commodore

comments and bulletins concerning your COMMODORE PET™

The Transactor

VOL 2
BULLETIN 11

PET™ is a registered Trademark of Commodore Inc.

Bits and Pieces

WordPro and the NEC Spinwriter

Those using WordPro 3 or 4 are probably just realizing the potential of the PET as a dedicated wordprocessing system. With a Spinwriter for letter quality hard copy, this potential is substantialy increased. However, the Spinwriter requires a little preliminary set-up before it will operate correctly with WordPro. The front panel switches of the NEC are covered in the WordPro manuals but some extra switches inside the printer are not.

Inside the Spinwriter are four large circuit boards near the back of the unit. (A smaller fifth board is also there but not important here) The two boards closest to the back of the housing contain these extra switches. A word of caution: these boards support some CMOS chips... excessive static discharge to pins on CMOS chips will result in ireparable damage. You may want to have qualified personel make these changes.

On the very back board lies one of these switches. The switch, labelled 'SW1', is actually a DIP switch with 8 small slide switches on it. The second most back board contains the other three DIP switches labelled 'SW1', 'SW2', and 'SW3'. Early versions of these boards require you to pull them out of their sockets to gain access to the switches. This also means removal of a bracket and four cable connectors, two of which are tucked away at the right of the unit. Newer versions have the DIP switches placed near the top edges of the boards which will have you finished these mods in a flash. NEC assures me that both versions operate identically, only the board artwork was changed.

Now for the switch positions. Each set of 8 slide switches for the four DIPs will be labelled from left to right 1 for on and 0 for off:

Back board :	SW1 =	0 0 0 0 1 0 1 1
Next board :	SW1 =	0 0 0 0 0 0 0 0
	SW2 =	1 0 1 0 0 0 0 0
	SW3 =	1 0 1 0 0 0 0 0

Index Transactor 11

Soft Disk Device Number

```
OPEN 1, 8, 15
PRINT#1,"M-W" CHR$(50) CHR$(0) CHR$(2) CHR$(9+32) CHR$(9+64)
```

The above command sequence will change a Commodore disk unit from device #8 to device #9. This works on the 2040 (DOS 1.0), the 4040 (DOS 2.0) or the 8050 (DOS 2.5). Once executed, another logical file must be OPENed to the command channel else a ?DEVICE NOT PRESENT ERROR will occur on the next PRINT#1. Alternately, since device #8 is no longer on the bus, CLOSE 1 and reOPEN using 9 instead of 8. The disk can actually be changed to any device number by substituting the 9 in the last two CHR$'s for any number between 8 and 15. Reset (PRINT#1,"U:" or "UJ") or power up will restore to device #8.

This works best when you need two disks on line but don't want to cut the jumpers of the main logic board inside the disk. Remember though, if two disks are powered up on the bus as device #8, the above sequence will change the both to device #9.

Commodore Education Advisory Board

Commodore has now received enough educational programs to produce and distribute 4 CEAB Diskettes, with a fifth one in the works. On behalf of Commodore, the Board and the recipients, I would like to thank all who have contributed. Through you we have successfully established a software share program for learning institutions across Canada and beyond. Let's keep it going!

TPUG Minutes

Richvale Telecommunications have available cassette recordings of the Toronto PET Users Group meetings. Richvale also has CEAB programs on tape for those operating without disk. For more information contact:

Richvale Telecommunications
10610 Bayview Ave. Unit 18
Richmond Hill, Ontario
L4C 3N8
416 884 4165

Supermon Notes

To get 'long' disassemblies on your printer, find the line-count with:

```
.H xxxx,yyyy A9 16 85 B5
```

where xxxx to yyyy is the memory range of Supermon. Change the '16' value to some higher number (maximum FF) to disassemble lots of lines at a time.

If you'd like the output split into pages on your 202X printer, that's all you need do. PET printers will page after every 60 lines of output and continue printing for the specified number of lines. But if you want a 'continuous' printout without paging, you should also do a hunt for:

```
.H xxxx,yyyy 86 B9 A9 93
```

and change 93 to 13. Remember to restore the 16 and 93 values if you plan to return to "screen" monitor.

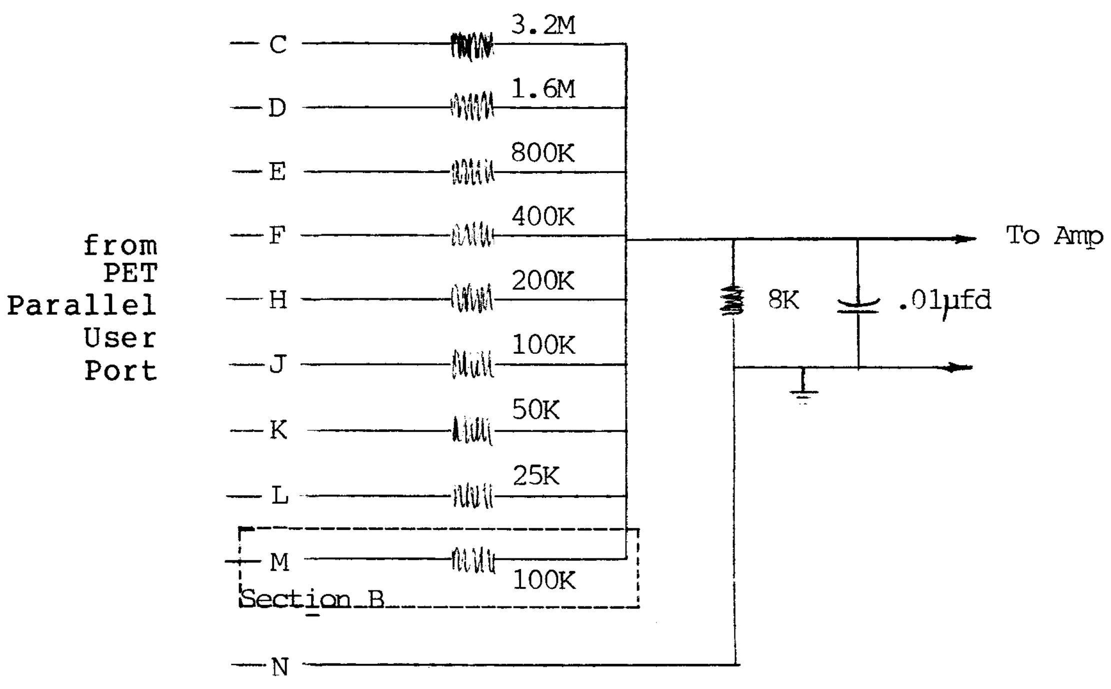
PET Sound

These next two items go hand-in-hand. The first was originaly printed in Volume 1 Transactor but, due recent inquiries, felt it worth reprinting in Volume 2. The second item is an inexpensive amplifier submitted by Tom Guzik of the Selkirk Electronics Club in Thunder Bay.

Poor Man's D/A Converter

Cheap; good for generating Chamberlin style music. Precision resistors are preferred, but most anything will generate a recognizable sound.

Section B of the diagram supports CB2 sound effects - so that this interface covers most sound requirements.

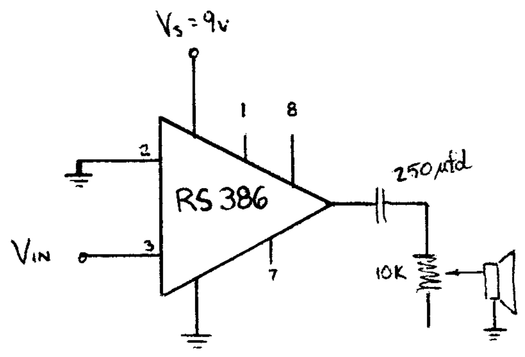


The capacitor provides some reduction of the sampling frequency (when generating music) ...tone controls on the amplifier will also help, if available.

The output of this D/A converter can be fed directly into an input of your stereo for excellent results.

500 Milliwatt Amplifier

This simple 500mw amp works on 9 volts available from pin 1 or 4 of the J11 connector inside the PET. All you need is a $2.00 I.C., a 50 cent capacitor, a spare potentiometer and a speaker.



Printer ROMs

In the last issue of The Transactor (Vol. 2, #10, pg 2) a fix was published to prevent Commodore 202X printers equipped with 04 ROMs from locking into lower case. The fix specified a delay loop to be inserted before each PRINT# to the format channel, secondary address 2. This should be changed such that the delay occurs prior to printing to secondary address 1; print data to specified format. About the simplest implementation of this would be to call the delay as a subroutine (lines 30 and 40 followed by a RETURN statement) before each PRINT# to secondary address 1 on the printer.

Computers at Waterloo

In Canada and around the world, the University of Waterloo is known for it's expertise in the computer science and engineering fields. Recently (December), the university swung into the microcomputer stream. After evaluating all the micros which were available, they chose to use the Commodore PET for their teaching and experimentation. Since their initial purchase of a Commodore PET, they have acquired over 150 units and spent thousands of man-hours working with them. The results have been tremendous. They have contributed many new innovations to the computer industry, improved upon some old ones, and set new trends in education with the micro-computer.

University of Waterloo periodicly will develop an item which is similar to an already existing product. For example, in early development, they found a need for an EPROM burner. They couldn't find one to fill their requirements so they took a couple of nights and built one. This is the philosophy which they follow at U of W. They are not in business to make money but rather, they are in the business of educating tommorrow's engineers, designers and all kinds of other professional people.

One of the first things that they undertook was a type of communications interface. It was necessary in order to have mass storage capability in conjunction with their PDP-11 mainframe. They call it ACIA (Asynchronous Communications Interface Adapter). It plugs into the memory expansion port on the PET and uses a 6551 UART (Universal Asynchronous Reciever Transmitter) with an RS-232 output.

The next project was the EPROM burner which I mentioned. It hooks into the parellel user port of the PET which communicates with an 8748 INTEL micro computer. All the information is transmitted by the PET to the processing unit which burns the EPROM. All of the Waterloo Basic chips were burned in this manner.

Because of the complexity and size of many projects they were undertaking such as Pascal and Fortran compilers, they needed much more 'core' memory. This caused the development of a 128K bank switching unit. It uses 32 - 4K ROM sockets and allows for tremendous flexability in resident languages and routines on power-up. This was the earliest version of the bank switching system and you could access any area of ROM by writing to a special control register that plugs into the memory expansion bus socket. This in turn would allow you to access a number of resident compilers such as COBOL, FORTRAN, BASIC OR APL. Newer versions of the bank switching system are 32 and 64K and plug right into the 9000 socket. Soon to follow was the development of a 64K RAM bank switcher. It was built mainly to assist in the development of newer versions of PASCAL. The system features Dynamic RAMs linked by a Motorola Dynamic RAM Controller.

Attacking the problem of available memory space from a different angle, they tried to compact the machine language code. To do this they changed the processor chip but found it would only work about 75% of the time. The problem was traced to incompatible timing signals with the new 6809 chip, but after designing a small compensator board for the new processor to plug into (which in turn would plug into the 6502 socket), it worked flawlessly. Since the 6809 chip has an expanded instruction code you can do more with less code. In some cases this amounted to 50% less coding. Although the chip is not actually any faster, it runs quicker because the instruction set is more powerful and less instructions need to be executed. With this added feature all the coding which was necessary would now fit in available ROM space.

The PASCAL compiler is running on the PET and should be totally debugged by the fall when it will be used in a programming course (CS-240). I might also add that U of W is running another excellent course (CS-250) which involves machine language programming with the 6502 chip on the PET.

They have also developed an APL character set for the PET and are currently working on the language as well as a FORTRAN compiler. All of the compilers which the U of W are working on are written in WSL (Waterloo Systems Language). WSL is a very unique language which they developed. It is an extremely high level language with the capability to do low level accessing (ie. machine language capability)

Another highlight of WSL is a software plug-in which will generate code for any micro. For example: they could write a compiler and burn EPROMs for the 6502 and a 6809 as well as numerous other processors.

In early summer the University embarked on a program to educate secondary school teachers on the micro computer in education. The program lasted for three days after which the participants were each loaned a computer and cassette to take home with them for three weeks. This allowed them to get a better feel for the machine and put into practice that which they had learned. The response was tremendous and the program a success. They are currently studying the feasibility of running a program like this indefinately as well as moving it to different locations in Canada.

In summary, the University of Waterloo is doing excellent things with education and micro computers. If they continue with micros the way in which they have approached projects in the past, they will change the pattern of education, industry and business in the world today on a scale no less than the advent of computers themselves.

Jim Butterfield
Toronto

A poor man's word processor? Not exactly, although this program can be used in that way. It's more accurately a general purpose text editor, and can create, revise and print most sequential data files.

The program is line-oriented: it gets a line, then outputs it. However, there are features that allow you to deal with smaller elements - words or characters - or larger elements such as paragraph or entire text.

Because it keeps only a line at a time, it will run on very small PETs and you won't be bothered by garbage collection delays. It's written entirely in BASIC: that makes it portable and easy to chang (say to cassette tape operation). It won't run too fast, but that's part of the plan: you'll be able to stop and correct information as you go.

For cassette tape operation, just change the OPEN statements in lines 120 and 160. If you still have original ROMs, you'll need to change SW=ST on line 410 to read IF ST <> 0 GOTO 470.

Operation

You can enter information from the keyboard and/or a file; you can write the output to either a file or the printer. Just answer the startup questions.

If you have an input file, you'll be prompted at the start and at other parts of the program run with a half-shaded character. This asks you to supply an input-mode. Your choices are:

I - Don't input; accept an insertion from the keyboard;
T - Input the entire text from the file;
S - Search; input until you receive a selected character string;
P - Input a paragraph;
L - Input a line;
W - Input a word;
C - Input a character.

Press any of the above characters, or press SPACE to continue in the same input-mode as before.

Insert mode, I, remains in force until you enter a null line, that is, a line with no characters (not even a space). At that time, you'll either prompt for a new input-mode, or quit if there's no input remaining.

If you don't have an input file or if your input file is finished, you'll go directly into Insert mode without prompting. Entering a null line will stop the program.

All input-modes except Insert can be changed while input is taking place. For example, if you're in Text mode, touch the L key for Line mode, and you'll stop at the end of the current line; or W will stop you after the next word.

When input pauses, you may press RETURN and input will resume, printing the next line, word, or whatever. Alternately, you may delete or insert text, pressing RETURN when you are finished.

If you have deleted or inserted text, you will be prompted for a new input-mode. Select it, or press SPACE to continue as before.

When you're in Paragraph mode, a deletion or insertion will signal the program that you probably want to add a paragraph to the text. In this case, pressing RETURN won't take you back to the prompt for inputmode. Like Insert mode, you can keep going until you enter a null line.

A word about null lines and blank lines. A null line, which has nothing on it, is never written. If you want a blank line to be written, you must put at least one space there. A blank line - containing one or more spaces - is used by the program to detect the end of a paragraph.

To review: a null line is not written; it's a good way of deleting a line entirely from a file. A blank line, with one or more spaces, will be written and mark the end of a paragraph.

For your convenience, the Delete key has an automatic repeat feature built in. Cursor control keys other than Delete are ignored.

As mentioned before, you can switch modes during input just by tapping a key. Input timing is rather brief, however, during Character mode or word mode. In this case it's easier to force the input-mode prompt with a "dummy" insertion: tap SPACE, then DELETE. You will have changed nothing, but the input-mode prompt will appear when you press RETURN.

NOTE : 'CL' IN SQUARE BRACKETS MEANS CURSOR-LEFT

```
100 PRINT"TEXT EDITOR     JIM BUTTERFIELD"
110 INPUT"INPUT FILE NAME   N[CLCLCL]";N$
120 IF N$<>"N"THEN M=2:OPEN1,8,3,N$
130 INPUT"OUTPUT FILE TYPE (DISK OR PRINTER)  P[CLCLCL]";T$
140 IF ASC(T$)<>68 GOTO 200
150 INPUT"OUTPUT FILE NAME";F$
160 OPEN 2,8,4,"0:"+F$+",S,W":GOTO 210
200 OPEN 2,4:U$=CHR$(17)
210 B$=CHR$(32)+CHR$(20)+CHR$(20)
220 P$=CHR$(175)+CHR$(157)
230 R$=CHR$(13)
240 S$=CHR$(32)
250 J$=CHR$(168)+CHR$(157)
260 POKE 59468,14
270 D$=R$:GOSUB820
280 IF N$="N"GOTO 480
300 REM TEST KEYBOARD FOR MODE
310 GET M$:GOSUB860
400 REM GET INPUT STUFF
410 GET#1,D$:SW=ST
420 IF D$=R$ OR (D$=K$ AND S=1) THEN GOSUB500
430 L$=L$+D$:IF D$<>S$ THEN S=1
440 PRINT D$;:IF M=6 THEN GOSUB500
450 IF D$=G$ THEN IF LEN(L$)>=H THEN IF RIGHT$(L$,H)=H$ THEN K=1:GOSUB500
460 IF SW=0 GOTO 300
470 CLOSE 1
480 M=0:GOSUB500
490 CLOSE2:END
500 F=0:S=0:REM  PAUSE FOR CHANGE/RETURN KEY EXITS
510 L=LEN(L$)
520 IF M=2 GOTO700
530 IF (M=3 OR  M=7) AND K=0 GOTO700
540 PRINT P$;
550 R=R+1:P=PEEK(151):GETC$:IFC$<>""THEN R=0:C=ASC(C$):GOTO580
560 IF P=255 OR C<>20 THEN C=0:GOTO550
570 IF R<20 GOTO550
580 IF C=20 AND L>0 THEN F=1:L$=LEFT$(L$,L-1):PRINTB$;
590 IF C=13 GOTO700
600 IF C=34 THEN PRINT CHR$(34);CHR$(20);
610 IF (C AND 127)>31 THEN F=1:PRINT C$;:L$=L$+C$
620 GOTO510
700 REM  RETURN - TEST FOR EXIT
710 IF D$<>R$ AND M>1 GOTO810
720 IF L=0 GOTO750
730 IF K=0 AND M=3 THEN FOR J=1 TO L:IF MID$(L$,J,1)=S$ THEN NEXT J:K=1
740 PRINT" ":PRINT#2,U$;L$;R$;:L$="":IF M<3 OR K=1 GOTO510
750 D$=""
800 REM CHECK FORMAT KEYS
810 IF F=0 GOTO920
820 IF M=0 GOTO920
830 PRINT J$;
840 GET M$:IF M$=""GOTO840
850 IF M$="I" THEN M=1:K$="":G$="":GOTO510
860 IFM$="S" THEN M=7:K$="":GOSUB930
870 IF M$="C" THEN M=6:K$="":G$="AA"
880 IF M$="W" THEN M=5:K$=S$:G$="AA"
890 IF M$="L" THEN M=4:K$="":G$="AA"
900 IF M$="P" THEN M=3:K$="":G$="AA"
910 IF M$="T" THEN M=2:K$="":G$="AA"
920 K=0:RETURN
930 PRINT:INPUT"SEARCH FOR";H$
940 H=LEN(H$):G$=RIGHT$(H$,1):RETURN
```

Commercial Salvaging of Information from 2040 Diskettes

Diskette salvaging should be seldom needed by the average PET user. If backup copies are made, and due care is exercised, there is little chance of losing information. Even so, there are occurences where vital information is lost and urgently needs to be retrieved, if possible.

Of course there are cases where diskettes are too badly damaged to recover. Such cases would include exposure to a strong magnetic field, physically corrupted disks (torn, folded, coffee stained, etc.), or even re-formatted diskettes. However, some forms of diskette damage can be overcome. For example, a disk that can be initialized but has an unreadable directory stands an excellent chance of being totally reclaimed. Even diskettes that can't be initialized can often be recovered.

There are now, in Toronto, 3 diskette repair stations prepared to offer this service on a commercial basis to the PET community.

Syntax Logic Design
32 Ecclesfield Drive
Agincourt, Ontario
M1W 3J6
416 498 1093
416 447 1750

Technical Data Services
19 Wagon Trailway
Willowdale, Ontario
M2J 4V4
416 497 0595

Bret Butler
17 Astoria Ave.
Toronto, Ontario
M6N 2V5
416 763 6758

Fees

Standard charges have been set up for all 3 stations. Anyone wishing diskette repair should send the diskette to one of the above addresses, amply protected for transit, and include a cheque or m/o for $25.00. Any pertinent information about the diskette would also be helpful (directory listings, WordPro files?).

Diskettes that cannot be repaired will be returned with a written report and a refund of $15.00.

Information that is recovered will be transfered to a new diskette and returned with the original and a written report.

The above applies to sequential type data only (i.e. PRG files, SEQ files or USR files). Direct access information will require custom work at an extra $25.00 or more.

It is suggested that customers call before sending any material.

HIGH RESOLUTION GRAPHICS FOR THE PET

Don White
47 Ariel Court
NEPEAN, Ontario
K2H 8J1

The Commodore PET computer is supplied with a very complete set of graphics characters which, with a little imagination, can be utilized to present extremely interesting graphics displays. The maximum resolution available for plotting is 80x50 however, and this is somewhat limiting. Fortunately, there is a readily available solution for PET users interested in high resolution graphics - the Integrated Visible Memory Board (K-1008-6) manufactured by Micro Technology Unlimited, P.O. Box 12106, Raleigh, N.C. 27605.

The features provided by this board are 1) 8K of memory for high resolution graphics and memory expansion, 2) 5 ROM sockets for preprogrammed ROM's (2332's), 3) provision for a light pen register, and 4) a KIM expansion bus for system expansion using other MTU boards. The Integrated Visible Memory considers the on-board features to be 8 devices which can be accessed by POKEing a value into the enable control register at 48895 ($BEFF). The 8 devices and the control register values required to enable them are listed in Table 1. If a number of devices are to be enabled simultaneously, the POKE value is the sum of the requisite control register values. For example, to enable the external KIM bus and both the lower and upper half of Visible Memory and disable the 5 ROM sockets you would execute POKE 48895,7. The board incorporates a bank switching feature that allows more than one device to share the same address space and thus, the enable control register allows the user to select which device is enabled to respond to common addresses at a particular time.

The board contains circuitry to display the contents of the 8K memory as 64,000 dots (320 wide by 200 high) on the PET monitor. By POKEing different values into the video control register at address 49151 ($BFFF) it is possible to select regular PET video, Visible Memory video, an overlaid image of both PET video and Visible Memory Video, or a blank screen. Table 2 lists the required POKE statements.

Installation of the Integrated Visible Memory board on an old style PET requires the K-1007-2 connector board. This board plugs onto the PET memory expansion port and provides a 60-wire ribbon cable connector that mates with the K-1008-6 ribbon cable. It also connects to the PET's video output and its monitor video input. Actual hook-up of the system is extremely simple and can be carried out in a matter of 15 minutes (after carefully reading the instructions). The K-1007-2 is plugged onto the PET memory expansion fingers, three wires are soldered to the power diodes on the main logic board, the 60-wire ribbon cable from the K-1008-6 is attached to the pins on the K-1007-2, the PET video connector is disconnected and a connector from the K-1007-2 is plugged onto the video pins on the main logic board, and the PET video connector is then plugged onto pins on the K-1007-2. At this point the system should be operational and it can be tested by running a short diagnostic routine listed in the manual. If you have a new PET, installation is essentially the same except that some of the PET's internal jumpers may have to be cut and a different connector board, K-1007-3, is required.

The dimensions of the K-1008-6 are 279 mm wide by 216 mm deep by 25 mm thick (11"x8.5"x1") exclusive of edge fingers. It can be installed inside the PET on brackets or externally in a card file, both items being available from MTU. If you intend to purchase other MTU products such as their 16K memory board (K-1016), floppy disk controller (K-1013), PROM-I/O board (K-1012), or prototyping board (K-1020) I would suggest obtaining the card file (K-1005-P) which will hold up to 5 boards.

The board is shipped configured for a new style 32K PET. The addressing of the board as shipped is as indicated in Table 3. There are three classes of jumpers on the board and they can be readily changed to meet the requirements of the user. Each of the 8 devices has its own set of address selection jumpers, allowing each device to be independently addressed on 4K boundaries. If more than one device occupies the same address space, it is important to ensure that only one device at a given address is enabled at a time. There is a second set of jumpers used to define the default enable which on power-up or reset writes the default configuration, previously set by the user, into the enable control register in order to return the system to the user defined standard configuration. An installed jumper will disable a device on power-up or reset and an ommitted jumper will enable the device. The effect of these jumpers can be overridden by POKEing the control register values (Table 1) into the enable control register, as previously mentioned. The remaining jumpers are used to enable the light pen register and to optimize the overlay of Visible Memory and PET video depending on the type of PET, old or new.

Graphics programming of the Visible Memory can be easily performed using the X-Y method (Cartesian co-ordinates) of identifying a particular dot position. If the origin of the co-ordinate system is assumed to be in the lower left hand corner of the screen then all the displayable points are in the first quadrant and X and Y are always positve. Converting from X-Y co-ordinates to a byte address and a bit number can be done as follows:

```
BYTE ADDR = VM BASE ADDR + INT((199-Y)*40)) + INT(X/8))
     BIT# = X - 8*INT(X/8)
```

The dot can then be turned on by

```
POKE BYTE ADDR, PEEK(BYTE ADDR) OR BIT#
```

or off with

```
POKE BYTE ADDR, PEEK(BYTE ADDR) AND BIT#
```

A simple example of plotting sines and cosines is given in Program 2.

You will notice in Program 2 the inclusion of some SYS commands. These call two machine language subroutines: one to clear the screen and the other to flip all the bits on the screen. The routines can be loaded using Program 1 and they are stored in the second cassette buffer.

Although it is possible to do all your plotting in BASIC it is much quicker to utilize machine language routines. If you prefer not to develop your own, two different software packages are available from MTU. The PET Graphics/Text Software (K-1008-3) is a set of utility routines written in machine language but callable from BASIC. The routines include high speed screen clear, point plotting, line plotting given X and Y endpoints, and text plotting up to 22 lines of 53 characters each. To facilitate running this package with the K-1008-6 two lines in the 'BASIC INTERFACE' require changes:

```
114 POKE 49151,2:RETURN:REM DISPLAY VM VIDEO
116 POKE 49151,1:RETURN:REM DISPLAY PET VIDEO
```

and two new lines should be added:

```
115 POKE 49151,3:RETURN:REM DISPLAY VM & PET VIDEO
117 POKE 49151,0:RETURN:REM BLANK SCREEN
```

These modifications are required because the K-1008-3 package was originally written for use with an older version of the Visible Memory.

If you acquire this package, the bit flipping routine can be added to it by including the following lines in the program 'MTU GRAPHICS':

```
1115 DATA = G138,* VMHI, 40
1160 DATA = H035,* FLIP VECT, 4C1FG1
1800 DATA = G11F,* FLIP
1810 DATA AE00H0, A000, 84J0, 86J1, B1J0, 49FF
1820 DATA 91J0, C8, D0F7, E8, EC38G1, D0EF, 60
```

To adapt this to any size PET the value of VMHI in line 1115 must be changed to the value of the top of the Visible Memory. For examaple, if your Visible Memory is addressed from $4000 to $5FFF then the value of VMHI is 60. To call the bit flipping routine the following line is added to the 'BASIC INTERFACE':

```
127 GOSUB 100:SYS(GL%+53):RETURN:REM FLIP BITS
```

One caution: If text is to be placed on the Visible Memory the bit flipping routine should be called only after the text is in place.

For VM users with 'new ROM' PETs and 16K or more of memory (excluding the Visible Memory) there is the 'Keyword Graphics Package' (KGP). This software requires 7K of memory and is assembled at three different locations: $2400-3FFF, $4400-5FFF and $6400-7FFF. When loaded, the program is linked to PET BASIC with a SYS command and it makes 47 new graphics related commands available to the user. The new commands are briefly described in Table 4. A simple example of using the KGP is included in Program 3.

Overall, I have found the MTU high resolution graphics system for the PET (I.E. hardware, software and documentation) to be of the highest quality. The Integrated Visible Memory board is solidly constructed and has given no problems after months of use, the software is easy to use and the documentation provided is exceptionally well written. Most important of all: Micro Technology Unlimited provides continuing support for their products. This is borne out by the fact that I have received two update sheets describing minor modifications that could be made to two of their products to prevent potential problems and advising the user that he could return the products to be modified in the plant if he did not feel comfortable in performing the work himself.

The MTU products are currently available in Canada through the following dealers:

Batteries Included
71 McCaul Street
TORONTO, Ontario
M5T 2X1

Conti Electronics
5656 Fraser Street
VANCOUVER, B.C.
V5W 2Z4

Kobetek Systems Ltd.
R.R.#1
WOLFVILLE, N.S.
V5W 2Z4

FUTUR BYTE INC.
1189 Phillips Place
MONTREAL, Que.

Table 1 : Enable Control Register

CONTROL REG DEVICE	VALUE	BIT NO.
External KIM Bus	1	0
Lower half of Visible Memory(top half of screen)	2	1
Upper half of Visible Memory(lower half of screen)	4	2
ROM socket 1	8	3
ROM socket 2	16	4
ROM socket 3	32	5
ROM socket 4	64	6
ROM socket 5	128	7

Table 2 : Video Control Register

POKE 49151,0	Blank the screen without erasing it
POKE 49151,1	Select regular PET video
POKE 49151,2	Select Visible Memory video
POKE 49151,3	Select combined image of PET and VM video

Table 3 : 'Standard' Jumper Configuration

DEVICE	ADDRESS	STATUS ON POWER-UP
Visible Memory Low Address	9000-9FFF	disabled
Visible Memory High Address	A000-AFFF	disabled
ROM socket 1	9000-9FFF	enabled
ROM socket 2	A000-AFFF	enabled
ROM socket 3	B000-BFFF	enabled
ROM socket 4	9000-9FFF	disabled
ROM socket 5	A000-AFFF	disabled
External KIM Bus	a) 0000-5FFF permanently assigned to PET addresses 2000-7FFF b) PET A000-AFFF = KIM 6000-6FFF PET B000-BFFF = KIM 7000-7FFF	disabled

Table 4 : KEYWORD GRAPHICS PACKAGE (KGP) Commands

COMMAND	PARAMETERS	DESCRIPTION
Display Control Commands		
PETMEM		Displays PET video.
VISMEM		Displays VM video
PVMEM		Displays combined PET and VM video
NOMEM		Blanks screen
NRMDSP		Normal display mode. Turning on point displays white dot. Turning off point displays black dot.
RVSDSP		Reverse display mode. Reverse of NRMDSP
FLPDSP		Flips display mode
GMODE	val(0,1 or 2)	1-drawing turns dots on 2-drawing turns dots off 0-drawing flips dots
RDGM	var	Reads current GMODE into variable 'var'
Screen Clearing Commands		
CLEAR		Clear VM
SCLEAR	X1,Y1,X2,Y2	Clears specified region of VM
WCLEAR		Performs 'HOME-CLEAR' on current window
SCFLIP	X1,Y1,X2,Y2	Flips specified region of VM
Boundary & Window Commands		
SETWIN	I,Xmin,Ymin Xmax,Ymax (I=0,1,2 or 3)	Creates new Window I boundary values in Memory
WINDOW	I(0,1,2 or 3)	Makes window I the current window and turns on boundary checking
BNDCHK		Turns on boundary checking
NOCHK		Turns off Boundary Checking
Graphics Commands		
WRPIX		Write pixel at the drawing cursor co-ordinates
RDPIX	var	Reads state of pixel at drawing cursor co-ords
LINE	X1,Y1,X2,Y2	Draws line from X1,Y1 to X2,Y2
MOVE	X,Y	Sets drawing cursor to X,Y
DRAW	X,Y	Draws line from drawing cursor co-ords to X,Y
DOTL	bval1,bval2	Sets parameters for dotted lines
XFFLG	I(0 or 1)	Enables or disables co-ord transformation
OFFSET	X,Y	Sets Xshift to X and Yshift to Y
SCALE	Xval,Yval	Sets Xscale to Xval and Yscale to Yval
Text Commands		
CHAR	bval or str (0 to 255)	Displays character whose value is bval or characters in string 'str'
AUTEXT	bval or str (0 to 255)	As for CHAR except that it performs CR-LF and vertical scrolling as required
CHSCALE	val (-128 to +127)	Scales characters
CHROT	val (0,1,2 or 3)	Rotates characters in steps of 90 degrees

SCROLL	Xt,Yt,X1,Y1,X2,Y2	Move the X1,Y1 corner of the region X1,Y1,X2,Y2 to Xt,Yt

<u>Cursor Commands</u>

GRACSR		Draws cross-hairs graphics cursor at drawing cursor co-ords
TEXCSR		Draws underline text cursor
RDCSR	Xvar,Yvar	Reads the transformed values of the drawing cursor co-ords into variables
RDXY	Xvar,Yvar	Reads drawing cursor co-ords into variables

<u>Miscellaneous Commands</u>

GRSHRT		Enables short form command mode
]X		Disables short form command mode
VMPAGE	bval (0 to 255)	Sets starting page of VM at bval
GKILL		Unlinks KGP from PET BASIC

<u>Shape Definition Commands</u>

CHDLOC	val	These are a set of commands that allow the user to define up to 255 shapes or characters of his own choosing. These shapes may then be recalled and drawn anywhere on the screen.
CHTLOC	val	
RDDLOC	var	
RDTLOC	var	
CHINIT	bval(0 to 255)	
CHDFC	bval(0 to 255)	
CHBLD		
CSETUP		
CHRESET		

```
10 REM      ********************************
15 REM      *                              *
20 REM      *          PROGRAM  1          *
25 REM      *                              *
30 REM      * LOADER FOR MACHINE LANGUAGE  *
35 REM      * ROUTINES TO CLEAR VISIBLE    *
40 REM      * MEMORY AND FLIP ALL BITS IN  *
45 REM      * VISIBLE MEMORY.              *
50 REM      *                              *
55 REM      *   CLEAR SCREEN - SYS826      *
60 REM      *      FLIP BITS - SYS829      *
65 REM      *                              *
70 REM      ********************************
75 :
80 :
```

```
85 REM              *** NEW OR OLD ROM  ***
90 :
95 IF PEEK(50000) THEN 150
100 :
105 REM             ***LOAD FOR OLD ROMS***
110 :
115 PRINT"[CS]CLEAR AND FLIP FOR [RVS]OLD ROMS"
120 PRINT"[HM DN DN DN DN]"TAB(16)"[RVS]LOADING"
125 FOR I = 826 TO 880
130 READ A
135 POKE I,A
140 NEXT
145 END
150 :
155 :
160 REM             ***LOAD FOR NEW ROMS***
165 :
170 PRINT"[CS]CLEAR AND FLIP FOR [RVS]NEW ROMS"
175 PRINT"[HM DN DN DN DN]"TAB(16)"[RVS]LOADING"
180 FOR I = 1 TO 55 : READ A : NEXT
185 FOR I = 826 TO 880
190 READ A
195 POKE I,A
200 NEXT
205 END
210 :
215 :
220 REM             ***DATA FOR OLD ROMS***
225 :
230 DATA  76,  64,   3,  76,  86,   3, 169,   0, 174, 111
235 DATA   3, 168, 133, 176, 134, 177, 145, 176, 200
240 DATA 208, 251, 232, 236, 112,   3, 208, 243,  96
245 DATA 174, 111,   3, 160,   0, 132, 176, 134, 177
250 DATA 177, 176,  73, 255, 145, 176, 200, 208, 247
255 DATA 232, 236, 112,   3, 208, 239,  96,  96, 128
260 :
265 :
270 REM             ***DATA FOR NEW ROMS***
275 :
280 DATA  76,  64,   3,  76,  86,   3, 169,   0, 174, 111
285 DATA   3, 168, 133,  94, 134,  95, 145,  94, 200
290 DATA 208, 251, 232, 236, 112,   3, 208, 243,  96
295 DATA 174, 111,   3, 160,   0, 132,  94, 134,  95
300 DATA 177,  94,  73, 255, 145,  94, 200, 208, 247
305 DATA 232, 236, 112,   3, 208, 239,  96,  96, 128
310 :
315 :
320 REM         ******************************
325 REM         *                            *
330 REM         * THE LAST TWO BYTES IN EACH *
335 REM         * SET OF DATA MUST BE THE    *
340 REM         * HIGH BYTE OF THE START AND *
345 REM         * END ADDRESSES OF THE VM.   *
350 REM         *                            *
355 REM         * START ADDR      DATA       *
360 REM         * ----------      ----       *
365 REM         * $2000           32,64      *
370 REM         * $4000           64,96      *
375 REM         * $6000           96,128     *
380 REM         * $9000           144,176    *
385 REM         *                            *
390 REM         ******************************
```

```
100 REM       ******************************
105 REM       *                            *
110 REM       *         PROGRAM 2          *
115 REM       *                            *
120 REM       * EXAMPLE OF HIGH RESOLUTION *
125 REM       * PLOTTING USING MTU VISIBLE *
130 REM       * MEMORY.                    *
135 REM       *                            *
140 REM       * PROGRAM USES CLR AND FLIP  *
145 REM       * ML ROUTINES AT SYS826 AND  *
150 REM       * SYS829, RESPECTIVELY.      *
155 REM       *                            *
160 REM       ******************************
165 :
170 REM     **INITIALIZE**
175 :
180 VLO = 6 * 4096            :REM VM LOW ADDRESS
185 AN  = 2 * π/320           :REM ANGLE CONSTANT
190 :
195 SYS 826                   :REM CLEAR VM
200 POKE 49151, 2             :REM DISPLAY VM
205 :
210 REM     **DRAW AXES**
215 :
220 :
225 YS = VLO                  :REM **********
230 YE = YS + 199*40          :REM *  DRAW  *
235 FOR I = YS TO YE STEP40   :REM *        *
240 POKE I, 128               :REM * Y-AXIS *
245 NEXT                      :REM **********
250 :
255 :
260 XS = VLO + 99*40          :REM **********
265 XE = XS + 39              :REM *  DRAW  *
270 FOR I = XS TO XE          :REM *        *
275 POKE I, 255               :REM * X-AXIS *
280 NEXT                      :REM **********
285 :
290 :
295 :
300 REM     **PLOT SINE CURVE**
305 :
310 FOR X = 0 TO 319 STEP2
315 Y = 99 * SIN(AN*X) + 99
320 GOSUB 500
325 NEXT
330 :
335 FOR I = 1 TO 1000 : NEXT
340 :
345 REM     **PLOT COSINE CURVE**
350 :
355 FOR X = 0 TO 319 STEP3
360 Y = 99 * COS(AN*X) + 99
365 GOSUB 500
370 NEXT
375 :
380 FOR I = 1 TO 1000 : NEXT
385 :
390 :
395 REM     **FLIP BITS**
400 :
405 SYS 829
```

```
410 :
415 FOR I = 1 TO 5000 : NEXT
420 :
425 REM       **FLIP AGAIN**
430 :
435 SYS 829
440 :
445 GET ZZ$ : IF ZZ$ = "" THEN 445
450 :
455 REM       **RESTORE PET VIDEO**
460 :
465 PRINT "[CS]"
470 :
475 POKE 49151, 1
480 :
485 END
490 :
495 :
500 REM       **PLOT POINT AT (X,Y)**
505 :
510 AD = 40 * INT(199-Y) + INT(X/8) + VLO
515 BT = 2^(7-(X AND 7))
520 POKE AD, (PEEK(AD) OR BT)
525 RETURN
```

```
100 REM           ***************************
105 REM           *                         *
110 REM           *        PROGRAM 3        *
115 REM           *                         *
120 REM           *   PLOTTING A SINE WAVE  *
125 REM           *    UTILIZING THE MTU    *
130 REM           * KEYWORD GRAPHICS PACKAGE *
135 REM           *                         *
140 REM           ***************************
145 :
150 REM ***INITIALIZATION***
155 :
160 AN = 2 * π/320                     :REM ANGLE CONST
165 CLEAR                              :REM CLEAR VM VIDEO
170 VISMEM                             :REM DISPLAY VM
175 NRMDSP                             :REM NORMAL DISPLAY
180 GMODE1                             :REM DRAWING TURNS POINTS ON
185 :
190 REM ***DRAW AXES***
195 :
200 DOT L2, 3                          :REM DOTTED LINE
205 LINE 0, 0, 0, 199                  :REM DRAW Y-AXIS
210 LINE 0, 100, 319, 100              :REM DRAW X-AXIS
215 :
220 REM ***PLOT SINE CURVE***
225 :
230 FOR X = 1 TO 319 STEP2
235 MOVEX,(99 * SIN(AN*X) + 100)       :REM MOVE CURSOR
240 WRPIX                              :REM PLOT POINT
245 NEXT
250 :
255 GET ZZ$ : IF ZZ$ = "" THEN 255
260 :
270 PETMEM                             :REM RETURN TO PET VIDEO
```

CARD PRINTING UTILITY

D. Hook
58 Steele St.
Barrie, Ontario
L4M 2E9

There are some occasions when you may wish to program a card game. By addition of a subroutine that gives good graphics (for the card symbols), most would be improved. Since this would represent too much work, the finished version fails to take advantage of the Pet's forte.

This program attempts to remove the drudgery from the task. By understanding its mechanics, I hope that you can add the feature. See me at the next TPUG meeting if you can't be bothered typing it all in.

I have included a variables cross-reference (thanks to Jim Butterfield) and a separate chart to make the graphics more easily entered.

Consult the listing as we work through the flow:

Line 10-80:
Data statements used in the initialization.

Subroutine 40000:
First seeds the random number generator. Creates the D%(array for D% decks of cards. The card values are:

0-12	A2345...K	clubs
13-25	A2345...K	diamonds
26-38	A2345...K	hearts
39-51	A2345...K	spades

These values are very important identifiers to recover the suit and value of the card in question.

Since D% is no longer needed, it is redefined to the total number of cards in the game (minus 1).

The I$(array is simply the index value to be printed in the corner of each card. These are read from Data Line 10.

The S$(array is two-dimensional. The first has 0,1,2 suit symbols read from blanks, S1$ and S2$. The second dimension refers to the suits: 0-3 in the same order as above.

The S%(array is for spot cards 1 through 10, and for rows 1 to 7 of each card to be printed. Line 20 provides the data. Note the lack of "0" entries, as the comma is sufficient.

The entries in the array indicate the NUMBER OF SUIT SYMBOLS that belong in each row. Since we are not concerned with the actual suit at this time, all the spot cards of a given value will be the same.

For the face cards, the F$(I,J) array is defined:

```
I = 1,2,3   for J,Q,K
J = 1-7     for rows 1-7
```

The data in Lines 40-80 give the strings for the card pictures. To facilitate entry of these graphics, the table below is provided:

GRAPHICS DATA LINES 40 - 80

```
Jack
 1.  " la RO  b Rv  )  b  b  "
 2.  "  b RO  < Rv  '  :  b Rv  b  b  "
 3.  "  b  ! RO  ) Rv  ) RO la Rv  b  b  "
 4.  "  b  &  )  V RO  ) Rv  &  b  "
 5.  "  b  b la RO  ) Rv  ) RO  ! Rv  b  "
 6.  "  b  b  '  P  b RO  ; Rv  b  "
 7.  "  b  b RO  )  b la  "

Queen
 8.  "  )  P  b  b  b  "
 9.  "  b RO  ) Rv  B  *  b  b  b  "
10.  "  b RO  b  b  b  b Rv  ;  b  "
11.  "  b  &  V  V  V  &  b  "
12.  "  b  < RO  b  b  b  b Rv  b  "
13.  "  b  b  b  4  ]  )  b  "
14.  "  b  b  b  L  )  "

King
15.  " la RO  b  b Rv  )  b  "
16.  "  b  b  '  b  &  B  b  "
17.  "  b RO  )  b  b  b  < Rv  b  "
18.  "  b  &  ?  ?  ?  &  b  "
19.  "  b RO  ;  b  b  b Rv  )  b  "
20.  "  b  ]  &  b  %  b  b  "
21.  "  b RO  )  b  b la  "
```

Code:

"	=	quote
b	=	blank
Rv	=	reverse on
RO	=	reverse off
la	=	shifted left arrow

All other keys are their "shifted" equivalents (i.e. graphics).

We now return to the main-line program.

Three subroutines are offered in the menu:

The DISPLAY CARDS is simply for use in de-bugging, and need not be part of any program that you may use.

The SHUFFLE is an integral part of any game program, and is both compact and fast.

The SUBROUTINE FOR GAMES allows the many options that are essential to the utility of this program.

DISPLAY CARDS--SUBROUTINE 42000

Virtually all of this is duplicated in Subroutine 43000, where the main discussion will take place.

The purpose is to print (on the sceen) the pictures for all the cards used in the game. Line 42020 defines the starting line, L%=7, sets up to print, A%=5, cards across the screen, and will start printing at tab, TB=0.

Variable L is the loop counter to print from card C%=0 to C%=D%, the last card of the last deck. Since no shuffling takes place, you may see the deck(s) flash by. Starting at the A of clubs, the K of spades will be the last, regardless of the number of decks selected.

SHUFFLE--SUBROUTINE 41000

Some sort routines require an extra array to store the intermediate values. Others require a pointer array to flag the cards already taken.

No such precautions need be taken here. The array is sorted in place. A card already chosen will not be shuffled again, so the process takes only N-1 passes for N cards. If you haven't seen this before, follow the logic below:

The loop variable is I, for the D% cards. Variable J% provides a random number from 0 to D% on the first pass. Thus all cards are available to be selected.

Assume we have one deck, so D% is 51. Assume the random number, or J%, is 14. Let's say that card number 14 is the A of clubs. In our deck, the card value is 0 (see above).

Define K%=D%(14), which means K%=0 this time.

Now comes the exchange, where we take the last value, D%(51), and put it into D%(14). (We haven't lost D%(14), since it is stored in K%).

Put K% into the last position, or D%(51), and the first pass is complete.

If you have observed that the loop counter, variable I, appears throughout, you may see what happens next.

As NEXT I is reached for the next pass, the upper boundary shrinks by one. The (former) last entry cannot be chosen for J%, nor be part of the subsequent exchange.

Each pass gets another card, and the deck(s) get shuffled. A pretty tidy routine!

SUBROUTINE FOR GAMES--SUBROUTINE 43000

Initially you are asked to respond to a series of questions which will establish the various variables to be used in your game. These prompts are only to provide a cue for your usage, so lines 43000 to 43040 can be dropped. Be advised that you will have to provide for these to be defined in your program.

P% is the total number of cards to be printed. Make it a large number if you plan to play your game for a while.

L% is the screen line number where the card is to be printed. The cards are 9 rows high, so watch where you start. You may define this differently before each subroutine call.

A% is the number of cards across the screen. The tab values are reset based on this value. Since the cards are 7 spaces wide, only 5 cards may fit across the screen. This too may be changed before calling.

TB% is the tab position for the first card on the line. Note that if 5 cards are printed, you must start at TB%=0. Whenever A% is checked, the tab position advances by 8 positions (Line 43150). Change this line if you want wider spacing between the cards.

M% is the variable to detect when to reshuffle the whole deck(s). Line 43040 sets this to the whole deck(s), so keep this if it suits you. Otherwise redefine it to a convenient number, based on the number of decks in play.

Note that this routine does not give an automatic first shuffle when the "game" begins. Do this yourself with a call to SBR 41000.

On to the meat of the routine:

Line 43100:
Initializes the card counter, C%, to select the next card from the deck. This is an index to the actual card array, D%(. The next check is to see whether it is time to shuffle--if it is, then the shuffle is done.

Please observe that I have included the "L" loop as part of the subroutine. In your game this would undoubtedly be part of the main code. It has been done this way to allow printing as part of this program.

Line 43130:
You will recall that our deck consists of coded (0-51) values to represent the cards. Here we extract the suit into variable S%=0,1,2,3 and the card value into variable V%=1-13 (A23...K).

Since you will want to employ these values in your game, you have them on return to your main routine.

Line 43140:
Checks to see if it is time to print back in the "first" location again. Depends on the afore-mentioned values for A%, TB%, L%. Recall that variable "L" simply is the counter for the total number of cards printed.

Line 43150:
Tabs ahead 8 spaces horizontally and moves the cursor up. Only used where several cards are to appear on the same screen line. Change as described above, if you wish.

Line 43160:
The top line of every card has its "index" name (upper left corner) and is filled out with blanks. We then enter the loop to print the next seven rows down.

Line 43170:
If the card is a face card, branch around to Line 43500.

Line 43250:
This part gets tricky...use the value of the card, V% and the row number,J as an index into the S%(array. That array will give you the number of suit symbols (0,1,2) to be printed on a given line for the card.

Then combine that with the suit variable, S% to determine which symbols to put on that line. The array, S$(gets the right ones to print.

For spot cards, this is repeated for each of the seven rows.

Lines 43500,43510:
For face cards, we need to print the proper suit symbol near the upper left and the lower right. If J=1 or J=7 then this is done at the start of each of these lines.
Then we look at array F$(to get the card value, V%-10 and the row number, J.

Line 43520:
If J<>1 or J<>7 then we just do the second half of the above.

We loop 7 times then print the bottom line of the card. The lower right corner also contains our "index" name.

Line 43800:
Since we are printing "L" cards, we cannot forget this loop. This, like the FOR in Line 43100 should be in your main program.

CROSS REFERENCE - PROGRAM CARD UTILITY

```
A%      42020  42030  43020  43140
C%      42000  42010  43100  43130
D%      40000  40010  40020  41000  41010  42000  43040
D%(     40020  41000  41010  42010  43130
E9      40000
F$(     40080  42500  42510  42520  43500  43510  43520
I       40000  40020  40030  40050  40060  40070  40080  41000  41010
I$(     40030  42050  42750  43160  43750
J       40000  40020  40070  40080  42050  42070  42500  42510  42520  42750  43160
        43250  43500  43510  43520  43750
J%      40000  41000  41010
K%      40000  41000  41010
L       42000  42030  42800  43100  43140  43800
L%      42020  42030  43010  43140
M%      43040  43100
P%      43000  43100
S$(     40050  40060  42070  43250
S%      42010  42070  42500  42510  43130  43250  43500  43510
S%(     40070  42070  43250
S1$     40040  40050
S2$     40040  40060
T%      42030  42040  42050  42070  42500  42510  42520  42750  43140  43150  43160
        43250  43500  43510  43520  43750
TB%     42020  42030  43030  43140
TI      40000
V%      42010  42050  42060  42070  42500  42510  42520  42750  43130  43160  43170
        43250  43500  43510  43520  43750
Z         130    140    150  15010  43000  43010  43020  43030  43040
Z$        120    130    160  15000  15010
```

Simple 8010 Modem Programs

Jim Butterfield
Toronto

The programs that come with the Commodore 8010 modem may not quite fit your needs. For one thing, a NULL character from the line will cause the program to stop, since the input arrives as a null string and the ASC function won't work. If you communicate with computers that send parity - an extra bit intended to safeguard transmission - you'll get some funny looking things on your PET screen.

Speed is of the essence in this kind of Basic program: waste a few moments and you may lose an incoming character. As a result, the programs are no-frills. Watch carefully for timing if you try dressing them up with your own features.

PET TO ASCII

We need to translate PET's internal code to ASCII, and vice versa; and we need to do it fast. Result: an array for quick translation each way. F(x) translates incoming characters from the line; T(x) translates to the line.

Most non-printing characters are dropped; I've preserved only the carriage return, CHR$(13), and the Delete, CHR$(20) to PET and CHR$(8) to the line. If your favorite computer needs other special characters, you may put them in the table: for example, if the computer recognizes Data Link Escape (DLE, sometimes called Control-P), you could code it as shifted-zero on the PET keyboard with: 250 T(176)=16.

The POKE 1020,0 on line 280 is needed for the new 4.0 systems to ensure that IEEE timeout works properly.

PET to PET

Much simpler, of course, since no translation is needed. Delete the POKE 59468, 14 (or change it to POKE 59468, 12) if you want to stay in graphics. This way, you can draw pictures on the other PET's screen.

All of the cursor controls and graphics work, of course. You can even clear the opposite screen remotely, if you wish.

For communications to an ASCII system:

```
100 REM     8010 INTERFACE      JIM BUTTERFIELD
110 REM     FOR ASCII LINES
120 REM     SET SWITCH TO HD
200 DIM F(255), T(255)
210 FOR J=32 TO 64 : T(J)=J : NEXT J : T(13)=13 : T(20)=8
220 FOR J=65 TO 90 : K=J+32 : T(J)=K : NEXT J
230 FOR J=91 TO 95 : T(J)=J : NEXT J
240 FOR J=193 TO 218 : K=J-128 : T(J)=K : NEXT J
250 REM    ADD EXTRA FUNCTIONS HERE
260 FOR J=0 TO 255 : K=T(J) : IF K THEN F(K)=J : F(K+128)=J
270 NEXT J
280 POKE 1020, 0 : POKE 59468,14
290 OPEN 5,5 : PRINT "ASCII I/O READY"
300 GET A$ : IF A$ <> "" THEN PRINT#5, CHR$(T(ASC(A$)));
310 GET#5, A$ : IF ST=0 AND A$ <> "" THEN PRINT CHR$(F(ASC(A$)));
320 GOTO 300
```

For Communications to Another PET:

```
100 REM     8010 INTERFACE      JIM BUTTERFIELD
110 REM     FOR PET INTERCOMMUNICATION
120 REM     SET SWITCH TO HD
280 POKE 1020, 0 : POKE 59468,14     If text mode desired
290 OPEN 5,5 : PRINT "PET I/O READY"
300 GET A$ : IF A$ <> "" THEN PRINT#5, A$;
310 GET#5, A$ : IF ST=0 THEN PRINT A$;
320 GOTO 300
```

Editor's Note

We're looking into the possibility of downloading PET programs using a simple BASIC driver. Attempts thus far have failed, mostly due to the fault of the driver. The task may require a little machine language, but we'll keep you posted.

An Incomplete PET/CBM Bibliography

Hands On Basic With A PET Herbert D. Peckham
McGraw-Hill Ryerson 330 Progress Ave., Toronto Ont., M1P 2Z5

Some Common Basic Programs PET/CBM Edition ed. L. Poole
Osborne/McGraw-Hill 630 Bancroft Way, Berkeley, Ca. 94710

Practical Basic Programs ed. L. Poole
Osborne/McGraw-Hill 630 Bancroft Way, Berkeley, Ca. 94710

6502 Assembly Language Programming Lance A. Leventhal
Osborne/McGraw-Hill 630 Bancroft Way, Berkeley, Ca. 94710

Pet and the IEEE 488 Bus (GPIB) Eugene Fisher / C.W. Jensen
Osborne/McGraw-Hill 630 Bancroft Way, Berkeley, Ca. 94710

PET/CBM Personal Computer Guide * C.S. Donahue & J.K. Enger
Osborne/McGraw-Hill 630 Bancroft Way, Berkeley, Ca. 94710

Care and Feeding of The Commodore PET
Elcomp Publishing 3873-L Schaefer Ave. Chino Ca. 91710

8K Microsoft BASIC Reference Manual
Elcomp Publishing 3873-L Schaefer Ave. Chino Ca. 91710

The Pet Subroutine Library * Nick Hampshire
Computabits Ltd. P.O. Box 13, Yeovil, Somerset, UK

The Pet Revealed 2nd ed * Nick Hampshire
Computabits Ltd. P.O. Box 13, Yeovil, Somerset, UK

6502 Software Design Leo J. Scanlon
Howard W. Sams 4300 West 62nd St., Indianapolis, Indiana 46206

Programming & Interfacing the 6502, With Experiments Dr. De Jong
Howard W. Sams 4300 West 62nd St., Indianapolis, Indiana 46206

Programming the 6502 Rodney Zaks
Sybex 2344 6th St. Berkeley, Ca.

6502 Games Rodney Zaks
Sybex 2344 6th St. Berkeley, Ca.

6502 Applications Book Rodney Zaks
Sybex 2344 6th St. Berkeley, Ca.

Microprocessor Systems Engineering R.C.Kemp/T.A.Smay/C.J.Triska
Matrix Publishers 30 N.W. 23rd Place Portland, Oregon 97210

Practical Microcomputer Programming: The 6502 W. J. Weller
Northern Technology Books Evanston, Il.

Programming a Microcomputer: 6502 Caxton C. Foster
Addison-Wesley Publishing 36 Prince Andrew Pl. Don Mills Ont.

PIMS
Scelbi Publishing Co. P.O. Box 133 PP Stn Milford , Ct. 06460

Pet Machine Language Guide Arnie Lee
Abacus Software P.O. Box 7211 Grand Rapids Mich. 49510

Understanding Your PET/CBM Vol. 1
TIS Workbooks: Getting Started With Your PET
PET String and Array Handling
PET Graphics
PET Cassette I/O
Miscellaneous PET Features
PET Control and Logic
TIS P.O. Box 921 Los Alamos, Ca.

PERIODICALS

The Transactor
Commodore Bus. Mach. 3370 Pharmacy Ave. Agincourt, Ont. Canada

CPUCN Newsletter (UK)
Commodore Systems Ltd. Slough Trading Estate 818 Leigh Rd.
Slough Berkshire, England SL1 6BB

Commodore U.S. Newsletter
Commodore Systems Ltd. 3330 Scott Blvd. Santa Clara, Ca. 95051

Compute, The Journal of Progressive Computing
P.O.Box 5406 Greensboro, NC 27403

Cursor
The Code Works Box 550 Goleta, Ca 93017

The PAPER
P.O. Box 524 East Setauket, New York 11733

PET Prime c/o Guy Leger
Metro Separate School Board 146 Laird Dr. Toronto, Ont. M4G 3V8

Kilobaud MICROCOMPUTING 1001001 Inc.
Peterborough , NH 27403

Micro: The 6502 Journal
P.O. Box 6502 Chelmsford, Ma 01824

Creative Computing
P.O. Box 789 Morristown, NJ 07960

Practical Computing
IPC Business Press Oakfield House, Perrymount Rd.,
Hayward Heath, Sussex RH16 3DH, UK

Printout
Greenacre, North ST. , Sussex RH16 3DH, UK

PET Benelux Exchange (Dutch text)
Burgemeester Van Such Telenstraat 46 7413 XP
Deventer, Holland

NOTES

The above books and periodicals cover a wide range of information and topics. The PET masochist reader will want (or already has them all) to include them in his or her library. The novice will find several elementary books. * Available from Commodore dealers.

The periodicals are directly related to the PET (or have significant monthly columns). The British magazines are usually available in large Metropolitan areas.

DOES SCHOOL MATHEMATICS LEAVE YOU (OR MEMBERS OF YOUR FAMILY) STRUNG OUT? IF SO, GET ON-LINE!!

Seneca College is initiating a pilot project to assess the feasibility of offering on-line Computer Assisted Instruction to owners of personal computers. If you have (1) access to a compatible acoustic coupler, (2) a telephone line that can be dedicated to CAI for significant periods of time, and (3) a member of your family who would like to upgrade his/her math skills, we can talk turkey.

Whether you (or members of your family) need developmental work in secondary-level math, remedial work, or just a little brush-up of previously-attained skills, Seneca's Centre for Independent Learning can probably be of help. We offer three different courses in mathematics spanning the range from Intermediate grade level to Grade 12 as follows:

Intermediate Arithmetic - Addition, subtraction, multiplication and division of whole numbers, factoring, roots of perfect squares, exponents, order of operations, addition, subtraction, multiplication and division of decimal numbers; addition, subtraction, multiplication and division of fractions; interconversion of decimal fractions and common fractions; two-term ratios and percent.

Integers and Rationals - Addition, subtraction, multiplication and division of integers; addition, subtraction, multiplication and division of rational numbers in fraction form; addition, subtraction, multiplication and division of rational numbers in decimal form. This course assumes competency in the skills covered by Intermediate Arithmetic.

Prerequisite Mathematics - Operations with signed integers; signs and common fractions; complex fractions; decimal notation and operations; percentage; laws of exponents; algebraic substitution; operations with monomials and polynomials; ratio, proportion and variation; linear equations; quadratic equations; scientific notation; significant digits; logarithms; formula manipulation. This course assumes the ability to perform basic operations with whole numbers and is the most advanced of the three courses listed here (in spite of its name.)

What are the fees? Zilch, nil, zero -- but there is a trade-off (see next paragraph.)

What is required of participants? In exchange for free access to the courses listed above, we expect your assistance in evaluating this pilot project. There will be questionnaires to fill out and perhaps a group meeting to attend (to ensure that we learn what we need to learn in order to make a decision concerning the later expansion of this service.)

What is the time-frame? Assuming timely distribution of this newsletter and a timely response from you, the Seneca HomeCAI pilot project can begin in early December and is expected to continue for at least two months, giving all participants time enough to complete their courses and evaluate our services.

If you are interested in participating actively in Seneca's pilot project, please complete and return the Form on the opposite side of this page. (Note: In the case of an overwhelming response to this offer, we reserve the right to limit the number of participants in the Seneca HomeCAI pilot project.)