

Re: Parallel port hardware

Source: <http://sci.tech--archive.net/Archive/sci.electronics.design/2007-10/msg00642.html>

- *From:* "Jon Slaughter" <Jon_Slaughter@xxxxxxxxxxx>
 - *Date:* Thu, 04 Oct 2007 20:23:40 GMT
-

"petrus" <iemand@xxxxxxxxxxx> wrote in message
[news:42ebf\\$47051e1e\\$5354d4b6\\$8627@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](news:42ebf$47051e1e$5354d4b6$8627@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

"Jon Slaughter" <Jon_Slaughter@xxxxxxxxxxx> schreef in bericht
[news:F\\$YMi.166\\$sm6.134@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](news:F$YMi.166$sm6.134@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

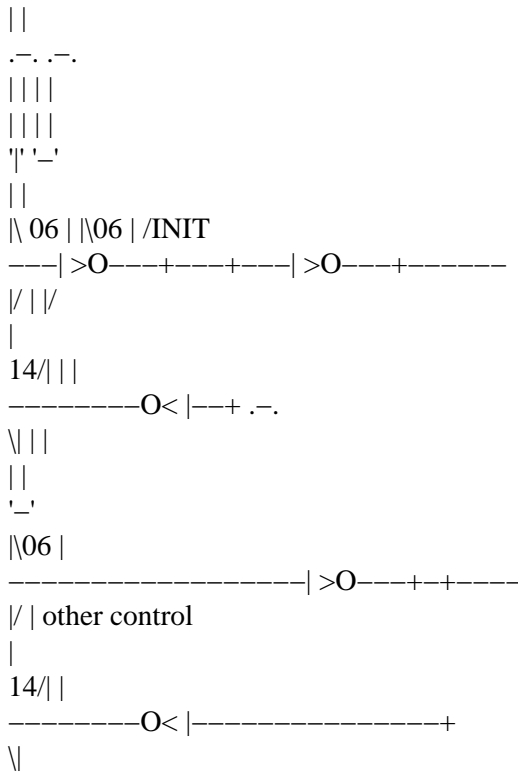
I still want to lay my hands on the original IBM hardware manual. One of the reasons is that parallel port. Nevertheless I saw several "compatibel" schematics all with several differences. One thing is clear to me: The output lines were never meant to do input. I have the schematic of a printerport that had no inputs but the status lines. Others had "inputs" on control- and data lines but they were only meant to read back the status of that outputs. In the old days inputs used to be SN74LS14 inverting Schmidt triggered buffers, as were the read back inputs of the control lines. The control line outputs used to be SN7406 inverting open collector buffers. As the control lines are open collector, you can use their read back inputs for real input when you drive that control lines high... most of the time. I have a schematic in front of me in which the read back of the /INIT control is taken from the input of the SN7406 buffer rather than from its output. So you will never read back the real status of that printer pin. No need to say it will

Re: Parallel port hardware

fail as an input as well.

I'm not sure. I think the control port was always able to do bidirectional because it was open collector(of course its true that not all parallel ports were open collector on the control port but most are)

Guesss you missed the point. That output *is* open collector but the read back buffer has not been connected to that open collector output. See below.



(created by AACircuit v1.28.6 beta 04/19/05 www.tech-chat.de)
Be aware that it is just one variant of the numerous schematics. Most printerports I met had all read backs connected to the output pins.

The schematic in the chapter "Using The Parallel Port to Input 8 Bits" will fail in this case.

So if you want to stay on the safe side, don't use output pins for

Re: Parallel port hardware

input.

If you have to, you will have to check the printerport involved for every (type of) computer. Once you'll have to do so, it will be worthwhile to check for other properties of the printerport at hand. Almost all but the oldest computers have printerports that somehow can do bidirectional data transfer. If you have the choice, use EPP ports (or USB :)

Well, I only have one other option and that is to use a status line to read in the data but then I have to "disengage" the output line from the data line or use the open collector of the control port to somehow do it (which is what I was going to do but since I can read the control port in the first place there's no real reason to use the status line because it ends up making it slower and I still have to disconnect the output line so screw everything up).

Right now I'm just trying to program the thing for my computer and I think I can do it with the control port only but it requires that I know how the hardware port works and I really have no clue. Of course experimenting tells me one thing but I'm not sure if I trust myself with it.

Can send you a scan of that printerport schematic I mentioned. Just to give you an idea about the general appearance of the hardware. As for how to handle them by the software, you have Beyond Logic. Can't see why it should be less difficult to use a control line rather than a status line for input. How many lines of what type do you need anyway? And what for?

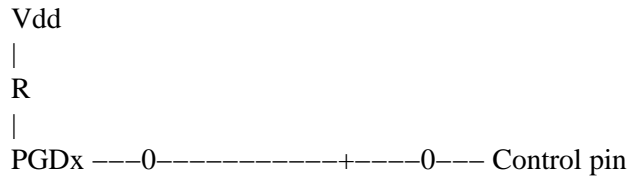
That would work. Maybe send to alt.binaries?

The only problem is because the PGDx line for the device I'm connecting to is bi-dir. There are times when it switches into send and sometimes into recv. If I just connect a status line to the line then then control pin will need to be disconnected. If not when the device goes into transmit it will short out if the control point is low impedance.

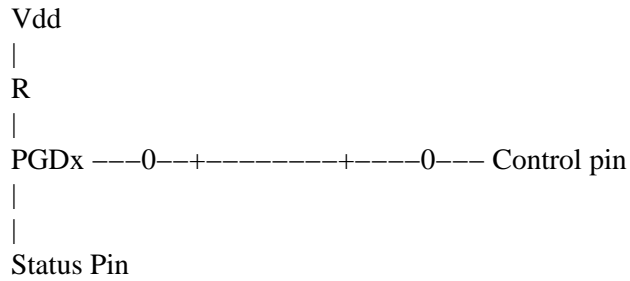
i.e.

Re: Parallel port hardware

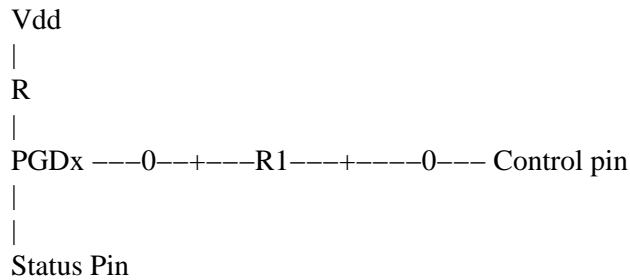
basically what I have is



Now if I hook a status pin for reading I have



If PGDx goes to Vdd then status pin reads Vdd but if control pin is low impedance (i.e., grounded because its open collector) then I have a short. I figure I can put a resistor in between like



And then I essentially can limit the current by R1 but I'm not sure how well this will work.

Just seems easier to use the "bi-dir" capabilities of the control pin in the first place and not have to worry about using a status pin.

I still have the probably of shorting if the control pin goes low-z so maybe the status pin is easier.

There is a problem of speed though as if I have to read and write to the

Re: Parallel port hardware

ports to "set them up" then it slows down the transmission speed. Writing or reading to a port takes about 3us on my computer. If I have to do it, say 3-4 times then I have significantly cut the speed down. I already do it 2-3 times because I have to have a clock that goes twice as fast. This is why if I do everything on one register I can potentially combine some writes or reads to maximize speed.

Thanks,
Jon

Using just the control port for what I want makes it very simple and "elegant" compared to "hacking" it by mixing the status port and control port. I guess the only way I'll know if it will work is to try it ;/ I really hate doing that though cause its pretty risky ;/

I like to use an extra printerport card while experimenting.

Thanks,
Jon

petrus bitbyter