EGR: 226 Microcontroller Programming and Applications

Winter 2021

Instructor Prof. Trevor Ekin

**Lab 9: Using GPIO interrupts to control PWM and 7-Segment LED**

Gabriel Gasbarre

March 24th, 2021

1. Objectives

* To develop a program for the MSP432 microcontroller that interfaces to a DC motor and

controls the speed using PWM by GPIO interrupts

* To get familiar with the 7-Segment LED

Equipment

|  |  |  |
| --- | --- | --- |
| **Part** | **Description** | **Model** |
| CCS (Code Composer Studio) | Integrated development environment to develop applications for Texas Instruments embedded processors. | 10.0.00010 |
| MSP432 | Mixed-signal microcontroller family from Texas Instruments. | MSP432P401x |
| EGR:226 Lab 6 Exercise | Interfacing a keypad with the MSP432 | N/A |
| Optocoupler | electronic component that transfers electrical signals between two isolated circuits by using light. | H11B1 |
| 300 DC Motor | rotary electrical motor that converts direct current electrical energy into mechanical energy | JQ24-35H390F |
| 7-Segment Display | Common-cathode display | N/A |

1. Introduction

3.1: Part 1- Controlling the Speed of a DC Motor using TimerA in PWM Mode

For part one of the lab, students are to use their motor circuit created in the previous lab and incorporate three pushbuttons to control this motor externally. When button 1 is pressed, the speed should increase by 10%. When button 2 is pressed, the speed should decrease by 10%. When the third button is pressed, the motor should stop.

## 3.2 Part 2 - Decimal counter using 7\_Segment LED

For part 2 of the lab, students should first familiarize themselves with the 7-Segment common cathode display pinout. For this part, students must incorporate the SysTick timer and interrupts to create a one second interval. This interval will be used in combination with a custom code so that the seven-segment display shows the numbers from 0 to 9 in one second intervals.

## 3.3 Part 3 - Decimal counter using 7\_Segment LED and pushbuttons

For part 3, students are to incorporate the 7-segment display with pushbutton switches to develop a code that can increment / decrement the number displayed. Each pushbutton should operate with port interrupts. The display should also be a rolling display, meaning when incrementing up from the number 9, the display rolls over back to 0.

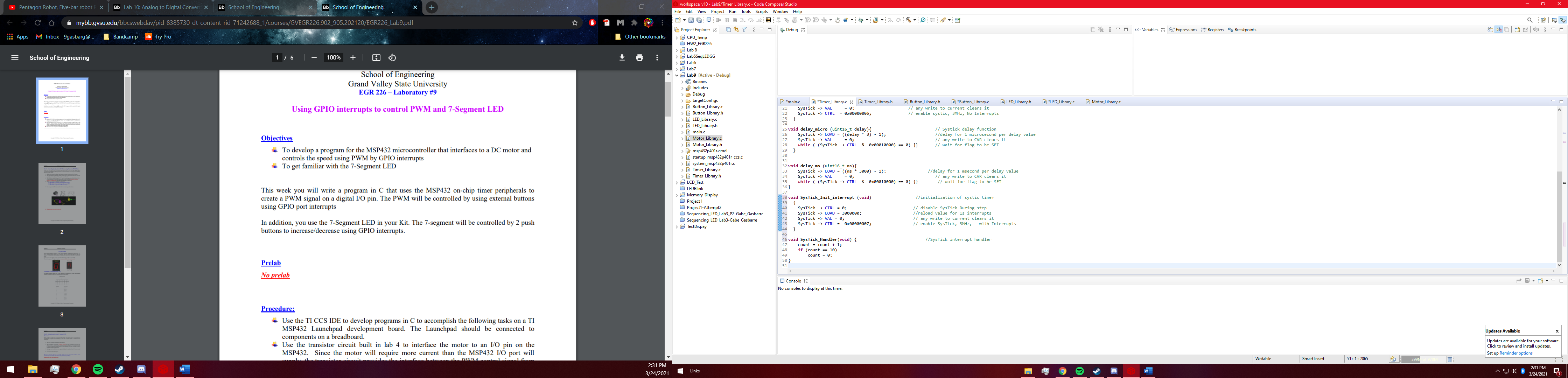
# 4. Procedure:

4.1: Part 1 - Controlling the Speed of a DC Motor using TimerA in PWM Mode

For part one, though not required, pin interrupts were used for pushbutton motor control as this is a requirement later in the lab, and it solves the issue of checking whether or not a button was pressed, saving some lines of code later down the line. Each button was configured with an internal pull up resistor for simpler hardware design, and each button interrupt was assigned a different function. For button one, an integer of 10 is added to the “percent” variable that is eventually passed to the motor control function (which accepts a duty cycle as a percentage.) This same idea was repeated for buttons 2 and 3, with button 2 removing 10%, and button 3 setting the percentage to 0.

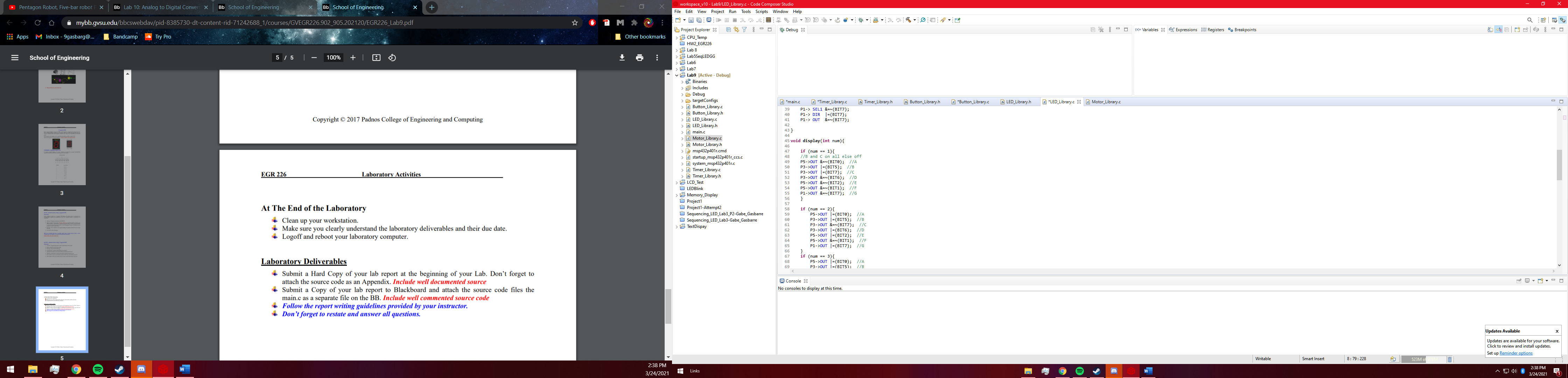
4.2: Part 2- Decimal counter using 7\_Segment LED

For part 2, the 7-segment display was incorporated into the MSP432 hardware setup using individual pins for each segment of the display, with the ground pin having a total resistance of 550Ohms going to ground for protection of the MSP432. Next, the SysTick timer initialization function was altered to be able to incorporate interrupts at a one second interval. At each interval, a value of “1” is added to the count variable in the interrupt handler for the SysTick timer. This count variable is then passed as an integer to the 7-segment display function, which automatically displays the corresponding number. Error handling is also included in the SysTick interrupt handler, such that increasing the count value above 9 rolls back to display 0 again and start the process over. Included below is the SysTick timer interrupt initialization function, along with the interrupt handler.



## 4.3 Part 3 - Decimal counter using 7\_Segment LED and pushbuttons

For part three, the already initialized pushbuttons with interrupts are re-used to instead control the 7-segment display. Using the fact that these buttons increment/ decrement the “percent” value by 10 (as discussed in part 1) This percent value is then divided by 10 to give an integer 0 through 9, which is then sent to the display function for immediate displaying. This display function works by receiving an integer, and then turning on/off each pin to display the corresponding number. A snippet of this code to display the number “1” is included below.



# Results:

* 1. Part 1 Results

Initially, part one was extremely simple as all that was to be done was add a couple buttons to the circuit in order to control the motor, which had been done multiple times before. Upon reading into the lab more, however, it was realized that using interrupts for these buttons would be much more efficient. As this was the first time using GPIO interrupts, there were certainly some finnicky issues that took a bit of time to sort out, such as what variables would be used and how these variables would be passed between not only individual functions, but libraries as well. Eventually, external volatile integers were initialized, and the motor/ pushbutton combination was working flawlessly.

5.2: Part 2 Results

Part 2 was successful in the implementation of the SysTick timer in combination with the newly introduced seven-segment display, however as always there were some complications. Getting the SysTick handler to pass a variable was again a bit of a struggle due to a few coding errors but after sorting everything out the code and hardware worked flawlessly. It was also discovered that using print statements as an error checking method seems to slow down the microchip processor, as the SysTick interrupt occurred at intervals much longer than one second.

## 5.3: Part 3 Results

Part three was by far the most satisfying as the external pushbuttons combined with the display created an entirely operational circuit that could work without being plugged into the computer. It was also at this point that the motor functions were combined with the seven-segment display which allowed the user to control the motor as well as see the operating percentage simultaneously displayed on the seven segment display.

# Conclusions.

This was a successful lab and a very informational one, not so much so in the hardware portion, however learning about the different interrupts and actually applying them is certainly something that will be implemented in later labs and the final project. It is also very satisfactory that multiple lab functions were combined to create something actually useful, which in this case was a motor with a displayed power percentage.