogram is just a bar graph that shows the total number of data items (frequency) in a number of particular intervals. Histograms show the distribution of a set of data items. The data items might be grades on an exam, measurements of a manufactured item, heights of children of a particular age, or any other data item that has some sort of measurement where the number of items in each interval of measurement is important. A histogram of exam grades would show the distribution of grades and would show how good any student's grade is compared with the rest of the class. It also shows how good the exam is by showing how well the distribution matches the ideal 'normal' (bell-shaped) distribution. A histogram of manufacturing measurements would show how well the product meets the necessary tolerances.

Other statistical information is also helpful and is also computed by this program. The **median** is the value of the middle data item. Fifty percent of the data items are greater than this median item, and 50% of the data are less than it. The **quartile** points are the medians of the upper and lower halves of the data. Twenty-five percent of the data are greater than the upper quartile point, and 75% of the data are below it (and vice versa for the lower quartile point). The **mean** is simply the average of the data items. The **standard deviation** tells how closely the data items are packed together. It is the average difference of all the data items. The **mode** is just the data item or items with the greatest frequency (the item that appears the most).

A histogram can show quite a bit that wouldn't be obvious from a list of the data. A programming example can also show quite a bit that wouldn't be obvious from a list of programming techniques. This program shows many programming techniques that can really improve your programs. These techniques are really what this article is about. Each technique can only be described briefly, for more information see **The PET/CBM Users Guide** from Osborne/McGraw Hill Books.

The first technique you'll notice is using the INPUT statment with a default input (i.e. lines 130, 140, 640). Printing a default answer for an INPUT and then moving the cursor back behind the default answer will leave the default answer after the question mark. The user then can just hit return to enter the default answer. To enter something else the user can type another answer over the default given. To give a default input use the INPUT statment with a prompt (a message telling the user what the computer wants) in quotes. At the end of the prompt type two spaces, the default input, and a cursor left for each character in the default. Add two extra backspaces and then close the quotes.

Using a default input not only is convenient for the user, it shows what type of input is expected and gives an appropriate input value. This method also prevents the user from dropping out of a program by just hitting return.

When a default can't be given (e.g. lines 190, 370, 380) a cursor character can be given instead. The shifted question mark works well here, but any other character can be used. You might want to use '$' for string input or '#' for numeric input, for example. If you prefer the PET's own cursor use shift space for the cursor character. Shift space prints as a space on the screen, but has an ASCII value of 96 instead of 32, so it is actually a different character. When using this cursor character if the user accidentally hits only return the cursor character will be in the input variable, and the user won't be staring at 'READY.' wondering what to do next.

Bulletproofing your program continues with checking the input values after they've been safely input (e.g. lines 130, 140, 220, 340-360). If the input value isn't one the program expects anything could happen, and if Murphy has his way it will be anything but what you wanted to happen.

Remember that the PET responds with '?REDO FROM START' if it is expecting a numeric input and doesn't get it. This only confuses the naive, and annoys everyone else. Whenever possible take the input into a string variable and

convert it to a number with VAL (e.g. line 190). VAL(S$) gives a number if S$ is a number in string form, otherwise it gives zero; so VAL("1234")=1234, but VAL("HI MOM")=0. Remember to check if S$="0" when zero is a valid input.

Another important part of people-proofing your program is to forgive mistakes and allow the user to correct them. The more things someone has to type the more likely that person is to make a mistake, and the more annoyed they'll be when they find out they have to type it all over again. This program only allows you to correct the last item entered. To do this is simple, to do more would require an editing subroutine. Usually being able to correct the last entry is all that is needed because you're most likely to realize you've made a mistake just after you hit return for the same reason you remember what you had to tell someone just as you hang up the phone.

Naturally it's impossible to completely bulletproof a program, because no matter how hard you make it for someone to make a mistake it is bound to happen anyway. The harder you make it to make mistakes, the better, but don't make the program harder to use. Remember to keep the program user friendly.

Another useful trick is using AND and OR with IF statments (e.g. lines 230, 450-470). In order for an AND function to be true both of the conditions on either side of it must be true. With the OR function if either condition is true the result is true. For example:

```
2+2=4 OR 1<2   is true      2+2=4 AND 1<2   is true
2+2=4 OR 1>2   is true      2+2=4 AND 1>2   is false
2+2=5 OR 1<2   is true      2+2=5 AND 1<2   is false
2+2=5 OR 1>2   is false     2+2=5 AND 1>2   is false
```

By combining these statements one IF statement can do the work of several IFs and several more GOTOs.

It's also a good idea not to raise a variable to a power with the uparrow when you can multiply it by itself (e.g. line 250). Use A*A or A*A*A to square or cube A. The reason for this is that the power function uses logs to compute the result. Taking the log of a number, multiplying it by a power, and taking the antilog wastes time and adds error to the answer (try squaring seven and printing the result).

When output might go to either the screen or the printer you can open a file to send output to either device. The screen is device three, and the printer is usually device four. OPEN 1,3 will direct all output sent to logical file 1 to physical device 3; so in this case PRINT#1 works just like PRINT. If the file was opened with OPEN 1,4 output would be sent to the printer. Output could be sent to any logical device (e.g. Tape #1, CBM disk, 8010 Modem, etc).

Output cannot be sent to the keyboard, of course, but the keyboard can be opened as a logical file for input. OPEN 2,0 will link logical file 2 to physical device 0 (the keyboard). INPUT#2,A$ will take input from the keyboard. With this method of input the user cannot drop out of the program because the PET has been fooled into thinking it really isn't taking input from the keyboard, so it won't take a null input for an answer. It also suppresses the '?' prompt; this can be useful at times

The "control G's" in the PRINT statements in lines 220 and 310 are the bell character that trips the new Fat 40 and 8000 PET's internal bleeper. On other PETs it doesn't do anything. You can use them to attract the user's attention to an important message. To insert them type the opening quote and delete it, then type another quote. This gets you out of quote mode. Now hit RVS, G, OFF; then continue typing the print line. This trick works for other ASCII control characters.

A useful trick lies in line 460. This trick allows the PET to print a number, whose length can vary, in a field of a certain length (similar to PRINT USING). STR$ is the opposite of VAL. STR$ changes a number into a string. RIGHT$(S$,N) prints the rightmost N characters of S$. By adding leading blanks to the number

and taking the rightmost four characters line 460 prints I in four characters no matter what the value of I is. The number will be right justified; to left justify a number use LEFT$(STR$(I)+" <N spaces> ",N).

I haven't mentioned structured programming, and I havn't used it in this example because there's no reason to make a simple program more complicated. Structured programming, that is writing your program in small easy to manage blocks and subroutines, can make complicated programs easier to write and debug. Breaking a large program down into small structures reduces the apparent complexity of a program, so it appears simpler.

Simplicity is always the key. In programming what you say is just as important as how you say it. Many programmers try to discount this. They argue that all that's important is that the program works properly. These programmers move blindly ahead and try to resolve the problem forcefully. When problems arise they try to resolve the problem by throwing more code at it. Although the program will work, no one with any self respect would put their name on it. Programs written this way are long, usually as long as they can be (garbage expands to fit the space allotted) and because they're long they're slow. Some careful thought could have reduced the problem.

```
100 REM HISTOGRAM-2 BY DOUG HALUZA
110 REM LATEST REVISION 10-NOV-81
120 REM PERMISSION TO USE, NOT TO SELL
130 INPUT"LOWER BOUND  0(31eft)";LB:IF LB<0 THEN 130
140 INPUT"UPPER BOUND  100(51eft)";UB:IF UB<=LB THEN 140
150 DIM A(UB):SI=UB:GI=LB:N=0:X=0:X2=0:REM *** INITIALIZE VARIABLES
160 PRINT"ENTER DATA ITEMS ONE AT A TIME
170 PRINT"USE -1 TO CORRECT LAST ITEM ENTERED
180 PRINT"USE -99 TO END DATA ENTRY":PRINT
190 PRINT N+1;:INPUT"   (31eft)";A$:A=VAL(A$):REM *** GET DATA ITEM
200 IF A=-1 THEN 290 *** CORRECT LAST ITEM
210 IF A=-99 THEN 320 *** END DATA ENTRY
220 IFA<LBORA>UBOR(A=0ANDA$<>"0")THENPRINT"(control G)BAD DATA ITEM--REENTER";
:GOTO190
230 A(A)=A(A)+1:REM *** HISTOGRAM
240 X=X+A:X2=X2+A*A:N=N+1:REM *** SIGMA-X, SIGMA-X{,NUMBER OF ITEMS
250 LI=A:LG=GI:LS=SI:REM *** LAST DATA ITEM AND PREVIOUS BOUNDS
260 IF A<SI THEN SI=A :REM *** SMALLEST DATA ITEM
270 IF A>GI THEN GI=A:REM *** GREATEST DATA ITEM
280 GOTO 190
290 REM *** CORRECT LAST DATA ITEM ENTERED ***
300 A(LI)=A(LI)-1
310 PRINT"(control G)REENTER ITEM";:GOTO190
320 REM *** COMPUTE RESULTS AND PRINT HISTOGRAM ***
330 INPUT"(clr)OUTPUT TO (rvs)P(off)RINTER OR (rvs)S(off)CREEN   (31eft)";O$
340 IF O$="P" THEN OPEN 1,4:GOTO 370 *** DEVICE 4 IS THE PRINTER
350 IF O$="S" THEN OPEN 1,3:GOTO390 ***DEVICE 3 IS THE VDU SCREEN
360 PRINT"TYPE 'P' OR 'S'":GOTO330
370 INPUT"TITLE LINE 1   (31eft)";T1$:REM *** 3 SHIFT SPACES, 3 CURSOR LEFTS
380 INPUT"TITLE LINE 2   (31eft)";T2$:REM *** 3 SHIFT SPACES, 3 CURSOR LEFTS
390 IF O$="P" THEN PRINT#1,CHR$(1)"  "T1$:PRINT#1,CHR$(1)"  "T2$
400 Y=0:MO=0:ME=0:LQ=0:UQ=0:REM COUNTER, MODE,MEDIAN, QUARTILE POINTS
410 FOR I=SI TO GI:M=M+A(I):Y=Y+A(I)
420 IF (M>=N/2) AND (ME=0) THEN ME=I:REM *** FOUND MEDIAN
430 IF (M>=N/4) AND (LQ=0) THEN LQ=I:REM *** FOUND LOWER QUARTILE
440 IF (M>=3*N/4) AND (UQ=0) THEN UQ=I:REM *** FOUND UPPER QUARTILE
450 IF A(I)>MO THEN MO=A(I):REM *** FOUND NEW MODE
460 PRINT#1,RIGHT$("    "+STR$(I),4)":";
470 IF A(I)>0 THEN FOR J=1 TO A(I):PRINT#1,"*";:NEXTJ
```

```
480 PRINT#1:NEXTI
490 IF O$="P"THEN 520 *** PRINT STATISTICS
500 PRINT"TYPE ANY KEY TO SEE STATISTICS";
510 GETA$:IFA$=""THEN510
520 PRINT#1
530 PRINT#1,"    UPPER 25%:"UQ
540 PRINT#1,"    MEDIAN   :"ME
550 PRINT#1,"    LOWER 25%:"LQ
560 PRINT#1,"    MEAN     :"INT((X/N)*100+.5)/100
570 PRINT#1,"    STANDARD DEVIATION:"INT(SQR((N*X2-X*X)/(N*(N-1)))*100)/100
580 PRINT#1,"    MODE(S) AT:";
590 FORI=SITOGI:IF A(I)=MO THEN PRINT#1,I;
600 NEXTI:PRINT#1
610 PRINT#1,"    NUMBER OF DATA ITEMS:"N
620 PRINT#1,"    GREATEST DATA ITEM  :"GI
630 PRINT#1,"    SMALLEST DATA ITEM  :"SI:PRINT#1
640 INPUT"ANOTHER PRINTOUT  N(31eft)";A$
650 IF A$<>"N" THEN PRINT"(clr)":GOTO390
660 CLOSE1:END
```

****

RND Function

by Mike Todd

The randomness of the PET's random number generator has long been a point of discussion among PET owners. I hope that this explanation will be the final word. My solution makes the clumsy PEEK-POKE method obsolete.

Even after all the publicity given the random number function few people seem to know how to use it to its best advantage. The RND function on the PET makes use of a simple algorithm to generate a psuedo-random number. It does this by taking a number --the seed-- and multiplying it by 0.0000000392767778, adding 11879546.4, swapping the bytes of the result, forcing the results into the range 0-1 and then putting the results in memory. Thus, if the seed remains the same, the random number will always be the same. If one uses RND(1), however, the seed is always the last random number generated -- the initial seed being set on switch-on.

Most game programs really need something a little more random than this, so the RND routine allows RND(0) to set a truly random number on the new ROM PETs. On the old PETs the seed is set from non-existant RAM locations which return a noise based number, while on the new ROMs the 1MHz timers are used and produce a number which is unlikely to be repeated once in four million times. Unfortunately, continuous calling of RND(0) is not recommended since the timers change only a small amount and the numbers returned, therefore, are very similar. However, if you use A=RND(0) at the beginning of your program, before you start using RND(1) throughout, the seed will be set randomly, and the consequent series produced will be as good a random sequence as you could want. RND(1) should always be used in the main program.

In addition, using A=RND(-1) at the start should allow easy debugging since this makes the seed .1, and the same series of numbers will be called every time the program is run. When debugged, just change the -1 to 0 or even -TI which has a similar effect. The -TI method is suggested for old ROM users but both methods generate a truly random series. So A=RND(0) or A=RND(-TI) is equivalent to RANDOMIZE and can be used throughout the program to give a further 'shake up' to the random series.

Questions and comments can be directed to me at Welwyn Garden City, Hertfordshire, England.

Relative Files: Part I

by Glenn Davidson

One of the reasons people buy disk drives is to employ them in an application which involves the storage and retrieval of large amounts of information. The storage must be reliable and the access must be fast and versatile. This information is stored on the disk in the form of a data file which is usually accessed by a program called a data base manager. Before we go on to discuss anything more about data files and how to develop a mailing list program or data base manager let's define some terms.

data base - A large set of pieces of related information. This might be a list of clients or an inventory.
record - One set of related facts from a data base such as the name and address of one person.
field - One piece of data from a record.
key - A numeric variable from 1 to 63999 that identifies an individual record.
DS - A reserved variable in PET BASIC 4.0 that reflects the disk status.
DOS - disk operating system

One of the biggest advantages of the new Commodore disk drives is their ability to do a type of file called a relative file. A programmer or program user is no longer tied to sequential files or the unreliable random files available on the older 2040 drive. Sequential files are still available but no will use them after they find out how easy and versatile relative files are. With relative files a programmer can immediately locate any record relative to the beginning of the file just like with random files. He can also read or write the records one at a time as if it was a sequential file. The greatest advantage is the time saved using relative files. By using a "key" to the file, a programmer can access any record in less than 2 seconds. With sequential files one must read all the records from the beginning of the file to the record wanted. When writing to a file an individual record may be written without scratching and rewriting the entire file as must be done with sequential files.

To use relative files is fairly simple and it may prove simplest to think of a relative file as a long line of boxes. All you would have to do is define the size of each box. Suppose you wanted to make a mailing list program which would be set up in the following way:

Field Name	Variable	Length
Mailing #	MN$	5
Name	NA$	20
Street address	AD$	25
City	CY$	20
State	SA$	2
Zip code	ZC$	10

This means that each record will have to be at least 82 characters so let's use 85. To set up such a file for the first time you must give a DOPEN statement such as: 10 DOPEN #1,"MAILING",D0,L85. This means that we will OPEN the file named MAILING as file #1 on disk drive 0. In addition, it says that each record in that file may have a maximum of 85 characters. This will create the file and need only be done the first time we create the file. Each initially empty record or box is filled with the pi character whose code is 255. You are now ready to write information to the newly created file. This is where the mailing number is important since it is used to position the record relative to the beginning of the file. Suppose we wanted to write information into record space number five.

A simple example might look like this:

```
10 DOPEN #1,"MAILING",DO,L85
20 CR$ = CHR$(13)
30 MN$ = "00005"
40 NA$ = "DOUG HALUZA"
50 AD$ = "KELLY B 302"
60 CY$ = "STONY BROOK"
70 SA$ = "NY"
80 ZC$ = "11794"
90 RN = VAL(MN$)
100 IF RN <= 0 THEN PRINT "RECORD # MUST BE > 0": GOTO 130
110 RECORD#1, (RN)
120 PRINT#1, MN$CR$NA$CR$AD$CR$CY$CR$SA$CR$ZC$
130 CLOSE1
```

Here we have opened the file correctly and then assigned values to each of the variables we mentioned. Carriage returns are often needed in writing files so we set a variable (CR$) equal to one. In line 90 our record number (RN) is set to the value of our mailing number. It is a good idea to check to see that the record number is greater than 0 since one that is less than or equal to zero will cause information to be written to the first record only. Notice that in line 110 we have the variable for record number (RN) enclosed in parentheses which is the only way it will work. Finally, we print our information to the file separating the different parts by carriage returns. This means that each piece of information will end up as a field in our record. Don't forget to close the file as is done in line 130.

After reading or writing to a relative file the record pointer is positioned to the next record. In our example this would be record six. This means that if we want to simply write or read a relative file sequentially we can do it easily. Here is a short program to write a relative file sequentially. This might be done when first setting up a mailing list or when adding a large group of names at one time.

```
10 DOPEN #1,"MAILING LIST",D1,L85
20 CR$ = CHR$(13)
30 INPUT "NAME";NA$
40 IF NA$="END" THEN CLOSE 1: END
50 INPUT "STREET ADDRESS";AD$
60 INPUT "CITY";CY$
70 INPUT "STATE";SA$
80 INPUT "ZIP CODE";ZC$
90 RN = RN+1: MN$=STR$(RN)
100 RECORD#1, (RN)
110 PRINT#1, MN$CR$NA$CR$AD$CR$CY$CR$SA$CR$ZC$
120 GOTO 30
```

This program will create and print to the file MAILING LIST starting at record number one. It will do so until you enter a name of END at which time it will close the file and end. In this example, the mailing number is automatically calculated and is simply the record number. This mailing number is printed to the file and can be used as a key. The key is simply how many records from the beginning of the file a particular record is and can be used to directly access this information later.

Instead of writing records one after the other you may want to skip around. You may have mailing numbers already assigned or you may want to change all or some of the fields in a record. Here is a program that allows you to enter a mailing number and then warns you if something exists for that record.

```
10 DOPEN #1,"MAILING LIST",D1
20 CR$ = CHR$(13)
30 INPUT "MAILING NUMBER";MN$
40 RN=VAL(MN$)
50 IF RN<1 THEN PRINT "MAILING # MUST BE >1 ":GOTO 30
60 RECORD#1, (RN)
70 INPUT#1, MN$,NA$,AD$,CY$,SA$,ZC$
80 IF MN$=CHR$(255) OR MN$="" THEN 150
90 IF DS=50 THEN 150
110 PRINT "THAT RECORD ALREADY EXISTS."
120 PRINT "HIT C TO CONTINUE OR A FOR ANOTHER RECORD"
130 GET SP$: IF SP$=""THEN 130
140 IF SP$="A" THEN 30
150 INPUT "NAME";NA$
160 IF NA$="END" THEN CLOSE 1: END
170 INPUT "STREET ADDRESS";AD$
180 INPUT "CITY";CY$
190 INPUT "STATE";SA$
200 INPUT "ZIP CODE";ZC$
210 RECORD#1, (RN)
220 PRINT#1, MN$CR$NA$CR$AD$CR$CY$CR$SA$CR$ZC$
230 GOTO 30
```

Here, starting in line 30, we ask the user for the mailing number, set the record number and check to see if it is 1 or greater. If it is not we ask for another number. If it is 1 or greater, we set the record pointer and then read the information using INPUT#. In lines 80 and 90 we use three ways to check to see if the record is empty. If it is not we give the user a warning and a chance to choose another mailing number. If the user chooses to go on then we ask for the information for the record, then set the record pointer and write the information to the file using PRINT#. The user ends the program by inputting END for a name.

We've seen how to write a relative file and a little about how to read one. As I already mentioned relative files are better than random or sequential files in the way you can retrieve data from them. With relative files you can use methods called direct or sequential keying. This means you can quickly access any record as long as you know the key for it. It is, therefore, a good idea to devise some consistent way of numbering the data that you are going to enter. Some applications suggest numbers themselves like the numbers on invoices or checks. For other applications like a mailing list you might have to invent numbers. The key values you should use should be small or should be able to be reduced to a small number in some consistent way. For example, your first person on a mailing list might be 40001 while the second is 40002 and so on. These are not small records but by subtracting 40000 from each they can be made smaller. It would be silly to use 40001 as a record number since the computer would set up 40000 blank records before getting to 40001.

To read records from our file in some random order that the user selects we will use a program much like the last one we wrote. Let's say that the mailing numbers we used start at 40001.

```
10 DOPEN #1,"MAILING LIST",D1
20 INPUT "MAILING NUMBER";MN$
30 IF MN$="END" THEN CLOSE 1: END
40 RN=VAL(MN$)-40000
50 IF RN<1 THEN PRINT "MAILING # MUST BE >1 ":GOTO 30
60 RECORD#1, (RN)
70 INPUT#1, MN$,NA$,AD$,CY$,SA$,ZC$
```

```
80 IF MN$=CHR$(255) THEN 110
90 IF MN$="" THEN 110
100 IF DS=50 THEN 110
105 GOTO150
110 PRINT "THAT RECORD IS EMPTY."
140 GOTO 20
150 PRINT "MAILING NUMBER: "MN$
160 PRINT "NAME: "NA$
170 PRINT "STREET ADDRESS: "AD$
180 PRINT "CITY: "CY$
190 PRINT "STATE: "SA$
200 PRINT "ZIP CODE: "ZC$
210 GOTO 20
```

To simply read a relative file sequentially from some point to the end of the file we could modify things slighty.

```
10 DOPEN #1,"MAILING LIST",D1
20 INPUT "STARTING MAILING NUMBER";MN$
30 IF MN$="END" THEN CLOSE 1: END
40 RN=VAL(MN$)-40000
50 IF RN<1 THEN RN=1
60 RECORD#1, (RN)
70 INPUT#1,MN$,NA$,AD$,CY$,SA$,ZC$
80 IF DS=50 THEN 160
90 PRINT "MAILING NUMBER: "MN$
100 PRINT "NAME: "NA$
110 PRINT "STREET ADDRESS: "AD$
120 PRINT "TOWN: "CY$
130 PRINT "STATE: "SA$
140 PRINT "ZIP CODE: "ZC$
150 RN=RN+1: GOTO 60
```

There are some problems here when it comes to searching for a record if we do not know the "key", in this case, the mailing number. If we have to search from the beginning of the file then we are doing no better than with sequential files. In the second part of this series I will explain how to set up a "key file" for your data base and how to use more sophisticated methods of manipulating relative files. Our goal is to work toward a SMALL mailing list program to write, read and change records.

When working with relative files keep in mind the following:

1) Relative files are always open to both read and write. Sequential files are only open to do one operation as designated by the OPEN statement.
2) DS will always be equal to 50 when you get to the end of the file. This is the most reliable way to check for that condition.
3) If the first variable read from a record is a CHR$(255) or a blank then the record is blank.
4) The PRINT# and INPUT# statements both move the record pointer one record forward when they are executed.
5) Set up a consistent "key" ahead of time and always access records for writing and reading by it.
6) The record's key should be its position relative to the beginning of the file.

Universal Keyprint

by Ralph Bressler

I seldom use the KEYPRINT program but it seems that I am in the minority. Many people have asked about it and how to get it to work on their system. Since KEYPRINT is too long to fit in the space now available in the second cassette buffer and since this buffer is used by other routines, the only solution is to put it into high memory. I am not clever enough to do this all in machine language but a close look at the program reveals that it is almost completely relocatable. The BASIC program below lowers the top of memory by 256 bytes and then POKEs the ML program into memory. This means it can run on machines with any memory configuration and can even coexist with other program which use the top of memory. After POKing in the program, my BASIC program adjusts the parts of the program which are not completely relocatable. It also corrects for BASIC 2.0 or 4.0 and for 40 or 80 column screens. I have tried this on machines with many different combinations of BASICs and memory. To date I have not used it on the 8032 but it seems to dump correctly when told that a 40 column machine has 80 columns. I hope this temporarily solves all the problems.

```
100 PRINT"(clr)UNIVERSAL KEYPRINT(down)"
110 INPUT"(rvs)4(off)0 OR (rvs)8(off)0 COLUMNS";NC
130 REM THIS SECTION SETS UP THE KEY USED TO DUMP THE SCREEN
140 PRINT"(clr)HOLD DOWN THE KEY YOU WANT TO USE
150 PRINT"UNTIL THE SCREEN IS ABOUT HALF FULL
160 PRINT"THEN RELEASE THE KEY.(down)"
170 GETSP$:IFSP$=""THEN170
180 IFPEEK(151)=255THEN180
190 K=PEEK(151):PRINTK;:IFK<SGN255THENR=K:GOTO190
210 REM DROPS TOP OF MEMORY 256 BYTES FOR PROGRAM STORAGE
220 L=(PEEK(53)-1):POKE53,L
230 REM FIGURES BEGINNING ADDRESS OF PROGRAM
240 LL=L*256+PEEK(52)
250 REM FIGURES BEGINNING OF INTERUPT PROGRAM
260 M=PEEK(52)+25:IFMSGN255THENF=1
270 REM READS AND POKES ML PROGRAM INTO MEMORY
280 FORI=LLTOLL+146:READN:POKEI,N:NEXT
290 REM ADJUSTS CALL TO BEGINNING OF INTERUPT PROGRAM
300 POKELL+21,PEEK(53)+F:POKELL+20,M
310 REM PRINTS CORRECT NUMBER TO ACTIVATE PROGRAM
320 PRINT"(clr)USE (rvs)SYS"PEEK(53)*256+PEEK(52)"(off)TO ACTIVATE."
330 REM POKES IN THE VALUE OF THE KEY TO BE USED
340 POKELL+16,R
350 REM CHECKS FOR VERSION OF BASIC
360 IFPEEK(50003)<SGN1THEN390
370 REM CHANGES CERTAIN SUBROUTINE CALLS FOR BASIC 2.0
380 POKELL+23,46:POKELL+24,230:POKELL+40,186:POKELL+43,45
390 REM ADJUSTS FOR 80 COLUMN SCREEN
400 IFNC=8THENPOKELL+122,80:POKELL+128,79
410 DATA120,165,53,133,145,165,52,105,13,133,144,88,96
420 DATA165,151,201,69,208,3,32,25
430 DATA127,76,85,228,169,128,133,32,169,0,133,31,169,4,133,176,133,212,32
440 DATA213,240,32,72,241,169,25,133,34,169,13,133,33,32,210,255,169,17,174
450 DATA76,232,224,12,208,2,169,145,32,210,255,160,0,177,31,41,127,170,177
460 DATA31,69,33,16,11,177,31,133,33,41,128,73,146,32,210,255,138,201,32
470 DATA176,4,9,64,208,14,201,64,144,10,201,96,176,4,9,128,208,2,73
480 DATA192,32,210,255,200,192,40,144,203,165,31,105,39,133,31,144,2,230,32
490 DATA198,34,208,166,169,13,32,210,255,76,204,255
```

PRICES

American Peripherals

122 Bangor Street
Lindenhurst, NY 11757

516-226-5849

COMMODORE PET MICROCOMPUTERS

FREE SOFTWARE PACKAGE

Model 4016

Standard Student PET
40 Columns Across, 16 K Memory
Graphics Keyboard

$995.00
Buy 2 get 3rd free
until SEPT 82

FREE SOFTWARE PACKAGE

Model SP9000

Super Pet
Multi-Language
Ability
80 Columns
Needs 8050
disk drive

$1,995.00

Model 8032

Data Processing
80 Columns
32 K Memory
Standard Keyboard
Great for
Word Processing
and Data Entry

$1,495.00
Buy 2 get 3rd free
until SEPT 82

FREE SOFTWARE PACKAGE

Model 4032

Student Model with Full
32 K Memory. 40 Columns Across.
Graphics Keyboard. Good for Both
Programming and Word Processing.

$1,295.00
Buy 2 get 3rd free
until SEPT 82

TAPE RECORDER C2N

(used with 4016, 4032 and 8032 units) $75.00

DISK DRIVES

Model 8050 Dual Floppy

1,000,000 Characters
(1 megabyte)

$1,795.00

Model 4040 Dual Floppy

340,000 Characters

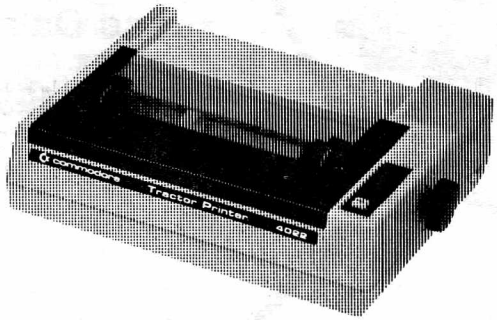
$1,295.00

Model 2031 Single Floppy

170,000 Characters

$695.00

PRINTERS



Model 4022 Printer

Dot - Matrix 80 Column
with Pet Graphics, Tractors

$795.00

Model 8023P Printer

Dot - Matrix 136
Column (wide paper)
Tractor Feed. Pet Graphics

150 cps. $995.00

Model 8300P

Letter Quality Daisy
Wheel Printer 40 cps.

$2,250.00

Cables:

Computer to Disk (P-I)	$35.95
Computer to Printer (P-I)	$35.95
Printer to Disk (1 meter) (I-I)	$44.95
Printer to Disk (2 meter) (2I-I)	$58.50

AMERICAN PERIPHERALS

112 BANGOR STREET • LINDENHURST, N.Y. 11757 • 516-226-5849

COMPUTER SCIENCE SERIES

PROGRAMS ABOUT PROGRAMMING

This series of 27 lessons on all aspects of computer science was developed by high school teachers with long experience on the PET teaching computer science.

They are available as individual programs on either tape or disk at $24.95 . The entire series can be purchased for $600 on individual tapes, or $350 on disk. (If you have 4000 series and 4040 disk, a $400 version has automatic record-keeping capability that tells who used the programs, when, and what scores were achieved.)

If you are a beginner, and want an easy-to-follow set of 3 programs with a 50 page workbook, get:

STEP-BY-STEP WITH THE PET $40.00

1 PRINT COMMANDS - explains how to use the print command, comas, semi-colon, tabs, strings, and other ways to print.

2 FOR-NEXT LOOPS - explains what they do, the loop counter, step, limits, applications.

3 GRAPHICS ON THE PET - many people buy PETs because of the incredible graphics capabilities. This lesson shows and explains many of the tricks of good graphics.

4 IF-THEN - This is the decision step in BASIC. Many examples are given, along with the pitfalls in its use.

5 STRING MANIPULATION - It takes a bit of study to understand how to change letters to numbers, strip off individual letters, concatenate, etc.

6 TYPES OF VARIABLES - Interger, real, floating point, binary, string-- they all are explained and illustrated.

7 ADVANCED COMMAND IN BASIC - Do you know how to use SYS, ABS, LEN, WAIT, and half a dozen others?

8 PARTS OF A MICROCOMPUTER - All computers are built from certain basic components, both electrical and logical.

9 POKES - Fundamental to PET programming is the ability to change memory directly by POKES. Many examples and explanations are given.

10 SOUND ON THE PET - How do you hook up sound and make the PET play musical notes and sound effects?

11 SUBSCRIPTED VARIABLES- Until you master x(j) and how one symbol stands for a whole class of numbers, your computing ability is limited.

12 SORTING - One of the most confusing operations to a beginner is sorting. This lesson leads you through the maze.

13 BOOLEAN OPERATORS - AND or OR are much deeper than they appear at first use. These are heavy topics in electrical engineering.

14 BINARY SYSTEM - Introduces student to counting and representation of numbers in binary, and why it is important in computers.

15 OCTAL AND HEXADECIMAL NUMBER SYSTEMS - Hex is the important one for the PET, but octal is much easier to explain. They are shown in comparison to one another.

16 STORAGE MEDIA - What are the differences between mag tape, cassette tape, hard disk, floppy disks, and bubble memory? What are the advantages and disadvantages of each?

17 DATA FILES - Whether you have 8, 16, or 32,K, you have limited storage. By using data files, your storage is unlimited, but it is not easy to master the details.

18 INTRODUCTION TO MACHINE LANGUAGE AND ASSEMBLY LANGUAGE(part I)- Tells about hex, binary, accumulators, program counters and machine language.

19 INTRODUCTION TO ASSEMBLY LANGUAGE AND MACHINE LANGUAGE(Part II)- Many illustrative examples showing simple routines and demos for machine language.

20 COMMON INTERFACES - When you try to hook a PET (such as a different brand printer) a miriad of problems ensue. These are solved by interfaces.

21 CONNECTING THE COMPUTER TO THE REAL WORLD - The explanation begins in a general way, but goes on to many specifics of what signals come out on what pins.

22 UNDERSTANDING TIMESHARING SYSTEMS - Many schools have access to timesharing terminals on a large computer. This program shows some features not experienced with the microcomputers.

23 HISTORY OF COMPUTING - Traces history from earliest forms of calculators to modern electronic computers.

24 INSIDE THE PET - Imagine that you removed the screws and were looking inside the PET at the board. This describes and developes the role of each component. ROM, RAM, VIA, PIA 6502 and so on.

25 DEBUGGING TECHNIQUES - Many kids are frustrated by programs that OUGHT to work but don't. There are established procedures for dealing with this common problem.

26 FOUR MAIN PARTS OF A COMPUTER - from the biggest IBM to the tiniest one chip computer, you always have IQ-memory-arithmetic-control.

27 DESCRIPTION OF FORTRAN - Once the student has mastered BASIC, it is easy to show him the similarities and differences of FORTRAN.

28 CHARACTER SETS- Each key on the PET represents 3 characters. In addition, the characters can mean control functions, graphics. We explore how characters are stored as numbers, and how they are utilized.

29 RANDOM NUMBERS-It is impossible to create interesting games and simulations without a basic mastery of random numbers.

Machine Language for Near Beginners: Part IV

by Ralph Bressler

I recently had the pleasure to attend a two day workshop in machine language presented by Jim Butterfield. I had seen Jim once before and was immediately as impressed by him in person as I had been with him in print. For a person with so much knowledge, Jim is surprisingly patient with beginners and is able to communicate effectively. I knew most of the factual material that he presented but somehow hearing someone present it began to tie things together for me. Jim's manner of presentation also pointed out some interesting things to me. If you get the chance to hear Jim Butterfield speak on any subject I suggest you make sure you don't miss it. Don't be afraid to ask questions in a group or after a talk. Jim won't put you down and he is a storehouse of information.

As I have stated before, this series is nearing its end. In fact, this is either the last or perhaps next to last in the series. From now on, Doug and I will present what we hope are clearly explained and commented routines and programs. I do have a few things to take up next time but these might be considered selected topics.

It has occured to me that I have made at least one huge mistake in doing these articles. I should have insisted that everyone who is really interested obtain a copy of one of the monitor extensions now available. These extensions include Supermon, Extramon and Micromon. Basically, they are listed in order or power from least to most powerful but also in order of ease of use from easiest to hardest. Supermon when loaded and run will automatically tuck itself away at the top of YOUR memory no matter what size machine you have. These extensions all have a mini-assembler which allows you to type in LDA #$A0 rather than A9 A0 and takes care of other pesky problems like calculating branch offsets. A disassembler command allows you to 'take apart' machine code and see the equivalent assembly mnemonics. Most of these extensions allow you to step through a program a command at a time which can be useful in debugging programs. They also allow you to move code from one place to another with a transfer. I suggest you pick up one of these immediately. Supermon and Micromon have appeared in COMPUTE in the last few months and various user groups have copies for almost every BASIC version. If you cannot find a copy anywhere else, I will make a copy of Supermon for you if you supply a tape, a self-addressed stamped envelope, and $2 to cover copying. Be sure to indicate which BASIC you have!

Monitor extensions are a big help but a full assembler is also nice to have. Briefly, an assembler will let you type in and save your source code so that you can easily make changes. It allows you to use labels for addresses and generally makes life easier. Many assembler listings have been published in The PAPER and the listings for this articles are from an assembler. I can think of more than a dozen good assemblers that are being sold for prices ranging from $40 to $190. If you think you want to buy a particular assembler I would be glad to comment on it if you ask.

In this installment we will look at a new addressing mode, the use of subroutines, and study how logical commands work. The new addressing mode is called post-indexed, indirect addressing. The name sounds impossible but in practice it is powerful and quite easy to use. Remember that the PET's memory is divided into 'pages' which each contain 256 bytes. Page 0 is very special and runs from location $0000 to $00FF. Most of this page is used by the PET for the storage of its own vital information but a few locations are free. Indexed, indirect addressing requires two bytes on page 0 to work and we will use $01 and $02. This addressing takes the two bytes on page 0, adds the Y register, and treats the result as an address of a location somewhere else in memory. The two bytes on page 0 are stored in the 'backwards', low byte-high byte order of the 6502. Thus $033A is stored as 3A 03. This means that we may reach anywhere in memory using this addressing mode. For example, if the Y register contained 0A,

$01 contained $00 and $02 contained $80, then the command STA ($01),Y would store the accumulator at $800A. By incrementing Y, $01, and $02 we can reach anywhere we want. Here is a short program which would fill the first line of the screen with stars:

```
line# loc  code      label  mnemonic      comment

100   0390 A9 00            LDA #$00      ;STORE LO BYTE OF START
110   0392 85 01            STA $01       ;...OF THE SCREEN AT $01
120   0394 A9 80            LDA #$80      ;STORE HI BYTE OF START
130   0396 85 02            STA $02       ;...OF THE SCREEN AT $02
140   0398 A9 2A            LDA #$2A      ;LOAD A W/ CODE FOR *
150   039A A0 00            LDY #$00      ;ZERO Y FOR COUNTER
160   039C 91 01    LOOP    STA ($01),Y   ;STORE A ON SCREEN
170   039E C8               INY           ;INCREMENT Y
180   039F C0 28            CPY #$28      ;DONE 40 SPACES?
190   03A1 D0 F9            BNE LOOP      ;...IF NOT KEEP GOING
200   03A3 60               RTS           ;...ELSE GO BACK TO BASIC
```

Of course, there are other ways to do this but this illustrates a simple use of indexed, indirect addressing. The program is well commented. Notice that this is an assembler listing so that you may type in the source code and comments on the right if you have an assembler. The line numbers are just for reference. If you just want to type in the actual machine language program then type in the numbers that appear under 'code' in the correct locations. The program starts at $0390 and ends with a return at $03A3. You would activate the program with a SYS912 form BASIC. Lines 100 to 130 set up the address for the beginning of the screen in lo-hi form as mentioned above. Line 160 takes the address found at $01 and $02 and adds the value of the Y register to obtain the effective address where the accumulator will be stored. The Y register is the only one which may be used with this addressing mode. Since Y is 0 the first time around we are storing at location $8000 or the first screen location. The next time Y is 1 and we store at $8001 and so on until we have done 40 spaces. The number of spaces is controlled by line 180.

The next program uses indexed, indirect addressing to reverse each character on the entire screen. When called from BASIC with a SYS912 this makes a very useful effect. The basic idea is to load a character from a location, reverse it, and then store it back again. We will finish one 'quarter' of the screen and then do the next until all four are done. Here is the program which does this:

```
line# loc  code      label  mnemonic      comment

100   0390 A9 00            LDA #$00      ;STORE LO BYTE OF START
110   0392 85 01            STA $01       ;...OF THE SCREEN AT $01
120   0394 A9 80            LDA #$80      ;STORE HI BYTE OF START
130   0396 85 02            STA $02       ;...OF THE SCREEN AT $02
140   0398 A0 00            LDY #$00      ;ZERO Y FOR COUNTER
150   039A A2 00            LDX #$00      ;ZERO X FOR QUARTER COUNTER
160   039C B1 01    LOOP    LDA ($01),Y   ;LOAD FROM SCREEN
170   039E 49 80            EOR #$80      ;FLIP CHARACTER
180   03A0 91 01            STA ($01),Y   ;STORE BACK TO SCREEN
190   03A2 C8               INY           ;BUMP Y
200   03A3 D0 F7            BNE LOOP      ;DO AGAIN IF 1/4 NOT COMPLETE
210   03A5 E6 02            INC $02       ;GET READY FOR NEXT QUARTER
220   03A7 E8               INX           ;COUNT 1/4'S TO SEE
230   03A8 E0 04            CPX #$04      ;...IF 4 ARE DONE
240   03AA D0 F0            BNE LOOP      ;...IF NOT DO ANOTHER
250   03AC 60               RTS           ;GO BACK TO BASIC
```

Here we set up the indirect address in lines 100 to 130 in the same way we did before. In 140 and 150 we simply zero x and y for counters. The Y register will keep track of what character we are working on within a quarter while X will count the quarters we have done. Lines 160 to 200 create a loop which handles a single quarter. In line 210 we bump the high byte of our indirect address to get ready for another quarter screen. We also increment X so that we can use it to count the quarters we have done. Lines 230 and 240 do this test and branch back if we have more to do.

Line 170 contains the EOR or 'exclusive-or' which flips normal characters to reverse and vice versa. EOR is usually used to flip bits in a byte for 0 to 1 or 1 to 0. Here we EOR with $80 to do the reversal but this must be converted to binary to see what is going on. First, the table for EOR looks like this:

```
0  EOR  0  gives  0
0  EOR  1  gives  1
1  EOR  0  gives  1
1  EOR  1  gives  0
```

Let's say we have a space whose code is $20 and we want to flip it to a reverse space whose code is $A0. The way we do this is to simply flip the 7th bit and this works for every character. Here's what it looks like:

```
space      $20  00100000
EOR with   $80  10000000
result is  $A0  10100000
```

We simply place a 1 in whatever bit position we want to flip and a 0 in every other position. We then convert the binary number to hex and use it in our program.

The next program is quite a bit more complex which means it is also more versatile. This program also will reverse the entire screen. However, it can be set to reverse any 'window' on the screen by setting certain values correctly. I have commented the program and will attempt some more explanation after presenting it.

```
line# loc  code      label  mnemonic       comment

100                  PTR1=$01             ;LOC FOR LO SCREEN ADDRESS
110                  PTR2=$02             ;LOC FOR HI SCREEN ADDRESS
120                  LENH=$B9             ;LOC FOR LINE LENGTH
130 ;
140   0390 A9 28            LDA #$28       ;LOAD A W/ LINE LENGTH
150   0392 85 B9            STA LENH       ;...AND STORE
160   0394 A2 00            LDX #$00       ;ZERO LINE COUNTER
170   0396 A9 80            LDA #$80       ;SET UP INDIRECT ADDRESS
180   0398 A0 00            LDY #$00       ;USING A AND Y REGISTERS
190   039A 85 02    LOOP    STA PTR2       ;STORE ADDRESS OF LINE
200   039C 84 01            STA PTR1       ;...IN POINTERS
210   039E A0 00            LDY #$00       ;LOAD Y W/ START COLUMN
220   03A0 B1 01    LINE    LDA (PTR1),Y   ;LOAD A W/ CHAR ON LINE
230   03A2 49 80            EOR #$80       ;...AND FLIP IT, THEN
240   03A4 91 01            STA (PTR1),Y   ;STORE BACK AT SAME LOC
250   03A6 C8               INY            ;BUMP Y FOR ANOTHER CHAR
260   03A7 C0 14            CPY #$14       ;CHECK TO SEE IF ALL DONE
270   03A9 D0 F5            BNE LINE       ;...IF NOT DO ANOTHER

280   03AB 18               CLC            ;MOVE TO NEXT LINE
```

```
290   03AC A5 01          LDA PTR1      ;...BY INCREASING POINTER BY
300   03AE 65 B9          ADC LENH      ;...LINE LENGTH
310   03B0 A8             TAY           ;STORE TEMPORARILY
320   03B1 A5 02          LDA PTR2      ;LOAD A FROM HI POINTER
330   03B3 69 00          ADC #$00      ;...AND ADD CARRY TO HI
340   03B5 E8             INX           ;INCREMENT X TO SEE IF
350   03B6 E0 0A          CPX #$0A      ;...ALL LINES DONE
360   03B8 D0 E0          BNE LOOP      ;...IF NOT DO ANOTHER LINE
370   03BA 60             RTS           ;RETURN TO BASIC
```

Call this program using SYS912. There are several interesting parts to this program which need to be further explained. Since we will constantly refer to locations $01 and $02, we give them a label of PTR1 and PTR2. We also do the same for a location which will contain our line length. In line 170 and 180 we set the starting line of the window. In this case we set it to be the first line which obviously has the same address as the start of the screen, $8000. The second line would be $8028 since it is forty more spaces. Therefore we would load A with $80 and Y with $28. The 10th line would be $8168, the 19th $82D0 and so on. line 190 and 200 are used to store the address of the line we are working on at the pointer locations.

In line 210 we load Y with the column we wish to start in. The Y register will then be used as a counter to keep track of what column we are working on. Lines 220 to 270 then set up a loop which loads a character, reverses it and stores it back on the screen. Line 260 checks to see if we have done all the characters we want to do by comparing Y to some value. In our case Y started at 0 and we are comparing it to $14 or decimal 20. This means we will reverse a line of 20 characters in columns 0 to 19. To do 10 characters from column 30 to 39 we would start Y at $1E in line 210 and compare it to $27 in line 260.

When we have finished one line we go on to the next starting at 280 where we prepare to add by clearing the carry. We load A with the lo byte of the line address and then add the line length in 300 and store the result in Y. Here is where a problem may result. Say we were on line number 7 whose address is $80F0. If we add 40 or $28 to the $F0 we would get $18. Now $8018 would not be the correct next line and the reason is that a carry has resulted in the addition. This carry must be added to the hi byte so that the address of the next line will be $8118. To accomplish this we load the accumulator with the hi byte and then, in line 330, add $00 without clearing the carry. If there was a carry it will get added to the hi byte and everything will be okay.

Finally, we increment X and in line 350 check to see if we have done the number of lines we wanted. In our case the number of lines was $0A or decimal 10. This program is somewhat complex but is very versatile. I suggest you type it in and try changing the size and position of the window to try to get an understanding of how it operates.

Subroutines, as you know from BASIC, can shorten a program and make things much neater. You can create your own routines in your programs or make use of the PET's own routines built into its ROMs or you can do both. The latter case can save a lot of work but is subject to some pitfalls. The major problem with using ROM routines is that Commodore keeps changing where they are located and this makes a program that relies heavily on them less portable. Some routines have stayed the same throughout all three versions of BASIC and seem safe to use. A routine at $FFD2 will write to the screen the character whose ASCII value is in the accumulator. I abbreviate this as WRT. To check to see if the stop key is being pressed we can use the routine at $FFE1 wich I call CHK. If we want to get a character from the keyboard we use the GET routine at $FFE4. The following program is very straightforward and uses simple indexed addressing and the WRT routine. It prints a message you have stored in ASCII to the screen:

```
line# loc  code      label  mnemonic       comment

100                 WRT=$FFD2             ;WRITE A CHARACTER TO SCREEN
110 ;
120   0390 A2 00            LDX #$00      ;ZERO XR FOR COUNTER
130   0392 A9 93            LDA #$93      ;LOAD A WITH CLR SCREEN
140   0394 20 D2 FF         JSR WRT       ;USE PET ROUTINE TO PRINT
150   0397 BD A3 03 LOOP    LDA MSG,X     ;LOAD A FROM MESSAGE
160   039A 20 D2 FF         JSR WRT       ;...AND PRINT CHARACTER
170   039D E8               INX           ;BUMP X BY ONE
180   039E C9 0D            CMP #$0D      ;LOOK FOR MESSAGE ENDING CR
190   03A0 D0 F5            BNE LOOP      ;...IF NOT THEN DO AGAIN
200   03A2 60               RTS           ;RETURN TO BASIC
210   03A3 50 45 54 MSG     TXT "PET"
220   03A6 0D               BYT $0D
```

The last two lines may confuse some of you who have not used an assembler. Without an assembler you would simply type in the values 50, 45, 54, and 0D right after the program. Most assemblers allow you to give labels to any area of memory. Here MSG is the label for the beginning of the message. TXT and BYT are psuedo-ops used by the assembler. TXT allows you to enter you message as a string surrounded by quotes. The assembler takes care of translating your message to the proper ASCII in hex. BYT allows you to enter a list of 8 bit values in almost any form. Now you can see how convenient an assembler is!

As I said, you can use the PET's routines or create your own or do both. We will now do both by first creating a short routine which will then become a subroutine which we will call from a larger program. This routine will wait for us to enter a single digit and will convert that digit to the actual number. Thus 0 which is ASCII $30 will be converted to a binary 0. We will also check to see if the stop key is being pushed. Here is that routine:

```
line# loc  code      label  mnemonic       comment

100                 WRT=$FFD2             ;WRITE A CHARACTER
110                 CHK=$FFE1             ;CHECK THE STOP KEY
120                 GET=$FFE4             ;GET A CHARACTER
130 ;
140   0390 20 E1 FF NUMIN   JSR CHK       ;TEST IF STOP KEY IS PRESSED
150   0393 20 E4 FF         JSR GET       ;GET A CHARACTER
160   0396 C9 30            CMP #$30      ;TOO LOW?
170   0398 90 F6            BCC NUMIN     ;...GET ANOTHER
180   039A C9 3A            CMP #$3A      ;TOO HIGH?
190   039C B0 F2            BCS NUMIN     ;...GET ANOTHER
200   039E 20 D2 FF         JSR WRT       ;PRINT TO SCREEN
210   03A1 29 0F            AND #$0F      ;CONVERT ASCII TO BINARY
220   03A3 60               RTS           ;RETURN TO BASIC
```

Activate NUMIN using SYS912. So, we get our character and compare it to $30. If the carry is set, then the ASCII value of what we got is greater than or equal to $30 and we go on to compare it to $3A. If the carry is clear then what we got was less than $3A and we can go on. In line 210 we encounter AND which is a relative of EOR and is another logical operator. AND is usually used to turn off certain bits and its table goes like this:

```
0  AND  0  is  0
0  AND  1  is  0
1  AND  0  is  0
1  AND  1  is  1
```

To turn off the bits we want we place a 0 in that position. All other position which are to be left alone get a 1. Let's suppose we hit a 3 whose ASCII value is $33. We want to make it a simple 3 so here's how it looks:

```
3 is ASCII $33 or  00110011
AND  with  $0F     00001111
which results in   00000011
```

We are simply stripping away the ASCII bits to leave a pure 3 behind. We could have subtracted $30 also but AND saves time and space. The proper way to use AND may not always be obvious but it is worthwhile.

We are now ready to use NUMIN as a subroutine in a larger program. This program will allow us to enter two digits and it will then add them for us. Better still, it will print the problem as we go with the addition and equals sign. The program below assumes that the routine above, NUMIN, has been entered starting at $0390 and ending at $03A3. So here is the main program that calls NUMIN:

```
line# loc  code      label  mnemonic      comment

230 ;MAIN PROGRAM LOOP
240 ;ACTIVATE WITH SYS932
250 ;
260   03A4 20 90 03        JSR NUMIN     ;GET A DIGIT
270   03A7 8D 3A 03        STA $033A     ;SAVE IT SOMEWHERE
280   03AA A9 2B           LDA #$2B      ;LOAD A W/ PLUS SIGN
290   03AC 20 D2 FF        JSR WRT       ;PRINT IT ON SCREEN
300   03AF 20 90 03        JSR NUMIN     ;GET ANOTHER DIGIT
310   03B2 AA              TAX           ;SAVE IT SOMEWHERE
320   03B3 A9 3D           LDA #$3D      ;LOAD A W/ EQUALS SIGN
330   03B5 20 D2 FF        JSR WRT       ;PRINT IT ON SCREEN
340   03B8 8A              TXA           ;BRING DIGIT BACK
350   03B9 18              CLC           ;GET READY TO ADD
360   03BA 6D 3A 03        ADC $033A     ;DO ADDITION
370   03BD 09 30           ORA #$30      ;CHANGE TO ASCII
380   03BF 20 D2 FF        JSR WRT       ;PRINT RESULT ON SCREEN
390   03C2 A9 0D           LDA #$0D      ;LOAD A W/ CR
400   03C4 20 D2 FF        JSR WRT       ;TO FINISH THINGS UP
410   03C7 60              RTS           ;RETURN TO BASIC
```

There are a few things here to point out. First, since our subroutine comes first we do not SYS to the beginning of the code but to the beginning of the main loop. Use SYS932 to start things up. Since we are using the accumulator to help print + and = we cannot leave the digits we enter in the accumulator. We store the first digit in some unused spot like $033A. The second digit we temporarily transfer to the X register which is not being used and then pull it back moments later. We could have stored this digit at $033B but the transfer instructions are quicker and save two bytes each way. After we get our result we must convert it to ASCII since ASCII $09 is an unprintable character. This conversion is done in line 370 by a relative of AND and EOR called ORA. ORA is usually used to turn bits on and thus is somewhat opposite in nature to AND. To turn bits on we simply place a 1 in that bit position and a 0 in any position which we do not want to effect. The table for ORA looks like this:

```
0  ORA  0  is  0
0  ORA  1  is  1
1  ORA  0  is  1
1  ORA  1  is  1
```

Let's take our $09 and ORA it with $30 to change it to ASCII $39.

```
Nine or $09 would be 00001001
ORA with $30         00110000
results in           00111001
```

Again, understanding how this works and implementing it at the correct time are two different things.

If you try this problem you will soon see that a problem occurs when the answer goes over 9. A good challenge is to add the code which will handle any answer from 0 to the largest possible answer of 9+9=18.

The PET ROM routines and subroutines in general need some comments before we close this session. If you have tried the examples, you know that the GET routine is a get just like in BASIC. You must provide a loop until the user presses a key. Also, just like in BASIC, there is no cursor. There is an INPUT routine which can be accessed but this has changed position in all three BASICs and would be a bad idea. For next time try to construct a rotuine which simulates input. Store the characters enetered by GET in your own buffer and have some location tell how many characters have been entered. Terminate on a carriage return but do not count it as one of the characters. Be sure to implement delete. Another comment, this time about the WRT routine. I'm sure you have noticed that it does not print a carriage return after each character. This can be both a boon and a bane. Notice, too, that it will print any character including cursor controls. There are special routines to clear the screen and position the cursor but these also have changed locations between different versions of BASIC. Another assignment is to construct a subroutine which when passed a 'down' and 'over' positions the cursor correctly.

We have used subroutines but have never said how the PET implements them. A little knowledge in this area is desireable since it may help us avoid some nasty pitfalls and introduces several interesting concepts. The PET, like most other microcomputers, reserves an area in memory called a 'stack'. In the PET the stack starts at 511 or $01FF and extends downward to about 256 or $0100. Information can be 'pushed' onto the stack or 'pulled' off and it is many times used for temporary data storage. The stack is a last in, first out or LIFO structure which means that if we put four pieces of data on we must take off four to get back to the first value we put there. As data is added to the stack it grows downward in memory; as data is removed it retreats. To keep track of the next available space for data the PET uses the 'stack pointer'. When you turn on the PET and call the monitor, the value $F8 will probably appear under SP in the register display. This means that the next byte sent to the stack will be stored at location $01F8. Let's look at an example of part of the stack and how it and the SP are effected:

```
Before push          After push
          Stack                Stack
SP=$F8    $01F9 A9   SP=$F7    $01F9 A9
 A=$EE    $01F8 XX    A=$EE    $01F8 EE
          $01F7 XX             $01F7 XX
          $01F6 XX             $01F6 XX
          $01F5 XX             $01F5 XX
```

Notice how pushing data onto the stack decremented the stack pointer once for each value pushed. Pulling data from the stack works in reverse as shown below:

```
Before pull          After pull

          Stack                Stack
SP=$F7    $01F9 A9   SP=$F8    $01F9 A9
 A=$00    $01F8 EE    A=$EE    $01F8 EE
          $01F7 XX             $01F7 XX
          $01F6 XX             $01F6 XX
          $01F5 XX             $01F5 XX
```

What is not shown is that when a push is done the information is pushed first and then the stack pointer is decremented. Conversely, when a pull is done the pointer is incremented first and then the data is pulled. Notice that the data on the stack remains the same, only the pointer changes. The commands that you can use to store and retrieve data from the stack are as follows:

PHA - push accumulator onto stack
PLA - pull accumulator from stack
PHP - push status register onto stack
PLP - pull status register from stack

We will talk about some uses for these commands next time but for now we must finish our discussion of subroutines and the stack. When a subroutine is called in BASIC or machine language the computer must know where to go back to when a RETURN or RTS is encountered. To record this the address to which the computer will return control is pushed onto the stack. Actually, the information pushed is one less than where the program really will go next. When the RTS pulls this back and places it into the PC, the PC increments by one to obtain the address of the next instruction. When the address is pushed onto the stack the low order byte is pushed first followed by the high order byte.

All of this is informative but there are just two rules to follow, in general. First, if you push always pull shortly afterward. Second, if you use JSR to get to a subroutine always get back using RTS. Violating either of these rules will cause the stack to fill up quickly and become useless. Experienced programmers can violate these rules but only with appropriate caution. For example, you could jump to a subroutine and then pull the return address off, push another on, and go anywhere you wanted. More about this in our next installment.

This has been a long lesson and a complicated one. I suggest you reread carefully, try all the examples and do the suggested exercises. In the next, and last, article in this series I will give the answers to the exercises and combine them into a short program. I will also discuss more about the stack, the rotate and shift instructions, how to put your program other places in memory, and getting BASIC variables from machine language programs. Also included will be a short piece on interrupts and wedges with a short program to constantly display any page of memory you want on the screen. Many examples will be included. Please call or write if you have questions or suggestions.

****

Time flies when...

Sorry it's been a while since we last met. Many of you called to see if we were still alive, and of cource we are. Publishing a newsletter in our spare time from two different places presented more problems than we thought, but please don't feel cheated. Remember that your subscription is by volume, not by year, so you will get all six issues, albeit on a somewhat irregular schedule.

Teaching Tools for Computer Literacy

by Bill Batcher

A few years ago, I went into a local Radio Shack to buy some integrated circuits. I explained to the young clerk that I was going to show some of my students something about how a computer worked. His advice, of course, was to buy a TRS-80. In one way, he was right. The best way for kids to learn about the "world of computers" is to fiddle around with a computer. A microcomputer (in my estimation, a PET microcomputer) is the best tool available for teaching computer literacy.

There are however, times when a teacher may want to consider some of the scores of other teaching aids available for teaching computer literacy. In this article I'll talk about 10 of these that I've had experience with, their assets, their pitfalls, and some of the ways they might be integrated into a computer course. I'll stick to materials for teaching about history, applications, and the inner workings of a computer, rather than getting into the vast field of aids for teaching programming.

To provide a broad overview of the computer field, its beginnings, its growth, how it is used in differeent fields, the parts of the computer, etc., there are two excellent tools: the first is an AV kit called **COMPUTERS FOR KIDS** and the other is a book called **WHAT IS A COMPUTER?**

COMPUTERS FOR KIDS was developed by Donald Spencer, and is available form Camelot Publishing Company, Ormond Beach, Florida. It consists of a set of 35mm slides with cartoon drawings, a taped narration, a teacher's manual and a couple of books for kids. The explanations are clear and interesting, although I usually omit the final section on programming. Most of the pictures on the slides can be ordered as 8 1/2 by 11 posters as well. Also the books can be purchased separately.

WHAT IS A COMPUTER was written in 1972 by Marion Ball, but only occasionally shows its age. It's published by Houghton Mifflin and the illustrations are highly spirited and colorful. The text sometimes slips into rhyme. Often, the author tells us some little know anecdotes about computer history which restore the human element to an area that can sometimes become too much nuts-and-bolts. Both **COMPUTERS FOR KIDS and WHAT IS A COMPUTER?** can be used with a wide range of students, from upper elementary to junior high.

I like my students to know something about the guts of the PET and I always have at least one micro in the classroom with the screws removed so it can be flipped open quickly. A number of aides are designed to teach about one aspect or another of how a computer works:

The CARDIAC was developed in 1968 by David Hagelbarger and Saul Singerman and was once available free from Bell Telephone Labs. Nowadays you have to pay for it. It gives kids an idea of how the CPU can interpret some numbers as data and others as instructions, how numbers can stand for operations, what an accumulator does, etc. The nice part about CARDIAC is that it is all done in decimal. For a student in high school who is going to start learning machine language, the CARDIAC provides a nice introduction. However, it is very tedious to go through an actual program step-by-step and I've found it of little use below 7th grade.

DIGITAL DESIGN is a kit put out by T E R C in Cambridge, MA. It's designed for high school kids learning about logic gates, AND, OR, and how they can be combined to produce a counter, a timer, an adder, etc. The kit sells for around $100. I used the kit without the instructions, with a group of gifted sixth graders and we had a ball. When the kids see the LEDs flickering on and off, either verifying or refuting their predictions, it makes the whole area of logic gates very clear. These kids were able to construct truth tables for circuits using several gates, were eventially able to count up to 16 in binary using LEDs, and in general were able to grasp the concept that a computer can get a

lot of mileage out of a series of two-state bits. I'd like to see a kit specifically designed for elementary kids. For one thing, the circuit board was about 2x5 inches, and was often covered with a maze of chips and wires, that made it difficult for even a small group of 8 kids to see just what was going where. Island Software may develop such a kit in the near future.

Many year ago I bought a SF-5000 "Electronic Digital Computer" from Science Fair (a division of Tandy) for about $30. It was a mistake. There were no electronics in the whole box, other than bell wire, a flashlight bulb and clips to attach a dry cell. The kit, originally developed in Germany by Rolf Lohberg does get into some elementary logic, with AND and OR gates. However, the switching is done mechanically and I found the plastic, manually operated slides kept knocking off the metal contacts. In all, it was very frustrating. I kept seeing a picture of this thing advertised under different trade names, so the teacher ought to be wary of anything that purports to be a computer kit that has no integrated circuitry.

HEXAPAWN is a game that was first introduced nationally in 1962 in Martin Gardner's excellent column in Scientific American. Later he included it in his 1968 book **THE UNEXPECTED HANGING.** IBM in Armonk, NY, has a version available commercially, but I'm prejudiced in favor of the original. The 1962 version used juicyfruit candy in matchboxes and whenever the computer made a mistake you punished it by eating one of its candy. The amazing things is that the punishment worked, for eventually the computer "learned" not to make mistakes. If you want a tool for teaching about heuristic learning or artificial intelligence, HEXAPAWN is the tool. Even if you buy one of the IBM versions, you still ought to have the kids make their own boards, and find all the possible moves themselves.

There are another three tools available from Creative Computing, Morris Plains, NJ 07950. First, is a board game, COMPUTER RAGE. Actually, the game is really beginning to show its age now, since it "simulates" the operation of a time-shared system, and many of the problems have no relevance to microcomputers. However, it does use a novel set of binary dice, and after playing a session of COMPUTER RAGE, kids really know their binary numbers. The game sells for $8.95, but they'll sell them in lots for $4.50 each. By the way, the set of 3 binary dice can be bought separately for $1.25, or 5 sets for $5. That might be a better investment, and let the kids make up there own games.

COMPUTER COIN GAMES, also from Creative Computing, is a book of games by Joe Weisbecker, that cleverly shows the ramifications of flip-flop gates. Unfortunately, the book doesn't teach much of anything about computers except flip-flop gates, but at $3.95 it's worth it just for that one explanation.

KATIE AND THE COMPUTER is Creative Computing's latest creation and is superb. Don't be put off by the title or the delightful imagery. In language, any fourth grader can understand, this book accurately describes the workings of the CPU, the program, what a byte is, what a ROM is, and even the meaning of a bug, while it traces Katie's Alice-like adventures in the world of Cybernia. For $6.95, I was pleasantly impressed.

Finally, I want to mention BIG TRAK, a programmable tank built by Milton Bradley. So simple, it's commonly bought for kids in the lower elementary grades, but don't overlook its potential for older kids. A teacher can give students problem such as make BIG TRAK form the letter N. Or make it cross the room, wait 4 seconds, dump its load, turn around and return at half speed. There's no limit and problems like these will force the kids to think about a program as a series of commands which are first entered into memory and then executed. A student can even see why programmers skip numbers when they program in order to add new commands. This is a natural way to give kids an idea of the concept "program" before they begin to learn a programming language.

So, these are some of the tools I have tried to increase computer literacy among my students. Please let me know about the ones you have tried. Many good tools have been developed by individual teachers. What do you use as "tools"?

Swap-Put: A Few Useful Routines

by Ralph Bressler

On the original ROM (BASIC 1.0) PETs it was possible to blank the screen without losing what was printed on it. This was done with a POKE and meant that graphics displays could be created and then caused to appear suddenly as if by magic. The screen could be updated while it was blanked and then be brought back at the proper time. The ability to do this has been disabled in the ROM version since the original but many people would still like to be able to do it. The questions and comments of several readers caused me to write the program below. I have included the BASIC listing and assembly source code.

Swap-put is a set of four routines which operate on the screen and its 'clone' which is created in high memory. The routines allow you to swap the screen and clone, clear the clone, move to screen to the clone or the clone to the screen. In the latter two cases a copy is made of what is moved but what existed where the copy is placed is destroyed. In other words, if I move the clone to the screen what was in the clone is copied to the screen and what was on the screen is lost. You create graphics displays by PRINTing or POKing to the screen or by POKing to the clone. The clone cannot be PRINTed to.

The BASIC program, which is well commented, automatically finds the top of YOUR memory and lowers it to accomodate the clone and the routines. Since these routines use relative addressing and branch commands they are completely relocatable. This means that the routines do not sit in the crowded second cassette buffer. A more elegant technique would be to have the machine language program do this itself as does Supermon. The BASIC program also should take care of adjusting for an 80 column machine but this has not been tested. When run, the BASIC program will tell you the SYS numbers for each routine and the number for the beginning of the clone.

```
100 REM SCREEN SWAP
120 PRINT"(clr)(rvs)8(off)0 OR (rvs)4(off)0 COLUMNS ?"
130 GETC$:IFC$=""THEN130
140 SP=PEEK(53)-1 : REM SET TOP OF MEMORY
150 POKE53,SP : REM ...DOWN 256 BYTES FOR PROGRAM
160 REM POKE PROGRAM INTO CORRECT LOCATIONS
170 FORI=SP*256TOSP*256+124:READN:POKEI,N:NEXT
180 SC=PEEK(53)-4 : REM SET TOP OF MEMORY DOWN
190 POKE53,SC : REM ...1024 BYTES FOR SCREEN COPY
195 IF C$<>"8" THEN 220
200 SC=SC-4 : REM SET TOP OF MEMORY DOWN ANOTHER
205 POKE53,SC : REM ...1024 BYTES FOR 80 COLUMN SCREEN COPY
210 POKE865,8:POKE920,8:POKE947,8 : REM ADJUST FOR 80 COLUMNS
220 PRINT"(clr)SCREEN COPY STARTS AT"SC*256
230 PRINT"SYS"SP*256"TO SWAP SCREEN AND COPY
240 PRINT"SYS"SP*256+43"TO PUT COPY ON SCREEN
250 PRINT"SYS"SP*256+61"TO PUT SCREEN INTO COPY
260 PRINT"SYS"SP*256+98"TO CLEAR CLONE
270 PRINT:PRINT"POKE"SP*256+109",X THEN SYS"SP*256+98"TO FILL
280 PRINT"CLONE WITH CHARACTER WHOSE CODE IS X
290 DATA165,52,133,1,165,53,133,2,169,0,133,185,169,128,133,186,162,0,160,0
320 DATA177,185,72,177,1,145,185,104,145,1,200,208,243,230,2,230,186,232
340 DATA224,4,208,232,96,165,52,133,1,165,53,133,2,169,0,133,185,169,128
350 DATA133,186,208,16,165,52,133,185,165,53,133
360 DATA186,169,0,133,1,169,128,133,2,162,0,160,0,177,1
380 DATA145,185,200,208,249,230,2,230,186,232,224,4
390 DATA208,240,96,165,52,133,1,165,53,133,2,162,0,169,32
400 DATA160,0,145,1,200,208,251,230,2,232,224,4,208,242,96
```

The following is the source code for the routines mentioned above. It is well commented and should be self explanatory for anyone with a passing knowledge of machine language. I do not want to give a blow by blow account of the code above what is given in the comments. If any questions arise, please drop me a note.

```
100 ;SWAP SCREEN WITH 'NEW SCREEN' AREA
110 ;ALLOWS BLANKING SCREEN OR CREATING
120 ;GRAPHICS DISPLAYS 'INVISIBLY'
125 *=32512
130 ;
135 ;THIS SECTION SWAPS SCREEN AND CLONE
140 7F00  A5 34          LDA $34        ;SET UP CLONE BY LOADING
150 7F02  85 01          STA $01        ;...FROM TOP OF MEMORY
160 7F04  A5 35          LDA $35        ;...POINTER AND STORING
170 7F06  85 02          STA $02        ;...AT LOCS $01 OR $02
180 7F08  A9 00          LDA #$00       ;SET UP SCREEN ADDRESS
190 7F0A  85 B9          STA $B9        ;...AND STORE AT
200 7F0C  A9 80          LDA #$80       ;...LOCATIONS $B9 OR
210 7F0E  85 BA          STA $BA        ;...$BA ON PAGE ZERO
220 7F10  A2 00          LDX #$00       ;ZERO XR FOR COUNTER
230 7F12  A0 00    LOOP1 LDY #$00       ;ZERO YR FOR COUNTER
240 7F14  B1 B9    LOOP2 LDA ($B9),Y    ;LOAD CHAR FROM SCREEN
250 7F16  48             PHA            ;...STORE ON STACK
260 7F17  B1 01          LDA ($01),Y    ;LOAD CHAR FROM CLONE
270 7F19  91 B9          STA ($B9),Y    ;...STORE ON SCREEN
280 7F1B  68             PLA            ;GET CHAR FROM STACK
290 7F1C  91 01          STA ($01),Y    ;...AND STORE IN CLONE
300 7F1E  C8             INY            ;BUMP YR BY ONE
310 7F1F  D0 F3          BNE LOOP2      ;...AND LOOP IF NOT DONE
320 7F21  E6 02          INC $02        ;IF DONE ONE QUARTER
330 7F23  E6 BA          INC $BA        ;...BUMP SCREEN AND CLONE
340 7F25  E8             INX            ;...AND COUNT TO SEE IF
350 7F26  E0 04          CPX #$04       ;...4 QUARTERS ARE DONE
360 7F28  D0 E8          BNE LOOP1      ;...IF NOT, DO ANOTHER
370 7F2A  60             RTS            ;...OTHERWISE RETURN
375 ;THIS PART PUTS CLONE TO SCREEN
380 7F2B  A5 34          LDA $34        ;SET UP
390 7F2D  85 01          STA $01        ;...CLONE
400 7F2F  A5 35[         LDA $35        ;.....ADDRESS AT
410 7F31  85 02          STA $02        ;.......$01 OR $02
420 7F33  A9 00          LDA #$00       ;SET UP
430 7F35  85 B9          STA $B9        ;...SCREEN
440 7F37  A9 80          LDA #$80       ;.....ADDRESS AT
450 7F39  85 BA          STA $BA        ;.......$B9 OR $BA
460 7F3B  D0 10          BNE OVER       ;SKIP OVER NEXT SECTION
465 ;THIS PART PUTS SCREEN TO CLONE
470 7F3D  45 34          LDA $34        ;SET UP
480 7F3F  85 B9          STA $B9        ;...CLONE
490 7F41  A5 35          LDA $35        ;.....ADDRESS AT
500 7F43  85 BA          STA $BA        ;.......$B9 OR $BA
510 7F45  A9 00          LDA #$00       ;SET UP
520 7F47  85 01          STA $01        ;...SCREEN
530 7F49  A9 80          LDA #$80       ;.....AT ADDRESS
540 7F4B  85 02          STA $02        ;.......$01 OR $02
550 7F4D  A2 00    OVER  LDX #$00       ;ZERO XR FOR COUNTER
560 7F4F  A0 00    LOOP3 LDY #$00       ;ZERO YR FOR COUNTER
570 7F51  B1 01    LOOP4 LDA ($01),Y    ;LOAD CHAR FROM ONE
```

```
580 7F53  91 B9              STA ($B9),Y       ;PLACE AND STORE ANOTHER
590 7F55  C8                 INY               ;BUMP YR BY ONE
600 7F56  D0 F9              BNE LOOP4         ;IF NOT DONE, DO ANOTHER
610 7F58  E6 02              INC $02           ;BUMP BOTH CLONE AND
620 7F5A  E6 BA              INC $BA           ;...SCREEN BY ONE
630 7F5C  E8                 INX               ;BUMP XR TO COUNT
640 7F5D  E0 04              CPX #$04          ;...SEE IF 4 DONE
650 7F5F  D0 F0              BNE LOOP4         ;...IF NOT, DO ANOTHER
660 7F61  60                 RTS               ;...OTHERWISE RETURN
665 ;THIS PART CLEARS CLONE
670 7F62  A5 34              LDA $34           ;SET UP
680 7F64  85 01              STA $01           ;...CLONE ADDRESS
690 7F66  A5 35              LDA $35           ;.....AT LOCATIONS
700 7F68  85 02              STA $02           ;......$01 OR $02
710 7F6A  A2 00              LDX #$00          ;ZERO XR FOR COUNTER
720 7F6C  A9 20              LDA #$20          ;LOAD A WITH SPACE CODE
730 7F6E  A0 00       LOOP5  LDY #$00          ;ZERO YR FOR COUNTER
740 7F70  91 01       LOOP6  STA ($01),Y       ;STORE A ON SCREEN
750 7F72  C8                 INY               ;BUMP YR BY ONE
760 7F73  D0 FB              BNE LOOP6         ;IF NOT DONE, DO ANOTHER
770 7F75  E6 02              INC $02           ;BUMP ADDRESS BY ONE
780 7F77  E8                 INX               ;COUNT QUARTERS
790 7F78  E0 04              CPX #$04          ;SEE IF 4 DONE
800 7F7A  D0 F2              BNE LOOP5         ;...IF NOT, DO ANOTHER
810 7F7C  60                 RTS               ;...OTHERWISE RETURN
```

The short program below is a very simple illustration of what can be done with these routines. Time and space have prevented the inclusion of a longer, more elaborate routine. I would be glad to publish a more complete program if anyone submits one.

```
100 REM DEMO FOR SWAP-PUT
110 PRINT"(clr)"
120 REM PRINT CROSS OF *'S ON SCREEN
130 FORI=1TO24:PRINTTAB(16)"* * *"
140 IFI/2=ABS(I/2)THENPRINTTAB(16)"(up) * * "
150 NEXTI
160 PRINT"(home)(10 down)";
170 FORJ=1TO5
180 IFJ/2=ABS(J/2)THEN200
190 FORI=1TO20:PRINT" *";:NEXT:GOTO210
200 FORI=1TO20:PRINT"* ";:NEXT
210 NEXTJ
220 SYS32573: REM PUT SCREEN TO CLONE
230 REM PRINT CROSS OF +'S ON SCREEN
240 PRINT"(clr)":FORI=1TO24:PRINTTAB(16)" + + "
250 IFI/2=ABS(I/2)THENPRINTTAB(16)"(up)+ + +"
260 NEXTI
270 PRINT"(home)(10 down)";
280 FORJ=1TO5
290 IFJ/2=ABS(J/2)THEN310
300 FORI=1TO20:PRINT"+ ";:NEXT:GOTO320
310 FORI=1TO20:PRINT" +";:NEXT
320 NEXTJ
330 REM SWAP SCREEN AND CLONE WITH DELAY
340 SYS32512:FORTD=1TO50:NEXT:GOTO340
```

POWER

A Product Review

Type: Firmware
Model: Any PET with BASIC 4.0
Source: Professional Software
166 Crescent Rd.
Needham, MA 02194
Price: $89.95

by Ralph Bressler

After reading some of my own reviews,it occurs to me that I take a long time to get to the real point and spend too many words explaining what it would be best to read in the manual. I will try to cure both of these shortcomings in this review.

In short, POWER is the best programming aid that I have seen and used to date. It is the only one I now use for my program development. POWER comes as an EPROM which plugs into the PETs PC and can be activated with a simple SYS command. Its 74 page manual, written by Jim Butterfield, is a masterpiece of clarity liberally laced with examples and only a few cautions. It even comes with stickers to place on the PET keyboard to remind you of important functions. POWER is easy to learn and contains many of the old standbys such as find and trace. However, it adds some powerful features such as the ability to change parts of a program, selective renumbering and a definable keyboard.

POWER has a few shortcomings, the first of which is its price. Many of its features are available for a copying fee in programs like BASIC Aid. Other similar products cost as little as half the price. I suspect that the casual programmer who already owns a Toolkit or similar product might want to stay with it. For the professional programmer, interested hacker, or someone just considering purchasing a programming aid, I feel POWER is the one. In the remaining paragraphs I will summarize the features I use most often and mention some other problems which I have encountered.

POWER not only gives you a repeating cursor if you lack one but it implements full scrolling through a program. Holding down the UP/DOWN cursor control allows you to move backwards or forwards through a program with ease. I no longer have to be careful to list just the line range I want or to hit STOP just as the line I want appears.

POWER will delete any range of lines using parameters similar to LIST. It does an extremely nice and very fast job of renumbering programs. The renumbering can be on all or only part of the program with any increment. This is very nice for keeping subroutines numbered the way you want them.

POWER uses the @ for finding text, variables or keywords. Several wildcards are available and the entire program may be searched continuously or only until the next match is found. POWER also implements a change command whose format is clear and which works well. Both finding and changing may be limited to a range of lines. One problem that I ran across is POWER's apparent failure to find some things at times. It seems to me that it occurs most often when a BASIC keyword is imbedded in what you are looking for. This is annoying but easily solved.

POWER has several commands to help in debugging programs. After a error in a line halts a program a simple WHY will point out the mistake. It also has the DUMp command to print out the value of the variables. I only wish it also printed the values for any arrays! The trace feature of POWER is almost overwhelmingly versatile. I don't use this much when programming but have seen it put to good use. The trace may display the line number only or it may include the code and variable values. The display may be at the top of the screen or not as you choose.

POWER's merge feature is somewhat different from others I have seen. After creating a block of code that will be merged into other programs, the code is saved as a sequential file on disk. When it is needed, the file is opened and

the XEC command is executed. This is quite a bit less handy than simply saying MERGE, but I find it usable.

POWER also has the ability to turn each key into an instant keyboard. When this feature is selected, a shifted key produces a BASIC keyword. You may also choose to redefine a key with your own complete phrase. This is done by placing a special REM statement at the beginning of your program which might look like: 1 R EM"V=PRINT N,X,T. Now a shifted V produces the values of these variables. POWER is smart enough to know that you might want a graphics character inside quotes and will produce it under these circumstances. If what you want to do won't fit in one line, then you can define a shited key as an entire subroutine. This again is set up by a REM statement. I used this feature to set up a subroutine to ask for a file name, copy the file to the other disk and then scratch it from the original. Quite useful and powerful.

POWER also has auto line numbering, a command to call the TIM monitor, one to fix BASIC pointers and even one to turn itself off. This is important since POWER uses the cassette buffers and any user ML programs may react unfavorably if POWER is turned on.

The manual also contains information for more advanced programmers on how POWER is written and how it works. This also includes a description of how to add commands. If you need a programming aid be sure to see this one before buying another. I think you'll be very satisfied just as I am.

SuperGraphics
A Product Review

Type: Software
Model: Any PET
Source: AB Computers
252 Bethlehem Pike
Colmar, PA 18915
Price: $30

by Ralph Bressler

My advice to you before you even read this review is to quickly call up AB Computers and order a copy of this program. If you have any interest in fast, smooth animation in 40 x 25 or 80 x 50 with music this $30 package will supply hours of fun. The program itself takes up about 2500 bytes at the top of memory and automatically protects itself from you. As you will see, all the commands are logical and mnemonic. The slim, 10 page manual could provide more examples and explanations but my disk version came with several demo programs which really helped.

SuperGraphics adds many new commands to BASIC. It works on the screen as a grid whose origin is 0,0 at the upper left hand corner. The lower right hand corner is 79,49 or 39,24 depending on which mode you are working in. The 'character graphics' commands treat the screen as a 40 x 25 grid. The command PRINT@10,20;"PET Hello" would print the message 10 spaces across and 20 lines down. No more cumbersome tabs or cursor strings. The command CSET 10,10,15,12 would define an area on the screen which could then be moved using CMOVEL, CMOVER, CMOVED, or CMOVEU. These commands move that entire block quickly and smoothly in any direction. CFILL takes the same parameter format as CSET except for a fifth number that tells what the area will be filled with. CPUT again takes the same parameter format but this time the fifth number tells where in memory this area of the screen should be stored. This is a little tricky since you can clobber other things that were there before. The manual explains how to set memory pointers and avoid problems. CALL brings the image back from memory and puts it on the screen. The parameters here indicate the coordinates of the new location of the image. The possible combinations of these commands combined with CMOVE commands are really outstanding.

If you want to work on the 80 x 50 grid another set of commands is available.

This part of the manual is misleading as it defines the dots you will control as pixels. To me a 40 column PET has 64000 pixels where 80 x 50 is only 4000. The SET and RESET commands control whether a point will be turned on or off. DRAW allows you to make a horizontal or vertical line or to quickly construct a box. FILL sets or resets an area. The MOVE commands can then be used to shift the filled area about the screen. PLOT can be used for horizontal, vertical or diagonal lines.

The SOUND commands are intriguing to say the least. SOUND has two parameters and operates in two modes. SOUND 100,10 would sound pitch 100 for 10 jiffies. If the first number is 255 or less a note is sounded for the indicated duration. When the first number is greater than 255 it specifies the decimal starting address of a song in memory. The second number then tells how many times this song should be played. The song may be POKed into memory or entered with the monitor. It consists of a note followed by a duration. The execution of SOUND 826,4 would play the series of notes starting at 826 until a 1 was reached in memory at which time the notes would play again. No big deal? All this occurs in background! This means the PET is playing Bach while you go on to write other programs or display cartoons.

The screen command PUT transfers the entire screen to some location in memory you specify. DISPLAY brings this screen back. SWAP exchanges what is on the screen with another area of memory. All easy to use and fully explained.

Other useful commands are included to make life easier. TEXT and GRAPHIC toggle the character sets of the PET thus reducing the number of POKEs to memorize. RVS instantly reverses the screen. PAUSE causes the PET to stop for a specified number of jiffies. CLS and HOME clear the screen and home the cursor respectively. I know the PET has programmable cursor controls to do this but I must admit CLS is easier to spot than a reverse fields heart. EXEC executes a program by loading and running it. This can be done under program control and the calling program need NOT be longer as is usually the case. PLIST simply lists a program to the printer. I've looked for this command for some time. SDUMP will transfer the screen to the printer in normal or enhanced characters.

I know that some people are wary of things that seem just too good to be true. I hope that you do pick up a copy of this package since it really is so good. Remember that it uses the CHRGET technique and I suspect the music must use interrupts. It also sits in the top of memory. These factors will render some other aids and programs unusable with it. This is not a major problem. My advice is buy a copy and have some fun!

Chipmate and Triple Flipper

A Product Review

Type: Hardware
Model PET: any PET
Source: Kansas City Computers
5214 Blue Ridge Blvd
Kansas City, MO 64133
Price: $15 for ChipMate, $40 for Triple Flipper

by Ralph Bressler

With the advent of BASIC 4.0 PET users were cut down to 2 free sockets for new ROM based packages. The increasing number of protection ROMs and valuable firmware products has further complicated this problem. Several products are available to allow users to place more than one chip in the same address space or socket. Kansas City Computers has entered the scene with two small, inexpensive but reliable products.

ChipMate isn't much taller than two ROM sockets standing on end. It will plug into any empty socket and allows two ROMs to share the same socket. This, of course, means that one ROM must occupy the lower 2K of the address space while another occupies the upper. This means that two chips that overlap in any way

cannot be used. However, chips that do use different parts of the address space may even be used at the same time providing their coding does not conflict. KCCI uses this primarily so that their Utilirom firmware may be used with other chips. Perhaps more manufacturers will realize that not every 2K chip should start at $A000 or $9000.

Triple Flipper is more expensive but also more versatile. It takes 3 separate ROMs in three sockets and allows switching between them with a simple three position switch. Only one chip may be used at a time. I have been using one for several months now and have found no problems at all. It easily fits into the PET even though I have an MTU board hanging inside. In fact, the Triple Flipper resides on this board in an upside down position and still has no problems. Any ROM that can plug into the PET's main circuit board can be used in the Triple Flipper. The small switch can be mounted just under the overhang of the case to allow easy access. The switching is done through a dc control line rather than a chip select line which helps make Triple Flipper less susceptible to "noise" than some other switches. I'm not sure this is all that significant since I haven't had problems with any of the switches which use the select line method. The top socket is supplied with a set of jumpers to modify it for use with a non-standard ROM or PROM like the 2732. Several Triple Flippers may be tied to the same switch to allow switching several chips with only one switch. The instructions provided are short but clear. They detail how and where to insert the chips and the Tripple Flipper on any PET board. They also show the simple modifications mentioned above. These products are well constructed and perform as advertised for a reasonable price.

Reverse Polish Language

A Product Review

Type: Software
Model: Any PET
Source: Samurai Software
PO Box 2902
Pompano Beach, CA 33062
Price: $49.95 for compiler only, $80.91 for compiler and simulator

by Ralph Bressler

RPL is a new language for the PET/CBMcreated by Tim Stryker of Samurai Software. It is a stack oriented language, thus making comparisons to FORTH inevitable. Tim has taken the best of what FORTH does and written a language which is easier to learn, slighty faster and more conservative of memory than FORTH. RPL is harder to learn than BASIC because of its strange stack-oriented structure and abbreviated commands. It is limited to about 50 special words in its vocabulary but all of these are useful. This means that RPL is not 'extensible'. This contrasts with FORTH which has many words seldom used by beginning programmers. RPL is not a threaded language like FORTH where one command is written in terms of others. In RPL the code generated is stored as p-code much like PASCAL.

Programming in RPL

The best part of programming in RPL is that it uses the PET's screen editor eliminating the need to learn a new editor before starting with the language. Line numbers are used for editing only but this means that programs can easily be saved in the normal fashion. A program is typed in using RPL structure much like a BASIC program would be entered. The example below adds two numbers and prints the result.

```
10 "answer = ",print,4,2,str$,print
```

When you type COMPILE, the compiler generates the p-code and prints the number of errors. If there are no errors, you simply type GO to execute the program. The program need only be compiled once and can be executed as many times as you want. The following program fills random spaces on the screen with random characters and does it VERY fast.

```
10 test: rnd rnd 1000 backslash 32768 + poke test goto
```

For those of you who are wondering, you would use the backslash, not type the word. (A slight failing in WordPro!). Here you see a simple looping structure using the label "test" and "goto". RPL contains the words and structures necessary to produce a neat, structured program. Here is an example which shows the use of nested loops. This one fills the screen in turn with all 256 characters. It is much faster than BASIC but also slower than machine language.

```
10 255,0,for,fn,1000,1,for,#,fn,32767,+,poke,next,.,next
```

Explaining how this program works or giving more examples would not really show much. RPL supports the IF..THEN...ELSE, full keyboard input commands like GET and INPUT, calling subroutines by name and much more. You can even set up arrays of data easily. The stack manipulation commands are in many ways more versatile than those supplied with some FORTHs. For example, there is a command that allows access to any value on the stack no matter how deep where many FORTHs are limited to the top few values.

What About Errors?

With any new language a programmer is bound to make mistakes. At compile time RPL will tell you about many syntactical errors much like a compiled BASIC. It will say "UNRES REF" if you refer to a label which has not been defined or "DUP SYM" if you define a label twice. Similar messages take care of the IF THEN structure. In all cases the line number of the offending syntax is clearly indicated. If your problems are in the logic you used in the program a symbolic debugger is available to trace through your programs step be step. I think this feature is extremely clever, easy to use and very helpful. You can view both stacks or look at what is really happening on the screen at any time. The stacks can be changed and the debugger allows single stepping through your program. You may also set breakpoints in several different ways. This makes it posible to execute a block of code, stop and see the results and then continue. This is a most helpful feature which can be overlokked. The last program example above caused me a few problems since I kept getting a stack overflow. When I ran it under the debugger it became clear that the parameter stack was filling up with useless numbers. A quick ´pop´ to remove a number after it was no longer needed did the trick.

What Can I Use It For?

One drawback of RPL is that it can only be used on a PET/CBM system since it is tied to the ROM routines. It may also seem that RPL can only be used by those who have the language system but this is not true. A programmer may develop individual program module in RPL, compile and debug them and then save them along with a "run-time" library. These modules can then be used on machines which do not have the full RPL system. Entire sets of subroutines can be created since the compiler has a "global symbol" feature. The price of the compiler includes the right to sell the modules and programs you develop. This is a truly remarkable feature for the price. Remember that RPL is about 10 times faster than BASIC depending on the application and easier to write than 6502 assembler. This makes it a good choice for applications that require speed and concise

coding.

Documentation

The most powerful and complex software package is useless without adequate documentation. In fact, a poorly documented package is frustrating since the potential user knows the power is there but is unable to use it. This is why the extensive (60 page) RPL documentation is so valuable. These pages are crammed with clear explanations of commands and program structure. Examples of programming techniques are also included with some short programs. For those that want to delve more deeply into the "hows and whys", information on the structure and operation of RPL is included. Eight appendices cover information like comparisons of RPL and FORTH, RPL memory usage, a quick reference sheet, an explanation of error messages and some common "gotcha's". All of this is packaged in a good quality 3-ring binder with a pocket for loose papers. The documentation is an example of what should be included with every program of this complexity and versatilty. The only thing the manual doesn't explain is how all this can be sold for well under $100.

A Few Problems

RPL was developed for the PET by a single company. It probably will not become another BASIC with many different companies producing versions. This means that the literature and information about RPL will come from only a few sources. This may not be a big problem since RPL seems best suited for developing program modules which will run independently of the language system.

The cryptic symbols representing some commands take some getting used to and the stack oriented nature of the language is difficult for some to learn at first. The manual does a nice job of explaining all the symbols and the stack manipulations begin to seem very logical after only a short time.

RPL will only handle 16-bit (2-byte) integers which limits some applications. It also does not support functions like square roots and the trig functions. Again, this limitation is not as severe as it may seem since RPL can be used in conjunction with BASIC which has these functions.

The Bottom Line

If you are just starting BASIC or are a casual programmer, I feel that it would be best to master the PET's native tongue first. However, if you want to develop routines which are fast and compact, I suggest you take a look at RPL. Assembly language is probably better in some ways but RPL is faster to learn and easier to use and debug. The documentation is excellent and will give you all the help you need to get started. I see RPL as a good way to quickly construct program modules to use with BASIC or machine language. I know of few langauge systems this complete, this well documented, for this kind of price.

****

REVIEWS

Doesn't anyone else have favorite products; programs or hardware that they use? Its hard for one or two people to review products fairly. A good review requires the reviwer to read the manual carefully, spend hours using the product and then some time writing the review. It usually takes me at least 12 hours to do the most cursory review. The best reviewers are people who use a product frequently. How about some comments on your favorite?

VIC POKES

by Arnie Friedman

Try the following POKEs on the VIC:

```
POKE 650,128          - all keys repeat
POKE 646,C            - controls color code
POKE 37879,0 to 255   - controls screen printing
                      - 255 is slow, 0 is fast
POKE 36869,X          - 240 is upper case, 242 is lower case
```

Screen dimension control:

```
POKE 36864,X          - X axis
POKE 36865,X          - Y axis
POKE 36866,X          - width of screen
POKE 36867,X          - height of screen
```

Try playing with these loactions in "Space Invaders":

```
$0E01 - base speed (change 2 to 1 to speed up)
$0E09 - Invader's firing speed
$0E0E - speed of mystery craft
$0623 - change to $66 to build a wall
$0E1A - control firing of base
$0E1C - control firing of base
$0E13 - controls high score
$0E38 - controls number of ships

POKE 1471,72 - makes base indestructable
```

The Index $14.95

Get your computer projects on track fast.

Start with the Index.

You want to start a computer project but you don't know where to begin. You know there's piles of information out there. But you don't have the time to dig through the mess. Start with the Index. And get on track, fast.

Over 30,000 Entries

Over 6 Years of Articles, Editorials & Columns Written on Personal Computers

References from Over 800 Issues of Personal Computer Magazines

Over 45 Publications

SEE YOUR LOCAL COMPUTER DEALER

or

Order from:

MISSOURI INDEXING, INC.
P.O. Box 301
St. Ann, Mo. 63074
(314) 997-6470

80 × 25

PET/CBM™

2000/3000/4000 Series

not using a CRT, or display controller chip

$275.00*

Select either
80 × 25 or **40 × 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

This Ad
Doesn't Belong
Here...

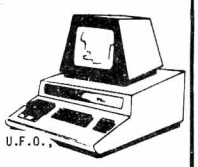
Yours Does!

For Information
Call Ralph Bressler at

(914) 439-3591

VIGIL

VIC

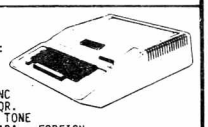


Exciting, new games interactive language.
* Easy to learn with 60+ powerful commands
* Double density graphics, large number display
* LOADing and SAVEing of VIGIL programs to cassette or diskette
* Nine complete programs included - Breakout, SpaceWar, AntiAircraft, U.F.O., SpaceBattle, Concentration, 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 postage.
Orders must be prepaid via check, money order or bankcard. Foreign orders may be paid for via international money order or bankcard: (Access,Eurocard,Barclaycard)

Perfectly Balanced

educational software

from

MICRO-ED

for

PET®

and

VIC®

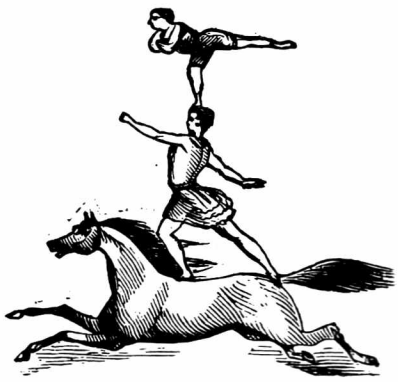
Send for our free catalog*

*please specify PET or VIC

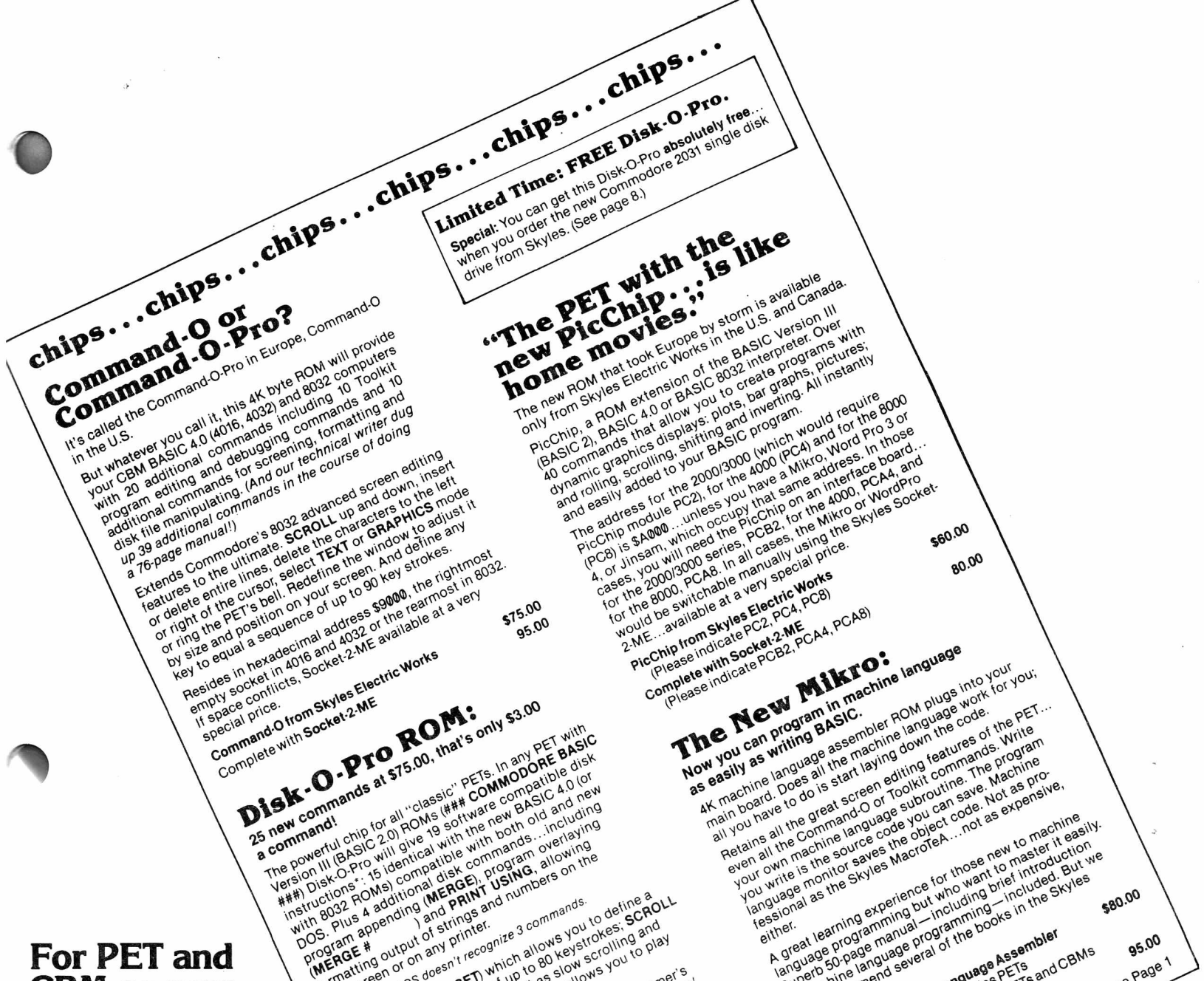
MICRO-ED, Inc. • P.O. Box 24156

Minneapolis, MN 55424

or telephone us at (612) 926-2292



VIC and PET are the registered trademarks for Commodore Business Machines.



chips...chips...chips...chips...chips...chips...chips...

Limited Time: FREE Disk-O-Pro.
Special: You can get this Disk-O-Pro **absolutely free**... when you order the new Commodore 2031 single disk drive from Skyles. (See page 8.)

Command-O or Command-O-Pro?

It's called the Command-O-Pro in Europe, Command-O in the U.S.

But whatever you call it, this 4K byte ROM will provide your CBM BASIC 4.0 (4016, 4032) and 8032 computers with 20 additional commands including 10 Toolkit program editing and debugging commands and 10 additional commands for screening, formatting and disk file manipulating. (*And our technical writer dug up 39 additional commands in the course of doing a 76-page manual!*)

Extends Commodore's 8032 advanced screen editing features to the ultimate. **SCROLL** up and down, insert or delete entire lines, delete the characters to the left or right of the cursor, select **TEXT** or **GRAPHICS** mode or ring the PET's bell. Redefine the window to adjust it by size and position on your screen. And define any key to equal a sequence of up to 90 key strokes.

Resides in hexadecimal address **$9000**, the rightmost empty socket in 4016 and 4032 or the rearmost in 8032. If space conflicts, Socket-2-ME available at a very special price.

Command-O from Skyles Electric Works **$75.00**
Complete with **Socket-2-ME** **95.00**

Disk-O-Pro ROM:

25 new commands at $75.00, that's only $3.00 a command!

The powerful chip for all "classic" PETs. In any PET with Version III (BASIC 2.0) ROMs (### **COMMODORE BASIC** ###) Disk-O-Pro will give 19 software compatible disk instructions*: 15 identical with the new BASIC 4.0 (or with 8032 ROMs) compatible with both old and new DOS. Plus 4 additional disk commands...including program appending (**MERGE**), program overlaying (**MERGE #**) and **PRINT USING**, allowing formatting output of strings and numbers on the PET screen or on any printer.

** **NOTE:** Old DOS doesn't recognize 3 commands.*

Plus softtouch key (**SET**) which allows you to define a key to equal a sequence of up to 80 keystrokes; **SCROLL** whereby all keys repeat as well as slow scrolling and extra editing features; **BEEP** which allows you to play music on your PET.

Completely compatible with the BASIC Programmer's Toolkit. (*The chip resides in the socket at hexadecimal address **$9000**, the rightmost empty socket in most PETS.*) And for the owners of "classic" (or old) PETs, we do have interface boards.

(For those owning a BASIC 4.0 or 8032, even though the Disk-O-Pro may not be suitable, the Command-O is. We have never abandoned a PET owner.)

Complete with 84-page manual written by Greg Yob.
Disk-O-Pro from Skyles Electric Works **$75.00**
Complete with **Interface Board** (for "Classic" PETs) **95.00**

"The PET with the new PicChip...; is like home movies."

The new ROM that took Europe by storm is available only from Skyles Electric Works in the U.S. and Canada.

PicChip, a ROM extension of the BASIC Version III (BASIC 2), BASIC 4.0 or BASIC 8032 interpreter. Over 40 commands that allow you to create programs with dynamic graphics displays: plots, bar graphs, pictures; and rolling, scrolling, shifting and inverting. All instantly and easily added to your BASIC program.

The address for the 2000/3000 (which would require PicChip module PC2), for the 4000 (PC4) and for the 8000 (PC8) is **$A000**...unless you have a Mikro, Word Pro 3 or 4, or Jinsam, which occupy that same address. In those cases, you will need the PicChip on an interface board... for the 2000/3000 series, PCB2, for the 4000, PCA4, and for the 8000, PCA8. In all cases, the Mikro or WordPro would be switchable manually using the Skyles Socket-2-ME...available at a very special price.

PicChip from Skyles Electric Works **$60.00**
(Please indicate PC2, PC4, PC8)
Complete with Socket-2-ME **80.00**
(Please indicate PCB2, PCA4, PCA8)

The New Mikro:

Now you can program in machine language as easily as writing BASIC.

4K machine language assembler ROM plugs into your main board. Does all the machine language work for you; all you have to do is start laying down the code.

Retains all the great screen editing features of the PET... even all the Command-O or Toolkit commands. Write your own machine language subroutine. The program you write is the source code you can save. Machine language monitor saves the object code. Not as professional as the Skyles MacroTeA...not as expensive, either.

A great learning experience for those new to machine language programming but who want to master it easily. Superb 50-page manual—including brief introduction to machine language programming—included. But we also recommend several of the books in the Skyles Book Barn.

Skyles Mikro Machine Language Assembler **$80.00**
MK 2/3 for 2000 or 3000 Series PETs
MK 4/8 for 4000 or 8000 Series PETs and CBMs **95.00**
Complete with **Socket-2-ME**

For PET and CBM owners only:

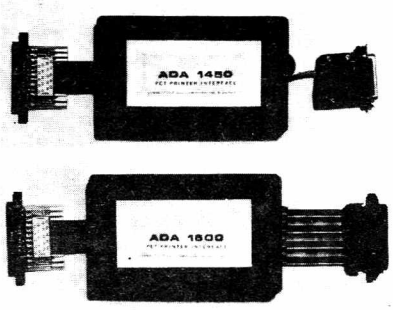
This is just 1 of 20 pages of the newest and biggest Skyles catalog, hot off the press.

We know you'll want this page, in its full 8½ x 10 splendor, and another 19 pages of peripherals, software and books that will make your PET or CBM computer even nicer to live with. So, if we missed sending you your very own copy within the last few weeks, call us at **(800) 227-9998 (unless you live in California, in which case call (415) 965-1735.**

From Skyles Electric Works, the oldest and largest professional specialists in the business.

Skyles Electric Works
231-E S. Whisman Road
Mountain View, California 94041

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.

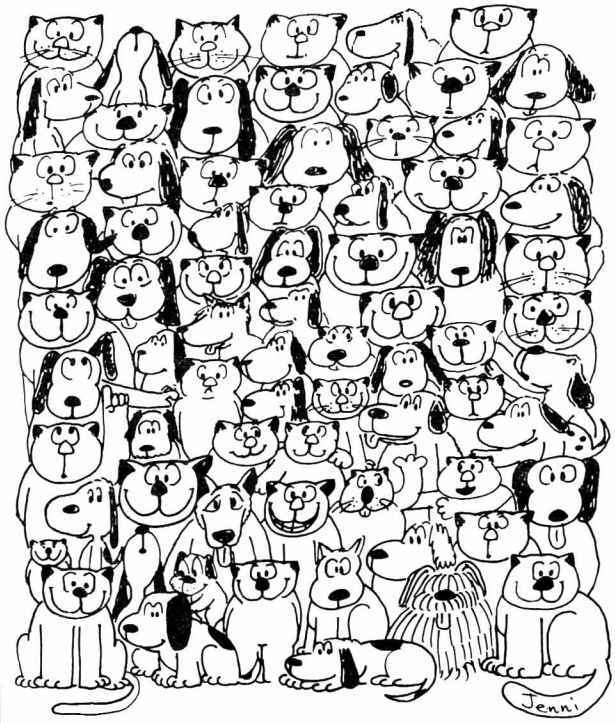
MENTION THIS MAGAZINE WITH YOUR ORDER AND DEDUCT 5% FROM TOTAL.

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

Every PET Needs a Friend.



CURSOR is the best friend your Commodore PET will ever have. Since July, 1978 we have published 150 of the most user-friendly programs for the PET available anywhere. When we write or edit a program, we spend lots of time fussing about how it will treat you. We pay attention to lots of little things that help make using a computer a pleasure instead of a pain.

Naturally, **CURSOR** programs are technically excellent. Each program that we purchase is extensively edited or re-written by a professional programmer. But *imagination* is just as important as being user-friendly and technically good! We delight in bringing you off-beat, unusual programs that "show off" the abilities of your PET or CBM.

CURSOR is user-friendly, technically great and full of imaginative programs. And every issue of **CURSOR** is still available! We continue to upgrade previously published programs so that they'll work on the three varieties of Commodore ROM's (Old, New, and 4.0). New issues also work on the 80 column CBM.

For only $4.95 you can buy a sample issue and judge for yourself. Or send $18 for a four-issue subscription. Each **CURSOR** comes to you as a C-30 cassette with five programs and a graphic Front Cover, ready to LOAD and RUN on your PET.

Who knows? After your PET meets **CURSOR**, things may never be the same!

AUTHORIZED DISTRIBUTORS:

Great Britain
AUDIOGENIC, Ltd.
P.O. Box 88
Reading, Berkshire

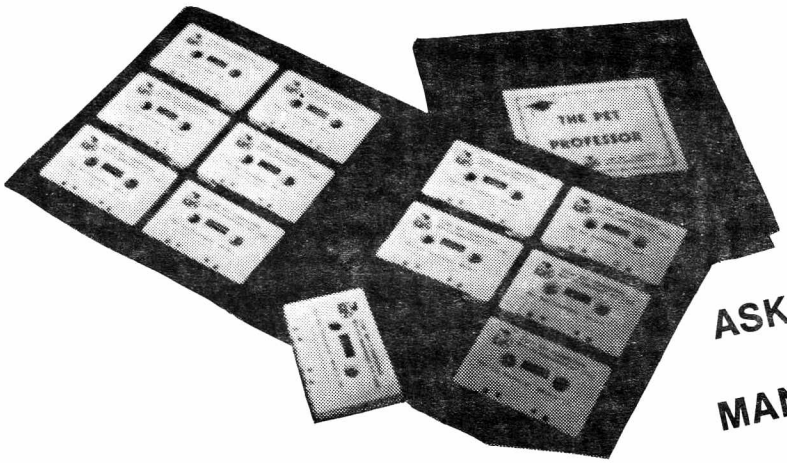
Holland
COPYTRONICS
Bergemeester
Van Suchtelenstraat 46
7413 XP Deventer

Japan
SYSTEMS FORMULATE CORP.
Shin-Makicho Bldg. 1-8-17
Yaesu, Chuo-ku, Tokyo 103

Australasia
MICROCOMPUTER HOUSE, LTD.
133 Regent Street
Chippendale, Sydney
N.S.W. Australia 2008

THE CODE WORKS
Box 550
Goleta, CA 93116
805-683-1585

Introducing the Pet Professor
All you need to do is decide what you need.



ASK ABOUT OUR NEW DISK-BASED MANAGEMENT SYSTEM

The Pet Professor Arithmetic Software

If you need to teach division of a 2-digit decimal by a 1-digit whole number, we have D-D-1. This program teaches the concept step by step.

Since you probably need to keep student interest high, we go very slowly with a moving cursor. The student is comfortable and involved.

Do you need to drill subtraction of a fraction from a mixed number? Just bypass the teaching part of program F-S-2 and go directly to drill. The nice part is if the student happens to forget, the teaching is still available.

If a test on subtraction of whole numbers with 4-digits, multiple zeros and borrowing is your need, program WN-S-6 is your answer.

You probably also need just about every arithmetic concept that there is. We have them all -- 77 programs.

The directions are simple. Use the **Pet Professor** for all the arithmetic you teach.

CBC COW BAY COMPUTING

BOX 515
MANHASSET, N.Y. 11030

For more information send $5 for a sample tape or write and tell us what you need.

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

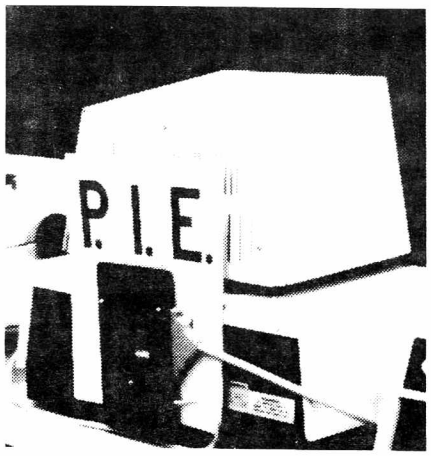
(For technical inquiries, please phone 305-782-9985)

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

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

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.

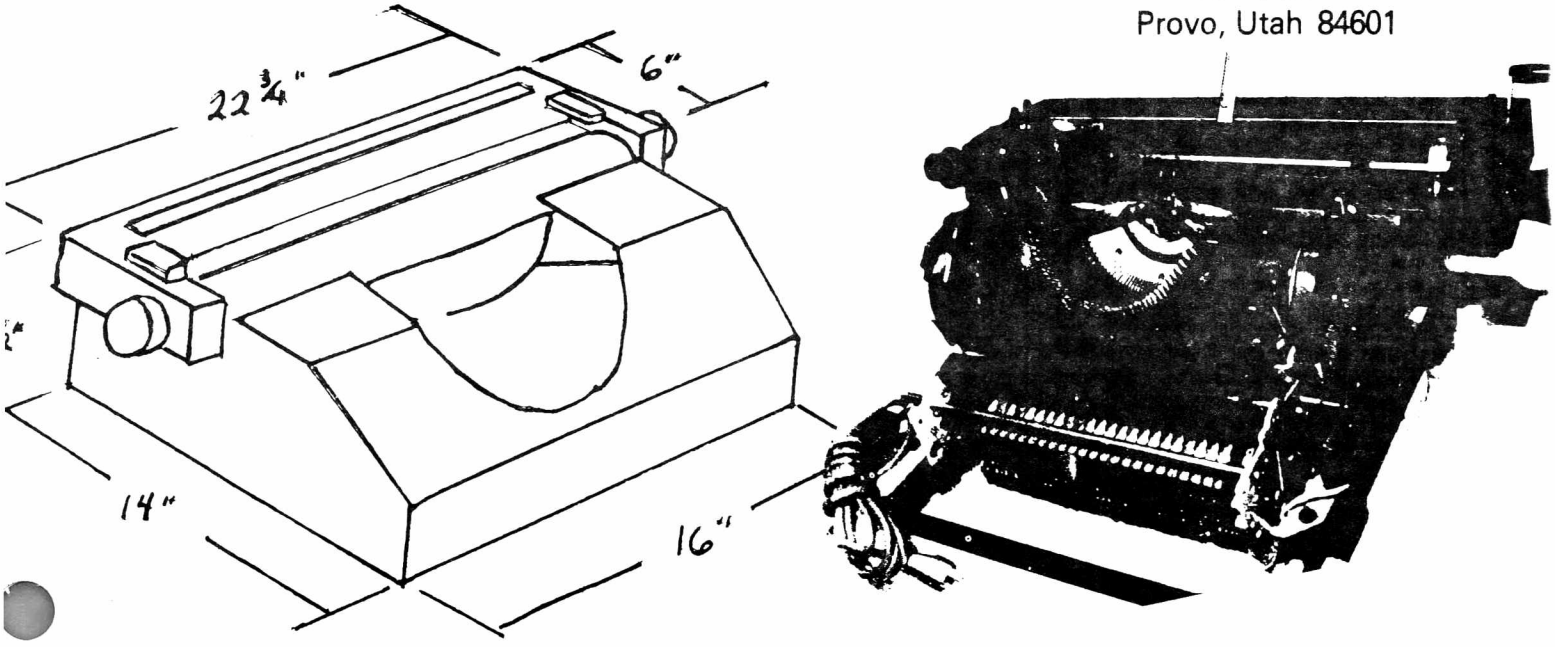
Assembled with custom case, CBM-TO-ASCII code converter and appropriate cable, the **P.I.E.-C** is only 129.95 (+$5 S&H). Md. Res. +5% tax. Specify printer and CBM models.

LemData Products, P.O. Box 1080, Columbia, Md. 21044 Phone (301) 730-3257
*PET/CBM are trademarks of Commodore Business Machines

THE COMPUTER WORKS Div. of C.U.E.S.

Shipping Weight 85 lbs.
Unpacked Weight 55 lbs.
Hammer drive motor 110v AC
Hammer and control solenoids are 24v DC at approx. 1 amp.
We have constructed an RS-232 interface which allows you to connect an ASCII keyboard to it and use it as a terminal.
It has been used on the Southwest Tech, Cromemco and DEC-10 computers.

735 South State Street
Phone (801) 374-0204
(801) 373-7522
P.O. Box N
Provo, Utah 84601



We purchased a good number of these units, they were used in word processors by Royal Litten. The unit is a wide carriage addler, heavy duty business machine with no case, all keys, ribbon select, and carriage return are controled by 24v solenoids. **Our special price $94.99.** FOB Provo, Utah.

☑ Check out some Books from the K-12 MicroMedia Library!

Apple II User's Guide The next best thing to having an Apple. Everything you need to know—and then some. 385pp ☐ **$15.00**

Apple Pascal Games Collection of popular games, clearly documented, flowcharted, and listed. 350pp ☐ **$14.95**

Are You Computer Literate? Good computer literacy text for middle grades thru high school by Karen Billings and David Moursund. 150pp ☐ **$8.95**

Background Math for A Computer World Newly revised self-teaching guide helps you brush up on all that math you forgot to remember—or never clearly understood. 307pp ☐ **$8.95**

BASIC 2nd ed. The classic high school programming text. Over 210,000 copies sold. 325pp ☐ **$9.95**

These books have been widely acclaimed as the definitive—and best—self-teaching guides available. Can turn a novice into a pro in a few weeks! Approx. 300pp each. ☐ TRS-80 BASIC ☐ Atari BASIC ☐ More TRS-80 BASIC ☐ **$8.95 (Each)** ☐ **$9.95**

32 BASIC Programs for the... ☐ **Apple** ☐ **PET** ☐ **TRS-80** Practical applications, games, and graphics, bug-free and ready to run. Approx. 275pp each. ☐ **$17.95**

Be A Computer Literate Colorfully illustrated first look at computers for lower elementary grades. ☐ **$4.95**

Computers and Education Clearly written, concisely presented look at the educational micro-revolution by noted educator, Dr. James Poirot. 96pp ☐ **$7.95**

Computers in Mathematics: A Sourcebook of Ideas Potpourri of problems, puzzles, programming & Teaching ideas, compuart and many other tidbits with a twist of math. 224pp ☐ **$15.95**

The Computer in the School: Tutor, Tool, Tutee Well reviewed collection of 19 seminal articles by five educational computing pioneers. 280pp ☐ **$14.95**

Fifty BASIC Exercise Learning problems in BASIC, chosen for their educational value, that would well supplement any HS+ intro programming course. Sequential in difficulty. 280pp ☐ **$12.95**

Golden Delicious Games for the Apple Readers learn how to use games as educational tools and create their own. Yummy! ☐ **$12.95**

Introduction to TRS-80 and Computer Programming Good text for Level II BASIC, with over 200 exercises and solutions, flowcharted illustrations, etc. 300pp ☐ **$10.95**

Katie and The Computer Delightfully illustrated, charmingly told tale of a journey into a computer. 42pp hardbound ☐ **$8.95**

Column Total ☐

Microcomputers and The Three Rs Highly recommended by CUE. Describes micros and classroom uses, planning and implementing a school computer program, provides good syllabus for six-week pre-service course on BASIC. 170pp ☐ **$8.85**

Mindstorms: Children, Computers and Powerful Ideas Rapidly becoming a classic, this is a profound and eloquent look at the way children learn by LOGO pioneer Seymour Papert. 230pp ☐ **$12.95**

Nailing Jelly to a Tree Education-oriented guide to finding and using good software. 275pp ☐ **$12.95**

Peanut Butter and Jelly Guide to Computers Amusingly written consumer's guide/tutorial provides an exceptionally readable overview of personal computing. 225pp ☐ **$8.95**

PET/CBM Personal Computer Guide 2nd ed. The most complete guide to everything you'll ever need to know about your PET. Essential. 512pp ☐ **$15.00**

SIMPLE: Basic Programs for Business Application Excellent instructional guide for educators who want to teach their students (or themselves) how personal computers can help in business. 386pp ☐ **$8.95**

A Young Person's Guide to Computers Charmingly illustrated first look at computers (K-3) ☐ **$7.95**

Workbooks That Work!

Computers for Kids Highly recommended programming manual written just for kids—could even be self-teaching—with special section of teaching suggestions. Check desired version: ☐ Apple ☐ Atari ☐ TRS-80 73pp each. ☐ **$3.95**

Excellent pair of sequentially organized workbooks —and teacher's guide—contain hands-on and hands-off exercises developed by an elementary teacher through actual use. Gr. 4-6 ☐ Feed Me, I'm Your PET Computer ☐ Looking Good with Your PET ☐ **$4.95 (Each)** ☐ Teacher's PET ☐ **$4.00**

Getting Down to BASIC Clear and concise hands-on workbook introduces students in grades 7-10 to programming basics. Eight labs conveniently divide the material into manageable lessons. Well paced. Fills a real need for this grade range. Differences among Apple, PET, TRS-80 are noted. 44pp ☐ **$3.95**

Instant BASIC Active participation workbook by one of the language's founders covers a lot of ground, would make a good text for an intro HS-College course. 159pp ☐ **$10.95**

I Speak BASIC (student text) ☐ Apple ☐ PET ☐ TRS-80 ☐ **$5.95**

Column Total ☐

I Speak BASIC (teacher's manual) ☐ Apple ☐ PET ☐ TRS-80 New field-tested, machine-specific junior-high school programming course for students with no prior experience. ☐ **$12.95 (Each)**

1 Computer/30 Kids Middle-grade (4-6) workbook provides scores of BASIC exercises that can be done away from the computer. 52pp ☐ **$5.00**

1 Computer/30 Kids Teacher's edition. ☐ **$10.00**

The BASIC Workbook: Creative Techniques for beginning Programmers Carefully sequenced approach shows how much can be done with just 20 keywords. Good HS+ text. 128pp ☐ **$7.15**

Apple PASCAL: A Hands-On Approach Excellent guide to the powerful PASCAL language for people with little or no prior experience with computers or math. 432pp ☐ **$14.95**

Apple LOGO Read all about the famous MIT-developed programming language that may shatter your preconceptions about what young children can and can't learn. 240pp ☐ **$14.95**

Hands-On BASIC with a PET Emphasizing programming rather than theory, this comprehensive machine-specific text covers its subject clearly and effectively. HS+ 256pp ☐ **$18.95**

NOTE: School orders for 25 or more copies of the same title will receive a 20% quantity discount.

AT LAST . . . CATALOG CARD & LABEL WRITER!

If you've been wondering when "they" ever were going to computerize this thankless little chore, your wait is over. You'll need an Apple and an 80-column printer (Epson, Centronics, Micro-line, etc.) to really appreciate the time and effort this program will save you. You input the information *just once*—and the computer becomes a specialized word processor that automatically formats standard catalog card formats and labels, as many copies as you need. Available in Dewey or L.C. version. User-friendly. Well-documented. Will pay for itself. Note: also requires a Lower Case Adaptor.

Catalog Card & Label Writer *with* Lower Case Adaptor ..... ☐ **$199.00**

Catalog Card & Label Writer *with-out* Lower Case Adaptor ..... ☐ **$169.00**

Column Total	
Total	
Shipping	
Tax	
Grand Total	

NOTE: FOR MULTIPLE ORDERS OF THE SAME TITLE PLEASE USE A PURCHASE ORDER.

Ordering Information Use this page as a handy requisition form. Official school purchase orders will be billed at net 30 days. ***Personal orders must be prepaid.*** The minimum order is $15.00. Please include 5% for shipping (10% foreign). New Jersey residents please add 5% sales tax. All prices are subject to change without notice.

NOTE: FOR MULTIPLE ORDERS OF THE SAME TITLE PLEASE USE A PURCHASE ORDER.

Satisfaction Guaranteed!
You must be fully satisfied with all books ordered from K-12 MicroMedia or you may return your order within thirty days of receipt for a full and unconditional refund. Please write us prior to any return, including our invoice or packing slip number, to ensure proper credit.

K-12 MicroMedia/172 Broadway
Woodcliff Lk, NJ 07675 201 391-7555

Name _____ P.O. #_____

School_____

Address_____

City_____ State_____ Zip_____

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

The PAPER
Box 460
Livingston Manor, NY 12758

ADDRESS CORRECTION REQUESTED
PLEASE RETURN INTACT