**Assignment 9**

**David Moussalli**

**CS 2230, Fall 2016**

**Dr. Robert Trenary**

**A9 Design**

Assignment 9 will be the final assignment in the CS 2230 class. This program will allow the user to select one of the colors of the **RGB LED** (red, green, or blue) using the onboard switch and adjust the brightness of that color, using the potentiometer. The combination of these three choices will change the color of the **RGB**, the value of the current brightness will be displayed as two digits on the **7-segment display**, and thename of the current color being displayed on the **RGB** will be displayed on the user’s host machine.

In the beginning of the program, I create a global integer called **NUM**, which is used to determine which color of the **RGB** is being edited by the potentiometer. **NUM** is initialized to 1, so it can adjust the color red on the **RGB** first. Along with **NUM**, I will define9 variables (A-G), one for each segment on the **7-segment display** and one for **XIN** and **XOUT**. I will make 1 integer for **RED, GREEN,** and **BLUE** to hold the brightness of each color on the **RGB**, from 0-7. Next, I will create the main method and include the usual holding of the watchdog timer and calibration of the microcontroller to 1 megahertz. Then, I will use **P1DIR** and **P2DIR** to set certain pins as output and use **P1SEL** and **P2SEL** to assign certain pins to belong to timers **A0** and **A1**. I will then set up timers **A0** and **A1** to use pulse width modulation on pin 1.6, set **TA0CCR0** and **TA1CCR0** equal to 0x3FF, and set **TA0CCR1**, **TA1CCR1**, and **TA1CCR2** equal to zero. I will enable interrupts on **TA0CCTL0** and set the output mode for **TA0CCTL1, TA1CCTL1,** and **TA1CCTL2** to **OUTMOD\_3.** I will set up pin 1.3 (the onboard switch), enabling interrupts, setting the interrupt edge select to high, clearing the interrupt flags. I will use **P1REN** to enable the pullup resistor for **BIT3** as well as set pin 1.6 as the **P1OUT** and set its initial state to high.

Next I will make an infinite loop where the program will run continuously. In this loop, I will delay cycles by 10 milliseconds to allow the **ADC** ref to settle, start sampling and converting, then, when after an interrupt happens and is handled, I will test **NUM** to see if it is equal to 1, 2, or 3 and change the corresponding color of the **RGB** **LED**. Then, while still in the continuous loop, I will output the name of the color that is currently being shown on the **RGB** onto Minicom. The name of this color will be taken from the "colors.h" file, which is set up as a one dimensional array and the index for the array is a number which is based on the brightness held in **RED, GREEN,** and **BLUE**.

I will have 2 interrupts -- One for **timer A0** and one for when the onboard switch is pressed. The interrupt for the **timer A0** is called **TIMER0\_A0\_VECTOR** and it converts the number held in **ADC10MEM** to hex and displays it on the **7-segment display,** testing an integer **DIGIT** to see which digit to output the number to. When the program returns to the infinite while loop, it will test **NUM** and then use the value kept in **ADC10MEM** to adjust the corresponding color on the **RGB**.

The interrupt for the onboard switch is called **PORT1\_VECTOR** and when it is pressed, it tests **NUM**, then changes it, so that when it returns to the infinite loop, the value of **ADC10MEM** will be applied to the next **RGB**. After **NUM** is tested, I then de-bounce the button and clear the interrupt flag.

**Wiring**

As far as wiring goes, I am using the suggested wiring, except I have a common anode RGB LED, so instead of having the longest pin of the RGB go to GND, I have it going to VCC. Here is a representation of it that I made using Fritzing:

