

[articles](#) [Q&A](#) [forums](#) [lounge](#)

Search for articles, questions, tips



# Serial Communication using C# and Whidbey



Tapan Dantre, 20 Oct 2004

CPOL

Rate:

★★★★☆ 4.18 (116 votes)

In this article, I will give you an introduction on how to do serial port communication on .NET platform using C#.



**Is your email address OK?** You are signed up for our newsletters but your email address is either unconfirmed, or has not been reconfirmed in a long time. Please [click here to have a confirmation email sent](#) so we can confirm your email address and start sending you newsletters again. Alternatively, you can [update your subscriptions](#).

## Introduction

In this article, I will give you an introduction on how to do serial port communication on .NET platform using C#. The .NET framework version 2.0 (beta) provides features for serial communication. The framework provides **System.IO.Ports** namespace. The new framework provides classes with the ability to access the serial ports on a computer, and to communicate with serial I/O devices. We will be using RS 232 C standard for communication between PCs. In full duplex mode, here I am not going to use any handshaking or flow control, I will use null modem connection for communication.

## The Namespace

The **System.IO.Ports** namespace contains classes for controlling serial ports. The most important class is the **SerialPort** class.

## The SerialPort Class

**SerialPort** class provides a framework for synchronous and event-driven I/O, access to pin and break states, and access to serial driver properties. It can be used to wrap Stream objects, allowing the serial port to be accessed by classes that use streams. That is, **SerialPort** class represents a serial port resource.

## Creating SerialPort Object

By creating an object of this type, we will be able to programmatically control all aspects of serial communication.

The methods of **SerialPort** class that we will use are:

- ✎ **ReadLine()**: Reads up to the NewLine value in the input buffer. If a New Line is not found before timeout, this method returns a **null** value.
- ✎ **WriteLine(string)**: Writes the specified string and the New Line value to the output buffer. The written output includes the New Line string.
- ✎ **Open()**: Opens a new serial port connection.
- ✎ **Close()**: Closes the port connection.

To create a **SerialPort** object, all we need to do is:

Hide Copy Code

```
//create a Serial Port object  
SerialPort sp = new SerialPort ();
```

In all, there are seven public constructors for creating **SerialPort**. But I am using the parameter less constructor, the constructor uses default property values. For example, the value of **DataBits** defaults to 8, and **StopBits** defaults to 1. The default communication port will be COM1.

The public properties of **SerialPort** class that we will use are:

- ✎ **BaudRate**: Gets or sets the serial baud rate.
- ✎ **StopBits**: Gets or sets the standard number of stopbits per byte.
- ✎ **ReadTimeout**: Gets or sets the number of milliseconds before a timeout occurs when a read operation does not finish.

There are many public properties, but except these three, all properties will have default values.

## About Serial Port (Hardware)

The serial port on your PC is a full-duplex device meaning that it can send and receive data at the same time. In order to be able to do this, it uses separate lines for transmitting and receiving data. Some types of serial devices support only one-way communication, and therefore, use only two wires in the cable - the transmit line and the signal ground.


## Data Transfer

In serial communication, a byte of data is transferred through a single wire one bit at a time. The packets contain a start bit, data, and stop bit. Once the start bit has been sent, the transmitter sends the actual data bits. There may either be 5, 6, 7, or 8 data bits, depending on the number you have selected. Both receiver and the transmitter must agree on the number of data bits, as well as the baud rate.

## Null Modem

A Null Modem cable simply crosses the receive and transmit lines so that transmit on one end is connected to receive on the other end and vice versa. In addition to transmit and receive, DTR & DSR, as well as RTS & CTS are also crossed in a Null Modem connection.

Here is the pin diagram:

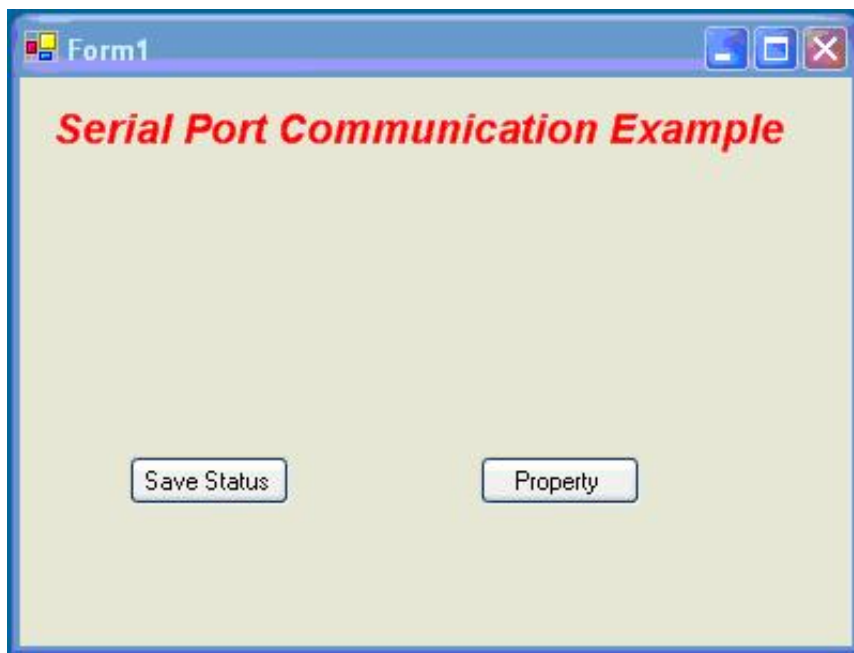
9 Pin Connector on a DTE device (PC connection)	
Male RS232 DB9	
Pin Number	Direction of signal:
1	Carrier Detect (CD)
2	Received Data (RD)
3	Transmitted Data (TD)
4	Data Terminal Ready (DTR)
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR)
7	Request To Send (RTS)
8	Clear To Send (CTS)
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

As we are not using any handshake, we will use three wire connections in which we will connect pin 2 of one connector to pin 3 of the other. For both the connectors, connect pin 5 (ground) of both the connectors with each other (common ground).

If you want, you can use only one PC as both transmitter and receiver for this communication. Then, just take a DB 9 connector and a small wire, and connect pin no 2 and 3 using the wire that will connect a loop back. That is, whatever you will send, the same data you will be receiving.

## The Example Application

The main form:



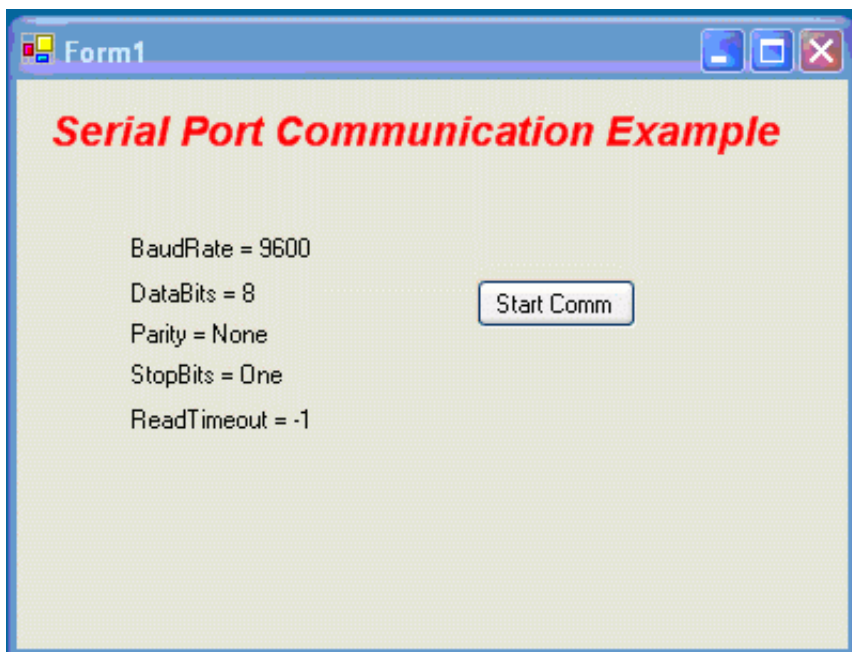
Here, if you want to work with default values for the public properties, then press Save Status. If you want to change the value for specified properties, then press Property.

Then the following dialog will pop:

A Windows-style dialog box titled "PropertyPage" with a blue title bar and standard minimize, maximize, and close buttons. The dialog has a light beige background. It contains two labels, "Baud Rate" and "Stop Bits", each followed by a white text box with a blue downward arrow on the right side, indicating a dropdown menu. At the bottom of the dialog are two buttons: "OK" and "Cancel".

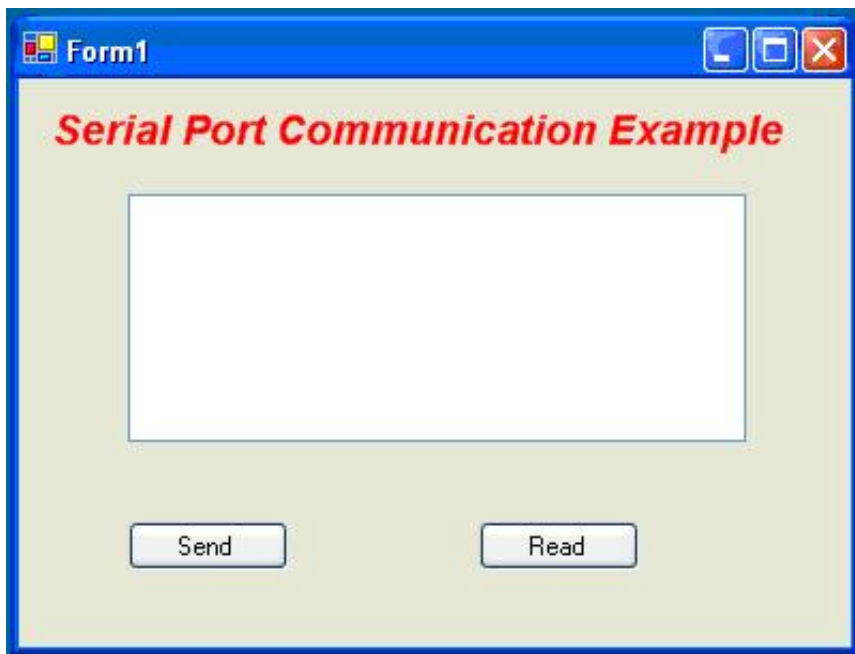
Here, you can select values for baud rate and stop bit. To save these changes, press OK.

If you directly press save status or save the changed values for properties, the following form will be displayed:

A Windows-style window titled "Form1" with a blue title bar and standard minimize, maximize, and close buttons. The window has a light beige background. At the top, the text "Serial Port Communication Example" is displayed in a bold, red, italicized font. Below this, the following properties are listed in a standard black font: "BaudRate = 9600", "DataBits = 8", "Parity = None", "StopBits = One", and "ReadTimeout = -1". To the right of these properties is a button labeled "Start Comm".

Which will display the values for some of the properties.

For starting communication, press Start Comm. As soon as you press the button, the following form will be displayed:



Type data in textbox which is to be transferred, followed by a new line, and press Send button. As soon as you press the button, the data will be send to the send buffer and later to the COM port.

For reading the data form COM port, press Read button. If there is any data in read buffer, it will be displayed in the textbox. If there no data to be read for 500 ms, then an error message will be displayed informing the same.

Here is a code example for the above application:

## Code for the Main app

Hide Shrink ▲ Copy Code

```
#region Using directives

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

#endregion

namespace Serialexpample
{
    partial class Form1 : Form
    {
        //create instance of property page
        //property page is used to set values for stop bits and
        //baud rate

        PropertyPage pp = new PropertyPage();

        //create an Serial Port object
        SerialPort sp = new SerialPort();

        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```
private void propertyButton_Click(object sender, EventArgs e)
{
    //show property dialog
    pp.ShowDialog();

    propertyButton.Hide();
}

private void sendButton_Click(object sender, EventArgs e)
{
    try
    {
        //write line to serial port
        sp.WriteLine(textBox.Text);
        //clear the text box
        textBox.Text = "";
    }
    catch (System.Exception ex)
    {
        baudRateLabel.Text = ex.Message;
    }
}

private void ReadButton_Click(object sender, EventArgs e)
{
    try
    {
        //clear the text box
        textBox.Text = "";
        //read serial port and displayed the data in text box
        textBox.Text = sp.ReadLine();
    }
    catch (System.Exception ex)
    {
        baudRateLabel.Text = ex.Message;
    }
}

private void Form1_Load(object sender, EventArgs e)
{
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    MessageBox.Show("Do u want to Close the App");
    sp.Close();
}

private void startCommButton_Click(object sender, EventArgs e)
{
    startCommButton.Hide();
    sendButton.Show();
    readButton.Show();
    textBox.Show();
}

//when we want to save the status(value)
private void saveStatusButton_Click_1(object sender, EventArgs e)
{
    //display values
    //if no property is set the default values
    if (pp.bRate == "" && pp.sBits == "")
```

```

    {
        dataBitLabel.Text = "BaudRate = " + sp.BaudRate.ToString();
        readTimeOutLabel.Text = "StopBits = " + sp.StopBits.ToString();
    }
    else
    {
        dataBitLabel.Text = "BaudRate = " + pp.bRate;
        readTimeOutLabel.Text = "StopBits = " + pp.sBits;
    }

    parityLabel.Text = "DataBits = " + sp.DataBits.ToString();
    stopBitLabel.Text = "Parity = " + sp.Parity.ToString();
    readTimeOutLabel.Text = "ReadTimeout = " +
        sp.ReadTimeout.ToString();

    if (propertyButton.Visible == true)
        propertyButton.Hide();
    saveStatusButton.Hide();
    startCommButton.Show();

    try
    {
        //open serial port
        sp.Open();
        //set read time out to 500 ms
        sp.ReadTimeout = 500;
    }
    catch (System.Exception ex)
    {
        baudRateLabel.Text = ex.Message;
    }
}
}
}

```

## Code for Property Dialog

Hide Shrink ▲ Copy Code

```
#region Using directives
```

```

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

```

```
#endregion
```

```
namespace Serialexample
```

```

{
    partial class PropertyPage : Form
    {

```

```

        //variables for storing values of baud rate and stop bits
        private string baudR="";
        private string stopB="";

```

```

        //property for setting and getting baud rate and stop bits
        public string bRate
        {
            get

```

```

        {
            return baudR;
        }
        set
        {
            baudR = value;
        }
    }

    public string sBits
    {
        get
        {
            return stopB;
        }
        set
        {
            stopB = value;
        }
    }

    public PropertyPage()
    {
        InitializeComponent();
    }

    private void cancelButton_Click(object sender, EventArgs e)
    {
        this.bRate = "";
        this.sBits = "";
        //close form
        this.Close();
    }

    private void okButton_Click_1(object sender, EventArgs e)
    {
        //here we set the value for stop bits and baud rate.
        this.bRate = BaudRateComboBox.Text;
        this.sBits = stopBitComboBox.Text;
        //
        this.Close();
    }
}
}

```

## Note:

- Both receiver and the transmitter must agree on the number of data bits, as well as the baud rate.
- All the connection for the DB 9 connector should be done accordingly.
- The article is based on pre-released documentation (.NET Framework 2.0 Beta) and is subject to change in future release.

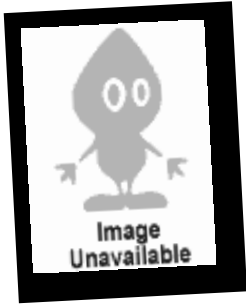
## License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](http://www.codeproject.com/Articles/8605/Serial-Communication-using-C-and-Whidbey)



[Share](#)[EMAIL](#)[TWITTER](#)

## About the Author



**Tapan Dantre** No Biography provided

Web Developer

India

## You may also be interested in...

[How-To Intel® IoT Technology Code Samples: Robot arm in C++](#)

[How-To Intel IoT Code Samples: Watering System](#)

[Capturing Customer Information from Driver's Licenses using LEADTOOLS](#)

[SAPrefs - Netscape-like Preferences Dialog](#)

[How-To Intel® IoT Technology Code Samples: Access control in C++](#)

[Generate and add keyword variations using AdWords API](#)

## Comments and Discussions

**Search Comments**[First](#) [Prev](#) [Next](#)