Serial Communication with VB.Net

One of the thing I really miss while using VB.Net is the lack of serial communication support. Yes, you can use VB6's MsComm32.ocx but if you, like me, have installed VB.Net on a separate hard disk you may experience some problems installing and using it in design mode.

Don't forget moreover that if you use that control, you must register it on user machine when you deploy your application, loosing the 'XCopy installation mode' provided by VB.Net.

With this class you can easily use serial communication using native VB.Net and API features.

Before describing you how to use the class, let me point out that I've just written it as a demonstration of VB.Net interface to serial communication, you'd probably need to customize it for your special 'Rs232' needs.

Initializing and Opening the Com port

Create an instance of CRs232 then set COM parameters before invoking the Open method

Here's an example:

```vbnet
Dim moRS232 as New Rs232()
With moRs232
  .Port = 1                        '// Uses COM1
  .BaudRate = 2400                  '// 2400 baud rate
  .DataBit = 8                       '// 8 data bits
  .StopBit = Rs232.DataStopBit.StopBit_1     '// 1 Stop bit
  .Parity = Rs232.DataParity.Parity_None      '// No Parity
  .Timeout = 500                      '// 500 ms of timeout admitted to get all required bytes
End With
//' Initialize and Open
moRS232.Open ()
```

You can, optionally control the state of DTR/RTS lines after the Port is open

```vbnet
//' Set state of RTS / DTS
moRS232.Dtr = True
moRS232.Rts = True
```

In case of an error/problems an exception is raised, so I suggest you to enclose the code within a Try...Catch block.

Transmitting data to COM Port

The class has 2 buffers one for Tx and one for Rx, to transmit data just set the TxData property with the informations you wish to send then invoke the Tx method.

```vbnet
example:
moRS232.Write(txtTx.Text)
```

Receiving data from COM Port

Just invoke the Rx method passing it the number of bytes you want to read from COM port, then read the Rxdata property.

```vbnet
example:
moRS232.Read(10)                      '// Gets 10 bytes from serial communication buffer
Dim sRead as String=moRs232.InputStreamString
```

Please note that thread is blocked for the period of time set by Timeout property and that in case the class cannot read all required bytes from COM buffer a Timeout exception is raised.

If the number of bytes to read is omitted, the class assumes 512 bytes have to be read from buffer.

The class is very simple and surely misses some error control, but as prefaced I just wanted to give you an example of what you can do with VB.Net without resorting to ocx'es or other 3rdy parties controls

More details

The zip file that you can download here includes also a small Windows Form example that can help you understand how to use my class.
How to use the sample

- Verify that COM1 or COM2 are not in use (sample uses COM1 or COM2 but class can handle more than 9 ports...)
- Select default timeout (in milliseconds) and Baudrate (other parameters like parity... are fixed inside code)
- Press **Open COM Port**
- Type the string you wish to send to device (e.g ATDT12345) and press **Tx**
- Verify on target device that bytes are sent.
- If you want to read some bytes from COM port, type the number of char you wish to read into **Bytes to read** textbox and press **Rx**
- Received data will be displayed inside Received data textbox

If not enough character are received inside allocated timeout, you will have an error and you will see what has been read from serial port.

If you want to test the status of Carrier Detect (CD) or RTS and other lines just select it from Status line combobox and press Check

RTS and DTR checkboxes will allow you to change the status of RTS and DTR lines respectively.

**Automatically receive bytes** allows you to read incoming bytes after you press Tx button simulating a client-server connection.

**Enabling events**
This is the most requested features I got from you and finally, here it is!
Check **Enable events** to start intercepting events coming from serial port (CD, RTS, Ring...) and to get characters as soon as they get into COM port.

Events are signaled by a CommEvent event, and you can use Mask parameter to detect what event has occurred, if you set a number of character into Bytes to read textbox, as soon as those character arrives into serial port you will get a CommEvent that you can use to get received characters.

Here's an extract from sample code:
If (mask And Rs232.EventMasks.RxChar) > 0 Then
Dim s as String = source.InputStreamString  '// Here you have a string with received characters
End If

Please note that in order to get data asynchronously you have to set EnableEvents property to true (see code on sample Form)

Enjoy!
Corrado Cavalli
corrado@mvps.org

You can download latest version here (example requires Visual Studio 2003)
Previous version is available here

To load my project on Visual Studio 2002

- Create a new windows forms project project
- Remove Form1
- Add Rs232 class and form you have on zip file
- Set added form as startup form

All code is freely redistributable, just let me know if you enjoy using it....

Project History
1st Public release Beta2 (10/08/2001)

Rev.1 (28.02.2002)
1. Added ResetDev, SetBreak and ClearBreak to the EscapeCommFunction constants
2. Added the overloaded Open routine.
3. Added the modem status routines, properties and enum.
4. If a read times out, it now returns a EndOfStreamException (instead of a simple Exception).
5. Compiled with VS.Net final

Rev.2 (01.03.2002)
Added Async support

Rev.3 (07.04.2002)
Minor bugs fixed

Rev.3a (05/05/2002)
Fixed BuildCommmDCB problem

Rev.4 (24/05/2002)
Fixed problem with ASCII Encoding truncating 8th bit

Rev.5 (27/05/2002)
Added IDisposable / Finalize implementation

Rev.6 (14/03/2003)
Fixed problem on DCB fields Initialization

Rev.7 (26/03/2003)
Added XON/XOFF support

Rev.8 (12/07/2003)
Added support to COM port number greater than 4

Rev.9 (16/07/2003)
Added CommEvent to detect incoming chars/events(!)
Updated both Tx/Rx method from Non-Ovelapped to Overlapped mode
Removed unused Async methods and other stuff.

Rev.10 (21/07/2003)
Fixed incorrect character handling when using EnableEvents()

Rev.11 (12/08/2003)
Fixed some bugs reported by users

Rev.12 (01/09/2003)
Removed AutoReset of internal buffers and added PurgeBuffer() method
Rev.13 (02/09/2003)
Update internal stuff now using Win32Exception instead of GetLastError+FormatMessage APIs

Rev.14 (14/09/2003)
Added IsPortAvailable() function (thanks to Tom Lafleur for the hint)
Revised some API declaration
Fixed some problems with Win98/Me OS (thanks to Alex Komissarov for the feedback)

Rev.15 (24/09/2003)
Fixed bug introduced on rev.14 (sorry for that...)

Rev.16 (16/10/2003)
Added SetBreak/ClearBreak methods for sending break signal over the line.

Rev.17 (02/11/2003)
Fixed incorrect field on COMMCONFIG Structure.

Rev.18 (03/03/2004)
Fixed bug causing troubles accessing already in use ports (folks, thanks for the feedback!)

Rev.19 (08/04/2004)
Fixed bug on DTR property (thanks to Charles-Olivier Théroux)

Rev.20 (12/07/2004)
CommEvent is no more raised on a secondary thread (please note that this is valid only if event handler is not associated with a static method)
pEventsWatcher now uses a background thread

Rev.21 (24/10/2004)
Fixed EscapeCommFunction declaration
Fixed incorrect Pariti enum entry

Rev.22 (05/03/2005)
Fixed memory leak causing random program termination without any message.
Thanks to Ralf Gedrat for testing this scenario.

Rev.23 (05/04/2005)
Fixed bug DisableEvents not working bug (Thanks to Jean Bédard)

Rev.24 (20/04/2005)
Fixed memory leak on Read() method
Added InBufferCount property
IsPortAvailable method is now shared (Thanks to Jean-Pierre ZANIER for the feedback)

back to home page