Serial Port Using Visual Basic .NET and Windows

Introduction

The serial (COM) port is one of the simplest ways to communicate between a PC and a microcontroller circuit. Most microcontrollers have hardware serial ports and most microcontroller compilers have built-in functions to read from and write to the hardware port.

Hardware serial ports with their 9-pin D connectors have disappeared from laptop and desktop computers, but are easily produced with a low-cost USB-to-serial cable adaptor. For embedded systems, a common approach is to add a FT232R USB-to-serial chip to the circuit so that the hardware connects to the PC though USB. Another option is to add a USB-to-serial module, for example the UM232R by FTDI or the USBMOD3 from DLP Design. These modules add glue circuits and a USB connector to the converter chip for an easy-to-use self contained solution.

On the PC side, the USB adaptor appears as a virtual serial port that can be accessed by an application program just like a hardware COM port. On the microcontroller side, the adaptor appears as a standard serial port.

If using an adaptor cable, the application circuit needs a convertor chip, for example the Dallas DS275 or the Maxim MAX233 to convert RS-232 serial levels to TTL logic levels.

Circuits and code for serial port connections to a PICmicro are in the ME 8243 Boot Camp exercises.

Checking the Virtual Serial Port Connection

Check that device appears as a COM port on the PC by using the Device Manager. Right-click on My Computer > Properties > Hardware > Device Manager > Ports (COM & LPT). Should see something like PL2303 COM Port (COM6), which means the cable is showing up as COM port number 6. Your port number may be different. The port number will change if you plug the cable into a different USB port on your computer.

For quick access to the Device Manager: Start -> Run -> devmgmt.msc

The simplest way to test a serial connection is to connect to a PC terminal program. In the terminal program, set to use the COM port that connects to the adapter cable. Set the baud rate to the rate specified in the PICmicro program. For PC terminal program, use TeraTerm (ver 4.64 or later, http://en.sourceforge.jp/projects/ttssh2/releases/). Open TeraTerm, attach to serial port at correct baud rate, then save session into teraterm.ini.

Software Examples

Example code in this document was written on the PC side in Visual Basic 2008 Express Edition and on the PIC chip microcontroller side in C using the CCS compiler.

Bare-Minimum VB.net Code

The serial port functions are in the .NET System.IO.Ports library.

At top of form, preceding Module or Class statement, add:

```
Imports System.IO.Ports
```

At top of form, under Class statement, create serial port instance:

```
Private mySerialPort As New SerialPort
```

In code, set properties in a setup procedure that is called at form load time, like this:

```
Private Sub CommPortSetup()
  With mySerialPort
    .PortName = "COM10"
    .BaudRate = 34800
    .DataBits = 8
    .Parity = Parity.None
    .StopBits = StopBits.One
    .Handshake = Handshake.None
  End With
End Sub
```

Open the serial port in the setup procedure and use Try/Catch handling to deal with errors:

```
Try
  mySerialPort.Open()
Catch ex As Exception
  MessageBox.Show(ex.Message)
End Try
```

Sending data is done with the Write or the WriteLine methods.

The Write method writes a string like this:

```
Dim instance As SerialPort
Dim text As String

instance.Write(text)
```

The Write method can also write a byte array like this:

```
Dim instance As SerialPort
Dim buffer As Byte()
Dim offset As Integer
Dim count As Integer

instance.Write(buffer, offset, count)
```

where buffer is the data array, offset is where the write should start (set to 0 to start at the beginning) and count is the number of bytes to write.

The WriteLine method writes the specified string and appends a NewLine (0Ah) value. Use like this:

```
      Dim instance As SerialPort
      Dim text As String

      instance.WriteLine(text)
```

More Sophisticated VB.NET Code

A function to call at form load that initializes a serial port. The Try-Catch surrounds the port open function to flag system errors, for example that the port does not exist.

```
  Private Sub CommPortSetup()
    With mySerialPort
      .PortName = "COM10"
      .BaudRate = 38400
      .DataBits = 8
      .Parity = Parity.None
      .StopBits = StopBits.One
      .Handshake = Handshake.None
    End With
    Try
      mySerialPort.Open()
    Catch ex As Exception
      MessageBox.Show(ex.Message)
    End Try
  End Sub
```

A code snippet to fill a string array with the names of the valid ports

```
    Dim myPortNames() As String
    myPortNames = SerialPort.GetPortNames
```

A procedure to send a frame in cmd,data format out the port.

```
  Private Sub SendSlave(ByVal sendCmd As Byte, ByVal sendData As Byte)
    'send command to slave PICmicro in form of cmd, data

    Dim buffer(2) As Byte
    buffer(0) = sendCmd
    buffer(1) = sendData
    mySerialPort.Write(buffer, 0, 2)
  End Sub
```

To send one byte, use this.

```
    Dim buffer() As Byte = {1}
    mySerialPort.Write(buffer, 0, 1)
```

To read one line (blocking).

```
    Dim returnValue As String
    returnValue = mySerialPort.ReadLine
```

To read one byte (blocking).

```
    Dim returnValue As Integer
    returnValue = mySerialPort.ReadByte
```

To read several bytes. Buffer is where the data is stored, set offset = 0 to start at the beginning, count is the number of bytes to read, returnValue is the number of bytes read.

```
    Dim buffer As Byte()
    Dim offset As Integer
    Dim count As Integer
    Dim returnValue As Integer
    returnValue = mySerialPort.Read(buffer, offset, count)
```

For receiving data at unexpected times, use the DataReceived event. This is a bit tricky because it runs in a different thread and requires a handler. (Virtual Serial Port Cookbook, Chapter 9 for details.) .

Create a procedure for the data received event, like this.

```
  Private Sub mySerialPort_DataReceived(ByVal sender As Object, ByVal e As
SerialDataReceivedEventArgs)
    'Handles serial port data received events
    Dim n As Integer = mySerialPort.BytesToRead
    Dim comBuffer As Byte() = New Byte(n - 1) {}
    mySerialPort.Read(comBuffer, 0, n)
    Console.WriteLine(comBuffer(0))
  End Sub
```

In the form load procedure, add a handler that points the data received event to the name of the procedure that does the work, using this line

```
    AddHandler mySerialPort.DataReceived, AddressOf mySerialPort_DataReceived
```

The handler runs in a different thread, which means it cannot directly access controls on the form. To get around this, use a delegate to pass data to the form.

Add these lines to the public area

```
  Private mySerialPort As New SerialPort
  Private comBuffer As Byte()
  Private Delegate Sub UpdateFormDelegate()
  Private UpdateFormDelegate1 As UpdateFormDelegate
```

Change the data received procedure to look like this

```
  Private Sub mySerialPort_DataReceived(ByVal sender As Object, ByVal e As
SerialDataReceivedEventArgs)
    'Handles serial port data received events
    UpdateFormDelegate1 = New UpdateFormDelegate(AddressOf UpdateDisplay)
    Dim n As Integer = mySerialPort.BytesToRead  'find number of bytes in buf
    comBuffer = New Byte(n - 1) {}  're dimension storage buffer
    mySerialPort.Read(comBuffer, 0, n)  'read data from the buffer

    Me.Invoke(UpdateFormDelegate1) 'call the delegate
  End Sub
```

Here is the function that will get triggered to update the display

```
  Private Sub UpdateDisplay()
    Label2.Text = CStr(comBuffer(0))
  End Sub
```

Odds and Ends

- See MovingBar VB.NET code for receiving 2-byte frame
- See schematic files for Generic PIC Chip circuits for hardware examples and connector pinouts.
- When wiring to a DB-9F connector, wire to pins 2, 3 and 5 only.
- See the Axelson N&V, April 2008 article for retrieving names of serial ports using GetPortNames method.

References

1. Pardue, J., Virtual Serial Port Cookbook.
2. Axelson, J, Serial Port Complete, 2nd ed.
3. Axelson J, Access Serial Ports with Visual Basic.NET, Nuts & Volts, April 2008, p 60.
4. Jan Axelson's web site: http://www.lvr.com/serport.htm
5. Microsoft serial port class documentation: http://msdn2.microsoft.com/library/30swa673(en-us,vs.80).aspx