Pulse Width Modulation (PWM)

with pwm generator

Project 4

Prepared for

CSE 410 Advanced Topics in Embedded System Design

By

Steven Collins (sac9)

October 20, 2024

Table of Contents

Section 1: Division of Work

I, Steven Collins, wrote the entirety of the pwm project. Some subroutines may have been used from CSE 379, on which my partner Frank Peretti and I worked together to implement.

Section 2: Program Overview

➔ Program Overview

- Run Program.
- User enters a numerical code on the keyboard, corresponding to a color, and the RGB LED lights up based on the color code entered.
- Color Codes:
    1 - Orange
    2 - Magenta
    3 - Light Pink
    4 - Purple
    5 - Capri Blue
    6 - Light Purple
    7 - Yellow-Green
    8 - Light Sea Green
    9 - Light Emerald
    0 - Rainbow
- The color code options are 0-9, where an invalid keystroke will be ignored (or pause the rainbow if code 0 is the current code).
- The user may press ‘q’ at any time to quit; otherwise the user can continually enter color codes to change the RGB LED.

➔ Program Summary

The pwm is set up to control the RGB LED. An interrupt timer interrupts the program every 0.1s, and based on the most recent number entered, changes the compare values of each of the RGB pins. This implementation of pwm is designed such that the clock counts starting at the LOAD value, and counts down to zero; also when the compare value for a pin is hit, that pin is driven high. So by a compare value closer to zero, that pin is on for less time, having a lower duty cycle. Increasing the compare value closer to the LOAD value (in this implementation the LOAD value is 401), will increase the duty cycle for that pin. If the compare value is exactly the load value, or 0, the pin will have an effective duty cycle of 0%.

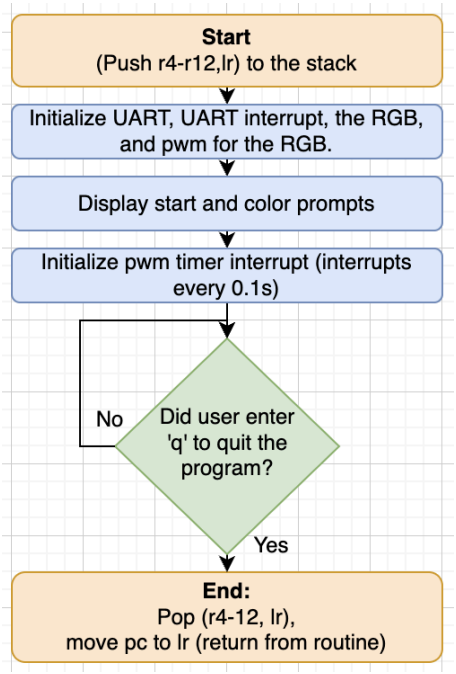
Based on user input, the compare values change in accordance with what color the entered code corresponds to. For example, the user enters 1 which is the code for orange, so the compare value for PWM2CMPB (red pin) is set to 400 (LOAD value - 1; 100% on), PWM3CMPA (blue pin) is set to 0 (for 0% duty cycle), and PWM3CMPB (green pin) is set to 200 (for 50% duty cycle). This is the process for color codes 1-9. For the rainbow, code 0, the process is more complex.

When the rainbow code is selected, the color starts as white. The light cycles through all the rainbow colors in order, white->red -> green->blue->white repeatedly.

The way the fluid color changes are achieved will be explained from red to green, but is the same respectively for the other changes (note 400 is duty cycle 100%, and respectively down to 0% at value 0). Starting at red, the PWM2CMPB is 100%, PWM3CMPA and PWM3CMPA is 0%. To transition to green, there is a register that has a set increment/decrement value. In this implementation, that value is 40, or 10% of the max compare value. The timer interrupts every 0.1s, and in this transition will check if the compare value for PWM3CMPB is at 100%. If it's not, the compare value is incremented by 40, and the handler is exited. If it is at 100%, then a flag is changed to indicate the next transition. When that occurs, the next transition is to decrement red to 0% (at this point it has taken approximately one second to transition from red 100% green 0%, to red 100% green 100%, since the increment/decrement value is 10% of the load value, and the timer interrupts 10 times every second). Now the PWM2CMPB is checked to be 0%; if it's not then the compare value is decremented and the handler is exited. If it is, then a flag is changed to indicate another new transition. At this point now the RGB is completely green. This method is executed for all of the transitions, giving the appearance of fluid color during code 0 - rainbow.

➔ High Level Flowchart

An overview of the entire program



Section 3: Subroutine Descriptions

Description of each subroutine's functionality, the arguments it passes through, and what the return values in each register are.

➔ Advanced_RGB_LED

- Functionality:
  - This subroutine is called every .1 seconds. It determines what the current status of the program is, continues to increment/decrement specific pin values, increments to the next status state once pin values are appropriate, or just changes the compare values directly to illuminate the pins with proper duty cycles based on the color code entered.
- Arguments:
  - r0 - color code (0-9)
- Return registers:
  - There are no return registers for the functionality of this subroutine.

Section 4: Subroutine Flowcharts

➔ Advanced_RGB_LED Flowchart