

Re: OT: EPP problems

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2006-02/msg01288.html>

- *From:* Robert Baer <robertbaer@xxxxxxxxxxxxxx>
 - *Date:* Tue, 07 Feb 2006 08:51:06 GMT
-

budgie wrote:

On Mon, 06 Feb 2006 07:38:06 GMT, Robert Baer <robertbaer@xxxxxxxxxxxxxx> wrote:

budgie wrote:

On Sun, 05 Feb 2006 04:05:30 GMT, Robert Baer <robertbaer@xxxxxxxxxxxxxx> wrote:

(snip)

Well, then would you be so kind as to post the software, and the hardware interface?

This particular piece of hardware is a custom EPROM programmer for a particular client. An NDA prevents posting schematics or the substance of the code. As a result, much of the code has been omitted but enough remains to hopefully convey what is being done.

The "interface" comprises:

----- SNIPped for brevity -----

Thanks for the program; i find that use of the subroutines make for greater readability.

There is very little time penalty for a call to and return from a subroutine.

Re: OT: EPP problems

Will look it over the next few days.

As far as why i am reading that port?

Well, if one looks at the EPP documentation (which it seems you missed), that is the EPP data R/W port and one is supposed to read incoming data there.

On the old SPP cards, one can readback data that has been sent to the printer (ie: written to the data output latch and thus the pins).

That capability has been retained, but instead of actually sensing the pin levels like IBM did in their printer port cards, the newer cards seem to wimp out and read the register.

That same failure to conform to the IBM "standard" happened in a number of the 286, 386 and 486 printer cards.

So...the only way to actually read pin logic levels seems to be via the EPP data I/O port.

A low level TTL input pullup is 40K and a low level schottky input pullup is 20K.

That is the standard as defined by Texas Instruments.

I had nothing to do with that and have no control over that.

The pins of all EPP ports that i have looked at have three states: output low, output high, and "neither"; the third state acts like a 2.4K pullup to +5V with a 2.2K or a 1K pulldown. Certainly *not* like an input to any TTL device!

I also have nothing to do with that nor do i have any control over that unless i damage them by over current in either direction.

Theoretically, a 420 ohm resistor to ground would get about 0.8V for a low and IFFI (if and only if) the "input" logic threshold abides by TTL standards, then that becomes marginally acceptable.

A 200 ohm pulldown would be far better, but that could possibly result in a few watts of dissipation in the ASIC if all pins were high.

Granted, one can use a decent TTL tristate driver with 200 ohm protection resistors in series, and enable output *only* when the EPP pins are supposed to be an input; that would make the life of the ASIC better.

You'll see from my code that I read in data (PULL) at the base address. That's all I've ever done, and it has always worked. From discrete printer/Multi-I/O cards on a '286 to on-board ASIC implementations. This platform is a P3-667 and the only things NOT on-board are video and NIC.

I try to not get too tangled in what the IBM TRM or anyone else's documentation, or "standard" or implementation does. I just "do it". Call me a humming-bird if you will.

I read the code and have tried only a little part of it; basically i can flip pin#1 high or low and the data either output or "input".

Still, the so-called "hi-z" is crap; am using 200 ohm for safety pulldown to low.

What is intriguing is that yuor program controls the machine, instead of the full EPP read protocol where the \WAIT would allow hardware handshaking for the data read - but *only* if the machine presents the data during the wait time (yes - it *does* time out).

Re: OT: EPP problems

If the machine is slow, then data is not read and something must be done in the software for a "oops" and re-try.

.