

Re: 8ms Timer for serial port access

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.apps/2003-10/0589.html>

From: Grant Edwards (grante_at_visi.com)

Date: 10/29/03

Date: 29 Oct 2003 00:54:54 GMT

In article <BBC476FC.1CA0%jens.schumache@gmx.net>, Jens Schumacher wrote:

>> *Do not try to teach the kernel that 24 bytes means something
>> special to you. It doesn't care. If you get 8 bytes,
>> wonderful. You'll get 16 more later. If you get 36 bytes,
>> fine, process 24 of them now and save 12 for the next time you
>> get some data.*
>
> *I meet some Problems when I tried this. I tried to read the
> data and checked how many bytes were read. When I had less
> than 24 I read again and asked for the missing bytes. But
> somehow some bits were corrupt and gave me wrong numbers.*

Then the baud rate is wrong, or the parity is wrong, or you're out-of-sync with the frame boundaries, or something else is wrong.

The serial driver has a 4KB receive FIFO. It's not going to loose data unless you stop reading, and it's not going to corrupt bits unless you have the serial port set incorrectly.

> *This is why I checked the bytes in the buffer first and when
> there are 24 bytes I read them.*

You're making this way harder than it really is. Just call read() until you have 24 bytes and then process them. If you're having trouble syncing up with the incoming data stream when the program starts, then solve that problem somehow in the processing of the data. You're not going to be able to use the 9ms periodicity of the data to sync on unless you do a lot of work in kernel-mode.

>> *No, you don't. You read the data when it's ready, not at some
>> magic time. You can use 'poll' to tell when there's data ready
>> or you can use a blocking read. Here's what you're doing:*
>>
>> *1) Wait X time
>> 2) Read Y bytes*

>> 3) Repeat.
>>
>> Here's what you should do:
>>
>> 1) Wait X time or block in 'poll' until data is available.
>> 2) Read however many bytes there are.
>> 3) Repeat.

When he said Wait X time, I presume he meant as a way of detecting when the data stops -- so that your program doesn't sit forever.

> 1) Sounds reasonable to me...but how to block it. Sorry maybe a stupid
> question.

By default, a read() will block if there is no data available. If you want to have a timeout on the blocking, use select() or poll().

> 2&3) I meet a problem described above.

Then there's something else wrong.

>>> I just need the data from the serial when a whole frame
>>> arrives every 8ms and need to process the data directly. I
>>> never thought this would be a problem on a modern system.
>>
>> It's not a problem, you'd just trying to teach the kernel that
>> 8 milliseconds and 24 bytes are special. It doesn't care and
>> so won't do what you want it to do. Just read however much
>> data it has whenever it has it and you'll be fine.
>>
>> You can even do this:
>>
>> 1) Wait 8 milliseconds (though it will probably really be ten)
>> 2) Read as many bytes as are ready without blocking.
>> 3) Process as many complete frames as you now have saving any leftover
>> bytes for the next pass.
>
> Then I will have another delay before I can process the data.
> I try to avoid any delay like this.

Don't delay. Don't call usleep(). Just call select/poll to wait for data or call read() in blocking mode (which is the default). Then just process the data stream as it arrives.

--
Grant Edwards

grante
at
visi.com

Yow! Make me look like
LINDA RONSTADT again!!