

Re: send more than 8 bits with parallel port

Source: <http://sci.tech-archive.net/Archive/sci.electronics.basics/2006-02/msg01023.html>

- *From:* Jonathan Kirwan <jkirwan@xxxxxxxxxxxxxxxx>
 - *Date:* Tue, 21 Feb 2006 18:06:01 GMT
-

On 21 Feb 2006 09:17:12 -0800, "Chris" <cfoley1064@xxxxxxxx> wrote:

Jonathan Kirwan wrote:

On Tue, 21 Feb 2006 14:15:56 GMT, NoSpam@xxxxxxxx (Bob Masta) wrote:

<snip>

Others have answered the hardware aspect of your question. However, note that if you plan to use this on PCs running Windows versions later than 9x, you will have to use a special ring 0 device driver to have access to the printer port. (GIVEIO and USERPORT are two that I have heard of, but haven't tried myself.) That might be one advantage for using the serial port, which can be accessed through more-or-less standard Windows API functions.

Another thing to think about is that parallel printer ports are rumored to be slated for extinction... but then again, aren't we all!

You know? What bugs me about this very true point is that we are losing all of the really good hobbyist interfaces for adapting a PC. The ISA bus was really nice and not too complex for a serious hobbyist to use, in adding boards. Gone now, or nearly so. Ever consider trying to do a PCI card as a hobbyist? Reflection wave bus, 2ns clock skew on 33MHz and 1ns clock skew on 66MHz with a 1.5" +/- 0.1" trace length for the clock (often serpentine in order to get there), etc. Just getting equipment to monitor the analog characteristics for debugging a design is a fortune. Lose the parallel, lose the serial, add USB 2.0, replace the old IDE controller cables with SATA 2, and what are you left with to use, anymore? It's getting to be a pain in the ...

Re: send more than 8 bits with parallel port

Well, of course, there are microcontroller boards. But then you lose out on the excellent and easy availability of very excellent and well documented development tools.

Jon

Contrivers of contrivances are going to have a more difficult time when the ol' parallel port is no more, Jon. Of course, we can squirrel away more older PCs (that is, until the War Department issues the Ultimatum again).

You should see my collection. I still multi-I/O boards, floppy interface boards, etc.

I would guess, though, you'll be seeing more standardized USB experimenter-type interfaces to replace the parallel port for classroom and hobbyist work.

That adds complexity that can be difficult to diagnose. But worse, how do you arrange for precise timing control of I/O pins?? And under Windows?!?!?! It's just not the same thing, you know.

Only a few years back, I developed software for aligning and calibrating a medical infusion pump motor/cam system. I had to spin the motor and monitor it in real time, because this is how the alignment worked — at operational speed, I had to see how the cam alignment looked with respect to the optical quadrature encoder. It was different, stopped or very slow, because of the cam follower's lag as it ran. The quadrature encoder gave me 10,000 pulses per rev. And I ran the motor at 50, 150, and 300 RPM. At 300, that's 5 revs per second. And I was getting 50,000 pulses each second on A and B side channels.

It was absolutely no problem on a modern CPU with an I/O card plugged into the ISA. I sat in a tight loop and pulled out I/Os at 750ns each. I coded the quadrature decoder in assembly.

Now?? What, with some USB thing?? How will I code that, in assembly, under DOS?? And if I use Windows, what chance do I have?

No, this is a serious loss.

You'll also see more use of PCI DIO type boards for the parallel I/O. You will be seeing a lot less hobbyist bus interface.

Re: send more than 8 bits with parallel port

Well, there is that. But you cannot build them yourself and PCI isn't all that easy under DOS-only. I often need to completely remove Windows when doing embedded PC instrumentation, as timing is crucial to control and a lot of stuff I do is `_fast_` and I require very tight control over latencies. Under Windows, this is ... very difficult, and will require VxD work to get there, if at all.

I've developed PCI testing software for Intel, while running a multi-CPU O/S of my own development. We'd load up Xilinx-FPGA cards with code so that we could literally fill up the PCI bus with dense traffic and try and break the chipset with it.

I'm really not looking forward to using PCI for instrumentation. In the case I earlier mentioned, about the testers, the customer is a very large medical instrumentation builder but their need for testers is perhaps a dozen or two systems. In some cases, I've actually done systems where there were only two. They may, themselves, test a large number of devices very quickly, but they are in effect one-off systems -- custom for the need and don't represent large volume to anyone, part-wise. Buying the boards won't mean much to the vendor, so support won't be all that great. For PCI, they may not even bother with a DOS-based PCI driver. And setting up the PCI board under DOS may require my configuring all of the PCI boards, if I don't like the BIOS defaults -- which the board maker may not even care about because Windows is PNP-aware and they don't have to mess with it much.

That's assuming, of course, that I can find what I need in PCI. In ISA, at least, I can do a board design and wire wrap it up on my own and test it out. Then do a final board(s) for the customer that meets the needs. PCI?? No way. I just don't have the equipment or the knowledge, except that I know enough to watch out.

But it should still beat those "thrilling days of yesteryear" (early '80s), when PCs cost thousands, you had to wait in line for one, and you took your career in your hands every time you turned on the power for a new prototype, consigning it as well as everything else in the box and possibly your job to the tender(?) mercies of a harsh 85 watt switching power supply.

What I'd like to do is be able to capitalize on modern low-cost systems and still have access to low-development-cost interfacing for small-run or one-off tester systems for customers or personal use. There is no sense in taking on an expensive PCI design project for what would otherwise merely be a simple, customized I/O board for external instrument control. Think of chicken hatcheries, IC testers, LED binning, etc. etc. Highly integrated, labor-intensive, one-off systems with a lot of knowledge buried in them about the specific requirements. ISA still makes sense for these, as even a custom logic

Re: send more than 8 bits with parallel port

Re: send more than 8 bits with parallel port

board is a no-brainer to do and test out. PCI raises the bar so high that many of these projects would find their labor growing very substantially for no good reason except the fact that ISA has disappeared.

I learned about the advantages of the trailing edge back then, picking up cheapie Kaypro IIs for their serial and parallel ports as IBM and Micro Soft (two words back then) cleaned Kaypro's and CP/M's clocks. Everyone who wanted to replace VisiCalc with Lotus had their first obsolete computer (aaaaww!), and after having it sit around unused for a few months, was willing to practically give it away (for a couple hundred dollars).

Those were the days. Womenfolk swooned as real engineers and techs walked by, TI-57s bobbling suggestively from their belt loops. ;-)

My first Kaypro was a 286i that I bought for about \$3k, new. It was the first PC-compatible on the market that was really PC-compatible based on careful testing. The other systems would fail to even run PC applications at least 10% of the time. More, often. But this one would run everything I tried, out of the box. And was the only system at the time, other than an IBM of course, that could.

I remember when visicalc first came out -- not on the PC, but I believe on the Apple II, as the PC didn't exist then (early 1980, I think, when I first saw it.) I laughed at it, because I thought "who'd buy this?" It seemed way too technical to reach any popular market and I figured programmers could just program the equations.

Oh, well. What do I know?

Jon

.