THE PAPER

Volume IV, Issue 1 | A CSD PUBLICATION | $3.00

Table of Contents

++

Subscription Codes

Please check the one letter subscription code found after your subscriber number on the mailing label. It indicates your status as of August 15. The interpretation is as follows:

P - Paid - Thank you
B - Bill - This IS your bill
S - Survey - You did say 'YES!'
L - LIPS - A loyal member?
N - Non-renewed Volume 3 subsciber

If your status is not P for PAID, this is your **bill** for Volume 4. Sorry, we cannot afford to send out another reminder. We hope everyone enjoys this issue and looks forward to those remaining.

++

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 East Setauket, NY 11733 (Permit #96). 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
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 3 Issue 8/9/10

Printed:	1000	USA:	621	Canada:	49	Foreign:	47

Software

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

Staff:

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

Staff writers: Bill Batcher
JoAnn Comito
Gerry Eisner
Jim Fowler

This issue has been produced using WordPro III Plus, a Diablo 630 letter quality printer with a Prestige Elite 12 printwheel and a CmC ADA 1450 interface. The printer is courtesy of American Peripherals, a large local dealer. If anyone can obtain a letter quality printer at a bargain price, please let us know. The lack of our own printer is the weakest link in our production.

Here We Come Again

I have waited to write this column until the very LAST possible moment. Many things have happened and will happen in the next few months to both me and The PAPER. I have just moved 200 miles north of my old location to a new job and new house. My wife and I are proud to annouce the birth of our second son, Kurt Herman, at 8:26 AM on August 7. Kurt weighed 7 lbs. and 15 oz. and is 20 in. long. Mother and child are doing fine. I've heard it said that only one major change should be made at a time. I'll let you know how true that is soon. As I hope you can understand, this move has meant a change in mailing permits, printers, and bank accounts. Still, we will publish at least 6 issues in one year on the first of every other month. We are pushing for new subscribers starting with an ad in the October issue of COMPUTE.

I would like to comment on two topics in this issue. My thoughts on both subjects were stimulated by comments and questions raised by readers.

The PAPER and Commodore's Products

If all goes as planned Commodore will have several new products well on the way to becoming reality by the end of the year. The VIC-20 is already on the market and seems to be gaining acceptance by dealers and users alike. It certainly compares favorably with its competitors in price and performance. Several peripherals designed for the VIC are in the works including a disk drive and printer. Already available are some expansion cartridges to plug into that large open spot in the back of VIC with more planned. Commodore has prototypes of many other products and has not stepped in to stop rumors about others. The 2-megabyte version of the 8050 disk seems to be close at hand. The MicroMainFrame may be ready by December in small quantities if a major design change is not made. A single disk drive is rumored as is a color PET. The 64K RAM upgrade to make 8032 into 8096 should be ready soon. This does not even take into consideration all the PET compatible products and software available form other sources.

The PAPER will support all Commodore hardware and software products and all those produced by other manufacturers as long as those products are of interest to the readers. I would very much like to publish VIC-20 articles, features on programming in PASCAL and COMAL, and discussions of how to use the MMF with 96K and several different languages. It is unlikely I will see much of this equipment, so the articles will have to come from the readers.

You Can't Please

Doug and I have been trying to find a way to summarize the recent reader survey. Many people returned the survey with interesting comments. Here are a few of those comments.

I am VERY pleased with The PAPER. I would like to see more utility programs.

I would like to see more articles comparing BASIC 2 and BASIC 4 ROM changes.

I find the grassroots articles interesting. I enjoy the focus on assembly.

Complete table of contents GREAT on front cover.

The articles are FIRST RATE. I would like to see some applications programs for small business, science and engineering

A+ for the new print. I like useful subroutines, sorts, formatting, input, etc. Anything useful in the classroom.

New name would be advisable.

No need for new name.

Use The PET PAPER. I would like to see some articles on Commodore TCL/PASCAL.

The PAPER is by far the most useful publication for the PET. If I could only subscribe to one magazine this would be it.

Product reviews and PET tricks are why I subscribe.

I find most articles difficult to understand at my novice stage. Which local universities provide a decent learning program without enrolling for a degree?

Would appreciate any articles pertaining to income tax software.

I do hope this can go on. Ralph's series on ML is good-good-Good.

The PAPER is fantastic. It has been a great help in the past but the last two issues have increased my knowledge of programming 100%.

Pleased to see articles on machine language. I feel too many authors assume too much background on the part of the reader.

PET will drop small B+W computers if VIC catches on. Switching to ATARI.

Articles on how to use assemblers (especially MAE) please. Most ML articles assume that the instructions on an assembler are enough for a beginner.

Be sure to keep the hints and articles for us less advanced programmers.

This is the only periodical which addresses the user who is not a pro. You are the only ones who print and address my letters for help.

I would like to see more articles for the construction of external devices. Also articles on system architecture, internal registers, user port and the like.

How about more on adventure type games. More assembly programming.

I feel that THIS magazine has done more to help me than any other. I will encourage others to subscribe.

How to use ML routines in BASIC would be useful.

These comments represent many other similar ones made by other readers. As you can see most people had positive comments and good suggestions. Almost everyone who commented liked the letter quality typeface and said so. Many people commented that they wanted more machine language articles and we have included more. I have no desire to dedicate The PAPER to any one group of users. There is room for the very beginner and the most experienced programmer. However, I feel I can only write about topics I am interested in. Right now MY interest is leaning toward machine language and languages other than BASIC. We need articles of ALL kinds from YOU, the reader. I have had several encouraging offers from people who promised they would write articles. Please send articles and ask others to do so. Keep the cards and letters coming. From the comments you can plainly see 'You can please some of the people all of the time and all of the people some of the time but ...'. Well, you get the idea.

Reader I/O

The purpose of Reader I/O is to try to answer questions from readers that may be of interest to other readers. Comments which require no real answer will also be accepted. Remember, Reader I/O cannot exist without your letters. Please take the time to pass on comments, tips and questions.

Ralph - Certain programs from CURSOR use a machine language joystick/keyboard routine for player control of a moving dot. These programs; CANYON!, DEMON!, PICKUP, and NAB!; do not work correctly on BASIC 4.0. The routine's data is in lines 61210 to 61240 and is accessed via the USR command. One change should fix all four programs. Do you know the change? - Bill Batcher

Bill - I thought for sure CURSOR would have published the correction for this problem but I couldn't find it in the Notes. The ML routine is stored from $028B to $02D1 (651 to 721) in the first cassette buffer. I ran the program so that the data would be stored and then used Micromon to disassemble the code. The following code is found at the very end of the routine:

```
02CC 4C 78 D2  JMP $D278
02CF 4C 6D D2  JMP $D26D
```

This is a reference to the PET ROM routine that converts integers to floating point numbers. When Commodore changed from BASIC 1.0 to BASIC 2.0 they extensively changed page zero memory. The CURSOR programs adjust for this fact. When Commodore then went to BASIC 4.0 they left page zero almost intact but changed the addresses of many ROM routines. The code for BASIC 4.0 should read:

```
02CC 4C C7 C4  JMP $C4C7
02CF 4C BC C4  JMP $C4BC
```

This reflects the change in address of the integer to floating point ROM routine. I can hear you saying 'Well, that's very nice, but how do I fix the program?'. You could just change the data in line 61240 to the correct values for BASIC 4.0. The problem then is that it will only work on a BASIC 4.0 machine. The best thing to do is make the program 'smart' so that it will automatically adjust for ANY present version of BASIC. This is much easier than it may sound. Location 50003 contains a 0 in BASIC 1.0, a 1 in BASIC 2.0 and 160 in BASIC 4.0. Therefore adding the following line will solve ALL the problems.

```
61016 IFPEEK(50003)=160THENPOKE717,199:POKE718,196:
      POKE720,188:POKE721,196
```

I tried it out and it works for both the joystick and keypad. - Ralph

Ralph - We have found that a POKE used to speed up printing on some machin s may have DISASTOROUS results on the new CBMs with the 12 inch screens. POKE 59458,62 will cut the voltage to the VIA in half and reduce the raster on the screen so that it is concentrated in the middle third of the screen. This quickly burns the phosphors off the screen and may cause some other cicuit damage. DO NOT use this POKE. REMOVE it from all programs. On a brighter note we have been able to convert the 4032 with 12 inch screens now shipped by Commodore to 8032 with only a few changes on the main PC board and a substitute ROM. - Jim Mendenhall

Jim - Thanks very much for the warning and note on the 4032. I hope everyone reads this and avoids a potentially nasty surprise.

Ralph - I would like to exchange information about Visicalc on the PET with other Visicalc users. Visicalc has a lot of potential which can be realized if people exchange information and data. - Joe Spatafora

Joe - I agree that exchanging information about the uses of Visicalc could only improve its usefulness. I believe there is a user's exchange based in Westchester County, New York. Maybe some readers can help. Write to Joe at:

Joe Spatafora
1870 Sailfish Rd.
St Petersburg, FL 33707

Ralph - Computhink provided us with a very good DOS in their 800P2 disk sysytem. Their 1.6 meg is equally as good. It's still the only disk I ever wrote turnkey programs on. There are some very good programmers out there and I feel we should solicit more input. - JFM & Associates

JFM - Computhink seems to be the disk of preference in England. Some of the things that appear in Printout are really interesting. Keep in mind I will publish anything I receive that I feel will interest the readers.

Ralph - In the last issue the auto repeat program had an RTS mnemonic right near the beginning. This would appear to negate the rest of the program. How do you get passed the RTS at $034A? - John Lamb

John - This program uses interrupts which are explained in this issue. Briefly the bit of code between $033A and $034A tells the PET that it should execute the code from $034A on 60 times every second and then go do its normal updating routine. To activate the repeat key you type SYS 826 once and then the PET will constantly 'run' the program that begins at $034A.

Ralph - I don't normally write letters to the editor but this time I feel I should make myself heard. I would like to see The PAPER support the VIC with the same fine quality it has supported the PET. I have not used many word processing programs but I think Michael Riley's PAPERMATE 60 is one of the best PET programs I have ever used. It is the only program my wife has used in the 2 years I have owned the PET. The new PET adventures from Aardvark Technical Services do exist and are fair to good. They will probably produce more if there is enough response. Send inquiries to Aardvark Technical Services, 1690 Bolton, Walled Lake, MI 48088. - Eugene Smith

Eugene - As I stated in my comments in Here We Come..., The PAPER will support, with quality articles, any products that interest the readers. I agree that PAPERMATE is a superior product but, then, I have yet to get anything from AB Computers that wasn't.

Ralph - I am trying to put together a SUPER ML monitor program which combines the best of programs like Supermon, Extramon and Micromon. I would appreciate suggestions or help from any readers. I have started the project but need feedback. - Jim Yost

Jim - Sounds like a good project. I like the newest version of Micromon from ATUG. Like you, I have often wished that the best of several versions of a propgram could be combined in one. The ML Utilities chip from Competitive

Software does this nicely. Readers should send hints, tips and encouragement to:

Jim Yost
PO Box 556
Somerville, MA 02143

Ralph - Please note our user group in your upcoming issue as follows:

PET User Group of Westchester
Ben Meyer
PO Box 1280
White Plains, NY 10602
914-428-7872 or 201-327-7544

Ralph - If Commodore has truly turned over a new leaf in regards to software support, why haven't they sent out SARGON II? Hayden Publishing Co. initially offered it for sale, then finally withdrew it from their advertisements (although they still offer a version for APPLE). A telephone inquiry to Hayden brought forth the reply that the PET version had bugs in it and had been sent back to Commodore. That was many months ago! Do you know what's happening? - Robert Kingshill

Robert - Who told you Commodore had turned over a new leaf in regards to software? I don't think Commodore USA will ever really support a full line of software of any kind. Still, as far as I know, it's hard to lay this one at their door. I believe that SARGON II for the PET is an independent effort on the part of the original authors, the Spracklens, and Hayden. To my knowledge Commodore has nothing to do with it. I too called and got the 'It should be ready some time soon.' routine. This is another example of putting the advertising before the product. If anyone knows more about this or any other chess program for the PET, I am sure there are many of us who would like to know.

Ralph - The second issue of COMPUTE has a machine language version of heap sort that does not use strings. It POKEs the separate letters from the strings above the 'top of BASIC'. When strings are entered as input or from a file they are stored at the top of memory and work their way down. These strings are useless since their letters are immediately being POKEd into memory. Even so they remain and finally meet the letters coming up from the 'top of BASIC' limiting the sort to about 5000 characters. Now, if we POKE the 'top of BASIC' (address 42-43 in BASIC 4.0 and 2.0, 124-125 in BASIC 1.0) to about 200 bytes below the top of memory the strings are collected frequently and the sort will handle 25,755 characters. The letters are still POKEd above the old 'top of BASIC' but the strings come down only to the new 'top of BASIC'. - Jack Clark

Jack - These short notes from readers are very useful. The more we communicate the more we learn from others. I believe that the program shown in the article works only for BASIC 2.0. Someone might want to try conversion to BASIC 4.0.

Many readers wrote about the problem of opening files in BASIC 4.0 with a variable for the file name. Specifically, DOPEN #1,A$ will NOT work. The proper way to solve this is simply DOPEN #1,(A$). Thanks Al Rosen, Jerry Key and Ben Meyer.

Background/Foreground

The purpose of Background/Foreground is to allow us, the editor and publisher, to make short editorial comments; announce new products, meetings and other PET related materials; and relay short tips, hints and comments we have gathered. We would greatly appreciate YOUR input and feedback on the issues we raise and the information we give.

Reviews

There is so much PET related hardware, software, firmware and courseware being produced now it is hard to know what to buy. Even if you know the product exists it is hard to know if it is worth the price and will perform as advertised. The purpose of reviews should be to help guide users in purchasing these products.

I have read many reviews that are simply summaries of the manuals provided with the product. A review should reflect the reviewer's actual experience with the program over an extended time period. One afternoon's hacking can hardly reveal all the advanatges and disadvantages of a product. When I review a product, I will have used that product in an actual application for several weeks. This makes for slower but more complete reviewing.

We tend not to give too much space to new product announcements since they are simply unpaid advertisements. We also tend to only publish good reviews since space is limited. If a user finds a product or company is so bad that he feels he should warn others, we will publish that review. There is little room to review mediocre products.

It would be impossible for us to buy even one-tenth of the PET products available. We, therefore, rely on readers to send us product reviews and on suppliers to send us review copies of products. So far, many companies have sent us products to review. We do not promise to publish a good review, just a review. The following companies have contributed products for review and these reviews will appear as space and time permit.

Product	Company
HESEDIT, HESBAL	Human Engineered Software
ROM Rabbit, ASSM/TED	Eastern House
VIGIL, Tiny PASCAL PLUS	Abacus Software
SWARM 100	Batteries Included
CALC	MATRIX Software
Disk-O-Pro, Command-O, PicChip, Mikro	Skyles Electric Works
Flexfile, Supersort, KMMM Pascal, EARL	AB Computers
Valdez, Flight Simulator	Dynacomp
ML Utilities, School PAC, BASIC Utilities	Competitive Software
Batter Up, MicroSail, various books	Hayden Book Company
Jinsam 4.0	Jini Micro Systems
Create-A-Base	Micro Computer Industries
Screen Pro, Chipmate, Triple Flip, UtiliRom	Kansas City Computers

Again, thank you to these people for supplying products for review. We are at least sure these products exist and we will review them.

I hate to keep harping on one subject but I am constantly reminded of this one. Recently, I have had occasion to ask for review copies of several products and to buy others. Many times I have been told 'The manual is at the printers but should be ready any day.' or 'Oh, that won't be ready for some time yet.' Mind you, these products had been included in full page ads in prominent magazines. One company has started advertising, for sale, Commodore's MicroMainFrame, Color Computer, single disk and so on. These aren't even fully designed yet, let alone available to dealers.

Since I do produce software for sale, I realize some of the problems that arise. Plans for ads must be made months in advance. You leave yourself some leeway but the product always takes longer to develop than you think. By the time you realize that things will be delayed for several weeks or months you have committed ads for months in advance. I can only say that the buyer should either call or send a letter asking when delivery is expected. Unless you know the product exists, NEVER simply send an order. You may find you are financing product development and that delivery 'will be unavoidably delayed'.

Upgrading

I held out a long time before upgrading to BASIC 4.0 and DOS 2.1. I had a lot of programs I was afraid would not like the change. I knew of the faster garbage collection, auto-initialization, ability to use relative files and simplified disk operating commands, but felt I could get along without this. Foolish person! Finally, programs began to appear that required BASIC 4.0 and DOS 2.1 and I knew I had to make the plunge. I now have a PET which has BASIC 2.0 and BASIC 4.0. I can select one or the other with a short ML routine and it works just as if the other was not there. I installed zero insertion force sockets in my disk drive and can switch between the two DOSes in a few seconds. The cost was $125 for a ROM switching board, $24 for three ZIF sockets, $50 for BASIC 4.0 ROMs and $50 for DOS 2.1 for a total of about $250. Most dealers will upgrade both disk and PET free if you turn in your old ROMs.

After working with the new system for about a month, I wonder why I didn't convert immediately. Programming is easier with the new DOS commands and it is a joy not to have to wait until the PET decides to return from a garbage collection holiday. Also, most programs will work on BASIC 4.0 with little or no change. It is not really necessary to have two BASICs or two DOSes unless you have some particular reason in mind. Mine is that I develop software for sale and must be SURE it runs on all versions of PET BASIC.

I found that many major software packages such as WordPro and KMMM PASCAL will work on BASIC 4.0. For those that do not, almost every company has a policy which allows you to trade in your BASIC 2.0 version for a new BASIC 4.0. My MAE ($170) from Eastern House cost $20 and I only had to wait 5 days. IDPC said they would upgrade FULLFORTH+ ($65) for $8 but I am still waiting after almost a month. There are some problems. I cannot yet find a version of Commodore's TCL PASCAL that works on BASIC 4.0. I assume this may be a problem with certain programs. Still, all in all, upgrading is not that painful and has many more positive than negative points.

Visible Music Monitor Upgrade

I wanted to make this a separate note so as many people as possible see it. I called AB Computers when I found that VMM would not work on my recently upgraded BASIC 4.0 system. A very nice woman informed me that two changes were all that was need. After making the following changes simply resave the program.

```
POKE 1167,227 : POKE 6363,136
```

New Books from TIS

I suppose we all have different opinions about the usefulness of various manuals for the PET. I have always found that the manuals from Total Information Services, PO Box 921, Los Alamos, NM 87544, to be helpful. They have just announced the 3rd edition of their "Understanding Your PET/CBM, Volume1: Basic Programming". This book combines, under one cover, five different workbooks. It also includes all three ROM sets and a ROM summary chapter. New titles will be published soon. The price of this one is $14.95.

NYSAEDS Conference

On October 18-20, the New York State Association for Educational Data Systems will hold its annual conference in Syracuse, NY. NYSAEDS, an affiliate of AEDS, is composed of people who have a common interest in computers and education.

The theme of this year's conference is "Software". A variety of workshops will be presented concerning the uses of microcomputers in education. For more information contact: Don Ross, Ardsley HS, Ardsley, NY 10502.

Queue Educational Software Show

Queue's second annual Educational Software Symposium will be held October 17-18 at the Stouffer's Inn, White Plains, NY. The Symposium will feature educational software from dozens of publishers for review and purchase. There will also be a wide variety of seminars, panels, and user interest group meetings on such topics as designing and evaluating educational software, computers in the elementary classroom and various curriculum areas. Registration is $45 in advance or $55 at the door. For more information contact: Monica Kantrowitz, Queue Inc, 5 Chapel Hill Dr., Fairfeild, CT 06432.

Waterloo BASIC Upgrades

Waterloo BASIC is a 2716 EPROM which allows you to do some very nice structured programming with your own PET. Release 1 includes a version for BASIC 2.0 and BASIC 4.0. These versions operate identically from the user's point of view. Users who have the old chip and wish to upgrade for BASIC 4.0 may do so by sending the old chip and $35 to Sharon Malleck, Computer Systems Group, University of Waterloo, Waterloo, Ontario, N2L 3G1, CANADA.

There may be a Release 2 containing more features when certain bugs are eradicated. Look for it in the late fall when information on pricing will also be available.

Data Base Managers

I am trying to prepare an article which would be an overview of some of the major data base managers on the market. The point would be to evaluate them for specific jobs and to compare their various features. This is a big job since it means actually using them to see the good points and the problems that occur. I have been using JINSAM 4.0 and FlexFile extensively along with a couple of homebrew programs. If anyone has anything to say about these programs or has used any of the other managers, please drop me a note. I have found that most of these programs end up doing the same thing, sometimes in the same way. Differences appear in the ease of use, documentation, reliability and support by the company.

Formatting on the 2022/4022 Printer

by Ralph Bressler

Among the other bits of information ignored by many PET users is the fact that the 2022/4022 printer does some very nice formatting. If you only have a few things to format and want to do it only once then this may not be of vital importance. It's one of those things I ignored for a long time and I now realize how foolish I was. It is relatively easy to do the formatting, it always works and saves a lot of time and aggrevation. I think the printer manual does a pretty good job of explaining this but several people have asked me to explain it. Here goes!

I had occasion to print a list of the programs that I sell. I wanted to distribute the list to customers and, therefore, wanted things to look nice. I wanted to display the program number, name, a short description, disk number and price on one line on my printer. I wanted it to look something like this:

```
CSD0023  TRIANGLES       - the Pythagorean Theorem          4       $15
```

After a little experimentation and translation of the Commodore manual, I was able to achieve this. I decided I wanted the CSD to appear in front of all numbers followed by the program number with leading zeros, if necessry. This would be followed by 2 spaces, up to 15 places for the name, 30 places for a description, 3 spaces, 2 places for the disk number with no leading zeros, 6 spaces and, finally, 4 places for the price with a floating decimal point. I then wrote the program which included setting up the format data, sending it to the printer, reading and then printing the data in formatted form.

Here is the program and an explanation:

```
10 OPEN 1,4,1
20 OPEN 2,4,2
30 F$="[R]C[R]S[R]DZZZZ  AAAAAAAAAAAAAAA [R]- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
40 F$=F$+"   99      $$$$"
50 PRINT#2,F$ : CLOSE2
60 FOR I = 1 TO 3 : READ N,N$,D$,D,P
70 PRINT#1,N;N$;CHR$(29);D$;CHR$(29);D;P
80 NEXT I : CLOSE 1
90 DATA 32,CHEM OX,PRACTICE FORMULA WRITING,3,15
100 DATA 23,TRIANGLES,THE PYTHAGOREAN THEOREM,4,15
110 DATA 3,ADD 1-2,ADDITION DRILL,1,10
```

NOTE: The [R] means the single RVS ON character!

Line 10 opens file #1 to the printer (device #4) with a secondary address of 1. When data is sent to the printer via PRINT#, the secondary address controls what that data will mean to the printer. A simple OPEN 1,4 would mean that any data sent by PRINT#1 would be printed exactly as sent since the secondary address is 0. This is how we normally use the printer.

Line 20 opens file #2 to the printer with a secondary address of 2. This tells the printer that any data sent by PRINT#2 will be special data that will not actually be printed on paper but stored in the printer's RAM. After this any data sent via PRINT#1 will be printed according to the format.

Line 30 and 40 set up the format string that will be sent to the printer in line 50. F$ looks complex but really is quite simple. The first three characters C, S, and D are each preceded by RVS ON. These are literals and will be printed just as they are. Whenever a literal is printed each character must be preceded by RVS ON. The next 4 Z's are used to format a number. They mean that any number will be right justified and leading zeros will be printed. For example, 2 will

be printed as 0002 but 1489 will show up as 1489. The next 15 A's and the 30 A's after that format string or alpha data. Any leading blanks in a string are stripped off and the string is left justified with spaces padding it to the right. If the string is too long, longer than 15 characters in this case, the extra ones are dropped. The - between these two fields is a literal. The 99 formats numeric data like the Z's but no leading zeros are printed. The number is right justified and is preceded by blanks, if necessary. That means 4 appears as 4 and 89 would show up as 89. The four $ are, again, numeric but obviously have to do with money. They allow a maximum of three spaces for numbers and will print a $ in front of our price. Spaces in the format string will show up as spaces.

Line 50 sends the format string to be stored in the printer RAM memory and then closes the file.

Line 60 sets up a loop and reads the data from lines 90 to 110.

Line 70 sends the data to be printed as per the format stored in the printer. Notice that all string data must be followed by a CHR$(29) or right cursor to generate a space. It is best to separate the variables by using semicolons.

Line 80 completes the loop and then closes the file when everything is done.

This example should give you a good idea of how things work. If you read the manual now, I hope it will make more sense. The numeric format characters can be combined for any desired result. In addition, a numeric format can be preceded by an S which will print a blank if the number is positive or a - if it is negative. You can also follow numeric fields by a - which will cause a trailing minus sign for negative numbers. Decimal points may also be included and will be fixed where they appear. Let's try one more example! In this case let's say we have a part number, name, quantity on hand and price. We want it to look something like this:

```
7412.567 BW   Z80 MICROPROCESSOR                1479      $   12.57
```

Here we want a 9 digit part number with 6 digits before and three digits after the decimal point and a two letter suffix. We want 30 places for our name, 6 places for the quantity with no leading zeros but a trailing sign to show backorders and five places for the price with room for fractional parts of a dollar and a fixed decimal place. The format string for this would look like this:

```
999999.999 AA   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  99999-     $$$$$$.99
```

To send data to this format we might have a line like this:

```
PRINT#2,N;S$;CHR$(29);N$;CHR$(29);Q;P
```

Keep in mind that only one format may be stored in the printer's RAM at a time. However, it is easy to store one format, use it and then store another and use it. Once the format strings are set up there's no problem. This method is infinitely easier than trying to format data using TAB and SPC in BASIC and saves memory. The printer does not always handle these commands as you might expect and using them can be frustrating. With a little practice this is the easiest, most efficient way to neatly output data to the printer.

I would be glad to answer questions about particular problems or any points not well explained here. Special problems may require creative and unique solutions. This is the second article on the special features of the Commodore printers. The first part showed an easy way to use the programmable character and gave some interesting examples. The next installment will discuss some other features of the printer. Look for another series of articles about the Commodore disk in our effort to show the user how to exploit the various PET peripherals.

Card Shuffle

by Bill Batcher

The program below is a collection of three different ways to shuffle a deck of cards. Each takes a different approach and may be better for a specific situation. The timing (lines 370,530,660) should be regarded as only a suggestion. Factors that effect timing included whether or not the cards are printed, and the size of the 'M' loop in line 450. Here is an explanation of the methods.

Method 1 is found in many books and requires two arrays DIMensioned to the number of cards you are shuffling. The C$ array holds the cards while the FL array records whether a card has been selected or not; it's a flag. The 'M' loop in lines 300-360 selects a random number up to the size of the deck, checks if it was flagged (picked before), flags it if not previously chosen and prints it. This method gets bogged down near the end as the computer tries to locate a random number not yet chosen. This gets worse as the "deck" gets longer. Still, I suspect this might be the best way to shuffle strings of varying lengths such as words.

Method 2 comes from Kathy Weston at Harborfields HS. It requires only one array, C$() since no flags are used. The 'M' loop in lines 450-500 selects two different random numbers up to the size of the deck, pulls out both cards and exchanges them in the deck. The limit of the loop, in this case 200, is arbitrary but 200 seems adequate. This method consistently takes the same amount of time depending on the upper limit of the 'M' loop. I expect this method works best with small decks, say around 52. The larger the deck the longer the 'M' loop should be.

Method 3 came from Don Coscia. It uses no arrays so no DIMensions are necessary. Line 580 constructs one string, D$, for the entire deck. The loop in 600-650 pulls one card out of D$, prints it and then redefines D$ as all of itself minus the chosen card. The upper limit of the random number is the number of cards left.

Notice in this case we have used a deck of 70 cards.

```
 10 REM CARD SHUFFLES 3
 20 REM BILL BATCHER
 30 :
100 REM THE DECK
110 DATA "*A","*B","*C","*D","*E","*F","*G","*H","*I","*J"
120 DATA "@A","@B","@C","@D","@E","@F","@G","@H","@I","@J"
130 DATA "%A","%B","%C","%D","%E","%F","%G","%H","%I","%J"
140 DATA "#A","#B","#C","#D","#E","#F","#G","#H","#I","#J"
150 DATA "!A","!B","!C","!D","!E","!F","!G","!H","!I","!J"
160 DATA "+A","+B","+C","+D","+E","+F","+G","+H","+I","+J"
170 DATA "&A","&B","&C","&D","&E","&F","&G","&H","&I","&J"
180 :
190 PRINT"(clr)";: FOR X=1 TO 7: FOR Y=1 TO 10:READ G$
200 PRINT" (rvs)"G$"(off)";:NEXT Y: PRINT: PRINT: NEXT X
210 PRINT"HERE ARE 70 DIFFERENT CARDS. WE WILL TRY"
220 PRINT"SHUFFLING THEM IN THREE DIFFERENT WAYS."
230 GOSUB3000
240 :
250 PRINT"(clr)METHOD #1"
260 DIM C$(70), FL(70)
270 RESTORE
280 FOR Z=1 TO 70: READ C$(Z): NEXT
290 TT=TI
300 FOR M=1 TO 70
```

```
 310 :V=INT(RND(TI)*70)+1
 320 :IF FL(V)=1 THEN 310
 330 :FL(V)=1
 340 :PRINT" (rvs)"C$(V)"(off)";
 350 :L=L+1: IF L=10 THEN PRINT: PRINT: L=0
 360 NEXT M
 370 PRINT: PRINT"FINISHED IN"(TI-TT)/60"SECONDS"
 380 GOSUB3000: CLR
 390 :
 400 PRINT "(clr)METHOD #2"
 410 DIM C$(70)
 420 RESTORE
 430 FOR Z=1 TO 70: READ C$(Z): NEXT
 440 TT=TI
 450 FOR M=1 TO 200
 460 :V1=INT(RND(TI)*70)+1
 470 :V2=INT(RND(TI)*70)+1
 480 IF V1=V2 THEN 460
 490 :T$=C$(V1): C$(V1)=C$(V2): C$(V2)=T$
 500 NEXT M
 510 FOR X=0 TO 6: FOR Y=1 TO 10: PRINT" (rvs)";
 520 PRINT C$((X*10)+Y)"(off)";: NEXT Y: PRINT: PRINT: NEXT X
 530 PRINT: PRINT"FINISHED IN"(TI-TT)/60"SECONDS"
 540 GOSUB3000: CLR
 550 :
 560 PPINT "(clr) METHOD #3"
 570 RESTORE
 580 FOR Z=1 TO 70: READ C$: D$=D$+C$: NEXT
 590 TT=TI
 600 FOR M=1 TO 70
 610 :V=INT(RND(TI)*(LEN(D$)/2))+1
 620 :PRINT " (rvs)"MID$(D$,(V*2)-1,2)"(off)";
 630 :L=L+1: IF L=13 THEN PRINT: PRINT: L=0
 640 :D$=LEFT$(D$,((V*2)-2))+RIGHT$(D$,LEN(D$)-(V+2))
 650 NEXT M
 660 PRINT: PRINT"FINISHED IN"(TI-TT)/60"SECONDS"
 670 END
 680 :
3000 PRINT "PRESS (rvs)SPACE(off) TO CONTINUE"
3010 GET A$: IF A$="" THEN 3010
3020 RETURN
```

****

Volume 3 Back Issues

There are a limited number of full sets of Volume 3 of The PAPER available. This includes 5 seperate issues and many pages of worthwhile information. We really can't use them ourselves so would like to see someone get them who can. If you are a new subscriber or don't have a full set of Volume 3, why not send for one TODAY!

$10.00 per Volume

****

Screen Scroll Prevention

by Wayne Mesard

In our high school we have two 32K PETs with BASIC 2.0 connected to one 2040 disk drive. Even if students are careful not to access the disk at the same time scrolling is a problem. A bug in the IEEE routines causes files to close and other nasty things to happen when screen scrolling occurs. I, therefore, began to write a small machine language routine to solve this problem. I used Micromon and the HESEDIT and HESBAL programs to write the routine.

A first version of the program as shown below simply homed the cursor when it reached the lower line of the screen. The routine is interrupt driven and will NOT be compatible with other programs that use interrupts.

BASIC 4.0 version

```
0392 78          SEI        ;disables interrupt to set new interrupt vector
0393 A9 03       LDA #$03   ;sets HIGH part of new vector
0395 85 91       STA $91    ;...to $03
0397 A9 9D       LDA #$9D   ;sets LOW part of new vector
0399 85 90       STA $90    ;...to $9D
039B 58          CLI        ;enables interrupt then
039C 60          RTS        ;returns to BASIC
039D A5 D8       LDA $D8    ;loads acc with present line position of the cursor
039F C9 18       CMP #$17   ;checks to see if cursor on line 23
03A1 F0 03       BEQ $03A6  ;...if is then goes to run short antiscroll routine
03A3 AC 55 E4    JMP $E455  ;...if NOT then jumps to PET's interrupt routine
03A6 A9 01       LDA #1     ;loads acc with 1 which when
0348 85 D8       STA $D8    ;...stored moves cursor to first screen line
03AA 4C 55 E4    JMP $E455  ;jumps to PET's interrupt routine and to BASIC
```

BASIC 2.0 (changes)

```
03A3 4C 2E E6    JMP $E62E
03AA 4C 2E E6    JMP $E62E
```

The program needs extensive changes for BASIC 1.0 and I could not find certain key locations.

This simple program can be used any time you want to prevent screen scrolling. It will work in a BASIC program or in immediate mode. To initialize the routine load it up, type NEW and then SYS 914. It will then work to prevent scrolling until a tape I/O operation is performed or some other interrupt program interferes with it. You may use the ML monitor to type it in and then SAVE it. This is fine if you want to just use it to prevent the IEEE problems. However, to include it in a BASIC program it is probably better to include the routine in DATA statements. The DATA must include the begining and ending address and the entire ML program all in decimal. The routine for the program above would look like this:

```
100 READ B,E : FOR I=B TO E
110 READ N : POKE I,N : NEXT
120 SYS 914
130 DATA 914,940,120,169,3,133,145,169,157,133,144,88,96,165
140 DATA 216,201,24,240,3,76,85,228,169,1,133,216,76,85,228
```

For BASIC 2.0 change the underlined numbers to 46 and 224. You could send the cursor to any line by changing the 1 at $03A7 (decimal 935) to the number that represents the line you want the cursor on. You could do this by changing the 1

in DATA line 140 above. The program can be relocated to any place in memory but the interrupt set up at $0393 and $0397 must be changed so that it points to your program.

The program works in the following way: (If you are not familiar with interrupts read the article 'Calling All Machine Language Programs' in this issue.) Locations $0392 to $039C set the interrupt vector to point to our short routine that starts at $039D. I disable the interrupt first so that it will not try to update anything while I am changing the vector. Before returning to BASIC with the RTS the interrupt is enabled with a CLI. The interrupt vector is stored at $90 - $91 in typical 6502 low-high form. Locations $039D to $03AC contain the routine that will be executed before the PET's regular interrupt routine. As shown in the comments it checks to see if we are on the 23rd screen line ($039D to $039F). If we are NOT, it jumps to PET's interrupt routine at $E455 ($E62E in BASIC 2.0) and thence to BASIC. If we are on the 23rd line, then we branch to $03A6, tell the PET to store 1 at the 'currect cursor line position' (loaction $D8) and THEN jump to the PET's interrupt routine. I could have included many other actions in the routine at $03A6.

The problem with this program is that during a program listing the cursor simply homes and overwrites what was printed on the screen. To solve this I attempted to cause a halt in the program when printing occurred on the next to last screen line. I wanted the user to be able to examine the screen and then hit any key. The screen would clear and printing would again start at the top. This is where the problems began. I tried several different methods such as sensing a key held down and looking for an entry in the keyboard buffer and NOTHING seemed to work. I found out that when using your own interrupt driven routine page zero is not updated by the PET's normal interrupt routine since it is locked out. All the locations and methods I was using needed the page zero update. As you will see in the program below I redirect the interrupt to its normal location before looking to see if a key is being pressed. This allows me to use the page zero locations I need. I then reset it to my own program and procede to take action as required. I set up the interrupt vectors in the same way I did before in locations $0392 to 039C. I then proceded with my own routine at $039D as shown below.

BASIC 4.0 version

```
039D A5 D8      LDA $D8   ;loads acc with present line position of the cursor
039F C9 18      CMP #$17  ;checks to see if cursor on line 23
03A1 F0 03      BEQ $03A6 ;...if is then goes to run short antiscroll routine
03A3 AC 55 E4   JMP $E455 ;...if NOT then jumps to PET's interrupt routine
03A6 78         SEI       ;disables interrupt to set interrupt vector
03A7 A9 E4      LDA #$E4  ;sets interrupt vector back
03A9 85 91      STA $91   ;...to point
03AB A9 55      LDA #$55  ;...at the PET's own
03AD 85 90      STA $90   ;...interrupt routine
03AF 58         CLI       ;enables interrupt then
03B0 A9 00      LDA #00   ;zeros # of keys in the
03B2 85 9E      STA $9E   ;...keyboard buffer
03B4 A5 9E      LDA $9E   ;looks at the keyboard buffer
03B6 F0 FC      BEQ $03B4 ;...and waits until some key is pressed
03B8 20 4B E0   JSR $E04B ;jumps to PET's clear screen routine
03BB 78         SEI       ;disables interrupt to set new interrupt vector
03BC A9 03      LDA #$03  ;sets HIGH part of new vector
03BE 85 91      STA $91   ;...to $03
03C0 A9 9D      LDA #$9D  ;sets LOW part of new vector
03C2 85 90      STA $90   ;...to $9D
03C4 58         CLI       ;enables interrupt then
03C5 4C 55 E4   JMP $E455 ;jumps to PET's interrupt routine and then to BASIC
```

```
03A3 4C 2E E6   JMP $E62E
03B8 20 29 E2   JSR $E229
03C5 4C 2E E6   JMP $E62E
```

Again, you can type this in and save it using the ML monitor or you may turn it into decimal DATA and put it in a BASIC program.

This program checks to see if we are at the bottom of the screen (locations $039D to $03A0) and returns to BASIC after doing PET's normal interrupt routine at $E455 ($E26E in BASIC 2.0). If we are at the bottom of the screen, we set the interrupt vector back to PET's normal routine (locations $03A6 to $03AF) and then wait until a key is pressed (locations $03B0 to $03B7). When a key is struck we jump to the clear screen ($E04B in BASIC 4.0 and $E229 in BASIC 2.0) and reset the interrupt vector back to our program ($03BB to $03C4). Finally, we jump to PET's interrupt routine and back to BASIC.

I am not claiming this is the greatest application in the world for interrupts. It does begin to illustrate how a solution to a problem evolves. I have tried to explain my procedures and programs as much as possible. If there are any questions of comments I would be glad to hear them. You may use the routine from the first program by using the short BASIC program even if you don't understand much of the explanation of how things work. Any readers who have other applications should come forth and submit them to The PAPER for publication.

****

Writing for The PAPER

It should be clear by now that we feel communication between various PET users is important. Readers appreciate the mix of articles produced when a variety of authors write. Some people have asked for guidelines so here are three.

1) Write about something interesting. This means you can write about anything since some group will like what you write.

2) Write about something you know. An interesting topic can be made boring by an uninformed author. When you have really struggled with a topic you learn it inside out. These articles are many times the best.

3) Write with as few assumptions about what your readers know as possible. It is better to explain in detail than to assume everyone understands a critical point.

Many readers have asked for the addresses and phone numbers of authors. Please include these with your article but indicate if they are to be released or not. If you do not want them distributed we will keep a file so that we may contact you in the future. We will soon be sending The PAPER to all other na onal magazines to see if we can improve the coverage of the PET. We will be inviting these magazines to contact our authors regarding republication of their articles. If you do not want your article considered for reprinting, please say so. If you might be interested, indicate the conditions placed on reprinting. We will make sure that any publication that may reprint an article contacts the author first. We hope this will generate more interest in the PET, improve coverage of Commodore products and bring more recognition to our authors. We also hope more people will write for The PAPER.

****

Contest

The PAPER is pleased to announce three contests in which our readers can win valuable software packages. These contests will start September 1 and end December 1. The winners will be announced in the Januaury 1 issue. The contests are as follows.

Contest 1 - Reader Recruitment

Sure you want to do your PET user friends a big favor! You also want to help out the struggling staff of The PAPER! Would you like to do BOTH and receive a prize for doing so? The reader who gets the most NEW subscribers to take The PAPER will receive as a reward a free software package. All you have to do is have readers who did not subscribe to Volume 3 send in their $15 and mention your name and subscriber number. We will do the rest. Remember this is really a chance to promote the PET and help out The PAPER.

Contest 2 - Companies

A computer cannot exist long without companies that support it. This includes dealers, software houses and manufacturers of hardware. Fortunately, the PET has good support from many companies, if not always from Commodore. Let's try to do our part to reward these people. Send your nomination for the best PET-related company now! Use a standard post card and include your name and subscriber number, the name and address of the business, their major interest and a brief comment. Only subscribers may participate. Only one nomination per subscriber!

The business with the most nominations will receive a small plaque to acknowledge their support, as well as, a feature article in The PAPER. All readers who sent in nominations will have a chance to win a software package. This is not limited to national companies. Smaller local dealers are just as important.

Contest 3 - Best PET Program

There are many programs out there for the PET. We all know that some are terrible, some good and some great. Which program do you think is best? The same rules apply here as in contest 2. List the three top PET programs by name and include the name of the programmer or company and a brief comment about the program. Be sure to include your name and number.

The program listed first will get three points, second two and last one. After tabulation of the results a certificate will be sent to the winner and all those sending in nominations will have a chance to win another prize. This contest is not limited to programs from 1981 but is for ALL PET programs available to date. Programs do not have to be commercial but may be available on exchange.

The prizes are no joke! The winners will be able to CHOOSE from such packages as:

Command-O	Tiny PASCAL	Flexfile	Disk-O-Pro
Matrix Sort	Vigil	IDPC Forth	ML Utilities
BASIC Utilities	KMMM PASCAL	Papermate	HESBAL

and many more. Please don't pass this up since at least 100 people must enter each contest to make it valid.

Using Long Strings

by Bill Batcher

Many of us learned our BASIC in the early days on a PDP-8 or similar computer. String variables, we discovered, were a powerful and versatile tool. The fact that the length of these strings was limited to six characters was just something we accepted like the need for an END statement or a closed quote after every one that was opened. We learned to handle words with more than 6 characters by concatenating strings (PRINT A$;B$). When we weren't sure of the length of a word, as in an input, we became pretty expert at guessing the length of the longest probable answers, and then dividing that estimate by six. If we had a long list of words, we made sure we put enough strings into the READ command to handle the longest word in the DATA lines. Of course, if the longest word had 25 characters, that meant that every line item of data had to use 5 strings. A three letter word had to be padded out with 4 null strings. But then as I said, I didn't know any better. This was just the way things were.

Then along came the PET and other microcomputers with Microsoft or some other fancy BASIC. Through reading or accident we found we could use strings with more than 6 characters. There are two things a programmer does whenever he discovers a hitherto unrecognized feature:

1) He tries to push the feature to its limit.
2) He tries to find an application for the feature.

So when I first realized I could define long strings, I wanted to find out how long? It was easy to see that I could at least make the string fill out the rest of the statement, and a statement can fill two lines on the PET screen. (It can actually fill more, but thats another article.) But could I make an even longer string, say more than 80 characters? To find out, I built a little test loop:

```
10 A$=A$+"A"
20 PRINT "(home)"A$;LEN(A$)
30 GOTO 10
```

If there was a maximum length for A$, I figured, something would happen when it was exceeded in line 10. Sure enough, a RUN verified that a limit did indeed exist, but that limit seemed extremely gratious to someone who had had to wrestle with 6 character strings. The limit was 255 characters!. (This causes one to suspect that a byte somewhere in memory is measuring the length of strings.)

I could define strings that were 255 characters long. WOW! But why would I want to? You can't write a string that long on one statement line. You have to build the string by concatenating smaller portions. Would this feature be of any practical value?

At the time of my discovery, I was preparing a straightforward dri_l on plurals. I had assembled a large data bank of nouns, cleverly coded so that PET knew what the correct plural (P$) of each was. I had the mechanics all written to select 10 random words, ask for and test inputs and give an option of going on to do 10 more, and 10 more and so on. Nothing really fancy.

Whenever the pupil wanted to (or when the teacher allowed him to) quit, I figured I'd have the PET give pupil and teacher one last look at all his errors. This meant storing all the missed words in an array until the end of the program, where a loop would print them all. The lines would go something like this:

```
140 INPUT I$
150 IF I$=P$ THEN PRINT "GOOD":GOTO100
155 PRINT "SORRY, THE ANSWER IS "P$: E=E+1: E$(E)=P$: GOTO100
900 FOR A = 1 TO E
910 PRINT E$(A),
920 NEXT A
930 END
```

As long as I could be sure the pupil would not make more than 10 mistakes, I was OK. But once E exceeeded 10, a wrong input would yield a BAD SUBSCRIPT ERROR IN LINE 155. Of course, I could dimension E$(E), but for what value? 20? 30? 40? How many errors would the worst student make? It seemed wasteful to me to reserve a lot of space for a subscripted variable that would never be used. This would certainly limit my growing data base of nouns.

But AHHA! I remembered that a string can be longer than one word. It could be 255 characters long. There's enough room on one string to hold a couple dozen of my average plurals. And using just a few subscripts would certainly cover even the mistakes made by the worst pluralizer.

I needed to put in a delimiter so the PET would be able to seperate the strings later into the individual words. For this I chose the asterisk (*). Now the lines of my program looked like this:

```
140 INPUT I$
150 IF I$=P$ THEN PRINT "GOOD": GOTO 100
155 PRINT "SORRY, THE ANSWER IS "P$
160 IF LEN(E$(E))+LEN(P$) > 253 THEN E=E+1
170 E$(E)=E$(E)+"*"+P$
180 GOTO 100
900 FOR A = 0 TO E
905 L=LEN(E$(A))
910 FOR B = 2 TO L
920 IF MID$(E$(A),B,1) <> "*" THE NEXT B
930 PRINT LEFT$(E$(A),B-1)
940 E$(A)=RIGHT$(E$(A),1+L-B)
950 IF E$(A)="" THEN 970
960 GOTO 905
970 NEXT A
980 END
```

The asterisk, as it turned out, got printed with the word and that looked kind of cool, so I left it. Since then, I found other times when a long string, built up in a program, could efficiently keep track of a lot of information for me.

****

LOAD and RUN from Tape

Remember how you used to be able to hit shift RUN/STOP and load and run a PET program? With BASIC 4.0 and a disk drive the first program on drive 0 will LOAD and RUN, otherwise you get ?DEVICE NOT PRESENT. Try typing LOAD "NAME": and THEN hit shifted RUN/STOP. This will solve the problem and will work on any PET.

Directories on BASIC 4.0

In BASIC 4.0 you can use DIRECTORY or CATALOG to see what files are on a disk. For example, CATALOG D0 will list everything on disk 0. To get a directory of just certain files is another matter. If you want just those files beginning with COMA then you will have to type LOAD"$0:COMA*",8 or use the wedge.

Calling All ML Programs

by Ralph Bressler

Many people have asked for more information on machine language programming. Many of these people have inquired about how to 'connect' BASIC and machine language programs. My knowledge of machine language is limited and someone else should probably write this article. Nobody else has volunteered!

There are four different ways to call a machine language program from BASIC. Each has its own way of doing things with its own advantages and disadvantages. These methods include 1) SYS, 2) USR, 3) Interrupts, and 4) CHRGET. I hope to briefly explain all four of these methods and give brief examples of the first two in this article. A future article, hopefully by someone more knowledgable than I, will deal in depth with the last two methods. CHRGET is particularly tricky at times and needs some real planning.

SYS

SYS is a command that is used to access a machine language program stored somewhere in memory. The program will only run when it is called and will return to BASIC, if you haven't made any errors, when it is done. It can be used in the direct mode or in a BASIC program. For example, SYS 826 would run the ML program starting at decimal address 826 ($033A). SYS is useful when you have several routines in memory at one time and wish to choose one when running a program.

There are many places to store machine language programs, some better than others. I do not want to discuss the relative merits of each choice in this article. I will say that the second cassette buffer starting at decimal 826 or hex $033A has been a favorite place. On machines with BASIC 4.0 AND a disk drive a problem occurs. BASIC 4.0 uses the lower locations in this buffer for disk operations. It seems you are safe if you use locations starting at $035A, if you don't have a disk, or if you avoid the new disk commands in BASIC 4.0.

The following program is a very simple program which will fill the screen with any character.

```
0392 A2 00      LDX #0       ;zero the X register to use as a counter
0394 A9 A0      LDA #160     ;load acc with ASCII of fill character
0396 9D 00 80   STA $8000,X  ;store acc at $8000 plus x register
0399 9D 00 81   STA $8100,X  ;...this will fill the screen by
039C 9D 00 82   STA $8200,X  ;...storing the same character in
039F 9D 00 83   STA $8300,X  ;...4 quarters of the screen
03A2 E8         INX          ;bump the X register, used as counter
03A3 D0 F1      BNE $0396    ;branch back to loop if quarters not full
03A5 60         RTS          ;when done go back to BASIC
```

As is this program should work on any PET and will fill the screen with reverse spaces. This is because of the 160 at location $0395 (917 decimal). To use a different character you could change the contents of $0395 with the ML mo itor. An easier way might be to use POKE to change this location. For example,

```
100 POKE 917,42 : SYS 914
```

would fill the screen with *.

You may want to use the ML monitor to enter this program and then save it. The problem is that this makes it difficult, I think, to use it with a BASIC program. It is much better to turn the ML program into decimal numbers, store it in the BASIC program in DATA statements and the READ the DATA and POKE it into place. For the routine shown above it would look something like this:

```
10 FOR I = 914 TO 933 : READ N : POKE I,N : NEXT
100 DATA 162,0,169,160,157,0,128,157,0,129,157,0,130,157,0,131,232,208,241,96
```

After doing this you can call the routine at any time as shown above.

USR

USR is a function NOT a command. You might say something like Y=USR(X). This would cause the machine language program whose beginning address is stored at locations 1 and 2 to be executed. You would have to make sure you placed the correct address in these locations with the high byte at 2 and the low at 1. The argument of the USR function, X in the above example, would be placed in the 'floating point accumulator'. Your program may or may not use this value. When your program is through it should return to BASIC and it may leave another value in the floating point accumulator. If it leaves a value, that value will also be found in the variable to the left of the USR function. It is best to 'look' at the value this way rather than try to PEEK at the floating point accumulator.

The program will execute only when it is called as with SYS. In this case we can pass a value from BASIC to the machine language program. An example of this type of program is given below. The PET does not have a LOG base 10 function so this program uses a combination of ML and BASIC to provide logs of number in base 10. This program is substantially the same as one presented in Abacus Software's excellent 'Machine Language Guide'. The BASIC part is the same for ALL PETs but note that the ML program differs some.

```
10 POKE 1,146 : REM SETS BEGINNING ADDRESS OF ML PROGRAM
20 POKE 2,3   : REM TO $0392
30 INPUT X    : REM ENTER NUMBER WE WANT LOG BASE 10 OF
40 Y=USR(X)   : REM CALL ML PROGRAM AND PASS ARGUEMENT X TOPROGRAM
50 PRINT "LOG  OF"X"IS"Y
```

BASIC 4.0

```
0392 20 20 CB   JSR $CB20  ;call natural LOG of FACC
0395 A9 A6      LDA #$A6   ;point to constant for conversion
0397 A0 03      LDY #$03   ;..using A and Y registers
0399 20 C2 CB   JSR $CBC2  ;move constant to AFAC
03A2 20 5F CB   JSR $CB5F  ;multiply FACC by AFAC
03A5 60         RTS        ;return to BASIC
03A6 7F 5E 5B              ;constant for log conversion in
03A9 D8 9A                 ;...floating point form
```

BASIC 2.0

```
0392 20 F6 D8   JSR $D8F6  ;call natural LOG of FACC

0399 20 98 D9   JSR $D998  ;move constant to AFAC
03A2 20 3C D9   JSR $D93C  ;multiply FACC by AFAC
```

BASIC 1.0

```
0392 20 BF D8   JSR $D8BF  ;call natural LOG of FACC

0399 20 5E D9   JSR $D95E  ;move constant to AFAC
03A2 20 00 D9   JSR $D900  ;multiply FACC by AFAC
```

You may want to convert the program to BASIC data as we did above.

Interrupts

You push a single key on the PET's keyboard and suddenly things start to happen that aren't in your BASIC program. You don't have any SYS or USR statements in your program. The most outstanding example is KEYPRINT. In this program you just hit the backslash and the entire PET screen is 'dumped' to your printer. When this is done you get control of your printer and PET back. This is done using an interrupt program.

The PET normally suspends trading every 60th of a second to run a special routine of its own. This routine updates the clocks, tests the keyboard and does many other vital functions. It interrupts whatever else is going on, hence the name. If you could add your own program to this procedure it would be executed along with the PET's normal routine at 60 times a second. Basically, you must tell the PET to do your stuff first and then its own work. To do this you set the 'hardware interrupt vector' at $90 and $91 in BASIC 4 and BASIC 2 ($0219, $021A in BASIC 1) to point to the beginning address of your program. At the end of your program you must jump to the location of the PET's own interrupt routine. Before making this change you must disable the interrupts by using SEI. If you do not do this before changing the interrupt vector the interrupt routine just may, well, interrupt, and find half a new address and half an old one and decide to close up shop. CLI at the end of the machine language program restores the interrupts.

Remember that you now have a program which will be executed all the time. However, it will only go ahead to do what you want when some condition is met. Usually this means pressing a certain key. Space Invaders works on interrupts. I've seen students insert a BRK (op code 00) in the middle of the program to stop it. They then type something that includes an A, 4 or 6 and watch the same actions occur as if they were playing the game. It would take too much space and time to give an example here. To begin to understand this technique see the short article on KEYPRINT and the article entitled 'Screen Scroll Prevention'.

CHRGET

This is, for me, the hardest method to use and to explain. This is sometimes called a 'wedge' since it elbows its way into BASIC and just sits there. It watches what you type and acts accordingly. In most cases it can be used to add new commands to BASIC as is done with BASIC AID and the Toolkit. These commands can be single letters or entire words. A BASIC routine called CHRGET is called everytime BASIC needs a character from your BASIC program.

You employ the method by placing a jump to your own program at the beginning of the CHRGET routine. Your program must then replace the code you destroyed, execute your own tasks and then jump back to the CHRGET routine. For a BASIC discussion of this see the article 'The Wonderful Wedge' by Jim Butterfield in COMPUTE, April 1981. In the next issue we will include a more detailed explanation and an example of this method.

****

Tips and Hints

Articles never seem to run a whole number of pages. There are many short comments which are very important but do not deserve an entire article. These two facts mean that we need short tips and hints. You may think everyone knows what you just discovered but this is probably not true. Send us short comments for us to publish and we will if we haven't heard them.

****

COMAL and the COMAL Users Group

by Ralph Bressler

A new language is now available free for PET/CBM systems. COMAL is a structred language with many of the powerful structures found in PASCAL but it is easy to learn like BASIC. The language was developed in Denmark by Borge ohristensen in an effort to make BASIC programs more readable with an easy to follow logic flow. Christensen and his associates noticed that students learned BASIC quickly but developed bad habits that made larger programs a nightmare to debug. BASIC actually forced confusing structures by using GOTOs which interrupted the flow of the program. Confusion was furthered because variable names were too short to really show what they represented. BASIC did have advantages such as a dynamic editor, interactive mode, and easy to use input/output statements. So Christensen decided to combine the best parts of BASIC and PASCAL. The result was COMAL for a NOVA minicomputer that was later adapted for the Z-80.

After more work in Denmark and England, COMAL was adapted for the PET. In May 1981, Commodore UK announced they would release COMAL into the public domain and distribute it free. Commodore US promptly announced they would not support COMAL. Up stepped Len Lindsay to organize the COMAL User's Group to support COMAL here. Although the COMAL interpreter is free from Commodore UK, the documentation is sketchy. Len has put together a COMAL Starter Kit which is slowly taking shape. The Kit includes:

Commodore COMAL Interpreter
Commodore COMAL Manual
COMAL Handbook
COMAL Help and Exchange disks
COMAL Reference Guide

All this is packaged with a note pad in a nice, padded three ring binder. The price is $39.95 for the complete package. I have received everything but the handbook and Len promises this is almost ready. Send for the package or for more information to: COMAL User's Group, 5501 Groveland Terrace, Madison, WI 53716

The present version of COMAL will run on any PET with BASIC 4.0, 32K of user memory and the 4040 disk drive. It will automatically adjust for either 40 or 80 column machines. The FULL version of COMAL places all of COMAL into memory at one time but leaves only about 4K for user programs. The SPLIT version is two programs. The IN program is the editor and prepass portion which allows you to write programs and save them on disk. The OUT program allows you to execute the program you wrote with IN after loading it from disk. The two programs are nicely linked together so that one will load the other. Remember that COMAL is an interpreter which means it is no faster than BASIC. Also, since it is a structured language, programs written in COMAL take up more memory than those written in BASIC. However, COMAL is easy to learn and the resulting programs are easier to read and debug. I have been able to write fairly complex programs after only a month's casual usage.

I would like to briefly explain some of the nice features of the COMAL editor and interpreter. A later article will show some examples of COMAL and how it compares to BASIC. Still other articles will explore more of COMAL's features and structure providing people are interested.

After loading COMAL you are ready to enter a program just as you would in BASIC. The line numbers in COMAL are used as references for the editor and are NOT used within the COMAL program. One thing you will find about COMAL is that it demands that you tell it EXACTLY what you want to do. This is both good and

bad. When you type in a direct command or a program line COMAL will immediately look at it and tell you if there is an obvious SYNTAX ERROR. Of course, it cannot evaluate errors in logic or structure until you run the program. One nice feature of COMAL is that it will automatically indent the statements that occur within structures whether you type them in that way or not. For example, you might type what is shown on the left but COMAL would automatically produce the easier to read version on the right.

```
0100 FOR I=1 TO 10 DO
0110 PRINT I
0120 IF INT(I/2) = I/2 THEN
0130 PRINT "EVEN"
0140 ELSE
0150 PRINT "ODD"
0160 ENDIF
0170 NEXT I
```

```
0100 FOR I=1 TO 10 DO
0110   PRINT I
0120   IF INT(I/2) = I/2 THEN
0130     PRINT "EVEN"
0140   ELSE
0150     PRINT "ODD"
0160   ENDIF
0170 NEXT I
```

Pretty clever, right? To avoid this when correcting errors a program can be listed using EDIT which produces the listing given on the left. Programs are executed using RUN and the LOAD and SAVE commands operate as in BASIC. However, the SAVE command does not store programs as source code but in a slightly altered form. There is a way to store source code procedures or subprograms and then merge them with the program in memory.

If you like, you may use the AUTO command to generate line numbers. To get rid of a line or lines you cannot just type the line number. Remember COMAL demands you describe your actions completely, so you must precede the lines to be deleted with DEL. Programs can easily be renumbered using the built in RENUM command. To send ouput to the printer is as easy as SELECT OUTPUT "LP" then its back to the screen with SELECT OUTPUT "DS". This can be done in the direct mode or in a program. STATUS reads the disk error channel and CAT is used to get a catalog from a diskette.

In general, COMAL is easy to learn especially if you already know BASIC. I suspect many of us will find it painful at first since old habits die hard. COMAL produces more readable programs which are easier to debug and modify. The price is right for a complete language system when Commodore PASCAL lists at $295. There are some drawbacks such as the BASIC 4.0, 32K and 4040 disk requirements but these are a must with a RAM-based language system. I feel COMAL is a big step forward and that the language will be supported through Len Lindsay and the many PET owners around the country.

****

Surveys

The response to our reader survey was good but I know some of you out there forgot to send it in. I realize that you are now feeling VERY guilty. You can ease your guilt and help us out by sending that survey right now! Remember all responses are confidential and we won't single you out as an example. We really would like to get a complete picture of our readers to provide the best information possible

In particular, we would like you to make comments about The PAPER. Include notes about how we are doing and what YOU would like to see. We need the addresses of local user's groups. It is also important that we know the types of equipment our readers use.

****

Library Tape Indexing Program

by Doug Haluza

In the early days of the PET one of the nitpickers' favorite complaints was that the PET's tape deck didn't have a tape counter. This did make finding programs on long library tapes (a tape with several programs saved on it) a problem, but after using the Atari and its tape system that relies on the counter I'm glad the PET doesn't need one.

To circumvent this problem several industrious programmers developed various "indexing" programs that positioned the tape to the beginning of a program that was needed. These programs used fast-forward so programs that were near the end of the tape could be found and loaded quickly. Many of the original newsletters carried articles about these programs and some of you may remember one that appeared in the old LIPS Newsletter.

Since these programs were written in BASIC they had to be reloaded each time they were needed. They were usually saved as the first program on a tape which was then 'divided' into blocks depending on the maximum size of the programs you were putting on that tape. The user could choose the program from an index, the tape deck would run at fast-forward speed until the proper amount of time had passed. When the tape was properly positioned a few POKEs turned off the cassette motor and the program could be loaded. You could also save a program at this point by removing the library tape, loading the program you wanted to save, replacing the library tape and saving the program.

Although this method works and is probably better than nothing, it has several disadvanatges. Because the index is a BASIC program, it must be reloaded each time the tape needs to be positioned. This destroys the program in memory and makes the creation of library tapes very tedious. Verifying requires rewinding the tape, reloading the index, repositioning the tape, reloading the original copy of the program you just saved and then verifying the program. You may still be in trouble if the program is a little too long and the next one recorded writes over its tail end. Of course, programs which are too short waste tape and positioning time.

The indexing method that I will show you here uses a machine language program that sits in the second cassette buffer and acts like a software tape counter. Once loaded it need not be reloaded unless that area of memory is disturbed by loading another program there, using tape deck #2, or employing the disk commands from BASIC 4.0. This program works with all versions of the BASIC ROMs.

This method does not require that the tape be divided into blocks of equal length; programs are simply saved end to end. The indexing program, DIRECT III, is usually saved as the first program on tape for convenience only. This indexing method can be used with tapes created by the old indexing method or tapes with no indexing where programs were just saved end to end.

Before creating a library tape you should type in each of the three programs needed for indexing; DIRECT III, SETUP III, and DATA TAKER III. Use the Machine Language Monitor to enter the code shown in the hex dump for DIRECT III. On old PETs (BASIC 1) you must load and RUN the TIM program; new PET (BASIC 2 and 4) owners may enter the monitor by typing SYS 4. Type 'M 033A 03F4 to display the area of memory and type the numbers shown in the hex dump over the numbers already there. Remember to hit RETURN at the end of each line. Double check the numbers and then save the program by typing 'S 01,DIRECT III,033A,03F5' on old PETs or 'S "DIRECT III",01,033A,03F5' for new PETs. After saving, try running the program by typing SYS 826. The PET should tell you to rewind the tape and prompt you with 'TYPE PROGRAM CODE #'. To try it out type an arbitrary number, say 20, and 'HIT F.FWD' when it says to. In about 5 seconds 'STOP TAPE' should appear. When you stop the tape the index program will jump to the PET's load routine and you will see PRES PLAY AND RECORD ON TAPE #1. Press STOP on the PET at this point since your tape is not yet set up.

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.

VIC 20™ personal computer

Here's the first computer you can really afford. Full PET-BASIC for $299* with color and sound.

Advantages:

1. Deductible Employee Business expense.
2. Use your own T.V. or buy a 12" color TV from us. (Also deductible).
3. Will this be the start of a side business??

* Add $40 for TV interface if not using a color monitor.
N.Y.S. Residents add 7% NYS tax.

AMERICAN PERIPHERALS

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

Now with 12" Screen & Bleeper!

Get a 16K PET for $530[1]

or a 16K APPLE for $2,500[2]

Limited Time Offer

Good for 16K PET,
32K PET, and 8032 CBM.

Here are the rules:

1. Send purchase order for:
   A. 2 units at regular price
   B. 1 free unit
2. After receipt of payment, free unit is shipped direct to you.

[1]This price applies for the 3 for 2 deal only. See details on back.

[2]This price assumes the most common Apple configuration: 16K computer ($1330), single disk drive and controller ($645), and 13 inch color monitor ($550).

In this offer, when you buy 2 computers at regular price, you receive an additional unit free. The list price of the 16K PET is $795 each, or 2 x $795 = $1590 for the 3 units, which comes to a price of $530 each. For the 4032, the price each is $1295, or 2 x $1295 = $2590 for 3, which comes to $866 each. If you need an 80 column display (for data processing or business), the 8032 is $1795, or 2 x $1795 = $3590, for the 3 units.

QUALITY SCHOOLS
Main Street
Anytown, U.S.A.

P.O. 10875

TO: AMERICAN PERIPHERALS
122 Bangor Street
Lindenhurst, N.Y. 11757

SAMPLE PURCHASE ORDER

QUANTITY	ITEM	UNIT PRICE	TOTAL PRICE
2	Commodore PET16K Microcomputers 16K RAM, 14K ROM	$ 795.00	$1590.00
1	Commodore PET16K Microcomputer (Special 3 for 2 deal)	FREE	FREE
3	External Cassette Recorders	95.00	285.00
2	Commodore PET32K Microcomputers	1295.00	2590.00
1	Commodore PET32K Microcomputer (Special 3 for 2 deal)	FREE	FREE
2	Commodore 8032 Microcomputers	1795.00	3590.00
1	Commodore 8032 Microcomputer (Special 3 for 2 deal)	FREE	FREE
3	External Cassette Recorders	95.00	285.00
1	4022 Tractor Printer	795.00	795.00

(Multiples of these amounts may be ordered)

Call for our latest software catalog: 516-226-5849 — Jackie or Alice

NOTE:

Be sure to order external cassette recorders.

If you are doing mainly word processing or data entry, order style 8032, which has a keyboard more oriented towards business applications.

The 3 for 2 deal applies only to the computers, not to printers, disks, or recorders.

All units come with a 90 day factory warranty, parts and labor.

If the PET bombs and won't respond to the STOP key, or if the program doesn't work for some other reason turn off the PET and start again. Load DIRECT III, enter the monitor and carefully recheck the data to make sure it matches the hex dump. Try resaving the program and testing it again.

Now type in and save the two BASIC support programs, SETUP III and DATA TAKER III, right after DIRECT III. If you have tapes which already have programs recorded on them then skip the next paragraph.

To organize your programs on library tapes you should break them up into groups of ten to twelve programs that you want on each C-60 tape (5 or 6 for a C-30). Put a blank label on all your blank library tapes and save DIRECT III at the beginning of each, but DON'T rewind. Now save your programs on each tape one after the other and write the names of the programs on the label. Write small but leave some room after the name to write the code number that will be found later.

Now load SETUP III and place a library tape that you have just recorded in the tape deck. Run SETUP III and follow the directions given. The program will read the file headers of each program on the tape. It then takes the amount of time it took to reach a program at play speed, uses a data table that has play vs. f.fwd codes in it, interpolates, finds the code number for the position of the program, and then prints the program name and code number on the screen. When SETUP III has finished reading the entire tape write the code numbers displayed next to the program names on the tape. Now when you want to access a program rewind the tape, load DIRECT III (if it's not already loaded), type SYS 826 to run it and type the program's code number when asked. HIT F.FWD, STOP, and PLAY on the tape deck when you are prompted to do so. The PET will position the tape to the beginning of the program and should start loading it in 15 to 20 seconds.

The DATA TAKER III program allows you to use this indexing method on any type or any length cassette. The relationship between play and fast-forward access times is not simple because the tape moves at a constant speed (1 7/8 ips) in play mode, but the speed varies when fast forwarding depending on how much tape is on the takeup reel. This is the reason for the data table found in SETUP III. This table was generated for "no label" C-60 and C-30 TDK tapes with the DATA TAKER III program. Although these values should be average, you may have to generate your own data if your tapes are thicker or longer. To use DATA TAKER III, load DIRECT III and then DATA TAKER III. Run DATA TAKER III and enter different program code numbers when you are asked. You may use the codes from SETUP III or make up your own. Fast-forward the tape when asked, stop it, flip it over and press play so that the tape will run at the slower speed. When the tape stops take down the number it displays; this is the number of jiffies of play time corresponding to that program code. Repeat this for more code numbers spaced at regular intervals (e.g. 10,20,30...100,200,300...1000). Repeat the procedure three times for each code and find the average of the tries. Use this data (code number and number of jiffies) to replace the data in the SETUP III program. You should be sure to use a few code numbers that make the tape run almost to the end. This is tedious but necessary only once to customize the data in SETUP III for your tapes.

Briefly the Direct III program works like this: Locations 1 and 2 in zero page are used to store the program code in Binary Coded Decimal (BCD). One digit is stored in four bits of each byte. These locations are the USR jump vector; they were used because they're about the only locations that are free on all versions of BASIC.

After zeroing loactions 1 and 2 the program calls the "MESS" subroutine. This routine uses the Y register to sequentially print the messages in the data stored at MSGS. After the first two messages are printed the PET waits for the user to input the program code. Only legal digits (0-9) are accepted and are used to build the code stored in 1 and 2 by shifting all digits already there up four bits and then adding the digit input. If more than four digits are typed,

the first few typed will be shifted out and lost. If no digits or zero is typed the code will be zero but will be interpreted as 10000.

Once return is hit the PET continuously tests the appropriate bit for a switch closure (hopefully F.FWD) on tape #1. The stop key is also tested so the user can break out if necessary. Once F.FWD is pressed the program begins waiting for 256 transitions of T2C-H in the 6522 and then does a 2-byte BCD subtraction on the code number in locations 1 and 2. This process is repeated until the code is decremented to zero and the tape is then properly positioned. The motor is then stopped and interrupts are disabled so the PET's interrupt routine doesn't start the tape motor while waiting for the user to stop the tape. When the tape is stopped the PET jumps to its LOAD routine and loads the program from the tape.

```
0 REM DATA TAKER III  BY DOUG HALUZA. USE TO TAKE DATA FOR SET UP II
2 REM
3 REM ***LOAD DIRECT III FIRST!!***
4 REM
5 POKE947,96
10 SYS826:PRINT:PRINT"FLIP TAPE AND PRESS PLAY";:WAIT59455,8,8:TI$="000000"
20 PRINT", OK...";:WAIT59455,8:PRINTTI:GOTO10
```

```
1 REM *** SET-UP III FOR USE WITH DIRECT III; BY: DOUG HALUZA ***
2 REM *** USE DATA TAKER PROGRAM TO GET DATA TO GET DATA FOR YOUR TAPES ***
5 DIMP(20),F(20):PRINT"[CLR,DOWN] REWIND AND STOP TAPE [DOWN]
7 T=PEEK(50003):P=241-29*T:S=62894-8*T:IFT=160THENS=62949:P=212
10 READP(J),F(J):IFP(J)>-1THENJ=J+1:GOTO10
15 IFA=0THENINPUT"30 OR 60 MIN. TAPE";A:IFA=30THENJ=0:GOTO10
20 POKEP,1:SYSS:TI$="000000":T=1:GOTO40
30 SYSS:T=TI
40 FORI=639TO658:PRINTCHR$(PEEK(I));:NEXT:GOSUB100:GOTO30
100 I=0:REM *** INTERPOLATION SUBROUTINE ***
200 IFP(I)<TTHENI=I+1:GOTO200
300 PRINTINT((T-P(I-1))/(P(I)-P(I-1))*(F(I)-F(I-1))+F(I-1)):RETURN
999 REM *** TDK C-60 DATA ****
1000 DATA0,0,710,10,1215,20,1735,30,2250,40,2766,50,4080,75,5465,100
2000 DATA11168,200,17392,300,24042,400,31240,500,38769,600,46854,700,51050,750
3000 DATA55355,800,74250,1000,84262,1100,-1,-1
3999 REM *** TDK C-30 DATA ***
4000 DATA0,0,694,10,1200,20,1707,30,2225,40,2750,50,4087,75,5420,100,11111,200
5000 DATA17271,300,23860,400,30968,500,38479,600,46430,700,55005,800,-1,-1
```

```
.:  039A D0 DF C9 00 D0 DB A0 09
.:  03A2 20 B9 03 78 A0 3D 8C 13
.:  03AA E8 AD 10 E8 29 10 F0 F9
.:  03B2 58 4C D5 FF 20 D2 FF C8
.:  03BA B9 BF 03 D0 F7 60 52 45
.:  03C2 57 49 4E 44 20 26 20 53
.:  03CA 54 4F 50 20 54 41 50 45
.:  03D2 00 0D 54 59 50 45 20 50
.:  03DA 52 4F 47 52 41 4D 20 43
.:  03E2 4F 44 45 20 23 00 0D 48
.:  03EA 49 54 20 46 2E 46 57 44
.:  03F2 0D 00 00 00 00 80 00 02
```

```
.:  033A A0 00 84 01 84 02 20 B9
.:  0342 03 20 B9 03 20 CF FF C9
.:  034A 0D F0 21 38 E9 30 30 F4
.:  0352 C9 0A B0 F0 48 A2 04 A5
.:  035A 01 0A 85 01 A5 02 2A 85
.:  0362 02 CA D0 F3 18 68 65 01
.:  036A 85 01 10 D8 20 B9 03 20
.:  0372 E1 FF AD 10 E8 29 10 D0
.:  037A F6 20 E1 FF AD 49 E8 CD
.:  0382 49 E8 F0 FB CA D0 F5 F8
.:  038A 38 A5 01 E9 01 85 01 A5
.:  0392 02 E9 00 85 02 D8 C5 01
```

Machine Language for the Near Beginner: Part 2

by R Bressler

Before you read this part of our machine language tutorial be sure to review Part 1 found in the last issue. Make sure you are familiar with all the terms that were introduced and that you know how to use the TIM monitor. Last time we wrote a simple program to add two numbers, hand assembled it and then looked at ways to type it into the PET. We learned the commands LoaD Accumulator (LDA), CLear Carry (CLC), ADd with Carry (ADC), STore Accumulator (STA)and BReaK (BRK). Along with these commands we discussed the immediate, absolute, implied and zero page addressing modes. Addressing modes are very important in 6502 programming and we will discuss them more completely in this session.

It might be best to understand binary numbers before we go further. Remember that the 6502 operates on binary numbers which we often represent in hexadecimal. It also works in base 2 or binary. At times it is necessary to convert between these two bases. Fortunately, this conversion is relatively easy. First let's take a brief look at binary numbers.

In base two there are only two digits, 0 and 1. These are the binary digits or 'bits'. Since the PET is an 8 bit machine, we only need worry about a total of 8 digits. The bit positions are labeled 0 to 7 from right to left. Let's look at some binary numbers.

Bit #	7	6	5	4	3	2	1	0
Place value	128	64	32	16	8	4	2	1
Bit	0	0	1	0	0	1	0	1

This binary number 0010 0101 is equal to 37 in decimal or $25 in hex. To convert from binary to hex or vice versa we may convert first to decimal but there is a better way. Lets look at some binary numbers with their decimal and hex equivalents.

Binary	Hex	Decimal
0000 0001	$01	1
0010 0100	$24	36
0110 1111	$6F	101

To convert binary to hex, group the binary digits in fours starting at the right. Now convert each group of four AS IF IT WAS A SEPARATE NUMBER. In the chart above 0110 is 6 and 1111 is F. Converting hex to binary is also easy. Convert each hex digits to binary as if it were a separate number. If we are doing simple, unsigned operations, then things are relatively easy. Working with signed numbers becomes a little complicated. The leftmost bit, bit 7, is looked at as the SIGN BIT. If it is 0 then the number is positive but if it is 1 the number is considered negative. This means 0111 1111 or $7F is the largest positive number we can represent in 1 byte or 8 bits. When we get to 1000 0000 or $80 we now have a negative number . This number is considered to be -128 decimal which means $FF is -1. With certain operations all this is of little consequence. However, we must consider this information carefully when doing subtraction or using branch instructions.

Now let's talk a little about subtraction. The subtraction instruction for the 6502 is a subtract with carry, SBC. This means that the SBC will subtract the contents of the selected memory location and the compliment of the carry (1-C) from the accumulator. Just as we clear the carry before an addition we must set the carry before a subtraction. If after the subtraction is completed, the carry remains set we know we subtracted a small number from a larger one. This means that we didn't have to borrow from the carry. If the carry has been cleared, then we did borrow which means we subtracted a bigger number from a

smaller. It might look something like this:

Carry Before (in dec)	Carry After	Numbers	Result	Carry Before (in hex)	Carry After	Numbers	Result
1	1	10-4	= 6	1	1	11-A	= 7
1	0	22-49	= 27	1	0	41-4F	= F2

In the last example we must realize that F2 is really decimal -14 which is the correct answer. Let's try a short program which will subtract the contents of a memory location from the accumulator and store the result.

```
Label    Mnemonic     Operand     Comment

         SEC                      * set the carry
         LDA          #$41        * load the accumulator with $A1
         SBC          LOC1        * subtract contents loaction 1
         STA          LOC2        * store accumulator loaction 2
         BRK                      * go back to monitor
```

First we set the carry as we always do before a subtraction. We use the immediate addressing mode to load the accumulator. Then we subtract the contents of LOC1 from the accumulator and store it at LOC2. Here is our translation if location 1 is $0390, location 2 is $0391 and we start at $0398.

```
0398 38
0399 A9 41
039B ED 90 03
039E 8D 91 03
03A1 00
```

Remember to put some data in $0390 before executing the program. Type G 0398 to run the program and then redump memory using M 0390 03A1. If we put 4F at LOC1 we will see F2 at LOC2 after the subtraction. We know this is a negative number and that a borrow should have occured which means the carry would be clear. To see what has happened we must look at the carry flag in the status register.

Status Register

This register is the one that indicates what has happened as a result of the last arithmetic or logical operation. It is the third number from the left when you use the R command or break into the monitor. This number is given in hex and to see what flags are set we must convert this number to binary and know which bits in this binary number represent what flags. A flag is said to be set if it is 1 and reset if it is 0. For our example, let's use 32 which is 0011 0010 in binary.

Bit #	7	6	5	4	3	2	1	0
Flag	S	V	-	B	D	I	Z	C
Bits	0	0	1	1	0	0	1	0

Bit 7 - S - Sign flag: This flag is identical to bit 7 of the result of the last operation. If it is set, equal to 1, then the result was negative. A 0 or reset means a positive result.

Bit 6 - V - Overflow flag: This is set equal to the exclusive-OR of the carries out of bit 6 and 7 during arithmetic operations. Don't worry too much about this now. Be aware that, at times, an answer may not be quite what it seems especially when interpreted by the microprocessor.

Bit 5 - Unused: This bit is unused. Its status is unimportant.

Bit 4 - B - Break flag: This flag will be 1 if a BRK has been executed

Bit 3 - D - Decimal mode flag: We usually work in hexadecimal but we can work in BCD, binary coded decimal. If we are operating in the decimal mode this flag will be 1.

Bit 2 - I - Interrupt disable flag: If the IRQ interrupts are disabled this flag will be 1. A zero here means the interrupts are enabled.

Bit 1 - Z - Zero flag: When an operation results in a zero this flag is set, equals 1. A 0 here represents a non-zero result.

Bit 0 - C - Carry flag: This flag is important in both addition and subtraction and some other less important operations. In addition, this flag is set if there is a carry out of the leftmost or "most significant" bit. For example, adding $10 and $0F would NOT set C but adding $AF and $F1 would set the carry. The sum of the two numbers is too big to fit in one byte and this sets the carry. In subtraction, C must be set BEFORE the operation. After the subtraction, C is reset if a borrow occurs. If no borrow is required then C remains set.

Examining the Status Register

The status register is one of the bits of informa ion displayed by the monitor after a program has been executed. You may never really have to interpret what this register means when writing programs but it would be good to get a basic understanding at this point. The best way to examine the meaning of SR is to do some additions and subtractions and then look at the answers and the status register. First, type in the simple two number addition program from the first article in this series. Now try to follow the chart below. Enter the first number given in hex in $0390 and the second in $0391. After each addition look at the answer and the SR.

```
1) hex       04 + 04 = 08        4) hex       02 + FF = 01
   decimal    4 +  4 =  8           decimal    2 + -1 =  1
   SR = 30 = 0011 0000              SR = 31 = 0011 0001
   flags set   -B                   flags set   -B    C

2) hex       7A + 09 =   83       5) hex       FF + 01 = 00
   decimal 122 +  9 =  131           decimal   -1 +  1 =  0
   SR = F0 = 1111 0000              SR = 33 = 0011 0011
   flags set SV-B                   flags set   -B   ZC

3) hex        80 + FF =   7F      6) hex        80 +   80 = 00
   decimal -128 + -1 = -129           deciaml -128 + -128 = -256
   SR = 71 = 0111 0001              SR = 73 = 0111 0011
   flags set  V-B    C              flags set  V-B   ZC
```

To discuss ALL these examples in full would take too much space. What you might try is some of your own examples. Try to predict the results, then execute the program and check your results. Notice that the break flag (B) is always set and that the status of the unused bit (-) is not important.

In the second example we added two positive numbers and got a negative result. The S flag is set to indicate a negative but the overflow is also set to indicate a possible misinterpretation of the answer. Similarly, the third example shows the addition of two negatives apparently gives a positive. The sign flag is not set but the overflow is to indicate a possible error. Notice that the carry is set since the answer will not fit in one byte. In the fifth problem the carry flag is set since the answer will not fit in one byte. The zero flag is also set since the result was a zero. In the last example two large negative numbers are added and the answer should be -256. Here the carry flag is

set since the answer will not fit in one byte and the zero flag is set to indicate a zero result. However, the overflow must also be set to indicate a possible error.

To try some subtraction examples we can use the program we wrote above. We load the accumulator immediately with the first number below and put the other in location 1 at $0390. The examples below give some of the types of results that may occur:

```
1) hex       10 - 08 = 08
   decimal   16 -  8 =  8
   SR = 31 = 0011 0001
   flags set    -B    C

2) hex       0A - 0F = FB
   decimal   10 - 15 = -5
   SR = B0 = 1011 0000
   flags set S -B

3) hex        81 - FF =   82
   decimal -127 - -1 = -126
   SR = B0 = 1011 0000
   flags set S -B

4) hex       FB -   85 =  76
   decimal   -5 - -123 = 118
   SR = 31 = 0011 0001
   flags set    -B    C

5) hex       12 - FE = 14
   decimal   18 - -2 = 20
   SR = 30 = 0011 0000
   flags set    -B

6) hex        A0 - 05 = 9B
   decimal  -96 -  5 = -101
   SR = B1 = 1011 0001
   flags set S -B    C

7) hex       80 - 05 = 7B
   decimal -128 - 2 = -130
   SR = 71 = 0111 0001
   flags set  V-B    C

8) hex       08 - 80 =   88
   decimal    8 - -128 =  136
   SR = F0 = 1111 0000
   flags set SV-B

9) hex       04 - 04 = 00
   decimal    4 -  4 =  0
   SR = 33 = 0011 0011
   flags set    -B   ZC
```

Example one is very straightforward with no catches. The carry flag remains set since no borrow occurred. In number 2 the sign flag is set to indicate a negative result and the carry flag is reset to show a borrow. This is a result of subtracting a big number from a smaller one and the answer in signed binary is indeed negative. The result in example three is similar to that in two. A small negative number is subtracted from a large negative number which produces a smaller negative number and the flags are set accordingly. The fourth problem shows a large negative number being subtracted from a smaller negative. The answer is positive and the sign flag is NOT set. Also, the carry flag remains set since there was no borrow. In number 5 the carry flag is reset since a borrow occurred. The result, however, is positive so the sign flag is not set. Example six is a little more interesting since a small positive is being subtracted from a negative number and the result is a larger negative number. Here the sign flag is set for the negative result and the carry remains set to indicate no borrow. Problem number seven shows a small positive number being subtracted from the largest negative number. This results in a problem since the answer should be a bigger negative number. The overflow is set to indicate the problem and the carry remains set since no borrow was necessary. Example eight is the result of subtracting the largest negative number from a positive number. The answer should be a positive but the result ends up looking negative. The overflow is set to indicate a possible error and the sign flag is also set. Finally in the last example a zero result is obtained and the zero flag is set. The carry remains set to indicate no borrow. Again, I think it best if you try to create your own examples and predict the results before executing the program and discovering the correct answer.

Unconditional Jump

The 6502 has one unconditional jump instruction which has two addressing modes. For now we will use JMP with the absolute or direct mode which has an op code of 4C. The problem with this instruction is that it is like GOTO in BASIC. It is unconditional and by itself can only produce an infinite loop or a simple jump ahead. For example, the shortest jump using this command might be: 0398 4C 98 03. This would put the PET in a very tight loop which it might be impossible to stop. Of course, we could create longer and more useful loops but they would all have no end. To solve this problem we need a conditional jump like the IF...THEN... in BASIC.

Branching

The 6502 has several branch commands and they all work on the basis of testing the status of the flags we mentioned before. The branch instructions come in pairs with one branching if a flag is set and its partner doing so if the flag is reset. Branching may be ahead or back but in either case the addressing mode is RELATIVE. Before we do this let's list the branch commands we might want to use now: BCC - Branch if Carry Clear and BCS - Branch if Carry Set both test the carry flag, BEQ - Branch if EQual to zero and BNE - Branch if Not Equal to zero test the zero flag, BMI - Branch if MInus and BPL - Branch if PLus test the sign flag.

Relative addressing is used with all branching commands and is based on counting how many steps you want to go forward or back. The number following the branch op code is a single byte sometimes called the offset. This indicates the number of memory locations to skip or 'branch' over NOT the actual address of the branch. This limits the forward branch to 129 bytes and the backward branch to 126. Before we worry about looping using branches let's try a simple 'branch on condition' program.

The program we will attempt is one which will find the larger of two numbers located in two memory locations. We will again use the second cassette buffer and start our program at $0398 and data storage at $0390. First, let's write the assembly version of the program.

```
Label   Mnemonic    Operand     Comment

        LDA         LOC1        * load accumulator from LOC1
        CMP         LOC2        * compare accumulator to LOC2
        BCS         STOR        * branch if first bigger
        LDA         LOC2        * load accumulator from LOC2
STOR    STA         LOC3        * store accumulator at LOC3
        BRK
```

We already know about the LDA, STA and BRK instructions. We have used the BCS command which will act on the result of another new command, CMP. CMP is the 6502 instruction for CoMPare. In this case we will compare the accumulator to LOC2 using absolute addressing. CMP actually subtracts the contents of the operand from the accumulator but does NOT change the accumulator or the operand. All CMP does is set the S, Z and C flags according to the result of the subtraction. If the contents of LOC1 is larger than the contents of LOC2, the CMP instruction will set the carry flag since no borrow will have to be made. If the opposite situation occurs the carry flag will be reset since a borrow will happen. Notice that it is not necessary to set the carry before CMP as it is before a subtraction. We use BCS instead of BPL since we saw how it is possible to misinterpret an answer. The sign flag may be incorrect but the carry flag will never be wrong. It is pretty easy to see that if no borrow occurs and the carry flag is set then the branch WILL occur and the contents of LOC1 will be

stored. If a borrow DOES occur then the carry flag will be reset and no branch will occur. If there is no branch then the accumulator will be loaded from LOC2 and this number will be stored as the larger. Let's try hand assembling this program:

```
0398  AD 3A 03
039B  CD 3B 03
039E  B0 03
03A0  AD 3B 03
03A3  8D 3C 03
03A6  00
```

Before we go on to type in and execute this program let's look more carefully at the BCS instruction with its RELATIVE addressing. Notice that the operand is 03 and the branch is forward. If the branch does not occur the program will continue at step 03A0. If the branch does occur we want to go to step 03A3. The difference between these two steps is 3 and, therefore, the operand is 03. The calculation of a backward branch to set up a loop is slightly different and we will discuss that later. Now type in the program using the monitor and be sure to add data to $0390 and $0391. Then type G 0398 and use the M command to display the memory locations again to see if the larger of the two numbers is, in fact, stored at $0392. So, to calculate a forward branch: 1) Call the address immediately after the offset 0; 2) Count ahead by one for every memory location. So in this case we count 1, 2, and 3; 3) Enter this number as the offset after the branch instruction.

Next time we will look at how to set up a program loop and use a screen filling program as an example. We will also look at new addressing modes an subroutines.

****

ATUG, CIPUG, and the MIDNITE GAZETTE

Some time ago I got a copy of a new PET-oriented newsletter, The MIDNITE Software Gazette. MIDNITE is published quarterly by the Central Illinois PET User's Group. Jim Strasma is aided by his wife Ellen and other members of CIPUG in the publucation of MIDNITE. MIDNITE is available for simply a self addressed stamped envelope with the issue number desired in the lower lefthand corner. MIDNITE started out as a hardware and software review for PET related products. Reviews are kept short to allow as many product reviews as possible. The newsletter has now begun to branch out to include editorials, comments and some PET tips. It really is good and certainly worth the 'price'. Even though you only have to send a SASE, I am sure Jim would gladly accept donations.

Through MIDNITE I learned of ATUG, the ASSM/TED User's Group, for the Eastern House assemblers. ATUG maintains a superb library of programs which are in the public domain. This is just another reason you should go out and buy the EHS MAE if you still haven't. Even if you don't own this assembler there are object code versions of many of the best programs. There are also BASIC programs that anyone can use. As of right now ATUG has 6 disks labeled UA to UF. To obtain them you should send a quality diskette for each volume you want enclosed in a good mailer. Be sure to include a SASE and a clear indication of which disks you want. You should also enclose AT LEAST $5 per diskette. In addition, Jim would prefer you send some of your own programs on disk rather than sending one blank.

If you want MIDNITE send your envelopes directly to Jim Strasma. At the same time you can inquire of Jim who is now handling the ATUG disks. By the way, Jim is one of the most knowledgable PET people around and has always responded to my inquiries and requests in record time. Jim's address is now: Jim Strasma, PO Box 647, Pawnee, IL 62558.

Observations on Vol. 3, Issue 8/9/10

by Roy Busdiecker

Congratulations to The PAPER on beginning Volume 4! It's good to know our old friend will be around another year.

Because there was some question as to whether or not there would be another volume, I postponed ... perhaps it would be more honest to say procrastinated ... writing this column long enough to get a personal invitation from Ralph (SEND OBSERVATIONS ... SOON!).

Where We've Been

The first observation should be one regarding where we are. From that original PET with its toy keyboard and built-in cassette unit, which showed up in late 1977, we've seen Commodore struggle through the introduction of a disk drive, followed by "upgrade ROMs" in a computer with a "grown up" keyboard. As the popularity (and company) grew, we saw more and more new products ... BASIC 4.0, the 80-column model 8032 computer (basis for the best word-processors available), model 8050 1-megabyte disk drives and VIC-20 ... even a cassette unit with a tape counter! Now on the horizon are the 8096 computer, the SUPER-PET (no longer called MicroMainFrame), single disk drives, more languages, and a raft of accessories for the VIC.

With the proliferation of systems from Commodore, we find that the users of CBM products tend to fall into at least three general communities, based on the type of computer ... 8000 series, 4000 series, and VIC. The VIC is a great first or game computer, but most owners of the larger systems have grown beyond the point where a VIC would satisfy all their needs. Time will tell.

Last Issue

Now, on to The PAPER. Issue 8/9/10 of Volume 3 was a winner! Its 62 pages (even in a triple edition) put it more in the magazine category than newsletter. Its letter-quality print continued the improvement of the previous issues. The mix of topics was good, as was the quality of the articles. I like having the ads together at the back, too. Even the troublesome typos seem to be coming under control ... with a little room still left for further improvement.

Background/Foreground had a good collection of info. From my vantage point, Commodore seems to be getting their act together ... but the local dealer is more important to most of us than the manufacturer. If you find a good one, support him. A few extra dollars in intial cost may save much aggrevation later on. Contrary to the article, I do find some diskettes better than others, especially on the 8050. My favorites are Maxell, Memorex, and BASF.

Doing the Expert's Job

I strongly endorse "ed's" response to Will Hendricks (from my native Kansas City area). If you're not technically qualified to do systems integration, then buy the system from a store. If you buy components seperately and assemble them yourself, the money you may save will be offset by the headaches ... and each vendor can legitimately claim his product works. If a store sells you a system that doesn't work properly (even though it may have the same components), it's then the store's responsibility to make it work. As a store owner, I'm biased ... but I've seen some folks get into big trouble trying to save a few dollars.

Incidentally, the same comment probably applies to the discussion of software prices on p. 4. What a program does is often no more important than the way it does it. If you're a programmer, you may not get shook up when an error occurs, the program drops out, and the system says "READY". If you're a beginner, and

don't know how to recover without losing data, it may be worth paying more to get a "bulletproof" version of the program that takes care of these problems.

The method of showing jumpers on page 8 was cute and effective. Good work!

Variables Where Constants Are Specified

Another solution to using string variables in BASIC 4.0's DOPEN is to enclose the string variables in parentheses. As a matter of fact, other disk command parameters can be variables, if they too are enclosed in parentheses. An example is shown below:

```
10 A$="filename"
20 R=3 : B=1
30 DOPEN#1, (A$), D0, L50
40 RECORD#1, (R), (B)
50 INPUT#1, C$: PRINT C$: DCLOSE
```

The example opens an existing relative disk file named FILENAME, and reads record #3, starting at byte 1 up to the first carriage return.

I worry about Phillip Chao's project to consolidate the software libraries of all user groups. Software exchange is a valuable service, as long as, those running it insure that it involves only public domain programs. Too many such arrangements are no more than organized piracy, and will eventually drive good software producers out of the market.

Ralph's article on machine language programming was excellent ... very few authors are close enough to being beginners to present the material in the sequence that seems natural. To be helpful, the "answers" must appear in about the same order as the questions that come up in the mind of the reader. Well done!!

I enjoyed Doug's article on computed GOTO's. The capability he provides, in a suprisingly short machine language routine, is a powerful extension to CBM BASIC.

The comment by Jim Fowler (Assembly Language Programming) that assembly (machine language) routines should be kept short is dead right. Debugging, changes, maintenance .. call it what you like, it's so much more difficult and risky at machine level than in so called "higher languages" like BASIC, FORTRAN or PASCAL. Today's microcomputer revolution would not be here if everyone had to struggle with machine language. It's great when you need it but use it in little doses, properly blended with the high level languages.

Will there be an "Auto Repeat Keys: Version Four"? Seems the theme is a popular one! Emil Volcheck's article is well presented. Importantly, it presents the machine code in three forms: monitor dump, disassembly and BASIC DATA statements. (Watch for Version 4 in the next issue. - ed)

Chris Werner's "PET Face Lift" is a popular idea. Prices are up now, but it's cheaper than buying a new machine!

Software producers, read Bill Batcher's article on page 23. Once your program does what you want it to do, make it do what he wants, before you sell it. Make it worth the money to the users. Users: recognize a good deal when you get it. Buy good programs instead of copying them. It's a double edged blade we deal with!

Commercial Products

With all due respect to the 2022, 2023, 4022 (Programmable Characters, pp. 28 - 30), anyone considering buying a printer should be sure to see the Epson MX-80. No PET graphics, but it has lots of other features at a great price.

PAIC's Toolkit (p. 33) now has a big brother. Command-O adds many new features including the renumbering of a selected range of lines. Sells for $79.95, from Skyles, a bargain.

I agree wholeheartedly that the 8032/8050/Spinwriter/WordPro4+ combination is the best word processing system going (p. 46).

Spacemaker II (p. 50) is well-made and useful ... if you need room for several ROMs in the same slot, this is a handy way to accomodate them.

Back in the ad section I noticed MAE from Eastern House. I've been using several versions of EHS assemblers for over a year, and have been more satisfied with them than other suppliers.

One of the consequences of a long issue is a long Observations column in the next issue. Sorry 'bout that!

Keep up the good work! If you get the urge to write something, do it quickly .. and stick it in the mail before you have time to change your mind.

Key Notes

by Jerry Key

Here are a few comments on some things that appeared in the last issue of The PAPER.

1) Page 5: I have used the @ to overwrite data in files and programs very frequently with no problems. Data ouput should be the most sensitive in this area and it should only be a concern if there are multiple files on a disk. On single file disks I see no problem. Personal opinion!

2) Page 6: Numerous articles have been written on lower case decension. I have solved the problem by placing a CHR$(141) at the end of all direct print statements. I do it without benefit of any formalities such as commas, semicolons or colons before or after it. I put them in liberally and then remove them until I find the one I need to leave in. Leave them all in if space is not a problem.

3) I think that the article dealing with machine language for the beginner is great. There is so much lacking of a tutorial nature in this area. It should be made clear that SYS1024 increments the stack pointer. BASIC 2.0 users should use SYS64785 while those with bASIC 4.0 should use SYS54386. These do not do a break to the monitor and are the correct entry points.

4) On refering to the various versions of BASIC, I would suggest The PAPER adopt the conventions used in MICRO and TRANSACTOR. BASIC 1 is the old or version 02 ROMs that are referenced in the Osborne publications. BASIC 2 is the version 03 ROMs sometimes refered to as the UPGRADE ROMs. According to TRANSACTOR BASIC 3 was never released. BASIC 4 we all probably recognize but can BASIC 5 b far behind?

5) As mentioned in MIDNITE #3, upgrades to Moser's ASSM/TED for printer and disk are available from EHS free and copies may be obtained of the upgrade code by an SASE to EHS. They also have a regular free newsletter. An upgraded version of ASSM/TED (BASIC 2) that will read and generate MAE files is available from the same address as MIDNITE with proof of ASSM/TED purchase. I can also provide a BASIC 4 version but purchase of MAE is the best route. THe BASIC 4 version will be available from the same address.

KEYPRINT 4.0

by Ralph Bressler

KEYPRINT is an interrupt driven machine language program which dumps the PET's screen to a printer. This occurs when the backslash key is pressed at any time during a program or in the direct mode. This program has proved very handy and originally appeared in COMPUTE, Nov/Dec 1980 for the BASIC 2.0 ROMs. An article in COMPUTE, March 1981 gave a revision for BASIC 1.0. I have not seen this program for BASIC 4.0 yet so here it is with some comments.

The versions of KEYPRINT already presented begin at address $033A. This is the beginning of the second cassette buffer and will be destroyed if you use a disk with BASIC 4.0. To avoid this problem I have relocated KEYPRINT and made the changes necessary for BASIC 4.0. I have presented it here first as a hex memory dump and also as DATA with a short BASIC program to place it in memory.

```
035C 78 A9 03 85 91 A9 67 85
0364 90 58 60 A5 97 C9 45 D0
036C 03 20 73 03 4C 55 E4 A9
0374 80 85 20 A9 00 85 1F A9
037C 04 85 B0 85 D4 20 D5 F0
0384 20 48 F1 A9 19 85 22 A9
038C 0D 85 21 20 D2 FF A9 11
0394 AE 4C E8 E0 0C D0 02 A9
039C 91 20 D2 FF A0 00 B1 1F
03A4 29 7F AA B1 1F 45 21 10
03AC 0B B1 1F 85 21 29 80 49
03B4 92 20 D2 FF 8A C9 20 B0
03BC 04 09 40 D0 0E C9 40 90
03CA 0A C9 60 B0 04 09 80 D0
03CC 02 49 C0 20 D2 FF C8 C0
03D4 28 90 CB A5 1F 69 27 85
03DC 1F 90 02 E6 20 C6 22 D0
03E4 A6 A9 0D 20 D2 FF 4C CC
03EC FF FF FF FF FF FF FF FF
```

```
10 FOR I = 860 TO 1004 : READ N : POKE I,N :NEXT
100 DATA 120,169,3,133,145,169,103,133,144,88,96,165,151,201,69,208,3,32,115
110 DATA 3,76,85,228,169,128,133,32,169,0,133,31,169,4,133,176,133,212,32
120 DATA 213,240,32,72,241,169,25,133,34,169,13,133,33,32,210,255,169,17,174
130 DATA 76,232,224,12,208,2,169,145,32,210,255,160,0,177,31,41,127,170,177
140 DATA 31,69,33,16,11,177,31,133,33,41,128,73,146,32,210,255,138,201,32
150 DATA 176,4,9,64,208,14,201,64,144,10,201,96,176,4,9,128,208,2,73
160 DATA 192,32,210,255,200,192,40,144,203,165,31,105,39,133,21,144,2,230,32
170 DATA 198,34,208,166,169,13,32,210,255,76,204,255
```

KEYPRINT is now initialized by typing SYS 860 and pressing the backslash will dump the screen. You might want to change the key that causes the screen dump. This is controlled by location $036A or 874 decimal in BASIC 4.0 ($0347 or 840 in BASIC 2.0). The easiest way to change this is to POKE the matrix coordinate of the key you want to use into this location. To find this number you could run the short program below and press the key you want as it runs. Note the number that results when you hold that key down.

```
10 PRINT PEEK(151) : GOTO 10
```

KEYPRINT is a good example of what can be done with interrupts. Other examples from readers would be appreciated.

Flex File

A Product Review

Type: Software by Dr. G. Piasecki
Model PET: BASIC 3.0 or 4.0 and DOS 1.0 or 2.1 (32K RAM recommended)
Source: AB Computers
252 Bethlehem Pike
Colmar, PA 18915
Price: $60

FlexFile consists of several programs which together can do file maintenance and print a mailing list. It is one of the many data base management programs that have cropped up lately. FlexFile is written in BASIC and machine language by Michael Riley of Papermate fame. Versions exists for BASIC 3 and 4 and for the 2040 or 4040 disk drive and 32K of user memory is suggested to get the most use out of the software. Printout can be done by a CBM or ASCII printer (NEC Spinwriter). You can even select the printer device number and line feed if your printer requires one.

The RANDOM ACCESS DATA BASE program allows you to collect data in the format you create. You can have up to 42 fields in a record and the data is saved by the random accesss method on disk. Total record size is 250 characters. This data can then be sorted by several key fields which you choose. The commands are ADD, REPlicate, CHANGE, DELete, NEXT, PREVious, GOTO, FIND, TRANSfer to another program, USER (to add your own subroutine) and EXIT. These commands allow you to quickly find a record, modify it, and browse backwards or forwards from the record in view. You can find a record with a 'wild card' format using a partial field followed by an asterisk. It allows field data to be identical such as several last names of Smith. This is not allowed in some data base managers.

The MAILING LABELS program uses data that has been set up by the DATA BASE program. When record size is 127 characters each disk can hold over 1000 records on a 4040 or 2800 on the 8050. This program allows you to create, change, and save the format for future use. You can specify the number of labels across, the number of lines per label and up to 3 fields combined on one line. After designing the format you can PRINT ALL, or select ONE or a COMBINATION of TYPES of record, or REPEATS of one record.

The REPORT WRITER program allows the DATA BASE to be OUTPUT in a FORMAT you can easily create. The format can be saved but when brought back can be changed quickly and easily by SCREEN COMMANDS. These changes can be in HEADER LINES, TITLES for columns, SELECTION of record criteria, POSITION and CONTENTS of columns and CALCULATE COLUMNS INFO. The contents of fields can be placed in columns and then functions such as +, -, *, /, log, sin, cos, and tan can be performed on them. Columns can be related mathematically to other columns and a column can be used to show a running total as each record is printed. This is accomplished by a cumulative ARITHMETICAL ADD to a particular column on SUCCESSIVE RECORDS (passed from row to row). Another routine allows you to right and left justify from a decimal point. At the end of the report a TOTAL and/or AVERAGE can be calculated for any column.

SELECTION of RECORDS is indeed another powerful feature. It allows you to LOGICALLY LINK the following 4 parameters: FIELD #; EQUALITY TYPES; ARGUMENT; and LOGICAL CONNECTIONS.

1. FIELD #: (You choose field).

2. EQUALITY TYPES: (You select one of eight).
   1.) EQUALS (Identical in each character)
   2.) LESS THAN (numerically)
   3.) GREATER THAN (numerically)
   4.) PRECEEDS (alphabetically)

5.) FOLLOWS (alphabetically)
6.) NOT EQUAL TO (not identical to)
7.) PATTERN MATCHES (?="wild character", *="wild ending")
8.) INCLUDED IN (field characters to argument characters)

3. ARGUMENT: (any alpha-numeric string; ex. could be A, B, ABC, etc. entered by you in the original DATA BASE in a field you could have designated as TYPE).

4. LOGIC CONNECTIONS: (AND, OR, THEN PRINT) - Choosing the AND or OR allows you to go back and create a 2nd LINKING of the above 4 parameters. When you have completed the desired LOGIC LINK you choose the THEN PRINT command.

OTHER PROGRAMS included with FlexFile are:

CREATE A SEQUENTIAL FILE: This is done from the RANDOM ACCESS FILE you have set up. This allows you to dump your original DATA BASE FILE, CHANGE the RECORD SIZE and the FIELDS.

LOAD FROM A SEQUENTIAL FILE: Use this to RELOAD the created SEQUENTIAL FILE into the new RANDOM ACCESS FILE. (A very useful procedure to avoid re-entering of data if you decide to change your data base format.)

ALPHABETICAL ORDERING: Used to return the loaded sequential file to correct alphabetical order. It can also be used to SORT a FIELD that was NOT previously designated as a KEY field.

The FlexFile program comes on DISK with a 28 page manual (soon to be revised according to Gene Beals. ed.) There is no ROM protection and the manual is easy to follow. It has 2 pages on PROGRAMMER INFORMATION and a list of variables used. The program is user friendly. Once the FORMATS have been set up and saved you do not have to refer to the book to make the complex changes. The questions for changes appear on the SCREEN. When ordering remember to specify the ROM type of your DISK and COMPUTER.

I have used FlexFile on data that requires: 15 fields, a sort on 6 key fields, rapid finding of a record to update specific fields, and an ALL or SELECTIVE printout. It works very well and so far I have encountered no disaster producing bugs. Ocassionally after a write my disk error light will stay on but I am able to continue without any obvious problems. To me this program is worth its weight in gold. At a price of $60 it appears underpriced relative to the other programs present on the market.

****

Programming Techniques

by Roy Busdiecker

Another way to "people proof" your input is to open a "keyboard file".

```
10 OPEN1,0: REM DEVICE #0 IS THE KEYBOARD
20 INPUT#1, A$ : REM TRY IT!!
```

Still another approach is to use a GET loop.

```
10 B$ = ""
20 GET A$: IF A$="" THEN 20
30 IF A$=CHR$(13) THEN PRINT B$ : END
40 B$=B$+A$ : GOTO 20
```

Some RND(X) Thoughts

by Jim Fowler

Someone once asked me if I knew a good, short machine language routine to generate random numbers. I had to say no at the time but I started reading up on the subject. As usual, I learned alot, but still cannot give a good, short machine language routine for generating random numbers. I felt that some of the things I learned might be of interest to others, so here they are.

First, there is nothing random about PET's 'random number generator.' All such routines are completely deterministic. Once you give me the value of RND(X), I can predict all the rest of the numbers the PET will give you. This is true until you change the 'seed'. It is more correct to call the sequence generated by the routine called by RND(X) 'PSUEDO-RANDOM numbers'. Mathematical tests are available to detect randomness, although they are actually testing for the opposite (correlations, cycles). Such tests tell you whether the sequence 'looks' random. Just as long as the user of the program finds the RND function unpredictable the sequence is the equivalent of true randomness. So much for philosophy!

PET generates its RND numbers as follows: When you power up there will a number will appear in a 5-byte register ($88-$8C in BASIC 4.0 and 2.0, $CB-$CF in BASIC 1.0). When you call RND(x) this number is multiplied by a constant in ROM, then another constant is added to the result, the original number is added to this result and the bytes are juggled around. All of this is done in floating point binary arithmetic. The final result is put into the register where it can be the basis for the next RND(x) call. The program can change to another 'seed' in that register by calling for RND(-X) where X is the new seed. To give a truly unpredictable next number the seed ought to fill all five bytes of the register.

It is clear that there ought to be some unpredictable basis for this calculation, but the only way you can get such a basis is by your selection of the seed. A useful source of unpredictable numbers is TI, the 'jiffy clock', which measures time in 60ths of a second and begins when the PET is turned on. If there are indeterminate time periods in the running of the program such as a user response, then the value of TI at certain places in the program will not be the same from run to run. You can use TI as a seed at strategic places in your program in the following way. First, TI is about three bytes long and will not give a good seed by itself, but RND(TI) will. So to get truly random (ie. unpredictable, uncorrelated, uncyclic) numbers, start with a seed of TI by calling RND(-TI). You have to have the minus sign to tell the PET that this is a new seed. Once the seed is set you can call RND(1) or any other positive number in parentheses and you will get the same sequence. In the new ROM PETs you can use RND(0) instead of the above, but that won't work on the old ROMs so you are better off not using it.

In machine language you can use JSR $D229 in BASIC 4.0, JSR $DF7F in BASIC 2.0 or JSR $DF45 in BASIC 1.0 and generate the next number in the sequence. The 5 byte register where the number appears (and where the seed is put if you are changing seeds) is at $88 to $8C in BASIC 4.0 and BASIC 2.0 and $CB to $CF in BASIC 1.0. If you want random bytes or random integers (2 bytes), use the least significant bytes (ones with the highest address) from the above. They will be as 'random' as the whole floating point number. I have a hunch the second (next to the last) byte might be better than the first, but I have no proof of this. The RND routine takes about 50 milliseconds to run depending on whether you are changing seeds or not and how much calculation it takes to get the new seed.

Command-O

A Product Review

Type: Firmware
Model PET: BASIC 4.0; series 4000 or 8000
Source: Skyles Electric Works
231 E South Whisman Rd.
Mountain View, CA 94041
Price: $75.00

by Ralph Bressler

When I upgraded to BASIC 4.0 it immediately became obvious that my old friend, the PAIC Toolkit, would no longer work. One problem was that many vital memory locations had changed in BASIC 4.0. The other problem was the slot, $B000, the Toolkit had lived in for over a year was now occupied by a BASIC 4.0 ROM. What to do? I began hunting for a replacement. I have decided that Command-O is the best replacement since it includes all the old Toolkit commands, improves on some and adds several others.

The manual that comes with Command-O is written by Greg Yob in his typical rambling style. It really gets the job done covering all situations and explaining what can happen if you are someone who feels directions are for the other jerk. After inserting the IC at $9000 you type SYS 36864 and Command-O is ready to go. Typing KILL will deactivate Command-O. There are some disadvantages in using this aid. It slows BASIC by about 20% since it examines BASIC programs for its commands before allowing BASIC to take over. This technique uses the CHRGET routine so that other rpograms that do the same conflict. Finally, it employs the 2nd cassette buffer for work space and you cannot use any other routines in this area unless you use KILL first. So what makes it so good? Here are a few ways to answer that question. The last comment represents my overall opinion of that command.

AUTO causes automatic line numbering starting at a line number you choose and increasing by your increment. Defaults to 100,10. Remembers where you were if you exit and then re-enter. OK

BEEP allows you to easily access the PET's CB2 output for sound in a program. Requires a duration and pitch parameter. Eliminates need for three separate POKES. OK

DELETE gets rid of a range of unwanted lines just as in the old Toolkit. HANDY

DUMP prints the values of all non-array variables at any time you stop the program. Useful for debugging programs. HANDY

EXECUTE loads and runs a BASIC program from disk. Allows wildcards as does DLOAD. EH!

FIND permits you to search for variables, literal strings or tokens in a program. You may specify a range of line numbers for the search. Whenever a match is made, the line is printed and a reverse field marker flags the occurrance of the search. This can be a problem since EVERY occurrance in every line is printed. HANDY

HELP points out errors in programs after they have occurred. Similar to the Toolkit with reverse field indicating the approximate position of the mistake. This is a little more accurate than the Toolkit but still must be used IMMEDIATELY after the error. HANDY

INITIALIZE should have been included by Commodore since DOS 2.1 will only

initialize if a new disk has a different ID than the last one. Disks which are different but have the same IDs can be ruined. OK

MERGE is the Toolkit APPEND command for disk. This command will add the program or routines you specify from disk to the end of the program in memory. HANDY

MERGE# allows you to chain programs by doing an overlay. It overlays a program from disk onto a specified line range in the program in memory. OK

PRINT USING helps to easily format numbers and strings to the screen or printer. You set up the image string and then use it to print your information. HANDY

RENUMBER is perhaps the biggest improvement over the Toolkit. This command renumbers a program. You specify the starting line and increment. You can ALSO give a line range on which the command will operate. This means you can renumber subroutines and main program loops independently! SUPER

SCROLL enables a series of functions. SET allows you to define a single key as almost anything. This could be an entire BASIC line and some very useful and creative functions can be devised. SCROLL also enables the STOP key to delete characters to the right of the cursor. It also causes any key to repeat after being held down for more than 1/2 sec. Finally, SCROLL allows you to use the up/down cursor key to scroll forwards or backwards through your program. This takes some getting used to but is better than constantly listing part of the program. SUPER

SEND transmits commands to the disk on channel 15. Any command can be sent. OK

TRACE uses the top line of the screen to display the current BASIC program line being executed. You can specify the speed of the trace. To execute the program you hold down SHIFT. Letting up on SHIFT will freeze the program. This slows program execution terribly and makes input almost impossible. HANDY

There are several other commands for the 8032 which I have not tested. They are mostly concerned with screen editing and window definition. I must say that I use Command-O mostly as a programming aid. I seldom include its commands in my programs since I want others to be able to use them. Each command has a single character abbreviation but I always use the full command. I find that I use RENUMBER, FIND, DUMP, DELETE and HELP most just as I did with the Toolkit. The repeat key and scroller are nice additions and I may occasionally use TRACE. MERGE is very useful and is a welcome addition.

I feel that this product is a must for serious software developers and would be a good addition to the collection of any programmer.

****

A Death in the Family

Have you heard that the heart of PET, Apple, Atari, Ohio Scientific and KIM/SYM/AIM is dying? That's right! Knowledgable people in the field of microprocessors feel that the 6502 is on the way out within perhaps a year. Many feel the highly touted 6809 is too little, too late and see the Motorola 68000 as the rightful successor. The $198 price tag for a single chip may initially discourage some manufacturers and hobbyists but prices will drop as more companies manufacture them. Several companies including Digital Acoustics are beginning to support the 68000. Digital is planning an 'attached processor' for the 8000 series PETs. More information is planned for the next issue. If anyone wants to comment, please do as this information is hard to come by.

```
LINE# LOC CODE LINE

0001  0000             ;***********************************
0002  0000             ;*                                 *
0003  0000             ;* DIRECT III BY DOUG HALUZA       *
0004  0000             ;* LIBRARY TAPE INDEXING PROGRAM   *
0005  0000             ;* FOR THE COMMODORE PET/CBM       *
0006  0000             ;* (ALL ROM VERSIONS)              *
0007  0000             ;* REVISED AUG-81                  *
0008  0000             ;* PERMISSION TO USE NOT TO SELL   *
0009  0000             ;*                                 *
0010  0000             ;***********************************
0011  0000             LBYT    = $01           ;LOW BYTE
0012  0000             HBYT    = $02           ;HIGH BYTE
0013  0000             RDT     = $FFCF         ;READ A CHAR
0014  0000             WRT     = $FFD2         ;WRITE A CHAR
0015  0000             LOAD    = $FFD5         ;LOAD TAPE PRG
0016  0000             T2CH    = $E849         ;TIMER LOCATION
0017  0000             CB2     = $E813         ;MOTOR CONTROL
0018  0000             SW      = $E810         ;CASS SW SENSE
0019  0000             STOP    = $FFE1         ;STOP KEY TEST
0020  0000                     *=$033A        ;START IN CASS2
0021  033A  A0 00       STRT    LDY #00
0022  033C  84 01               STY LBYT
0023  033E  84 02               STY HBYT       ;INITIALIZE
0024  0340  20 B9 03           JSR MESS       ;'REWIND/STOP'
0025  0343  20 B9 03           JSR MESS       ;'TYPE CODE #'
0026  0346  20 CF FF    READ    JSR RDT         ;GET NEXT CHAR
0027  0349  C9 0D               CMP #13        ;IS IT RETURN?
0028  034B  F0 21               BEQ OUT        ;IF SO GO ON
0029  034D  38                  SEC
0030  034E  E9 30               SBC #48        ;ELSE ADJUST
0031  0350  30 F4               BMI READ       ;AND
0032  0352  C9 0A               CMP #10        ;CHECK
0033  0354  B0 F0               BCS READ       ;IF 0-9
0034  0356  48                  PHA            ;THEN SAVE IT
0035  0357             ;*** 2 BYTE 4 BIT SHIFT ***
0036  0357  A2 04               LDX #04
0037  0359  A5 01       SHFT    LDA LBYT
0038  035B  0A                  ASL A          ;SHIFT LOW HALF
0039  035C  85 01               STA LBYT
0040  035E  A5 02               LDA HBYT
0041  0360  2A                  ROL A          ;INTO HIGH HALF
0042  0361  85 02               STA HBYT
0043  0363  CA                  DEX
0044  0364  D0 F3               BNE SHFT       ;FOUR TIMES
0045  0366  18                  CLC
0046  0367  68                  PLA
0047  0368  65 01               ADC LBYT       ;ADD LAST DIGIT
0048  036A  85 01               STA LBYT
0049  036C  10 D8               BPL READ       ;AND GET ANOTHER
0050  036E             ;*** WAIT FOR F.FWD BUTTON ***
0051  036E  20 B9 03    OUT     JSR MESS        ;'HIT F.FWD'
0052  0371  20 E1 FF    FFWD    JSR STOP        ;TEST STOP KEY
0053  0374  AD 10 E8             LDA SW
0054  0377  29 10               AND #$10       ;CHECK SWITCH BIT
0055  0379  D0 F6               BNE FFWD       ;LOOP TILL PRESSED
```

```
LINE# LOC  CODE LINE

0056  037B                ;*** TIMING ROUTINE ***
0057  037B  20 E1 FF     TIME   JSR STOP        ;TEST STOP KEY
0058  037E  AD 49 E8     CNT    LDA T2CH
0059  0381  CD 49 E8     WAIT   CMP T2CH        ;WAIT FOR
0060  0384  F0 FB                BEQ WAIT       ;TRANSITION
0061  0386  CA                   DEX
0062  0387  D0 F5                BNE CNT        ;256 TIMES
0063  0389                ;*** BCD SUBTRACTION ***
0064  0389  F8                   SED            ;USE BCD MODE
0065  038A  38                   SEC
0066  038B  A5 01                LDA LBYT
0067  038D  E9 01                SBC #1         ;SUBTRACT ONE
0068  038F  85 01                STA LBYT
0069  0391  A5 02                LDA HBYT
0070  0393  E9 00                SBC #0         ;FROM BOTH
0071  0395  85 02                STA HBYT       ;BYTES
0072  0397  D8                   CLD
0073  0398  C5 01                CMP LBYT       ;ARE THEY EQUAL?
0074  039A  D0 DF                BNE TIME       ;NO: CONTINUE
0075  039C  C9 00                CMP #0         ;EQUAL TO ZERO?
0076  039E  D0 DB                BNE TIME       ;NO: CONTINUE
0077  03A0                ;*** TURN OFF MOTOR ***
0078  03A0  A0 09                LDY #09        ;SET MESS PTR TO:
0079  03A2  20 B9 03             JSR MESS       ;'STOP TAPE'
0080  03A5  78                   SEI
0081  03A6  A0 3D                LDY #$3D
0082  03A8  8C 13 E8             STY CB2        ;STOP MOTOR
0083  03AB  AD 10 E8     OFF    LDA SW
0084  03AE  29 10                AND #$10       ;WAIT FOR
0085  03B0  F0 F9                BEQ OFF        ;STOP BUTTON
0086  03B2  58                   CLI
0087  03B3  4C D5 FF             JMP LOAD       ;LOAD PROGRAM
0088  03B6                ;*** MESSAGE ROUTINE ***
0089  03B6  20 D2 FF     MORE   JSR WRT         ;PRINT THE CHAR
0090  03B9  C8            MESS   INY             ;BUMP POINTER
0091  03BA  B9 BF 03             LDA MSGS-1,Y   ;GET NEXT CHAR
0092  03BD  D0 F7                BNE MORE       ;LOOP UNTILDONE
0093  03BF  60                   RTS            ;THEN RETURN
0094  03C0                ;*** MESSAGES ***
0095  03C0  52 45         MSGS     .BYT 'REWIND & STOP TAPE'
0096  03D2  00 0D                 .DBYTE $000D
0097  03D4  54 59                 .BYT 'TYPE PROGRAM CODE #'
0098  03E7  00 0D                 .DBYTE $000D
0099  03E9  48 49                 .BYT 'HIT F.FWD'
0100  03F2  0D 00                 .DBYTE $0D00
0101  03F4                       .END

ERRORS = 0000
```

Programs for Commodore's PET®

Present this ad from THE PAPER and receive $2 off your purchase price. Valid at your local dealer or when ordered direct.

• PROFESSIONAL TOOLS

- Business Researcher (Rvsd. Simplex) . . . . . (16k) **$50**
- RNAV3 Navigator (Western U.S.) . . (16k) **$30** (8k) **$25**
- Education Pack (High School Math/Sci) **$15**

• DISK BOWLING SYSTEM (PET/CBM)

- Leaguebowl-24 . . . . . (Disk 32k) **$145**
- Archivebowl . . . . . (for above) **$40**
- Allsweepbowl . . . . . (for above) **$40**
- Tournamentbowl . . . . . (Cass. 16k) **$30**

• HOME & OFFICE DATA MGMT.

- Deluxe Address (16k) **$40**
- Home Address Book **$25**
- Grocery Mart **$15**
- Home Inventory . . . **$20**
- Shopper **$20**
- Dinner's On! **$15**

• GAMES & ADVENTURES

- Mansion! **$15**
- Museum! **$15**
- Pentagon! **$15**
- Fur Trapper **$15**
- High Seas **$15**

Write for details or ask your local dealer.

BRILEY SOFTWARE
P.O. Box 2913
Livermore, CA 94550
(415) 455-9139

Dealers: Letterhead inquiries invited. Photocopies of this ad are NOT valid coupons. One coupon per purchase. This coupon may be redeemed for face value plus 15¢ for handling if it was received from customer upon purchase of one of the above programs. Offer void where restricted by law.

The Good Books from *Cow Bay Computing*

FEED ME, I'M YOUR PET
(Book 1)

LOOKING GOOD WITH YOUR PET
(Book 2)

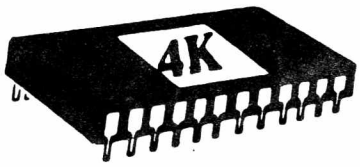
TEACHERS' PET
(Lesson Plan, Answer Key)

Instruction, Classwork, homework worksheets, quizzes for classroom use.

Workbooks are $4.95. TEACHERS' PET is $4.00

COW BAY COMPUTING
BOX 515
MANHASSET, N.Y. 11030

For the Commodore PET/CBM



PLUG IN MORE POWER!

IMMEDIATE DELIVERY

MACHINE LANGUAGE UTILITY-PAC

Rom based firmware includes 43 COMMANDS to ENHANCE use of your computer including D.O.S (WEDGE)!, ASSEMBLER, DISASSEMBLER, HUNT MEMORY, QUICK TRACE, COMPARE MEMORY, TRANSFER MEMORY, RELOCATOR, WALK CODE, INTEGRATE MEMORY (Hex Code and Ascii), VIDEO SCREEN DUMP (STANDARD OR ENHANCED), FILL MEMORY, FAST TYPE HEX ENTRY, HEX TO DECIMAL & ASCII CONVERSIONS and VISE VERSA! Most functions to screen or printer. Makes handling and understanding of machine code programming easier. Also included are these programs accessible from Basic. D.O.S.(WEDGE), LOW CASE LIST, SCREEN DUMPS (STANDARD & ENHANCED), RE-NEW, AUTO REPEAT, DISK APPEND, REV.SCREEN, DISPLAY. AVAILABLE FOR 3.0, 4.0 & 8032 COMPUTERS IN LOCATIONS $A000 or $9000; SPECIFY WHEN ORDERING. MANUAL included. Does not lower user memory. A MUST for new or advanced programmers alike! We accept VISA & MASTERCARD. 30 DAY MONEYBACK TRIAL! SEE REVIEW IN COMPUTE! JUNE 1981 ISSUE! ORDER NOW!

4K ROM for 3.0 (A000) or (9000) $79.95 + $2 S&H
4K ROM for 4.0 (A000) or (9000) $79.95 + $2 S&H
4K ROM for 8032 (A000) or (9000) $79.95 + $2 S&H
DEALER INQUIRIES INVITED!

BASIC UTILITIES 3.0 or 4.0

This 4K Rom contains 19 COMMANDS for Basic programming. INCLUDED are AUTO - RENUMBER - DELETE - FIND - APPEND (TAPE) - DUMP - HELP - TRACE - STEP - OFF - D.O.S. - SCREEN DUMP - ENHANCED SCREEN DUMP - RE-NEW - LOW CASE LIST - AUTO REPEAT - APPEND (DISK) - REV.SCREEN - DISPLAY - THIS ROM IS LOCATED AT $9000. These programs do not lower user memory and will greatly enhance your programming ability through use of the automatic disk & printer routines! 30 DAY MONEYBACK TRIAL, ORDER NOW!

4K ROM.....$79.95 + $2 S&H
2K ROM W/FIRST 10 COMMANDS ONLY..$39.95 + $2 S&H
PLEASE SPECIFY WHICH ROM SET YOU HAVE.

SEND $1 FOR CATALOG AND $5 OR $10 OFF OF YOUR NEXT ORDER!

COMPETITIVE SOFTWARE
21650 Maple Glen Drive
Edwardsburg, MI 49112

Skyles Electric Works

BASIC Programmer's, Toolkit™, Disk-O-Pro™, Command-O™

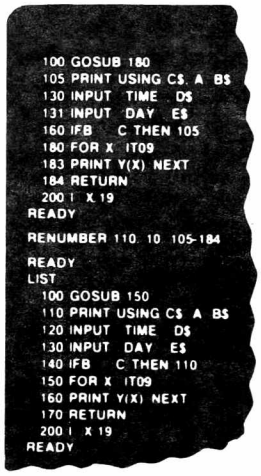
For CBM™ Owners Who Want More Fun And Fewer Errors with Their Programs

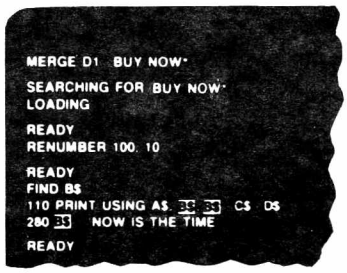
Here are nineteen commands you'll need, on a single chip you can install in two minutes without tools, **on any CBM or CMB system.** 4KB of ROM firmware on each chip with a collection of machine language programs available from the time you turn on your PET to the time you shut it off.

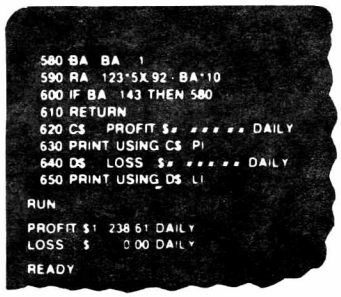
For CBM 8016 and 8032; BASIC 4.0

BASIC Programmers Command-O™

AUTOed **DUMP**ed **DELETE**ed **FIND**ed (improved) **HELP**ed **KILL**ed **OFF**ed **TRACE**ed (improved) **RENUMBER**ed (improved) **INITIALIZE**BS **MERGE**BS **MOVE**BS **EXECUTE**BS **SCROLL**ed **OUT**ed **SET**ed **SEND**BS **PRINT USING**BS **BEEP**BS







COMMANDO-O AVAILABLE IMMEDIATELY ALL MODELS PET / CBM Please Specify

PRICES:

BASIC Programmers **Toolkit™** (chip only)	$40.00
BASIC Programmers **Disk-O-Pro™** (chip only)	$75.00
BASIC Programmers **Command-O™** (chip only)	$75.00
Interface boards (needed sometimes)	$20.00-$50.00
Instruction Manual (with redeemable $5.00 coupon)	$5.00

Shipping and handling $2.50 USA/Canada, $10.00 Europe/Asia

California residents please add 6% or 6-1/2% sales tax as required

Reserve your Disk-O-Pro, Command-O today

Toolkit™, Disk-O-Pro and Command-O available for immediate delivery

VISA, MASTERCHARGE ORDERS CALL (800) 227-9998 (except California residents)
CALIFORNIA ORDERS PLEASE CALL (408) 965-1735

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

Microphys Programs

PET

Software Specialists
Science and Education

APPLE

Microphys is pleased to announce the availability of its educational software for use with the Commodore PET/CBM and Apple/Bell & Howell microcomputers. These programs have been successfully employed in **Chemistry, Physics, Calculus** and **Mathematics** classes on both the high school and college levels.

The programs are supplied on C-10 cassettes and are accompanied by complete instructions so that even those with little or no computer experience may immediately utilize the software in their classrooms. Each cassette retails for $20 and may be obtained from leading computer dealers or directly from Microphys.

Each **Physics** and **Chemistry** cassette has both a computer-assisted and individualized-instruction program recorded on opposite sides of the cassette. The **CAI program** guides the student through interacts with the computer and receives immediate evaluation of his responses and/or assistance when needed. The **I/I program** generates a unique set of problems for each student. The computer can supply answers so that the student may check his own work. If the teacher directs the computer to suppress these answers, the student completes his work at home and then feeds his results into the computer which grades his work, supplying the answers to those questions incorrectly solved by the student. **NOTE:** each time a particular program is run, a different set of numerical values is generated. In most instances, an entirely new problem is presented. The Mathematic and Calculus cassettes have only the individualized-instruction feature.

For those using disk drives, the programs have been coherently grouped and are available on diskettes. The price of each diskette is $180 which represents a considerable savings with regard to the individual cassette price.

A partial list of the programs available appears below. Please write for the Microphys Winter Catalog which describes the complete line of educational software for use on the PET/CBM and Apple/Bell & Howell microsystems.

CALCULUS CASSETTES

PC726-Differentiation of Algebraic Functions
PC727-Maxima/Minima Problems: Part I
PC728-Maxima/Minima Problems: Part II
PC729-Relative Rate Problems: Part I
PC730-Relative Rate Problems: Part II
PC731-Integration of Algebraic Functions
PC732-Differentiation of Trigonometric Functions
PC733-Integration of Trigonometric Functions
PC734-Integration: Areas of Plane Figures
PC735-Integration: Volumes of Solids
PC736-Integration: Arc Lengths
PC737-Integration: Surface Areas of Solids

M4 **Calculus I Diskette** contains 726-737

PHYSICS AND CHEMISTRY CASSETTES

1 Linear Kinematics
2 Projectile Motion
3 Momentum and Energy
4 Energy and the Inclined Plane
5 Inelastic Collisions
6 Centripetal Force
7 Pulley Systems — Machines
8 Specific Heat Capacity
9 Calorimetry
10 Heats of Fusion/Vaporization
11 Specific Gas Laws
12 General Gas Laws
13 Thermodynamics I
14 Thermodynamics II
15 Transverse Standing Waves
16 Longitudinal Standing Waves
17 Lenses and Mirrors
18 Refraction of Light
19 Series Circuit Analysis
20 Parallel Circuit Analysis I
20A Parallel Circuit Analysis II
21 Series Parallel Circuit Analysis
22 Faraday's Law
23 Gram-Molecular Mass
24 The Mole Concept
25 The Molarity Concept
26 The Normality Concept
27 The Molality Concept
28 Stiochiometry: Mass/Mass
29 Stoichiometry: Mass/Volume
30 Stoichiometry: Volume/Volume
31 Stoichiometry: General
32 Percent Concentration
33 pH Concept
34 EMF of Electrochemical Cells
35 Electric Field Analysis
36 Photoelectric Effect
37 Symbols and Valence Drill
38 Names of Compounds Drill
39 Formulas of Compounds Drill
40 Total Internal Reflection

P1 **Physics I Diskette** contains the following programs
1 2 3 4 5 6 7 8 9 10 301 302 305 306
P2 **Physics II Diskette** contains the following programs
11 12 13 14 15 16 17 18 19 20 20A 21 35 36 40 303 304

Microphys Programs

2048 Ford Street
Brooklyn, New York 11229
(212) 646-0140

PET & APPLE II USERS

TINY PASCAL

Plus+
GRAPHICS

The TINY Pascal System turns your APPLE II or PET micro into a 16-bit P-machine. You too can learn the language that is slated to become the successor to BASIC. TINY Pascal offers the following:

- **LINE EDITOR to create, modify and maintain source**
- **COMPILER to produce P-code, the assembly langauage of the P-machine**
- **INTERPRETER to execute the compiled P-code (has TRACE)**
- **Structured programmed constructs: CASE-OF-ELSE, WHILE-DO, IF-THEN-ELSE, REPEAT-UNTIL, FOR-TO/DOWNTO-DO, BEGIN-END, MEM, CONST, VAR ARRAY**

Our new TINY Pascal PLUS+ provides graphics and other builtin functions: GRAPHICS, PLOT, POINT, TEXT, INKEY, ABS AND SQR. The PET version supports double density plotting on 40 column screen giving 80 x 50 plot positions. The APPLE II version supports LORES and for ROM APPLESOFT owners the HIRES graphics plus other features with: COLOR, HGRAPHICS, HCOLOR, HPLOT, PDL and TONE. For those who do not require graphics capabilities, you may still order our original Tiny Pascal package.

PET BASIC 4.0 version also available

TINY Pascal PLUS+ GRAPHICS VERSION-
PET 32K NEW Roms cassette..... $55
PET 32K NEW Roms diskette..... $50
APPLE II 32K/48K w/DOS 3.2 or 3.3..... $50

TINY Pascal NON-GRAPHICS VERSIONS-
PET 16K/32K NEW Roms cassette..... $40
PET 16K/32K NEW Roms diskette..... $35
APPLE II w/ROM Applesoft 32K w/DOS..... $35
APPLE II w/RAM Applesoft 48K w/DOS..... $35

USER's Manual (refundable with software order)..... $10
6502 Assembly Listing of INTERPRETER-graphics..... $25
6502 Assembly Listing of INTERPRETER-non graphics..... $20

master charge

VISA

FREE postage in U.S. and CANADA. Orders may be prepaid or by bankcard (include card number and expiration date). Michigan residents include 4% state sales tax. Orders accepted via THE SOURCE CLO852

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

VIGIL

INTERACTIVE GRAPHICS/GAME LANGUAGE

FOR THE PET/CBM

VIGIL is an exciting new interactive language for your PET/CBM micro. **VIGIL** - Video Interactive Game Interpretive Language - is an easy to learn graphics and game language that lets you quickly create interactive applications.

- More than 60 powerful commands permit you to easily manipulate graphics figures on the screen
- Double density graphics give you 80 X 50 plot positions on your 40 column PET/CBM
- Large number display capability, access to two event timers and tone generation (if you have ext. speaker)
- Load and save your **VIGIL** programs to cassette or diskette
- Nine interactive programs demonstrate the power of **VIGIL** - Breakout, SpaceWar, AntiAircraft, U.F.O., SpaceBattle, Concentration, Maze, Kaleidoscope & Fortune
- Comprehensive user's manual with complete listings of enclosed programs

VIGIL comes on cassette, or diskette ready to run on any 40 column PET/CBM micro with at least 8K of memory. Specify ROM-set when ordering. 6502 listing of the **VIGIL** Interpreter available separately.

	US & Canada	Foreign
VIGIL FOR Pet/CBM on Cassette or Diskette (w/9 programs)	$35	$40
VIGIL User's Mannual (refundable with software)	$10	$12
VIGIL Interpreter listing (6502 Assembly language)	$25	$30
PET MACHINE LANGUAGE GUIDE	$8	$10

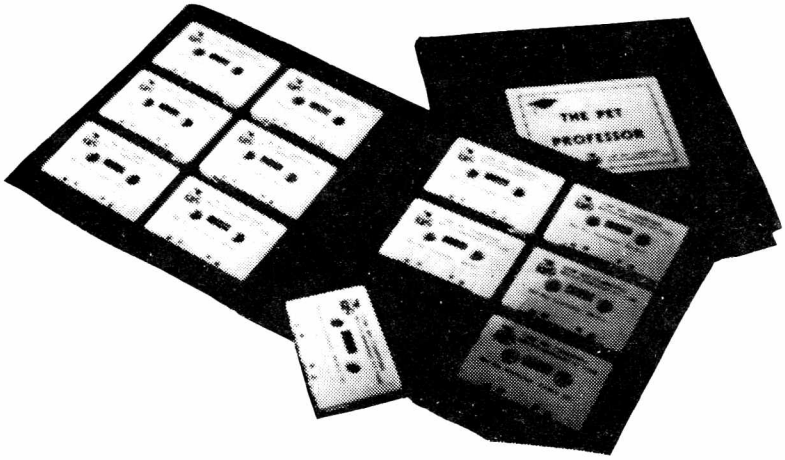
VISA

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

master charge

Prices include postage. Michigan residents include 4% sales tax. Orders must be prepaid or via bankcard (Mastercard, VISA, Eurocard, Access, etc.). Include card number and expiration date.

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



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.

KRELL SOFTWARE

COMPETENCY EXAM PREPARATION SERIES

This comprehensive set of programs consists of simulated exam modules, a thorough diagnostic package, and a complete set of instructional programs. It is designed to teach concepts and operations, provide drill and practice and assess achievement levels through pre and post testing. The Competency Exam Preparation Series provides a structured, sequential, curriculum encompassing mathematical, reading and writing instruction.

The C.E.P.S. program is designed for individual student use or use in a classroom setting. Programs provide optional printer capability, worksheet generation and performance monitoring. C.E.P.S. are available in two software formats.

National Proficiency Series $1,299.00
N.Y.S. Regents Competency Test, Preparation Series $1,299.00

If desired separate Mathematics and Verbal packages are available for $799.00 ea. A Spanish language version of the Mathematics Instruction Package is available at no extra charge.

COLLEGE BOARD PREPARATION SERIES 81/82

for TRS-80 NORTHSTAR™ PET, APPLE PDP11 OSI

Each program confronts the user with a virtually limitless series of questions and answers. Each is based on past exams and presents material of the same level of difficulty and in the same form used in the S.A.T. Scoring is provided in accordance with the formula used by College Boards.

S.A.T., P.S.A.T., N.M.S.Q.T., set includes 25 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, and Mathematics. Price $149.95

EDUCATOR EDITION - includes all of the above programs plus detailed solutions and explanations. Price $229.95

Independent Tests of S.A.T. series performance show a mean total increase of 70 points in students' scores.

Update Pack to 81/82 specs. Available to previous owners. Price $69.95

ODYSSEY IN TIME

This spectacular adventure game adds a new dimension of excitement and complexity to **Time Traveler.** Players must now compete with the powerful and treacherous adversary in their exacting quest for victory.

To succeed they must vanquish this adversary in combat that rages across 24 time periods.

Odyssey In Time includes all the challenges of Time Traveler plus 10 additional eras, including those of Alexander the Great, Emperor Asoka of India, Attila the Hun, Genghis Khan. Each game is unique, and may be **interupted and saved** for later play.

available for APPLE & TR-80 PET, 32K - $39.95

ISAAC NEWTON

Perhaps the most fascinating and valuable educational game ever devised — ISAAC NEWTON challenges the players to assemble evidence and discern the underlying "Laws of Nature" that have produced this evidence. ISAAC NEWTON is an inductive game that allows players to intervene actively by proposing experiments to determine if new data conform to the "Laws of Nature" in question. Players may set the level of difficulty from simple to fiendishly complex.

In a classroom setting the instructor may elect to choose "Laws of Nature" in accordance with the complete instruction manual provided.

For insight into some of the basic principles underlying **ISAAC NEWTON** see GODEL, ESCHER, BACH by Douglas R. Hofstadler, Chapter XIX and Martin Gardner's MATHEMATICAL GAMES column in **Scientific American,** October, 1977 and June, 1959. $24.95.

TIME TRAVELER

Confronts players with complex decision situations and the demand for real time action. Using the **Time Machine**, players must face a challenging series of environments that include; The Athens of Pericles, Imperial Rome, Nebuchadnezzar's Babylon, Ikhnaton's Egypt, Jerusalem at the time of the crucifixion, The Crusades, Machiavelli's Italy, The French Revolution, The American Revolution, and The English Civil War. Deal with Hitler's Third Reich, Vikings, etc. At the start of each game players may choose a level of difficulty . . . the more difficult, the greater the time pressure. To succeed you must build alliances and struggle with the ruling powers. Each game is unique.

$24.95

Send $2.00 for complete Catalogue.

$5.00 Discount Coupon included in Catalogue.

PROGRAMS AVAILABLE FOR
TRS-80, APPLE II & PET
(unless otherwise indicated)
☐ disk or ☐ cassette (please specify)

VISA

KRELL SOFTWARE CORP.

Send check or money order to
21 Milbrook Drive, Stony Brook, NY 11790
(516) 751-5139

All programs require 16K TRS-80 programs require LEVEL II BASIC APPLE programs require APPLESOFT BASIC

NY State Residents Add Sales Tax

The PAPER
Box 524
East Setauket, NY 11733

THIRD CLASS POSTAGE
P A I D
PERMIT NO. 96
EAST SETAUKET, NY