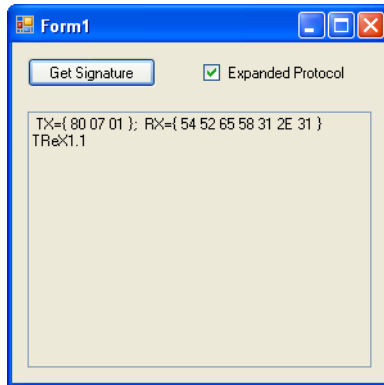


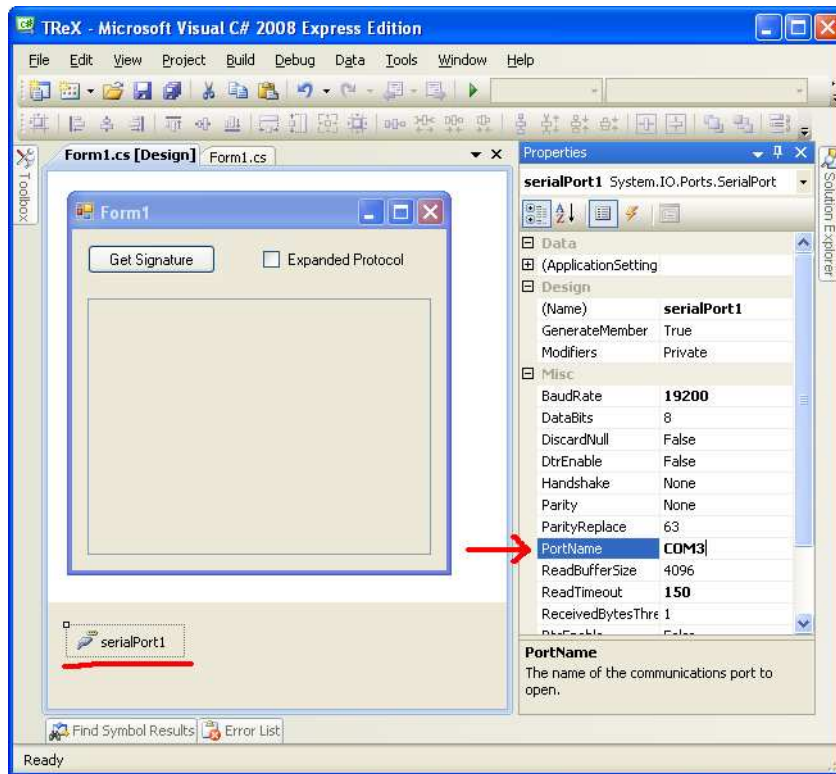
Sample C# Program for Communicating with the TReX and TReX Jr



The following program, written using Visual C# 2008, provides a simple example of how you can send and receive data from the **TReX** [<http://www.pololu.com/catalog/product/777>] or **TReX Jr** [<http://www.pololu.com/catalog/product/767>] using **Microsoft's visual C#** [<http://www.microsoft.com/express/vcsharp/>] and your computer's serial port. It sends the TReX a "get signature" command when the **Get Signature** button is pressed and displays the response from the TReX in the text box.

The `sendCommandPacket` method is a simplified version of the one used by the **TReX Configurator utility** [http://www.pololu.com/file/download/TReXConfigurator.zip?file_id=0J58] (297k zip).

You can download the project files here: **trex.zip** [http://www.pololu.com/file/download/TReX.zip?file_id=0J186] (60k zip). Don't forget to configure the `serialPort1` object's `portName` property for the COM port to which your TReX or TReX Jr is connected.



Sample C# TReX program: configuring the SerialPort object.

```

using System;
using System.Windows.Forms;

namespace TReX
{
    public partial class Form1 : Form
    {
        int deviceNum = 7; // default device number for the TReX and TReX Jr
        bool serialEcho; // does the TReX echo everything you transmit? (true for RS-232)
        byte[] buffer; // array to hold our command packets and received data

        public Form1()
        {
            InitializeComponent();
            buffer = new byte[32];
        }

        private bool sendCommandPacket(byte[] packet, int sendBytes, int readBytes, ref string str)
        {
            /* This function transmits the specified command packet and then waits to receive
            * the specified number of bytes in response. Bytes to transmit are provided via the
            * 'packet' array, and received bytes are stored in the 'packet' array.
            * It blocks execution until the desired number of bytes have been received from the
            * the TReX or until the serial read times out (150 ms).
            * packet - an array of bytes to transmit to the TReX; when the function is through,
            * this array will contain any bytes received from the TReX, so the size of
            * this array must be greater than or equal to Max(sendBytes, readBytes)
            * sendBytes - the number of bytes in the command packet
            * readBytes - the number of bytes to try to receive from the TReX in response
            * str - a string that contains information about what was sent and received;
            * used for debugging/feedback purposes
            */

            int i;
            str = "";

            try
            {

```

```

// if there are any unread bytes in the read buffer, they are junk
// read them now so the buffer is clear to receive anything the TReX
// might send back in response to the command
while (serialPort1.BytesToRead > 0)
    serialPort1.ReadByte();

str += " TX={ ";
if (expandedProtocolCheckBox.Checked)
{
    for (i = sendBytes - 1; i >= 0; i--)
        packet[i + 2] = packet[i];
    packet[0] = 0x80;
    packet[1] = (byte)deviceNum;
    packet[2] -= 0x80;    // clear MSB of command byte
    sendBytes += 2;
}

for (i = 0; i < sendBytes; i++)
    str += packet[i].ToString("X2") + " ";
serialPort1.Write(packet, 0, sendBytes);
if (serialEcho)
{
    str += "; Echo={ ";
    for (i = 0; i < sendBytes; i++)
        str += serialPort1.ReadByte().ToString("X2") + " ";
}
str += "; RX={ ";
for (i = 0; i < readBytes; i++)
{
    packet[i] = (byte)serialPort1.ReadByte();
    str += packet[i].ToString("X2") + " ";
}
str += " ";
}

catch (Exception)
{
    str += " *TIMEOUT*";
    return false;
}

return true;
}

private void signatureButton_Click(object sender, EventArgs e)
{
    try
    {
        logTextBox.Text = "";
        serialPort1.Open();
        buffer[0] = 0xFF;
        serialPort1.Write(buffer, 0, 1);    // send the null command
        try
        {
            int data = serialPort1.ReadByte();    // look for an echo
            if (data == 255)
                serialEcho = true;    // we get here if we are using RS-232
        }
        catch (TimeoutException)
        {
            serialEcho = false;    // we get here if we are using TTL serial
        }

        // read the device signature
        string str = "";
        buffer[0] = 0x81;
        if (!sendCommandPacket(buffer, 1, 7, ref str))
            logTextBox.Text += "failure\r\n" + str;
        else
        {
            logTextBox.Text += "" + str + "\r\n";
            for (int i = 0; i < 7; i++)
                logTextBox.Text += (char)buffer[i];
        }
    }
    catch (Exception)
    {
    }
}

```

```
        logTextBox.Text += "Exception!";
    }
    if (serialPort1.IsOpen)
        serialPort1.Close();
    }
}
```