

Re: Serial Communications – Lost Event

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2005-11/msg00225.htm>

- *From:* "Alexander Grigoriev" <alegr@xxxxxxxxxxxxxx>
 - *Date:* Wed, 9 Nov 2005 20:26:53 -0800
-

"Completion of write" is a moment when all data from the app buffer is consumed by the driver. The driver can keep the data in its internal buffer while it's being sent, though, after the write is completed.

You can use `WaitCommEvent/EV_TXEMPTY`, to detect when the last byte is sent to the hardware, but again, it may still be held in FIFO.

"krisk" <krisk@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message <news:1CED0C92-C539-47DD-B8AB-4E80AD55F5D9@xxxxxxxxxxxxxxxxxxxx>

> Hi All,
>
> Thanks for your replies. Didn't mean to sound like I was deriding Windows
> or anything (I really do think it's wonderful, just a bit difficult to
> code
> for sometimes..lol).
>
> What I'm trying to create is a com port monitor that works well with the
> UI
> messaging loop. I want incoming data to result in a message posted to my
> UI. I also want a message posted when outgoing data has been transmitted,
> so
> that I can monitor response times. My immediate application is for a
> bootloader, but I'd like to be able to re-use the code later for real-time
> serial port monitoring and test applications.
>
> So I don't want the communications code to serialize reads and writes
> behind
> the scene and I don't want reads and writes affecting the timing of each
> other, so they need to operate completely independently of each other. I
> have not seen any examples on the web that support this.
>
> Jochen, regarding your comment:
>
>> No. It just signals if there is enough space in the drivers buffer to
>> receive new data. Currently there is `_no_` sign which will show you that
>> data was sent...
>
> The WriteFile API documentation states:

Re: Serial Communications – Lost Event

>
> "The event specified in the OVERLAPPED structure is set to the signaled
> state upon completion of the write operation."
>
> I guess I'm assuming that 'completion of the write' operation in the
> context
> of a serial port means when all of the data has been transferred. Is this
> not
> the case?
>
> Still, I'm wondering if I'm violating some undocumented internal OS
> constraints by having two different overlapped I/O events active on the
> same
> port handle. Is this a legit thing to do? (BTW, I've also tried
> duplicating
> the port handle and using two threads, but I run into the same problem).
>
> Oh, and I am using manual reset events.
>
> Regards,
>
> Kris
>

• **References:**

- ◆ **[Re: Serial Communications – Lost Event](#)**
 ◇ *From:* Jochen Kalmbach [MVP]

- Prev by Date: **[Re: Program Name From Handle](#)**
- Next by Date: **[Re: Windows Xp Unique ID](#)**
- Previous by thread: **[Re: Serial Communications – Lost Event](#)**
- Next by thread: **[Re: Serial Communications – Lost Event](#)**
- Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**