

Kurtis Hessler
2/24/07
ECE 480

How to Build a Serial Hub/Switch

Executive Summary:

The following application note is an extension of a basic microcontroller and GUI development lab used in a senior electrical engineering course (referenced below). The note goes over the basic wiring and programming of the PIC and GUI for the application and provides helpful tips and recommendations along the way.

Introduction:

A serial Hub is a device that allows for the interfacing of multiple devices when serial ports are limited. The idea is to create an automatic switching method to replace the tedious task of having to unplug one serial connector and plug in another serial device. There are several ways of accomplishing this task. One method, that is somewhat as inefficient as the plug/unplug method, is to use a switch board. This type of device would require the user to manually switch on the desired device as well as have all other devices switched off. The preferable method would be to use a solid state device such as a microcontroller for switching. In this application note we will explain how to use the PIC18F4520 microcontroller as well as several other components to accomplish this task.

Design:

The circuit idea is a simple one. The hub simply turns on the power to the correct device. The data lines for each device are all tied together and to the output serial port. This method of integration is ideal because it eliminates having to run the information lines through the serial hub microprocessor, which means smaller code, less hassle.

Parts Needed:

Amount	Name	Description
1	PIC18F4520 Microcontroller	Serial switch Microcontroller
1	MXO45IIS Clock	Clock for the Microcontroller
1	MAX232 Serial Buffer	Used for serial communication
4	10 Microfarad Capacitors	General circuit capacitors
1	10k ¼ Watt Resistor	Pull up resistor for microcontroller
1-X*	Serial Male/Female DB9 Conn.	Serial connections for devices
1	Winford PBC6P6C	For programming microcontroller
1	Bread Board/Perf Board	For connecting devices
X*	General Equipment	Wire/Solder/Power Supply/etc.

* "X" denotes an arbitrary amount depending on the quantity desired or needed.

Step 1: Wiring the Device:

There are a couple of things to pay attention to when wiring up your circuit.

- I recommend first wiring on a bread board as it is much easier to diagnose and fix mistakes.
- NEATNESS COUNTS! The cleaner your circuit looks the easier it will be to navigate to correct problems as well as to add additional components that may be desired in the future.
- Make sure the power supply is OFF when doing any additional wiring or changes to the circuit.
- Double check your wiring before plugging in the power supply.
- If the power supply displays an overload in current when initially turned on then there is a serious problem and you should immediately turn off the power supply.
- Another important note is to pay attention to the polarities on the capacitors, in some cases an incorrect wiring of a capacitor can result in the device exploding and causing injury to the eye or other body parts.
- If no problems are apparent and everything looks in order you are ready to move on to the programming of the microcontroller.

Step 2: Programming the PIC Microcontroller:

This part can be a little tricky and frustrating. A simple error in syntax can result and lots of lost time.

- Begin by plugging the RJ11 cord from the MPLAB circuit debugger into the RJ11 female port on your serial hub board.
- The next step involves opening and setting up the program. Go to START>PROGRAMS>MICROCHIP>MPLAB>MPLABIDE.
Under the projects tab select the PROJECT WIZARD. Select next and in the following window make sure the device 18F4520 is selected. Hit next again and make sure the C18 tool-suite is selected in the following window. Hit next. In the next screen create a project name and then select the directory in which you wish to store your file. Hit next. We now must add an existing software file to the project. To do so select
C:/>PROGRAMFILES>MICROCHIP>MCC18>LKR>18F4520.LKR.
Once selected click add and then the next button. Click finish.
- Now we can start a new file by clicking FILE>NEW. A blank page will pop up.
- Copy the following code and paste it into your new file (the // command is used for comments, the text immediately following these commands can be very useful for understanding the operation of your program.):

```
#include <p18cxxx.h>
#include <usart.h>
#include <ADC.h>
#pragma config LVP=OFF
#pragma config WDT=OFF
```

```
void rx_handler (void);           //Declare the ISR function
long int count;
```

```

int dloop, altflash, adc_result;
void main()
{
    OpenUSART (USART_TX_INT_OFF & USART_RX_INT_ON &
               USART_ASYNC_MODE & USART_EIGHT_BIT &
               USART_CONT_RX & USART_BRGH_LOW, 63);
    RCONbits.IPEN = 1;          /* Enable interrupt priority */
    IPR1bits.RCIP = 1;         /* Make receive interrupt high priority */
    INTCONbits.GIEH = 1;       /* Enable all high priority interrupts */
    TRISD = 0x00;
    PORTDbits.RD0 = 0;         //initialize output to zero (power is off to device1)
    PORTDbits.RD1 = 0;         //initialize output to zero (power is off to device2)
    PORTDbits.RD2 = 0;         //initialize output to zero (power is off to device3)
    PORTDbits.RD3 = 0;         //initialize output to zero (power is off to device4)
    PORTDbits.RD4 = 0;         //initialize output to zero (power is off to device5)
    PORTDbits.RD5 = 0;         //initialize output to zero (power is off to device6)
    //The PORTDbits.RDX specifies the pin on the D register. 1 for on and 0 for off.
    while(1);
}
#pragma code rx_interrupt = 0x8
void rx_int (void)
{
    _asm goto rx_handler _endasm
}
#pragma code

//Lets the compiler know that this function is the ISR
#pragma interrupt rx_handler
void rx_handler (void)
{
    unsigned char c;
    c = getcUSART();          //get a single character off the USART line c is set with the
                             //character from the serial line
    while(BusyUSART());

    if (c == 'a')            //this command check to see if the letter, or "protocol" is the letter
                             // 'a'. A protocol is like an identifier for the device you wish to
                             //turn on. If it is 'a' the code within the { } will be executed.
    {
        PORTDbits.RD0=1;    //turns the device 1 on
        PORTDbits.RD1=0;    //turns device 2 off
        PORTDbits.RD2=0;    //etc.
        PORTDbits.RD3=0;
        PORTDbits.RD4=0;
        PORTDbits.RD5=0;
    }
}

```

```

if (c == 'b') //second device
{
    PORTDbits.RD1=1;
    PORTDbits.RD0=0;
    PORTDbits.RD2=0;
    PORTDbits.RD3=0;
    PORTDbits.RD4=0;
    PORTDbits.RD5=0;
}
if (c == 'c') //third device
{
    PORTDbits.RD2=1;
    PORTDbits.RD0=0;
    PORTDbits.RD1=0;
    PORTDbits.RD3=0;
    PORTDbits.RD4=0;
    PORTDbits.RD5=0;
}
if (c == 'd') //fourth device
{
    PORTDbits.RD3=1;
    PORTDbits.RD1=0;
    PORTDbits.RD2=0;
    PORTDbits.RD0=0;
    PORTDbits.RD4=0;
    PORTDbits.RD5=0;
}
if (c == 'e') //fifth device
{
    PORTDbits.RD4=1;
    PORTDbits.RD1=0;
    PORTDbits.RD2=0;
    PORTDbits.RD3=0;
    PORTDbits.RD0=0;
    PORTDbits.RD5=0;
}
if (c == 'f') //sixth device, you can have as many devices as
//there are unique protocols.
{
    PORTDbits.RD5=1;
    PORTDbits.RD1=0;
    PORTDbits.RD2=0;
    PORTDbits.RD3=0;
    PORTDbits.RD4=0;
    PORTDbits.RD0=0;
}

```

```

    PIR1bits.RCIF = 0;           //reset the ISR flag.
}

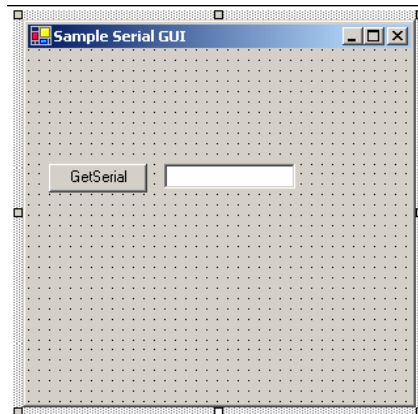
```

- Set your power supply 5V and turn it on. In the MPLAB program select PROGRAMMER>SELECTPROGRAMMER>MPLAB ICD 2. Then select DEBUGGER>SELECTTOOL>MPLAB ICD 2.
- Now you are ready to compile your code and program the chip. Select PROJECT>BUILDCALL. You should not get any red errors in the build menu. If you do, return to your code and double check it is accurate. When the build is successful select PROGRAMMER>PROGRAM. This function will load the program into the debugger. For loading the program on the chip, all you have to do is select “none” under the debugger menu and then program the PIC using the MPLab programmer. When everything is programmed you are ready to move on to the GUI.

Step 3: Writing the GUI (GRAPHICAL USER INTERFACE):

The GUI for this application is fairly simple. The GUI will send a character or “protocol” through the serial communication line to the PIC microcontroller and thus signaling the PIC to turn on the proper corresponding device.

- The first step would be to purchase the Microsoft Visual Studio .NET program.
- As a base, a file labeled LAB4VB has included to get us started. Extract the files and open the folder. Double click the file labeled “sampleserial.vbproj”. This function will open up Microsoft Visual Basic.
- When the program window opens there will be a directory on the left hand side. Double click the file titled “Form1.vb”. This action will open a box that looks like the following:



- You can think of this as the construction window for a panel of buttons that will select the device you want to use. First we must create buttons for each device.
- Select the “GetSerial” button and right click select copy. Then paste as many buttons as you have devices. Delete the original text box.
- We need to give each button unique names. Select each button one at a time and in the bottom right property box change the properties “label” and “name” fields to a unique name for the device you want to turn on. Do this for all of the buttons.

- Now we need to create code for the buttons. Double click each of the buttons one at a time. This will bring up the code for the buttons. The code portion will look something like the following:

```
Private Sub Data2_Click(ByVal sender As System.Object,
    setRs232.Write("c")
    setRs232.Read(1)
    serial_in = setRs232.InputStreamString
    TextBox1.Text = serial_in
End Sub
```

- Data2_Click will be replaced by the name of the button you have double clicked on. We only need the setRs232.Write('') code. You can delete the 232.Read, serial_in, and textbox1 form all of the button code segments. The following is what our code will look like when finished for each button:

```
Private Sub Data2_Click(ByVal sender As System.Object,
    setRs232.Write("■")
End Sub
```

- The red square will be replaced with the proper character “protocol” for the corresponding device stored in the PIC code. **Remember:** each device has its own unique protocol. The GUI is simply sending the protocol to the PIC to select the proper device. It is important that the correct protocol the GUI button is sending matches up with the correct protocol on the PIC so the right device is turned on.
- After the coding is complete we are ready to run our final product.

Step 4: Running the final product.

- Turn your power supply on and make sure there are no current overloads.
- Next you need to run the program in the MPLAB program window as previously mentioned.
- Now we need to run our GUI. Return to the visual studio program and select DEBUG > RUN. A window should pop up similar to the window in which you made your buttons in. Now we are ready to see the final product run.
- For testing purposes I placed LEDs in series with a 330 ohm resistor just to see that the on signal is working. Select each button on your GUI and watch the LED’s turn on. Now remember we coded the program so when one LED is on and you select another button it will turn the Second LED on and shut the First one off.

Step 5: Troubleshooting.

- If your serial Hub is not working properly go back and check that your program syntax is correct. It is very easy to make a mistake in the programming stage. If this all checks out then make sure your wiring is correct.
- Thanks for reading my application note and I hope you enjoy using your new serial Hub!

Conclusions:

The serial is a great beginner project for learning how to write GUIs and program PIC microcontrollers. The PIC application for this device is great because of the vast amount of other things that can be done with the PIC. Another good thing about using a solid state switching device is that the size is greatly reduced. Cost and durability is also improved over a traditional switch box setup.

Recommendations:

I recommend using the microcontroller output to turn on the device. A separate power supply should be used to power the actual device being turned on.

References:

A special thanks to Kyle Thomson for the base visual studio project file and base MPLAB file. The application note was designed as a modification of his lab written for ECE480 at Michigan State University. For a reference to Lab4 here:

<http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/>

This lab used the PIC18f4520 microcontroller but many other PIC microcontrollers can perform this task. For the PIC18f4520 datasheet see:

<http://ww1.microchip.com/downloads/en/DeviceDoc/39631a.pdf>

This document was created with Win2PDF available at <http://www.win2pdf.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.
This page will not be added after purchasing Win2PDF.