ECE383: Microcomputers – Lab 6
Basic and Finite State Machine LED and Switch I/O Programming

Goals: The goals of this lab are to continue to instruct students in a PIC24-based hardware system using PIO ports for switch input and LED output using basic and finite state machine (FSM) processing.

1. Introduction

This lab introduces basic PIO port configuration and utilization for switch-based input and LED output. A C-based software finite state machine is used for an LED/switch I/O problem. The tasks in this lab include:

- Modifying and expanding the PIC24HJ128GP502 reference system schematic and printed circuit board to include additional pushbutton and LED components.
- Programming the Microstick II to implement a basic LED problem.
- Programming the Microstick II to implement a software finite state machine for an LED/switch I/O problem using additional pushbutton and LED components on the breadboard.

2. Prelab

For this lab assignment, Tasks 1, 2, and 3 should be completed as a pre-lab assignment prior to your assigned lab time. You are also strongly encouraged to have the majority of the software written for Task 4 and 5 complete before your assigned lab time.

TA check: As soon as you enter lab, provide the TA with a pre-lab report that includes the complete schematic, PCB layout, and C programs. Lab time will be devoted to debugging the program execution and correcting any errors within MPLAB noted by the lab instructor. Make sure your group member names and date are on the pre-lab report.

3. TASK 1: Expanding the PIC24 Reference System Schematic

Using PCB Artist, expand and modify the basic PIC24 system previously created as shown in Figure 1 below. If your previous PIC24 system schematic was designed and constructed neatly and correctly you may use your schematic from the previous lab as a starting point for this design. Otherwise you should construct a new system. Note that the pushbuttons use a different component from previous labs. The pushbutton component to use for this design, named PUSHBUTTON, is located in the ECE383 PCB Artist library located on the class website.

TA check: **Upon entering the lab, show the TA the schematic design. Use a screen capture tool to capture the schematic window and include it in your lab report.**

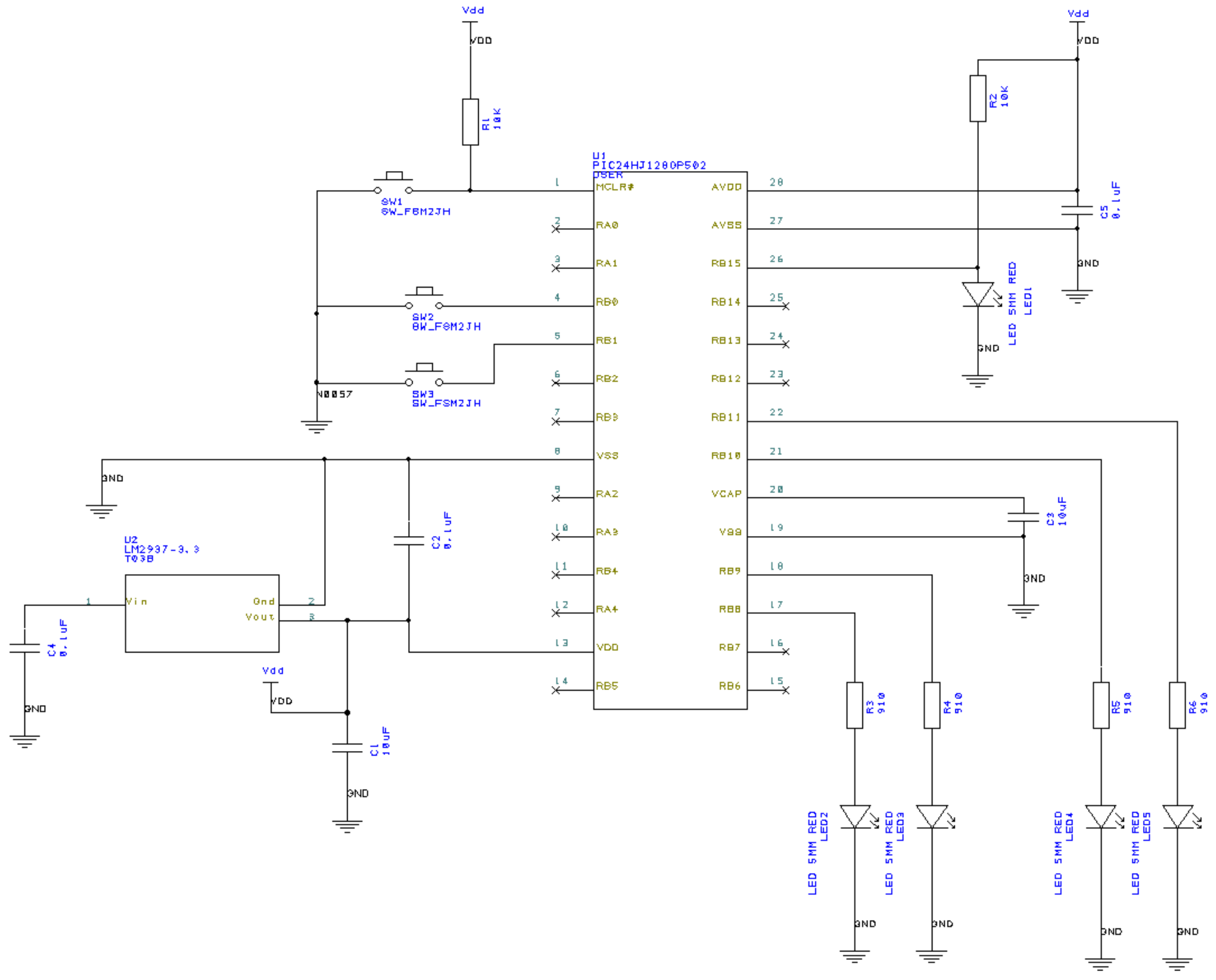

Figure 1. Expanded PIC24 System Schematic

4. TASK 2: Expanded PIC24 System Printed Circuit Board Layout

For this task we will create a printed circuit board layout from the PIC24 schematic created as a part of task 1. The board size should be 80mm x 80mm. The board part number should be "ECE383-LAB6" and the Revision Number should be "001". These parameters can be set through the PCB wizard.

Use the `Settings->Grids` function to set the **Working Grid** and the **Screen Grid** to 1mm x 1mm. Use the `Settings->Coordinates` function to set the Coordinate System Origin to the lower left corner of the board. Left click on the bottom edge of the board, press the "=" key and note the Y coordinate value. Left click on the left edge of the board, press the "=" key and note the X coordinate value. Enter the X and Y values for the Coordinate System Origin using the `Settings->Coordinates` function. This should set the origin to the lower left corner of the board as indicated by a ⊗ symbol. Another method for setting the system origin is to highlight an item (i.e. the bottom edge of the board), right click and select `Origins->Set System Origin At Item`.

We will create four mounting holes for this PCB using the `Add->Board->Circle` function. The diameter for each mounting hole should be 3mm and they should be placed in the four corners of the board with each hole being exactly 2mm from each edge of the board. Use the `Tools->Measure` function to assist you in this layout.

Each of the components should be placed in specific locations on the PCB. Highlight a component and press the "=" key. This will allow you to type in exact locations for the origins of each of the components. Table 1 gives the X and Y values for the location of each component in your design.

Component	X location	Y location
PIC24 (U1)	10	60
LM2937-3.3 (U2)	70	65
SW1	57	72
SW2	47	72
SW3	37	72
LED1	70	44
LED2	20	10
LED3	30	10
LED4	40	10
LED5	50	10
R1	24	30
R2	9	68
R3	30	30
R4	36	30
R5	42	30
R6	48	30
C1	64	67
C2	6	33
C3	22	33
C4	71	56
C5	22	54

Table 1. Expanded PIC24 System Component Locations.

Use your experience from previous labs to create a PCB similar to the diagram shown in Figure 2 below. After the PCB has been completed, use **`Tools->Design Rule Check`** to verify the design passes all the basic electrical design rules checks built into PCB Artist. Checking the **Spacing**, **Nets** and **Manufacturing** checkboxes should ensure all design rule checks are performed. The design rule check file created should indicate "No errors found" under the Results section if all design rule checks successfully pass. Generate a bill of materials CSV by running the **`Output->Reports->User Reports->bill of materials CSV`** report. This report should list all components in the design. Include the report output in an appendix of your lab report.

Generate a component positions report by running the **`Output->Reports->User Reports->component positions cvs`** report. This report should list the exact positions of all components in the design. Include the report output in an appendix of your lab report. Note that the positions reported may differ slightly from the location values from Table 1. They will only be identical if the center of the component is also the origin for that component.

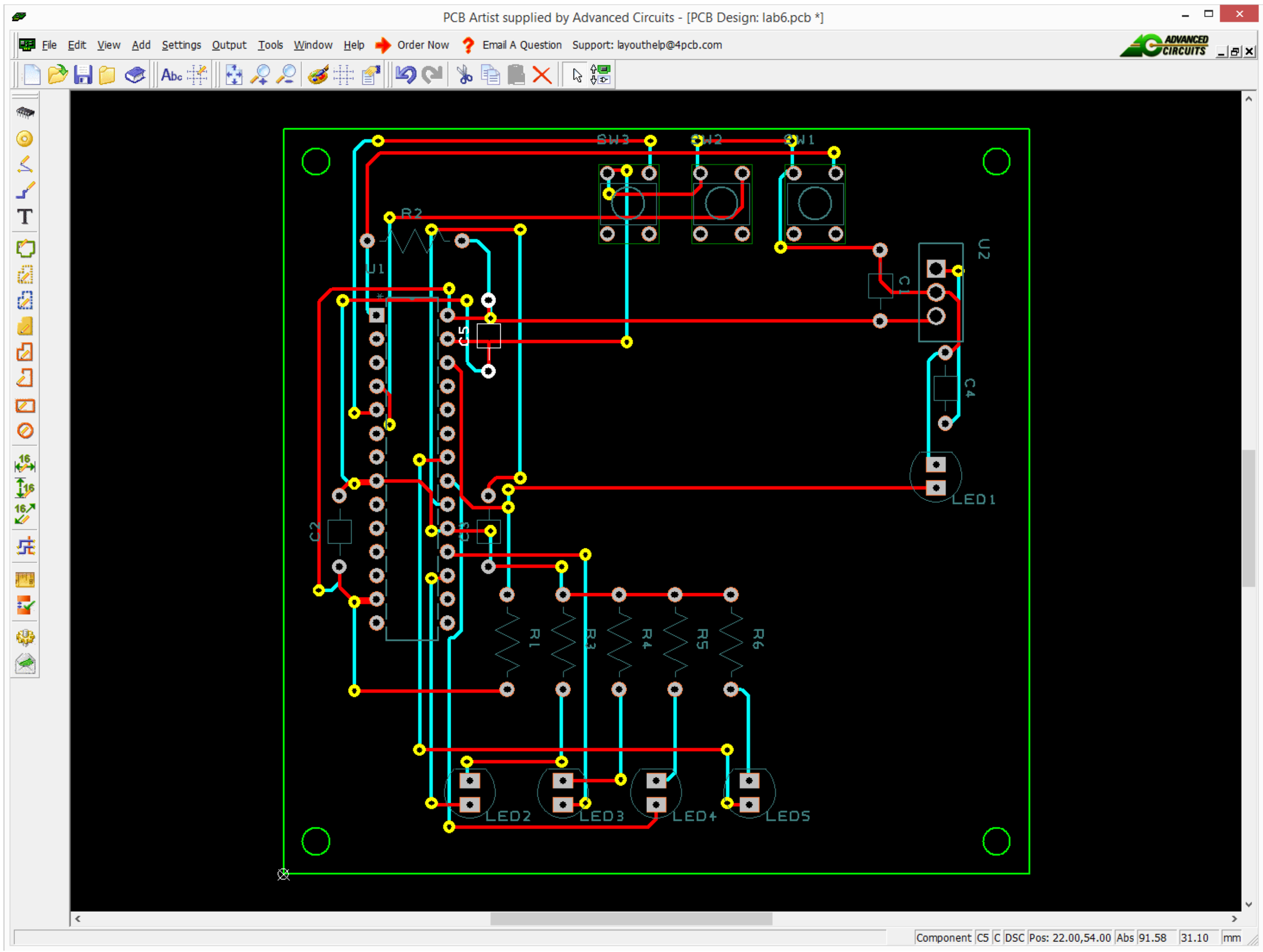
PCB Artist supplied by Advanced Circuits - [PCB Design: lab6.pcb *]
File Edit View Add Settings Output Tools Window Help Order Now Email A Question Support: layouthelp@4pcb.com
SW3
SW2
SW1
R2
U1
U2
C1
C5
C4
C2
C3
LED1
R1
R3
R4
R5
R6
LED2
LED3
LED4
LED5
Component C5 C DSC Pos: 22.00,54.00 Abs 91.58 31.10 mm

Figure 2. Expanded PIC24 System Printed Circuit Board

TA check: Upon entering the lab, show the TA the printed circuit board you completed. Include a printout of your PCB in your lab report.

5. TASK 3: A Basic LED Problem

This task will require you to write a C program to implement a basic LED problem. Perform the following steps:

- Start the MPLAB IDE. Use **`Project->Project Wizard`** for the creation of an MPLAB project.
  - Step One: Select the device *PIC24H128GP502* for the MicroStick II.
  - Step Two: Select the Active Toolsuite as *Microchip C30 Toolsuite*.
  - Step Three: *Create a New Project File* in a unique directory on the local hard disk (*C:\temp\task3a.mcp* as an example).
  - Step Four: Skip the addition of existing files to the project. After the project is open, use **`Project->Build Options`** to add the *C:\microchip\lib\include* directory to the *Include Search Path* directory.

    - Step Five: Configure the clock source for the processor. Select *Configure->Configuration Bits…* Uncheck the "Configuration Bits Set in code" checkbox. Change to FNOSC filed by clicking on the "Setting" area showing "Internal Fast RC (FRC) with divide by N". Change the setting to "Internal Fast RC (FRC) w/ PLL". Recheck the "Configuration Bits Set in code" checkbox.
- Enter the C program below and save it with the name *task3a.c* as a part of the project.

```
#include "pic24_all.h"
#if __PIC24HJ128GP502__
     #define LED1 _LATA0      // MicroStick II definitions
     #define CONFIG_LED1() CONFIG_RA0_AS_DIG_OUTPUT()
#endif

int main(void) {
     CONFIG_LED1();
     LED1=0;
     while (1) {              // Infinite while loop
          LED1 = !LED1;       // Toggle LED1
          DELAY_MS(100);      // Delay 100ms
     }
     return 0;
}
```

- Use **`Project->Add Files to Project`** to add the following files to your project:
    - C:\microchip\lib\common\pic24_clockfreq.c
    - C:\microchip\lib\common\pic24_serial.c
    - C:\microchip\lib\common\pic24_uart.c
    - C:\microchip\lib\common\pic24_util.c
- Compile the project by selecting **`Project->Build All`**.
- Enable the MPLAB IDE debugger by selecting **`Debugger->Select Tool->Starter Kit on Board`** for the MicroStick II.
- Download your code into a device on the board by selecting **`Debugger->Program`**. (Note: This is an important step to follow. Do not just click **Run** because it will run the previous program in the PIC24 memory. Every time, you edit the code. Build first, and then click **`Program`**).
- Run the application by selecting **`Debugger->Run`**. This should toggle an LED on the PIC24 board.

TA check: Show the TA the operation of the flashing LED program.

Using the procedure described above, create a new project (task3b.mcp), containing a C program (task3b.c). This program should alternate the rate at which the LED should toggle. The program should toggle the LED at a rate of 10 times per second for 5 seconds then change to 5 times per second for another 5 seconds. The process should then repeat. Use a variable that keeps track of elapsed time by counting the number of times the DELAY_MS() function is executed.

TA check: Show the TA the operation of the modified flashing LED program.

6. TASK 4: Software-Based Finite State Machine for LED/Switch I/O

For this task you use the MicroStick II and the breadboard and interface the following components. Two pushbuttons will be connected to RB12 and RB14 similar to the schematic created in Task 1. You will use the pushbuttons purchased as a part of your lab components. Two LEDs will be connected to RB1 and RB15. Resistors (910 ohm) should be connected to the cathode of each LED. The other terminal of the resistor should be connected to a Vss (GND) pin of the PIC24, which is either pin 19 or 8. The constructed circuit should resemble the circuit shown in Figure 3. Note: All power will be provided to the PIC24 system via the USB cable.

Implement the following LED/Switch I/O problem:

1. Turn on LED1 (RB15). On a press and release of pushbutton SW1, turn off the LED1 and go to step 2.
2. After a press and release of a pushbutton (SW1), blink LED1 two times and the freeze the LED1 on.
3. After a press and release of a pushbutton (SW1), if SW2 = 0, go to (1), else go to the next step.
4. When the pushbutton (SW1) is pressed and held down, blink LED2 five times per second. LED2 should be off if SW1 is not pressed. After the **second release**, go to (5).
5. Blink LED2 rapidly (twice as fast as step 4). On press and release of the pushbutton (SW2), go to (1).

Your first task should be to determine the states required and construct an **ASM chart** for implementing your assigned LED/switch I/O problem.

Create an MPLAB project *task4.mcp* and the corresponding C program *task4.c* to implement the finite state machine. Download your code to your PIC24 system and test the operation of your software design using the debugging capabilities of MPLAB and the Microstick II.

TA check: Show the TA your ASM chart and that your LED/switch code functions as expected.

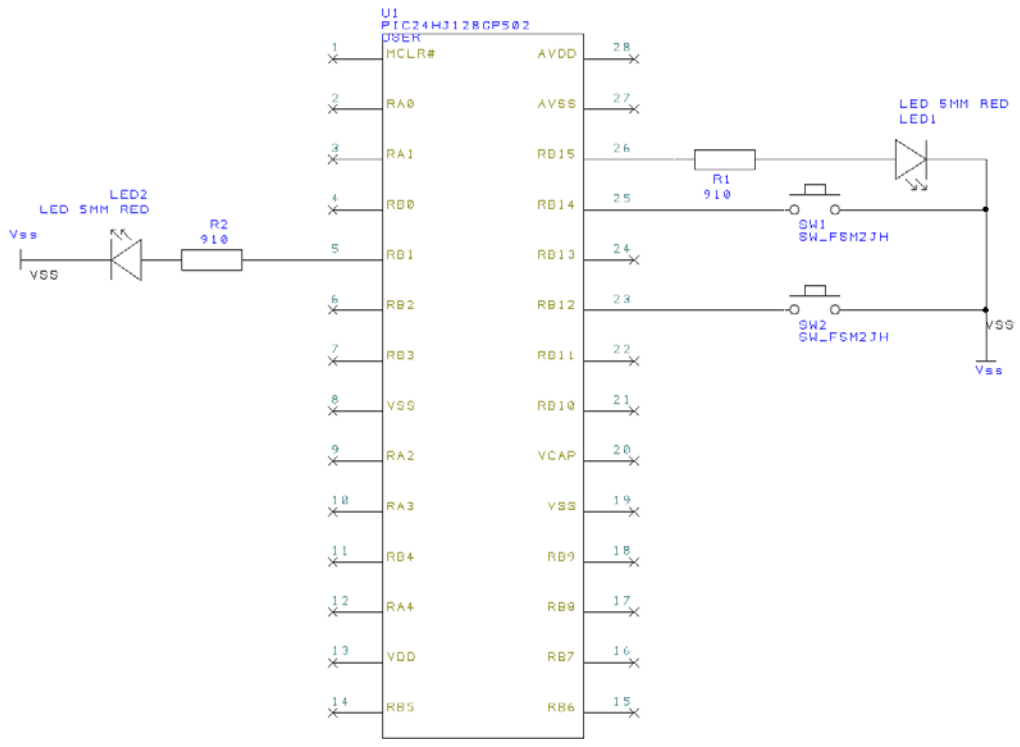

Figure 3. Task 4 schematic

7. TASK 5: Variable Rotating LED

For this task, you will implement a rotating LED program using the RGB LED interfaced with the PIC24 system. RGB (Red-Green-Blue) LEDs are actually three LEDs in one. Most RGB LEDs have four pins: one for each color and a common cathode pin. In this task, you will use a common cathode RGB LED which is shown in Figure 4. The data sheet link: http://www.kingbrightusa.com/images/catalog/SPEC/WP154A4SUREQBFZGC.pdf

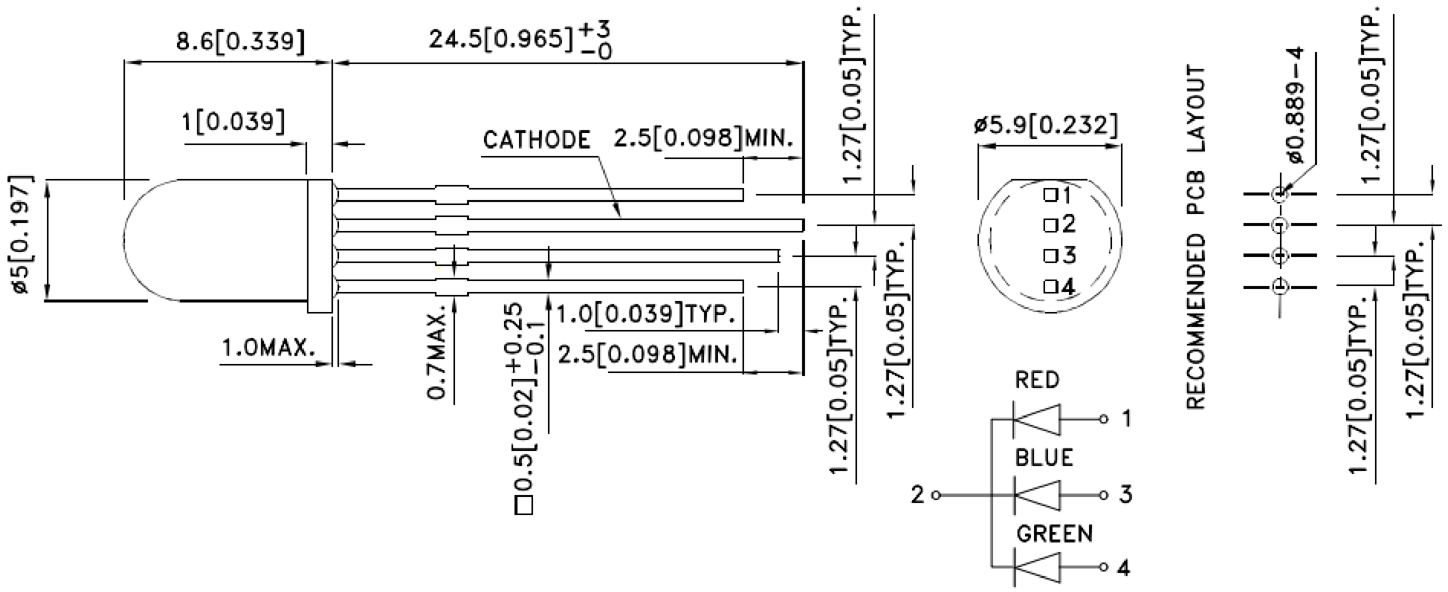

Figure 4. Common Cathode RBG LED

In this task, you are required to control the RGB LED using both binary codes and gray codes. Since there are only 3 leds in one RGB LED, we only consider binary codes and gray codes with 3-bits. Table 2 shows each 3-bit binary code and its corresponding gray code.

Binary	000	001	010	011	100	101	110	111
Gray	000	001	011	010	110	111	101	100

Table 2. Three-bit Binary Code and Gray Code

You are required to write a function in C which converts a binary code to its corresponding gray code. The input parameter of this function is a char type data which represents a binary code while the output is the gray code also in data type char. There are numerous methods for this conversion process. One possible approach is by bit shifting and an XOR operation. One example is given below.

Example:
Suppose we want to convert binary code '010' to gray code '011'.

Step 1:	00000010	
Step 2:	**0**0000001	(shift the code to right by 1bit, system would add **0** to last digit)
Step 3:	00000011	(XOR)

Once the code convert function is written write a program that implements the functionality given in Table 3 below.

SW1	SW2	Y location
Not pressed	Not pressed	Turn on all LEDs (R,G,B)
Not pressed	Pressed	Repeatedly display binary code from ‘000’ to ‘111’ on RGB LED with each combination on for 0.5 sec
Pressed	Not pressed	Repeatedly display gray code from ‘000’ to ‘100’ on RGB LED with each combination on for 0.5 sec
Pressed	Pressed	Blink the RGB led (all three colors) at the rate of 10 times per second

Table 3. Task Five Program Functionality

Name the project task5.mcp and the corresponding C program task5.c. Download your code to your PIC24 system and test the operation of your software design. Note that all the gray codes must be converted from the binary code instead of coming from Table 2 directly.

The constructed circuit should resemble the circuit in Figure 5.

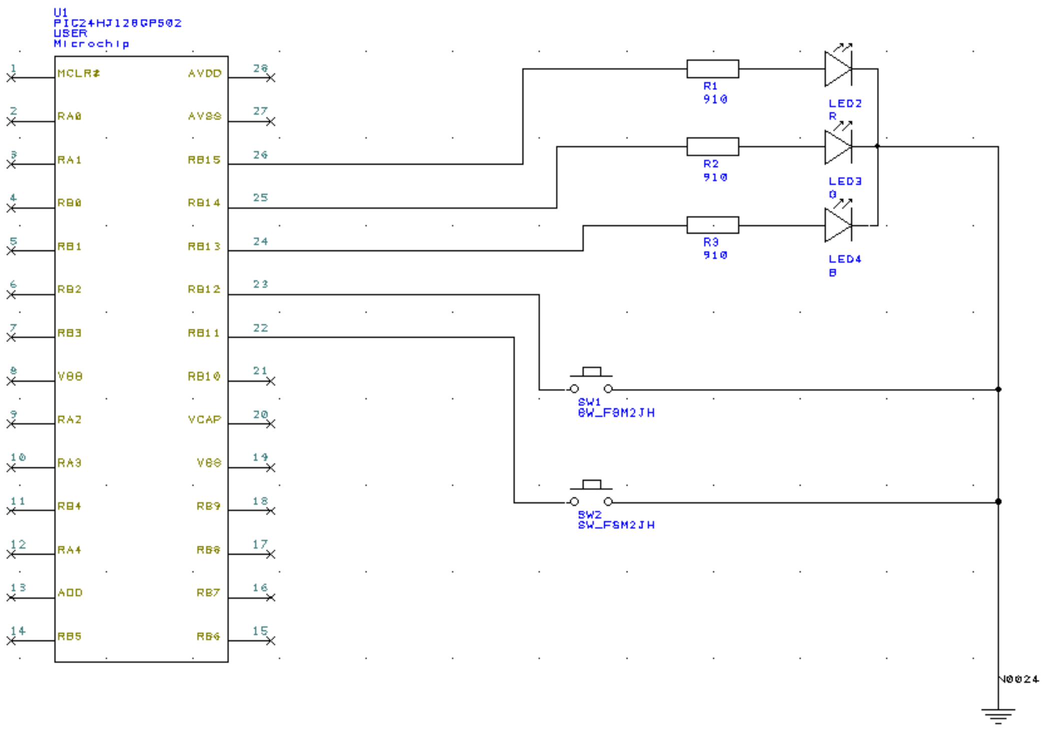

Figure 5. Task 5 schematic

Provide answers to the following questions in your lab report.

1. What are the advantages of binary codes and gray codes respectively?
2. Can we convert a gray code value back to binary code? If the answer is yes, please describe one possible method. Otherwise please explain the reason.

TA check: Show the TA that your program functions as expected.

8. Laboratory Report

No later than a week from the day the lab is performed, provide the TA a printed copy of the lab report following the ECE383 Lab report Template given on the class website. Each lab group will submit one joint lab report to the TA. Your report should have the reporting requirements needed for Tasks 1-5. **The TA will take off a significant number of points (15 points from total lab grade) if your C source does not have the required comments.**

9. Grading Policy

1. Completion of Task 1 with results included in lab report (10%)
2. Completion of Task 2 with results included in lab report (10%)
3. Completion of Task 3 with results included in lab report (20%)
4. Completion of Task 4 with results included in lab report (20%)
5. Completion of Task 5 with results included in lab report (20%)
7. Completeness, quality, and correctness of the lab report (20%)