

Re: DoModal isn't reentrant but failure mode could be improved

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2005-12/msg01766.html>

- *From:* Joseph M. Newcomer <newcomer@xxxxxxxxxxxxx>
 - *Date:* Thu, 22 Dec 2005 16:28:02 -0500
-

See below...

On Tue, 20 Dec 2005 10:45:19 +0900, "Norman Diamond" <ndiamond@xxxxxxxxxxxxxxxxx> wrote:

>"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxxx> wrote in message

>news:1o9cq1919ki9e0046kug7bru64qai8ig4s@xxxxxxxxxxxx

>> See below...

>> On Mon, 19 Dec 2005 11:28:43 +0900, "Norman Diamond"

>> <ndiamond@xxxxxxxxxxxxxxxxx> wrote:

>>>"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxxx> wrote in message

>>>news:uut6q15ivvun043k81tpahov96dp6u516@xxxxxxxxxxxx

>>>>

>>>> so to call DoModal twice, it means you have not declared the variable as

>>>> a local variable in a function.

>>>>

>>>>I see what you mean. The dialog should be newly created every time it's

>>>>going to be displayed, instead of being created once at program start and

>>>>then every time assigning two of its CString members before calling

>>>>DoModal.

>>>>

>>>> Correct. The cost of creating the CDialog is so infinitesimal that there

>>>> is no reason it would be created at program startup. You save nothing

>>>> important.

>>>>

>>>>OK.

>>>>

>>>>Yes this would avoid the present behaviour that uses 100% of the CPU and

>>>>renders the program non-responding. As well, if it's really appropriate

>>>>for the application to have two (or more) modal dialog boxes on the

>>>>screen, it will be able to do so.

>>>>

>>>> This is a very odd concept. What does it mean to have two modal dialogs

>>>> up at the same time? While a given dialog invoked by DoModal can itself

>>>> invoke a second dialog,

>>>>

>>>>No, the parent creates a second modal dialog with the same type as the first

>>>>one. The parent receives a Windows Message that causes it to inform the

>>>>user that a serial overrun (or other error) occurred. The reason such a

>>>>Windows Message reaches the parent even while the first modal dialog is

Re: DoModal isn't reentrant but failure mode could be improved

>still on the screen is that it's not a keyboard or mouse input message, and
>CDialog's message loop retrieves it and dispatches it.
>
>>>The unexpected interactions cannot be designed out of existence, but the
>>>responses can be designed to avoid attempting reentrant calls to DoModal.
>>
>> There are a lot of problems when multiple modal dialogs get involved.
>> [...] Example: closing a dialog enables the parent of the dialog. What
>> is the system going to do when you close one of the modal dialogs but not
>> another? I suspect that nothing good is going to happen.
>
>Ack. It's a good thing that I settled on displaying modeless dialog boxes
>for situations that were previously doing reentrant calls to DoModal.
>
>> Serial port overruns can be avoided by increasing the internal buffer size
>> for the serial port.
>
>I've seen a few web pages recommend reducing the internal receive buffer
>size from 14 to 8. I can't quite figure this out. Temporarily I seemed to
>understand the idea of trying to make the UART send an additional interrupt
>when the buffer's 8th byte gets filled (or maybe on every byte) but we still
>really want the UART to continue buffering while it's still necessary.
>Later I figured out that if video drivers or other unknown parts of Windows
>XP are masking the interrupt for more than 1ms, it's not going to matter how
>aggressively the UART asserts the interrupts. By private e-mail someone
>pointed me to some USB-serial adapters that have 384-byte receive buffers,
>and this does seem to be helping.

The on-chip FIFO stores bytes as they come in. The interrupt threshold can be manipulated to interrupt early or late (the typical FIFO size is 16), but taking the interrupt doesn't stop bytes from flowing into the FIFO while the processing continues.

At 115200, character time is about 86us, FIFO time for overflow is about 1.3ms. At that point, you're close to the 1ms limit you're concerned with. I've run perfmon, and seen interrupts peaking at 1000/sec, so at 10us there could be a fairly large number of interrupts preempting the serial port. Unfortunately, Windows currently does not allow you to set interrupt priorities.

>
>> I've done a lot of high-speed serial communication and I've never seen a
>> data overrun, even at 115200.
>
>I have.
>
>> If there's something that masks interrupts for > 1ms, you have a very bad
>> driver in the OS.
>
>Agreed. Several people have posted opinions that video drivers are a big
>cause of this.
>
>> Popping up a dialog box each time there's a serial overrun is bad

Re: DoModal isn't reentrant but failure mode could be improved

>> practice. You could, if they happen as often as you think, generate
>> dozens of these. There are few things more annoying than coming in after
>> lunch and having to dismiss two dozen message boxes
>
>Well, the original code wasn't expecting the possibility that more than one
>could be detected in that short a time. That's how this thread got started.
>And again it seems fortunate that I settled on a non-modal dialog for these
>kinds of occurrences, and yes I do add each as it occurs to a list in one
>non-modal dialog.
>
>> In fact, if there's even one [modal dialog], it is a Really Bad Idea,
>> because it steals the focus of attention from the user at inappropriate
>> times.
>
>The intention of the modal dialog was to ask the user from what point the
>user wants to resume or retry.
>
>Consider an example which fortunately isn't the kind of device we're dealing
>with this time, but suppose it had mechanical moving components. And
>suppose we didn't wait for the user to choose what next action to perform,
>suppose we just proceeded to send the next command to the device without
>knowing what the device's response was to the previous command (because of
>overrun). We might lose clients in a somewhat more violent manner than you
>were thinking of.

This is why devices that have real external manifestations have more robust protocols. For example, I work with embedded HVAC technology. If we lose a packet due to some communication error, we have to resync and ask the device what state it is in, before sending the next command. The idea here is that we have a distributed FSM, where the real FSM is in the remote device and the FSM is modeled locally so we know what commands to send. It is critical that we keep these two models in sync, so a communication loss means that I have to do some kind of query to see what the state really is. So no issue about wondering what to do next arises. These devices actually change external state, turning pumps and fans on and off, opening and closing vents, etc., so the external state really matters. In some cases, the resync simply meant resending the command because if it was already executed, the second send was redundant. (The protocol has no "step one unit" or "toggle state" commands; everything is absolute sets, so redundant requests are harmless). Each device also had a "failsafe reset" mode; if we were really in trouble, we could either reset the device and start over, or otherwise cause it to enter a mode in which nothing bad could happen. I didn't design the protocol, by the way; it was designed by an industry consortium, and designed to be robust. Lost messages should not result in problems, and requiring a user interaction generates yet more problems; for example, while it may be useful to know what to do when there was one error, what is the result of the cascaded errors? And if there is one error, why does anything else matter? Robustness in protocols is a serious design problem. I have to deal with this all the time, including an embedded mass spectrometer, where we do things like ask it to enter a state, then confirm that it has entered the state, and ask it to leave the state, and verify that it left the state; if we get certain states that could result in long-term malfunction, we force a reset and that will definitely revert to a safe state.

I see this all the time in certain devices (I teach device drivers); the engineer who

Re: DoModal isn't reentrant but failure mode could be improved

designs the device assumes perfect communication, and sometimes there isn't even an ack/nak protocol on messaging. Probably 80% of my work in the last 15 years has been on various remote embedded devices, and protocol design is one of the most critical aspects of the design. I once found a protocol which had the property that when I built a device simulator (because they couldn't get me a device) I added a feature to force a NAK, and I went into an infinite messaging loop. I studied my code, and realized that I'd built my code exactly according to their protocol. I pointed this out to them. They tried it and indeed, the protocol was defective. "But we can't fix it", they said, "One of our customers has burned this into ROM on embedded chips, and there are hundreds of thousands of them out there". What was bad was that there was no way to do sane recovery, although I put a recovery component in. It would break out of the message back-and-forth loop about 80% of the time (think "stateful packet inspection", a phrase I didn't know in 1991). It is hard to get protocols right.

Joseph M. Newcomer [MVP]

email: newcomer@xxxxxxxxxxxxx

Web: <http://www.flounder.com>

MVP Tips: http://www.flounder.com/mvp_tips.htm

• *Follow-Ups:*

- ◆ ***Re: DoModal isn't reentrant but failure mode could be improved***
◇ *From:* Norman Diamond

• *References:*

- ◆ ***DoModal isn't reentrant but failure mode could be improved***
◇ *From:* Norman Diamond
- ◆ ***Re: DoModal isn't reentrant but failure mode could be improved***
◇ *From:* Joseph M . Newcomer
- ◆ ***Re: DoModal isn't reentrant but failure mode could be improved***
◇ *From:* Norman Diamond
- ◆ ***Re: DoModal isn't reentrant but failure mode could be improved***
◇ *From:* Joseph M . Newcomer
- ◆ ***Re: DoModal isn't reentrant but failure mode could be improved***
◇ *From:* Norman Diamond

- Prev by Date: ***Re: Making sure data is saved on closing application***
- Next by Date: ***Re: Deleting CButton***
- Previous by thread: ***Re: DoModal isn't reentrant but failure mode could be improved***
- Next by thread: ***Re: DoModal isn't reentrant but failure mode could be improved***
- Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***