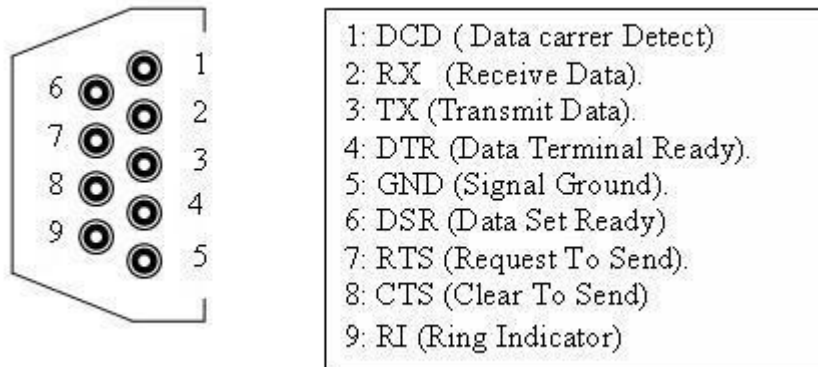


Tutorial Tranceiver data use COM port

Trong ví dụ này, tôi không đi sâu vào lý thuyết cấu trúc cổng COM (các bạn có thể tìm đọc cấu trúc các thanh ghi trong vi UART điều khiển cổng COM), mà hướng dẫn thực hành cụ thể lập trình từng bước từng bước sử dụng Visual C#.NET 2005.

Bố trí chân của cổng COM



Hình 1

Các thông số cổng COM cần lưu ý:

Thông số của cổng COM cần quan tâm là:

1. Địa chỉ cổng (Port Name).
2. Tốc độ baud (baudrate),
3. Số bit trong một khung dữ liệu (data bits),
4. Số bit dừng giữa các khung dữ liệu (stop bit),
5. Bit kiểm tra chẵn lẻ (parity bit).

Thông thường (default) cài đặt : Port Name= COM1 (địa chỉ mã Hexa: 3f8),baudrate = 9600, data bits = 8, stop bit = 1, parity = none.

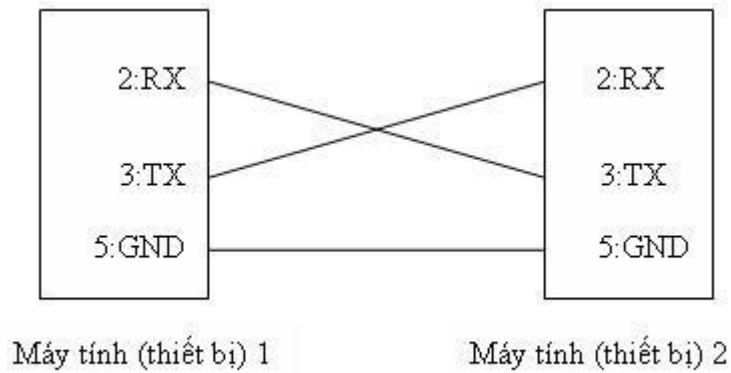
Cơ chế bắt tay cứng (Handshaking):

Có 4 loại :

- None,
- XON/XOFF,
- CTS/RTS,
- CTS/RTS XON/XOFF

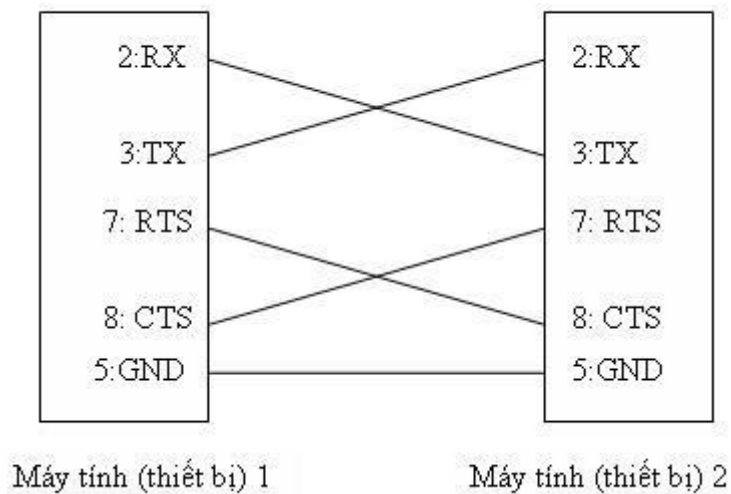
Phụ thuộc vào từng cơ chế bắt tay mà có cách nối dây khác nhau.

Nối dây cho cơ chế NONE và XON/XOFF:



Hình 2

Nối dây cho cơ chế CTS/RTS, CTS/RTS XON/XOFF:

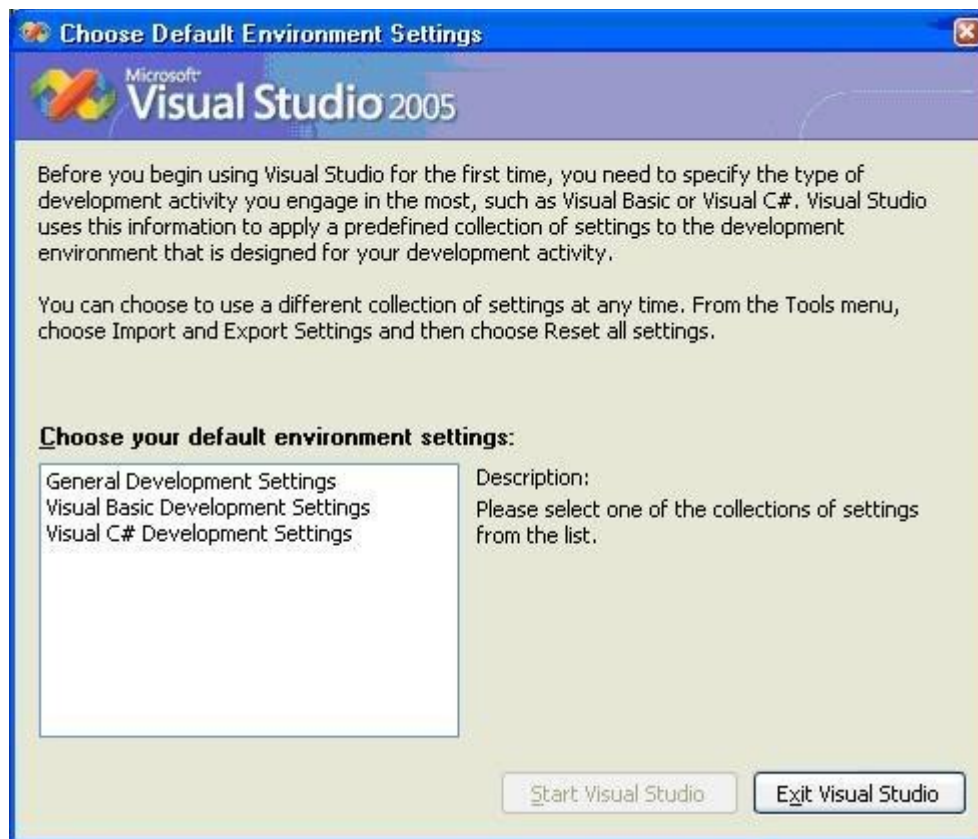


Hình 3

Thông thường người ta sử dụng cơ chế handshaking: None, vì vậy chỉ cần 3 dây nối như hình 2.

Như vậy về phần connecter là xong. Ta đi vào lập trình.

1. Programs→Microsoft Visual Studio 2005→Microsoft Visual Studio 2005.EXE.
2. Nếu lần đầu tiên bạn chạy Microsoft Visual Studio 2005 thì bảng hiện ra là:



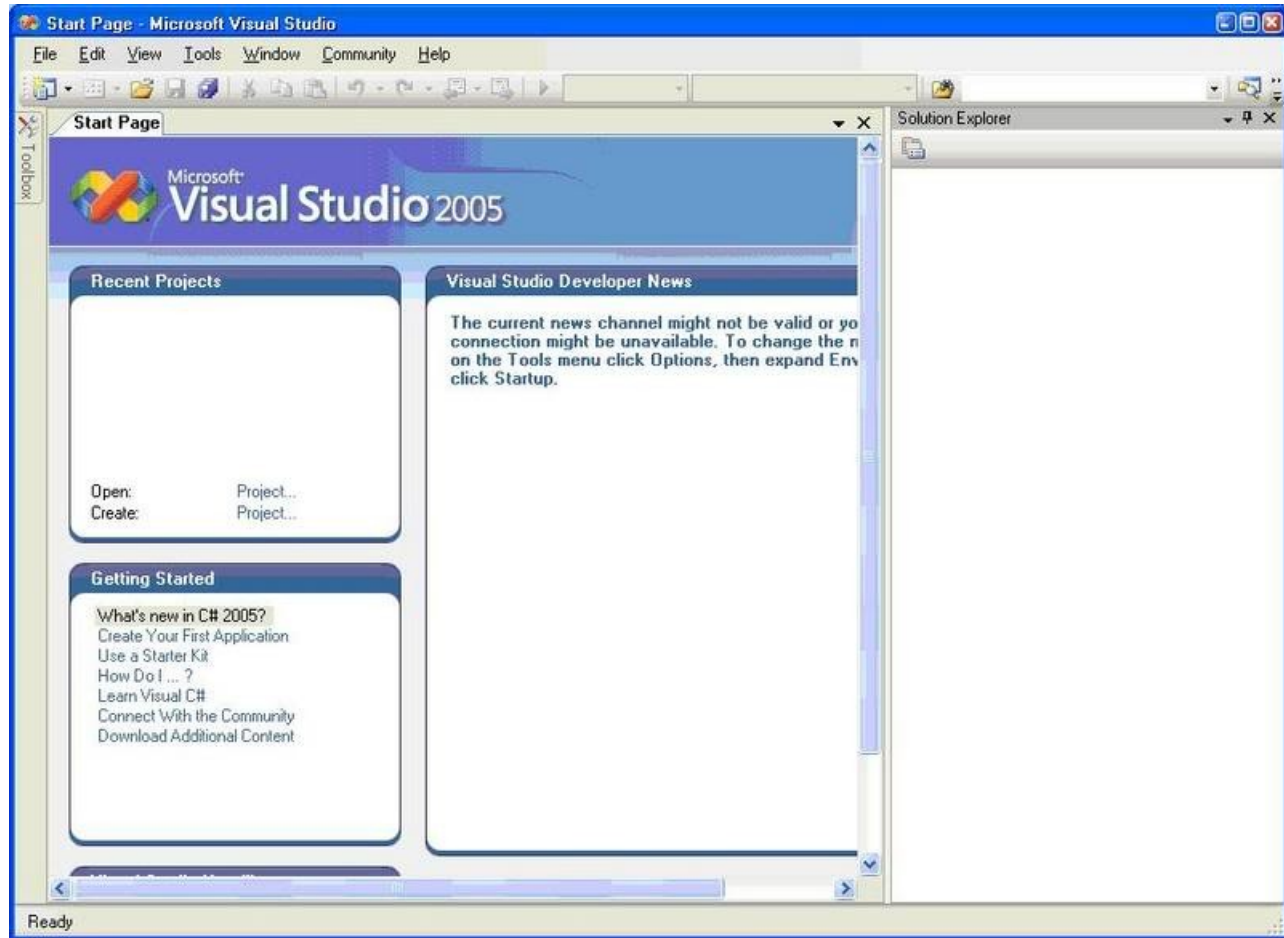
Hình 4

3. Lựa chọn Visual C# Development Settings → Kịch nút Start Visual Studio:



This image has been resized. Click this bar to view the full image. The original image is sized 800x586.

4.



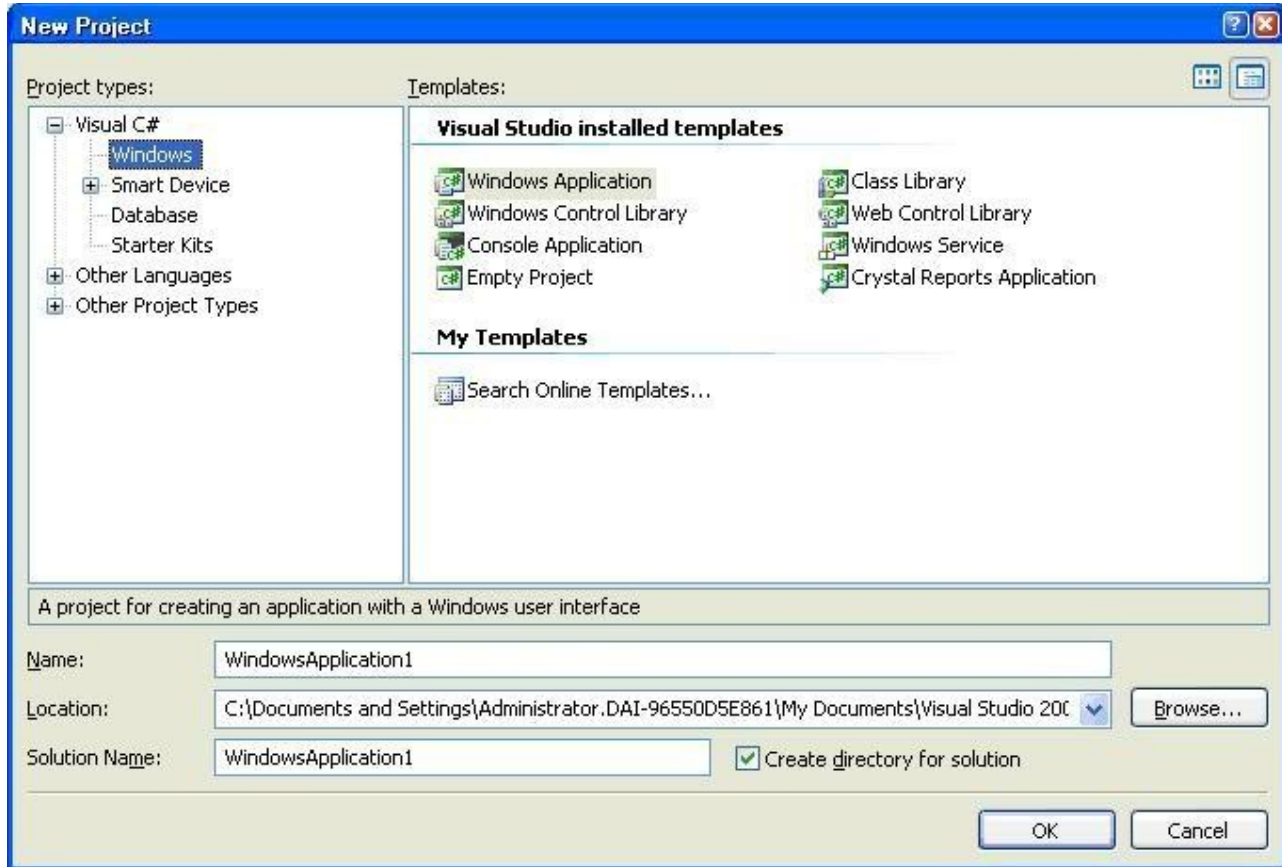
Hình 5

5. Chọn File → New → Project:




This image has been resized. Click this bar to view the full image. The original image is sized 681x464.

6.

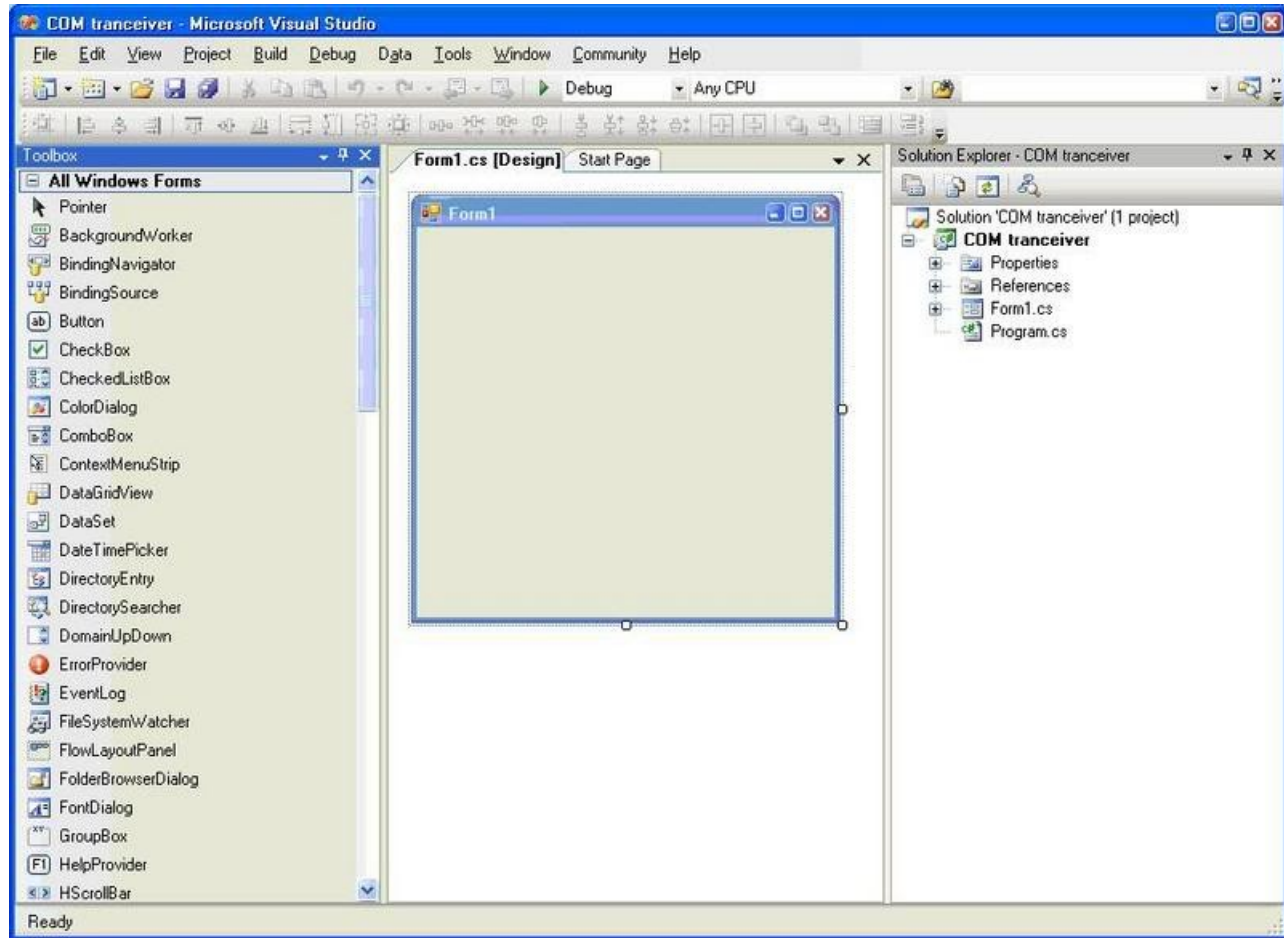


Hình 6

7. Trong cửa sổ Templates chọn Windows Application.→ Thay đổi tên project và vị trí lưu project như ví dụ Project Name: COM tranceiver.
Locatuion: E:\Nam+ Thang\CShape Project → Click OK→ Màn hình thiết kế xuất hiện:

 This image has been resized. Click this bar to view the full image. The original image is sized 800x586.


8.



Hình 7

9. Các điều khiển chứa trong Toolbox (Nếu không thấy xuất hiện trên màn hình thiết kế bạn có thể vào View → Toolbox).

10. Thiết kế giao diện:


 This image has been resized. Click this bar to view the full image. The original image is sized 800x136.

11.

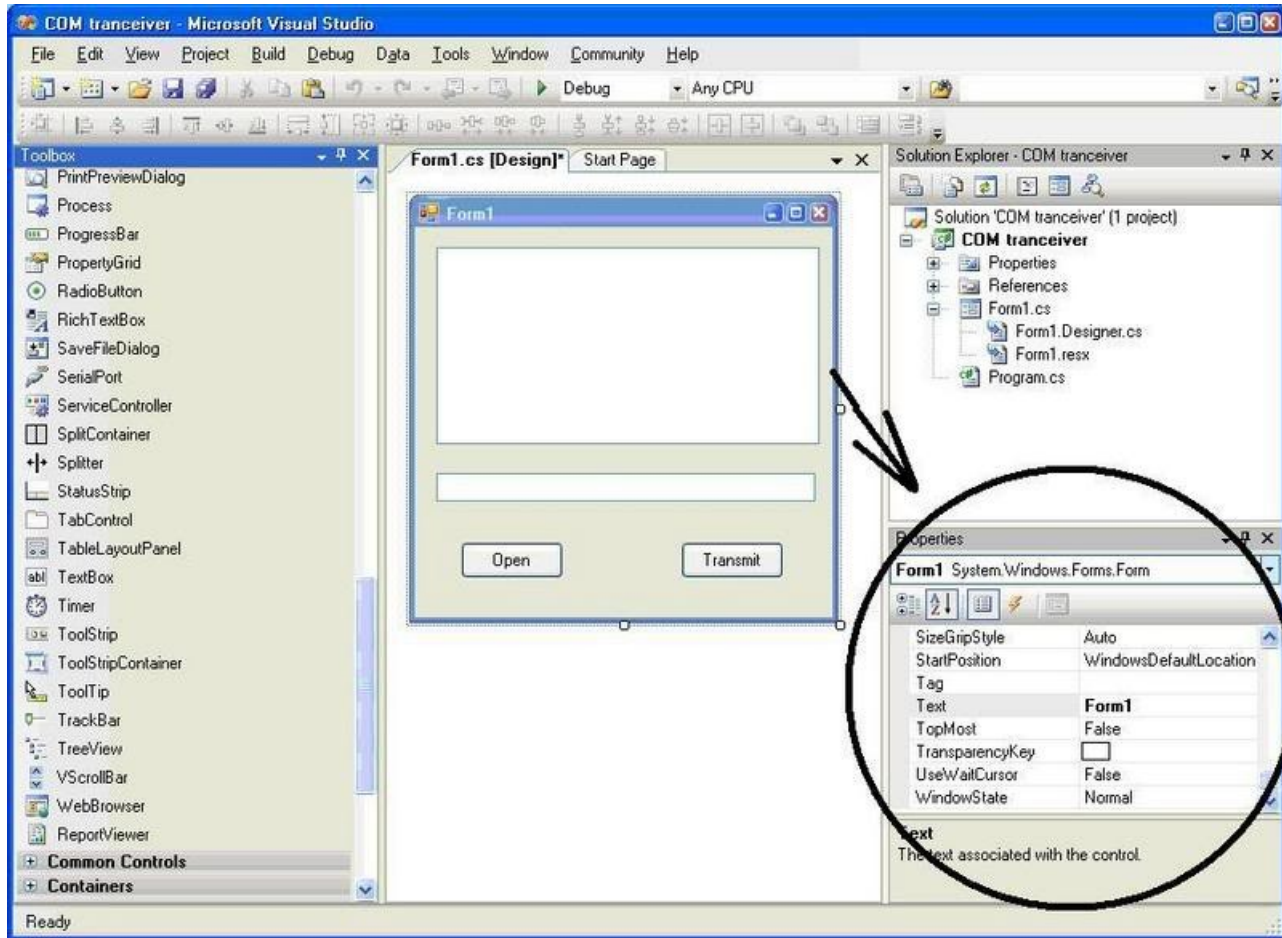
Tên đối tượng	Thuộc tính chỉnh sửa	Loại điều khiển	Chức năng
Tbox_Trans	Multiline=True	TextBox	Hiển thị dòng tin truyền
Tbox_Recei		TextBox	Hiển thị dòng tin nhận
Button_Open	Text=Open	Button	Mở và đóng cổng COM
Button_Trans	Text= Transmit	Button	Truyền tin

Hình 8

12. Muốn thay đổi tên cũng như các thuộc tính của đối tượng ta lựa chọn đối tượng click chuột phải chọn Properties.

 This image has been resized. Click this bar to view the full image. The original image is sized 800x586.

13.

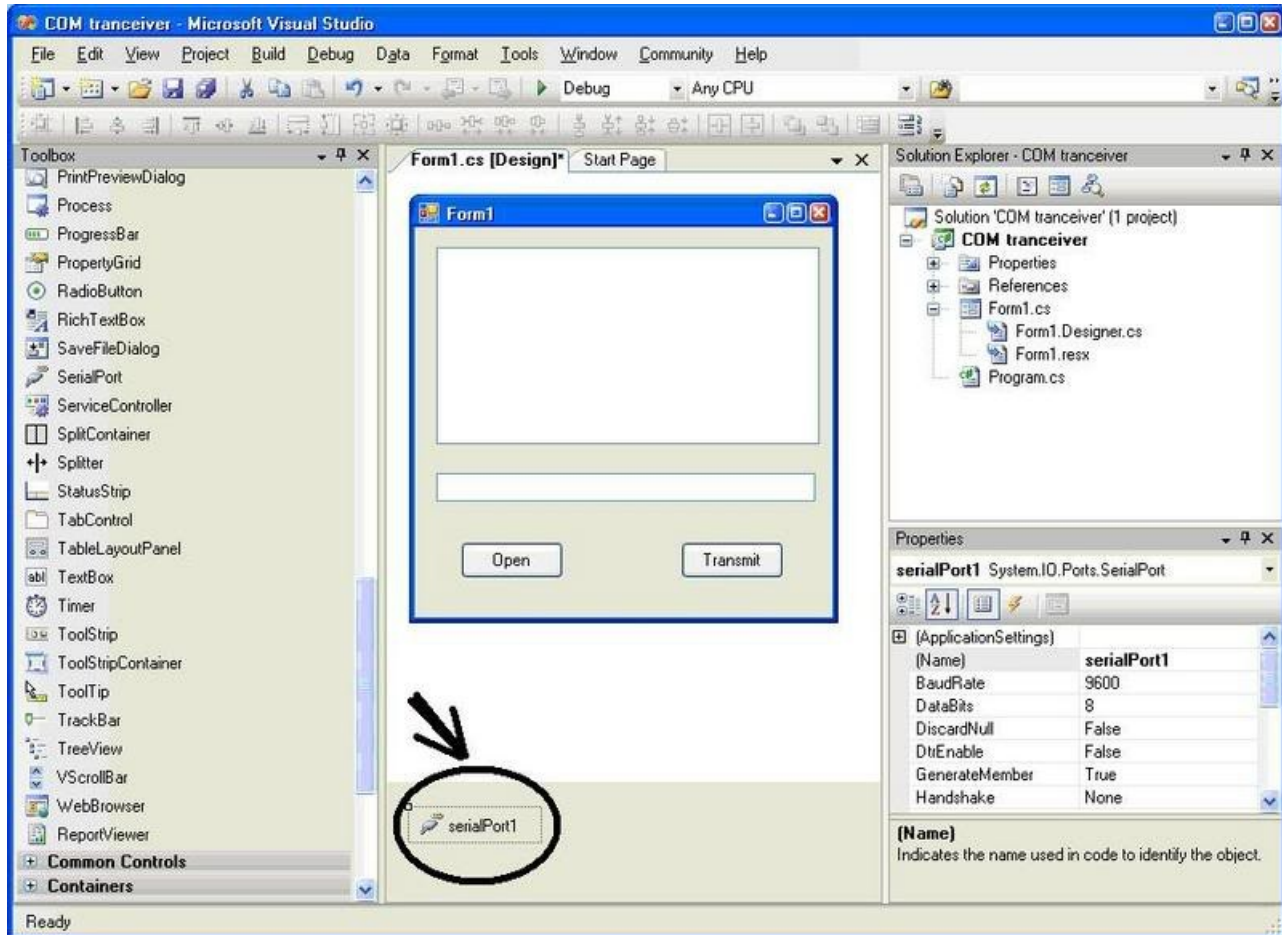


Hình 9

14. Trong ToolBox chọn Điều khiển SerialPort kéo vào cửa sổ Form1 → Xuất hiện đối tượng SerialPort1 ở phía dưới.

⚠ This image has been resized. Click this bar to view the full image. The original image is sized 800x586.

15.



Hình 10

16. Và những thuộc tính của điều khiển serialPort1 ở góc dưới bên trái: Ở đây ta có thể cài đặt các thông số của cổng COM. Để các tham số ở trạng thái default: Baudrate=9600, databits=8, handshake=none, stopbit = one, parity = none. Portname =COM1

Trong Visual Studio 2005 đã xây dựng lớp đối tượng SerialPort với đầy đủ các hàm hỗ trợ cho phép người dùng có thể tác động đến cổng COM một cách dễ dàng.

Một số hàm cơ bản trong SerialPort thường dùng:

- SerialPort.IsOpen(): Trả lại trạng thái của cổng là đang đóng hay mở.
- SerialPort.Open(): Mở cổng với thông số đã cài đặt.
- SerialPort.Close(): Đóng cổng.
- SerialPort.WriteLine(String data): Truyền một string xuống bộ đệm cổng để truyền đi.
- SerialPort.ReadExisting(): Đọc một string từ bộ đệm cổng.
- SerialPort.ReadChar(): Đọc một giá trị kiểu char từ bộ đệm cổng.
- SerialPort.ReadByte(): Đọc một giá trị kiểu Byte từ bộ đệm cổng.

Một số lưu ý về thuật giải của chương trình:

Quá trình truyền:

Quá trình truyền thì đơn giản, vì khi nào người dùng click nút truyền thì dữ liệu mới được truyền đi:

Trong Form1.cs[Design] click đúp vào nút Transmit trong Form1: Và chèn đoạn code sau vào hàm vừa tạo ra:

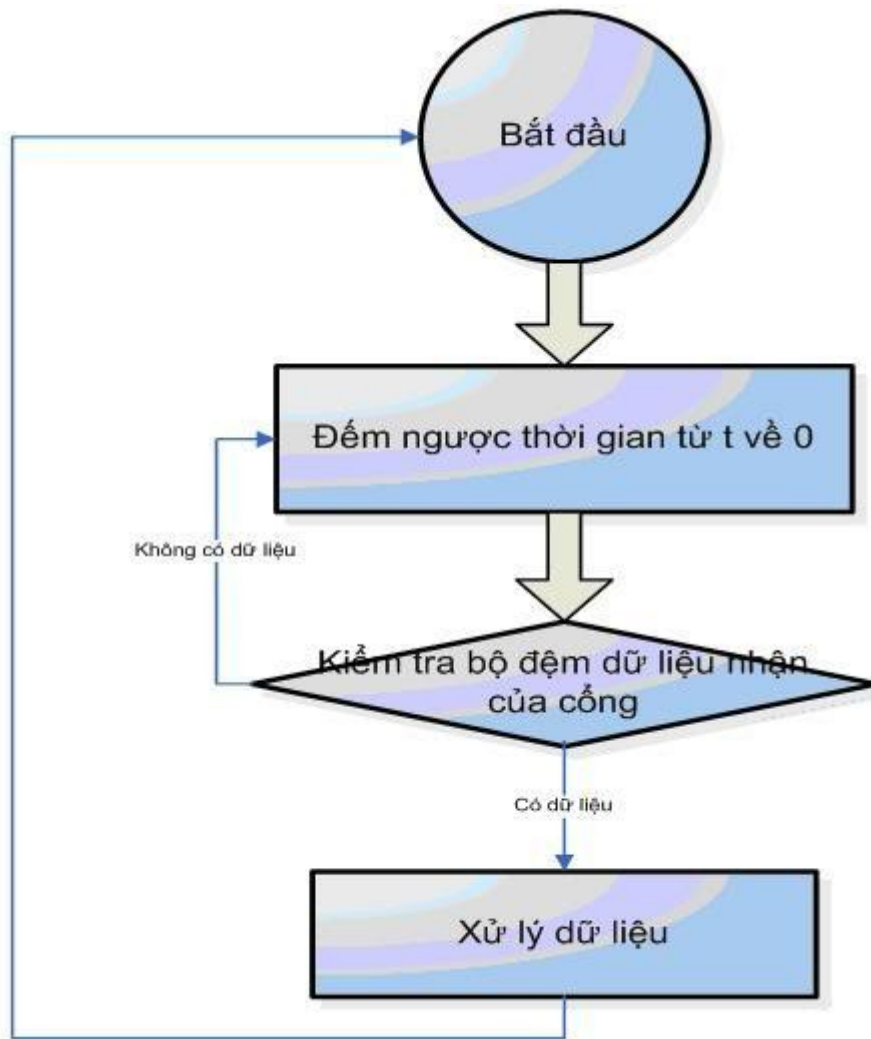
```
private void Button_Trans_Click(object sender, EventArgs e)
{
    if(serialPort1.IsOpen)
    {
        serialPort1.WriteLine(Tbox_Trans.Text);
    }
    else
    {
        MessageBox.Show("Cong chua mo! Hay mo cong","Chu y");
    }
}
```

Quá trình nhận dữ liệu:

Quá trình nhận dữ liệu là rắc rối hơn một chút, vì dữ liệu đến lúc nào là chưa biết. Có hai cách để thực hiện quá trình quá trình truyền:

- Cách một: Thường xuyên kiểm tra cổng xem dữ liệu có được truyền đến không.
- Cách hai: Phải bắt được sự kiện dữ liệu đến, khi dữ liệu đến ta gọi hàm xử lý.

Với cách 1 ta sử dụng một Timer ngắt thời gian, cứ sau một thời gian nhất định (mà ta cài đặt Timer) hàm kiểm tra và xử lý sẽ được gọi.



Hình11

Với cách 2 ta sử dụng sự kiện *SerialDataReceivedEvent*. Khi có sự kiện này ta sẽ đọc và xử lý dữ liệu.

Trong 2 cách trên ta thấy ngay cách thứ hai là đơn giản và thuật toán chương trình đơn giản hơn, tiết kiệm được thời gian xử lý. Do vậy tôi xin giới thiệu cách thứ 2 (cách 1 các bạn tự tìm hiểu. OK).

Chọn Fom1.cs[Design]. Click chuột phải vào Form1→View code→ Fom1.cs.
Thêm các đoạn code sau:

Thêm trong phần using: *using System.IO.Ports;*

Trong hàm Form1() trong Class Form1:

```

public Form1()
{
    InitializeComponent();
    serialPort1.DataReceived += new
  
```

```
SerialDataReceivedEventHandler(serialPort1_DataReceived);
}
```

serialPort1_DataReceived: là tên hàm sẽ được gọi đến khi có dữ liệu trong bộ đệm thanh ghi nhận.

Trong Class Form1, Thêm hàm serialPort1_DataReceived:

```
private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    string data = ""; // Lưu dữ liệu nhận
    if (sender == serialPort1)
    {
        data = serialPort1.ReadExisting();
        // Hiện thị lên text box
        Tbox_Recei.Text = Tbox_Recei.Text + data;
    }
}
```

Mở cổng: Khai báo biến Open kiểu Boolean trong class Form1:

```
public partial class Form1 : Form
{
    public bool Open = true;
    public Form1()
    {
```

```
.
.
.
```

Click đúp vào nút Open trong Form1.cs[Design]. Thêm code vào hàm vừa tạo ra:

```
private void Button_Open_Click(object sender, EventArgs e)
{
    if (Open)
    {
        Open = false;
        Button_Open.Text = "Close";
        serialPort1.Open();
    }
    else
    {
        Open = true;
        Button_Open.Text = "Open";
        serialPort1.Close();
    }
}
```

Thêm đoạn code sau vào class Form1 để xử lý khi chương trình bị đóng:

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
```

```
{
    MessageBox.Show("Bạn muốn thoát khỏi chương trình", "Chú ý");
    serialPort1.Close();
}
```

}
 Ấn F6 để build và F5 để debug.

Thế là xong một chương trình hết sức đơn giản! Do thời gian có hạn nên ví dụ con đơn giản và có thể còn có sai sót, mong các bạn góp ý. Từ ví dụ này bạn có thể phát triển và ứng dụng vào những chương trình hay hơn. Chúc các bạn thành công!

OK! Chúng ta hoàn toàn có thể thay đổi toàn bộ các thông số của cổng COM. Bạn có thể chèn thêm hàm khởi tạo cổng vào class Form1:

```
private void InitializePort(long int baudrate)
{
    //create a new serial port

    if (serialPort1.IsOpen)
    {
        serialPort1.Close();
    }
    //set all important data
    serialPort1.BaudRate = baudrate;
    serialPort1.DataBits = 8;
    serialPort1.Encoding = Encoding.ASCII;
    serialPort1.Parity = Parity.None;
    serialPort1.PortName = "COM1";
    serialPort1.StopBits = StopBits.One;
    serialPort1.BytesToReadThreshold = 1;

}
```

Và như vậy bạn có thể tùy thích thay đổi các thông số của cổng. Chỉ cần trước khi muốn truyền với tốc độ khác bạn gọi lại hàm *InitializePort(tốc độ bạn thích)*. baudrate max có thể đạt được là 115200 baud. Ví dụ bạn có thể tạo một TextBox (Tbox_baudrate) để nhập tốc độ và một nút để thay đổi tốc độ (Button_ChangeBaudRate). Thêm hàm:

```
private void Button_ChangeBaudRate_Click(object sender, EventArgs e)
```

```

{
long int baudrate=0;
baudrate=Convert.ToInt16(Tbox_baudrate.Text);
InitializePort(baudrate);

}

```

OK! hi vọng có thể giúp ích cho bạn.

Vao ra du lieu voi Serial Port

```

using System.IO.Ports;
using System.IO;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

private SerialPort comport = new SerialPort("COM1", 9600, Parity.None,
8, StopBits.One);

public Form1()
{
    InitializeComponent();

    private void btnKetThuc_Click(object sender, EventArgs e)
    {
        Close();
    }

private void btnTestPort_Click(object sender, EventArgs e)
{
    try
    {
        comport.Open();
        this.comport.DtrEnable = true;
        this.comport.RtsEnable = true;

    }
    catch
    {
        txtThongBao.Enabled = true;
        txtThongBao.Text = "Có sự cố khi mở cổng Com, hãy kiểm
tra lại hệ thống, Chương trình chưa thực hiện được!";
    }
    if (comport.IsOpen)
    {
        btnTestPort.Enabled = false;
        btnHexTx.Enabled = true;

        btnClosePort.Enabled = true;
    }
}

```

```

        txtRx.Enabled = true;
        txtHexTx.Enabled = true;
        txtTx.Enabled = true;
        txtThongBao.Enabled = true;
        txtThongBao.Text = "Cổng Com 1 đã được mở rồi, bắt đầu
chiến đấu đi";
    }
    else this.Focus();
}

private void dulieuden(object sender,
SerialDataReceivedEventArgs e)
{
    txtRx.Enabled = true; // Hiển thị TextBox nếu Dữ liệu đến
    btnRx.Enabled = true; // Hiển thị button Ẩn nếu Dữ liệu đến
    comport.DataReceived += new
SerialDataReceivedEventHandler(dulieuden);
}

// Đọc toàn bộ dữ liệu lưu tại buffer, triệu tập dữ liệu lưu trong hàm
dulieuden
private void btnRx_Click(object sender, EventArgs e)
{
    string docdulieu = comport.ReadExisting().ToString();
    this.txtRx.Text = docdulieu;
    txtThongBao.Text = "Dữ liệu nhận được ứng với mã Hexa là: "
+ AtoX(docdulieu);
}

private void btnClosePort_Click(object sender, EventArgs e)
{
    comport.Close();
    txtThongBao.Text = "Cổng Com đã được đóng";

    btnTestPort.Enabled = true;
    btnHexTx.Enabled = false;

    txtTx.Enabled = false;
    txtHexTx.Enabled = false;
    txtThongBao.Enabled = false;

    btnClosePort.Enabled = false;
}

```

Chuyển đổi Chuỗi [ASCII](#) sang Định Dạng Hexa.

```

private string AtoX(string asc)
{
    int nLines;
    int nChars;
    int offset;

```



```

        string hex = " ";

        // tính toán số lượng dòng Hexa.
        if ((asc.Length % 16) > 0)
            nLines = asc.Length / 16 + 1;
        else
            nLines = asc.Length / 16;

        // Chuyển đổi sang những dòng Hexa.
        for (int i = 0; i < nLines; i++)
        {
            offset = i * 16;
            if ((asc.Length - offset) > 16)
                nChars = 16;
            else
                nChars = asc.Length - offset;
            hex += this.HexLine(i, asc.Substring(offset, nChars)) +
"\r\n";
        }
        return hex;
    }

    /// <summary>
    /// Chuyển một chuỗi ASCII 16 byte sang định dạng hexa.
    /// </summary>
    private string HexLine(int lNum, string asc)
    {
        string hex = "";

        // Copy dòng vào bộ đệm.
        char[] c = new char[16];
        asc.CopyTo(0, c, 0, asc.Length);

        // Tạo offset.
        hex += String.Format("{0:X8} - {0:X8}", lNum * 16, (lNum +
1) * 16 - 1);
        hex += " ";

        // Chuyển các ký tự sang định dạng chuẩn hexa.
        for (int i = 0; i < asc.Length; i++)
        {
            if ((i != 0) && ((i % 4) == 0))
                hex += " ";
            hex += String.Format("{0:X2}", (int)c[i]);
        }

        // Đệm thêm.
        int nSpaces = 62 - hex.Length;
        for (int i = 0; i < nSpaces; i++)
            hex += " ";

        //Chuyển ASCII tới cuối dòng.
        for (int i = 0; i < asc.Length; i++)
        {
            if (((int)c[i] < 32) || ((int)c[i] > 126))
                hex += ".";
            else
                hex += c[i].ToString();
        }
    }

```

```

        // Trả lại dòng hexa .
        return hex;
    }

    private void btnHexTx_Click_1(object sender, EventArgs e)
    {
        try
        {
            byte[] data = HexStringToByteArray(txtHexTx.Text);
            comport.Write(data, 0, data.Length); //Đưa mảng số Hexa
            qua cổng com với định dạng chuẩn
            string a = this.txtTx.Text;
            comport.Write(AtoX(a));
            txtThongBao.Text = "Thông tin được gửi ra cổng RS232 với
            mã Hex là: " + txtHexTx.Text.ToUpper() + "\r\n \r\n" + "Thông tin được
            gửi ra cổng RS232 chuyển sang mã Hex Tập tin Text là: \r\n \r\n" +
            AtoX(a).ToString();

        }

        catch (FormatException)
        {
            txtThongBao.Text = "Có thể cổng com đóng hoặc bạn nhập
            chính xác Mã Hexa cần Truyền Nhận. Không nhận được định dạng chuỗi sau:
            " + txtHexTx.Text.ToUpper();
        }
        return;
    }
}

```

**Thấy anh em quan tâm, tôi ngồi xây dựng một Class riêng để anh em có thể
nhúng trực tiếp vào chương trình khi thiết kế ứng dụng phần mềm!**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO.Ports;
using System.Threading;

class ComPorts
{
    public Boolean OpenComPortOk;
    public ComPorts()
    {

    }

    ~ComPorts()
    {

    }
}

```

```

    /// <summary>
    /// Mo cong COM1
    /// </summary>
    /// <returns></returns>
    public SerialPort Open(String Port, int Baud, Parity Prt, int DtB,
StopBits StB)
    {
        //SerialPort ComPort = new SerialPort("COM1", 9600,
Parity.None , 8, StopBits.One);
        SerialPort ComPort = new SerialPort( Port,Baud, Prt, DtB, StB);
        try
        {
            if (!ComPort.IsOpen)
            {
                ComPort.Open();
                this.OpenComPortOk = true;
            }
        }
        catch (Exception e)
        {
            MessageBox.Show("Lỗi mở cổng:"+e.ToString());

            this.OpenComPortOk = false;
            return ComPort;
        }
        ComPort.RtsEnable = true;
        ComPort.DtrEnable = true;

        return ComPort;
    }
    public void Close()
    {
    }

    /// <summary>
    /// Send(SerialPort ComPort, String data) se
    /// Gui chuoai data ra cong ComPort
    /// </summary>
    /// <param name="ComPort"></param>
    /// <param name="data"></param>
    public void Send(SerialPort ComPort, String data)
    {
        ComPort.Write(data);
    }

    public String Receive(SerialPort ComPort)
    {
        String ReceiveData = "";
        ReceiveData = ComPort.ReadExisting();
        return ReceiveData;
    }

    public byte ReceiveByte(SerialPort ComPort)
    {
        byte Value;
        Value = Convert.ToByte(ComPort.ReadByte());
        return Value;
    }

```

```
}
```

Code mẫu lập trình công COM trên C#

```
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) { }
    }
}

public static string SetPortName(string defaultPortName)
{
    string portName;

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

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

    if (portName == "")
    {
        portName = defaultPortName;
    }
    return portName;
}

public static int SetPortBaudRate(int defaultPortBaudRate)
{
    string baudRate;

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

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

    return int.Parse(baudRate);
}

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("Parity({0}):", defaultPortParity.ToString());
        parity = Console.ReadLine();

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

        return (Parity)Enum.Parse(typeof(Parity), parity);
    }

    public static int SetPortDataBits(int defaultPortDataBits)
    {
        string dataBits;

        Console.Write("Data Bits({0}): ", defaultPortDataBits);
        dataBits = Console.ReadLine();

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

        return int.Parse(dataBits);
    }

    public static StopBits SetPortStopBits(StopBits defaultPortStopBits)
    {
        string stopBits;

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

        Console.Write("Stop Bits({0}):",
defaultPortStopBits.ToString());
        stopBits = Console.ReadLine();

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

        return (StopBits)Enum.Parse(typeof(StopBits), stopBits);
    }

    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("Stop Bits({0}):",
defaultPortHandshake.ToString());
    handshake = Console.ReadLine();

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

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

```

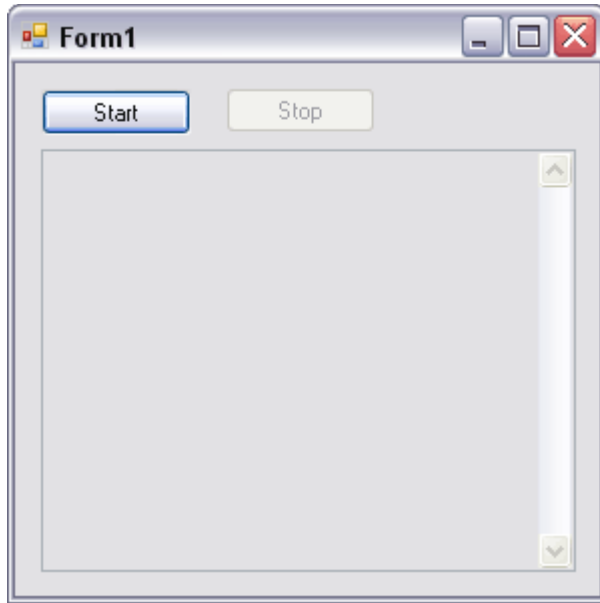
Simple Serial Communication with Microsoft Visual C# Express

Introduction

Here's a simple Windows serial communication program you can write yourself in 10 minutes. Microsoft's FREE Visual C# Express with the .Net Framework now includes a serial port class that eliminates the cumbersome setup for threads and overlapped I/O. They've made serial communication easy again. This project can readily be adapted to a variety of applications requiring serial communication.

The example has 2 buttons (Start and Stop) and a TextBox. When the application is running, click Start to open a COM port. Once the port is open, incoming serial data will appear in the TextBox. Characters typed into the TextBox will be transmitted out the serial port. Click Stop to close the COM port. That's it. The example code is explained below. You can either follow along and write your own version or simply download the Visual C# Express [Simple Serial project](#) directly. A complete [code listing](#) appears at the end of this page.

Create the Application



Open Visual C# Express and select File->New Project. Chose the "Windows Application" icon and name the new project "Simple Serial" (the Project Name edit box is at the bottom of dialog). This will create a blank Form1 and skeleton code for the application.

From the Toolbox palette, add two Buttons to Form1. Select button1. In the Properties palette change the **button1** Name to **buttonStart**. Change its its Text property to **Start**. Name the other button **buttonStop**. Change its Text property to **Stop** and set its Enable property to **False**.

Add a TextBox to Form1. Change its Multiline property to **True**, ReadOnly property to **True**, and Scrollbars property to **Vertical**. Resize the Textbox and arrange the Buttons to look like the image here.

Last, add a SerialPort class from the Toolbox palette to Form1. Since it's a non-visual class, its icon does not stay on Form1 but appears at the bottom of the Form1 design page.

Select the Start button in Form1. In the Properties palette, select the Events icon. Find the event labled "Click" and double-click it. This will create skeleton code for the Start button "Click" event-handler. Leave it empty for now. We'll add our code to it shortly. Go back to the design tab, select the Stop button and add a "Click" event-handler for it just like you did for the Start button. Using the same technique, add a "KeyPress" event-handler to the TextBox, a "DataReceived" event-handler to the SerialPort, and a "FormClosing" event-handler to Form1 (be sure to select FormClosing... not FormClosed).

Now, cut and paste code from the event handlers in the [listing](#) below into the empty event handlers in your project. Cut and paste the variable declaration for **RxString**. Be sure to place it immediately before the Form1() function as shown in the listing. Last, cut and paste the entire DisplayText() function. That's it. You're done.

Test the Application

In the Visual C# Express IDE, press F5 (Run) or the green-arrow Run button. With the Simple Serial application running, press the Start button to open the COM port. Incoming

serial text data will appear in the textBox1. Anything you type in textBox1 will be transmitted. If you don't have a connected serial device suitable for testing the program, use a loopback connector. Make one by simply jumpering pins 2 and 3 of a serial cable or DB9F connector. With a loopback connector, anything you type in textBox1 will be echoed back through the port and displayed in textBox1. Press the Stop button to close the port.

If the program doesn't work, carefully review your code and serial connections. In the code for buttonStart_Click(), be sure the COM port you specify is the one you're connected to. It must physically exist on your computer. COM1 is most common but you might be connected to COM2, COM3, etc. If the port is connected to an external serial device, the comm parameters (baud rate, etc.) of the device and your program must agree. Also note that there is no serial handshaking in our application. An external device might require it. Change parameters if needed.

How it Works

All the heavy lifting is performed behind the scenes thanks to the SerialPort component. The rest is pretty simple. The only tricky part is the Invoke method that allows our DisplayText() function to update textBox1. Let's look at the code for each event handler.

In buttonStart_Click(), we begin by setting the port name and baud rate. In our example PortName is COM1 but you can set it to any other port available on your computer. Notice the PortName is a string and must be in quotes. The baud rate must agree with the baud rate of whatever is on the other end of the serial connection. We then call the Open() function. If the port opened OK, we disable the Start button, enable the Stop button, and allow writing in textBox1.

```
private void buttonStart_Click(object sender, EventArgs e)
{
    serialPort1.PortName = "COM1";
    serialPort1.BaudRate = 9600;

    serialPort1.Open();
    if (serialPort1.IsOpen)
    {
        buttonStart.Enabled = false;
        buttonStop.Enabled = true;
        textBox1.ReadOnly = false;
    }
}
```

Once the serialPort1 is open, any incoming serial characters will cause a DataReceived event to fire. Inside the event handler we read all existing characters from the internal serial receive buffer into string RxString. *The next part is critical and not obvious.* serialPort1 runs in it own separate thread behind the scenes. This thread cannot directly call any functions in the main thread of our application. However, a special function, Invoke(), will allow it. So we use Invoke to call our DisplayText() function. RxString is the global string variable accessible by both threads.

```
private void serialPort1_DataReceived
(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
{
    RxString = serialPort1.ReadExisting();
    this.Invoke(new EventHandler(DisplayText));
}
```

Our DisplayText() function is simple. We just append the text in RxString to whatever is already in textBox1.

```
private void DisplayText(object sender, EventArgs e)
{
    textBox1.AppendText(RxString);
}
```

The textbox1 KeyPress() function captures characters typed into textBox1 and writes them to serialPort1. Write() can only send characters from a char type array so we declare one with a length of [1] and assign the KeyChar value to it. With the arguments in Write(), we tell it to send the characters in the buff, offset of 0 chars into the array, and a length of 1 char. We set the event to "Handled" to prevent the typed character from appearing in textBox1. If you want it to appear (local echo), omit the line.

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    // If the port is closed, don't try to send a character.
    if(!serialPort1.IsOpen) return;

    // If the port is Open, declare a char[] array with one element.

    char[] buff = new char[1];

    // Load element 0 with the key character.
    buff[0] = e.KeyChar;

    // Send the one character buffer.
    serialPort1.Write(buff, 0, 1);

    // Set the KeyPress event as handled so the character won't
    // display locally. If you want it to display, omit the next line.

    e.Handled = true;
}
```

Clicking the Stop button calls buttonStop_Click(). If serialPort1 is open, we close the port and set the button enables and textBox1 ReadOnly state back to their previous value.

```
private void buttonStop_Click(object sender, EventArgs e)
{
    if (serialPort1.IsOpen)
```

```

        {
            serialPort1.Close();
            buttonStart.Enabled = true;
            buttonStop.Enabled = false;
            textBox1.ReadOnly = true;
        }
    }
}

```

Last but not least, if the application is closed while serialPort1 is open, the port must be closed first or the program will hang.

```


private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (serialPort1.IsOpen) serialPort1.Close();
}

```

Final Thoughts

For the sake of clarity and simplicity, our example application omits a few things.

- textBox1 cannot properly display a CR without a LF. TextBoxes have the quirky need for a CR-LF pair and they must be in that order. If a CR arrives by itself, you'll need to add the LF.
- textBox1 holds a lot of characters but will eventually run out of memory and just stop displaying anything new that arrives. You can limit the number of characters it will display by setting the textBox1 MaxLength property but then it will just run out of space sooner. A better approach is to limit the number of lines without affecting the display.
- Our program has no handshaking. Some devices require it.

You are free to use all the information presented here. There are no restrictions. There's no support available but the program is simple enough that you probably won't need it. Send comments to  csharpcomments@simpleserial.com You need JavaScript to see my email address

Links

[SimpleSerialCS.zip](#) Download Simple Serial Project for Visual C# Express
[Microsoft MSDN Serial Communication Resources](#) WIN32 API serial communication reference.
www.microsoft.com/express/vcsharp/ Microsoft's FREE Visual C# Express

Full Code Listing

```

using System;
using System.Collections.Generic;

```

```

using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace SimpleSerial
{
    public partial class Form1 : Form
    {
        // Add this variable

        string RxString;

        public Form1()
        {
            InitializeComponent();

            private void buttonStart_Click(object sender, EventArgs e)
            {
                serialPort1.PortName = "COM1";
                serialPort1.BaudRate = 9600;

                serialPort1.Open();
                if (serialPort1.IsOpen)
                {
                    buttonStart.Enabled = false;
                    buttonStop.Enabled = true;
                    textBox1.ReadOnly = false;
                }
            }

            private void buttonStop_Click(object sender, EventArgs e)
            {
                if (serialPort1.IsOpen)
                {
                    serialPort1.Close();
                    buttonStart.Enabled = true;
                    buttonStop.Enabled = false;
                    textBox1.ReadOnly = true;
                }
            }

            private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
            {
                if (serialPort1.IsOpen) serialPort1.Close();
            }

            private void textBox1_KeyPress(object sender,
KeyPressEventArgs e)
            {
                // If the port is closed, don't try to send a character.

                if(!serialPort1.IsOpen) return;

                // If the port is Open, declare a char[] array with one
element.
                char[] buff = new char[1];

```



```

        // Load element 0 with the key character.

        buff[0] = e.KeyChar;

        // Send the one character buffer.
        serialPort1.Write(buff, 0, 1);

        // Set the KeyPress event as handled so the character
won't        // display locally. If you want it to display, omit the
next line.    e.Handled = true;
    }

    private void DisplayText(object sender, EventArgs e)
    {
        textBox1.AppendText(RxString);
    }

    private void serialPort1_DataReceived
e)        (object sender, System.IO.Ports.SerialDataReceivedEventArgs
    {
        RxString = serialPort1.ReadExisting();
        this.Invoke(new EventHandler(DisplayText));
    }
}

```
