Visual C# Serial Communication

Overview

This simple program, initializes the COM port and sends a Move Relative, Time Based (MRT) command to the attached QuickSilver Device. The device's response is received and displayed.

It is assumed the reader is familiar with Windows, Visual C#, QuickControl®, programming QuickSilver products, and QuickSilver's serial communication. For more information see:

- QCI-TD053 Serial Communications
- SilverLode User Manual

This document is meant to explain the setup and general design of the example. The details concerning communicating with a QuickSilver device are left up to the documents mentioned above and the source code comments.

This example was created using Visual C# 2008 Express which is offered free from Microsoft.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Setup

The following assumptions are made about the setup:

- The device has already been initialized (using QuickControl) and connected to COM 1.
- 8 Bit Protocol
- 57600 Baud Rate
- Unit ID of 16

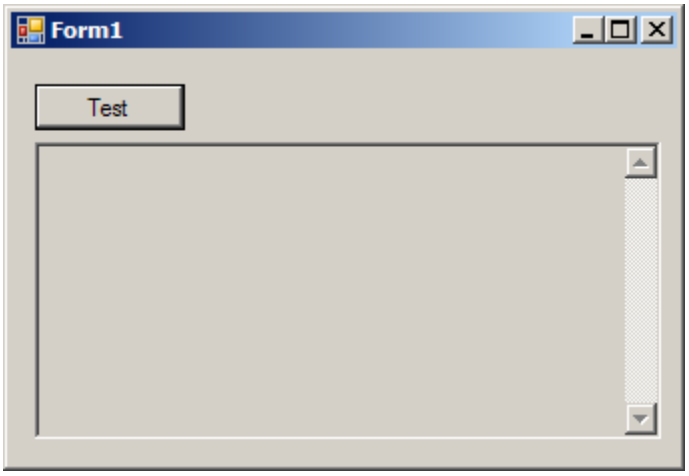
Supplied Files

File/Folder	Description
CommTest	Visual C# Project Folder

Application Overview

Dialog Box

Pressing the 'Test' button will send an MRT command. Responses are displayed in the text box.



SerialPort class

This example uses the SerialPort class supplied with Visual C#.

You can add this control to your project by selecting Toolbox->Components->SerialPort and dragging it onto your project.

Form1.cs
This is the main class for the dialog box. It contains the code that enables us to communicate with the device.

The COM1 is setup in Form1_Load when the form first loads.

When the 'Test' button is pressed, CmdTest_Click transmits the MRT command and appends the string to textBox1.

The event handler serialPort1_DataReceived is called when data is received. The handler appends the received data to textBox1.

```
namespace CommTest
{
    public partial class Form1 : Form
    {
        // Used to pass rx data.
        //  For production code, QCI
        //  suggests using a cicular queue
        string strRx;

        // Constructor
        public Form1()
        {
            InitializeComponent();
        }

        // Configure COM when form opens
        private void Form1_Load(
            object sender, EventArgs e)
        {
            ConfigureCOM();
        }
        private void ConfigureCOM()
        {
            serialPort1.PortName = "COM1";
            serialPort1.BaudRate = 57600;
            serialPort1.Open();
        }
        // Close COM port when form closes
        private void Form1_FormClosing(
            object sender, FormClosingEventArgs e)
        {
            if (serialPort1.IsOpen) serialPort1.Close();
        }
        // Send MRT when Test pressed
        private void cmdTest_Click(
            object sender, EventArgs e)
        {
            // send move relative, time based(mrt)
            //  mrt 40000 count,
            //  100ms ramp time, 1000ms total time
            string strCmd = "@16 177 40000 833 8333 0 0 \r";
            serialPort1.Write(strCmd);

            // Write to textBox1
            textBox1.AppendText("\n");
            textBox1.AppendText(strCmd);
            textBox1.AppendText("\n");
        }
        // Display controller response in text box.
        private void DisplayText(object sender, EventArgs e)
        {
            textBox1.AppendText(strRx);
        }
        private void serialPort1_DataReceived(
            object sender,
            System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            strRx = serialPort1.ReadExisting();
            this.Invoke(new EventHandler(DisplayText));
        }
    }
}
```