Serial Port example

```
// Use this code inside a project created with the Visual C# > Windows Desktop > Console Application template.
// Replace the code in Program.cs with this code.

using System;
using System.IO.Ports;
using System.Threading;

public class PortChat
{
    static bool _continue;
    static SerialPort _serialPort;

    public static void Main()
    {
        string name;
        string message;
        StringComparer stringComparer = StringComparer.OrdinalIgnoreCase;
        Thread readThread = new Thread(Read);

        // Create a new SerialPort object with default settings.
        _serialPort = new SerialPort();

        // Allow the user to set the appropriate properties.
        _serialPort.PortName = SetPortName(_serialPort.PortName);
        _serialPort.BaudRate = SetPortBaudRate(_serialPort.BaudRate);
        _serialPort.Parity = SetPortParity(_serialPort.Parity);
        _serialPort.DataBits = SetPortDataBits(_serialPort.DataBits);
        _serialPort.StopBits = SetPortStopBits(_serialPort.StopBits);
        _serialPort.Handshake = SetPortHandshake(_serialPort.Handshake);

        // Set the read/write timeouts
        _serialPort.ReadTimeout = 500;
        _serialPort.WriteTimeout = 500;

        _serialPort.Open();
        _continue = true;
        readThread.Start();

        Console.Write("Name: ");
        name = Console.ReadLine();

        Console.WriteLine("Type QUIT to exit");

        while (_continue)
        {
            message = Console.ReadLine();

            if (stringComparer.Equals("quit", message))
            {
                _continue = false;
            }
            else
            {
```

```
            _serialPort.WriteLine(
                String.Format("<{0}>: {1}", name, message));
        }
    }

    readThread.Join();
    _serialPort.Close();
}

public static void Read()
{
    while (_continue)
    {
        try
        {
            string message = _serialPort.ReadLine();
            Console.WriteLine(message);
        }
        catch (TimeoutException) { }
    }
}

// Display Port values and prompt user to enter a port.
public static string SetPortName(string defaultPortName)
{
    string portName;

    Console.WriteLine("Available Ports:");
    foreach (string s in SerialPort.GetPortNames())
    {
        Console.WriteLine("   {0}", s);
    }

    Console.Write("Enter COM port value (Default: {0}): ", defaultPortName);
    portName = Console.ReadLine();

    if (portName == "" || !(portName.ToLower()).StartsWith("com"))
    {
        portName = defaultPortName;
    }
    return portName;
}
// Display BaudRate values and prompt user to enter a value.
public static int SetPortBaudRate(int defaultPortBaudRate)
{
    string baudRate;

    Console.Write("Baud Rate(default:{0}): ", defaultPortBaudRate);
    baudRate = Console.ReadLine();

    if (baudRate == "")
    {
        baudRate = defaultPortBaudRate.ToString();
    }

    return int.Parse(baudRate);
}
```

```
// Display PortParity values and prompt user to enter a value.
public static Parity SetPortParity(Parity defaultPortParity)
{
   string parity;

   Console.WriteLine("Available Parity options:");
   foreach (string s in Enum.GetNames(typeof(Parity)))
   {
      Console.WriteLine("   {0}", s);
   }

   Console.Write("Enter Parity value (Default: {0}):", defaultPortParity.ToString(), true);
   parity = Console.ReadLine();

   if (parity == "")
   {
      parity = defaultPortParity.ToString();
   }

   return (Parity)Enum.Parse(typeof(Parity), parity, true);
}
// Display DataBits values and prompt user to enter a value.
public static int SetPortDataBits(int defaultPortDataBits)
{
   string dataBits;

   Console.Write("Enter DataBits value (Default: {0}): ", defaultPortDataBits);
   dataBits = Console.ReadLine();

   if (dataBits == "")
   {
      dataBits = defaultPortDataBits.ToString();
   }

   return int.Parse(dataBits.ToUpperInvariant());
}

// Display StopBits values and prompt user to enter a value.
public static StopBits SetPortStopBits(StopBits defaultPortStopBits)
{
   string stopBits;

   Console.WriteLine("Available StopBits options:");
   foreach (string s in Enum.GetNames(typeof(StopBits)))
   {
      Console.WriteLine("   {0}", s);
   }

   Console.Write("Enter StopBits value (None is not supported and \n" +
    "raises an ArgumentOutOfRangeException. \n (Default: {0}):", defaultPortStopBits.ToString());
   stopBits = Console.ReadLine();

   if (stopBits == "" )
   {
      stopBits = defaultPortStopBits.ToString();
   }
```

```
        return (StopBits)Enum.Parse(typeof(StopBits), stopBits, true);
    }
    public static Handshake SetPortHandshake(Handshake defaultPortHandshake)
    {
        string handshake;

        Console.WriteLine("Available Handshake options:");
        foreach (string s in Enum.GetNames(typeof(Handshake)))
        {
            Console.WriteLine("   {0}", s);
        }

        Console.Write("End Handshake value (Default: {0}):", defaultPortHandshake.ToString());
        handshake = Console.ReadLine();

        if (handshake == "")
        {
            handshake = defaultPortHandshake.ToString();
        }

        return (Handshake)Enum.Parse(typeof(Handshake), handshake, true);
    }
}
```