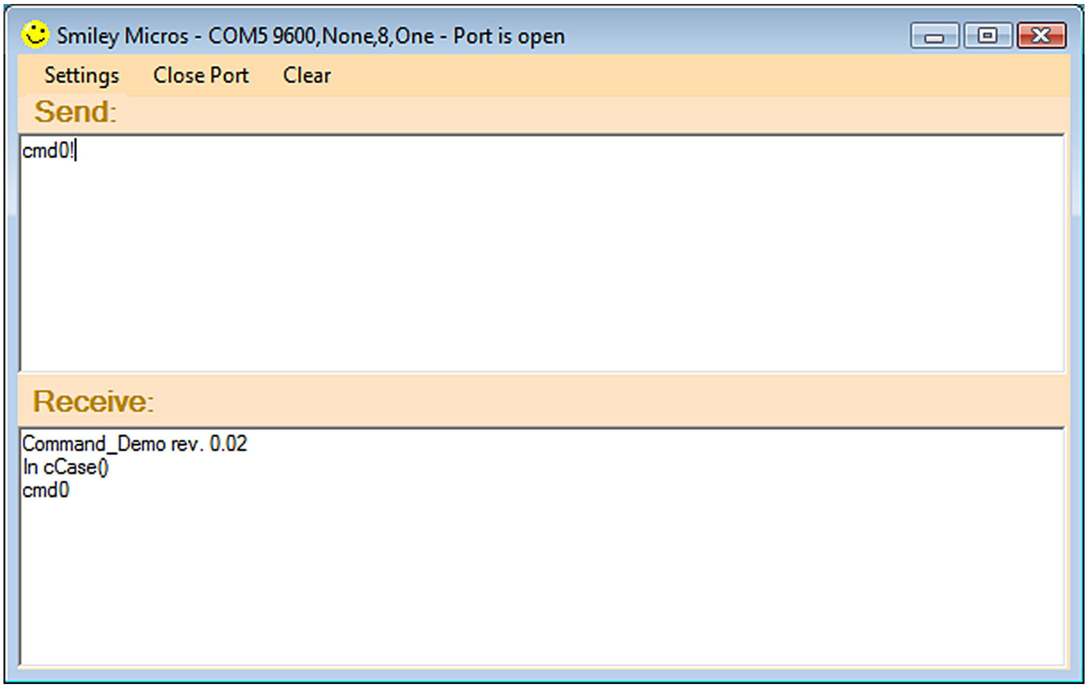
Smiley's Workshop 18: Serial Communications Part 1 – Graphical User Interfaces

Joe Pardue November 9, 2009



```
cmd0!
```

```
Command_Demo rev. 0.02
In cCase()
cmd0
```

Figure 1: A Simple Terminal

Recap

Last month we finished with our introduction to the Arduino Projects Kits (available from Nuts&Volts and Smiley Micros) and used all the parts to do some interesting things. This month we will start to look a bit deeper into communicating between the Arduino and a PC. We will begin a three part series about serial communications between a PC and a microcontroller. In the first part we will learn a bit about the virtual serial port used in the Arduino and introduce writing programs on the PC using the free Microsoft C# and Visual Basic Express .NET applications and use this to build Graphical User Interfaces. In the second part, next month, we will apply this knowledge to build a Simple Terminal program. And the month after that, in the third part we will build a GUI for an Arduino Volt Meter.

SMILEY'S WORKSHOP

AVR MICROCONTROLLER: C PROGRAMMING - HARDWARE - PROJECTS

Smiley's Workshop 18: Serial Communications Part 1 – Graphical User Interfaces

Virtual Serial Port Introduction

Why Imitate a PC Serial Port with USB?

In the olden days the PC serial port had Windows© driver and hardware link based on the RS232 electrical specification and used a DB-9 connector and a UART (Universal Asynchronous Receiver Transmitter). It wasn't exactly simple for a novice PC user to hook up a serial link since it required the user to select software interrupts and set hardware jumpers - something you and I, as certifiable nerds, like to do - but something that normal people hate and tend to mess up. These and other complications led to the development of the Plug and Play initiative (more commonly and correctly known as Plug and Pray). One part of all this was to replace the serial port with USB to help simplify things, which it did for the user, but made the developer's life (yours and mine) MUCH more complicated. USB also obsoletes lots of perfectly good serial devices and a couple of decades of knowledge of how to do robust RS232 style serial communications between PCs and external serial devices.

I've read the USB specification, and I'm here to testify (brother amen) that the old ComPort/UART/RS232 was a piece of cake compared to USB. I worked with USB when it first came out and my brain is worse for the wear, but fortunately some genius's thought that they would simplify the life of not just the user but the developer by creating a transitional concept: to have an old fashioned RS232 style serial port that runs over USB. The FTDI folks call this a Virtual Communications Port (VCP). It allows legacy applications to continue to use the old microcontroller code and the windows COM port software with the USB part all tidily bound up in a black box that the developer doesn't have to open. These transitional devices give us the best of both worlds: the ease of using the serial (COM) port and the ubiquity of using USB. The developer has the option of adding RS232 level converts to completely emulate the old way of doing things, or leaving the level converters off and outputting voltage levels directly compatible with a microcontroller's UART. The latter is exactly what the Arduino does to allow serial communications with a PC

The FTDI chip (FT232R) used on the Arduino is the same as that used on the BBUSB that was discussed in detail in the article "The Serial Port is Dead, Long Live the Serial Port' by yours truly in the June 2008 issue of Nuts&Volts. You can also get the book "Virtual Serial Programming Cookbook" (also by yours truly) and an associated projects kit from either Nuts&Volts or Smiley Micros. Much of the information in this article on using C# and VB for serial communications is derived from that book.

Why communicate with a PC?

A microcontroller is usually a small cheap black lump used to control something. The something could be the ignition timing on your car; the water temperature in your washing machine; the obnoxious tune (that you think is cute) on your cell phone; etc. etc.

etc… The word 'ubiquitous' seems almost invented to describe the current uses of microcontrollers: they really are everywhere.

Most often the microcontroller knows how to do its job and doesn't need any advice from you. If it does deign to accept suggestions, you usually give them via a button (snooze alarm) or touch pad (microwave oven). But some of us, mostly students, hobbyists, and developers, want to spend a lot of time in conversation with a microcontroller. This may be because we want to learn how it works, or get it to follow complex commands that aren't easy to give with buttons, or because we are designing a complex system and we need direct access to the micro's brains while we are trying to figure out why things aren't working like they are supposed to. If you are one of these folks then you might see the benefit of being able to use the vast resources of a PC (if you are using Windows©, perhaps 'half-vast' would be more appropriate?) to carry on a conversation with a microcontroller. We will look at some things to help you to do this.

Why use C# and Visual Basic Express .NET ?

One good reason for selecting either language in Express .NET is because both are free. You like free don't you? I decided to present the software in both C# and Visual Basic Express.NET because there are lots of folks who program in one language and think the other language is the vile realm of Hell bound heretics. However, the concepts that are being programmed are the same no matter what language is being used (religious wars aside), examples are done in both languages with the C# example shown here, and the Visual Basic example included in Workshop18.zip. Just skip over the language you don't like.

You will find a treasure trove of free tools and learning materials at: http://www.microsoft.com/express/. You will notice that Microsoft says that both C# and Visual Basic Express are 'easy to use', 'fun', and 'easy to learn'. Well… maybe not 'easy' by most of our definitions, but, C# and Visual Basic are all three of those things compared to earlier methods of creating a Graphical User Interface (GUI – pronounced gooey) to communicate with external serial devices. However, this is a little like saying a root-canal is 'easy' or 'fun' compared to thumbscrews. Ouch!

If you are entirely new at programming GUIs on Windows then I'd like to suggest that you consider using C#, since it is similar to C which is likely the language you will choose for the other side of the cable: the microcontroller, which I personally prefer for many religious reasons, including having written a book on the subject. If you are already a microcontroller BASIC fan or have used VB on a PC, then you will probably want to use the Visual Basic Express, **BUT** do be prepared for a lot of culture shock. The IDE shown in many following illustrations will be for C#, but the Visual Basic IDE is virtually identical, so it shouldn't be difficult to transpose the concepts.

Smiley's Workshop 18: Serial Communications Part 1 – Graphical User Interfaces

The Microsoft learning resources are really great and free! I already knew how to program in C# and thought I'd skim some of it as a review. Instead I went through 16 hours of free video and learned a lot (mainly that I didn't know as much as I thought I did). For our purposes, you don't need to view all the lessons, just the introductory materials and the parts on forms, and common controls. This is great stuff and let me repeat: it is free (shock!) What's with all the free stuff from Microsoft; did the Tin Man get a heart? So, to the tune of Wizard of Oz: *We're off to learn C# the wonderful C# of Microsoft… OR We're off to learn Visual Basic the wonderful Visual Basic of Microsoft…* Depending on which you choose. Either way, watch for flying monkeys.

While you are off learning the preliminaries, try to keep in mind the ultimate goal: We are learning to use tools that will allow us to build GUIs on a PC that we can use to communicate with microcontrollers connected to either a real serial port or a USB Virtual Serial Port.

So dog-ear this page and go learn some stuff. See you in a day or so…

Time passes… dear reader learns…

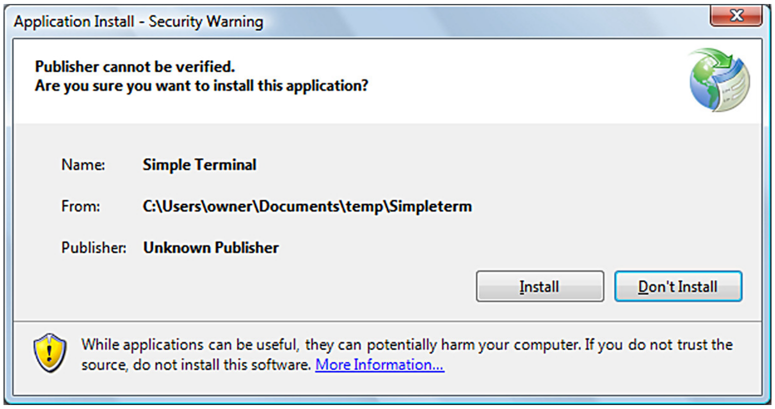
Okay, now that you can build simple forms with textboxes and buttons you are ready to look at the Simple Terminal software source code. Pay careful attention as we go through the steps to create this program. But before we write it, lets play with the finished version to get some insight into where we are going as we later write the code.

Running the Simple Terminal

The Simple Terminal is available in two forms, first is a publish version that allows you to install and run the program on your PC. The second version is the source code that loads in Visual Studio Express .NET.

Workshop18.zip
..\Simple Terminal Application
..\Arduino Command_Demo.txt
..\SimpleTermGUI_C#_Source
..\SimpleTermGUI_VB_Source

Workshop18.zip can be downloaded from the Nuts&Volts or Smiley Micros website. Unzip Workshop18.zip and open the Simple Terminal Application directory, then click on the setup application. You will see the warning shown in Figure 2 – Yet another bogus warning. Click install if you are using Vista, cuss it out if asks you if you are sure. It should open and look like Figure 1: A Simple Terminal except that the text boxes should be empty.



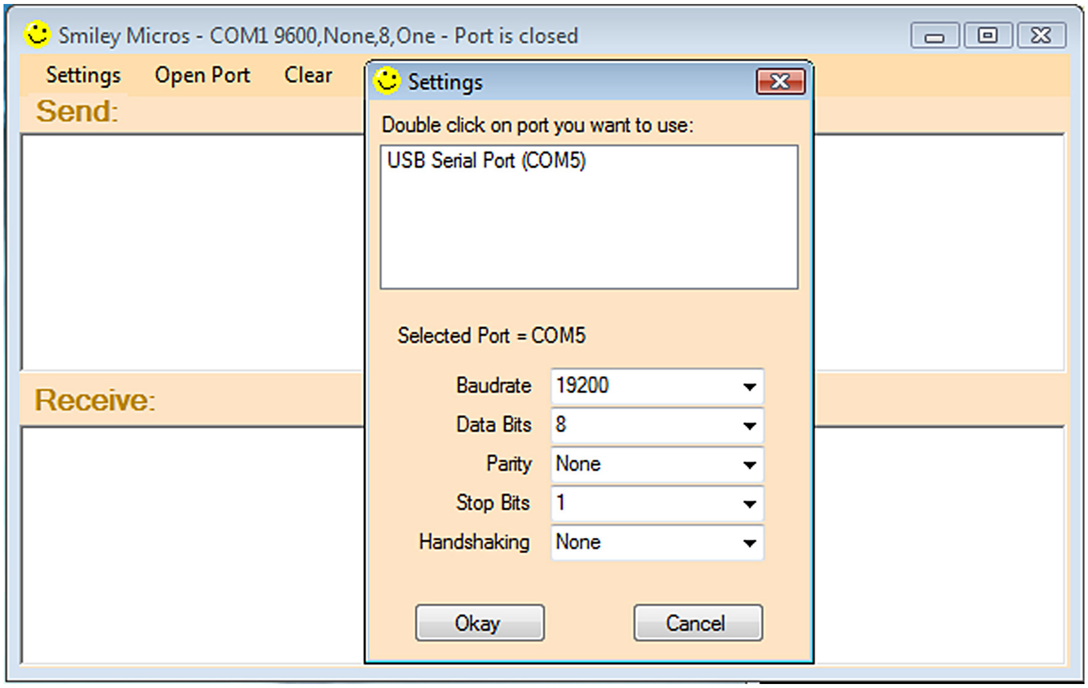
Publisher cannot be verified.
Are you sure you want to install this application?

Name: **Simple Terminal**
From: **C:\Users\owner\Documents\temp\Simpleterm**
Publisher: **Unknown Publisher**

While applications can be useful, they can potentially harm your computer. If you do not trust the source, do not install this software. More Information...

Figure 2: Yet another bogus warning.

Click on the 'Settings' menu item and in the settings dialog click the COM port connected to the Arduino shown in Figure 3: Settings. In my case the only port available is COM5 – but your Arduino will probably be on a different port so click on that one. Make sure the serial port you selected shows up in the Serial Port label ('Serial Port = COM5" – if you don't click it then the port defaults to COM1). The Arduino Command_Demo that we will be using has a Baudrate set to 19200, which is the default for the Simple Terminal so you should now click 'Okay'.


Smiley Micros - COM1 9600,None,8,One - Port is closed

Double click on port you want to use:
USB Serial Port (COM5)

Selected Port = COM5

Baudrate: 19200
Data Bits: 8
Parity: None
Stop Bits: 1
Handshaking: None

Figure 3: Settings

Next open the Arduino Command_Demo directory on your PC and then upload the Command_Demo program into the Arduino as discussed in earlier workshops (copy/paste/run). Press the reset button on the Arduino and you should see 'Command_Demo rev. 0.02' in the Receive text box. Finally, enter 'cmd0!' in the Send text box and you should see the response in the Receive box as shown in Figure 1: A Simple Terminal.

Cool… Now lets follow a recipe and cook up a Serial Terminal GUI for ourselves, shall we?

Build a PC Graphical User Interface

The Main Form

- Open C# or Visual Basic 2008 Express, they are nearly identical so all the following introduction to the IDE, though in C# Express, works equally well for either.
- From the 'File' menu select 'New Project' (Figure 4: New Project)

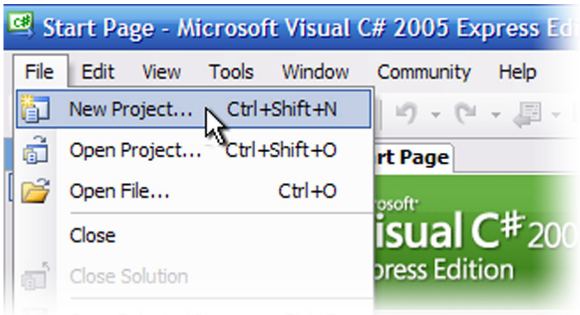
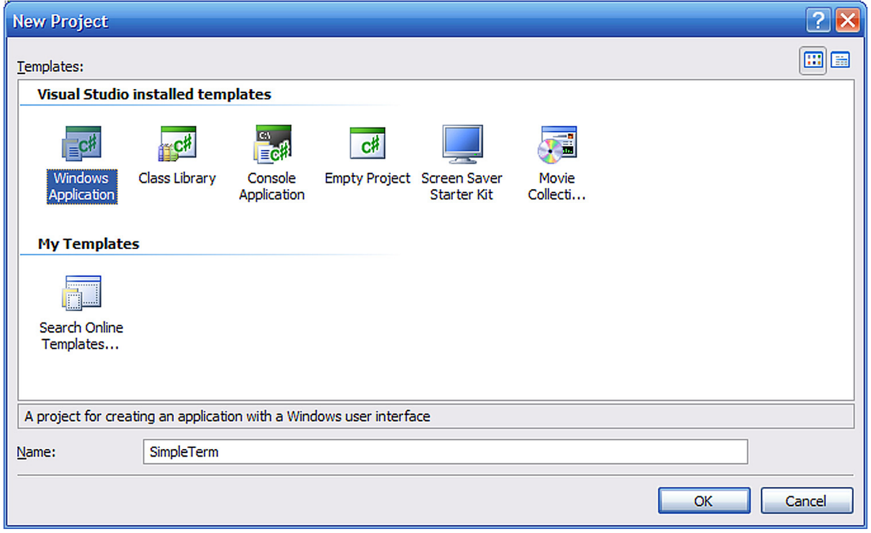


Figure 4: New Project

- In the 'New Project' form, highlight Windows Application and change the name to Simple Terminal and click the 'OK' button (Figure 5: Select Windows Application)



A project for creating an application with a Windows user interface

Name: SimpleTerm

Figure 5: Select Windows Application

- The IDE should look like Figure 6: AVRStudio IDE.

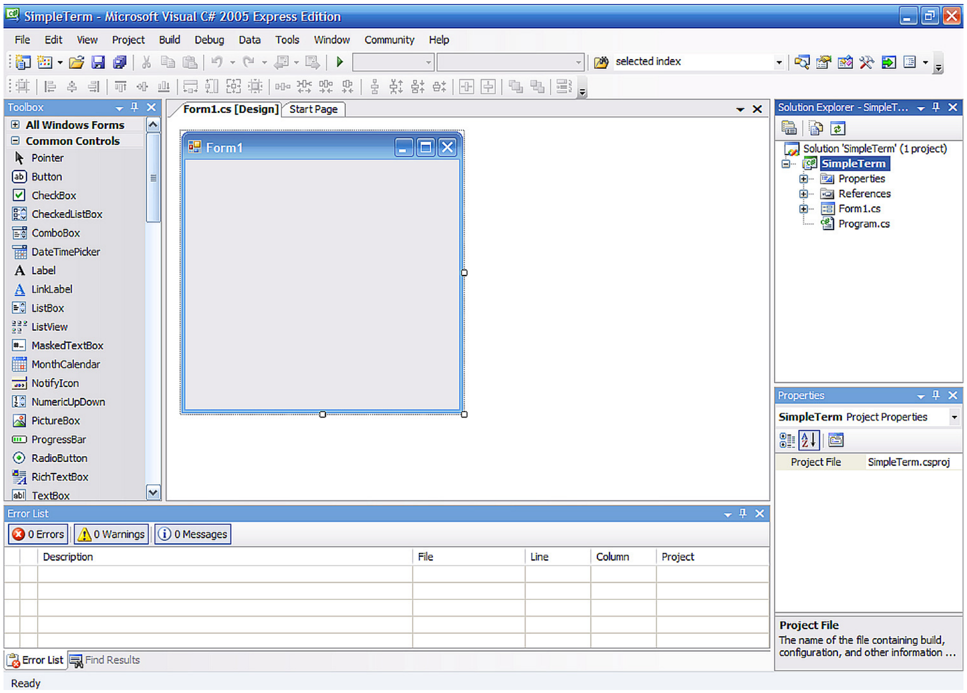
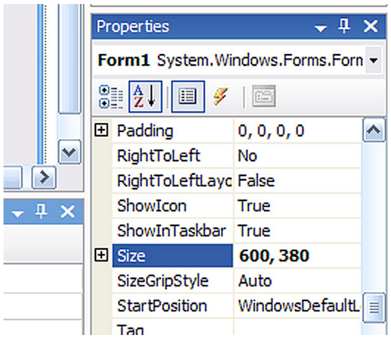


Figure 6: AVRStudio IDE

- Take note of the various panels. We will call the central panel the 'Editor Window', the left panel the 'Toolbox', the upper right panel the 'Solutions Explorer', the lower right panel the 'Properties Window'.

- In the Properties Window change the size from 300,300 to 600,380. (Figure 7: Properties)



Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayc	False
ShowIcon	True
ShowInTaskbar	True
Size	**600, 380**
SizeGripStyle	Auto
StartPosition	WindowsDefaultL
Tag	

Figure 7: Properties

- In the Toolbox click and hold the MenuStrip then drag and drop it on Form1.

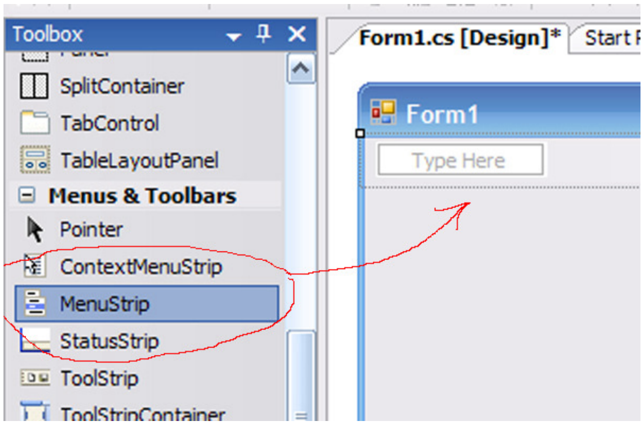


Figure 8: Add MenuStrip

- Note that the specific instance menuStrip1 of the class MenuStrip appears below Form1 in the Edit Window. (Figure 9: menuStrip1)

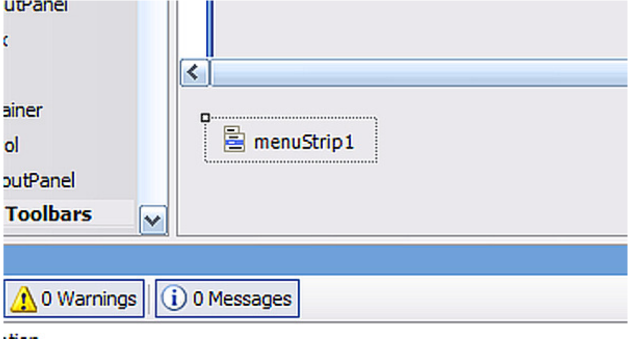


Figure 9: menuStrip1

- In the Toolbox click and hold the RichTextBox then drag and drop it on Form1. (Figure 10: Add RichTestBox)

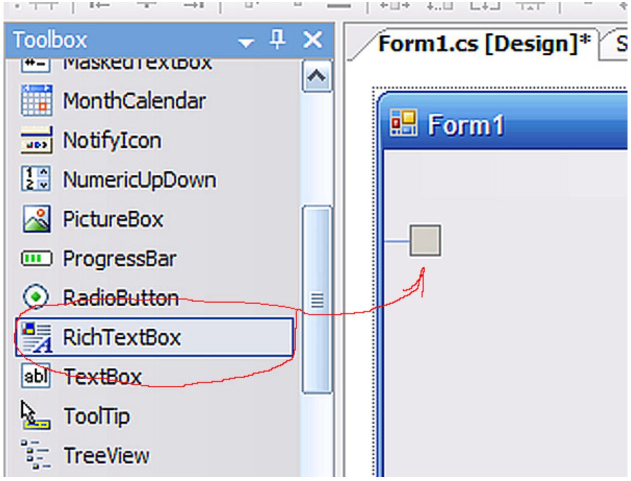


Figure 10: Add RichTestBox

- In the Properties Window change the richTextBox size from 100,96 to 590,136.
- In the Properties Window change the richTextBox location to 0,44.
- You may be wondering where I'm getting these funky dimensions. Well, I just used the cursor to size the items until they looked right, which you can also do, but if you want to get your Simple Terminal to look exactly like mine you'll need to hand-input the dimensions.
- Add another richTextBox (same size), richTextBox2, below the first. Change the location to 0,209.
- From the ToolBox select 'Label' and drag and drop it between the MenuStrip and richTextBox1.
- In the Properties Window change 'Text' from label1 to Send:
- Select a second label and drop it between richTextBox1 and richTextBox2. Change the Text from label2 to Receive:

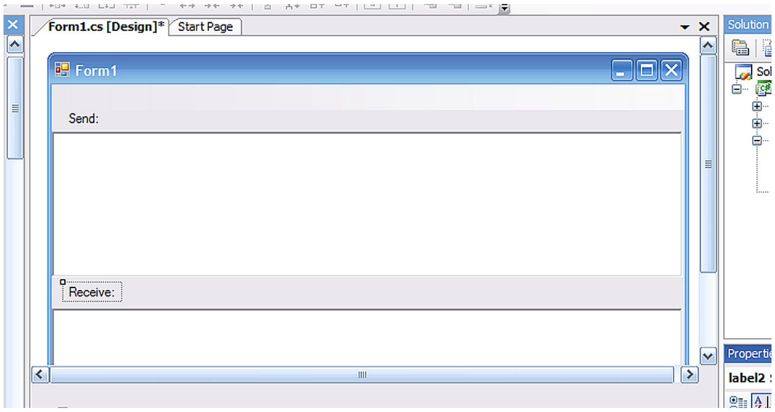


Figure 11: RichTextBoxes

- BORING! Plain old gray just won't cut it. Let's tart this up a bit. (Figure 11: RichTextBoxes)
- Select Form1 and in the Properties Window select BackColor and click the down arrow to show the color menu:

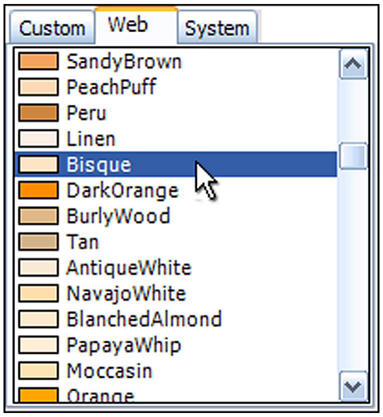
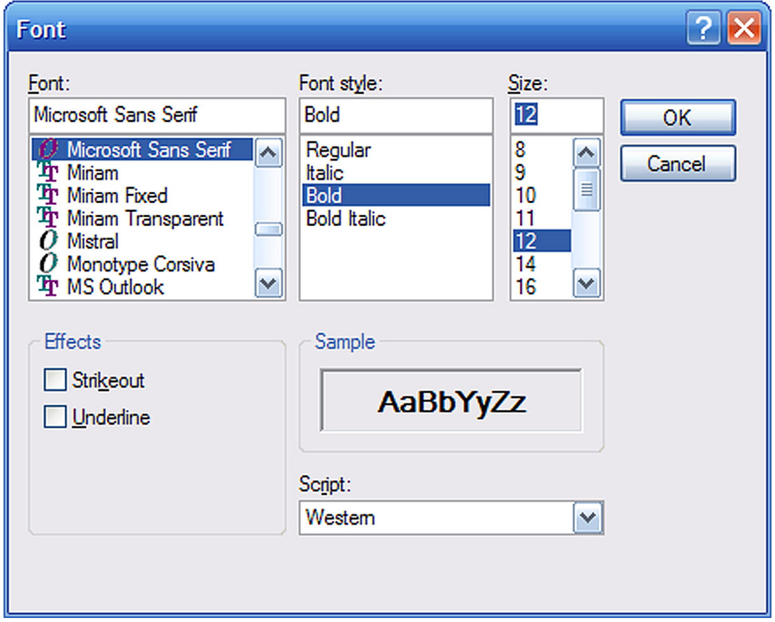


Figure 12: BackColor

- Select Bisque.
- In the Edit Window select the menuStrip1 and in the Properties Window select 'BackColor' and choose the Web color NavahoWhite. (Figure 12: BackColor)
- Select label1 and in the Properties Window choose Font. Change Font Style to Bold and Size to 12. (Figure 13:Font)



Font: Microsoft Sans Serif
Font style: Bold
Size: 12
Script: Western

Figure 13: Font

- In the Properties Window for label1 select ForeColor and change to Web DarkGoldenrod.

- Select Form1 and in the Properties Window change text to 'Smiley Micros – Simple Terminal' or some other less commercial name if you prefer.
- Still in the Form1 Properties Window select Icon and then select 'Smiley.ico' (located in the \Software\Graphics\ directory).

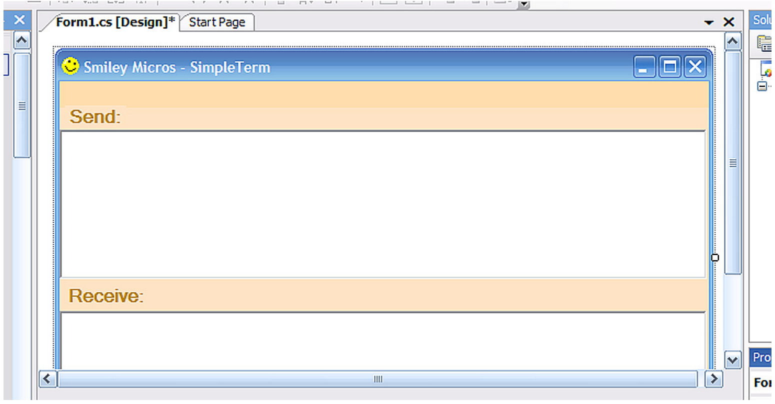


Figure 14: GUI with makeup

- Now you've just got to admit that a little makeup helps. (Figure 14: GUI with makeup)
- Select the MenuStrip and highlight the 'Type Here' box. (Figure 15: Add menu item)

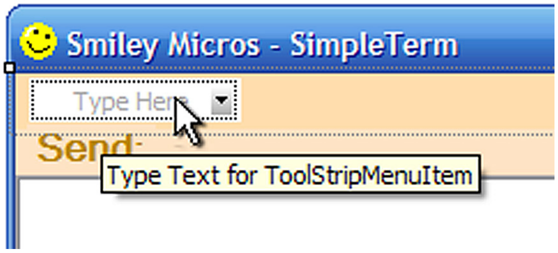


Figure 15: Add menu item

- Type Settings.
- Move your cursor to the next menu position to the right, type Open Port.
- And in the next right type Clear. (Figure 16: Menu items)

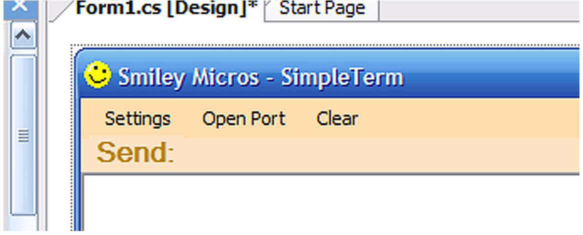


Figure 16: Menu items

- **NOW SAVE YOUR WORK! (aka Save Your Butt) –** Do this every time you have done enough work that you'd feel bad if you lost it. If you didn't already know this, you will learn it the hard way like the rest of us.

- In the Express IDE menus, select Debug and click Start Debugging. (Figure 17: Start debugging)

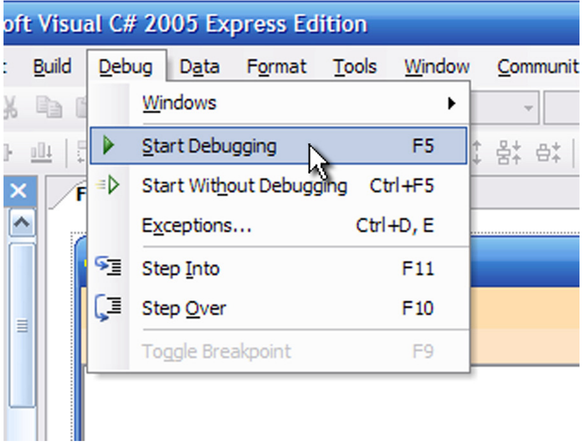


Figure 17: Start debugging

- Whoa, look at that! **<u>You didn't write a word of software</u>** and yet you just created a GUI for a serial terminal! Move your mouse over the menu items and notice how they change colors. This is not just pretty, but pretty useless as is. So let's make it useful.
- Click the debug form's close button (the little X in the upper right).

Making the Simple Terminal GUI do something

Add functionality to the menu items

- This is what we see in the C# Edit Window:

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

namespace WindowsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

- In the IDE double click on the menuStrip1 Settings button.
- In the Edit Window click the 'Form1.cs [Design] tab to view the Design Editor panel. Now click the 'Open Port' menu item.
- Repeat for the 'Clear' menu item.

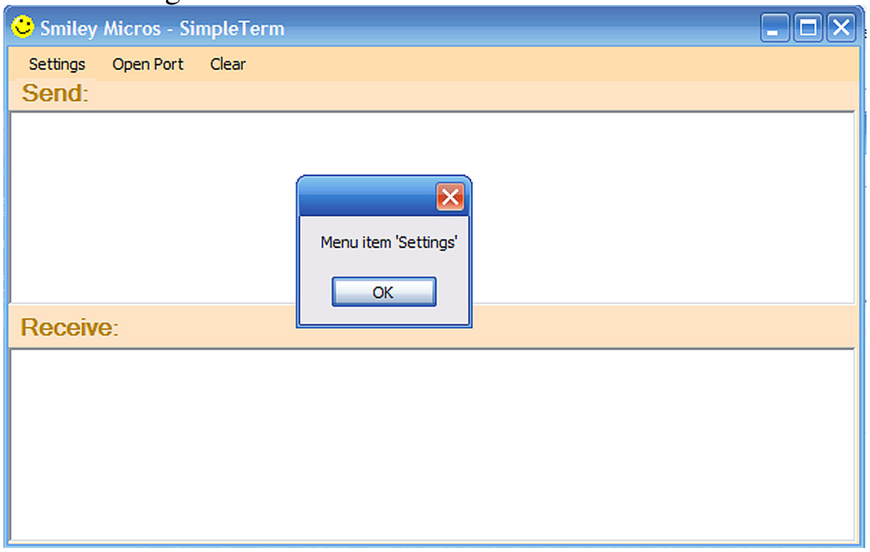
- In the Form1.cs code window you'll see that the IDE has created three functions that will be run when you click the menu item.
- In C# add the following text to each:

```
private void settingsToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Menu item 'Settings'");
}

private void openPortToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Menu item 'Open Port'");
}

private void clearToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Menu item 'Clear'");
}
```

- Run the program in debug mode again (Debug/Start Debugging)
- Click the 'Settings' menu item.



Menu item 'Settings'

Figure 18: Click the Settings menu item

- Likewise test the 'Open Port' and 'Clear' menu items.
- Close the debug form.
- Select Form1.cs and change the Clear menu function to:
- In C# add:

```
private void clearToolStripMenuItem_Click(object sender, EventArgs e)
{
    richTextBox1.Text = "";
    richTextBox2.Text = "";
}
```

- This will cause the text in the rich text boxes to be cleared.
- Run the program in Debug mode and type some text in each richTextBox then click the Clear menu item to see it work.

SMILEY'S WORKSHOP

AVR MICROCONTROLLER: C PROGRAMMING - HARDWARE - PROJECTS

Smiley's Workshop 18: Serial Communications Part 1 – Graphical User Interfaces

- The source code is in the \Software\Chapter 4 - Simple Terminal GUI \ directory

If you are anxious to get a jump start on next month's article, you can get the details for building the Simple Terminal in the free first six chapters of '*Virtual Serial Port Cookbook'* from www.smileymicros.com, or if you find yourself interested in the FTDI FT232R chip used in the Arduino for serial communications then you can purchase the book and a projects kit from Nuts&Volt or Smiley Micros. I will start a thread on www.avrfreaks.net titled 'Smiley's Simple Terminal 'and I'll try to check by several times a week to see if there are any questions.

In this workshop we went thru the Microsoft tutorials and learned just enough to be dangerous. We built the GUI for a Simple Terminal. In the next workshop we will build a dialog form (Settings) to get data from the user for selecting a serial port and setting up the UART. We will then learn to use the Serial Port Class, and finally we will build an Arduino Volt Meter that displays output on a PC.

The Arduino Projects Kit:

Smiley Micros and Nuts&Volts are selling a special kit: The Arduino Projects Kit. Beginning with Workshops 9 we started learning simple ways to use these components, and in later Workshops we will use them to drill down into the deeper concepts of C programming, AVR microcontroller architecture, and embedded systems principles.

With the components in this kit you can:

- Blink 8 LEDs (Cylon Eyes)
- Read a pushbutton and 8-bit DIP switch
- Sense Voltage, Light, and Temperature
- Make Music on a piezo element
- Sense edges and gray levels
- Optically isolate voltages
- Fade LED with PWM
- Control Motor Speed
- And more…

LINKS: You can find the source code in Workshop18.zip on the Nuts&Volts and Smiley Micros websites.