

Re: Linux serial port dropping bytes

Source: <http://coding.derkeiler.com/Archive/General/comp.arch.embedded/2008-04/msg00108.html>

- *From:* Anton Erasmus <nobody@xxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 02 Apr 2008 23:50:44 +0100
-

On Wed, 02 Apr 2008 08:57:52 -0500, CBFalconer <cbfalconer@xxxxxxxxxx> wrote:

David Brown wrote:

CBFalconer wrote:

<snip>

I left the whole thing unsnipped. The time has come for me to crave forgiveness. I think I have been afflicted with age or something. The bits/persec crowd are absolutely correct, and I am wrong.

I don't think you need forgiveness – you just made a mistake.

So that leaves the real problem handling throughput of approximately 1 char each 10 microsec.

You need to handle an *average* of 1 character per 10 us. But the cost of handling each character is peanuts – even if the UART is on a slow bus, you should be able to read out characters at something like 20 per us. The cost is in the handling of the interrupt itself – context switches, cache misses, etc. That's why you use a UART with a buffer – it takes virtually the same time to read 128 bytes out the buffer during one interrupt, as to read 1 byte from the buffer during the interrupt. So if you've set your UART to give an interrupt every 100 characters, you get an interrupt every ms and read out a block of 100 characters at a time.

Re: Linux serial port dropping bytes

That depends on your CPU speed. Within the interrupt, you have to handle something like:

```
REPEAT
test for more in FIFO
take one, stuff in buffer, while checking for buffer full.
test for overflow or other errors.
if any, call appropriate handler
UNTIL FIFO empty
clear interrupt system
rearm interrupt
exit
```

Note that some operations will require several accesses to the UART. Those will eat up time. They will be much slower than on-chip memory access.

This can be surprisingly slow. On a recent project I used an STR9 ARM MCU with the onboard UARTs as well as an external Exar UART. On the Exar UART one could read the number of characters available in the RX FIFO, on the MCU uarts one can only check for character available, or FIFO full.

So with the EXAR one could:

```
read number of chars available
repeat
read char
if buffer not full stuff into buffer
until chars read.
```

This turned out to be 5x faster than with the onboard UARTs where one had to check the FIFO not empty flag every time.

On a 50MHz ARM9 it took about 25us per 16 characters having to do it the way CBFalconer described it, while it only took about 5us per 16 characters where one could read how many chars were in the Rx FIFO.

Regards
Anton Erasmus

.