

Re: Console app freezes

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.compactframework/2004-12/1250.htm>

From: Michael--J (*MichaelJ_at_discussions.microsoft.com*)

Date: 12/20/04

Date: Mon, 20 Dec 2004 15:47:06 -0800

My app uses OpenNETCF.org's serial comms code with a few modifications in their Port.cs class. This class simply sets up a thread called `CommEventThread` that continually loops the associated com port for data. However, I had to modify this to reflect the modifications I made to the lower level serial drivers. The serial drivers were changed such that it suited the data it was processing. The data was basically composed of packets separated by parity errors. I couldn't use a parity error event to distinguish between packets because the app couldn't keep up with the number of parity error events being raised. Instead, I modified the serial driver such that it places an escape `0x00` character in the receive stream whenever it detected a parity error and my app simply needed to locate this `0x00` character. To distinguish between an escape `0x00` character between an actual data `0x00`, I replaced the data one with a 6-byte signature which my higher level app also searches for. So in pseudo code, this is how the `CommEventThread` in Port.cs looks like:

```
private void CommEventThread()
{
    while(hPort != invalidHandle)
    {

        // Wait for a comm event to take place;
        WaitForCommEvent();

        if(error event)
        {
            // Handle error;
        }

        if(data received)
        {
            do
            {
                ReadFile() to rxFIFO buffer;
            }while(bytesread > 0)
        }
    }
}
```

```
while(rxFIFO.Count > 0)
{
    byte u = rxFIFO.Dequeue();

    if(u == any byte)
    {
        // Add ~u™ to progressBuffer ArrayList
        progressBuffer.Add(u);
    }

    if(u == ~f0™)
    {
        // Raise ParityError() event and clear progressBuffer;
        ParityError();
        progressBuffer = new ArrayList();
    }

    if(u is part of 6–byte signature)
    {
        // Take note of it;
    }

    if(u is last byte of 6–byte signature)
    {
        // Insert ~f0™ in the data;
        progressBuffer.Add(u);
    }
}
}
```

I also added the method `ReadProgressBuffer()` in `Port.cs` which gets the current `progressBuffer` contents when a parity error event is raised. This will be called by my main app.

```
public byte[] ReadProgressBuffer(){
    return (byte[])progressBuffer.ToArray(typeof(byte));
}
```

In my main code, I instantiate 4 port objects which means that I introduce 4 `CommEventThreads` executing the above code in non–stop looping. My main also handles the `ParityError` events raised by each port by creating 4 identical event handlers that process the data by firstly calling `portX.ReadProgressBuffer()`. Below is the pseudo code:

```
private void portX_ParityError()
{
    byte[] packet = portX.ReadProgressBuffer();

    // Process the byte array here;
    // This predominantly involves packet validation –
```

```
// CRC checks, and updating a set of global data;  
}
```

When this handler returns, the CommEventThread continues by clearing the progressBuffer and repeating the whole process.

In addition to this, I have 3 timers, TCP/IP stuff, and log keeping (as described in my first post). Would it definitely be a CPU overload issue? If you need more information regarding my app or any of its code, please feel free to ask. Thanks again.

""Ilya Tumanov [MS]"" wrote:

- > *This behavior seem to be a result if excessive CPU load.*
- > *Do you have some tight loops anywhere? Say, waiting for something?*
- > *Do you know what your app will do in case more data is coming from serial*
- > *port(s) than can be processed?*
- > *Could it wait for available buffer in a loop without sleep() in it?*
- > *That would cause the "serial" thread to use 100% CPU, which would slow down*
- > *all other thread, which would slow down data processing, which would cause*
- > *"serial" thread to wait... and so on. As you remove incoming data, your app*
- > *would eventually process data and unlock itself.*
- >
- > *Best regards,*
- >
- > *Ilya*
- >