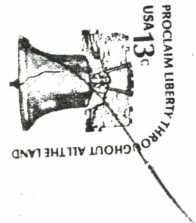
PET User Group
PO Box 371
Montgomeryville, PA
18936

FIRST CLASS



David Gordon
16956 Tupper St.
Sepulveda, CA 91343

PET User Notes

Publication of the PET User Group P.O. Box 371 Montgomeryville, PA 18936

Volume 1 Issue 2 January 1978

General Notes

By now, I'm sure all PET owners have received their copy of "An Introduction to your new PET" from Commodore. Although the manual is not a step by step guide to turn you into a computer professional, at least it is a start. It does include a memory map and edge connector descriptions for the interfaces. Commodore intends to have more detailed manuals available. These will be: User Manual; BASIC Manual; Technical Manual; Service Manual (for authorized repair personnel). The User Manual should be ready in about 3 months, with BASIC shortly thereafter.

A number of people wrote saying they had received a 4K PET. The 4K unit apparently only needs a jumper and more RAM chips to become an 8K unit (I even heard about a 4K owner and an 8K owner sharing resources and each running 6K until RAM's are available). Commodore has now stopped producing 4K units on an indefinite basis.

A Monitor tape is still in the works from Commodore for all PET purchasers.

Commodore now has the second cassette drive ready. I believe the price is $99.95, but you should check with them before ordering. No dates yet for other peripherals (floppy disk, printer, memory, modem still planned).

A number of magazines have featured articles about the PET. One of the most informative has been People's Computers. Their Nov - Dec 1977 issue had an excellent drawing program for PET, and the forthcoming March-April 1978 issue will have an enhanced version which will include capability to save a PET CRT drawing.

Hardware

We have heard from several organizations planning PET compatible hardware. Forethought Products (Box 386, Coburg, OR 97401) indicated they are working on a PETSI (you have probably seen their KIMSI, a KIM-1 to S-100 Motherboard). HUH Electronic Music Productions (Box 259, Fairfax, CA 94930) announced the PET's 100 to interface to S-100 bus. Microcomputer Associates Inc. (2589 Scott Blvd., Santa Clara, CA 95050) said they have plans for a complete line of PET IEEE-488 hardware as well as a PET to S-100 expansion card.

The Net Works (5014 Narragansett #6, San Diego, CA 92107) has announced their TNW-488 Low Speed Modem Module to interface IEEE-488 (PET connector version) to the telephone network using the Bell 103A standard. Doug Gage, one of TNW's proprietors sent preliminary announcement specs and some documentation. He also said they had a prototype running for some time, and are now producing the

PET User Notes published 6 (or more) times a year by Gene Beals, PET User Group, Box 371, Montgomeryville, PA 18936. Membership/subscription on annual basis is $5 (U.S.) in U.S. and Canada, and $10 in other countries.

first units at $225 assembled. This sounds like a great device to me -- I can see access to computer networks, program and data exchange, and personal communications using the telephone network and my PET.

I also received a note from Mr. Richard Rosner of Connecticut Microcomputer (150 Pocono Rd., Brookfield, CT 06804) announcing a PET-488 to RS-232 interface for a printer. He has a prototype version running a GE Terminet 300. The assembled board will be $98.50 (without case or power supply). The device is not intelligent, and recognizes everything coming out as its data. Mr. Rosner will probably have an intelligent device with multiple ports at a later date.

Software

Several organizations have announced software for the PET. I have only seen programs from two of them to date, however. Don Alan Enterprises (Box 401, Marlton, NJ 08053) has a 10 program cassette of games and demos which are nicely done as well as being informative in terms of using the CRT. Personal Software (Box 136, Cambridge, MA 02138) also has programs adapted for PET. Although I only saw a few of them, they appeared to be very nicely done (if you aren't careful, the Kingdom simulation can trap you for hours at a time).

Both of the above organizations are looking for additional software authors on a royalty basis. If you wish to have them market your software, contact them (Commodore, also, intends to market software on a royalty basis). The contact at Personal Software is Dan Fylstra, Don Alan is Wayne Zarfos, and Commodore is Terry Laudereau.

Also received an announcement from C.R. Lufkin, Softbyte, 315 Dominion Dr., Newport News, VA 23602 stating that a Federal Income Tax program was available on PET cassette for $16.60.

Software Exchange

If you are interested in software exchange, please let me know how it should be handled. I have had several suggestions so far:

1. Run the exchange in a similar fashion to DECUS -- we would distribute contributed programs for the cost of a cassette plus a small charge per program copied onto the cassette. This would be about $2.00/(cassette and mailing) plus about 50¢ per program copied (at this price, I don't think I would take the time to do much verification).
2. Distribute fixed packages of programs on cassette. This would allow high speed copying of a cassette and save a lot of our time. Estimated cost for this would be $3-4 for a cassette of programs and mailing. I believe the Altair User Group (Altair Software Distribution Center) uses this approach for some of their distribution.
3. List names and addresses of authors along with abstracts of software they wish to exchange. Others could then contact the author to exchange whatever (software?) they wish. This method would be easier on me, but obviously has drawbacks for people with no programs to exchange.
4. Print programs of general interest in the User Notes. With the next issue, I plan to do more of this. I also hope to get together enough money for a printer connected to PET so that listings will be a little easier.
5. As new hardware becomes available, other means could be added (floppy disk, computer-printed listings, or telephone network as modems become available).

We are not limited to a single approach, and any comments or suggestions you have would be welcome.

In conjunction with method 3, Mr. Evan Foreman, PO Drawer F, Mobile, AL 36601 has programs for 4K PET: GUESS; BRACKETS; COMPU-ART; AIR ACE; and LUNAR LANDER (no descriptions). He wishes to trade one or more of these programs, and will return them via cassette to anyone sending him a like number of programs on cassette.

PET Errors and Bugs

Although several people have written that they had problems with their particular machine, the PET in general has been remarkably free of BASIC and Operating System bugs. Some of the bugs that people have written about:

1. Array addresses are calculated modulo 256 (0-255 size limit).
2. SPC(0) results in SPC(256)
3. RIGHT$(A$,0) causes ILLEGAL QUANTITY error message.
4. RND(0) gives the same number each time it is called, not the same sequence. RND for negative integer numbers (-1,-5,etc.) returns value greater than 1, rather than between 0 and 1. RND for negative arguments does strange things -- i.e. -.01,-.02,-.04,-.08,-.16, etc. all return the same number: -.03,-.06,-.12, -.24, etc. return same number; apparently negative fractions times a power of 2.
5. When writing lots of elements to cassette file, the file cannot always be read back properly (see notes on cassette usage).

Commodore has fixes for most of these, and expects to make new ROM's available for sale to existing PET owners.

A number of people have also written about mechanical or electronic problems. The most common are: CRT is too bright; SPACE key needs to be struck repeatedly (or on left side, exactly in middle, etc.); RETURN key has problems; and cassette tapes created on any PET are not always readable by another PET.

Miscellaneous

Michael Erlewine (editor of MATRIX, a magazine publishing computer oriented material of interest to astrologers/astronomers, 1041 N. Main St., Ann Arbor, MI 48104) sent a note suggesting a standard for typed program listings (especially since cassette tape compatibility for PET is temporarily bad, he noted). He suggests denoting graphic characters with the regular ASCII character for the key preceded by $. Thus a "heart" symbol would be $S.

Sounds good, but something should be added for cursor and screen control characters, and for lower case and reverse field characters as well. I tried using lower case abbreviations for typed listings for cursor and screen: cs/clear screen, h/home, cu/cursor up, cd/cursor down, cl/cursor left, cr/cursor right. Any further suggestions?

We received a lot of letters from people wanting to hook up another keyboard. A copy of the PET keyboard matrix is included for anyone wishing to pursue this.

This issue is a little late getting printed and mailed -- will get the next one out sooner. If you have anything you want to share or have included, please send it. I will do my best to get a printer hooked up so we can list more programs.

PET KEYBOARD MATRIX

! A1	" A2	# B1	$ B2	% C1	' C2	& D1	\ D2	(E1	) E2	← F1						
Q A3	W A4	E B3	R B4	T C3	Y C4	U D3	I D4	O E3	P E4	↑ F3						
A A5	S A6	D B5	F B6	G C5	H C6	J D5	K D6	L E5	: E6	F5						
Z A7	X A8	C B7	V B8	B C7	N C8	M D7	, D8	. E7	? E8	RETURN F7						
SH A9	RVS A10	@ B9	[B10	] C9	SPACE C10 D9		< D10	> E9	STOP E10	SH F9						

HOME G1	↓ G2	↑ H1	DEL H2						
7 G3	8 G4	9 H3	/ H4						
4 G5	5 G6	6 H5	* H6						
1 G7	2 G8	3 H7	+ H8						
0 G9	. G10	- H9	= H10						

20 Pin Female Connector
Brown MX-5 2695 From Keyboard

CHAINING, or How To Run Programs Too Large For Your Memory

by Dan Fylstra, 22 Weitz St., Boston, MA 02134

What can you do when your program is too long for your 4K or 8K of user memory? One alternative is to read text messages or initial data from a cassette data file rather than, say, DATA statements. If the program text itself is very long, however, you may have to resort to chaining, or having one program load another program from cassette to continue execution. You can do this on the PET and preserve data values in memory from one cassette load to another, provided that you watch out for a few restrictions. The basic idea is simple, however: when you reach the point in the program logic where the next cassette load is needed, just include a statement such as:

```
100 LOAD "NEXTPART"
```

After loading NEXTPART, the PET seems to continue execution from the first BASIC statement loaded. All BASIC statements previously in memory are erased, but data values are preserved (if you're careful). You can chain to additional program segments with LOAD statements in the newly loaded program.

The basic restriction you must obey is that the largest program segment must be loaded first in order to preserve data values. This is because variables and arrays are allocated memory space immediately following the BASIC program at the time execution is begun, so loading a longer BASIC program in place of the existing one would wipe out some of the data values.

Another restriction is more subtle, since it depends on the implementation of strings. A string variable such as A$ is really a pointer to a character string somewhere in memory. Many character strings are actually stored within the BASIC program itself. So if you execute a statement such as:

```
10 A$="THIS IS A STRING"
```

the variable A$ will thereafter point to a constant string "THIS IS A STRING" within the BASIC program. If you load a new BASIC program in place of the old one, "THIS IS A STRING" will be erased, and A$ will be left pointing at something invalid. To avoid this problem, you can write:

```
20 A$=A$
```

which is deliberately implemented so that "THIS IS A STRING" will be copied to a "safe place" (the free string area in the upper part of memory), and A$ will be left pointing to this copy.

Another point to remember is that any DIM statements should be executed only once; they should not reappear in the newly loaded program segment, or you'll get a REDIM'ED ARRAY ERROR. The following program, which you can run to try out program chaining, includes logic to avoid re-executing the DIM statement. The first time the program is loaded and run, each A(I) will contain 0 and each A$(I) will contain a null string, which prints as nothing. After the program is reloaded, however, it should print two columns of the first 20 integers and their squares.

```
 10 IF A THEN 40
 20 DIM A(20),A$(20)
 30 A=-1
 40 FOR I=1 TO 20
 50 ?A(I),A$(I)
 60 A(I)=I: A$(I)=STR$(I*I)
 70 NEXT I
 80 IF B THEN STOP
 90 B=-1
100 LOAD "PROG"
```

You should either SAVE "PROG" twice, or SAVE it once and be prepared to rewind the tape during program execution so that PROG can be found and loaded again. You'll notice that the message PRESS PLAY ON TAPE #1 is displayed if PLAY is not pressed on the cassette drive, but the messages FOUND PROG and LOADING are suppressed. In the above example, there are no constant strings assigned to variables to worry about; the statement A$(I)=STR$(I*I) causes A$(I) to point to a newly created string such as "1", "4", etc. in the free string area.

PET Cassette File Usage

by Gene Beals

Many people have written asking about using the cassette for data storage. I have tried to summarize my experiences with my own PET in the process of writing a small inventory-bill of materials program, and have included some material from Commodore as well.

First, the BASIC file commands:

OPEN [logical file 1-255] [, device number 1 for cassette #1; 2 for cassette #2] [, I/O option 0 for read; 1 for write; 2 for write with EOT] [, "FILENAME"]
All parameters after Logical File are optional. Device Number defaults to 1, I/O Option defaults to 0, and FILENAME defaults to null. Prior parameters cannot be omitted (OPEN 1,,,"INFILE" doesn't, but OPEN 1,1,0,"INFILE" does). A maximum of 10 files may be open at once.

CLOSE [logical file 1-255]
Closes file for whichever I/O option you OPEN'ed. If you were writing, the CLOSE writes the last buffer and then an end-of-file (EOF) mark. If option 2 was OPEN'ed, an end-of-tape (EOT) mark is written as well. Don't forget the CLOSE if writing.

PRINT# [logical file 1-255], variable
?= will not work, and I have not been successful with INPUT# when more than one variable per PRINT# is used.

INPUT# [logical file 1-255], variable list
The input buffer limit for PET is 80 characters. If using INPUT#, make sure PRINT# terminates with carriage return rather than continuation (;), or at least don't let continuation go beyond 79 characters.

GET# [logical file 1-255], [string or numeric variable]
Reads tape 1 character at a time. If numeric variable, reads only characters 0-9 and +,-, blank. The latter three result in a 0 value.

ST -- Status Word
(ST) is set after every I/O operation with bit or bits corresponding to the following conditions:

4	short block
8	long block
16	unrecoverable read error
32	checksum error
64	end-of-file (EOF)
128	end-of-tape (EOT)

Following are some examples of cassette file usage to show how these commands work.

```
10  X$="DATA"
100 OPEN 1,1,1
110 FOR N=1 TO 15
120 PRINT #1,N
130 PRINT #1,X$
140 NEXT N
150 CLOSE 1
160 ?"REWIND & PRESS ANY KEY"
170 GET X1$:IF X1$="" THEN 170
180 OPEN 1
190 FOR N=1 TO 15
200 INPUT #1,X,Z$
210 PRINT X,Z$
220 NEXT N
230 CLOSE 1
240 PRINT "END OF DEMO"
250 END
```

Again, I could not use PRINT#1,N,X$ because the INPUT# would give errors. Multiple elements for a INPUT# seem to work, but be careful not to exceed 80 characters including carriage returns. Each PRINT# results in a carriage return unless it ends with a semicolon.

```
10   X$="DATA"
100  OPEN 1,1,1:SZ=0
110  FOR N=1 TO 500
120  PRINT #1,N: T$=STR$(N):GOSUB 1000
130  PRINT #1,X$: T$=X$: GOSUB 1000
140  NEXT N
150  CLOSE 1
160  PRINT " REWIND & PRESS ANY KEY"
170  GET X1$: IF X1$="" THEN 170
180  OPEN 1
190  FOR N=1 TO 500
200  INPUT #1,X,Z$
210  PRINT X,Z$
220  NEXT N
230  CLOSE 1
240  PRINT "END OF DEMO"
250  END:REM **************************
1000 REM *****CHECK BUFFER FULL ******
1010 SZ=SZ+LEN(T$)+1
1020 IF SZ<192 GOTO 1100
1030 POKE 59411,53
1040 T=TI
1050 IF (TI-T)< 6 GOTO 1050
1060 POKE 59411,61
1070 SZ=SZ-191
```

The above program segment is almost the same as the previous example. It does, however, write more elements to the file, and because of this requires a check subroutine to circumvent a temporary system error. When writing to a tape, PET blocks characters in a 191 character length buffer, and physically writes a 191 character block to tape when the buffer is full (or when a CLOSE is executed -- remember to CLOSE to get the last partial block out and write EOF mark). When multiple data blocks are written, not enough space is left on the tape between blocks, resulting in a later INPUT# error.

The example checks when a physical record of 191 characters is written, and turns on the tape motor briefly (6 TI increments -- 0.1 seconds) to create a physical

record gap. It checks for record written by counting characters up through 191 in the subroutine after each PRINT#. Statement 1010 accumulates length -- remember to add 1 for the carriage return character. A POKE to location 59411 turns on the motor to advance tape for 6/60 second ([TI-T]=6) and then another POKE turns the motor off.

For a nice example of GET# and ST usage, I have included a "Show Tape" program written by Commodore. The program is also very helpful if you want to look at your data tapes. Line 215 checks for Carriage Return character ASC(B$)=13 and prints something to depict the character. I used a single character for return, since it makes it easier to count 80 character segments (40 characters/line).

```
40   PRINT "cs       --SHOW TAPE PGM BY COMMODORE--"
50   PRINT
60   PRINT "PUT YOUR DATA TAPE IN
70   PRINT "CASSETTE #1 AND REWIND IT.
80   GOSUB 1000
90   PRINT "THE TAPE WILL BE READ AND
100  PRINT "SHOWN TO YOU IN 80 CHARACTER
110  PRINT "HUNKS.  WHEN YOU WANT TO STOP
120  PRINT "PRESS ANY KEY.  THE PROGRAM
130  PRINT "WILL ASK IF YOU WANT MORE
140  PRINT "DATA TO BE SHOWN.
150  GOSUB 1000
160  OPEN 1
170  PRINT "cs " :H=0
180  H=H+1:PRINT "HUNK # "H
190  FOR J=1 TO 80
200  GET #1,B$
210  IF ST > 0 THEN 400
215  IF ASC(B$) = 13 THEN PRINT "    ";:GOTO 230
220  PRINT B$;
230  NEXT J
240  PRINT
250  GET A$
260  IF A$ = "" THEN 280
270  PRINT "MORE?";
280  GET A$
290  IF A$ = "" THEN  280
300  IF A$ = "Y" THEN PRINT: GOTO 180
310  END
400  PRINT: PRINT "STATUS WORD IS: "ST
410  IF (ST) AND 4 THEN PRINT "SHORT BLOCK
420  IF (ST) AND 8 THEN PRINT "LONG BLOCK
430  IF (ST) AND 16 THEN PRINT "READ ERROR
440  IF (ST) AND 32 THEN PRINT "CHECKSUM ERROR
450  IF (ST) AND 64 THEN PRINT "END OF FILE
460  IF (ST) AND 128 THEN PRINT "END OF TAPE
470  END
1000 PRINT: PRINT "PRESS ANY KEY
1010 GET A$: IF A$ = "" THEN 1010
1020 PRINT: RETURN
```

PET's FOR-NEXT Loops, Arrays, and a Simple Sort Program
by Jim Butterfield, Toronto

Are all Basic's the same? Of course not .. but sometimes the differences can be sneaky. Take the simple FOR - NEXT loop which almost all Basics have. They look the same, but there can be subtle differences which can make programs hard to transfer from one Basic to another.

Try to figure out what this simple Basic program will print: 100 M=0: 110 FOR J=7 TO 5 : 120 M=J+M : 130 NEXT J : 140 PRINT M;J
Then try it on your PET. Did you guess that you'd get 7 and 8? Don't feel bad; I guessed wrong, too, and you'll get wildly different answers on different Basics.

PET Basic insists that a FOR - NEXT loop will be exercised at least once, no matter how funny the range looks. Try experimenting with various STEP values (including negative ones) and see if you can establish what you get after you've exited the loop.

This characteristic (oddity?) of PET Basic can be put to use in a handy little insertion sort program. One other oddity is needed: the fact that PET Basic allows a subscript of zero on arrays. I won't explain this one; see if you can see where the zero subscript on the array is used.

Program instructions: this program inputs a list of names; after the last one it outputs them in alphabetical order. When prompted NAME? input a name .. don't forget to put the name in quotes if it contains a space character. After you've entered all the names, reply to NAME? with END (please don't spell it wrong) and the names will be output in alphabetical order.

```
100 DIM T$(30)                    up to 30 names
110 N=0                           no names to start
120 PRINT "NAME";                 prompt next name
130 INPUT A$                      input the name
140 IF A$="END" GOTO 220          part 2 if END
150 FOR J=N TO 1 STEP -1          find the spot
160 IF A$>=T$(J) GOTO 190         found it?
170 T$(J+1)=T$(J)                 no, make room
180 NEXT J                        & keep looking
190 T$(J+1)=A$                    found; insert new name
200 N=N+1                         bump name-counter
210 GOTO 120                      go for next name
220 PRINT                         separate output
230 FOR J=1 TO N                  for all names
240 PRINT T$(J)                   print 'em
250 NEXT J
```

ed. note: Jim's discussion of FOR-NEXT brings up another point. For PET, make sure you complete a FOR-NEXT loop so it gets cleaned off the stack. Several people wrote that OTHELLO in 10/77 BYTE did not run. This was due to branching out of FOR-NEXT loops before proper completion. To make the program run, I changed the FOR-NEXT loop at 1820-1840 to a counter loop, and forced the FOR-NEXT loops in the subroutine 2620-2720 to go to proper completion.

```
2620  F1=-1: FOR I1=-1 TO 1  2640  IF A(.....)<> T2 THEN 2650
2645  J1=1: I1=1: F1=0  2680 and 2690 deleted  2710  F1=F1+1
```


Using The PET's 8 Bit Parallel I/O Port
by Dan Fylstra, 22 Weitz St., Boston, MA 02134

The PET employs the IEEE 488 bus for general purpose interfacing of external devices, however, for "quick and dirty" interfacing problems, it may be simpler and cheaper to use the 8 bit parallel I/O port. This port is capable of handling many common peripherals including an ASCII keyboard, a printer or a paper tape reader, but only one device can be connected to the port at a time without some external switching logic.

The 8 bit port is actually part of an MOS Technology MCS 6522 Versatile Interface Adapter (VIA). You can get a copy of the VIA data sheet from MOS Technology, 950 Rittenhouse Rd., Norristown, PA 19401, (215) 666-7950. Most of the VIA's features apparently are used for the PET itself, leaving only an 8 bit port and two handshake lines, which are really quite simple to use. This discussion will limit itself to input through the 8 bit port, which I have actually tested with a REACO optical paper tape reader, but the essential information for output through the port will be included.

The new PET user manual briefly describes the 8 bit port edge connector: pins A and N are grounded, pin B is CA1, the input handshake line, pin M is CB2, the output handsake line, and pins C through L are the 8 data lines, with C being the high order (leftmost) and L the low order bit. When the PET is turned on, the 8 data bits are programmed to act as inputs and CA1 is programmed to recognize a negative transition (from 1 to 0). If the handsake or data strobe line on your peripheral device produces a positive transition, you can reprogram CA1 with the BASIC statement:

```
POKE 59468, PEEK (59468) OR 1
```

which changes the CA1 control bit in the VIA's Peripheral Control Register (address 59468) from 0 to 1.

When a transition occurs on CA1, meaning that data is ready to be read from the data lines, the next to low order bit in the VIA's Interrupt Flag Register (the CA1 flag bit) will be set. You can test for this with the BASIC statement:

```
WAIT 59469,2
```

which takes the contents of the Interrupt Flag Register, AND's it with 2 or binary 00000010, and tests the result, repeating the test until the result is nonzero. (Note that execution of the WAIT statement cannot be interrupted with the RUN/STOP key, so you should have a way of manually creating a transition on CA1 when you're testing the interface.)

After execution of the WAIT statement, the data present at the 8 bit port is ready to be read with the BASIC statement:

```
D=PEEK (59457)
```

which reads the VIA's Port A and stores the data in the BASIC variable D as an unsigned integer between 0 and 255. A side effect of the PEEK is to reset the CA1 flag bit in the Interrupt Flag Register, thereby setting things up for execution of the next WAIT statement.

Thus, to read a whole line of ASCII characters ending with a carriage return (binary 00001101 or 13) into a string variable, you could use the following program segment:

```
10 A$=""
20 FOR I=1 TO 72
30 WAIT 59469,2
40 D=PEEK(59457) AND 127
50 IF D=13 THEN 80
60 A$=A$+CHR$(D)
70 NEXT I
80 ? A$
```

Here statement 20 simply limits the number of characters read to 72; statement 40 masks the data read to 7 bits to eliminate any parity bit; and statement 60 uses string concatenation to convert the data into a single string. Although the PET's internal character code is essentially ASCII, some character code translation will be needed in most practical applications. This can easily be done with an array in BASIC.

To use the 8 bit port for output, you must first program the data lines to act as outputs with the BASIC statement:

```
POKE 59459,255
```

which sets all bits of the VIA's Data Direction Register A to 1s. Handshaking is considerably less convenient, since only the CB2 line is brought to the edge connector. You can force CB2 to a logic 1 with the BASIC statement:

```
POKE 59468,PEEK (59468) OR 224
```

and reset it to 0 with:

```
POKE 59468,PEEK (59468) AND 31 OR 192
```



```
100 REM  C.E.Saul, 620 South Brandon, Kokomo, IN 46901
110 REM RANDOM SYMETRIC PATTERN GENERATOR
120 REM INPUT IS FOUR NUMBERS REPRESENTING
130 REM THE BYTE VALUES FOR THE CHARACTERS
140 REM TO BE DISPLAYED IN THE FOUR QUARTERS
150 REM OF THE SCREEN:    Q1  Q2
160 REM                   Q3  Q4
170 REM     NUMBERS TO TRY  -----
180 REM  160,160,160,160
190 REM  86,86,86,86
200 REM  214,214,214,214
210 REM  77,78,78,77
220 REM  78,77,77,78    -----  85,73,74,75
230 REM  79,80,76,122   -----  112,110,109,125
240 REM  122,76,80,79   -----  127,255,255,127
250 REM
260 INPUT  G1,G2,G3,G4
270 PRINT CHR$(147)
280 FOR K=1 TO 100
290 I=INT(RND(3)*11+1)
300 J=INT(RND(5)*19+1)
310 M=I+1
320 L1=32768+40*I+J
330 L2=32768+40*(24-M)+J
340 L3=32768+40*I+(40-J)-1
350 L4=32768+40*24-M)+(40-J)-1
360 POKE L1,G1
370 POKE L2,G3
380 POKE L3,G2
390 POKE L4,G4
400 NEXT K
410 GOTO 260
420 END
430 REM  A RANDOM PATTERN IN PRODUCED IN ONE QUARTER
440 REM  OF THE SCREEN  -  THEN MIRROR IMAGES ARE
450 REM  PRODUCED IN THE OTHER THREE QUARTERS.
```

```
10    REM LOOK AT CRT CHAR  by PET User Group
20    PRINT "cs cd cd cd cd cd cd cd cd cd cd"  clear screen and print 10
                                                  cursor downs
100   POKE 33178,152
110   POKE 33180,152
130   POKE 33258,152
140   POKE 33260,152
150   FOR I=0 TO 255
170   POKE 33219,I
180   PRINT TAB(13);I;"cu"                      print I and a cursor up
190   FOR J=1 TO 800: NEXT J                    wait
200   NEXT I
210   END
```

```
1     REM  C.E. Saul, 620 South Brandon, Kokomo, IN 46901
3     REM -- SIMDIS -- A SIMPLE DISSEMBLER ---------
5     REM -- PROGRAM RECOGNIZES OPCODE --------
7     REM -- AND CONVERTS MEMORY TO HEX -------
10    DIM NO(16,15),H$(15)
20    FOR I=1 TO 16
30    FOR J=1 TO 15: READ NO(I,J): NEXT J
40    NEXT I
50    FOR I=1 TO 15: READ H$(I): NEXT I
100   PRINT "START ADDRESS": INPUT A
104   PRINT CHR$(147)
106   FOR I=1 TO 25
108   Z=PEEK(A)
110   GOSUB 300
112   IF MM>=1 GOTO 140
114   PRINT "  "
120   GOSUB 400
130   GOTO 150
140   MM=MM-1
150   PRINT A$;B$;"  ";
160   A=A+1
170   NEXT I
180   MM=0
190   GOTO 100
300   L=INT(Z/16)
310   A$=H$(L)
320   IF L<1 THEN A$="0"
330   K=Z-16*L
340   B$=H$(K)
350   IF K<1 THEN B$="0"
360   RETURN
400   L=L+1: K=K+1
410   IF K>15 THEN PRINT "*";
420   IF K>15 GOTO 470
430   MM=NO(L,K)
440   MM=MM-1
450   PRINT A;
460   IF MM<0 THEN PRINT "*";
470   RETURN
600   DATA 1,2,0,0,0,2,2,0,1,2,1,0,0,3,3
601   DATA 2,2,0,0,0,2,2,0,1,3,0,0,0,3,3
602   DATA 3,2,0,0,2,2,2,0,1,2,0,0,0,3,3
603   DATA 2,2,0,0,0,2,2,0,1,3,0,0,0,3,3
604   DATA 1,2,0,0,0,2,2,0,1,2,0,0,3,3,3
605   DATA 2,2,0,0,0,2,2,0,1,3,0,0,0,3,3
606   DATA 1,2,0,0,0,2,0,0,1,2,0,0,3,3,0
607   DATA 2,2,0,0,0,2,0,0,1,3,0,0,0,3,0
608   DATA 0,2,0,0,2,2,2,0,1,0,1,0,3,3,3
609   DATA 2,2,0,0,2,2,2,0,1,3,1,0,0,3,0
610   DATA 2,2,2,0,2,2,2,0,1,2,1,0,3,3,3
611   DATA 2,2,0,0,2,2,2,0,1,3,1,0,3,3,3
612   DATA 2,2,0,0,2,2,2,0,1,2,1,0,3,3,3
613   DATA 2,2,0,0,0,2,2,0,1,3,0,0,0,3,3
614   DATA 2,2,0,0,2,2,2,0,1,2,1,0,3,3,3
615   DATA 2,2,0,0,0,2,2,0,1,3,0,0,0,3,3
616   DATA 1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
```

```
RUN
START ADDRESS
?57600

57600  A2  A9
57602  2C  85  09
57605  20  D2  C9
57608  A2  68
```

Table does not include added command which is in later 6502's

AV = 100/M(0)

P_1(X) = INT((M(X) * AV)+.5 ? P(1)

FNY(X)= INT((M(X) *100/M(0))+.5)

FNZ(X) = INT((100* M(X)/ M(X-6))+.5)