

# Re: OT: DOS programming EPP

---

*Source:* <http://sci.tech-archive.net/Archive/sci.electronics.design/2006-01/msg03966.html>

---

- *From:* Robert Baer <[robertbaer@xxxxxxxxxxxxx](mailto:robertbaer@xxxxxxxxxxxxx)>
  - *Date:* Mon, 23 Jan 2006 02:33:12 GMT
- 

PaulCsouls wrote:

On Sun, 22 Jan 2006 03:14:06 GMT, Robert Baer  
<[robertbaer@xxxxxxxxxxxxx](mailto:robertbaer@xxxxxxxxxxxxx)> wrote:

I have done a websearch, and there is nothing available that explicitly shows how to program a parallel port in the EPP mode, and how to \*safely\* interface with it.

The best (incomplete) source i found was on the beyondlogic.org site.

However, no matter what i do, the nominally input printer pins (pin 1 = strobe, pin 13 = select and pin 15 = error bar) act like outputs.

Data lines when low safely sink 2mA (did not try more as i did not want to zap the MB) R=45 ohms, and when high safely source 1mA (did not try more R=2.2K).

"Strobe" line pin 1 was always high and safely sink 1.5mA R=730 ohms.

"Select" line pin 13 was always high and safely sink 1.0mA R=2.2K.

There seems to be \*no\* specifications or equivalent circuits for the parallel port as implemented on the ASICs used in modern PCs.

Therefore, it is completely unknown as to the maximum safe sink current to a logic low pin or the maximum source current from a logic high pin.

It is not wise to force a pin that is acting as an output, into the opposite state; so the info is necessary for safety.

I want and need to program this in DOS.

## Re: OT: DOS programming EPP

\*\* as an aside, it was interesting to see that when Windoz booted after my fiddling, that i saw "detecting new hardware" etc.

Go back to the beyond logic page and read the parallel port FAQ from the beginning. The EPP page assumes you did. You need to set up the extended control register (BASE + 0x400). In EPP mode Strobe is the WRITE/NOT READ bit and an output. I think you need to do some C or BASIC code to do any of this. I don't think you can just use DOS shell commands.

Paul C

I said \*NOTHING\* about DOS shell commands!  
What follows is a listing of a BASIC program that i have used for preliminary investigation (use monospacing font width 90):

```
' Attempt to read data from parallel port using EPP protocol
DEFINT A-Z
PRNT = &H378
DATAs = PRNT + 0: STATUS = PRNT + 1: CONTROL = PRNT + 2 'SPP
' ^--r/w          ^--read only          ^--r/w (from IBM)
ADDRESSrW = PRNT + 3: DATArW = PRNT + 4 'EPP
' ^--pin 17 pulse*          ^--pin 14 pulse*  *=if only one used
BIT5 = &H20: PIN17 = &H8
' The following pins (as output) must be high:
' Address Strobe=pin 17, Data Strobe=pin 14, Write=pin 1, Reset=pin 16.
' NOTE: After system boot, all pins default high except data pins and pin 17.
'          CONTROL bit 5 defaults low.

CLS
LOCATE 1, 1
PRINT "  First read STATUS bits (cannot write them)          meas OK?  init"
SEN = INP(STATUS)
PRINT "D0:"; (SEN AND &H1)/&H1, "ghost Pin 01 /STROBE (reads inverted) 1  N  0"
PRINT "D1:"; (SEN AND &H2)/&H2, "ghost Pin 14 /AUTO FD (reads inverted) 1  N  1"
PRINT "D2:"; (SEN AND &H4)/&H4, "ghost Pin 16 /INIT                          1  Y  1"
PRINT "D3:"; (SEN AND &H8)/&H8, "Pin 15 /ERROR                          1  Y  1"
PRINT "D4:"; (SEN AND &H10)/&H10, "Pin 13 SLCT (from printer)                1  Y  1"
PRINT "D5:"; (SEN AND &H20)/&H20, "Pin 12 PE (reads inverted)                1  Y  1"
PRINT "D6:"; (SEN AND &H40)/&H40, "Pin 10 /ACK (reads inverted)                1  Y  1"
PRINT "D7:"; (SEN AND &H80)/&H80, "Pin 11 BUSY (reads inverted)                1  Y  0"
PRINT "  Then read CONTROL bits"
CTL = INP(CONTROL)
```

## Re: OT: DOS programming EPP

```

PRINT "D0:"; (CTL AND &H1)/&H1, "Pin 01 /STROBE (reads inverted)      1   Y   0"
PRINT "D1:"; (CTL AND &H2)/&H2, "Pin 14 /AUTO FD (reads inverted)    1   Y   0"
PRINT "D2:"; (CTL AND &H4)/&H4, "Pin 16 /INIT                          1   Y   1"
PRINT "D3:"; (CTL AND &H8)/&H8, "Pin 17 /SLCT IN (to printer)         0   N   1"
PRINT "D4:"; (CTL AND &H10)/&H10, "no pin IRQ EN                                0"
PRINT "D5:"; (CTL AND &H20)/&H20, "no pin bidirectional (added for EPP) 0"
PRINT "D6:"; (CTL AND &H40)/&H40, "ghost Pin 10 /ACK (reads inverted)  1   Y   1"
PRINT "D7:"; (CTL AND &H80)/&H80, "ghost Pin 11 BUSY (reads inverted)  1   Y   1"
PRINT "  Read SPP data:; HEX$(INP(DATAs)); ";";
PRINT "  Read EPP address:; HEX$(INP(ADDRESSrw)); ";";
PRINT "  Read EPP data:; HEX$(INP(DATArw))
PRINT "One can pulse pin 14 high by writing anything to EPP DATA port."
PRINT "Setting bit 3 in CONTROL low makes pin 17 high.
NCTL = CTL OR BIT5      'sets I/O bit
  WHILE INKEY$ = ""
SOUND 2000, 1
FOR I = -32767 TO 32766
  OUT CONTROL, NCTL      'output 1uSec pulses seen during pin 1 high pulse:
  OUT DATArw, &H0        ' high 2.3V@xxxx load from 0.08V low
  OUT DATArw, &HFF       ' low 2.4V@xxxx load from 3.5V high
NEXT I
  WEND
PRINT "CONTROL PORT INIT:; HEX$(CTL), "NOW:; HEX$(INP(CONTROL))
  OUT CONTROL, &HCC      'Does not completely reset...
SYSTEM
*****
  It is obvious that the data port (pins 2-9) "wants" to be an input, but
  only when pin 1 is high.
  Also, *NO* pin seems to act like an input at *any* time.
  I am afraid of using a heavier load, as at all times all pins are active
  drivers (outputs) except perhaps during the "magic" time.
  I certainly do not want to damage my motherboard!

```