Communicate with Serial Port in C#

-
- Ryan Alford
-
- Feb 11, 2023
-
- 1.6m
- 60
- 24
-

SerialPortCommunication.zip

Download .NET & JAVA Office File API For Free

The SerialPort class in C# allows you to communicate with a serial port in .NET. This article will demonstrate how to write and receive data from a device connected to a serial port in C# and .NET. We will be writing the received data to a TextBox on a form, so this will also deal with threading.

In the past, to communicate with a Serial Port using .NET 1.1, you had to either use the Windows API or third-party control. With .NET 2.0 and above, Microsoft has added this support with the inclusion of the SerialPort class as part of the System.IO.Ports namespace. Implementation of the SerialPort class is very straightforward. To create an instance of the SerialPort class, you pass the SerialPort options to the class's constructor.

```
// all of the options for a serial device
// ---- can be sent through the constructor of the SerialPort class
// ---- PortName = "COM1", Baud Rate = 19200, Parity = None,
// ---- Data Bits = 8, Stop Bits = One, Handshake = None
```

```
SerialPort _serialPort = new SerialPort("COM1", 19200, Parity.None, 8,
StopBits.One);
_serialPort.Handshake = Handshake.None;
```
C#
Copy

To receive data, we will need to create an EventHandler for the "SerialDataReceivedEventHandler":

```
// "sp_DataReceived" is a custom method that I have created
_serialPort.DataReceived += new
SerialDataReceivedEventHandler(sp_DataReceived);
```
C#
Copy

You can also set other options, such as the ReadTimeout and WriteTimeout.

```
// milliseconds _serialPort.ReadTimeout = 500;
_serialPort.WriteTimeout = 500;
```
C#
Copy

Once you are ready to use the Serial Port, you will need to open it:

```
// Opens serial port
_serialPort.Open();
```
C#
Copy

Now, we are ready to receive the data. However, to write this data to the TextBox on a form, we need to create a delegate. .NET does not allow cross-thread action, so we need to use a delegate. The delegate writes to the UI thread from a non-UI thread.

```
// delegate is used to write to a UI control from a non-UI thread
private delegate void SetTextDeleg(string text);
```
C#
Copy

We will now create the "sp_DataReceived" method that will be executed when data is received through the serial port,

```
void sp_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    Thread.Sleep(500);
    string data = _serialPort.ReadLine();
    // Invokes the delegate on the UI thread, and sends the data that
was received to the invoked method.
    // ---- The "si_DataReceived" method will be executed on the UI
thread, which allows populating the textbox.
    this.BeginInvoke(new SetTextDeleg(si_DataReceived), new object[] {
data });
}
```

C#

Copy

Now we create our "si_DataReceived" method,

```
private void si_DataReceived(string data) { textBox1.Text =
data.Trim(); }
```

C#

Copy

We can now receive data from a serial port device and display it on a form. Some devices will send data without being prompted. However, some devices need to send certain commands, and it will reply with the data the command calls for. For these devices, you will write data to the serial port and use the previous code to get the data that will be sent back. In my example, I will be communicating with a scale. For this particular scale, sending the command "SI\r\n" will force it to return the weight of whatever is on the scale. This command is specific for this scale. You will need to read the documentation of your serial device to find commands that it will receive. To write to the serial port, I have created a "Start" button on the form. I have added code to its Click_Event:

```
private void btnStart_Click(object sender, EventArgs e)
{
    // Makes sure serial port is open before trying to write
    try
    {
        if(!(_serialPort.IsOpen))
        _serialPort.Open();
        _serialPort.Write("SI\r\n");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error opening/writing to serial port :: " +
ex.Message, "Error!");
    }
}
```

C#

Copy

And that is all you need to do. I have attached the Visual Studio 2005 solution.

How to receive data from com port

Here is the complete application on how to receive data from a COM port in 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("Handshake({0}):", 
defaultPortHandshake.ToString());
        handshake = Console.ReadLine();

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

        return (Handshake)Enum.Parse(typeof(Handshake), handshake);
    }
}
```
C#
Copy

Summary

This article taught you how to communicate with a COM port in C#. You also learned how to receive data from a COM port using C#.