Gregory Yob
Box 354
Palo Alto CA 94302

PET User Port Cookbook

This is a sneak preview of part of The PET Manual *by author/publisher Greg Yob. Greg is taking pre-publication orders now and says the book will go into printing on April Fool's Day.*

Note: Pre-publication orders for The PET Manual *are $16, plus $2 for shipping, and can be sent to Mind's Eye Software, PO Box 354, Palo Alto CA 94302. No checks will be cashed until the book is printed; price will increase after printing.*

The PET personal computer has several expansion capabilities, including one known as the *user port.* This is a set of eight bidirectional lines and two handshake lines intended as a parallel port for the hobbyist to use in his experimental projects. Commodore has not released much information regarding the user port, and the object of this article is to explain the user port and its use.

Fig. 1 shows the location of the user port on the back of the PET and the pin-out of the PC edge. If you do not have a 12-position, 24-contact edge connector, use a larger one and cut it off to the 12-position size. If you do this, be sure to insert a polarization key in your connector; I found that it was easy to misalign a sawed-off connector with the PC edge, causing various mysterious glitches. Also, be sure that the top and bottom connections are really separate—the upper edge has a variety of signals that will interfere with the correct operation of the user port.

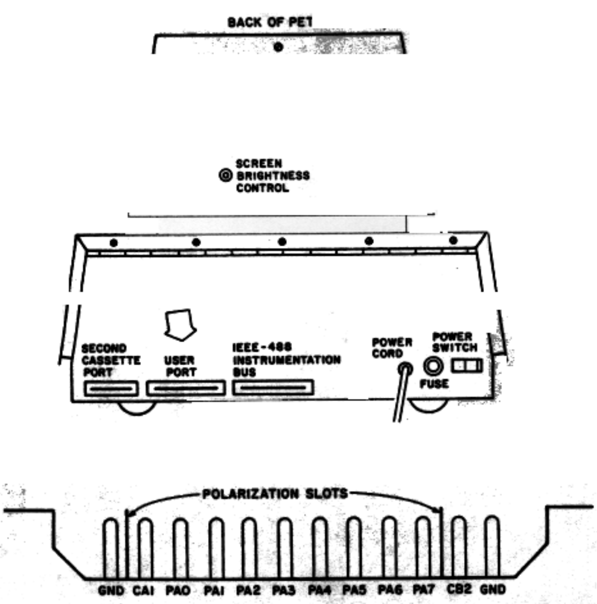

Fig. 1. The user port—location and pin-out. The user port pin-out as seen from the top. The user port pins are on the bottom of the PC card edge. The pins on top carry a variety of signals that are not related to the user port. Electrically, the lines correspond to one TTL source or load, depending on whether the line is in output or input mode. Use buffering or short cables if high data rates are required. The CB2 line does not have a pull-up resistor, so you may have to provide one if you are using CB2 in input mode.

The pin designations correspond to those on a MOS 6522 VIA (Versatile Interface Adapter), which is a complex LSI I/O chip produced by MOS Technology. (Write MOS Technology, 950 Rittenhouse Road, Norristown PA 19401, for the specification sheet.) The user port is connected directly to the VIA within the PET, and the lines are capable of sourcing or sinking *one* TTL load. If your application calls for a high data rate, note that your cables should be short or some buffering will be required.

As with all of the 650X microcomputer systems, the input and output appear to the microprocessor as a group of memory locations. PET's BASIC does not have any PRINT or INPUT statements for the user port, which requires you to use the PEEK and POKE statements. This also places another limitation, that is, BASIC's speed, which limits I/O through the user port to around 50 characters per second. If you want to use a more rapid rate, you must use machine language.

Since this article is concerned with the mechanics of using the user port, most of the examples will be in BASIC. Table 1 shows the memory locations for the 6522 in the PET.

At this point I must warn you: all of the other VIA lines are used within the PET for internal uses. If you fail to restore the VIA to its original state when you are finished, you will find that the PET behaves strangely, especially when dealing with the tape drives.

When I wrote the program for display of the VIA registers (which you will see later on), I didn't save it until I had it debugged. The PET wouldn't verify or even find the copy I had tried to save, and after handwriting the program, I realized the next morning that the VIA registers were not in their original states. Fortunately I had left the PET on overnight, and when I restored the registers, I was able to save the program.

The Blinkin' Lights Machine

For experimentation with the user port it is convenient to build a miniature "front panel" to indicate the state of each line and to control the lines via manual switches. A breadboard and some $20 worth of parts (bought at the local costly retail outlet) provided a handy "Blinkin' Lights Machine" that hooked to the user port and used the +5 volt supply from the second cassette drive.

```
10  REM SIMPLE OUTPUT EXAMPLE
20  REM SET DATA DIRECTION REGISTER TO OUTPUT
30  POKE 59459,255
40  REM COUNT FROM 0 TO 255
50  FOR J=0 TO 255
60  REM POKE TO OUTPUT REGISTER
70  POKE 59471,J
80  NEXT J
90  REM DO IT AGAIN
100 GOTO 50
```

Example 1. Simple output example for user port.

Name	Address(hex)	Address(decimal)	Function
ORB	E840	59456	## (internal to PET)
ORA	E841	59457	Data with Handshake
PDRB	E842	59458	##
DDRA	E843	59459	Data Direction
T1L-W	E844	59460	##
T1C-H	E845	59461	##
T1L-L	E846	59462	##
T1L-H	E847	59463	##
T2L-W	E848	59464	##
T2C-H	E849	59465	##
SR	E84A	59466	Shift Register
ACR	E84B	59467	Auxiliary Control
PCR	E84C	59468	Peripheral Control
IFR	E84D	59469	Interrupt Flags
IER	E84E	59470	Interrupt Enable
ORA	E84F	59471	Data (no handshake)

Table 1. PET VIA register addresses. The named registers may be used to work with the user port. Some of the settings used may disable other PET functions, such as tape I/O, so you should restore the original settings when you are done. The registers with "##" in the Function column are used internally by the PET. If you are bold, there are two other I/O chips in the PET. These are MOS 6520s, with one starting at $E810 (59408) for internal uses and one at $E820 (59425) for the IEEE-844 bus.

Note that the circuit draws 200 mA, which is close to the maximum you can steal from the PET. If you have other PET extensions that use the PET supply, power the Blinkin' Lights externally.

```
10  POKE 59459,255
20  K=1
30  POKE 59471,K
40  FOR J=1 TO 200 : NEXT
50  K=K*2
60  IF K=256 THEN 20
70  GOTO 30
```

Example 2. Another simple output example.

Fig. 2 shows the circuit for the Blinkin' Lights Machine. The extra inverter and capacitor on the CB2 line are for an audio output to attach to your hi-fi set for some simple music making. One of the best ways to build this device is on a Vector breadboard, which has the fingers for an edge connector. This permits putting the Blinkin' Lights in series with a device under test to help with debugging the interface software and hardware.

Most of the examples shown below make use of the Blinkin' Lights Machine, so building one might be handy.

Simple Output

The simplest thing to do is output bytes to the user port. To do this, you must first set the Data Direction register to 255 (all bits set) and then set the Output register to the byte(s) that are to be output. Example 1 is a short program that counts from 0 to 255 and outputs the count to the user port.

The Data Direction register controls the PA0 through PA7 lines' data direction. If the bit is set for a given line (i.e., bit 0 is for line PA0), the line will be an output. If the bit is zero, the line will be an input.

When the PET is turned on with the Blinkin' Lights attached, all the LEDs will be lit. The PA0-PA7 lines are initially set for input, and the Blinkin' Lights will see lines in the high-impedance state as "high" (pulled up by the 7404s), turning on the LED for the line.

When the program (Example 1) is RUN, the data lines show that a binary count appears, which cycles through about once every three seconds. To slow the rate down so that the least significant bits (PA0 and PA1) will change state, add:

```
65  FOR K=1 TO 50 : NEXT
```

This will slow the counting loop down to around 10 Hz.

To see the effect of changing the Data Direction register, change line 30 to:

```
30  POKE 59459,15
```

Now the lines PA0-PA3 will count, and lines PA4-PA7 will remain lit (recall that an unconnected line will float to high with the Blinkin' Lights).

Example 2 shows another short program. Try it and see what it does! Note that in PET BASIC the NEXT statement may omit the loop counter if the innermost loop is being terminated. Another diversion is to change the program in Example 2.

```
20  K=1 : L=128
30  POKE 59471, K OR L
50  K=K*2 : L=L/2
```

(Just change these lines and let the others remain the same.)

Simple Input

To see simple input, POKE the Data Direction register to input mode and connect the switches to the PA0-7 lines. Note that the Blinkin' Lights has some DIP switches to isolate the manual switches from the data lines. This is because if they were always tied in, the switch setting would force the line to the switch's state.

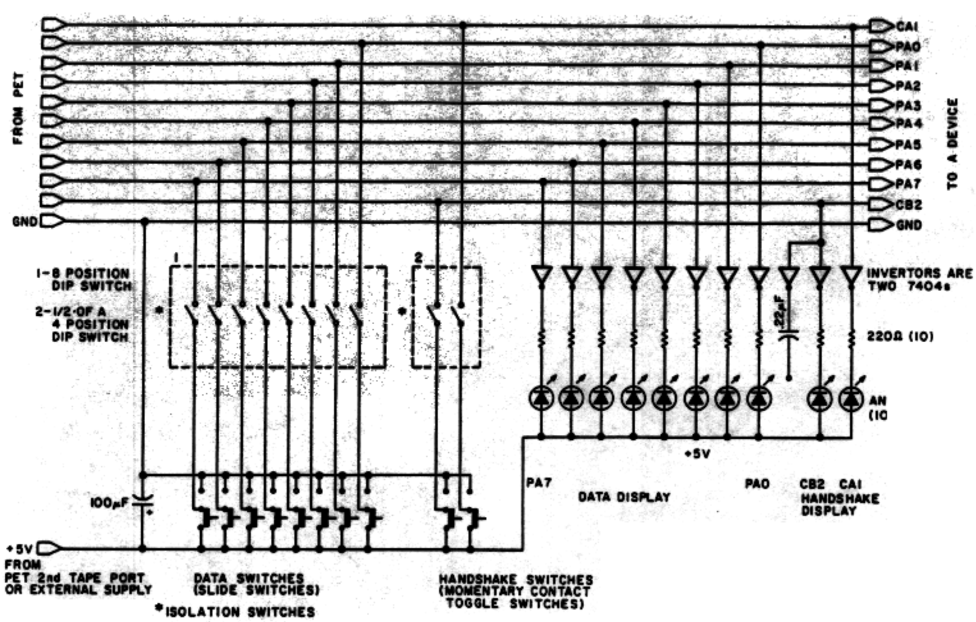

Fig. 2. Blinkin' Lights—PET user port switch register and indicator.

Shown	What It Represents
b	SPACE character (when not clear)
⊡	A lowercase character in a square box represents the corresponding graphics character. For example, [a] is the spade graphics character, or SHIFT-A:
Ⓒ	Clear Screen
Ⓗ	Home Cursor
Ⓤ	Cursor Up
Ⓓ	Cursor Down
Ⓡ	Cursor Right
Ⓛ	Cursor Left
Ⓘ	INST key
Ⓓ	DEL key

Table 2. PET program listing special characters.

Data Register	DATA	59471
Data Register, Handshake	HDATA	59457
Peripheral Control Register	PCR	59468
Auxiliary Control Register	ACR	59467
Interrupt Flag Register	IFR	59469

Table 3.

Then, PEEK the Data register and display the result on the PET display screen in a loop. As you change the switches, the number displayed will change. Example 3 is a program that does this. (Note: Table 2 shows how this article represents PET listings.) Line 70 homes the cursor and prints the value of the Data register. It then prints a CURSOR LEFT and three blanks. The reason for the CURSOR LEFT is that the PET has an oddity when it prints numbers onto the screen. When a number is printed, the format is: (SPACE or +)(Digits of Number)(CURSOR RIGHT).

When a short number is printed over a longer one, the printing stops after the CURSOR RIGHT. It is necessary to erase the old numbers with some blanks, so the cursor is moved left once and three blanks are printed. This prevents spurious numbers, such as "328," appearing on the display. (Try it, you won't like it!)

RUN this program and try the manual switches one at a time. You should see the sequence 0, 1, 2, 4, 8 . . . 128 appear on the PET screen.

If you set all the manual switches to zero and disconnect one of them with the DIP switch, the line will go high and the PET will see the bit as set. Be careful of this when you are using the Blinkin' Lights for debugging.

Joysticks

A simple and enjoyable way to use the user port is to attach a switch-operated pair of joysticks to your PET. Each joystick has four switches—one for each direction—that are closed when the stick is pointed that way. Fig. 3 shows a joystick circuit.

The program in Example 4 sets up the screen with a solid and hollow ball. Each joystick controls one of the balls, and both balls may be in motion at the same time. The switches and bit settings are the same as in Fig. 3.

Lines 170 and 180 clear the screen and print the character for the right and left joysticks. The PEEK sets the cursors (C1 and C2) to the value needed for use by POKE later. The value 32768 is the first address in memory in the display, which occupies memory locations 32768 to 33767.

```
10  REM SIMPLE INPUT EXAMPLE
20  REM SET DATA DIRECTION TO INPUT
30  POKE 59459,0
40  REM CLEAR SCREEN
50  PRINT " Ⓒ ";
60  REM PEEK DATA REGISTER & SHOW IT
70  PRINT" Ⓗ "PEEK(59471)" Ⓛ bbb";
80  REM DO IT AGAIN
90  GOTO 70
```

Example 3. Simple input example for user port.

Line 260 fetches the data from the user port. Since the joysticks ground the lines to indicate switch closures, the byte is complemented. It is then ANDed with 255 to return to eight bits, as the integer operations of the PET are 2's complement for 16 bits.

In Line 2010, the value for Z must be shifted right by four bits. This is done by dividing by 16 and truncating.

Lines 3020 and 3140 place a blank and the cursor, respectively, on the screen. The multiplication by 40 for Y is because the PET screen is 40 characters wide. If you delete line 3020, the motions of the joysticks will leave trails and let you draw pictures.

Transferring Data with Handshakes

The CA1 and CB2 lines permit data transfer with full handshaking for input and output. The 6522 VIA has a variety of options, and these are controlled by the registers in Table 3. In the 6522, the Peripheral Control register and the Auxiliary Control register select the various options for the operational modes for the VIA. Some of these bits affect the CA1 and CB2 lines and will be described in detail later.

The Interrupt Flag register has bits for the detection of several conditions that may be used for interrupts. In the PET, the use of the interrupts is a hazardous affair, as the PET has a 60 Hz internal interrupt, which handles various housekeeping tasks such as scanning the keyboard and maintaining the internal clock. Since these functions can only be handled in machine language, this article will not discuss how to handle the Interrupt Enable register.

To detect a condition, such as the transition of the CA1 line, PEEK the Interrupt Flag register and AND for the desired bit. The bit in the Flag register will remain set until other actions are taken, usually the reading or writing of data through the Data Handshake register.

If the above sounds confusing, that is because it *is* confusing, and with this in mind, you should attempt the examples in the following sections when you try to use the PET user port.

Using CA1

The CA1 line is an input-only line usually used to detect the handshakes for data transfers. For example, if a device is send-

```
5 REM BY GREGORY YOB, MAY 1978
10 REM DUAL CURSORS FOR JOY-STICKS
20 REM ATTACHED TO USER PORT WITH
30 REM BITS AS FOLLOWS:
40 REM LINE GROUNDED MEANS SWITCH IS
50 REM CLOSED AND TO MOVE CURSOR
60 REM  BIT 7 = LEFT STICK  UP
70 REM   "  6 =             DOWN
80 REM   "  5 =             RIGHT
90 REM   "  4 =             LEFT
100 REM  "  3 = RIGHT STICK UP
110 REM  "  2 =             DOWN
120 REM  "  1 =             RIGHT
130 REM  "  0 =             LEFT
140 REM DISPLAY IS WRAPAROUND
150 REM
160 REM PUT YOUR OWN CURSORS HERE
170 PRINT"Ⓒ [q]";:C1=PEEK(32768)
180 PRINT"Ⓒ [w]";:C2=PEEK(32768)
190 REM INITIALIZE SCREEN & POSITIONS
200 PRINT"Ⓒ";
210 X1=4:Y1=12:X2=35:Y2=12
220 POKE 33252,C1:POKE 33283,C2
230 REM SET UP DATA DIRECTION REG
240 POKE 59459,0
250 REM LOOK AT PORT
260 P=NOT(PEEK(59471))AND 255
270 REM CHECK RIGHT & LEFT
280 IF P AND 15 THEN GOSUB 1000
290 IF P AND 240 THEN GOSUB 2000
300 GOTO 260
500 REM ROUTINES 1000 & 2000 SET UP
510 REM X,Y = POSITION
520 REM Z = SWITCH SETTINGS
530 REM C = CURSOR CHARACTER
540 REM FOR ROUTINE 3000 WHICH
550 REM DOES MOVING & WRAPAROUND
560 REM
1000 REM RIGHT STICK
1010 X=X1:Y=Y1:Z=P AND 15:C=C1
1020 GOSUB 3000
1030 X1=X:Y1=Y:RETURN
2000 REM LEFT STICK
2010 X=X2:Y=Y2:Z=INT((P AND 240)/16)
2020 C=C2:GOSUB 3000
2030 X2=X:Y2=Y:RETURN
2500 REM
3000 REM MOVE CURSOR
3010 REM ERASE OLD ONE
3020 POKE 32768+40*Y+X,32
3030 REM FIND NEW POSITION
3040 IF Z AND 8 THEN Y=Y-1
3050 IF Z AND 4 THEN Y=Y+1
3060 IF Z AND 2 THEN X=X+1
3070 IF Z AND 1 THEN X=X-1
3080 REM WRAPAROUND CHECK
3090 IF X>39 THEN X=0
3100 IF X<0  THEN X=39
3110 IF Y>24 THEN Y=0
3120 IF Y<0  THEN Y=24
3130 REM POKE IN NEW CURSOR
3140 POKE 32768+40*Y+X,C
3150 RETURN
```

Example 4. Program to move two cursors with the joysticks in Fig. 3.

ing data to the PET, the CA1 line will be used to say that the data is now valid. If the PET is sending data, the CA1 line is used by the device to signal that it is ready for the data.

Using the CA1 line involves these steps:

1. Select the options you want and POKE the Peripheral Control register (PCR) and Auxiliary Control register (ACR) accordingly.
2. In a loop, check the CA1 Flag bit in the Interrupt Flag register (IFR) until it is set.
3. PEEK or POKE the HDATA (Data with Handshake) register with the data. This will reset the CA1 bit in the IFR.

Your options are as follows:

1. *Positive or negative transition*. CA1 will set its flag bit when the line goes high or low, depending on bit 1 in the PCR.

For a *negative* transition, use:

POKE (59468), PEEK(59468)AND 254

This is the value the PET initializes to when it is powered up. The reason it uses a PEEK instead of just POKEing to a 1 is that the other bits in the PCR should not be changed because they control other things.

For a *positive* transition, use:

POKE (59468), PEEK(59468)OR 1

2. *Latching of the input data*. If the input data is latched, the values present on the data lines will be latched when the CA1 line makes the correct transition. If the data is not latched, the values in the HDATA register will change as the data lines change. It is safest to use the latched mode when handshaking your data.

To enable latching, use this statement:

POKE (59467), PEEK(59467)OR 1

To disable latching, use:

POKE (59467), PEEK(59467) AND 254

To detect the Flag bit in the IFR, use a statement of the form:

IF PEEK(59469)AND 2 THEN ——

or

WAIT 59469,2

If you use the WAIT statement, note that the STOP key will be ignored by the PET, which means you must be *sure* that the CA1 line will make a transition—otherwise your PET will be hung up. For debugging, use the IF-THEN form. For reading or writing the HDATA register use:

PEEK (59457)

or

POKE 59457, ——

At last it is time for some examples. First, let's try counting from 0 to 255, with a wait for the CA1 line to be toggled before the next value is sent to the user port. Enter the program in Example 5, recalling Example 1.

When this program is run, the data lights will go out and will stay out until the CA1 switch is toggled. (If it doesn't, be sure that your DIP swich has been closed for CA1.) The first light (PA0) will then light, and as you toggle the CA1 swich, the Blinkin' Lights will count in binary.

Two things should be noted. First, the bounce of the CA1 switch will guarantee that *both* transitions occur, so the setting of the transition bit doesn't matter. Also, the speed of BASIC is slow enough that the bounce of CA1 doesn't cause double or more rapid counts. (If you try the equivalent program in machine language, your CA1 will count 10 to 25 times each time you flick the switch unless you have debounced it.)

Second, you can shorten your program by using the inverse condition in line 110, eliminating line 120:

110 IF(PEEK(59469)AND 2) = 0 THEN 110

Beware of the precedence of operators. If you tried:

110 IF PEEK(59469) AND 2 = 0 THEN 110

your lights would have counted up ignoring the CA1 line. The reason for this is that the operator = is evaluated *before* AND is. So, the sub-expression 2 = 0 is evaluated, giving a −1, which is ANDed with the IFR with the result that *any* bit will make the relation true. In this case, no other bits are set; the program then thinks that the CA1 line had toggled; and it drops through the loop.

Try it out—this error is quite common, and that's the reason

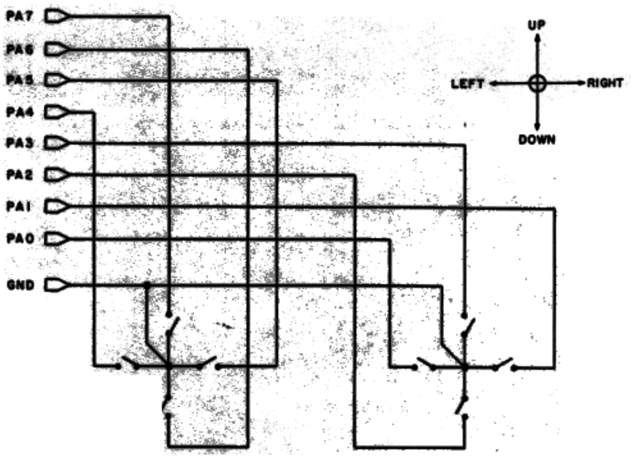

Fig. 3a. Joysticks for the PET. The switch arrangement for my PET joysticks is shown here. The switches are normally open.

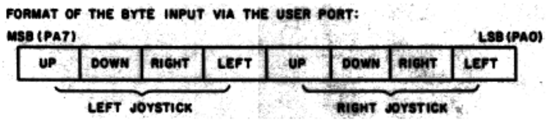

Fig. 3b. The byte input from the user port is shown here. This design exploits the fact that the PET lines PA0 to PA7 will float to high when they are disconnected. When a line goes low, the corresponding switch is closed.

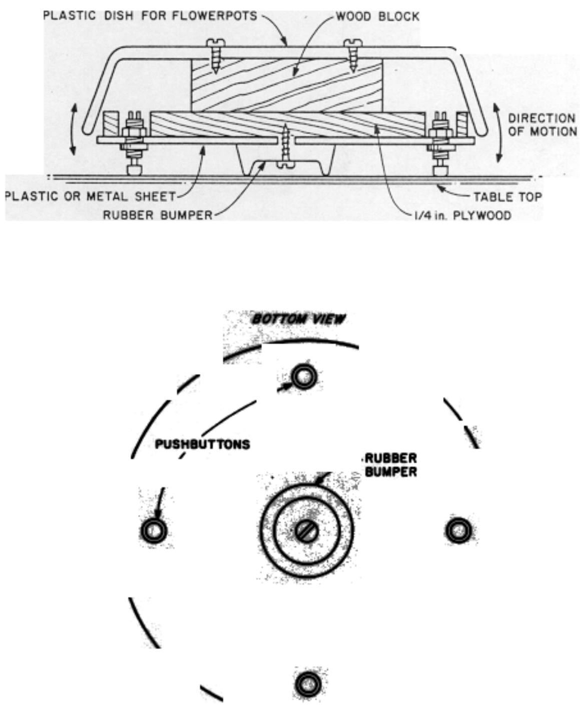

Fig. 3c. The Wobbilator—a low-cost alternative to joysticks that is easier to use as well. Eight low-cost miniature push buttons are used to build two of these units. Either normally open or normally closed push buttons may be used. (If normally closed, change lines 260 in Example 4 accordingly.) The push buttons should not be "snap action" or "detent" or go "click" when depressed, and should only move about 1/8 inch for closure. Use a bit of ribbon cable to attach the connector for the user port to the Wobbilators. Mark each Wobbilator with a dot for "Up" and "Right" and "Left." Choose a dish that fits your hand comfortably.

```
10 REM SIMPLE OUTPUT WITH HANDSHAKE
20 REM SET DDR TO OUTPUT
30 POKE 59459,255
40 REM SET POSITIVE TRANSITION FOR CA1
50 POKE 59468,PEEK(59468)OR 1
60 REM COUNT 0 TO 255
70 FOR J=0 TO 255
80 REM OUTPUT TO PORT
90 POKE 59457,J
100 REM WAIT FOR FLAG BIT
110 IF PEEK(59469)AND 2 THEN 130
120 GOTO 110
130 NEXT J
140 REM DO IT AGAIN
150 GOTO 70
```

Example 5. Simple output with handshake for PET user port. This program waits for a strobe on CA1 before sending the data from the PET.

```
10 REM SIMPLE INPUT VIA HANDSHAKE
20 REM DDR TO INPUT
30 POKE 59459,0
40 REM NEGATIVE CA1 TRANSITION
50 POKE 59468,PEEK(59468)AND 254
60 REM CLEAR SCREEN
70 PRINT" © ";
80 REM WAIT FOR CA1
90 IF (PEEK(59469)AND 2)=0 THEN 90
100 REM FETCH DATA & DISPLAY
110 C=C+1
120 A=PEEK(59457)
130 PRINT" Ⓗ bbbbbbbbbbbbbbbbbbbb Ⓗ ";
140 PRINT"COUNT"C"DATA"A
150 GOTO 90
```

Example 6. Simple input with handshake for PET user port. This program waits for a low on CA1 before accepting the data and then displays the decimal value of the data on the PET screen.

```
10 REM INPUT ASCII FROM KEYBOARD
20 REM CONVERT & DISPLAY ON SCREEN
30 GOSUB 1000: REM INITIALIZE
40 GOSUB 2000: REM GET CHAR AS A$
50 PRINT A$;
60 GOTO 40
1000 REM INITIALIZE PORT & TABLE
1010 POKE 59468,PEEK(59468)OR 1
1020 POKE 59467,PEEK(59467)OR 1
1030 DIM TB(31)
1040 FOR J=0 TO 31
1050 READ TB(J): NEXT J
1060 MD=0 : RETURN
1100 DATA 0,0,0,0,19,145,29,0,0,18,0,0
1110 DATA 0,13,0,146,0,147,0,157,0,20
1120 DATA 0,0,17,148,0,0,0,0,0,0
1130 REM
2000 REM FETCH CHAR & CONVERT
2010 IF(PEEK(59469)AND 2)=0 THEN 2010
2020 CH=PEEK(59457)AND 127
2030 REM TEST IF CTRL CHAR
2040 IF CH>31 THEN 2130
2050 REM MODE FLAG TESTS
2060 IF CH=10 THEN MD=0
2070 IF CH=27 THEN MD=128
2080 REM CONVERT VIA TABLE
2090 CH=TB(CH)
2100 IF CH=0 THEN 2010
2110 GOTO 2160
2120 REM CASE CONVERT
2130 IF CH>95 THEN CH=CH-32
2140 REM MODE CONVERT
2150 CH=CH OR MD
2160 A$=CHR$(CH): RETURN
```

Example 7. Input ASCII from keyboard, convert for all PET keys and display on PET screen. This program will accept the ASCII codes from the user port and follow the convention in Table 5 and in the text.

for this lengthy explanation. Be sure your expression is doing what you want it to, and if you aren't sure, use parentheses or try trial variations and print the results on your screen.

The next thing to try is entering a value on the data switches with the Blinkin' Lights and have the PET accept the data when the CA1 line is toggled. The program in Example 6 shows how.

When the program is run, you may set the switches to a value (be sure your DIP switches are closed or you will just get 255s), and when you toggle the CA1 switch, the count and value will appear at the top of the PET display. The count is used so you can tell when you reenter the same data value. Though the desired transition for CA1 is not important in this example, line 50 shows the opposite direction from the preceding output example. In line 140, the delimiter ";" is ignored because PET BASIC will permit this.

A Keyboard Via the User Port

As an example of a useful project for the user port, I interfaced an ASCII-encoded keyboard to the PET. Since I am a fair typist, the PET keyboard is frustrating for program entry and debugging. The following example is specific to my keyboard, but almost any full ASCII keyboard and most "Dumb Teletype" keyboards can be interfaced in a similar way.

The pin-out for the keyboard was determined and wired to the PET user port as shown in Table 4. Since the keyboard drew 500 mA, it was connected to a separate 5 volt supply.

At this point, the card edge on the Blinkin' Lights was very handy. The keyboard was connected to the Blinkin' Lights and the Blinkin' Lights *not* connected to the PET. Some investigation revealed that the keyboard did encode the parity bit and that it had a 2-key rollover.

The CA1 LED would turn on when a key was depressed and when a second key was depressed, it would flicker when the first key was released. This indicated that the strobe was a positive transition and that there was a 2-key rollover.

The keyboard was then attached to the PET, and the Simple Input via Handshake program (Example 6) was tried with line 50 changed to a positive CA1 transition. After a short warm-up, each keypress showed a value, and the rollover worked just fine.

Now that the keyboard was working electrically, a dilemma appeared: How can you emulate all the PET keyboard functions? A careful study of the PET keyboard, character set and cursor control functions reveals that there are 138 functions and that the ASCII code has only 128 characters in it.

The solution I chose (feel free to choose one of your own) was to let the control character represent the various nonprinting keys (cursor movements, RVS and so on) and to convert all other characters from the keyboard to uppercase. Since the high bit for a given PET character is set if the character is a graphics character, I decided to have a Mode flag—if you pressed ESCAPE, all further alphanumeric keys would show their graphics character, and when you pressed LINEFEED, the mode would be "normal," and the character would appear.

It should be noted that the PET character set is *not* ASCII but is similar to ASCII. This resulted in some further translation steps, and the entire conversion routine used these steps:

1. Get the character from the user port and remove the Parity bit
2. If it was a control character (0 to 31), do the following:
   (a) Find a value in a 32-element translation array for the correct PET character.
   (b) If the table value is zero, ignore and go to step 1.
   (c) Print the character on the screen and go to step 1.
3. If the character is in the range 96 to 127, subtract 32. (Converts lowercase to uppercase.)
4. If the Mode flag is set (for graphics), OR with 128 to set the highest bit.

5. Print the character on the PET.

6. Go to step 1.

Note: In step 2, if the character was an ESCAPE or a LINE-FEED, the Mode flag would be set or reset, respectively, and the table entry for these characters would be a zero.

The next thing to do was to choose the meanings for the control characters. Some control characters, such as CTRL-M and CTRL-J, were already used for RETURN, LINEFEED, etc. Keys were chosen for their convenience on the keyboard in Table 5.

The appropriate PET character values were then placed in a 32-value table for lookup by the translating routine. A BASIC program was written to test this scheme out (see Example 7). Note that RETURN is the same value, 13, as the value fetching it (i.e., CH is 13 also). In line 2020, the masking is done to remove parity when the character is read from the user port. The Mode flag is set to 0 or 128, which permits the use of OR in line 2150.

```
10 REM **** PET MACHINE CODE LOADER ****
20 REM BY GREGORY YOB, 1978
30 REM READS DATA STRINGS IN FORMAT
40 REM IDENTICAL TO PET MONITOR AND
50 REM LOADS INTO INDICATED MEMORY
60 REM LOCATIONS. FIRST NUMBER IS
70 REM START ADDRESS, NEXT 8 VALUES
80 REM ARE BYTES TO LOAD.
90 REM   IF A BYTE IS 'XX' IT WILL NOT
100 REM BE LOADED, AND MEMORY CELL WILL
110 REM BE UNCHANGED, AND NEXT BYTE
120 REM LOADED INTO NEXT CELL.
130 REM IF A BYTE IS '**' OR AN ADDRESS
140 REM IS '****', THE LOAD WILL STOP.
150 REM LINE 20000 GUARANTEES END IF
160 REM '**' OR '****' IS NOT FOUND.
170 REM
180 REM NOTE: THIS PGM MORE USEFUL IF
190 REM EXTENDED TO DATA TAPE FILES.
200 REM
300 PRINT"Ⓒ bPET LOADER PROGRAM"
310 READ A$: IF A$="END" THEN 950
315 PRINT" ⒹⒹ "A$"Ⓓ"
320 GOSUB 400 : REM GET ADDR
330 IF ADDR <0 THEN 950
340 FOR B = 1 TO 8
350 GOSUB 500 : REM GET BYTE
355 IF BYTE = -2 THEN 380
360 IF BYTE < 0 THEN 950
370 POKE ADDR,BYTE : REM DO THE DEED
375 PRINT ADDR;TAB(10);BYTE
380 ADDR=ADDR+1 : NEXT B
390 GOTO 310
400 REM ** PARSE ADDRESS **
410 B$=MID$(A$,1,4)
420 IF B$="****" THEN ADDR=-1 :RETURN
430 GOSUB 600 : REM HEX CONVERTER
440 ADDR=HEX
450 RETURN
500 REM ** PARSE BYTES **
510 B$=MID$(A$,B*3+3,2)
520 IF B$="XX" THEN BYTE=-2 :RETURN
530 IF B$="**" THEN BYTE=-1 :RETURN
540 GOSUB 600 : REM HEX CONVERTER
550 BYTE =HEX
560 RETURN
600 REM HEX CONVERTER
610 HEX=0
620 FOR H=1 TO LEN(B$)
630 H$=MID$(B$,H,1)
640 IF H$ <"Ø" THEN 900        ("Ø" is zero)
650 IF H$ >"F" THEN 900
660 IF H$ <":" THEN 700
670 IF H$ <"A" THEN 900
680 D=ASC(H$)-55 : GOTO 710
700 D=ASC(H$)-48
710 HEX=HEX*16 + D
720 NEXT H
730 RETURN
900 PRINT"ⒹⒹ #### BAD VALUE IN DATA ####"
910 PRINT"Ⓓ LOAD ABORTED":END
950 PRINT"ⒹⒹ LOAD FINISHED":END
1000 DATA"0338 XX XX 78 A9 75 8D 19 02"      (Note: all 0
1010 DATA"0340 A9 03 8D 1A 02 A9 00 8D"       are zeroes)
1020 DATA"0348 43 E8 8D C7 03 AD 4C E8"
1030 DATA"0350 09 01 8D 4C E8 AD 4B E8"
1040 DATA"0358 09 01 8D 4B E8 58 60 78"
1050 DATA"0360 A9 85 8D 19 02 A9 E6 8D"
1060 DATA"0368 1A 02 58 60 A9 00 48 48"
1070 DATA"0370 48 48 4C 85 E6 AD 4D E8"
1080 DATA"0378 29 02 D0 07 20 6C 03 EA"
1090 DATA"0380 4C 7E E6 AD 41 E8 29 7F"
1100 DATA"0388 C9 1F 10 30 C9 0A D0 07"
1110 DATA"0390 A9 00 8D C7 03 F0 E5 C9"
1120 DATA"0398 1B D0 07 A9 80 8D C7 03"
1130 DATA"03A0 D0 DA AA BD C8 03 F0 D4"
1140 DATA"03A8 EA AE 0D 02 9D 0F 02 E8"
1150 DATA"03B0 E0 0A D0 02 A2 00 8E 0D"
1160 DATA"03B8 02 4C 7C 03 C9 60 30 02"
1170 DATA"03C0 E9 20 0D C7 03 D0 E2 00"
1180 DATA"03C8 00 00 00 00 13 91 1D 00"
1190 DATA"03D0 00 12 00 00 00 0D 00 92"
1200 DATA"03D8 00 93 00 9D 00 14 00 00"
1210 DATA"03E0 11 94 00 00 00 00 00 00"
1220 DATA"03E8 ** ** ** ** ** ** ** **"
20000 DATA"END"
```

Machine-Language Source

```
        ! FOOL THE PET INTO READING THE USER PORT AS THE
        ! COMMAND KEYBOARD IN PARALLEL WITH THE NORMAL
        ! KEYBOARD BY READING THE USER PORT WHEN THE 60 HZ
        ! INTERRUPT IS SERVICED.  IF A CHARACTER IS
        ! PRESENT, TRANSLATES ACCORDING TO SCHEME DESCRIBE
        ! IN USER PORT ARTICLE AND PUTS CHARACTER INTO
        ! THE PET INPUT BUFFER.
        !         THIS CODE TAKEN FROM AN IDEA BY RICHARD
        ! TOBEY. IMPLEMENTED BY GREGORY YOB.
        !
        ! *** INITIALIZATION CODE ***
        !         TURN OFF INTERRUPTS, AND SET THE PET
        ! "INTERRUPT VECTOR" TO POINT TO THE ACTIVE CODE.
        ! SET UP THE USER PORT TO READ THE KEYBOARD, AND
        ! SET THE MODE VARIABLE TO "CHARACTER MODE" (0)
        !
        ! NOTE*** THIS CODE RESIDES IN THE SECOND CASSETTE
        ! BUFFER ( 033A TO 03FF )
        !
033A  78        XON   SEI        ! DISABLE INTERRUPTS
033B  A9 75           LDA #$75   ! SET UP NEW
033D  8D 19 02        STA $0219  !    "INTERRUPT
```

Though this program is suitable for entering data into a BASIC program, the keyboard cannot be used in direct mode, that is, entering BASIC statements or LIST, etc. Example 8 shows a BASIC program which, when run, will load a machine-language program into the second cassette buffer. When this machine-language program is executed (by SYS(826)), the keyboard attached to the user port will operate "in parallel" with the PET keyboard. If you follow the cautions indicated in Example 8, you will be able to use the auxiliary keyboard for other programs, etc.

The first program, A BASIC Machine-Language Loader, will load any machine-language code in this format: AAAA HH HH HH HH HH HH HH HH. AAAA is the starting address for the first hexadecimal value, HH. Eight hexadecimal values are permitted per DATA string. Each string must begin with the address, and a space must separate the values.

If the characters in an HH field are "XX," the program will not load a value into the corresponding byte (skipping it). The characters "**" in an HH field, or "****" in an AAAA field, will end the load.

This data format (except "XX" and "**","****") is identical to the one used by the PET TIM monitor, so at a later time you can easily use the PET monitor to directly load this code from the DATA statements.

The DATA statements in this program contain the object code for the second command keyboard program described in the text. To start the machine program, enter SYS(826) and press RETURN. The PET tape I/O will not work while the machine code is running! Use SYS(863) to stop the machine code and make the tape I/O workable.

Input from the second keyboard follows the rules in Table 5 and as described in the text.

It is beyond the scope of this article to describe the details of the machine-language program. A source listing is provided in Example 8 for those who wish to puzzle it out.

A User Port Monitor Program

When you are attempting to interface to the user port, it is often necessary to write several small programs to set and display the VIA registers. The program in Example 9 performs these functions and will often

	Keyboard Pin	PET User Port
1	INT Key	—
2	RPT Key	—
3	No connection	CB2
4	No connection	—
5	GND	GND
6	+5 Volts (separate supply)	—
7	Strobe	CA1
8	Parity	PA7
9	Bit 4	PA3
10	Bit 3	PA2
11	Bit 1	PA0
12	Bit 7	PA6
13	Bit 2	PA1
14	Bit 6	PA5
15	Bit 5	PA4

Table 4. ASCII keyboard to PET user port wiring list. Your keyboard will, no doubt, have a different pin-out—just notice the data and handshake lines. If your keyboard requires an acknowledge, connect your ACK to CB2.

```
0340  A9 03             LDA #$03   !       VECTOR"
0342  8D 1A 02          STA $021A
      !
0345  A9 00             LDA #$00   ! SET UP USER PORT & MODE
0347  8D 43 E8          STA $E843  ! DATA DIRECTION REGISTER
034A  8D C7 03          STA MODE   ! MODE CELL
034D  AD 4C E8          LDA $E843  ! PERIPHERAL CONTROL REGISTER
0350  09 01             ORA #$01
0352  8D 4C E8          STA $E84C  ! PCR
0355  AD 4B E8          LDA $E84B  ! AUXILIARY CONTROL REGISTER
0358  09 01             ORA #$01
035A  8D 4B E8          STA $E84B  ! ACR
035D  58                CLI        ! ENABLE INTERRUPTS
035E  60                RTS        !  AND RETURN TO CALLER
      !
      ! *** RESTORATION CODE ***
      !         RESTORE THE "INTERRUPT VECTOR" SO THAT TAPE
      ! I/O CAN WORK PROPERLY.
      !
035F  78        XOFF    SEI        ! DISABLE INTERRUPTS
0360  A9 85             LDA #$85   ! SET UP OLD
0362  8D 19 02          STA $0219  !    "INTERRUPT
0365  A9 E6             LDA #$E6   !       VECTOR"
0367  8D 1A 02          STA $021A
036A  58                CLI        ! ENABLE INTERRUPTS
036B  60                RTS        !  AND RETURN TO CALLER
      ! *** STACK ADJUSTMENT ROUTINE ***
      !
036C  A9 00     STAX    LDA #$00   ! DUMMY PUSHES TO PET STACK FOR
036E  48                PHA        ! CORRECT OPERATION OF THE
036F  48                PHA        ! RESTORATION CODE
0370  48                PHA
0371  48                PHA
0372  4C 85 E6          JMP $E685  ! JUMP TO PET INTERRUPT HANDLER
                                   ! TO CONTINUE PROCESSING

      ! *** ACTIVE CODE ***
      !         CHECKS USER PORT IFR FOR CHARACTER. IF NOT
      ! PRESENT, RETURNS TO PET INTERRUPT PROCESSOR.
      !         IF PRESENT, TRANSLATES ACCORDING TO SCHEME
      ! AND PUTS INTO THE INPUT BUFFER.
      !
0375  AD 4D E8  PCODE   LDA $E84D  ! INTERRUPT FLAGS REGISTER
0378  29 02             AND #$02
037A  D0 07             BNE KEYS   ! DETECTED CHARACTER
037C  20 6C 03  FINISH  JSR STAX   ! SET UP TO CALL THE
037F  EA                NOP        ! PET RESTORATION CODE
0380  4C 7E E6          JMP $E67E  ! WHICH IS FROM HERE
      !
      !CHARACTER PROCESSING
      !
0383  AD 41 E8  KEYS    LDA $E841  ! ORA HANDSHAKE DATA REGISTER
0386  29 7F             AND #$7F   ! MASK OFF PARITY
0388  C9 1F             CMP #$1F
038A  10 30             BPL NCTR   ! IF POSITIVE, ISN'T A CONTROL CHAR
038C  C9 0A             CMP #$0A
038E  D0 07             BNE NLFD   ! CHAR ISN'T A LINEFEED
0390  A9 00             LDA #$00
0392  8D C7 03          STA MODE   ! SET MODE TO CHARACTERS
0395  F0 E5             BEQ FINISH ! BEQ SAVES A BYTE
      !
0397  C9 1B     NLFD    CMP #$1B   ! ESCAPE?
0399  D0 07             BNE CTRL   ! OTHER CTRL CHARS
039B  A9 80             LDA #$80   ! SET MODE TO
039D  8D C7 03          STA MODE   !     GRAPHICS
03A0  D0 DA             BNE FINISH ! SAVE ANOTHER BYTE
      !
      ! PROCESS CONTROL CHARS BY TABLE LOOKUP
      !
03A2  AA        CTRL    TAX
03A3  BD C8 03  LDA TABL, X
03A6  F0 D4     BEQ FINISH ! IGNORE IF TABLE RETURNS ZERO
03A8  EA        NOP
      !
      ! *** STASH CHARACTER INTO INPUT BUFFER ***
      !         NOTE THAT BUFFER POINTER MUST BE CHECKED &
      ! CORRECTLY ADJUSTED.
      !
03A9  AE 0D 02  STASH   LDX $020D  ! PET INPUT BUFFER POINTER
03AC  9D 0F 02          STA $020F,X  ! BASE OF INDEX IS START OF BUFFER
03AF  E8                INX
03B0  E0 0A             CPX #$0A   ! CHECK IF FULL
03B2  D0 02             BNE *+4    ! SHORT JUMP (SKIP ONE INSTR)
03B4  A2 00             LDX #$00
03B6  8E 0D 02          STX $020D  ! SAVE NEW POINTER
03B9  4C 7C 03          JMP FINISH ! I KNOW, I COULD HAVE SAVED A BYTE.
      !
03BC  C9 60     NCTRL   CMP #$60   ! CONVERT TO UPPER CASE
03BE  30 02             BMI NCASE
03C0  E9 20             SBC #$20
03C2  0D C7 03          ORA MODE   ! CONVERT TO GRAPHIC IF
03C5  D0 E2             BNE STASH  !  MODE > 0
      !
      ! *** DATA STORAGE AREA ***
      !
03C7  00        MODE    ! MODE BYTE = 0 IF CHARACTERS
                        !             128 IF GRAPHICS
03C8            TABL    ! CONTROL CHARACTERS CONVERSION TABLE
03C8                    .BYTE.  00,00,00,00,13,91,1D,00
03D0                    .BYTE.  00,12,00,00,00,0D,00,92
03D8                    .BYTE.  00,93,00,9D,00,14,00,00
03E0                    .BYTE.  11,94,00,00,00,00,00,00
      !
03E8  ! *** END OF CODE ***
```

Example 8. PET machine code program for a second command keyboard.

save some time and trouble.

Some comments concerning the code are in order:

Lines 70 to 90 hold the register names, which are similar to, and in the same order as, those in Fig. 2.

Line 210 puts a colon and some blanks at the end of each register name for display purposes.

Line 250 sets the Flags array to display the most commonly used registers when the program starts.

Notice the *three* blanks between the 4 and the 3 in line 310.

Line 320 moves the menu to a position that will not be overwritten when the program is displaying all 16 registers.

Cursor movements are used extensively to control the display. Be sure to count them carefully.

Lines 1000 to 1050 display a number in binary by moving a mask bit (variable Z1) to the right and printing the sign of the result (line 1030).

Subroutine 2000 is required to permit you to choose the time to access the Handshake Data register. The reason is that each access to this register will reset the Interrupt Flag bit. The D (DATA) command will read this register.

Subroutine 3000 lets you change the registers you want to see displayed. If you forget the names (I often do), enter a meaningless name, such as "XXX," and all the names will be shown.

Since the display is in binary, so is the input (see subroutine 4500).

Subroutine 4990 provides a "False Cursor," which is handy in many programs.

When the CB2 line is toggled, the original values of the PCR and ACR are saved, and after toggling, restored. CB2 is forced both high and low to guarantee a handshake pulse.

Using the User Port Monitor Program

After you have tried out the various commands and are familiar with them, attach the Blinkin' Lights to the user port and run the Monitor program. Close all of the Data Isolation switches and set the Data switches to low. If you are starting from a reset PET (you haven't changed any of the user port registers), the PET display will look like this:

```
         7 6 5 4 3 2 1 0
DDRA  :  0 0 0 0 0 0 0 0
ACR   :  0 0 0 0 0 0 0 0
PCR   :  0 0 0 0 1 1 0 0
IFR   :  0 1 1 0 0 0 0 0
DATA  :  0 0 0 0 0 0 0 0



D=DATA  P=POKE  S=SHOW
H=HELP  Q=QUIT  T=TOGGLE
```

The "1" bits are aspects of the registers used internally by the PET for its housekeeping functions. If you set the low

Character + CTRL	PET Function
Q	Clear Screen
D	Home Cursor
E	Cursor Up
S	Cursor Left
F	Cursor Right
X	Cursor Down
Y	INST
U	DEL
I	RVS on
O	RVS off

Table 5. Control characters for PET special keys.

```
10  REM CB2 BLINKER
20  POKE 59467, PEEK(59467) AND 227
30  POKE 59468,(PEEK(59468) AND 31) OR 192
40  FOR J = 1 TO 300 : NEXT
50  POKE 59468, PEEK(59468) OR 244
60  FOR J = 1 TO 300: NEXT
70  GOTO 30
```

Example 10. CB2 Blinker program. The CB2 LED in the Blinkin' Lights will blink at about 1Hz.

four bits on the Blinkin' Lights Data switches to high, the DATA: line will become 0 0 0 0 1 1 1 1. As you change the switch settings, you will notice that there is a lag of about one second before the display responds.

This illustrates how the Monitor program can show the data you input to the user port. Now disconnect the Data switches by opening the Data Isolation switches—the DATA: will now become all ones.

With the P command, change the DDRA to 1 1 1 1 1 1 1 1. The DATA: is now 0 0 0 0 0 0 0 0. This is the initial value stored in the PET. Using P again, change the DATA register to some other value and watch it appear on the LEDs on the Blinkin' Lights. This illustrates data output.

If you close the Data Isolation switches and change these registers with the P command, you can demonstrate input via handshake with the CA1 line:

DDRA	set to	0000 0000	
PCR	set to	0000 1100	(Negative transition)
ACR	set to	0000 0001	(Enable latching)

When you return to the display, the IFR may look like: 0110 0010. If it does, press D and then press any key. The IFR will now return to: 0110 0000, indicating that the Flag bit was reset when the Data with Handshake was read.

Set the Blinkin' Lights Data switches to some value and watch the DATA: on the display. The value will follow the switch settings. Now, flick the CA1 toggle switch (be sure the isolation switch is closed), and the IFR will show bit 1 as set. If you now change the Data switches, the DATA: value will not change. It will remain latched until you do the D command. This illustrates input with latching and handshaking.

Feel free to experiment with other settings for the user port with the Monitor program.

The CB2 Line

The CB2 line is the most complex of the user port lines. It can be operated in a variety of modes, including the provision of an output handshake and the serial transfer of data. As most of the CB2 modes can only be controlled from machine language, this article will cover only the two modes that are usable from BASIC.

CB2 as an Output or Handshake

The CB2 line may be turned off or on directly to provide either a handshake line or a 9th output bit for the user port. In either case, the shift register modes must be disabled by setting the Auxiliary Control register (ACR) as follows:

```
POKE 59467, PEEK(59467) AND 227
```

(In most cases the ACR is already zero, so this may be ignored. However, safety first!)

```
10 REM 6522 VIA DISPLAY AND MONITOR
20 REM PROGRAM
30 REM BY: GREGORY YOB, 1978
40 REM SET UP R$= REGISTER NAMES,
50 REM A()=REGISTER ADDRESSES,
60 REM F()=SHOW REGISTER IF>0
70 DATA "ORB","ORA","DDRB","DDRA"
80 DATA "T1LC-L","T1C-H","T1L-L","T1L-H"
90 DATA "T2LC-L","T2C-H","SR","ACR"
100 DATA"PCR","IFR","IER","DATA"
110 REM 'DATA' IS ORA WITHOUT HANDSHAKE
120 DIM R$(16),A(16),F(16)
200 A=59456: FOR J=1 TO 16
210 READ A$:R$(J)=LEFT$(A$+"bbbbbbbb",6)+";"
220 A(J)=A:A=A+1
230 NEXT J
240 REM SET FLAGS FOR INITIAL DISPLAY
250 F(4)=1:F(12)=1:F(13)=1:F(14)=1:F(16)=1
300 REM SET UP DISPLAY
310 PRINT"Ⓒ bbbbbbbb7bb6bb5bb4bbb3bb2bb1bb0"
320 PRINT"ⒹⒹⒹⒹⒹⒹⒹⒹⒹⒹⒹⒹⒹⒹⒹⒹⒹⒹ";
330 PRINT"D=DATA  P=POKE  S=SHOW"
340 PRINT"H=HELP  Q=QUIT  T=TOGGLE"
400 REM DISPLAY LOOP
410 PRINT"Ⓗ Ⓓ Ⓓ ";
420 FOR J=1 TO 16
430 IF F(J)=0 THEN 450
440 Z=PEEK(A(J)):PRINTR$(J);:GOSUB1000
450 NEXT J
460 REM IF NO INPUT DO LOOP AGAIN
470 GETA$:IFA$=""THEN 410            ( "" is a null string)
500 REM DO COMMANDS
510 IF A$="D" THEN GOSUB 2000
520 IF A$="P" THEN GOSUB 2500
530 IF A$="S" THEN GOSUB 3500
540 IF A$="H" THEN GOSUB 3000
550 IF A$="T" THEN GOSUB 5500
560 IF A$="Q" THEN END
700 GOTO 310
1000 REM DISPLAY IN BINARY
1010 Z1=128
1020 FOR Z2=1 TO 8
1030 PRINT SGN(Z AND Z1);
1040 IF Z2=4 THEN PRINT "b";
1050 Z1=Z1/2 : NEXT Z2: PRINT : RETURN
2000 REM DISPLAY HANDSHAKE REGISTER
2010 Z = PEEK(59457) :PRINT"Ⓓ"R$(2); GOSUB 1000
2020 PRINT"Ⓓ"; : GOSUB 4990:RETURN
2500 PRINT"Ⓒ POKE REGISTERⒽⒹⒹ"
2510 GOSUB 4000
2520 GOSUB 4500
2530 POKE A(Z),B
2540 RETURN
3000 PRINT"Ⓒ bb 6522 REGISTER DISPLAY AND CHANGE ⒹⒹ
3010 PRINT"THIS SHOWS THE VALUES FOR THE PET'S
3020 PRINT"VIA REGISTERS. YOU CAN LOOK AT ALL OF
3030 PRINT"THEM. THOSE USED FOR THE USER
3040 PRINT"PORT ARE SHOWN WHEN THE PROGRAM
3050 PRINT"STARTS. Ⓓ ⓁⓁⓁ THE DISPLAY IS REFRESHED ABOUT ONCE
3060 PRINT"PER SECOND. PRESS A KEY TO DO A COMMAND
3070 PRINT"Ⓓ bbbD=DATA  READS ORA WITH HANDSHAKE
3080 PRINT"   P=POKE  LETS YOU POKE A REGISTER
3090 PRINT"   S=SHOW  SELECTS REGISTERS TO DISPLAY
3100 PRINT"   Q=QUIT  STOPS PROGRAM
3110 PRINT"   T=TOGGLE  TURNS CB2 ON, THEN OFF TO
3120 PRINT"             FORCE HANDSHAKE & THEN
3130 PRINT"             RESTORES TO PRIOR STATE
3300 PRINT"ⒹⒹ";:GOSUB4990:RETURN
3500 REM CHANGE DISPLAYED REGISTERS
3510 PRINT"Ⓒ SHOW REGISTERSⒹⒹⒹ
3520 GOSUB 4000
3530 PRINT"S=SHOW,E=ERASE,X=FINISHED";:GOSUB 5000
3540 IF A$="S" THEN F(Z)=1
3550 IF A$="E" THEN F(Z)=0
3560 IF A$="X" THEN RETURN
3570 PRINT"ⒽⒹⒹⒹ";
3580 GOTO 3520
4000 REM GET REGISTER NAME, RETURN Z=INDEX
4010 PRINT"ⒹⒹ REGISTER NAME:bbbbbbbbbbbbⓁⓁⓁⓁⓁⓁⓁⓁⓁⓁⓁⓁ";
     :INPUT A$
4020 RESTORE:FORZ=1TO16:READB$
4030 IFB$=A$THEN RETURN
4040 NEXTZ:PRINT"ⒹⒹⒹ THE REGISTERS ARE CALLED:
4050 FOR J=1TO16:PRINT LEFT$(R$(J),6)"bbbb";:NEXT J
4060 PRINT"ⓊⓊⓊⓊⓊⓊⓊⓊⓊⓊⓊ";: GOTO 4010
4500 REM - GET BINARY NUMBER
4510 PRINT"BINARY VALUE: ";:INPUT A$:Z1=128:B=0
4520 IF LEN(A$) < 8 THEN PRINT "Ⓤ";: GOTO 4510
4530 FOR J=1TO8
4540 IF MID$(A$,J,1)="1"THEN B=B OR Z1
4550 Z1=Z1/2:NEXT J
4560 RETURN
4990 PRINT"PRESS A KEY";
5000 GET A$: PRINT"[&]Ⓛ";:FOR K=1 TO 20: NEXT K
5010 PRINT"bⓁ";:FOR K = 1 TO 20: NEXT K
5020 IF A$=""THEN 5000               ( "" is a null string)
5030 RETURN
5500 REM TOGGLE CB2
5510 A=PEEK(59467):B=PEEK(59468)
5520 C=B AND 131 OR 192
5530 D= B OR 224
5540 POKE 59468,C
5550 POKE 59468,D
5560 POKE 59468,B
5570 POKE 59467,A
5580 RETURN
```

Example 9. PET user port display and monitor program.

Then, the CB2 line is set high by:

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

and it is set low by:

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

The parentheses are required to ensure that the operations AND and OR are done correctly. Example 10 is a short "CB2 Blinker" that blinks CB2 at about 1 Hz.

Interfacing the Writehander

The Writehander is a one-handed input keyboard manufactured by the NewO Company, 246 Walter Hays Drive, Palo Alto CA 94303 (see *Kilobaud* No. 23, p. 9, for a description of the Writehander).

The Writehander is a gray plastic ball about six inches across with switches placed so that the fingers and thumb may touch them. By altering the finger arrangements, you can send any of the 128 ASCII codes to the computer. When the byte is ready, the Writehander provides a strobe and then requires an acknowledge signal before it sends the next byte.

The wiring to the PET user port is shown in Table 6. The grounds were connected together for the power supply, the PET and the Writehander. The Writehander has several jumper options that were wired as:

1) Strobe goes active low   + to − ⅂_
2) Acknowledge active low   + to − ⅂_
3) Parity (Bit 8) set low   Gnd

This means that the following steps are required to talk with the Writehander.

1. Poke the DDR to all inputs
2. Set CA1 to detect the Hi to Low transition
3. Disable the CB2 Shift Register mode
4. Enable latching with CA1
5. Turn CB2 on (high)
6. Wait for the interrupt flag in the IFR
7. Read the Data with Handshake
8. Mask off the parity bit and display the data (or whatever)
9. Turn CB2 off (low)
10. Go to step 5

These steps were incorporated into a program, Example 11, which was only intended to accept characters from the Writehander and display their values on the PET screen. See the program in Example 7 for a more complete processing of the characters. (If you are a real diehard, modify the assembly program in Example 8 to provide the required CB2 logic.)

Lines 30 and 40 can be combined, but this program keeps them separate to show the different things being done. If you want to show the character rather than the value, use:

```
90  PRINT CHR$( X AND 127 );
```

I encountered several frustrating experiences during the development of the above (simple!) program:

1. The Writehander would work perfectly when attached to the Blinkin' Lights by itself, and the program would work perfectly when it was attached to the Blinkin' Lights . . . and (guess), when the Writehander was attached to the PET, it *wouldn't* work! After much fiddling, I discovered that the Writehander required that the ACK (CB2) be *high* before it would bring the Strobe (CA1) low. Thus CB2 had to be set high before trying to look for a character.

2. The parenthesis around the PEEK in line 80 is required for the CB2 to be set low due to the precedence relations of AND and OR.

3. PET ASCII isn't ASCII, so the "wrong" character would be displayed (see A Keyboard Via the User Port section for a detailed discussion).

```
5   PRINT" © ";
10  POKE 59459,0
20  POKE 59468, PEEK(59468) AND 254
30  POKE 59467, PEEK(59467) AND 227
40  POKE 59467, PEEK(59467) OR 1
50  POKE 59468, PEEK(59468) OR 224
60  IF (PEEK(59469) AND 2)=0 THEN 60
70  X=PEEK(59457)
80  POKE 59468, (PEEK(59468)AND 31) OR 192
90  PRINT X AND 127;
100 GOTO 50
```

Example 11. Writehander input program.

CB2 as a Shift Register

The CB2 line may be made to act as a shift register by setting a combination of bits 2, 3 and 4 in the Auxiliary Control register (ACR). Only one of these modes is usable from BASIC. The others require the use of machine language to be controlled properly (see the 6522 VIA specification for details).

One nice way to experiment with this is to use the PET to make "square wave music." Fig. 4 shows two ways to attach an audio extension to the PET. Each of these simply uses the CB2 line for the audio signal.

Checking It Out

Once you have your audio extension together, one way to check it out is to toggle CB2 in Handshake mode as fast as BASIC will go:

```
10  POKE 59467,PEEK(59467)AND 227
20  A=59468: X=PEEK(A)AND 131 OR 192
30  Y=PEEK(A) OR 224
```

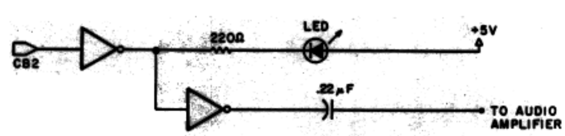

Fig. 4a. Add the inverter and capacitor to the output of the CB2 inverter in the Blinkin' Lights. Fig. 2 has this addition indicated.

Line	Color	Function	PET
1	Brown	Bit 1	PA0
2	Red	+7 to +23 V power (unused)	
3	Orange	Bit 2	PA1
4	Yellow	Ground	GND
5	Green	Bit 3	PA2
6	Blue	+5 V (separate power supply)	
7	Violet	Bit 4	PA3
8	Gray	—	
9	White	Bit 5	PA4
10	Black		
11	Brown	Bit 6	PA5
12	Red		
13	Orange	Bit 7	PA6
14	Yellow	Strobe	CA1
15	Green	Bit 8	PA7
16	Blue	Acknowledge(ACK)	CB2

Table 6. Writehander wiring list.

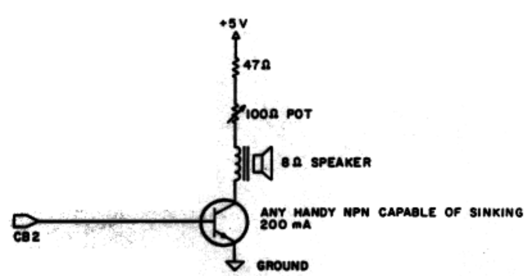

Fig. 4b. This circuit lets you add sound effects, etc., for you PET without any additional equipment. Take the +5 volts from the second tape port. (That's the top or bottom pin, second in from the side of the PET. Check your first tape recorder to find whether it is on top or bottom—Commodore makes both kinds!) Find a 2 or 3 inch speaker and any handy NPN transistor capable of 200 mA current. The 47 Ohm resistor should be 1/2 Watt or larger and should not be omitted. My unit was put on a 3×5 inch perfboard with connectors glued to one edge, which makes it easy to hook to my PET.

Data Directions Register	
POKE 59459, 255	Set user port to 8 bits output.
POKE 59459, 0	Set user port to 8 bits input.
Simple Input and Output (no handshakes)	
(value) = PEEK(59471)	Input (value) from user port.
POKE 59471, (value)	Output (value) to user port.
Input and Output with Handshaking	
POKE 59468, PEEK(59468) AND 254	CA1 will trigger on falling edge.
POKE 59468, PEEK(59468) OR 1	CA1 will trigger on rising edge.
POKE 59467, PEEK(59467) OR 1	Data is latched when CA1 triggers.
POKE 59467, PEEK(59467) AND 254	Data is not latched.
IF PEEK(59469) AND 2 THEN —	Three ways of detecting the CA1 Flag Bit.
WAIT 59469, 2	Be careful with using WAIT.
nnn IF(PEEK(59469) AND 2) = 0 THEN nnn	
(value) = PEEK(59457)	Reads from user port, resets CA1 flag bit.
POKE 59457, (value)	Writes to user port, resets CA1 flag bit.
POKE 59468, PEEK(59468) OR 224	Set CB2 line high.
POKE 59468, (PEEK(59468) AND 31) OR 192	Set CB2 line low.
Shift Registery	
POKE 59467, PEEK(59467) AND 227 OR 16	Sets shift register to free running mode.
POKE 59467, PEEK(59467) AND 227	Disables shift register modes.
POKE 59466, (value)	Puts (value) into shift register.
POKE 59464, (value)	Sets timer 2 to (value)
Miscellany	
(value) = PEEK(515)	Reads matrix value of key pressed. 255 = no keys pressed.
(value) = PEEK(516)	Reads shift keys. 1 if pressed, 0 otherwise.

Table 7. Summary of BASIC statements used to control the PET user port.

```
40 POKE A,X:POKEA,Y: GOTO 40
```

Line 10 disables the Shift Register mode, and line 40 turns CB2 on and off. The reason that variables are used in line 40 for the addresses is that BASIC runs much faster when variables are substituted for constants.

RUN the program, and a buzz will emerge from your speaker.

Try changing line 40 to:

```
40 POKE59468,X:POKE59468,Y:GOTO 40
```

and you will notice that the pitch of the buzz is much lower. (Note: You will also hear a variation in the pitch of the buzz. This is caused by the PET's interrupt routines "beating" with the execution of the BASIC program.)

A last variation before going on to the shift register is to change the above program as follows:

```
40 Z=515
50 POKE A,X:FOR J=1 TO PEEK(Z):
   NEXT: POKE A,Y: GOTO 50
```

Pressing different keys will vary the rate of clicking. (Note: Location 515 indicates which key is depressed on the PET keyboard. This is not in PET ASCII but represents the matrix position of the key.)

Shift Register Mode

When the ACR bits 4, 3 and 2 are "100" the shift register is in "free running mode." Two addresses are now of interest:

SR	Shift Register	59466
T2L-W	Timer-2	59464

At a rate determined by the contents of Timer-2, the contents of the shift register are placed on the CB2 line. When eight bits have been shifted out, the shift register is again shifted out. This creates a continuous stream of bits that repeats every eight Timer-2 cycles.

Timer-2 accepts a number from 0 to 225 and counts it down to zero at the PET clock rate. When it reaches zero, the shift register is shifted and the least significant bit (bit 0) is placed on the CB2 line.

By placing an appropriate number into Timer-2 for the pitch and a 15 into the shift register, square waves at audio frequency will emerge from CB2. Here is the world's clumsiest musical instrument (see Example 12). Try it and you will know why. Line 50 inputs a waveform to be put into the shift register when a key is pressed. Line 60 guarantees that the waveform will result in a sound (a 0 or a 255 will come out as a dc voltage).

Line 90 detects the state of the PET keyboard matrix. When no key is depressed, the value in this address is 255. Line 100 puts a zero into the shift register, turning the sound "off." Then the keyboard is checked again.

If a key is depressed, the "pitch," or the matrix value of the key, is put into the timer and the timbre is put into the shift register. Now a sound is heard (for most of the keys; some will

```
10 REM CLUMSY MUSIC MACHINE
20 REM SET S.R. MODE IN ACR
30 POKE 59467, PEEK(59467) AND 227 OR 16
40 PRINT"TIMBRE :";
50 INPUT TC
60 IF TC<1 OR TC>254 THEN 40
70 REM CHECK FOR KEYPRESSES
80 PRINT"PRESS KEYS FOR TONES"
90 K=PEEK(515)
100 IF K=255 THEN POKE 59466,0: GOTO 90
110 POKE 59464,K: POKE 59466,TC
120 K=PEEK(515): IF K=255 THEN 100
130 GOTO 120
```

Example 12. A clumsy music machine.

```
10 POKE 59467,PEEK(59467)AND 227 OR 16
20 POKE 59466,15
30 FOR J=0 TO 255; POKE 59464,J: NEXT
100 GET A$: IF A$="" THEN 30
110 POKE 59466,0
```

Example 13. Program for effect 1.

```
30 FOR J=10 TO 255 STEP 10: POKE 59464,J: NEXT
40 FOR J=255 TO 10 STEP -10: POKE 59464,J: NEXT
```

Example 14. Changes in Example 13 for effect 2.

```
30 FOR J=1 TO 100: POKE 59464, 240*RND(1)+10: NEXT
```

Example 15. Change in Example 13 for effect 3.

```
30 FOR J=1 TO 30: POKE 59464,100: POKE 59464,200: NEXT
40 FOR J=1 TO 30: POKE 59464,150: POKE 59464,250: NEXT
```

Example 16. Changes in Example 13 for effect 4.

make inaudibly high notes). Line 120 waits until the key is released before starting over at line 100.

Some time spent with a calculator or scope will yield about two octaves of pitches that are reasonably close to the musical scale(s). Feel free to write your own musical programs.

Since the CB2 line, once in Shift Register mode, will run independently of the PET's other activities, other computations may be done while a tone is sounded. Another aspect is the making of sound effects for games. See Examples 13-17 and try them out to find out what they do.

Lines 100 and 110 in Example 13 provide a way of turning the sound off. If you don't do this, the PET will squeak at you after you press the STOP key—and only a direct version of line 110 will turn the squeak off! Examples 14-16 show changes to Example 13.

```
10  REM BETTER WOLF
20  REM GREGORY YOB
30  REM CB2 ON USER PORT & AMP
100 POKE 59467,16 :POKE 59466,15
110 FOR L=180 TO 50 STEP -3:POKE 59464,L:NEXT
111 FOR J=1 TO 6:NEXT
112 POKE 59466,0
115 FOR J=1 TO 150: NEXT
117 POKE 59466,15
120 FOR L=150 TO 80 STEP -2: POKE 59464,L:NEXT
130 FOR L=90 TO 190: POKE 59464,L
132 FOR J=1 TO L/70: NEXT
134 NEXT
135 POKE 59467,0
140 PRINT"PRESS KEY TO DO IT AGAIN"
150 GET A$: IF A$="" THEN 150
160 GOTO 100
```

Example 17.

Summing Up

The PET user port is a versatile way with which to communicate between the PET and the rest of the world. This article has shown you the "nuts and bolts" required to interface many devices, including joysticks, keyboards and music makers, that add to the capabilities or your PET.

For your convenience, Table 7 summarizes the various BASIC statements used to control the user port. Now let me see . . . robots, turtles, printers, my lawn sprinklers. . . .■