

RE: Read Serial Ports With Visual Basic .NET 2003

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.remoting/2006-01/msg00018.html>

- *From:* Tolis Carpenter <TolisCarpenter@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Thu, 5 Jan 2006 14:20:03 -0800
-

Thank you for your help Robert

For me is great helper because I have to setup my software somewhere in mounds with snow and cold and I have to be sore.

There is an easy solution they say for reading the value from text file, <maybe if you read it with your one eyes> that weight machine but with different language platform that they don t known what is or what kind of software is that, and in different boundaries how known what happen with file share or streams that lock this file or threads so I believe is to be sore for me and not depend in Integration good will

"Robert" wrote:

```
> This is a class for RS232 – is working very good
>
> using System;
> using System.Runtime.InteropServices;
> using System.Text;
> using System.Threading;
>
> namespace RS232
> {
> /// <summary>
> /// Obiekt odpowiedzialny z obsBuge poBaczenie RS232 w komputerze
> /// </summary>
> ///
>
> public delegate void dataReceived(RS232 Source, Byte[] DataBuffer);
> public delegate void txCompleted(RS232 Source);
> public delegate void odczyt(byte[] dane);
> public delegate void blad(string wiadomosc);
>
> public class RS232 //: System.IDisposable
> {
>
> public delegate void commEvent(RS232 Source, EventMasks Mask);
>
> public event dataReceived DataReceived;
```

RE: Read Serial Ports With Visual Basic .NET 2003

```
> public event txCompleted TxCompleted;
> public event odczyt Odczyt;
> public event blad Blad;
>
> private Int32 FzhRS = -1; // Handle do Com Port
> private int FzPort = 1; // Domy[lny jest COM1
> private Int32 FzTimeout = 50; // Timeout w ms
> private Int32 FzBaudRate = 9600;
> private DataParity FzParity = 0;
> private DataStopBit FzStopBit = 0;
> private Int32 FzDataBit = 8;
> private Int32 FzBufferSize = 512; // Domylny rozmiar bufora 512
> private Byte[] FzRxBuf; // Bufor odczytu
> private Mode FzMode;
> private Boolean FzWaitOnRead;
> private Boolean FzWaitOnWrite;
> private Boolean FzWriteErr;
> private OVERLAPPED FzOverlapped;
> private OVERLAPPED FzOverlappedW;
> private OVERLAPPED FzOverlappedE;
> private Byte[] FzTmpTxBuf; // Tymczasowy buffer ulyty do
> asynchronicznego odczytu Tx
> private Thread FzThreadTx;
> private Thread FsThreadRx;
> private Int32 FzTmpBytes2Read;
> private EventMasks FzMask;
> private Boolean mbDisposed;
> private System.Collections.Queue do_zapisu;
> public int ODSTEP_OD_CZYTANIA = 20;
> public int ILOSC_ZNAKOW_DO_CZYTANIA = 1;
>
> //-----
>
> public enum DataParity
> {
> Parity_None=0,
> Pariti_Odd,
> Parity_Even,
> Parity_Mark
> };
>
> public enum DataStopBit
> {
> StopBit_1 = 1,
> StopBit_2
> };
>
> public enum PurgeBuffers
> {
> RXAbort = 0x2,
> RXClear = 0x8,
```

```

> TxAbort = 0x1,
> TxClear = 0x4
> };
>
> public enum Lines
> {
> SetRts = 3,
> ClearRts = 4,
> SetDtr = 5,
> ClearDtr = 6,
> ResetDev = 7, // Reset urz dzenie je|eli jest to mo|liwe
> SetBreak = 8, // Set the device break line.
> ClearBreak = 9 // Clear the device break line.
> };
>
> [Flags()] public enum ModemStatusBits // Modem Status
> {
> ClearToSendOn = 0x10,
> DataSetReadyOn = 0x20,
> RingIndicatorOn = 0x40,
> CarrierDetect = 0x80
> };
>
> public enum Mode // Working mode
> {
> NonOverlapped,
> Overlapped
> };
>
> [Flags()] public enum EventMasks // Comm Masks
> {
> RxChar = 0x1,
> RXFlag = 0x2,
> TxBufferEmpty = 0x4,
> ClearToSend = 0x8,
> DataSetReady = 0x10,
> ReceiveLine = 0x20,
> Break = 0x40,
> StatusError = 0x80,
> Ring = 0x100
> };
>
>
> //-----
>
> [StructLayout(LayoutKind.Sequential)] private struct DCB
> {
> public Int32 DCBlength;
> public Int32 BaudRate;
> public Int32 Bits1;
> public Int16 wReserved;

```

RE: Read Serial Ports With Visual Basic .NET 2003

```
> public Int16 XonLim;
> public Int16 XoffLim;
> public Byte ByteSize;
> public Byte Parity;
> public Byte StopBits;
> public Byte XonChar;
> public Byte XoffChar;
> public Byte ErrorChar;
> public Byte EofChar;
> public Byte EvtChar;
> public Int16 wReserved2;
> }
>
> [StructLayout(LayoutKind.Sequential)] private struct COMMTIMEOUTS
> {
> public Int32 ReadIntervalTimeout;
> public Int32 ReadTotalTimeoutMultiplier;
> public Int32 ReadTotalTimeoutConstant;
> public Int32 WriteTotalTimeoutMultiplier;
> public Int32 WriteTotalTimeoutConstant;
> }
>
> [StructLayout(LayoutKind.Sequential)] private struct COMMCONFIG
> {
> public Int32 dwSize;
> public Int16 wVersion;
> public Int16 wReserved;
> public DCB dcbx;
> public Int32 dwProviderSubType;
> public Int32 dwProviderOffset;
> public Int32 dwProviderSize;
> public Byte wcProviderData;
> }
>
> [StructLayout(LayoutKind.Sequential)] public struct OVERLAPPED
> {
> public Int32 Internal;
> public Int32 InternalHigh;
> public Int32 Offset;
> public Int32 OffsetHigh;
> public Int32 hEvent;
> }
>
>
> //-----
>
> private const int PURGE_RXABORT = 0x2;
> private const int PURGE_RXCLEAR = 0x8;
> private const int PURGE_TXABORT = 0x1;
> private const int PURGE_TXCLEAR = 0x4;
> private const uint GENERIC_READ = 0x80000000;
```

```

> private const int GENERIC_WRITE = 0x40000000;
> private const int OPEN_EXISTING = 3;
> private const int INVALID_HANDLE_VALUE = -1;
> private const int IO_BUFFER_SIZE = 1024;
> private const Int32 FILE_FLAG_OVERLAPPED = 0x40000000;
> private const Int32 ERROR_IO_PENDING = 997;
> private const Int32 WAIT_OBJECT_0 = 0;
> private const Int32 ERROR_IO_INCOMPLETE = 996;
> private const Int32 WAIT_TIMEOUT = 0x102;
> private const UInt32 INFINITE = 0xFFFFFFFF;
>
> //-----
> [DllImport("kernel32.dll")] private static extern Int32 SetCommState(Int32
> hCommDev, ref DCB lpDCB);
> [DllImport("kernel32.dll")] private static extern Int32 GetCommState(Int32
> hCommDev, ref DCB lpDCB);
> [DllImport("kernel32.dll")] private static extern Int32
> BuildCommDCB(String lpDef, ref DCB lpDCB);
> [DllImport("kernel32.dll")] private static extern Int32 SetupComm(Int32
> hFile, Int32 dwInQueue, Int32 dwOutQueue);
> [DllImport("kernel32.dll")] private static extern Int32
> SetCommTimeouts(Int32 hFile, ref COMMTIMEOUTS lpCommTimeouts);
> [DllImport("kernel32.dll")] private static extern Int32
> GetCommTimeouts(Int32 hFile, ref COMMTIMEOUTS lpCommTimeouts);
> [DllImport("kernel32.dll")] private static extern Int32
> ClearCommError(Int32 hFile, Int32 lpErrors, Int32 l);
> [DllImport("kernel32.dll")] private static extern Int32 PurgeComm(Int32
> hFile, Int32 dwFlags);
> [DllImport("kernel32.dll")] private static extern Boolean
> EscapeCommFunction(int hFile, Int64 ifunc);
> [DllImport("kernel32.dll")] private static extern Int32 WaitCommEvent(int
> hFile, ref EventMasks Mask, ref OVERLAPPED lpOverlap);
> [DllImport("kernel32.dll")] private static extern int WriteFile(int hFile,
> Byte[] Buffer, int nNumberOfBytesToWrite, ref int lpNumberOfBytesWritten, ref
> OVERLAPPED lpOverlapped);
> [DllImport("kernel32.dll")] private static extern int ReadFile(int hFile,
> Byte[] Buffer, int nNumberOfBytesToRead, ref int lpNumberOfBytesRead, ref
> OVERLAPPED lpOverlapped);
> [DllImport("kernel32.dll")] private static extern int
> CreateFile([MarshalAs(UnmanagedType.LPStr)] String lpFileName, int
> dwDesiredAccess, int dwShareMode, int lpSecurityAttributes, int
> dwCreationDisposition, int dwFlagsAndAttributes, int hTemplateFile);
> [DllImport("kernel32.dll")] private static extern int CloseHandle(int
> hObject);
> [DllImport("kernel32.dll")] private static extern int FormatMessage(int
> dwFlags, int lpSource, int dwMessageId, int dwLanguageId,
> [MarshalAs(UnmanagedType.LPStr)] String lpBuffer, int nSize, int Arguments);
> [DllImport("kernel32.dll")] private static extern Boolean
> GetCommModemStatus(Int32 hFile, ref Int32 lpModemStatus);
> [DllImport("kernel32.dll")] private static extern Int32 CreateEvent(Int32
> lpEventAttributes, Int32 bManualReset, Int32 bInitialState,

```

RE: Read Serial Ports With Visual Basic .NET 2003

```
> [MarshalAs(UnmanagedType.LPStr)] String lpName);
> [DllImport("kernel32.dll")] private static extern Int32 GetLastError();
> [DllImport("kernel32.dll")] private static extern Int32
> WaitForSingleObject(Int32 hHandle, Int32 dwMilliseconds);
> [DllImport("kernel32.dll")] private static extern Int32
> GetOverlappedResult(Int32 hFile, ref OVERLAPPED lpOverlapped, ref Int32
> lpNumberOfBytesTransferred, Int32 bWait);
> [DllImport("kernel32.dll")] private static extern Int32 SetCommMask(Int32
> hFile, Int32 lpEvtMask);
> [DllImport("kernel32.dll")] private static extern Int32
> FormatMessageA(Int32 dwFlags, Int32 lpSource, Int32 dwMessageId, Int32
> dwLanguageId, StringBuilder lpBuffer, Int32 nSize, Int32 Arguments);
>
> //-----
>
>
> //-----
>
> public RS232()
> {
> this.do_zapisu = new System.Collections.Queue();
> }
>
> ~RS232()
> {
> if(this.FzhRS!==-1) this.Close();
> GC.SuppressFinalize(this);
> }
>
> public virtual int Port
> {
> get
> {
> return this.FzPort;
> }
> set
> {
> this.FzPort = value;
> }
> }
>
> public virtual int Timeout
> {
> get
> {
> return this.FzTimeout;
> }
> set
> {
> this.FzTimeout = (value==0?500:value);
> this.SetTimeout();
> }
```

```
> }
> }
>
> public virtual DataParity Parity
> {
> get
> {
> return this.FzParity;
> }
> set
> {
> this.FzParity = value;
> }
> }
>
> public virtual DataStopBit StopBit
> {
> get
> {
> return this.FzStopBit;
> }
> }
```

• **References:**

◆ **[RE: Read Serial Ports With Visual Basic .NET 2003](#)**

◇ From: Robert

- Prev by Date: **[Custom Sink with Compression and Encryption](#)**
- Next by Date: **[Re: How to reference the server with MBV ?](#)**
- Previous by thread: **[RE: Read Serial Ports With Visual Basic .NET 2003](#)**
- Next by thread: **[Windows service talking to a windows program.](#)**
- Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**