



PET USES ITS MEMORY

Not long ago, People's Computer Company published a table of PET's memory usage, and we're including it here for those of you who missed it there. (PCC Nov/Dec 1977)

<u>BASIC</u>	<u>BYTES</u>	<u>USED FOR</u>
BASIC	1028	I/O buffers Tables Scratch Pad
Line Numbers	4	
BASIC keywords	1	
Characters	1	Includes RETURN
Variables	7	If a value is assigned (for integers and floating point variables)
	7 + length of string	for string variables
Arrays	$* (S+1)+(2*D)$	
(Size includes the	Where A=5	Floating point array
0th element)	A=3	String array
	A=2	Integer array
	and S=	Size of array
	D=	Number of dimensions



commodore



COMMODORE BUSINESS MACHINES, INC.
901 CALIFORNIA AVENUE
PALO ALTO, CALIFORNIA 94304
TELEPHONE: (415) 326-4000 TELEX: 345-589
CABLE ADDRESS COMBUSMAC PLA

PET AND ASCII

When the BASIC PET Manual is published, it will describe and explain in some detail the uses of the ASC, CHR\$, VAL, and STR\$ functions. In the meantime, perhaps these explanations will tide you over.

ASC("character") will return the ASCII numeric representation of the specified character enclosed in the quotation marks.

ASC(A\$) will return the ASCII numeric representation of the first character of the specified string.

CHR\$(number) will return the character represented by the ASCII number specified. Numeric variable names and numeric expressions may also be used.

VAL("numeric string") will convert the string to a floating point format. If the characters in the string are not numeric, VAL will return 0. If only the first character in the string is numeric, VAL will convert only that character to a floating point format.

STR\$(number)

will convert a number to string format.

Numeric variables and numeric expressions may
also be used.















That's all well and good, you say? What are the Code numbers
for which characters, you ask? Ah...glad you asked!

Alphabet (A-Z upper case) 65 through 90
Numbers (0 through 9) 48 through 57
and 112 through 121

PET SYMBOL CODES

CHARACTER	CODE	CHARACTER	ASCII	CHARACTER	CODE
blank	32 and 96)	41 and 105		60
!	33 and 97	*	42 and 106	=	61
"	34 and 98	+	43 and 107		62
#	35 and 99	,	44 and 108	?	63
\$	36 and 100	-	45 and 109	@	64
%	37 and 101	.	46 and 110		91
&	38 and 102	/	47 and 111		92
'	39 and 103	:	58		93
(40 and 104	;	59		94
					95

SPECIAL KEYS/FUNCTIONS

CODE	SYMBOL INSIDE QUOTES	FUNCTION
<i>D</i> 13	-	CARRIAGE RETURN
<i>11</i> 17		 CURSOR DOWN
<i>12</i> 18		REVERSE FIELD ON
<i>13</i> 19		HOME CURSOR
<i>14</i> 20		DELETE CHAR
<i>1D</i> 29		 CURSOR RIGHT
<i>8D_H</i> 141	-	SHIFT CARRIAGE RETURN
<i>91₄</i> 145		 CURSOR UP
<i>92_H</i> 146		REVERSE FIELD OFF
<i>93_H</i> 147		CLEAR/HOME
<i>94_H</i> 148		INSERT CHAR
157		 CURSOR LEFT

You can get a good look at PET's graphics and their Code representations by typing in the following program:

```

10 FOR I = 160 TO 255 STEP 4:C=1
20 FOR J = I TO I + 3
30 IF J = 256 THEN 60
40 PRINT TAB(C);J;CHR$(J);
50 C = C+10:NEXT J:PRINT:NEXT I
60 POKE 59468,12

```

```
70 FOR I = 1 TO 1500:NEXT
80 POKE 59468,14
90 FOR I = 1 TO 1500:NEXT
100 GOTO 60
```

Just for general interest, you might like to know that Code 160 represents an upper-case (shifted) blank, or space. Remember that the Code numbers given here are in decimal, not in octal, and have a good time watching that program do its thing!

Note that the letter, symbols, and numbers have codes identical to the ASCII convention.

CA1 DATA ACCEPTED
CB2 STROBE



BASIC BUGS

We'll publish all the bugs we know about, and a few months from now, when we've found and fixed all of them, we'll produce a new ROM which you'll be able to buy and plug in.

1. 10 IF F OR I = 10 THEN 10 gets collapsed to
10 IF FOR I = 10 THEN 100 and yields a ?SYNTAX ERROR

We've found and fixed this one. The only reserved word that can have embedded spaces is GOTO, which may appear as GO TO. Therefore, "FOR" in this example will no longer be converted to a reserved word.

2. The BYTES FREE message number and the amount of bytes free when PRINT FRE(0) is typed just after start-up are different.

This is not a bug. PRINT FRE(0) uses 3 bytes of RAM.

3. The SAVE command should respond with "PRESS REC AND PLAY" instead of "PRESS PLAY AND REC", since the latter sequence doesn't work.

This is liveable and probably won't be fixed.

4. The POS function is not effected by Pokes and other cursor movements. It does not keep track of where the cursor is moved with POKES and other cursor movements.

POS will be deleted from BASIC.

5. SPACE and shifted SPACE characters have different ASCII values.

This is not a bug. Shifted and unshifted characters are indexed separately.

6. When a quotation mark (Code 34 or 98) is output, the rest of the line treats cursor movement literally. Example:

```
10 PRINT CHR$(34),34      (Try it and see)
```

This will not be changed at present.

7. SPC(0) returns 256 spaces.

Fixed.

8. Direct lines beginning with colons, ":", are ignored.

Fixed.

9. Arrays with more than 255 elements fail.

Fixed.

10. Random Number Function -- How does it work?

NAME	EXAMPLE
RND (X)	170 PRINT RND (X)

Generates a random number between 0 and 1. The argument X controls the generation of random numbers as follows:

$X < 0$ generates a new sequence of random numbers using X as a seed. Calling RND with the same X where $X < 0$ will generate the same random for each X if X does not change.

Example: RND (-1) gives
2.99196472E-08 for as many times as you use -1.
2.99205567E-08 for as many times as you use -2.

This is useful for debugging where you want the same random number to be generated. You can get a different but constant random number with any minus number.

$X = 0$ generates ⁵⁹⁶⁰⁷⁸⁸³¹.564705882 each time you call

$X > 0$ will generate the next randomly sequenced random number if X does not change. If X changes, the new X is used as a seed to a new sequence of random numbers.

If you want to verify what the RND actually does, enter the program:

```
10 INPUT R
20 X=RND (R)
30 PRINT X
40 GOTO 10
```

Then try various values for the input.

11. CHR\$ accepts string arguments.

Fixed.

12. DEF FN fails in one out of 256 cases.

Fixed.

The following is an example of a PET QUIRK. It is not a bug, and happens because the character printed after a number is a "cursor right" rather than a carriage return or a space.

Output of a number is:

S	I	G	N		N	U	M	B	E	R		C	U	R	S	O	R		R	I	G	H	T
---	---	---	---	--	---	---	---	---	---	---	--	---	---	---	---	---	---	--	---	---	---	---	---

Which can cause havoc with screen overwrites if you aren't aware of it.

```
10 PRINT "[S]";  
20 FOR I = 1 TO 10  
30 PRINT "BBBBBBBB"  
40 NEXT  
50 PRINT " [S]"  
60 FOR I = 1 TO 10  
70 PRINT I*100"HI!"  
80 NEXT
```

The black "S" on a white field is the character used to represent "home cursor".

And lo! on your screen will appear:

100BHI! BBB
200BHI! BBB
.

commodore



COMMODORE BUSINESS MACHINES, INC.
901 CALIFORNIA AVENUE
PALO ALTO, CALIFORNIA 94304
TELEPHONE: (415) 326-4000 TELEX: 345-569
CABLE ADDRESS COMBUSMAC PLA

A LIST OF IEEE-488 DEVICES TO USE WITH PET

You can get the IEEE-488 specs, by sending \$10.00 and \$2.00 for postage and handling to

IEEE SERVICE CENTER

445 HOES LANE

PISCATAWAY, NJ 08854

While we list an RS232/IEEE-488 interface, it really doesn't exist yet. Microcomputers Associates will have the unit on the market sometime during the first quarter of next year for about \$250.

IEC/IEEE PRODUCT INTRODUCTIONS

66(C.O.) 22, IEEE 488, ANSI MC1.1 COMPATIBLE

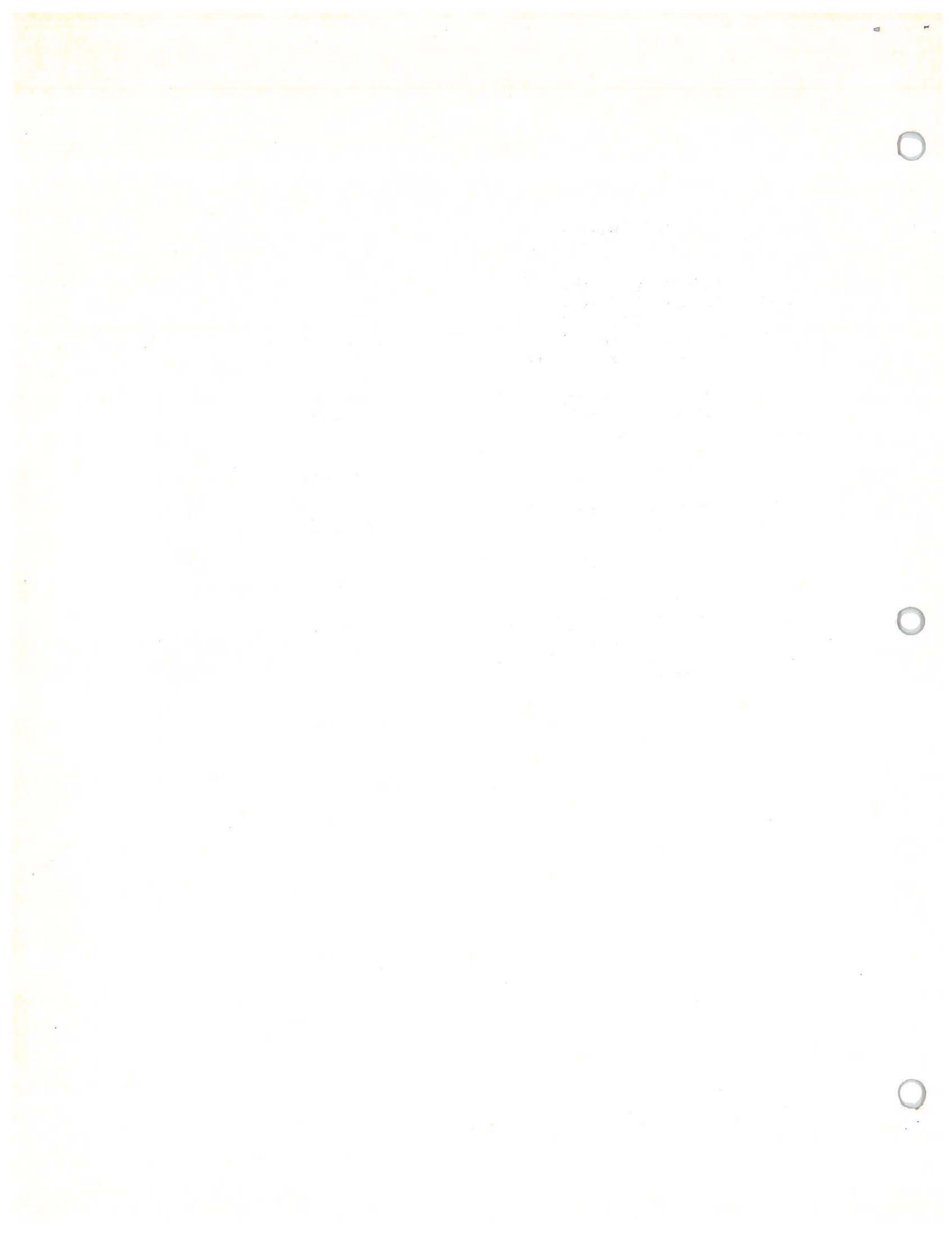
5500B	UNIVERSAL COUNTER TIMER	BALLANTINE
76A	AUTOMATIC CAPACITANCE BRIDGE	BOONTON ELECTRONICS CORP.
3347	AUDIO FREQUENCY ANALYZER	BRUEL & KJAER
4426	NOISE LEVEL ANALYZER	"
1554	STRAIN INDICATOR	"
	BUS CABLE ASSEMBLY	BUNKER-RAMO CORP.
DSM44	DIGITAL MULTIMETER	CALIFORNIA INSTRUMENT CO.
	BUS CABLE ASSEMBLY	COMPONENT MFG. SERVICES
340	MATERIALS TESTING FUNCTION GENERATOR	
605-145	WAVEFORM GENERATOR, ASCII PROGRAMMER	DANA EXACT ELECTRONICS INC.

801	FREQUENCY SYNTHESIZER	DANA EXACT ELECTRONICS INC.
802	FREQUENCY SYNTHESIZER	
55	MICROPROCESSING GPIB (5000, 5900, 6900 DVMS)	
9015	MICROPROCESSING TIME/COUNTER	DANA LABS INC.
9035	MICROPROCESSING TIMER/COUNTER	
7500	DIGITAL MULTIMETER	DATA PRECISION CORP.
101	UNIVERSAL TIMER/COUNTER	
103	"	
105	"	
111	DIGITAL FREQUENCY COUNTERS	DATA TECHNOLOGY (RACAL)
113	"	
115	"	
117	"	
4880	BUS INTERFACE COUPLER	DATA WORKS INSTRUMENTATION
3000	HF COMMUNICATIONS RECEIVER	DECCA COMMUNICATIONS LTD.
IEC11-A	CONTROLLER (PDP-11)	DIGITAL EQUIPMENT CORP.
1015A	9 TRACK TAPE	DYLON CORPORATION
1015PE	9 TRACK TAPE	
1015B	7 TRACK TAPE	
331	MICROWAVE COUNTER	EIP EXACT
351D	COUNTER	
451	MICROWAVE PULSE COUNTER	
296	AUTOMATIC LRC DIGITAL METER	ELECTRO SCIENTIFIC INDUST
501J	PROGRAMMABLE VOLTAGE STANDARD	ELECTRONIC DEVELOPMENT COR
	PROGRAMMABLE OSCILLATOR (A.C. POWER)	ELGAR CORPORATION
9880	INTERFACE COUPLER	FAIRCHILD INSTRUMENTATION SYSTEMS
FF303	ATE SYSTEM	FAULTFINDERS INC.
1953A	UNIVERSAL COUNTER-TIMER	
6010A	SYNTHESIZED SIGNAL GENERATOR	
6011A	SIGNAL GENERATOR	FLUKE MFG. CO.
8500	SYSTEMS MULTIMETER	

1792	LOGIC TEST SYSTEM (I/O PORT)	GENRAD
436A	POWER METER	HEWLETT-PACKARD PRODUCTS
3320B	FREQUENCY SYNTHESIZER (11235A)	"
3330B	AUTOMATIC SYNTHESIZER/SWEEPER (11235A)	"
3455	VOLTMETER	"
3490A	DIGITAL MULTIMETER	"
3495A	SCANNER	"
3571A	TRACKING SPECTRUM ANALYZER	"
3745A	SELECTIVE LEVEL MEASURING SET	"
3964A	INSTRUMENTATION TAPE RECORDER	"
3968A	INSTRUMENTATION TAPE RECORDER	"
4261A	LCR METER (OPT. 101)	"
5150A	THERMAL PRINTER	"
5312A	INTERFACE MODULE (5300B MEASURING SYSTEM)	"
5328A	UNIVERSAL COUNTER	"
5340A	FREQUENCY COUNTER	"
5341A	FREQUENCY COUNTER	"
5345A	ELECTRONIC COUNTER	"
5353A	FREQUENCY COUNTERS, CHANNEL PLUG-IN	"
5363A	TIME INTERVAL PROBES	"
5354A	CONVERTER PLUG-IN	"
5942A	TRANSMISSION IMPAIRMENT MEASURING SET	"
8016A	WORD GENERATOR	"
8503A	AUTOMATIC RF NETWORK ANALYZER	"
8505A	AUTOMATIC RF NETWORK ANALYZER	"
8620C	MICROWAVE SWEEP OSCILLATOR	"
8660A/C	SYNTHESIZED SIGNAL GENERATOR	"
8672A	MICROWAVE SYNTHESIZER	"
9871A	IMPACT PRINTER	"
10745A	LASER TRANSDUCER SYSTEM COUPLER	"
47310A	A/D CONVERTER	"
59301A	ASCII PARALLEL CONVERTER	"
59303A	DIGITAL-TO-ANALOG CONVERTER	"
59304A	NUMERIC DISPLAY	"
59306A	RELAY ACTUATOR	"
59307A	DUAL VHF SWITCH	"
59308A	TIMING GENERATOR	"
59309A	DIGITAL CLOCK	"
59310A/B	21MX COMPUTER INTERFACE	"
59401A	BUS SYSTEM ANALYZER	"
59403A	HP-IB COMMON CARRIER INTERFACE	"
59405A	9820, 9830, CALCULATOR INTERFACE	"
59500A	MULTIPROGRAMMER (6940B) INTERFACE	"
98034A	HP-IB I/O (9825A)	"
98135A	HP-IB I/O (9815A)	"

3050B	DATA ACQUISITION SYSTEM	HEWLETT-PACKARD PRODUCTS
3042A	NETWORK ANALYZER SYSTEM	"
3044A	SPECTRUM ANALYZER SYSTEM	"
3045A	SPECTRUM ANALYZER SYSTEM	"
8507A	NETWORK ANALYZER SYSTEM	"
DTS70	DIGITAL TEST SYSTEM (PORT)	"
8580B	AUTOMATED SPECTRUM ANALYZER (PORT)	"
9500D	AUTOMATIC TEST SYSTEM (PORT)	"
RS432	MICROPROCESSOR DATA & TIMING GENERATOR	"
RS648	TIMING SIMULATOR/WORD GENERATOR	
IM5200	FPLA LOGIC ARRAY	INTERSIL
SPG-800	SIGNAL GENERATOR	INTERSTATE ELECTRONICS
395	LOCK-IN ANALYZER	ITHACO
7802-	SYSTEM 1 (I/O PORT)	
ISB	(5900.6900. DMM VIA MICROPROCESSING GPIB)	KEITHLY INSTRUMENTS INC.
SN-488	POWER SUPPLY	KEPCO
	RELAY DRIVER	MICROCOMPUTER ASSOC
	RS-232/IEEE-488 INTERFACE	"
1180	DATA ACQUISITION & PROCESSING SYSTEM	NICOLET INSTRUMENT CORP.
PM2441	DIGITAL VOLTMETER	
PM2460	SCANNER	
PM2467	DIGITAL VOLTMETER	N.V. PHILLIPS
PM2527	PRINTER	
PM6625	COUNTER	
PM6650	COUNTER	
	S 100 to IEEE	PICKLES & TROUT
4001	PROGRAMMABLE LOW-PASS FILTER	PRECISION FILTERS INC.
488	FLEXIBLE CARTRIDGE DISC SYSTEM	PROCESS DYNAMICS INC.
FFT/S15	REAL TIME SPECTRUM ANALYZER	ROCKLAND SYSTEMS CORP.
PCL/PCW	CARD READER/CODE CONVERTER (SMU, SMDV, DPVP)	RHODE & SCHWARZ
SMPU	RADIO SET TEST ASSEMBLY	

2017	UNIVERSAL COUNTER	
2711	UNIVERSAL COUNTER	SCHLUMBERGER
6054B/C	MICROWAVE COUNTERS	
6063	AUTOMATIC COUNTER	
7115	DIGITAL MULTIMETER	
DPSD-50	DIGITAL POWER SOURCE	SYSTRON-DONNER
1600	MICROWAVE SYNTHESIZER	
4051	GRAPHIC COMPUTING SYSTEM	
4662	DIGITAL PLOTTER	TEKTRONIX
4924	MAGNETIC TAPE UNIT	
1625	LOGIC ANALYZER	VECTOR ASSOC. INC.
2254	COMPUTER BASED CONTROLLER (2200)	WANG
152	FUNCTION GENERATOR	
158	WAVEFORM GENERATOR	WAVETEK
159	WAVEFORM GENERATOR	
172	PROGRAMMABLE SIGNAL SOURCE	
4311B	FREQUENCY/PHASE-LOCK MEAS. SYSTEM	WEINSCHEL ENGR.



commodore



COMMODORE BUSINESS MACHINES, INC.
901 CALIFORNIA AVENUE
PALO ALTO, CALIFORNIA 94304
TELEPHONE: (415) 326-4000 TELEX: 345-569
CABLE ADDRESS COMBUSMAC PLA

SOFTWARE BULLETIN NUMBER ONE

PET CASSETTE FILES

First of all find some suitable blank tapes.* At least three tapes are needed, and eight of them will let you get through the bulletin with a minimum of re-running or re-entering your programs.

Secondly, follow the directions EXACTLY. Do not take any 'short cuts', as these will lead you to some of the errors shown in Part III.

Third, Part IV describes the cassette related BASIC statements and variables in detail.

The BASIC statements for cassette files are:

OPEN	Open a file
CLOSE	Close a file
PRINT#	Write to a file
INPUT#	Read to a file
GET#	Read a single character from a file

The BASIC variable used for file status is

ST	Status word
----	-------------

* Don't use the "three-for-a-dollar" type tapes! We use a good, low noise, high energy tape costing around \$2.00 - \$3.00.
(Such as Maxell or TDK)

CASSETTE (continued)

I. Some examples that work:

Example 1: Writing and reading numbers

Try out this little program, being sure to type it in exactly as it appears here.

```
10 OPEN 1,1,1
20 FOR J = 1 TO 20
30 PRINT#1,J           Spell out the word PRINT ...
40 NEXT J              Do not use ?#
50 CLOSE 1
60 PRINT "REWIND YOUR TAPE AND THEN PRESS A KEY"
70 GET A$: IF A$ = "" THEN 70
80 OPEN 1
90 FOR J = 1 TO 20
100 INPUT#1,X
110 PRINT X
120 NEXT J
130 CLOSE 1
140 PRINT "PHEW!!"
```

Now list the program and compare each line with the listing above. (Please!! Don't skip this step!)

And finally, save the program on a cassette using the SAVE command:

```
SAVE "PGM 1"
```

CASSETTE (continued)

Mark the cassette with the label "PGM 1". If you don't save and mark your program, you'll have to type it in again later. This Bulletin assumes from now on that you know how to save and load programs by name.

Remove the program cassette and put a fresh cassette in the recorder unit. Be sure the tape has been rewound, and then run the program. The screen will show

```
RUN
```

```
PRESS PLAY & RECORD ON TAPE #1
```

When you have pressed the right buttons on the cassette unit, the screen will display "OK"

```
RUN
```

```
PRESS PLAY & RECORD ON TAPE #1  
OK
```

The program will write data onto the tape, and when it is finished, you should see on your screen:

```
RUN
```

```
PRESS PLAY & RECORD ON TAPE #1  
OK  
REWIND YOUR TAPE AND THEN PRESS A KEY
```


CASSETTE (continued)

So . . . rewind your tape and then press a key. Be sure the tape is fully rewound. Then, as the screen instructs, press PLAY on the cassette unit. The program now reads the numbers from the cassette and puts them on the screen:

```
OK
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
PHEW!!
READY
█ ← Cursor
```

Rewind the tape, label it "DATA 1", and put it away for the time being.

Now LIST the program. If you are experienced in BASIC, have patience, for here comes a line-by-line explanation of what the program does.

CASSETTE (continued)

```
30 PRINT#1, "I AM A PET!"
```

Again, don't use ?#

```
100 INPUT#1,X$
```

```
110 PRINT X$
```

Now RUN your new program and watch while the screen shows:

OK

```
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
I AM A PET!  
PHEW!!
```

READY



LIST the program and check that lines 30, 100, and 110 are changed. Save this program as "PGM 2" on a fresh cassette. Don't destroy PGM 1 on the first cassette!

CASSETTE (continued)

If you find this too dull, feel free to look at the descriptions of the statements at the back of this bulletin while the rest of us look at the explanation.

```
Line 10      Opens logical file 1 on tape drive 1, for write only
              (I/O option 1)

Lines 20      Writes the integers 1 through 20 on the tape
to           |
40           |

Line 50      Closes the file

Lines 60      Tells you to rewind the tape and waits for you to
to           |
70           | press a key

Line 80      Opens logical file 1, with defaults of tape drive 1
              for read only (equivalent to 80 OPEN 1,1,0)

Lines 90      Reads the numbers from tape and shows them on the
to           |
120          | display screen.

Line 130     Closes the file

Line 140     Lets you know the program completed correctly.
```

Example 2: Writing and reading strings

Change the following lines in PGM 1. Hopefully, you still have the program in your PET. If you don't, load it from tape (you DID save it, didn't you?) or type it again. In any event, when you're ready, type in these new lines, and PET will replace the old lines with the new ones.

CASSETTE (continued)

Use a fresh cassette for the data, and label it "DATA 2". a

From now on, we'll show the results, and not show the step-by-step displays, or mount/discount cassettes, etc. You should have four cassettes by now: "PGM 1", "DATA 1", "PGM 2", and "DATA 2".

Example 3: Mixing strings and numbers

Make these changes to "PGM 2" and save the new program as "PGM 3".

```
35 PRINT#1,J
105 INPUT#1,X
115 PRINT X
```

Note that lines 30, 100, and 110 are still present.

```
10
I AM A PET!
11
I AM A PET!
12
I AM A PET!
13
I AM A PET!
14
I AM A PET!
15
I AM A PET!
16
I AM A PET!
17
I AM A PET!
18
I AM A PET!
19
I AM A PET!
20
PHEW!!
```

READY

CASSETTE (continued)

Run the program and save the data on a cassette marked "DATA 3".
Now you have three programs and three data tapes which work! In the following sections, we will use these programs to show you the use of GET# and ST. Later, some common errors will be examined.

II. Looking at data on tapes.

First of all, read the descriptions of GET# and ST at the end of this bulletin. Then clear the program out of your PET by typing "NEW" and enter the following program EXACTLY:

```
10 REM SHOW CONTENTS OF CASSETTE TAPES
20 REM TO SOLVE TAPE MYSTERIES
30 REM BY COMMODORE
40 PRINT "♥ - - SHOW TAPE PGM - -
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
```

The reverse
field heart indicates
a "clear Screen"
character.


CASSETTE (continued)

```
170 PRINT "♥ " :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$) = 13 THEN PRINT " <RETURN> ";:GOTO230
220 PRINT B$;
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:GOTO180
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
```

CASSETTE (continued)


Note that the  in lines 40 and 170 is the "clear screen and home the cursor" character.

Save this program on a fresh cassette and label it "SHOW TAPE". You will find it handy for seeing what is on your tapes ... often what you intended to do is not what you did!


Now run this program, using the "DATA 1" tape. The CRT will show:

```
HUNK #1
 1 <RETURN> 2 <RETURN> 3 <RETURN> 4 <RETURN>
 5 <RETURN> 6 <RETURN> 7 <RETURN> 8 <RETURN>
 9 <RETURN> 10 <RETURN> 11 <RETURN> 12 <RETU
RN> 13 <RETURN> 14 <RETURN> 15 <RETURN> 16
RETURN> 17 <RETURN> 18 <RETURN> 19 <RETURN>
 20 <RETURN>

STATUS WORD IS: 64
END OF FILE

READY.

```

And here's the same program using the "DATA 2" tape:

```
HUNK #1
I AM A PET! <RETURN> I AM A PET! <RETURN>
I AM A PET! <RETURN> I AM A PET! <RETURN>
I AM A PET! <RETURN> I AM A PET! <RETURN>
I
HUNK #2
AM A PET! <RETURN> I AM A PET! <RETURN> I
AM A PET! <RETURN> I AM A PET! <RETURN> I
AM A PET! <RETURN> I AM A PET! <RETURN> I
AM
MORE? 
```

CASSETTE (continued)

Here, a key was pressed during HUNK #2. Some of the critical lines in "SHOW PROGRAM" are:

20 GET#1,B\$	This gets the character from the file.
210 IF ST>0 THEN 400	If any file condition is encountered, this jumps to a report of the condition. Since GET# will read past end-of-file marks, the status must be checked each time a character is read.
215	Detects RETURN and displays it in a visible form.
400 - 460	Reports status. Note the "AND" is a logical mask operation which checks for the appropriate bit in ST.

Run this program with "DATA 3" and see if the file looks like you expect it to look.

III Some Examples that Don't work

It is easy to make errors with cassette files. Some will give a ?SYNTAX ERROR and others will stop BASIC and force you to turn the PET's power off and start over. Be sure you have the tapes "SHOW TAPE", "PGM 1", and "DATA 1" available before you start this section.

ERROR #1: THE "?#" SHORTCUT

Load "PGM 1" and change line 30. Type in: 30 ?#1,J

CASSETTE (continued)

Now try to run it

?SYNTAX ERROR IN 30

So LIST 30

30 PRINT#1,J

Mysterious, isn't it? Now you know that "?#1" does not work. You must always spell out the word PRINT# when using cassette files.

Okay, here's the explanation. When you typed in line 30, BASIC converted the PRINT, or "?" into a token. However, the tokens for PRINT and PRINT# are different.

Suppose PRINT becomes ☐X

and PRINT# becomes ☐Y

Then, if you type in 30 PRINT J BASIC stores it as 30 ☐X J

And, if you type in 30 PRINT#1,J BASIC stores it as 30 ☐Y 1,J

30 ?J becomes 30 ☐X J --- but 30 ?#1,J becomes 30 ☐X #1,J

So, 30 ?#1,J LISTS as 30 PRINT#1,J and looks correct. However, when it is run, BASIC sees the # following the PRINT token.

Since # is not a number or a legal variable name, BASIC gets upset and tells you you have a syntax error. In line 30.

```
*****
*FIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIX*
*I                                                                 I*
*X      REMEMBER!!!!!!  IF YOU GET A SYNTAX ERROR IN A PRINT    X*
*F                                                                 F*
*I      TO A FILE STATEMENT, AND IT LOOKS OK WHEN YOU LIST      I*
*X                                                                 X*
*F      IT, THE FIX IS:                                          F*
*I                                     RETYPE THE LINE USING      I*
*X                                     THE FULLY SPELLED WORD      X*
*F                                                                 F*
*I                                     PRINT#1                    I*
*X                                                                 X*
*FIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIXFIX*
*****
```


CASSETTE (continued)

IV. The Output image

The PRINT# statement prints exactly as it is told to. If more than 40 characters are output to a file without a carriage return, no carriage return will be inserted. Type in the following program:

```
10 OPEN 1,1,1
20 X$="1234567890"
30 FOR J = 1 TO 5
40 PRINT X$;
50 PRINT#1,X$;
60 NEXT J
70 CLOSE 1
```

Put the "DATA 1" tape in the cassette drive, rewind it, and run the program. Then load "SHOW TAPE" and use it to look at the "DATA 1" tape. Notice that when you ran the first program, the screen looked like this:

PRESS PLAY & RECORD ON TAPE #1


OK

1234567890123456789012345678901234567890

1234567890

READY.




No carriage return
was put here!

CASSETTE (continued)

The line ran off the right edge of the screen and appeared on the next line. But no carriage return was ever printed. The "SHOW TAPE" program proves this: No <RETURN> appears in HUNK #1.

Try a few more combinations. PRINT# will always write what it is told to write on the tape. Remember that PRINT# writes on the tape just like PRINT does on the screen (if you had a mile-wide screen, that is).

V. The Input Image

The INPUT statement in PET BASIC has some oddities. To understand this, some examples without using cassettes are in order.

Example 1. Discard of Extra Input

```
10 INPUT A,B,C
```

```
20 PRINT A,B,C
```

RUN, and enter 1,2,3,4,5 when the question mark appears on the screen.

The screen will show

```
RUN
? 1,2,3,4,5
? EXTRA IGNORED
  1         2         3

READY.
■
```

CASSETTE (continued)

Example 2: 80 Character INPUT limit

Try: 10 INPUT X\$

20 PRINT X\$

Run, and enter AAAAAAAAAA.....until you have 100 "A"s entered.

The screen will show:

```
RUN
? AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA ←
READY.
PET printed this line
```

This is because the input buffer can only accept 80 characters at a time. If more are entered, the first 80 are lost.

Example 3: Too Little INPUT

Try: 10 INPUT A,B,C,D,E,F

20 PRINT A;B;C;D;E;F

Now RUN, and enter 1,2,3 RETURN
4,5,6 RETURN and you get:

```
RUN
? 1,2,3
?? 4,5,6
1 2 3 4 5 6
READY
```

CASSETTE (continued)

INPUT will look past a carriage return until all the variable list is satisfied (A,B,C,D,E,F in the example). Now try this one again, but enter 1 2 3 4 5 6 RETURN

You will get a ??. Enter RETURN until you see READY. INPUT ignores blanks when reading numbers and will keep asking for more u u until it is finished. Note that the number for A is 123456.

This digression really will help you understand the following rules for writing cassette file data.

RULES FOR WRITING ON CASSETTE FILES

1. Be sure to have matching INPUT# and PRINT# variable lists. If your INPUT list is too short, you will lose data.
2. Don't ever print more than 79 characters without a carriage return. If INPUT# reads over 80 characters, it will either lose the first 80 characters or CRASH BASIC!
3. Extra carriage returns don't hurt anything.
4. If you want to write several numbers on a line, separate them with a comma "," or else, when they are read, you will get the wrong values.

Here are some examples to show what happens with tapes. In each example, load your "PGM 1" and modify it accordingly. Don't save it, though, since each example assumes you use the original "PGM 1". Then use "SHOW TAPE" to see what is on your data tape afterwards. Use the "DATA 1" data tape for these examples.

CASSETTE (continued)

ERROR #2 No carriage returns or commas between numbers

Type 30 PRINT#1,J; and run it. The program will eventually show

```
OK
1.23456789E+31
```

```
?OUT OF MEMORY ERROR IN 8224
FORMULA TOO COMPLEX ERROR IN 8224
```

and CRASH! You have to turn the power off and start over. This is sometimes unpredictable - once the tape just kept running, and BASIC didn't crash until I tried to STOP the tape! If you use "SHOW TAPE" with "DATA 1", you will see 1 2 3 4 5 6 without carriage returns or commas.

ERROR #3 Loss of additional INPUTs

Load "PGM 1" and type 30 PRINT#1,J","; and run it

```
a "1" appears
```

and CRASH! Again, INPUT# tried to read past the end of the file, looking for a <RETURN> , with disastrous results. When I tried it, I also changed line 105: 105 IF ST>0 THEN PRINT "END OF FILE":GOTO140
I got:

```
END OF FILE
PHEW!!
```

So.... If you test the status after INPUT#, you can avoid a crash, though more than likely you will not have done what you wanted to.

ERROR #4 More than 80 characters in a line

Again, load "PGM 1" and change it as follows:

```
30 PRINT#1 "ABC123";
```

CASSETTE (continued)

```
90      |  
120      | Delete these lines  
100 INPUT#1,X$  
110 PRINT X$
```

Now run it

and CRASH! (Yet again. I hope you can see why the rules are to be followed!)

ERROR #5 Substring file name matching

If you open a file with a name, the PET will read the tape until it finds a suitable name. Enter the following program:

```
10 OPEN 1,1,1,"FILE1"  
20 PRINT#1,"THIS IS FILE1"  
30 CLOSE 1  
40 OPEN 1,1,1,"FILE"  
50 PRINT#1,"THIS IS FILE FILE"  
60 CLOSE 1  
70 PRINT "FILENAME";  
80 INPUT F$  
90 PRINT "REWIND YOUR TAPE . . . ."  
100 GET A$: IF A$ = "" THEN 100  
110 OPEN 1,1,0,F$  
120 INPUT#1, X$  
130 PRINT X$  
140 CLOSE 1
```

Notice that you have to include all the parameters here.

CASSETTE (continued)

If you get a syntax error in 10 (or 20 or 110) chances are you forgot the last comma!

Run this program, and enter "FILE" as the filename (F\$). You will see the contents of FILE1 displayed.

So . . . Be sure your file names do not match each other even in substrings, like: COM and COMMODORE, or MODE and REMODEL.

ERROR #6 Reading a program as data

Enter this program. When you run it, use your "PGM 1" tape for the data tape.

```
10 OPEN 1
20 INPUT#1,X$
30 PRINT X$
40 CLOSE 1
```

You will get a BREAK IN 10 message. If you type ?ST, you will see a 0 because the file never opened successfully.

ERROR #7 Going past End-of-file with GET#

Load "PGM 1" and run it using "DATA 1". Then do a "NEW" and enter this program:

```
10 OPEN 1
20 GET#1,X$
30 PRINT X$;
40 GOTO 20
50 CLOSE 1
```

CASSETTE (continued)

The numbers 1 to 20 will appear, then a 1, then the cassette goes on. It will go on with garbage or other stuff appearing; you have done a lot with "DATA 1"! Now, you can fix it with:

```
25 IF ST 0 > then 50
```

and run it again. Note: You may stop functioning (well, your PET may) and have to start over again. Halting during a tape read is hazardous!

NOW it works!

ERROR #8 Not fully rewinding the tape

In many instances the author failed to fully rewind a tape, and the program failed to read it. So be careful. If you fail after checking your program carefully and after three careful attempts to load, you may have a hardware problem.

CASSETTE RELATED BASIC STATEMENTS AND VARIABLES

OPEN Opens a file for input/output. The syntax is:
OPEN [logical file #] , [physical device number] , [I/O option]
 , [filename]

The keyword "OPEN" and the logical file number are required. The other items are optional. If they are not specified, a default value will be used.

Logical File # This number is used in the CLOSE, PRINT, INPUT, and GET statements to refer to this file. The logical file number can be from 1 to 255. Up to 10 files may be open at the same time.

NOTES: If you try OPEN,0 you will get a syntax error

OPEN,-1 or OPEN,256 gets "ILLEGAL QUANTITY ERROR"

If you open more than 10 files at once, the PET will "hang" and you will have to turn off the power. If you open a file with the same logical file number as one which is already open, you will get a "FILE OPEN ERROR".

Physical Device Number For cassettes, the numbers 1 and 2 are legal. Number 1 refers to the cassette in the PET and 2 is for the auxiliary, or external cassette unit. The default value is 1.

NOTES: The physical device number may be from 0 to 255. The PET currently recognizes devices 0 - 15. If the device number is out of range, you will get an "ILLEGAL QUANTITY ERROR".

CASSETTE (continued)

I/O Option This tells the cassette whether to read or write to the file. 0 is for read only, 1 is for write only, and 2 is for write only with end-of-tape marker. The default value is 0 (read).

NOTES: If option 2 is used, an EOT will be written on the tape when the file is closed. An error will result if, at a later time, you attempt to read past the EOT mark.

WARNING:

WARNING:

WARNING:

WARNING:

WARNING:

A. If you use physical device numbers other than 1 or 2, you must follow the rules for that device. For instance, the IEEE buss conventions are different from the cassette rules.

B. If you attempt to use

OPEN

OPEN 1,

OPEN,1

OPEN,,1

etc., you will get a syntax error.

FILENAME The filename is used to identify a file. It may be up to 187 characters long. However, please note that:

1. If you ask in your program for a filename, the longest string you can input is 80 characters.

CASSETTE (continued)

2. If you are searching for a file with a short name, and your tape has a file with a long name on it, the search looks for a matching sub-string. This means that if you are looking for a file named "CAT", and the tape has on it a file named "CONCATENATE", it will be recognized as "CAT".
3. Only the first 16 characters of a filename will be displayed after the "SEARCHING" message appears on the screen.

To summarize,

- * Keep your filenames short (preferably under 16 characters)
- * Avoid files named alike, such as

ACCOUNT

ACCOUNTS

ACCOUNTING

If you are looking for a file named "COUNT", any of these files will be recognized as correct files.

- * The default filename is a null string "". An "OPEN 1" will open any file, as null is always recognized as a filename, regardless of the name of the file.

CLOSE This tells the PET to:

- | | | |
|---------------------------------|-----|---------------|
| 1. Stop reading a file | or, | Open Option 0 |
| 2. Stop writing a file and | | Open Option 1 |
| make an end-of-file mark | or, | |
| 3. Stop writing a file and | | Open Option 2 |
| make an end-of-file mark and | | |
| make an end-of tape mark | | |

CASSETTE (continued)

depending on how the file was opened.

All files which have been opened should be closed before a program ends and before you remove the cassette. If you do not close the file, your files may become garbled. For example, if you are writing a file on a previously used tape, and you don't close it, then when you read it later, you will either have a read error or you will read past the file's end into the old (and meaningless) junk on the tape.

`PRINT# [logical file #] , [variable list]`

`PRINT#` writes an exact copy of the characters produced by an equivalent `PRINT` statement. This includes graphics, upper/lower case and cursor control characters.

NOTES: `10 FOR J = 1 TO 100:PRINT "X";:NEXT J` will print 100 successive "X"s on the display. Though this will appear as 2-1/2 lines of "X" on the screen, no carriage returns are present at the ends of the first two lines. When `10 FOR J = 1 TO 100:PRINT#1,"X";:NEXT J` is executed, 100 successive "X"s are written onto the cassette with no carriage returns.

Some warnings are worth noting. First, `?#` will not work. If you get a syntax error and the `LIST` gives a correct appearing line, try retyping the line using `PRINT#`. Do not use the screen editor unless you type "PRINT" over the `PRINT` which appears on the screen.

CASSETTE (continued)

If you intend to read the file later, using the INPUT# command, be sure that carriage returns are liberally included and that no more than 79 characters in succession appear without a carriage return. This applies only to INPUT#, and not to GET#.

INPUT# [logical file #] , [variable list]

This reads tape exactly as if it were the keyboard. As with keyboard input, a maximum of 80 characters, including the carriage return, may be entered.

This means the tape cannot have more than 79 successive characters without a carriage return if it is to be read successfully with INPUT#.

WARNING:	WARNING:	WARNING:	WARNING:	WARNING:
----------	----------	----------	----------	----------

If you attempt to INPUT# from a tape with more than 79 characters between carriage returns, BASIC will either go away entirely (CRASH) and you'll have to turn the power off and start over, or strange errors and unidentified flying glitches may appear.

Note that the 79 character limitation is due to an 80 character input buffer and is not currently modifiable.

GET# [logical file #] , [string or numeric variable]

This reads the tape one character at a time. There are two varieties:

CASSETTE (continued)

1. GET# [logical file#] , [numeric variable]
 - if the character in the file is a digit (0 - 9), then the numeric variable will be set to the value of the digit.
 - if the character is one of these: + - blank then the numeric variable is set to zero.
 - any other character or the end-of-file marker will produce a syntax error. And THIS error will NOT tell you the line number in which it occurs!!

2. GET# [logical file#] , [string variable]

This reads the file, one character at a time, and returns the character in the string variable as a one character string.

If the status word is not checked, successive applications of GET# will read the file past the end-of-file mark.

NOTE: It is not necessary to write carriage returns at 79 characters or less if you read your data back with a GET# command.

STATUS WORD

The Status word can be checked after each I/O operation for certain conditions.

To detect the status, use the "AND" operation in BASIC as shown in the examples in this bulletin. Another way to do it is 10 GET#1, A\$:B\$=B\$+A\$:IF NOT((ST)AND64)THEN 10 which will read data into B\$ until an EOF is encountered.

CASSETTE (continued)

The status word ST is updated each time there is an I/O operation, with a code indicating the outcome of that operation. To indicate a unique condition, one bit is set at a time. Multiple bits may be set so it is necessary to break down the decimal number returned into its binary powers to determine which bits were set. For example, if ST = 56, then bits 8, 16, and 32 were set: $56=32+16+8$

STATUS CODES FOR TAPE I/O

4	Short Block
8	Long Block
16	Unrecoverable read error
32	Checksum error
64	End of file
128	End of tape

SHORT BLOCK (4) When reading a block from tape, shorts (the delimiter between blocks) were encountered before the expected number of bytes had been read from that block. Possible cause: attempting to read a short load file as a data record.

CASSETTE (continued)

LONG BLOCK (8) When reading a block from tape, shorts were not encountered after the expected number of bytes had been read from that block. Possible cause: reading a long load file as data.

UNRECOVERABLE READ ERROR (16) ****FATAL ERROR**** Return to BASIC and print error message. Cause: More than 31 errors on the first block of redundant blocks - or - an error that could not be corrected because it occurred in the same place in both blocks.

CHECKSUM ERROR (32) After a LOAD or reading of data, a checksum is computed over the bytes in RAM and compared to a byte received from the input device. If they do not match, this bit is set. Possible cause: faulty RAM - or - multiple bit error in data transmitted. This bit is also set if data or program fails a verify operation.

END OF FILE (64) This bit is set when an attempt to read data from a tape file is made when there is no more data.

END OF TAPE (128) ****FATAL ERROR**** An EOT record was found before the file being searched for was encountered.

All contents of this bulletin - including all software -
have been prepared by Gregory Yob, Software Editor for Commodore.

APPENDIX TO THE CASSETTE TUTORIAL

Making the Files Write Reliably

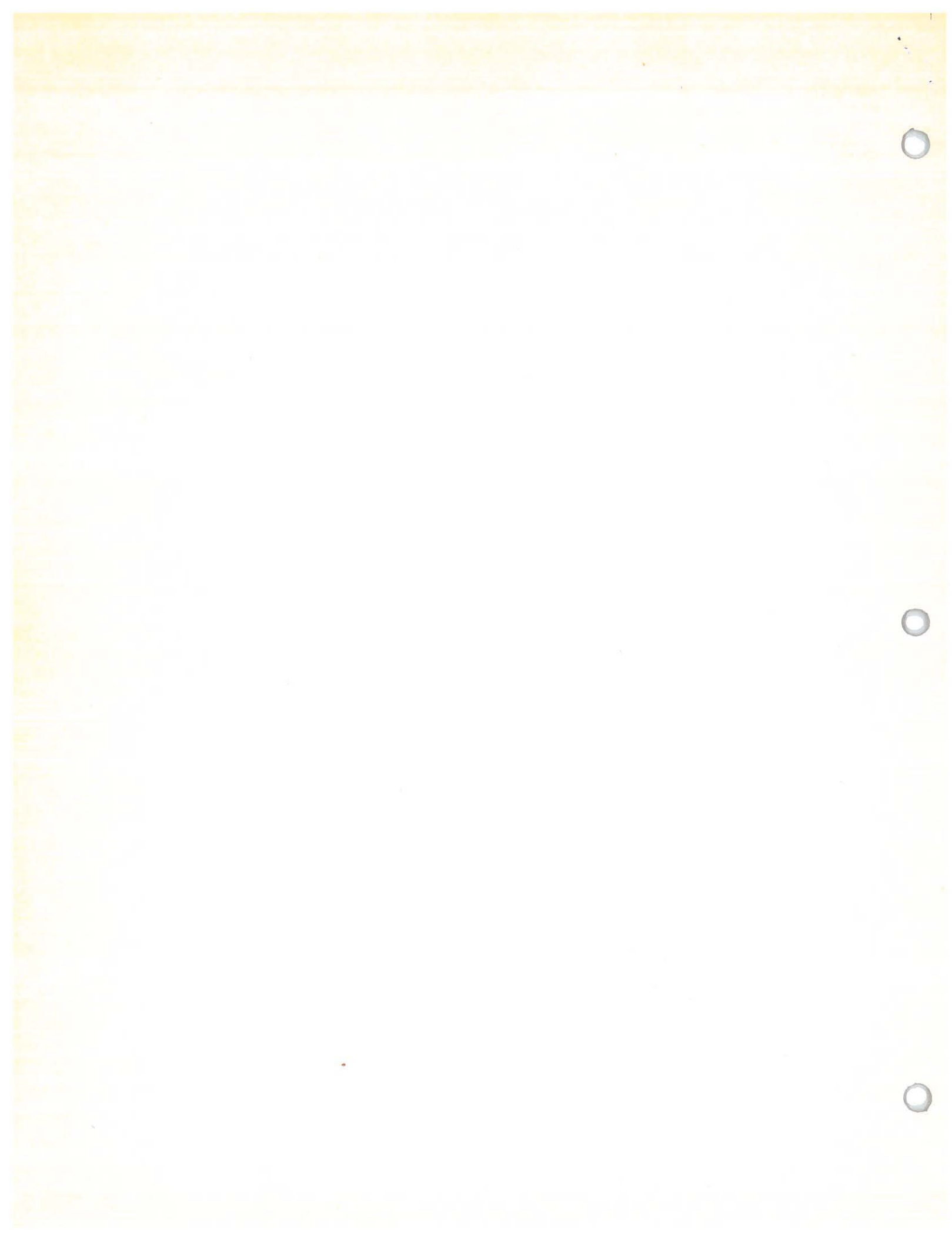
When writing long files, you will often lose records when you try to read them later. When the PET writes several records without delays between records, not enough space is left on the tape between records. When the read is attempted, some of the records are seen as 'bad', even though they aren't. The PET discards these records, leaving you with lost data.

This problem is definitely an operating system bug in the PET, which is being worked on. Meanwhile, we have found a fix you can use when writing long data files.

The way it works is to detect when the PET has just written a record on the file. At this point, the cassette motor is turned on for 5 jiffies (a jiffie is 1/60 of a second) to advance the tape a small amount after the record is written to the tape. The cassette buffer holds 191 characters, so when more than 191 characters have been written onto the file, it is time to turn on the motor.

The following little program shows how this is done:

```
10 FS=-1:FD=-1:FC=191
20 OPEN 1,1,1
30 FOR I=1 TO 1000
40 PRINT #1,I
50 B$=STR$(I)
60 J=LEN(B$)
70 FE=INT((FS+J)/FC)
80 IF FE=FD GOTO 130
90 POKE 59411,53
100 T=TI
110 IF (TI-T)<5 GOTO 110
120 POKE 59411,61
130 FS=FS+J
140 NEXT I
150 CLOSE 1
```



APPENDIX, CONTINUED

Line 50 obtains the length of the number being written to the file. If you are writing strings, be sure to find their length as well. FC contains the length of the buffer, 191 characters. The variable FS is used to store the total number of characters written onto the file. Line 70 sets FE. FE will be -1 if the buffer is not yet full, and 0 if it is filled.

Line 90 turns the cassette motor on. Line 110 waits for 5 jiffies, and line 120 turns the motor back off. If you are using Cassette #2, a different address will have to be POKEd. Line 130 updates the characters count.

This program can be used to test if the above program worked correctly. It just checks to see that the numbers being read are in increasing order by ones (1,2,3,4 ...etc).

```
10 OPEN 1
20 FOR I = 1 TO 1000
30 INPUT#1,A
40 IF (ST) AND 64 THEN 80
50 PRINT A
60 IF A <> B+1 THEN PRINT "ERROR" : J=J+1
70 B=A : NEXT I
80 CLOSE 1
90 PRINT J "ERRORS"
```



commodore



COMMODORE BUSINESS MACHINES, INC.
901 CALIFORNIA AVENUE
PALO ALTO, CALIFORNIA 94304
TELEPHONE: (415) 326-4000 TELEX: 345-569
CABLE ADDRESS COMBUSMAC PLA

SOME QUESTIONS AND ANSWERS

QUESTION: Will COMMODORE help me design a program or a system for my specific application?

ANSWER: No. Manuals and bulletins are in the works; a software library will be available soon. COMMODORE cannot afford to help each person design for his specific needs.

QUESTION: How do I get an array of graphics to print on the CRT?

ANSWER: Use a semicolon in your PRINT statement (PRINT "...;"). It will look like you built a string and printed it. Maximum string length is 255 characters, and prints in 3.2 lines. If you're trying to print all of PET's graphics, you may be blanking the screen when you try to print CHR\$(146), which is a "clear screen" character.

QUESTION: PET prints .003 as 3E-03. Can I suppress scientific notation?

ANSWER: Yes. You'll have to write a formatting program to do it.

QUESTION: Why are the squares of integers not integers? For example, $7^2 = 49.0000001$ while $\text{EXP}(\text{LOG}(7)*2) = 49$.

ANSWER: Logarithms are use and there are built-in round off problems in binary representation of decimal numbers.

QUESTION: How can I get around the 255 element array limitation?

ANSWER: A) For multidimensional arrays, use separate arrays. For example, DIM A(100,3)→DIM A(100), B(100), C(100).

B) Pack your values, two or three to an element.

C) Change your algorithm to not require arrays.

QUESTION: Can PET do matrix arithmetic?

ANSWER: You will have to write a program to do it.

QUESTION: Why doesn't the OTHELLO program from October BYTE work?

ANSWER: To many GOSUBS without returns. PET can only accept 26 levels of GOSUB nesting.

QUESTION: Can I write my own tape header?

ANSWER: Yes. Just SAVE"FILENAME", then LOAD"FILENAME". Or, to use data files, OPEN 1, 1, 1,"FILENAME". "FILENAME" can be a string; i.e., F\$.

QUESTION: How can I create a data file on tape?

ANSWER: The Cassette Bulletin will give you the answer to this one.

QUESTION: Do you have to Fast Forward off a cassette leader before saving a program?

ANSWER: No. The operating system software provides about 7.5 seconds to move the tape off the leader before beginning recording of data.

QUESTION: How fast is cassette data storage?

ANSWER: The data rate is 30-50 CHAR/SEC.

QUESTION: What is the tape format?

ANSWER: The format is a unique COMMODORE scheme.

QUESTION: What does PET look for on tape when it searches?

ANSWER: The header block on the tape file.

QUESTION: How many files can be open at one time?

converted to floating point, and if assigned to an integer variable, the result is appropriately truncated or left alone.

QUESTION: Can you program in machine language from BASIC and not use a monitor?

ANSWER: Yes. By using the POKE command, it is possible to load RAM. The process can be automated with a BASIC loader program which contains the bytes of the machine code program in DATA statements. To be safe, poke into cassette buffer #2.

QUESTION: How is SYS used?

ANSWER: The parameter for SYS is a decimal address. This is evaluated and used as a target for a JMP instruction. Return to BASIC via RTS.

QUESTION: How is USR used?

ANSWER:

1. POKE the address of the subroutine
 - location 1 gets the low byte
 - location 2 gets the high byte
2. Call USR (i.e., A=USR(I))
3. The parameter is evaluated and placed in the floating accumulator.
4. The function value is returned in the floating accumulator.
5. Return to BASIC via RTS.

QUESTION: How do you get lower case letters?

ANSWER: POKE 59468,14 for lower case

POKE 59468,12 for graphics

Lower case letters and graphics cannot be displayed on the screen simultaneously. Only use masks 12 and 14 or you may disable the keyboard interrupts. The POKE command sets a chip address select on the character generator ROM.

QUESTION: Where is BASIC text in memory?

ANSWER: It begins at \$0400 and extends to \$0FFF or \$1FFF, depending on whether it is a 4K or an 8K PET.

QUESTION: Where are variables stored, and can they be passed from one program to another?

ANSWER: During program execution, strings are created and stored downward from highest memory. Integers and real numbers are stored upward from the end of BASIC text. They may be passed to an overlay program if the overlay is less than or equal in size to the program which initiated the LOAD.

QUESTION: How do I use the diagnostic routines?

ANSWER: Special hardware is required which is currently available only to dealers and service people.

QUESTION: Where and when can I get the necessary hardware to run the diagnostic routines?

ANSWER: Only authorized PET service people will have the required hardware for the present.

QUESTION: Can I get an O.S. source listing or a BASIC source listing?

ANSWER: This will be discouraged for a purpose of maintaining software compatibility between PET users.

QUESTION: What level of BASIC is provided in PET's ROMs?

ANSWER: PET BASIC is very close to MITS BASIC by Microsoft, and has been expanded in the area of I/O and arithmetic precision.

QUESTION: Does PET have a SORT function?

ANSWER: No. SORTing must be done by a BASIC program. See Knuth, "The Art of Computer Programming" for a variety of algorithms.

QUESTION: Is PET base page limited?

ANSWER: No. At the BASIC programming level this is transparent to the user. In machine code programming page 0 is always at a premium.

QUESTION: How does PET compare strings?

ANSWER: In alphabetical order according to ASCII code, for example,
"A"<"AA" and "ABCD"<"ABCE"

QUESTION: Is the screen refreshed from a specific 1K of memory?

ANSWER: Yes, starting at \$8000.

QUESTION: Can I POKE the locations for cursor control?

ANSWER: We do not recommend using POKE to control the cursor. The cursor is controllable from the keyboard cursor control keys, and from Basic.

QUESTION: Can PET be reset without destroying RAM content?

ANSWER: No.

QUESTION: Can PET be switched from 110v and 60Hz to 220v and 50Hz?

ANSWER: Use an adaptor such as can be used with a hair dryer for travel abroad.

QUESTION: What is the PET's power consumption?

ANSWER: 150 watts.

QUESTION: Why is the PET only expandable to 32K RAM?

ANSWER: Because the upper 32K is reserved for O.S., I/O, and ROM, and the 6502 can only address 65K.

QUESTION: Is the 6502 a tristate chip?

ANSWER: The 6502 has some lines which are tristate and some which aren't. Contact MOS Technology for specs.

QUESTION: Does PET disrupt TV or radio?

ANSWER: PET is extremely well shielded and emits very little RF interference. The only time you may notice it is if you place a TV set inches away from PET and tune to an extremely weak station. Try that with a pocket calculator or a digital clock!

QUESTION: How do you access the user port?

ANSWER: The user port is on a MOS 6522. The simplest I/O is accomplished by POKEing a data direction register at 59459 and then PEEKing or POKEing a data register at 59471. The I/O logic levels are TTL.

QUESTION: Can the IEEE-488 be adapted to S-100?

ANSWER: The IEEE-488 is an I/O peripheral bus. The S-100 is a memory bus. They are not the same thing. PET does have a memory expansion bus which can be adapted to drive many S-100 peripherals.

QUESTION: What changes need to be made to an HP printer to get it to work on the PET?

ANSWER: Most HP instruments work on the IEEE bus which PET supports. We have tested a thermal printer (HP 5150A) and an impact printer (HP 9871A) successfully. Use an edge-card connector instead of the standard pin connector.

QUESTION: How do you get a listing on an IEEE printer?

ANSWER: Essentially: Open the file, tell the device to "listen", and then LIST.

```
OPEN 4,4      establish output channel - open file
CMD 4         create alternate output device - tell the
              device to listen
LIST          list to that device
CLOSE 4       close alternate channel - close the file
```

QUESTION: Can I use someone else's RAMs to build my own memory board?

ANSWER: Yes. See the memory expansion pinout.

QUESTION: Can PET be hooked to a terminal?

ANSWER: An IEEE-488/RS232 interface is in the works.

QUESTION: What causes my PET's CRT to get the 'jitters'?

ANSWER: Probably the 12v regulator. Write to PET service.

QUESTION: Why won't my PET load and save my programs?

ANSWER:

1. Are you using bad tapes?
2. Have you fully rewound the tape before a save or load?
3. Have you recently cleaned and demagnetized the deck heads?
4. If every one of these questions is answered correctly and PET still won't read tapes, it could be due to poor alignment to the read/record heads. Check with PET service.

QUESTION: If the RETURN key glitches out in the middle of a program, how can I save myself? Do I HAVE to reset?

ANSWER:

1. If the cursor can be seen, press RETURN.
2. If the cursor can't be seen, press the RUN/STOP key.
3. If neither works, you must reset. Check for possible hardware malfunction. Is the keyboard connector firmly attached to the main board?
4. And, if all else fails, check to be sure you haven't left any tape or printer files open. PET may be sending the RETURN to the file.

QUESTION: Is COMMODORE going to send a Monitor tape to all PET owners as part of the purchase package?

ANSWER: Yes. It is similar to the TIM for the 6502.

QUESTION: Will COMMODORE be bringing out a bigger CRT?

ANSWER: Probably not. You can use your own monitor on the user port.
See the pinout for this port in this issue.

QUESTION: Will Commodore be making a cassette with a counter?

ANSWER: Possibly, but not for a while.

QUESTION: Will COMMODORE be making a bigger keyboard?

ANSWER: Yes. On a bigger PET. With a bigger price tag.

QUESTION: If I buy a printer from someone else, will COMMODORE help
me get it running on the PET?

ANSWER: No. An RS232/IEEE-488 interface is in the works.

QUESTION: Will COMMODORE provide a monitor and a disassembler as part
of the purchased package?

ANSWER: We'll provide the monitor. The disassembler is already in
the public domain, and you can use it.

QUESTION: What peripheral is COMMODORE planning for the next year?

ANSWER: Second cassette, printer, floppy disk, telephone modem, and
memory board, to start. More will be contemplated later.



ANIMATING YOUR PET

It's easy to move an object smoothly across the CRT, thanks to PET's programmable cursor controls. The listings below give you the fundamental right-left-up-down motions, **late we will show you** how to program an unidentified flying object.

Move a ball right & left across the screen.

```
10 PRINT "CLR";
```

clear the screen

```
20 FOR I = 1 TO 39
```

```
30 PRINT "  ● ← " ;
```

← cursor left

```
40 NEXT
```

```
50 FOR I = 1 TO 39
```

□ blank

```
60 PRINT "□ ← ← ● ← " ;
```

we used a ball ●

```
70 NEXT
```

(shifted "Q" character)

```
80 GO TO 20
```

for simplicity

Move a ball up & down on the screen

```
10 PRINT "CLR";
```

```
20 FOR I = 1 TO 24
```

```
30 PRINT "□ ← ↓ ● ← " ;
```

cursor down ↓

```
40 NEXT
```

```
50 FOR I = 1 TO 24
```

```
60 PRINT "□ ← ↑ ● ← " ;
```

cursor up ↑

70 NEXT

80 GOTO 20

And, of course, moving a ball diagonally across the screen (top left to bottom right)

10 PRINT "CLR" ;

20 FOR I = 1 TO 24

30 PRINT " " "←" "←" "↓" "●" "←" ;

40 NEXT

50 FOR I = 1 TO 24

60 PRINT " " "↑" "●" "←" ;

70 NEXT

80 GOTO 20

The characters which will appear on the screen in places where we've used arrows in these listings will appear as PET graphic characters:

	PET SHOWS		WE USE
reverse field heart	♥	clear screen	CLR
reverse field right bracket]	cursor right	→
reverse field dot or ball	●	cursor up	↑
reverse field capital Q	Q	cursor down	↓
reverse field capital S	S	home cursor	HOME
reverse field square with line graphic	□	cursor left	←