Newsgroups: comp.sys.cbm
From: rhialto@mbfys.kun.nl (Olaf Seibert)
Subject: PET RAM memory map
Organization: University of Nijmegen, The Netherlands
Date: Wed, 14 Dec 1994 19:22:02 GMT

This could be improved upon - not all locations have labels. I must note I've never seen a PET memory map with labels, so in that respect this is a first. I could add the locations for the Basic 1 ROMs in a next version. Corrections welcome. Enjoy.

```
;    Commodore PET RAM Memory Map
;    for Basic 2 and 4 (40 and 80 columns)
;    V1.1 13 dec 1994
;
;    Note that the labels in this list are matched up against C-64
;    memory maps. All attempts were made to secure plausibility of
;    placement.
;    Locations marked with (64#..) are found by taking the label given
;    for the C-64 and comparing ROM disassemblies. Many of these
;    locations have RS-232 specific names.


 LABEL    HEX       DECIMAL
     ADDRESS        LOCATION  DESCRIPTION

 USRPOK   0000      0    USR Function Jump Instr (4C)
 USRADD   0001-0002 1-2       USR Address Low     Byte / High Byte
 CHARAC   0003      3    Search Character
 ENDCHR   0004      4    Flag: Scan for Quote at End of String
 COUNT    0005      5    Input Buffer Pointer / No. of Subscripts
 DIMFLG   0006      6    Flag: Default Array DiMension / array name
                    initial / AND, OR flag
 VAUYP    0007      7    Data Type: $FF = String, $00 = Numeric
 INTFLG   0008      8    Data Type: $80 = Integer, $00 = Floating
 GARBFL   0009      9    Flag: DATA scan/LlST quote/Garbage Coll
 SUBFLG   000A      10   Flag: Subscript Ref / User Function
```

```
Call
 INPFLG   000B      11   Flag: $00 = INPUT, $40 = GET, $98 =
READ
 TANSGN   000C      12   Flag TAN sign / Comparison Result
     000D      13   3: Flag to suppress PRINT or PRINT# when -ve
     000E      14   3: File# of current I/O device (as 0010)
     000F      15   3: terminal width (unused-carried over from
TTY)
     0010      16   3: width of source (unused - from TTY)
        000D-000F   13-15     4: Disk Status DS$ descriptor
     0010      16   4: File# of current I/O device (when non-
zero
                    suppresses INPUT prompt etc)
 LINNUM   0011-0012 17-18     Temp: Integer Value
 TEMPPT   0013      18   Pointer Temporary String
 LASTPT   0014-0015 19-21     Last Temp String Address
 TEMPST   0016-001E      22-30     Stack for Temporary Strings
 INDEX    001F-0022 31-34     Utility Pointer Area
 RESHO    0023-0027 35-39     Floating-Point Product of Multiply
 TXTTAB   0028-0029 40-41     Pointer: Start of BASIC  Text
 VARTAB   002A-002B 42-43     Pointer: Start of BASIC  Variables
 ARYTAB   002C-002D 44-45     Pointer: Start of BASIC Arrays
 STREND   002E-002F 46-47     Pointer End of BASIC Arrays (+1)
 FRETOP   0030-0031 48-49     Pointer: Bottom of String Storage
 FRESPC   0032-0033 50-51     Utility String Pointer
 MEMSIZ   0034-0035 52-53     Pointer: Highest Address Used by
BASIC
 CURLIN   0036-0037 54-55     Current BASIC Line Number
 OLDLIN   0038-0039 56-67     Previous BASIC Line Number
 OLDTXT   003A-003B 58-59     Pointer: BASIC Statement for CONT
 DATLIN   003C-003D 60-61     Current DATA Line Number
 DATPTR   003E-003F 62-63     Pointer: Current DATA Item Address
 INPPTR   0040-0041 64-65     Vector: INPUT Routine
 VARNAM   0042-0043 66-67     Current BASIC Variable Name
 VARPNT   0044-0045 68-69     Pointer: Current BASIC Variable
Data
 FORPNT   0046-0047 70-71     Pointer: Index Variable for FOR/
NEXT
     0048-0049 72-73     Y-save; op-save; BASIC pointer save
     004A      74   Comparison symbol accumulator: bits 0,1,2
                    are <, =, >.
     004B-004C 75-76     Pointer to temp storage in RAM for FN
DEF,
                    TAN, &c
     004D-0050 77-80     Pointer to string, length and garbage
                    collect constant
```

```
 TEMPF1   0054-0058      Temporary storage for FLPT value.
 TEMPF2   0059-005D      Temporary storage for FLPT value.

 FACEXP   005E      94   Floating-Point Accumulator #1: Exponent
 FACHO    005F-0062 95-98     Floating Accum.     #1: Mantissa
 FACSGN   0063      99   Floating Accum.     #1: Sign
 SGNFLG   0064      100  Pointer: Series Evaluation Constant
Pointer
 BITS     0065      101  Floating -accum. #1: Overflow Digit
 ARGEXP   0066      102  Floating-Point Accumulator #2: Exponent
 ARGHO    0067-006A 103-106   Floating Accum. #2: Mantissa
 ARGSGN   006B      107  Floating Accum. #2: Sign
 ARISGN   006C      108  Sign Comparison Result: Accum. # 1 vs
#2
 FACOV    006D      109  Floating Accum. #1. Low-Order
(Rounding)
 FBUFPT   006E-006F 110-111   Pointer: Cassette Buffer
 CHRGET   0070-0087 112-135   Subroutine: Get Next Byte of BASIC
Text
 CHRGOT   0076      118  Entry to Get Same Byte of Text Again
 TXTPTR   0077-0078 119-120   Pointer: Current Byte of BASIC
Text
     0070           INC $77
                    BNE $0076
     0076           LDA $xxxx
                    CMP #$3A
                    BCS $0087
                    CMP #$20
                    BEQ $0070
                    SEC
                    SBC #$30
                    SEC
                    SBC #$D0
     0087           RTS
 RNDX     0088-008C 136-140   Floating RND Function Seed Value
 TIME     008D-008F 141-143   Real-Time Jiffy Clock (approx)
1/60 Sec
 CINV     0090-0091 144-145   Vector: Hardware Interrupt
 CBINV    0092-0093 146-147   Vector: BRK Instr. Interrupt
 NMINV    0094-0095 148-149   Vector: Non-Maskable Interrupt
 STATUS   0096      150  Kernal I/O Status Word: ST
 LSTX     0097      151  Current Key Pressed: 255 = No Key
 SFDX     0098           Flag: Print Shifted Chars.
        0099-009A              Jiffy clock correction: 623rd
1/60 sec
                           does not increment time
 STKEY    009B      155  Flag: STOP key / RVS key
```

```
 SVXT     009C      156  Timing Constant for Tape
 VERCK    009D      157  Flag: 0 = Load, 1 = Verify (Kernel)
 NDX 009E      158  No. of Chars. in Keyboard Buffer (Queue)
 RVS 009F      159  Flag: Print Reverse Chars. -1=Yes, 0=No Used
 C3PO     00A0      160  Flag: Serial Bus-Output Char. Buffered
 INDX     00A1      161  Pointer: End of Logical  Line for INPUT
        00A2      162  Not Used
 LXSP     00A3-00A4 163-164   Cursor Y-X Pos. at Start of INPUT
 BSOUR    00A5      165  Buffered Character for IEEE Bus
     00A6      166  Key Image
 BLNSW    00A7      167  Cursor Blink enable: 0 = Flash Cursor
 BLNCT    00A8      168  Timer: Countdown to Toggle Cursor
 GDBLN    00A9      169  Character Under Cursor
 BLNON    00AA      170  Flag: Last Cursor Blink  On/Off
?SYNO     00AB      171  Cassette Sync No. (64#0096)
?NXTBIT   00AB      171  Tape EOT Flag: EOT received from tape
 CRSW     00AC      172  Flag: INPUT or GET from  Keyboard
     00AD      173  X save in tape handling (saves cassette #)
 LDTND    00AE      174  No. of Open Files / Index to File Table
 DFLTN    00AF      175  Default Input Device (0)
 DFLTO    00B0      176  Default Output (CMD) Device (3)
 PRTY     00B1      177  Tape Character Parity
 DPSW     00B2      178  Flag: Tape Byte-Received
 PSW=DPSW
     00B3      179  Temporary save eg. logical address or DOS
                    wedge

     00B7      183  Temp Data Area (64#00A3)

     00B9      185  Temp Data Area (64#00A4)

 BUFPNT   00BB           Pointer: Tape I/O Buffer #1
          00BC           Pointer: Tape I/O Buffer #2
 INBIT    00BD      189  Cassette Temp (64#00A7)
 BITCI    00BE      190  Cassette Temp (64#00A8)
 RINONE   00BF      191  RS-232 Flag: Check for Start Bit
(64#00A9)
 FNMIDX 00C0          192     Index to Cassette File name/
Header ID for
                             Tape write.
 PTR1     00C0      192  Tape Pass 1 Error Log
 PTR2     00C1      193  Tape Pass 2 Error Log
 RIDATA   00C2      194  Cassette Temp (64#00AA) read flags:
0=scan,
                    1-15=count, $40=load, $80=end of tape marker
 RIPRTY   00C3      195  Cassette Short Cnt (64#00AB): counter
of seconds
```

```
                    before tape write / checksum
 PNT 00C4-00C5 196-197   Pointer: Current Screen  Line Address
 PNTR     00C6      198  Cursor Column on Current Line
 SAL 00C7-00C8 199-200   Pointer: Tape Buffer/ Screen Scrolling
 EAL 00C9-00CA 201-202   Tape End Addresses/End of Program
 CMP0     00CB-00CC 203-204   Tape Timing Constants
 QTSW     00CD      205  Flag: Editor in Quote Mode, $00 = NO
 BITTS    00CE      206  Cassette Temp (64#00B4): Tape read
timer flag
                    =IRQ enabled for Timer 1
     00CF      207  End of tape read
     00D0      208  Read character error
 FNLEN    00D1      209  Length of Current File Name
 LA  00D2      210  Current Logical File Number
 SA     00D3           211     Current Secondary Address
 FA  00D4      212  Current Device Number
 LNMX     00D5      213  Physical Screen Line Length
     00D5           213     4.80: right side of window
 TAPE1    00D6-00D7 214-215   Pointer: Start of Tape Buffer
 TBLX     00D8      216  Current Cursor Physical Line Number
 DATAX  00D9           217     Current Character to Print
 FNADR    00DA-00DB 218-219   Pointer: Current File Name
 INSRT    00DC      220  Flag: Insert Mode, >0 = # INSTs
?ROPRTY   00DD      121  Cassette Temp
 FSBLK    00DE      222  Cassette Read / Write Block Count
 MYCH     00DF      223  Serial Word Buffer
 LDTB1    00E0-00F8 224-248   3+4.40: Screen Line Link Table /
Editor Temps
 SCTOP  00E0           224     4.80: first line of window
 SCBOT  00E1           225     4.80: last line of window
 SCLF   00E2           226     4.80: first column of window
 XMAX     00E3      227  4.80: Size of Keyboard Buffer
 XMAX     03EB      1003 4.40
 RPTFLG   00E4      228  4.80: Flag: REPEAT Key Used, $80 =
Repeat
                                    $40 = disable
 RPTFLG   03EE      1006 4.40
 KOUNT    00E5      651  4.80: Repeat Speed Counter
 KOUNT    03EA      1002 4.40
 DELAY    00E6         230    4.80: Repeat Delay Counter
 DELAY    03E9      1001 4.40
     00E7      231  4.80: Chime Time
     03EC      1004 4.40: Chime Time
     00E8      232  4.80: Home Count
     00E9-00EA       233-234  4.80: input from screen vector
(from E006)
     00EB-00EC       235-236  4.80: print to screen vector (from
```

```
E009)
     00ED-00F7 237-247   4.80: not used
     00F8      248  4.80: Counter to speed TI by 6/5
     03ED      1005 4.40: Counter to speed TI by 6/5
 CAS1     00F9      249  Tape Motor Interlock #1
 CAS2     00FA      250  Tape Motor Interlock #2
 STAL     00FB-00FC 251-252   I/O Start Address
 MEMUSS   00FD-00FE 253-254   Tape Load Temps
     00FF      255  Not used
     0100-01FF 256-511   Micro-Processor System Stack Area
     0100-010A 256-266   Floating to String Work Area
 BAD 0100-013E 256-318   Tape Input Error Log
 BUF 0200-0250 512-592   System INPUT Buffer
 LAT 0251-025A 593-602   KERNAL Table: Active Logical File No's.
 FAT 025B-0264 603-612   KERNAL Table: Device No. for Each File
 SAT 0265-0270 613-623   KERNAL Table: Second Address Each File
 KEYD     0270-027A 624-633   Keyboard Buffer Queue (FIFO)
 TBUFFR   027A-0329 634-825     Tape I/O Buffer #1
     027A             Type of tape file:
                      1=program header for SAVE "",1,0
                      2=data block
                      3=absolute load SAVE "",1,1 (VIC-20 and
later)
                      4=data file header
                      5=end of tape block: SAVE "",1,2
     027B-027C        Start address for load
     027D-027E        End address for load
     027F-028E        File name
 TBUFFR   033A-03F9 826-1017  Tape I/O Buffer #2
     033A             4: DOS byte parameter in RECORD / char ptr
     033B             4: DOS drive 1 number
     033C             4: DOS drive 2 number
     033D             4: DOS length / write flag
     033E             4: DOS 8-bit syntax checking flag
     033F-0340        4: DOS diskette ID
     0341             4: Length of DOS command string
     0342-0352        4: Buffer for filename
     0353-0380        4: Full DOS command string buffer
     03EE-037F        4.80: Table of 80 bits to set TABs
 DELAY    03E9      1001 4.40
 KOUNT    03EA      1002 4.40
 XMAX     03EB      1003 4.40
     03EC      1004 4.40: Chime Time
     03ED      1005 4.40: Counter to speed TI by 6/5
 RPTFLG   03EE      1006 4.40
     03F0-03F9        4.40: Table of 80 bits to set TABs
     03FA-03FF 1018-1023 Unused
```

```
 TIMOUT   03FC      1020 4: Flag: Kernal Variable for IEEE
Timeout
        0400-8000   1024-32767      Basic program area
     0400           0 byte at start of Basic program
     0401-0402      first link to next Basic line
     0403-0404      first line number
     0405-               tokenized basic line, terminated with
00
                    followed by next link

     8000-83E7 32768-33767 40 column screen memory
     8000-87EF 32768-34767 80 column screen memory

A000-AFFF free space for 4K EPROM
B000-BFFF 3: free space for 4K EPROM
    4         3
B000-DFFF C000-DFFF Basic keywords and operators, and general
processing
E000-EFFF E000-EFFF Mostly screen editor functions
F000-FFFF F000-FFFF Kernel: tape processing, IEEE-488, jump
table.


-Olaf.
--
___ Olaf 'Rhialto' Seibert    rhialto@mbfys.kun.nl
What's the use of
\X/  racism if you can't even see if a person belongs to your
abhorred kind?

rom: rhialto@mbfys.kun.nl (Olaf Seibert)
Subject: PIA and VIA info (PET)
Organization: University of Nijmegen, The Netherlands
Date: Tue, 13 Dec 1994 11:13:11 GMT


PIA 6520 and VIA 6522
---------------------

Summarised by Olaf Seibert, from "Programming the PET/CBM" (by
Raeto Collin
West) chapter 14 and The Transactor, Volume 4 Issue 05.

In the PET, you find the two PIAs at E810 and E820, and the VIA
at E840.

The PIA
```

The PIA has two 8-bit I/O ports, A and B, which are mostly identical in function. All 8 bits can be set to input (1 in the DDR) or output (0 in the DDR) independently. Bit 2 in CRx determines whether the DDRx or Px are accessed. Each port has 2 control lines: CA1, CA2, CB1, CB2. The Cx1 are input only, the Cx2 can be input or output.

PIAs have two interrupt lines: IRQA and IRQB. They may go low on a change on the inputs of the Cxy lines. These interrupts may be enabled on either an 1->0 or an 0->1 transition. The selected transition is called the "active" transition. The flags which register that an active transition has occurred are reset by reading the appropriate PORT register.

Register map:

E810 PORT A or DDR A: Data Direction Register A
E811 CRA: Control Register A
E812 PORT B or DDR B: Data Direction Register B
E813 CRB: Control Register B

Control registers:

CRA:

bit	meaning			
7	CA1 active transition flag. 1= 0->1, 0= 1->0			
6	CA2 active transition flag. 1= 0->1, 0= 1->0			
5	CA2 direction	1 = out	0 = in	
4	CA2 control	Handshake=0	Manual=1	Active: High=1 Low=0
3	CA2 control	On Read=0	CA2 High=1	IRQ on=1, IRQ off=0
		Pulse =1	CA2 Low=0	
2	Port A control: DDRA = 0, IORA = 1			
1	CA1 control: Active High = 1, Low = 0			
0	CA1 control: IRQ on=1, off = 0			

CRB works identical for CB1 and CB2, except for the differences

in
handshaking.

The Cx2 handshake is not identical between ports. For port A, the handshake
is on reading the PORT A register, for CB2 the handshake is sent on writing
the PORT B register.

On the listening side:

BIT 3 LOW with CA2: CA2 is now controlled by two events:
    (i) CA1 active transition sets it high ("Data Valid")
    (ii) a READ operation sets it low ("Data Accepted").

On the talking side:

BIT 3 LOW with CB2: CB2 is now controlled by two events:
    (i) CB1 active transition sets it high ("Ready For Data")
    (ii) a WRITE operation sets it low ("Data Valid").

Bit 2 HIGH: Causes pulse output, CA2 or CB2 going low for one cycle only
after a read or write operation. This pulse may be too short for some uses.

For operation with a sending and a receiving PIA one would connect

```
talker      listener
------      --------
Port B   -> Port A
CB2      -> CA1 with active = 1->0 (data valid)
CB1     <-  CA2 with active = 1->0 (data accepted and ready for more)
```

Use of PIA signals in a PET:

PIA 1

```
E810 PORT A    7   Diagnostic sense (pin 5 on the user port)
          6   IEEE EOI in
          5   Cassette sense #2
          4   Cassette sense #1
             3-0 Keyboard row select (through 4->10 decoder)
E811 CA2         output to blank the screen (old PETs only)
                 IEEE EOI out
```

```
     CA1          cassette #1 read line
E812 PORT B    7-0 Contents of keyboard row
                   Usually all or all but one bits set.
E813 CB2          output to cassette #1 motor: 1=on, 0=off
     CB1          screen retrace detection in

PIA 2

E820 PORT A       Input buffer for IEEE data lines
E821 CA2          IEEE NDAC out
     CA1          IEEE ATN in
E822 PORT B       Output buffer for IEEE data lines
E823 CB2          IEEE DAV out
     CB1          IEEE SRQ in
```

The VIA

The VIA is a superset of the PIA. Many of the principles apply here as well, though the organisation is slightly different.

The VIA has two ports, PA and PB, 4 control lines C[AB][12], an 8-bit shift register SR and 2 timers TA and TB.

Like the PIA, the Cx1 lines are input only, the Cx2 lines are I or O.

Port A has two registers. One register causes handshaking with CA1 to happen, the other doesn't. Port B occurs in memory before port A. There are control registers CRA, CRB, ACR, PCR, and interrupt registers IFR and IER. The data direction registers DDRA and DDRB have their own addresses, unlike the PIA.

I'll only describe the extra or different features.

Timers. The VIA has two 16-bit timers T1 and T2. When written to the timers start counting down on each clock cycle, and when reaching 0000 will flag and may cause an interrupt or other special action. When the low byte of the timer is read, the interrupt flag is cleared. When writing

to the high
byte clears the flag and starts the timer counting.

T1 has a latch register. This is a place to store the timer value before it
will be used. When T1 reaches 0, the latch value is moved into the timer so
that the countdown may begin all over again, if so enabled.

Ports. PA and PB may be latched, so that on an active transition of CA1 the
value in the PA register is retained indefinitely (or until the next active
transition on that pin), and similarly with CB1 and PB.

The shift register. This 8-bit register is connected to CB2. On command the
SR performs 8 shifts, moving 8 bits to or from CB2 one at a time. The most
significant bit is moved first (the register shifts left). The SR can be
timed by T2 and at the same rate as the 6202, using the phase 2 clock phi2.
Alternatively an external clock may time it.

It seems there are bugs in the shift register. For instance, when doing
tape I/O, the CB2 sound, which uses the SR, must be turned off.

The ACR controls the timers, shift register and latch status of PA and PB.
(T1 has effects on pin PB7, which are not described in the book, but I
suspect they are similar to the features of the CIA in this respect. -Olaf)

ACR E84B

7-6 Timer 1 control
    0 = PB7 unused
    0 0 = one shot
    0 1 = continuous, i.e. on underflow timer restarts at latch value.
    (The following 3 lines are guessed, based on the Amiga's CIA description)
    1 = PB7 used
    1 0 = T1 underflow toggles PB7

```
    1 1 = T1 underflow pulses PB7
5   Timer 2 control
    0 = one shot
    1 = count set no. of PB6 pulses
4-2 Shift register control
    000 = shift reg disabled
    001 = shift in by timer 2
    010 = shift in by phi2
    011 = shift in by external clock (PB6?)
    100 = free run by timer 2 (setting for sound)
          this means keep shifting the same byte out over and
over.
       T2 is hereby set to continuous mode.
    101 = shift out by timer 2
    110 = shift out by phi2
    111 = shift out by external clock (PB6?)
1   Port B latch
    0 = disabled
    1 = enabled on CB1 transition (in/out)
0   Port A latch
    0 = disabled
    1 = enabled on CA1 transition (in)

The Periheral Control Register PCR  controls the operating modes
of the
control lines CA1-CB2.

PCR E84C

7-5 CB2 control
 7  direction
    1 = output
    1 0 = do handshake
    1 0 0 = on write
    1 0 1 = pulse?
    1 1 = manual CB2 control
    1 1 0 = CB1 low
    1 1 1 = CB1 high
    0 = input
    0 0 x = Active high: 0->1
    0 1 x = Active low:  1->0
    0 x 0 = Clear IFR/ORB (don't ask me what this means -Olaf)
    0 x 1 = Clear IFR
4   CB1 control
    0 = active transition High
    1 = active transition Low
3-1 CA2 control (similar to CB2 control)
```

```
 3  direction
    1 = output
    1 0 = do handshake
    1 0 0 = on read
    1 0 1 = pulse
    1 1 = manual CA2 control
    1 1 0 = CA1 low
    1 1 1 = CA1 high
    0 = input
    0 0 x = Active high: 0->1
    0 1 x = Active low:  1->0
    0 x 0 = Clear IFR/ORB (don't ask me what this means -Olaf)
    0 x 1 = Clear IFR
```

The handshaking is not specifically described in the book, so I presume it
is identical to that of the PIA.

Interrupt Flag Register IFR and Interrupt Enable Register IER. These
registers are symmetrical wrt each other. The IER enables specific
interrupts (i.e., allows an event to trigger an interrupt), and the IFR
flags if the event took place.

When writing to the IER bit 7 controls the meaning of the 1 bits in the
data: when set each 1-bit sets the interrupt enable bit, when cleared it
resets that interrupt enable bit.

IFR E84D
IER E84E

```
7   IFR: master flag bit: 1 when any of the other bits are set
    IER: 0 disables, 1 enables interrupts
6   Timer 1 underflow
5   Timer 2 underflow
4   CB1 active transition
3   CB2 active transition
2   Shift register full/empty
1   CA1 active transition
0   CA2 active transition
```

Use of VIA signals in a PET

```
E840 PORT B    7   DAV in
          6   NRFD in
          5   Video retrace in
          4   Tape #2 motor (note 2)
          3   Tape data out
          2   ATN out
          1   NRFD out
          0   NDAC out
E841 PORT A        USER PORT with CA2 handshake (note 1)
E842 DDRB 7-0 normal bits set to 0 0 0 1 1 1 1 0
E843 DDRA 7-0 USER PORT data direction register
E844 Timer 1 LO
E845 Timer 1 HI  (set to $FF on system power-on)
E846 Timer 1 latch LO
E847 Timer 1 latch HI
E848 Timer 2 LO
E849 Timer 2 HI
E84A Shift register
E84B ACR  Aux. control register; set to $00 at power on
          7-6 timer 1 control
             5   timer 2 control
          4-2 shift register control
             1   port B latch
          0   port A latch
E84C PCR  Peripheral Control Register; set to $0C or $0E at
power on
          7-5 CB2 control (user port pin M) (note 3)
          4   CB1 control (note 3)
          3-1 CA2 control (graphics mode) (note 3)
          0   CA1 control (note 3)
E84D IFR  Intertrupt Flag Register; set to $00 at power on
          7   IRQ on/off
          6   T1 interrupt flagged
          5   T2 interrupt flagged
          4   CB1 interrupt flagged
          3   CB2 interrupt flagged
          2   shift register interrupt flagged
          1   CA1 interrupt flagged
          0   CA2 interrupt flagged
E84E IER  Interrupt Enable Register; set to $80 at power on
          7   1=enable, 0=disable
          6-0 enable interrupts; same bits as in IFR.
E84F PORTA         USER PORT without CA2 handshake

Note 1: E84F is the preferred user port register, since CA2
controls screen
graphics.
```

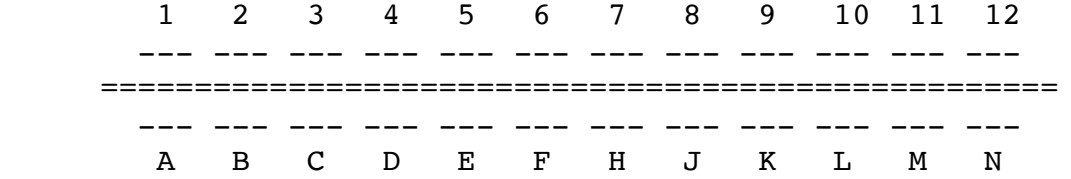
Note 2: The motor is on when this line is low, and off when it is high.

Note 3: CA1 is connected to pin B of the user port. Pins B-L correspond to
port A, which is invaiably E84E. CB2 (connected to the shift register) also
connects to pin M of the user port; square wave tones (see chapter 9 of
"Programming the PET/CBM") use these facts. CB1 signals input from cassette
#2. CA2 controls screen graphics: it is configured for output, and, when
low, gives lower case characters and others. When high, the mode is upper
case and graphics.

IEEE Port Pinouts

This info taken from Transactor Volume 4, Issue 05.

rear view for IEEE and User port:



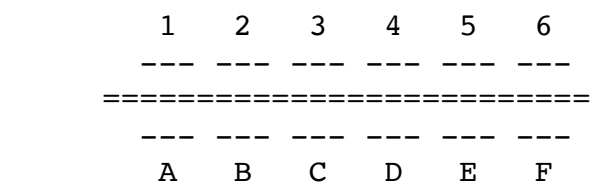
The User port is the second port from the left, the IEEE-488 port is
the third port.

PET Pin#	IEEE Pin#	Mnemonic	Definition
1	1	DIO1	Data Input/Output Line #1
2	2	DIO2	Data Input/Output Line #2
3	3	DIO3	Data Input/Output Line #3
4	4	DIO4	Data Input/Output Line #4
5	5	EOI	End Or Identify (PIA1 PA6 in, PIA1 CA2 out)
6	6	DAV	Data Valid
7	7	NRFD	Not Ready For Data
8	8	NDAC	Not Data Accepted

9	9	IFC	Interface Clear
10	10	SRQ	Service Request (PIA2 CB1)
11	11	ATN	Attention (VIA PB2 out, PIA2 CB1)
12	12	GND	Chassis Ground (IEEE cable shield)
A	13	DIO5	Data Input/Output Line #5
B	14	DIO6	Data Input/Output Line #6
C	15	DIO7	Data Input/Output Line #7
D	16	DIO8	Data Input/Output Line #8
E	17	REN	Remote Enable
F	18	GND	DAV Ground
H	19	GND	NRFD Ground
J	20	GND	NDAC Ground
K	21	GND	IFC Ground
L	22	GND	SRQ Ground
M	23	GND	ATN Ground
N	24	GND	Data Ground (DIO1-8)

Pin#	Function	Description
1	Ground	System Ground
2	TV Video	Video Out for external displays
3	SRQ	Connected to IEEE-488 SRQ (PIA2 CB1)
4	EOI	Connected to IEEE-488 EOI (PIA1 PA6 in, PIA1 CA2 out)
5	Diagnostic Sense (PIA1 PA7)	Held low causes power up to diagnostic routines or monitor
6	Read 1	Connected to cassette #1 read line (PIA1 CA1)
7	Read 2	Connected to cassette #2 read line (VIA CB1)
8	Write	Diagnostic tape write verify
9	Vert	TV Vertical for external displays
10	Horiz	TV Horizontal for external displays
11	GND	
12	GND	
A	GND	
B	CA1	Edge sensitive input for 6522 VIA
C	PB0	PB0-7 are independently programmable
D	PB1	for Input or Output
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	CB2	Special IO pin of VIA, connected to shift register
N	GND	Digital ground

Cassette port



Pin#	Name	Description
A-1	GND	Digital Ground
B-2	+5V	+5 Volts to operate cassette circuitry only
C-3	Motor	Computer controlled +6V for cassette motor
D-4	Read	Read line from cassette
E-5	Write	Write line to cassette
F-6	Sense	Monitors closure of any locking type cassette switch

-Olaf.
--
___ Olaf 'Rhialto' Seibert rhialto@mbfys.kun.nl Ooey-Gooey-Fluffy-Barfie
\X/ I'm not weird, I'm differently perceived. D787B44DFC896063 4CBB95A5BD1DAA96

```
;    Commodore PET 8032 ROM Memory Map
;    V1.2 19 DEC 1994

;
; Data types in headers (for reassembler):
;
;    DATA      Misc data
;    TEXT      String terminated with 00
;    WORD      Vectors in LO/HI byte pairs
;    CHIP      I/O Area
;    EMPTY          ROM containing FF's or AA's
;


;
; BASIC 4.0 interpreter ROM ($B000 - $CFFF)
;

b000 stmdsp    BASIC Command Vectors              WORD
b066 fundsp    BASIC Function Vectors             WORD
b094 optab     BASIC Operator Vectors             DATA
; Each Operator Vector is preceded by a priority code.
```

```
b0b2 reslst    BASIC Command Keyword Table        DATA
b13d msclst    BASIC Misc. Keyword Table          DATA
b154 oplist    BASIC Operator Keyword Table       DATA
b161 funlst    BASIC Function Keyword Table       DATA
b1b1 -    Additional Action Keywords         DATA
b20d errtab    Error Message Table            DATA
b306 okk  Misc. Messages                  DATA
b322 fndfor    Find FOR/GOSUB Entry on Stack
b350 bltu Open Space in Memory
b393 getstk    Check Stack Depth
b3a0 reason    Check Memory Overlap
b3cd omerr     Output ?OUT OF MEMORY Error
b3cf error     Error Routine
b3f4 errfin    Break Entry
b3ff ready     Restart BASIC
b406 main Input & Identify BASIC Line
b41f main1     Get Line Number & Tokenise Text
b425 inslin    Insert BASIC Text
b4b6 linkprg   Rechain Lines
b4e2 inlin     Input Line Into Buffer
b4fb crunch    Tokenise Input Buffer
b5a3 fndlin    Search for Line Number
b5d2 scrtch    Perform [new]
b5ee clear     Perform [clr]
b622 stxpt     Reset TXTPTR
b630 list Perform [list]
b6ab qplop     Handle LIST Character
b6de for  Perform [for]
b74a newstt    BASIC Warm Start
b75f ckeol     Check End of Program
b77c gone Prepare to execute statement
b785 gone3     Perform BASIC Keyword
b7a7 -    Perform [go]
b7b7 restor    Perform [restore]
b7c6 stop Perform [stop], [end], break
b7ee cont Perform [cont]
b808 run  Perform [run]
b813 gosub     Perform [gosub]
b830 goto Perform [goto]
b85d return    Perform [return]
b883 data Perform [data]
b891 datan     Search for Next Statement / Line
b8b3 if   Perform [if]
b8c6 rem  Perform [rem]
b8d6 ongoto    Perform [on]
b8f6 linget    Fetch linnum From BASIC
```

```
b930 let  Perform [let]
b94f putint    Assign Integer
b961 ptflpt    Assign Floating Point
b964 putstr    Assign String
b96e puttim    Assign TI$
b9ab      -get character string
b9ba getspt    Add Digit to FAC#1
ba4e      ?
ba88 printn    Perform [print#]
ba8e cmd  Perform [cmd]
baa2 strdon    Print String From Memory
baa8 print     Perform [print]
bac0 varop     Output Variable
bad2 -    Add zero terminator to string
badf crdo Output CR/LF
baf0 comprt    Handle comma, TAB(, SPC(
bb1d strout    Output String
bb3a outspc    Output Format Character
bb41      -Print '<cursor right>'
bb44      -Print '?'
bb46 -    Output Character in A
bb4c doagin    Handle Bad Data
bb7a get  Perform [get]
bba4 inputn    Perform [input#]
bbbe input     Perform [input]
bbe8 bufful    Read Input Buffer
; Bufful scans for a null input and calls DATA to skip to the
next statement
; On old PETs, END was called, thereby crashing the program.

bbf5 qinlin    Do Input Prompt
bc02 read Perform [read]
bc31 rdget     General Purpose Read Routine
bcf6 exint     Input Error Messages               DATA
bd19 next Perform [next]
bd5b donext    Check Valid Loop
bd84 frmnum    Confirm Result
bd98 frmevl    Evaluate Expression in Text
be81 eval Evaluate Single Term
bea0 pival     Constant - PI                 DATA
bea5 qdot Continue Expression
becc -    Evaluate <equal>
bee9 parchk    Expression in Brackets
beef chkcls    Confirm Character
beef      -Test ')'-
bef2      -Test '('-
bef5      -Test comma-
```

```
bef7      -Confirm Character in A
bf00 synerr    Output ?SYNTAX Error
bf05 domin     Set up NOT Function

bf0c      Patch for getspt
bf21      ?
bf2e      Patch for [input]
bf41      Unused                          EMPTY

bf8c isvar     Search for Variable
bfad tisasc    Convert TI to ASCII String
c003      -read real time clock
c047 isfun     Identify Function Type
c051 strfun    Evaluate String Function
c071 numfun    Evaluate Numeric Function
c086 orop Evaluate <or>
c089 -    Evaluate <and>
c0b6 dorel     Evaluate <less> (comparison)
c0bb numrel    Numeric Comparison
c0ce strrel    String Comparison
c11e dim  Perform [dim]
c12b ptrget    Identify Variable
c187 ordvar    Locate Ordinary Variable
c1b6 isletc    Does A hold an alphabetic character?
c1c0 notfns    Create New Variable
c1cb notevl    Create Variable
c2c8 aryget    Allocate Array Pointer Space
c2d9 n32768    Constant 32768 in Flpt             DATA

c2dd intidx    Evaluate Text for Integer
c2ea ayint     FAC#1 to Positive Integer
c2fc isary     Get Array Parameters
c343 fndary    Find Array
c370 bserr     ?BAD SUBSCRIPT
c373 -    ?ILLEGAL QUANTITY
c378 -    ?REDIM'D ARRAY
c38c notfdd    Create Array
c439 inlpn2    Locate Element in Array
c477 umult     Number of Bytes in Subscript
c4a8 fre  Evaluate <fre>
c4bc givayf    Convert Integer in (AC/YR) to Flpt
c4c9 pos  Evaluate <pos>
c4cf errdir    Confirm Program Mode
c4d7 -    ?UNDEF'D FUNCTION
c4dc -    Perform [def]
c50a getfnm    Check Syntax of FN
c51d fndoer    Perform [fn]
```

```
c58e strd Evaluate <str$>
c598 strlit    Set Up String
c5fe putnw1    Save String Descriptor
c61d getspa    Allocate Space for String
     garbag    Garbage Collection
     dvars     Search for Next String
     grbpas    Collect a String
c74f cat  Concatenate Two Strings
c78c movins    Store String in High RAM
c7b5 frestr    Perform String Housekeeping
c811 frefac    Clean Descriptor Stack
c822 chrd Evaluate <chr$>
c836 leftd     Evaluate <left$>
c862 rightd    Evaluate <right$>
c86d midd Evaluate <mid$>
c897 pream     Pull String Parameters
c8b2 len  Evaluate <len>
c8b8 len1 Exit String Mode
c8c1 asc  Evaluate <asc>
c8d1 gtbytc    Evaluate Text to 1 Byte in XR
c8d4      -Eval Byte Parameter
c8e3 val  Evaluate <val>
c8eb strval    Convert ASCII String to Flpt
c921 getnum    Get parameters for POKE/WAIT
c92d getadr    Convert FAC#1 to Integer in LINNUM
c943 peek Evaluate <peek>
c95a poke Perform [poke]
c963 wait Perform [wait]
c97f faddh     Add 0.5 to FAC#1
c986 fsub Perform Subtraction
c989 -    Evaluate <subtract>
c998 fadd5     Normalise Addition
c99d fadd Perform Addition
c9a0 -    Evaluate <add>
ca7d negfac    2's Complement FAC#1
cab4 overr     Output ?OVERFLOW Error
cab9 mulshf    Multiply by Zero Byte
caf2 fone Table of Flpt Constants            DATA

;caf2     1.00
;caf7     #03       (counter)
;caf8     0.434255942
;cafd     0.57658454
;cb02     0.961800759
;cb07     2.885390073
;cb0c     0.707106781    SQR(0.5)
;cb11     1.41421356     SRQ(2)
```

```
;cb16     -0.5
;cb1b     0.693147181    LOG(2)
;

cb20 log  Evaluate <log>
cb5e fmult     Perform Multiply
cb61 -    Evaluate <multiply>
cb8f mulply    Multiply by a Byte
cbc2 conupk    Load FAC#2 From Memory
cbed muldiv    Test Both Accumulators
cc0a mldvex    Overflow / Underflow
cc18 mul10     Multiply FAC#1 by 10
cc2f tenc Constant 10 in Flpt           DATA
cc34 div10     Divide FAC#1 by 10
cc3d fdiv Divide FAC#2 by Flpt at (AC/YR)
cc45 fdivt     Divide FAC#2 by FAC#1
cc48 -    Evaluate <divide>
ccd8 movfm     Load FAC#1 From Memory
ccfd mov2f     Store FAC#1 in Memory
cd0a -    Store FAC#1 at (AC/YR)
cd32 movfa     Copy FAC#2 into FAC#1
cd42 movaf     Copy FAC#1 into FAC#2
cd51 round     Round FAC#1
cd61 sign Check Sign of FAC#1
cd6f sgn  Evaluate <sgn>
cd8e abs  Evaluate <abs>
cd91 fcomp     Compare FAC#1 With Memory
cdd1 qint Convert FAC#1 to Integer
ce02 int  Evaluate <int>
ce29 fin  Convert ASCII String to a Number in FAC#1
cee9 n0999     String Conversion Constants        DATA

cef8      Unused              EMPTY

cf78 inprt     Output 'IN' and Line Number
cf93 fout Convert FAC#1 to ASCII String
d01e foutim    Convert TI to String
d0c7 fhalf     Table of Constants            DATA

d108 sqr  Evaluate <sqr>
d112 fpwrt     Evaluate <power>
d14b negop     Negate FAC#1
d14b -    Evaluate <greater>
d156 logeb2    Table of Constants            DATA

;d156     1.44269504     (1/LOG to base 2 e)
;d15b     #07       (counter)
```

```
;d15c     2.149875 E-5
;d161     1.435231 E-4
;d166     1.342263 E-3
;d16b     9.6414017 E-3
;d170     5.550513 E-2
;d175     2.402263 E-4
;d17a     6.931471 E-1
;d17f     1.00
;

d184 exp  Evaluate <exp>
d1d7 polyx     Series Evaluation
d221 rmulc     Constants for RND            DATA
d229 rnd  Evaluate <rnd>
d282 cos  Evaluate <cos>
d289 sin  Evaluate <sin>
d2d2 tan  Evaluate <tan>
d2fe pi2  Table of Trig Constants           DATA

;d2fe     1.570796327    pi/2
;d303     6.28318531     pi*2
;d308     0.25
;
;d30d     #05  (counter)
;d30e     -14.3813907
;d313      42.0077971
;d318     -76.7041703
;d31d      81.6052237
;d322     -41.3417021
;d327      6.28318531
;

d32c atn  Evaluate <atn>
d35c atncon    Table of ATN Constants            DATA

;d35c     #0b  (counter)
;d35d     -0.000684793912
;d362      0.00485094216
;d367     -0.161117018
;d36c      0.034209638
;d371     -0.0542791328
;d376      0.0724571965
;d37b     -0.0898023954
;d380      0.110932413
;d385     -0.142839808
;d38a      0.19999912
;d38f     -0.333333316
```

```
;d394      1.00
;

d399 initat    CHRGET For Zero-page
d3b1 rndsed    RND Seed For zero-page             DATA
;d3b1     0.811635157

d3b6 init BASIC Cold Start

     initcz    Initialize BASIC RAM
d417 initms    Output Power-Up Message
     initv     Initialize Vectors
d44b words     Power-Up Message              DATA

d472 ?


;
; Machine Language Monitor
;

d4ba *Get Command

d531 Print Space
d534 New Line
d539 Increment Pointer
d544 Commands                        DATA
d54c Command Vectors                           DATA
d55c '<cr>     pc  irq  sr ac xr yr sp'        DATA

d579 ?

d587 Perform [r]
d5bc Perform [m]
d5fb Perform [;]

d61d Perform [:]
d633 Perform [g]
d66b Perform [x]
d675 Perform [l/s]

d722 Output 2-digit Byte
d731 Output 2 Ascii Chars
d73a Byte to 2 Ascii
d744 Swap Add_ with Add_

d7a4 Print '?'
```

```
d7af Perform [record]
d873 Perform [catalog/directory]
d942 Perform [dopen]
d977 Perform [append]
d9d2 Perform [header]
da07 Perform [dclose]
da65 Perform [collect]
da7e Perform [backup]
daa7 Perform [copy]
dac7 Perform [concat]
db0d Perform [dsave]
db3a Perform [dload]
db55 Perform [rename]
db66 Perform [scratch]

de9e ?              DATA
dea4 ?              DATA
dec2 Unused              EMPTY


;
; Editor (E000)
;

e202 Output to Screen
e3b6 Output <CR>
e442 Main IRQ Entry Point
e600 Exit Interrupt
e788 Unused              EMPTY


;
; Kernel (F000-FFFF)
; Tape processing, IEEE-488, jump table
;


f185 Output Kernal Error Message
f2a6 clrchn    Restore Default I/O

f266 chrout    Output One Character
f2dd closet    Perform [close]
f401 verfyt    Perform [load]
f4f6 verfyt    Perform [verify]
f560 opent     Perform [open]
f6c3 syst Perform [sys]
```

```
f6dd savet     Perform [save]
f7af chkin     Set Input Device
f7fe chkout    Set Output Device
fd16 -    Power-Up RESET Entry
fd49 -    NMI Transfer Entry
fd4c -    Kernal Reset Vectors              WORD
fd5d -    Unused                        EMPTY


;
; PET 8032 Jump Table
;

ff93      Perform [concat]
ff96      Perform [dopen]
ff99      Perform [dclose]
ff9c      Perform [record]
ff9f      Perform [header]
ffa2      Perform [collect]
ffa5      Perform [backup]
ffa8      Perform [copy]
ffab      Perform [append]
ffae      Perform [dsave]
ffb1      Perform [dload]
ffb4      Perform [catalog/directory]
ffb7      Perform [rename]
ffba      Perform [scratch]

ffc0      Perform [open]
ffc3      Perform [close]

ffc6 chkin     Set Input
ffc9 chkout    Set Output
ffcc clrch     Restore I/O Vector
;ffcf     chrin     Input Vector, chrin
ffd2 chrout    Output Vector, chrout


ffd5      Perform [load]
ffd8      Perform [save]
ffdb      Perform [verify]
ffde sys  Perform [sys]

ffe4 jmp  getin     Get From Keyboad


;fff6     Vectors
```

```
fff6 [AAAA]         -         WORD
fff8 [AAAA]         -         WORD

;fffa     Transfer Vectors
fffa [fd49]         NMI       WORD
fffc [fd16]         RESET           WORD
fffe [e442]         IRQ       WORD
```