

## Re: Events on Serial port

---

*Source:* <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2008-07/msg00346.html>

---

- *From:* Derek Philip <derek.philip@xxxxxxxxxx>
  - *Date:* Wed, 09 Jul 2008 00:05:52 +0100
- 

Rolf Schroedter wrote:

Derek wrote, On 07.07.2008 22:33:

...  
Looking on the scope a buad rate of just 115200 seems to be fine, I'm keeping to this as I can't predict the performance of the inbuilt uarts. I assume that a modern PC will do 115200.

Yes, generally there is no problem. Everything depends on the total data amount and your required response time.

... Since starting this thread I found that Windows does not want to play ball.

I know that the script side of things can handle detecting and processing the data.  
I know that the H/W can deliver the data at the rate required, it all works under Linux  
However it seems under windows I'm missing some aspect of the chain of command, I do have to do more testing to determine if there is any thing "on the wire" that could be giving windows cause for concern.

I suspect my use of the flow control lines to indicate data available may be causing the windows a problem.

Cant remember the fconfigure off the top of my head, I'll post it later.

Yes, please do this.  
And please try to describe briefly your communication requirements.  
Who is the master, who is the slave ?

The PC is the master it controls the flow of actual data across the uart and the I2C bus. It only polls the interrupt/DSR line to see if data is available.

## Re: Events on Serial port

Do both sides send data synchronously/asynchronously (logically) ?

My findings today suggest that this is not quite as simple as I thought as I have two protocols running sequentially, so in theory I could write over the uart while receiving data from the I2C.

Fixed block sizes ? Required timing ?

No fixed block, I request from 1 to 27 bytes, depending on the Radio register I want to read, there is no direct register addressing. If you want register number 27 you have to read them all in!

My observations on the scope suggest that the data can be requested and sent from the target in under 6 ms. This is looking at the Tx and Rx lines of the uart mind and does not account for OS latency, although on the read request, using some debug signaling via the DTR line, there seems to be little latency, unless it is also affecting the control lines. Again using the same debug signaling I have seen 5 or more milli seconds of latency before processing the data starts.

Regards, Rolf.

I did make some progress today.  
I found that sending 20+ bytes of data through the uart to I2C device followed quickly by a read of 20+ bytes screws up.

Looking on the scope the write only takes a couple of milliseconds, but I'm leaving the procedure and calling the next read, before this in some cases. with a 15– 20ms delay in its gets further.

What I should do is query the I2C status registers on the uart to I2c chip to confirm that the data went over the I2C. There are timeouts and nack etc status fields.

I also managed to poll the stae of the DSr line and get some RDS data in.

Again observations on the scope indicate that the after command isn't as reliably timed as under Linux. I set the after for 8 milliseconds and get 15!, shrinking it down to 3 didn't seem to do much, but that was a quick test before leaving for the day. In this case I loose at least 4 blocks of RDS data on a regular basis.

Things I should check now then is how reliable the after command is on Windows. I'm assuming the poll period on the uart setting does not affect reading the state of the control lines.

Check that I can read the state of the DSR reliably.

My fconfigure by the way is

```
fconfigure $fd -mode 115200,n,8,1 -handshake none -translation binary -blocking 0 -buffering none
```

More timing test tomorrow I think, find out where

Derek

Re: Events on Serial port

Re: Events on Serial port