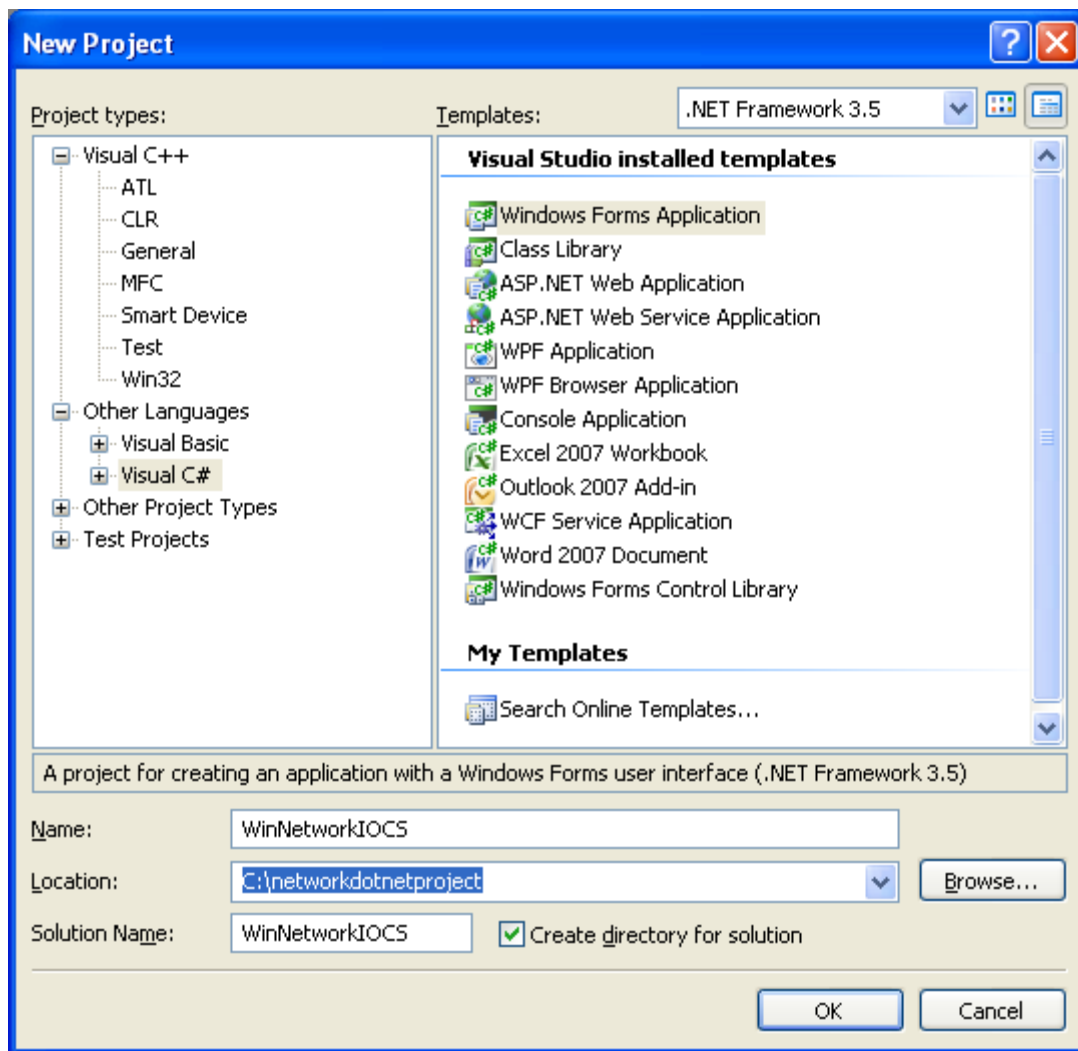
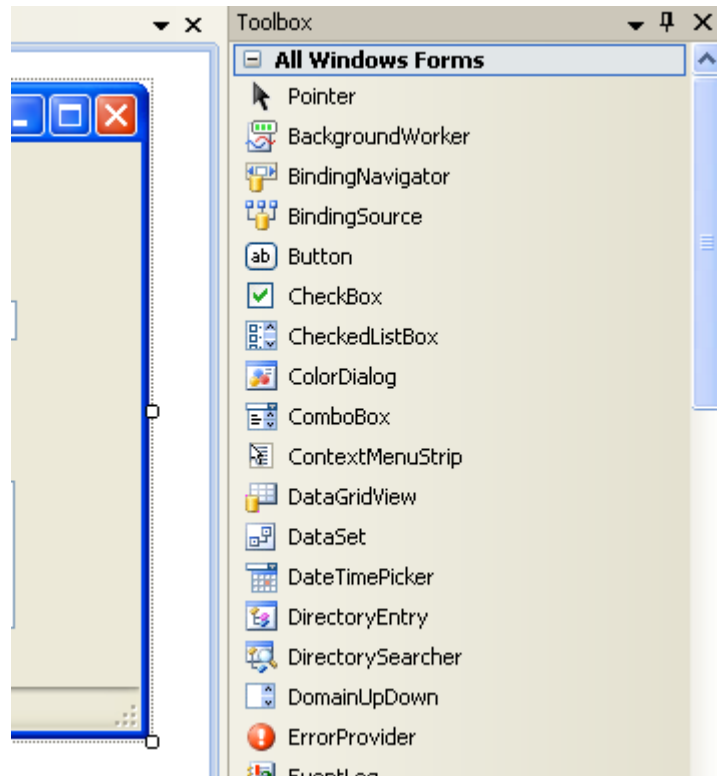


## C# WinForm Program Example

Create a new Windows Form Application project.

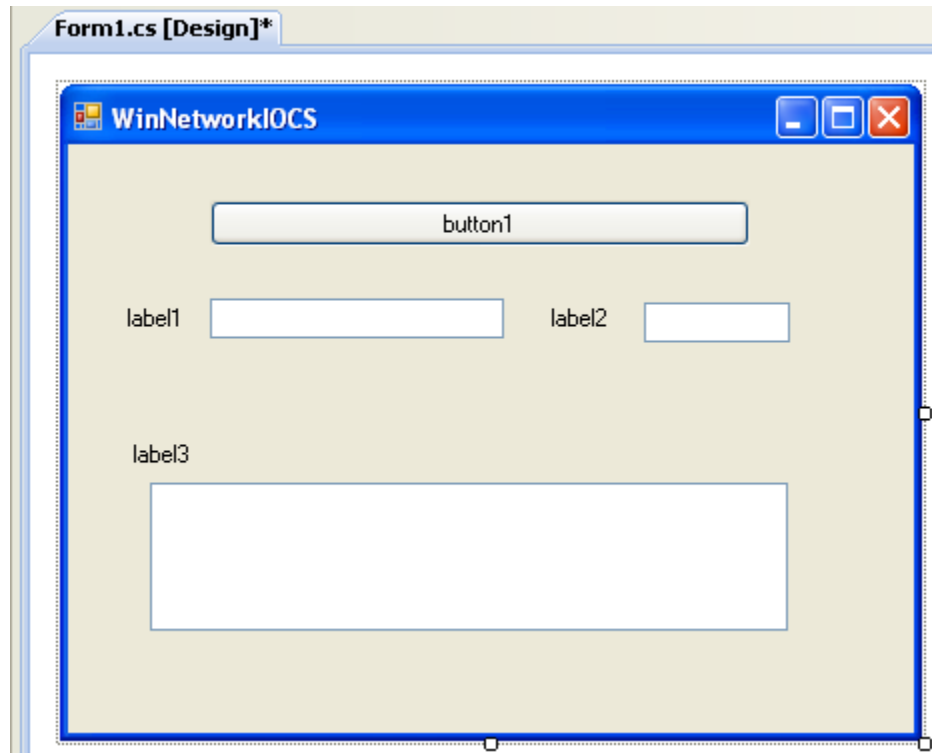


Drag, drop and rearrange the related controls from the Toolbox window and customize the properties.

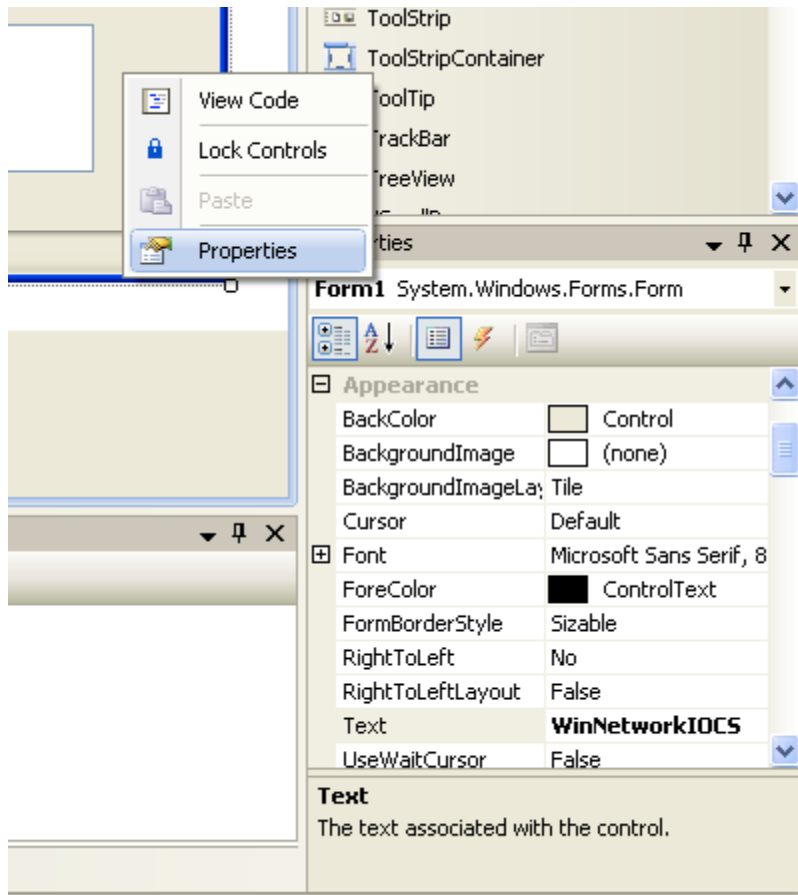


Customize the form using the following information.

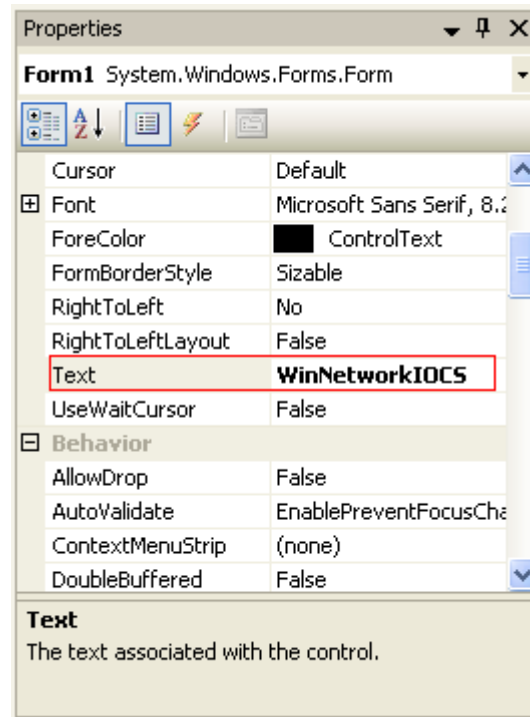
Control	Property	Value
Form1	Text	WinNetworkIOCS
Button	Text	Connect, Send, Close
	Name	ConnectButton
Label1	Text	IP Address:
TextBox1	Name	IPAddressBox
	Text	127.0.0.1
Label2	Text	Port:
TextBox2	Name	PortBox
	Text	5150
Label3	Text	Message to send:
TextBox3	Name	SendMessageBox
	Text	This is a test message...
	Multiline	True



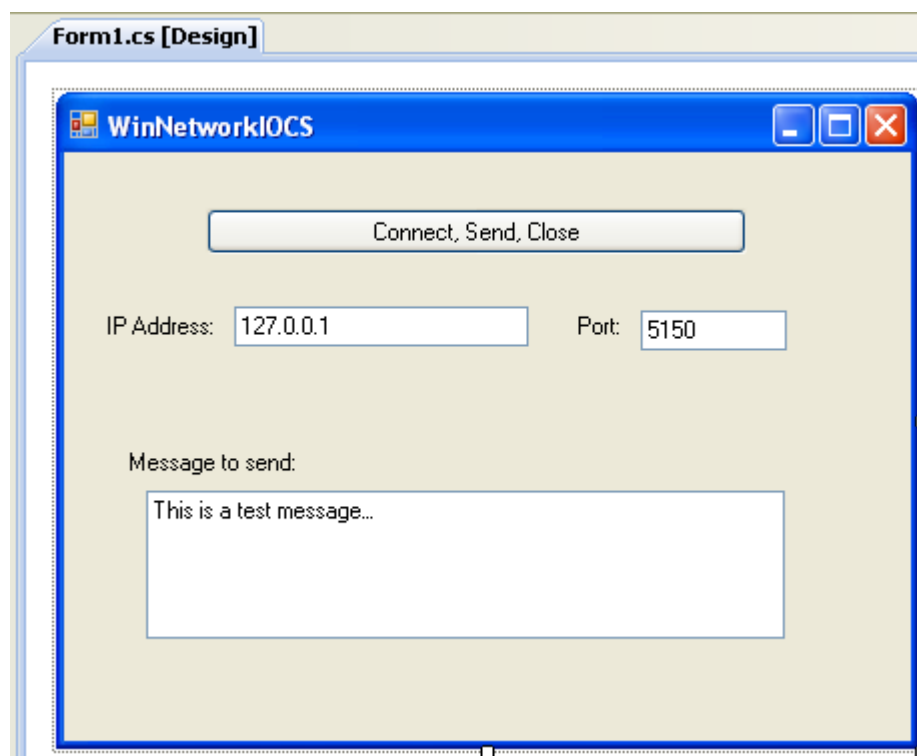
To invoke the Properties page (if any), select object > Right click mouse > Select Properties context menu. If the Properties page already docked in the VS IDE, then selecting the object will bring the Properties page for the object automatically.



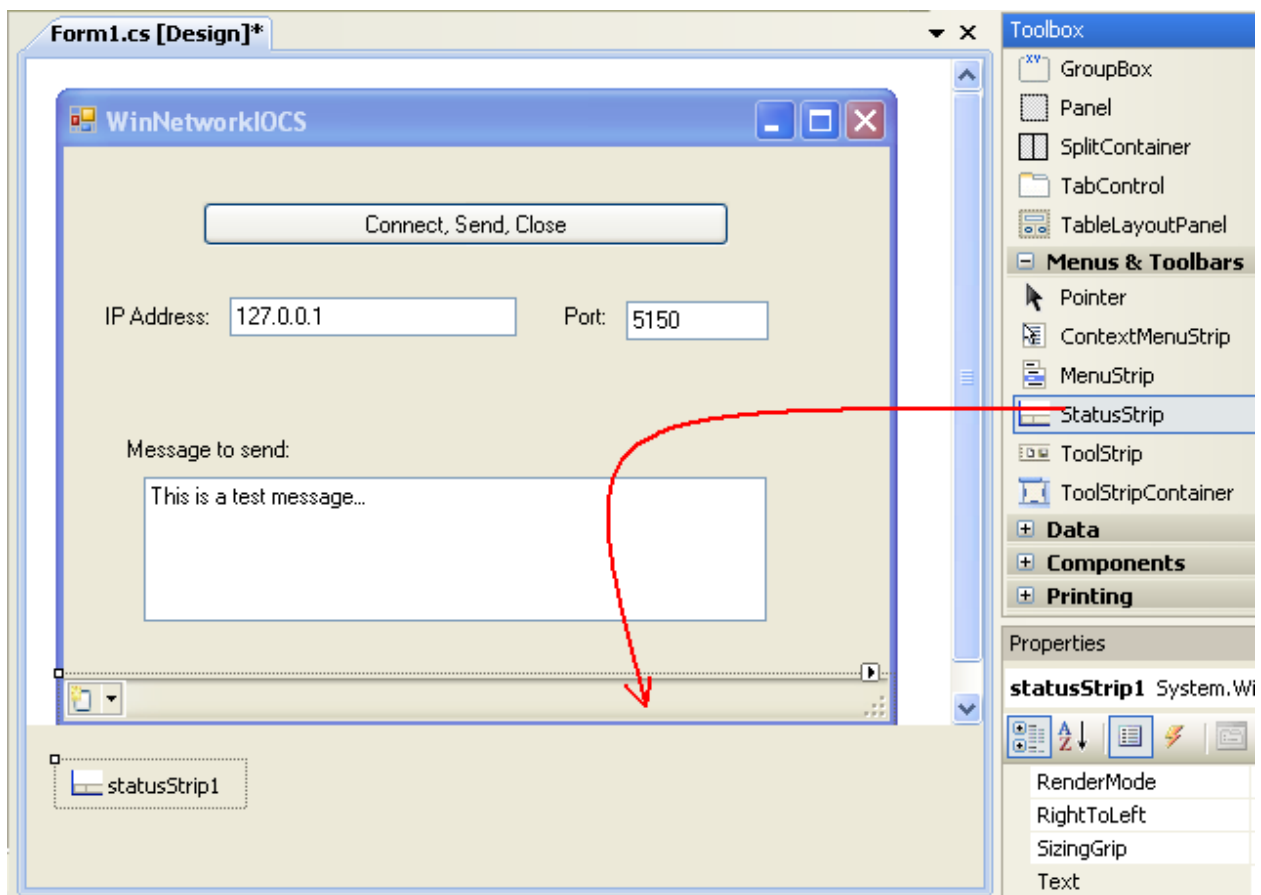
The following is the Properties page for Form1 example.



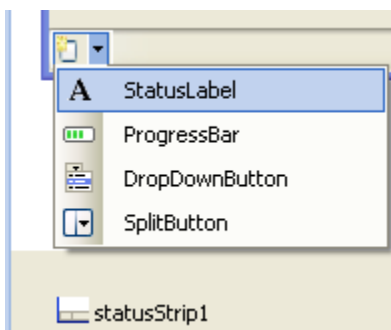
The following Figure shows the almost complete Windows form.



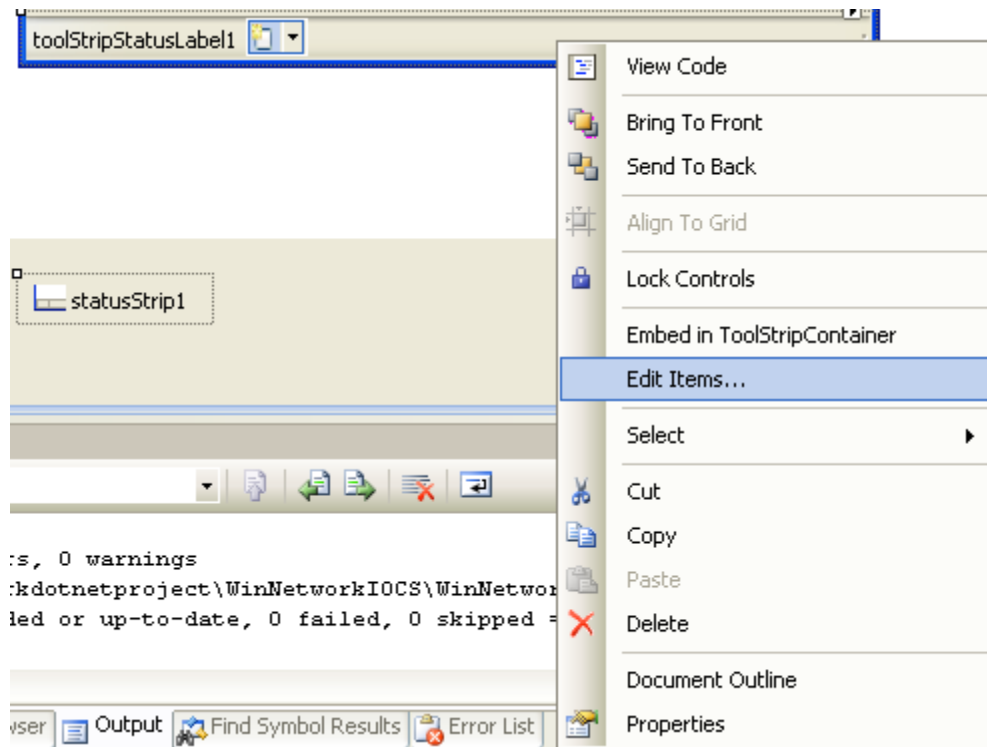
Next, drag and drop a StatusStrip container component at the bottom of the Windows form.



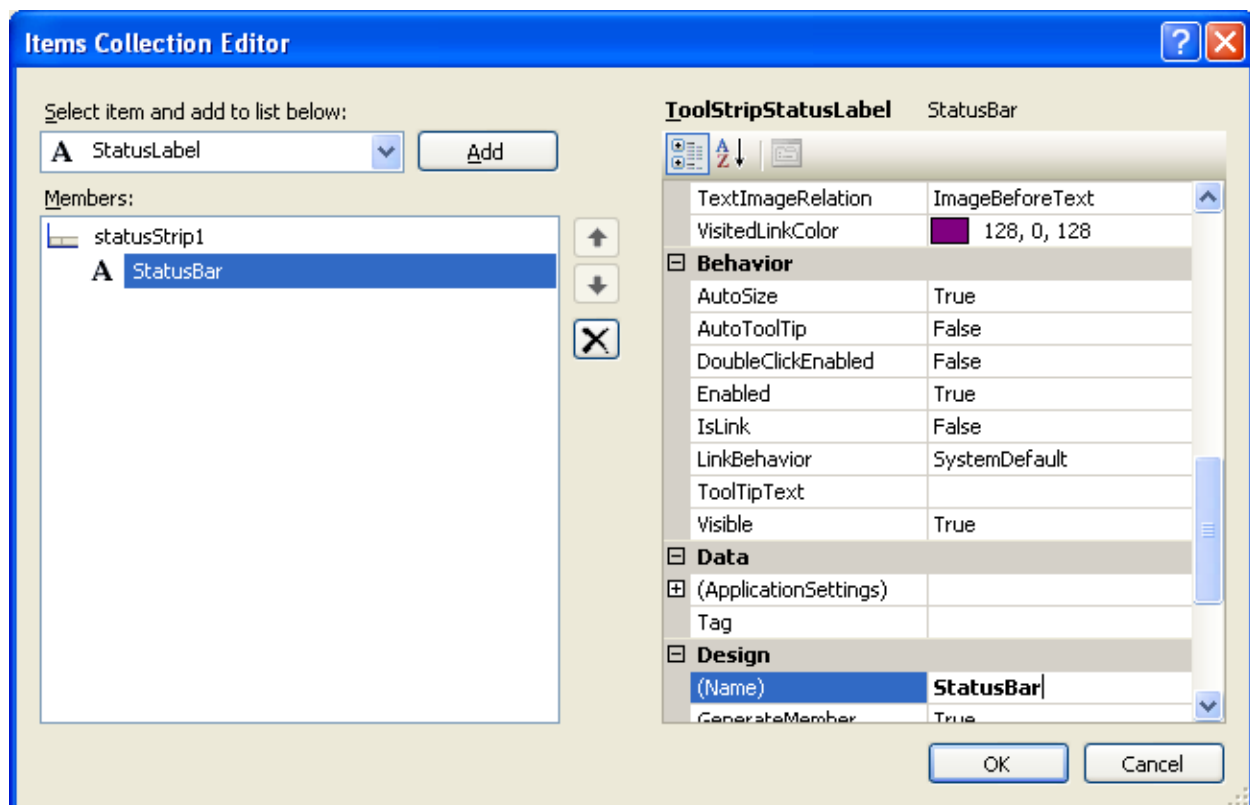
Then, add a StatusLabel for the StatusStrip.



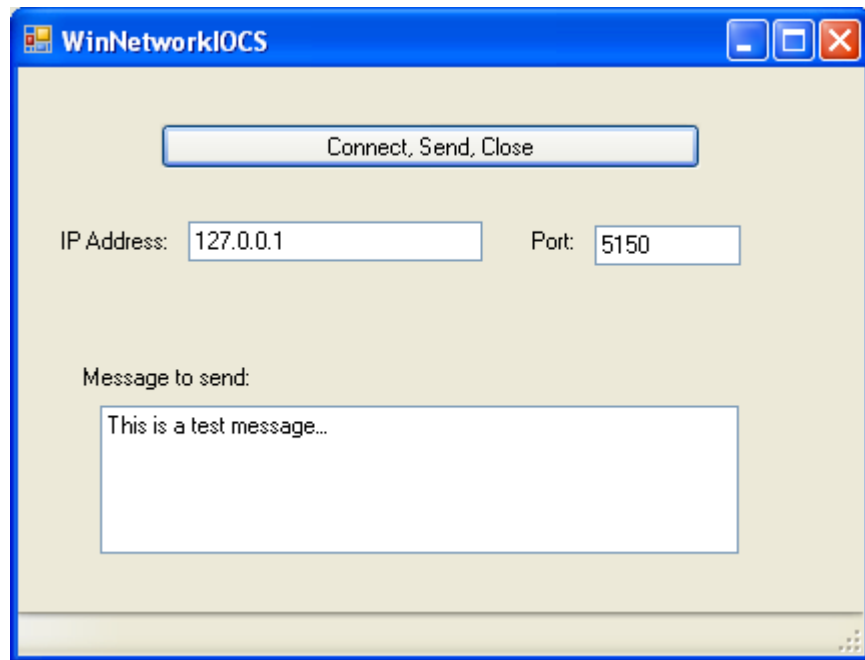
Do some customization for the StatusLabel. Select the statusStrip1 > right click mouse > select Edit Items context menu.



Select the StatusLabel1 in the left window (Members:). Change the Text to None and Name to StatusBar. Click OK.



The following is a completed Windows form design for this project. Now, we are ready to add codes.



Adding codes for the actions. First of all, double click the top button. This will bring the code page with a button click template. Add the following two lines code to the ConnectButton\_Click().

```
private void ConnectButton_Click(object sender, EventArgs e)
{
    DisableFields();
    DoNetworkingConnection();
}
```



```

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

        private void ConnectButton_Click(object sender, EventArgs e)
        {
            DisableFields();
            DoNetworkingConnection();
        }
    }
}

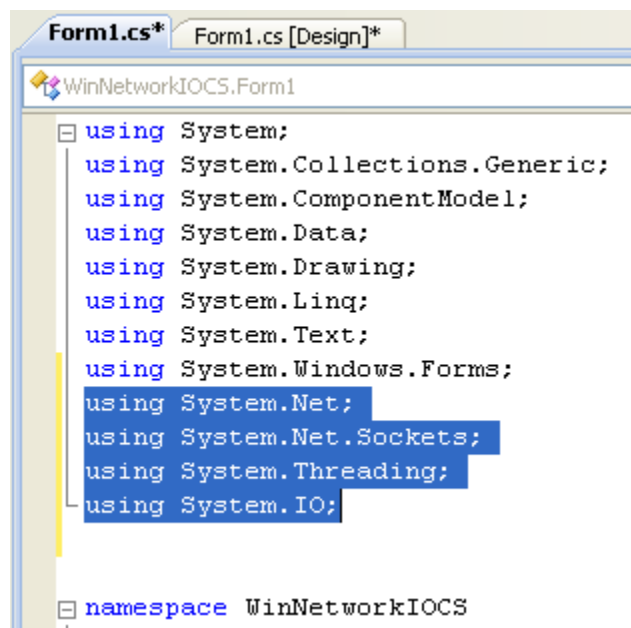
```

Don't forget to add the following using directives.

```

using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;

```



```

Form1.cs*  Form1.cs [Design]*
WinNetworkIOCS.Form1
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;


namespace WinNetworkIOCS
{

```

Continue adding more codes just after the ConnectButton\_Click() closing bracket.

```
private void DisableFields()
{
    PortBox.Enabled = false;
    IPAddressBox.Enabled = false;
    SendMessageBox.Enabled = false;
    ConnectButton.Enabled = false;
}

private void EnableFields()
{
    PortBox.Enabled = true;
    IPAddressBox.Enabled = true;
    SendMessageBox.Enabled = true;
    ConnectButton.Enabled = true;
}
```



```
private void ConnectButton_Click(object sender,
{
    DisableFields();
    DoNetworkingConnection();
}

private void DisableFields()
{
    PortBox.Enabled = false;
    IPAddressBox.Enabled = false;
    SendMessageBox.Enabled = false;
    ConnectButton.Enabled = false;
}

private void EnableFields()
{
    PortBox.Enabled = true;
    IPAddressBox.Enabled = true;
    SendMessageBox.Enabled = true;
    ConnectButton.Enabled = true;
}
}
```

Add the following code to write messages to the status bar of the Windows form.

```
private void WriteToStatusBar(string Message)
{
    EnableFields();
    StatusBar.Text = Message;
}
```

```

        SendMessageBox.Enabled = true;
        ConnectButton.Enabled = true;
    }

    private void WriteToStatusBar (string Message)
    {
        EnableFields();
        StatusBar.Text = Message;
    }

```

Then, add code for the networking.

```

private void DoNetworkingConnection()
{
    Thread MyThread = null;

    try
    {
        ThreadStart ThreadMethod
= new ThreadStart(ConnectTo);

        MyThread = new Thread(ThreadMethod);
    }
    catch (Exception e)
    {
        WriteToStatusBar("Failed to create thread
with error: " + e.Message);
        return;
    }

    try
    {
        MyThread.Start();
    }
    catch (Exception e)
    {
        WriteToStatusBar("The thread failed to start
with error: " + e.Message);
    }
}

```

```

        StatusBar.Text = Message;
    }

    private void DoNetworkingConnection()
    {
        Thread MyThread = null;

        try
        {
            ThreadStart ThreadMethod = new ThreadStart(ConnectTo);

            MyThread = new Thread(ThreadMethod);
        }
        catch (Exception e)
        {
            WriteToStatusBar("Failed to create thread with error: " + e.Message);
            return;
        }

        try
        {
            MyThread.Start();
        }
        catch (Exception e)
        {
            WriteToStatusBar("The thread failed to start with error: " + e.Message);
        }
    }
}

```

Finally, add the following code for the real connection.

```

private void ConnectTo()
{
    string ServerName = this.IPAddressBox.Text;
    int Port = System.Convert.ToInt32(this.PortBox.Text);

    WriteToStatusBar("IP Address: " + ServerName + "Port: " + Port);
    Socket ClientSocket = null;

    try
    {
        // Let's connect to a listening server
        try
        {

```

```

        ClientSocket
= new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);
        WriteToStatusBar("Socket is OK...");
    }
    catch (Exception e)
    {
        throw new Exception("Failed to create client Socket: " + e.Message);
    }

    IPEndPoint ServerEndPoint
= new IPEndPoint(IPAddress.Parse(ServerName), Convert.ToInt16(Port));

    try
    {
        ClientSocket.Connect(ServerEndPoint);
        WriteToStatusBar("Connect() is OK...");
    }
    catch (Exception e)
    {
        throw new Exception("Failed to connect client Socket: " + e.Message);
    }
}
catch (Exception e)
{
    WriteToStatusBar(e.Message);
    ClientSocket.Close();
    return;
}

// Let's create a network stream to communicate over the connected Socket.
NetworkStream ClientNetworkStream = null;

try
{
    try
    {
        // Setup a network stream on the client Socket
        ClientNetworkStream = new NetworkStream(ClientSocket, true);
        WriteToStatusBar("Instantiating NetworkStream...");
    }
    catch (Exception e)
    {
        // We have to close the client socket here because the network
        // stream did not take ownership of the socket.
        ClientSocket.Close();
    }
}

```

```

        throw new Exception("Failed to create a NetworkStream with error: " +
e.Message);
    }

    StreamWriter ClientNetworkStreamWriter = null;

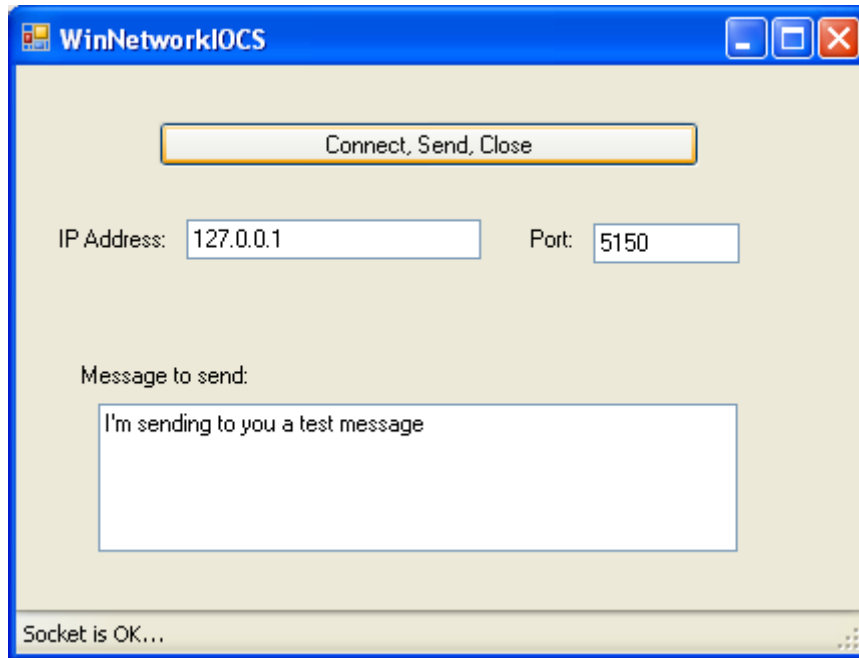
    try
    {
        // Setup a Stream Writer
        ClientNetworkStreamWriter = new StreamWriter(ClientNetworkStream);
        WriteToStatusBar("Setting up StreamWriter...");
    }
    catch (Exception e)
    {
        ClientNetworkStream.Close();
        throw new Exception("Failed to create a StreamWriter with error: " +
e.Message);
    }

    try
    {
        ClientNetworkStreamWriter.Write(this.SendMessageBox.Text.ToString());
        ClientNetworkStreamWriter.Flush();
        WriteToStatusBar("We wrote
" + this.SendMessageBox.Text.Length.ToString() + " character(s) to the server.");
    }
    catch (Exception e)
    {
        throw new Exception("Failed to write to client NetworkStream with error:
" + e.Message);
    }
}
catch (Exception e)
{
    WriteToStatusBar(e.Message);
}
finally
{
    // Close the network stream once everything is done
    ClientNetworkStream.Close();
}
}

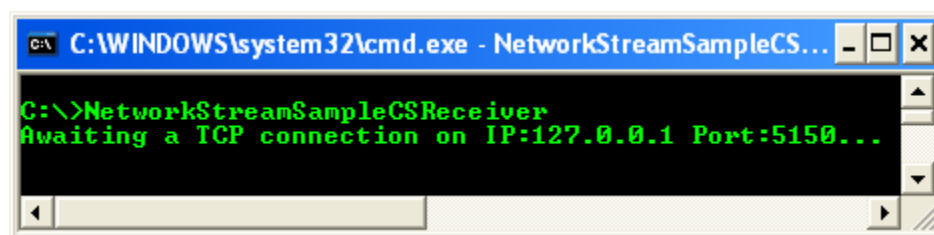
```

Build and run the project. The following is a sample output and it is a client/sender program. In order to see the real activities, we will test this client together with the

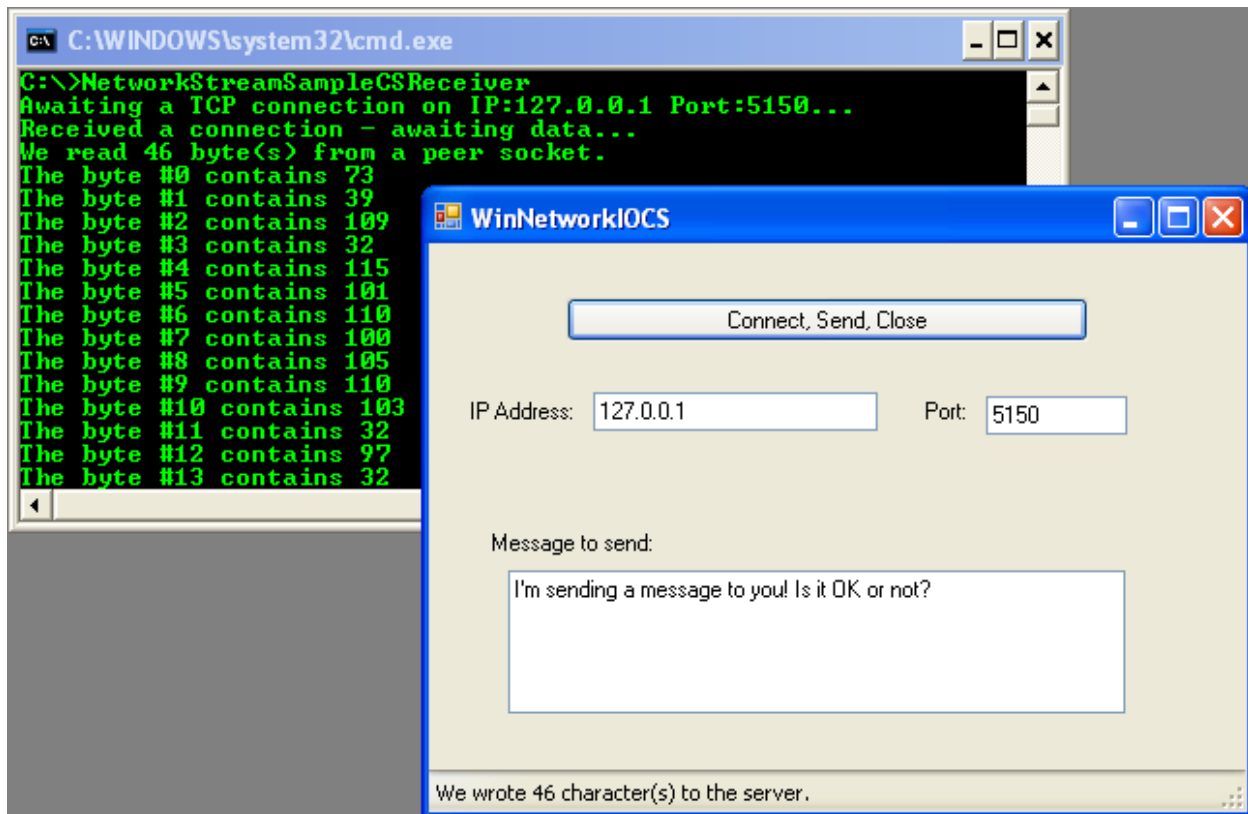
previously created stream server/receiver program (NetworkStreamSampleCSReceiver.exe). In this case we test both programs on the same local host and you may want to test it on different hosts which mean run the receiver on host A and run the sender on host B and change the IP address of the sender accordingly.



Running both, the receiver and the sender programs. Firstly run the previous receiver program (NetworkStreamSampleCSReceiver.exe). The receiver is waiting a connection from sender.



Then, run the sender/client program (WinForm application). The following is a sample output for both the sender and the receiver.



Take note that the numbers output in the receiver program represent the [ASCII code for the characters sent](#), for example the first alphabet 'I' is equivalent to '73' ASCII. A complete source codes for this part is given below.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;

namespace WinNetworkIOCS
{
    public partial class Form1 : Form
```



```

{
    public Form1()
    {
        InitializeComponent();
    }

    private void ConnectButton_Click(object sender, EventArgs e)
    {
        DisableFields();
        DoNetworkingConnection();
    }

    private void DisableFields()
    {
        PortBox.Enabled = false;
        IPAddressBox.Enabled = false;
        SendMessageBox.Enabled = false;
        ConnectButton.Enabled = false;
    }

    private void EnableFields()
    {
        PortBox.Enabled = true;
        IPAddressBox.Enabled = true;
        SendMessageBox.Enabled = true;
        ConnectButton.Enabled = true;
    }

    private void WriteToStatusBar(string Message)
    {
        EnableFields();
        StatusBar.Text = Message;
    }

    private void DoNetworkingConnection()
    {
        Thread MyThread = null;

        try
        {
            ThreadStart ThreadMethod = new ThreadStart(ConnectTo);

            MyThread = new Thread(ThreadMethod);
        }
    }

```

```

        catch (Exception e)
        {
            WriteToStatusBar("Failed to create thread with error: " + e.Message);
            return;
        }

        try
        {
            MyThread.Start();
        }
        catch (Exception e)
        {
            WriteToStatusBar("The thread failed to start with error: " + e.Message);
        }
    }

    private void ConnectTo()
    {
        string ServerName = this.IPAddressBox.Text;
        int Port = System.Convert.ToInt32(this.PortBox.Text);

        WriteToStatusBar("IP Address: " + ServerName + "Port: " + Port);
        Socket ClientSocket = null;

        try
        {
            // Let's connect to a listening server
            try
            {
                ClientSocket
= new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);
                WriteToStatusBar("Socket is OK...");
            }
            catch (Exception e)
            {
                throw new Exception("Failed to create client Socket: " + e.Message);
            }

            IPEndPoint ServerEndPoint
= new IPEndPoint(IPAddress.Parse(ServerName), Convert.ToInt16(Port));

            try
            {
                ClientSocket.Connect(ServerEndPoint);
                WriteToStatusBar("Connect() is OK...");
            }

```

```

    }
    catch (Exception e)
    {
        throw new Exception("Failed to connect client Socket: " + e.Message);
    }
}
catch (Exception e)
{
    WriteToStatusBar(e.Message);
    ClientSocket.Close();
    return;
}

// Let's create a network stream to communicate over the connected Socket.
NetworkStream ClientNetworkStream = null;

try
{
    try
    {
        // Setup a network stream on the client Socket
        ClientNetworkStream = new NetworkStream(ClientSocket, true);
        WriteToStatusBar("Instantiating NetworkStream...");
    }
    catch (Exception e)
    {
        // We have to close the client socket here because the network
        // stream did not take ownership of the socket.
        ClientSocket.Close();
        throw new Exception("Failed to create a NetworkStream with error: " +
e.Message);
    }

    StreamWriter ClientNetworkStreamWriter = null;

    try
    {
        // Setup a Stream Writer
        ClientNetworkStreamWriter = new StreamWriter(ClientNetworkStream);
        WriteToStatusBar("Setting up StreamWriter...");
    }
    catch (Exception e)
    {
        ClientNetworkStream.Close();
    }
}

```

```

        throw new Exception("Failed to create a StreamWriter with error: " +
e.Message);
    }

    try
    {
        ClientNetworkStreamWriter.Write(this.SendMessageBox.Text.ToString());
        ClientNetworkStreamWriter.Flush();
        WriteToStatusBar("We wrote
" + this.SendMessageBox.Text.Length.ToString() + " character(s) to the server.");
    }
    catch (Exception e)
    {
        throw new Exception("Failed to write to client NetworkStream with error:
" + e.Message);
    }
}
catch (Exception e)
{
    WriteToStatusBar(e.Message);
}
finally
{
    // Close the network stream once everything is done
    ClientNetworkStream.Close();
}
}
}
}

```