CMPE 443 PRINCIPLES OF EMBEDDED SYSTEMS DESIGN

LAB #003

"Data Structures"

"Pulse Width Modulation"

Motivation

In this experiment, we introduce data structures to write a driver. Recall that a driver is a piece of software which directly accesses the hardware and facilitates hardware-independent coding for the upper levels in the software hierarchy. A data structure can be used to collectively access multiple hardware registers of the same type of peripheral unit in the microcontroller.

In addition, we also introduce Pulse Width Modulation (PWM) which generates a square wave of desired frequency and duty cycle. It is also a practical means of generating an analog value by exploiting the duty cycle of a digital signal.

Hence, in this lab, you will learn to

- develop a data structure for the GPIO port
- use the developed data structure for a GPIO driver of LPC4088.
- configure the PWM ports of a microcontroller
- use the PWM function which generates a square wave with a given duty cycle.
- control brightness of a LED by using the PWM module.
- control blink period of a LED by using the PWM module.
- implement a circuit on a breadboard.

1) Problem Description

In this lab, you will use the a LED which is on the custom circuit and Joystick which is located on the Experiment Base Board. The brightness of the LED and the blink period of the LED will be changed according to the pressed Joystick button.

- When Joystick Left button is pressed, LED will blink (2 times in a second).
- When Joystick Up button is pressed, LED will be turned on with full brightness .
- When Joystick Down button is pressed, LED will be turned off.
- When Joystick Center button is pressed, LED will be turned on with half brightness .
- When Joystick Right button is pressed, LED will blink (10 times in a second).
- When Joystick no button is pressed, LED will continue to perform the last action.

*Note: When a question ends with **(*)** notation, that means write on the code. When a question ends with **(?)** notation, that means write on the paper. When you see **(*?)**, the answer of this question should be written on the paper and code.*

2) Block Diagram *1 pt*

Show the inputs and outputs of this system with a System-Level Structural Diagram. **(?)**

3) Sequence Diagram *2 pts*

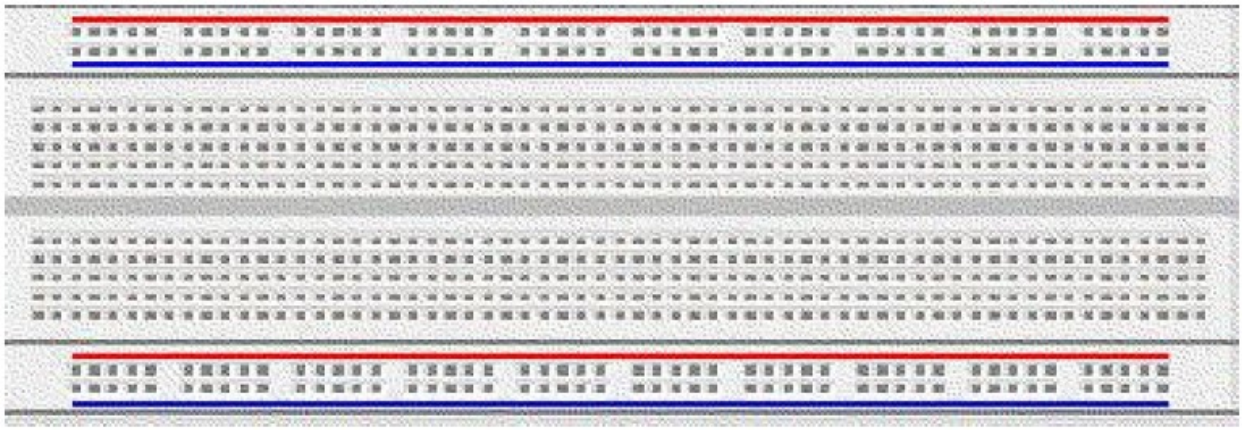
Draw the Sequence Diagram of this system. **(?)** Sequence graph must have the following blocks:

- a HW_ext_in/Pins_[X:Y], which is connected to Joystick. Define port and pin numbers for X and Y.
- a HW_ext_out/PinA, which is connected to the LED. Define port and pin numbers for A.
- a SW module

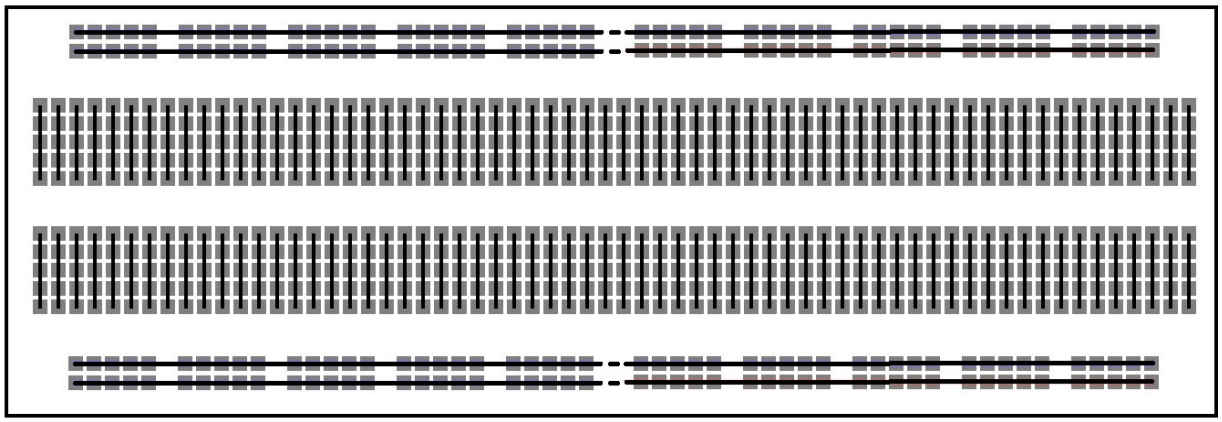
Directed edges between the blocks show how HW and SW parts interact with each other. We expect you to write variable and/or port names with their corresponding values on the edges if necessary. Note that direction of the edge is important because it shows the flow between blocks.

Write an explanation below the sequence diagram where you define your variables and values written on the edges.

4) Understanding Breadboard



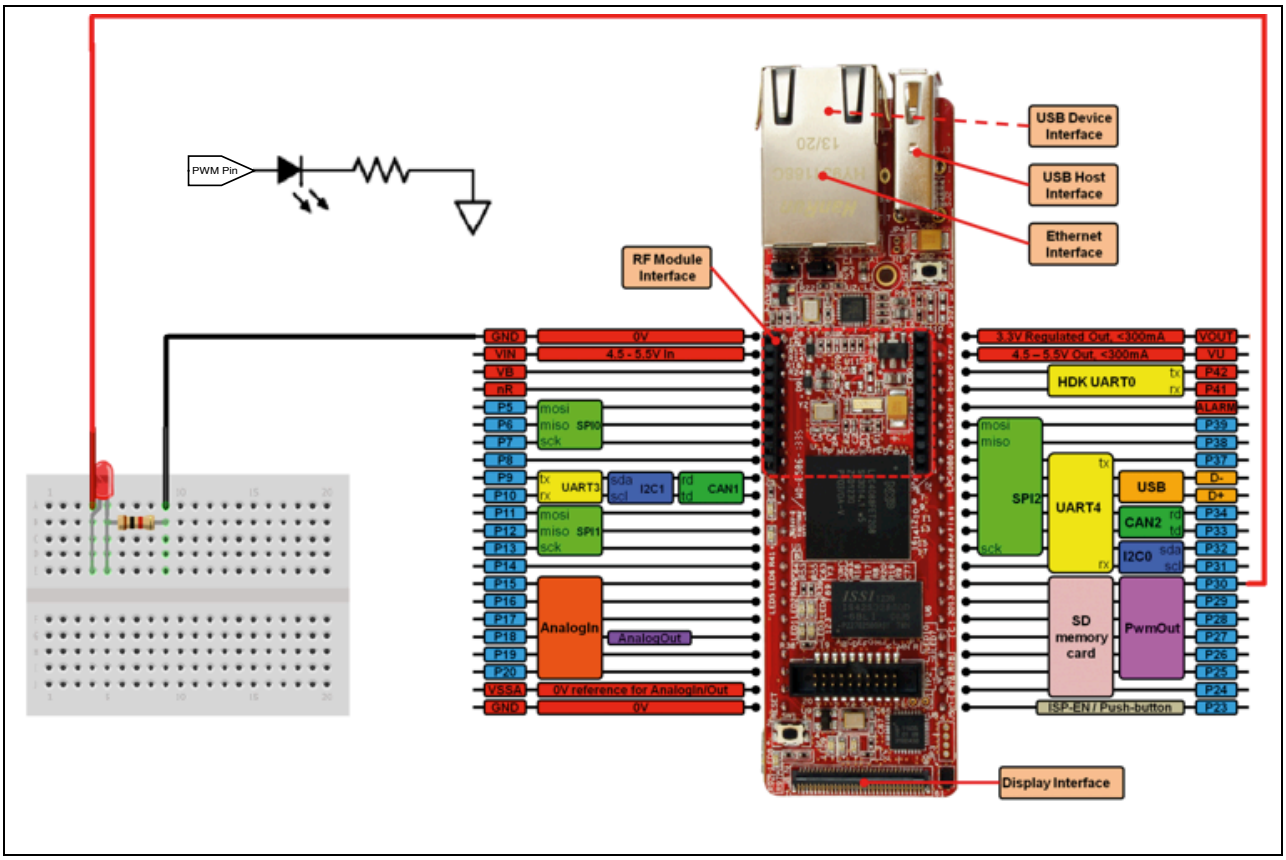
In breadboard, holes in the middle segment are connected vertically. Holes in the top and bottom segments are connected horizontally. This is shown in the below figure. Dashed line segments indicate that some breadboards have those wirings, some do not. Actually, this difference might be indicated by red and blue lines.



5) Connecting LED to LPC4088 *1 pt*

The longer leg of the LED (**ANODE**) should be connected to higher voltage and short leg of the LED (**CATHODE**) should be connected to lower voltage. For the circuit, we will give the breadboard and the **1Kohm** resistor with LED.

Component Terminals	LPC4088 Pins
LED +	P30 (P1_2)
1Kohm Resistor	GND



6) Using Struct for Port Definition *3 pts*

Data types are necessary for writing the readable codes. Therefore you will use that structure in the code. In the **Library/GPIO.h** and **Library/PWM.h** file, GPIO_TypeDef and PWM_TypeDef are defined incorrectly (Use the name in the datasheet for the variables).

- Correctly define GPIO_TypeDef struct. **(*?)** _____ *1 pt*
- Correctly define PWM_TypeDef struct. **(*?)** _____ *2 pts*

7) Initialize Pulse Width Modulators (PWM) *5 pts*

In the LPC4088, P30 (P1_2) can be used as PWM Pin. Therefore its function should be configured as PWM. In order to change the functionality of a pin, IOCON register is used.

- What is the IOCON register address of the P30 (P1_2) pin? **(*?)** (*IOCON_LED_ADDRESS*) _____ *0,5 pts*

Every pin has multiple purpose in this board.

- What is the possible functionalities of P30 (P1_2) pin? **(?)** _____ *0,5 pts*

In LPC4088, there is more than one PWM (PWM0 and PWM1).

- Which PWM is connected to the P30 (P1_2) pin? **(*?)** ***(Library/PWM.h** PWMX)* _____ *0,5 pts*
- Change the function of P30 (P1_2) pin as PWM. **(*?)** _____ *0,5 pts*

In order to use PWM, PWM should be initialized. In **Library/PWM.c** file **PWM_Init** method:

- Turn on PWM from PCONP register. **(*?)** _____ *0,5 pts*
- Enable PWM output for corresponding pin. **(*?)** _____ *0,5 pts*

MR0 register is used for controlling the PWM Cycle Rate. The Peripheral Clock Frequency for the LPC4088 is set as 60 MHz.

- Configure MR0 register for giving pulse every 20 ms. **(*?)** _____ *1 pt*

- Enable PWM Match 0 and 1 Latch. **(*?)** _____ *0,5 pts*
- Write the Correct Values to TCR for Enabling Counter and PWM. **(*?)** _____ *0,5 pts*

8) Changing Duty Cycle of the PWM *2 pts*

MR0 register determines PWM Cycle Rate, but other match registers determines the Duty Cycle. For example, when MR3 register has the value which is 30% of the MR0 register, that causes 30% Duty Cycle. When MR3 register has the value which is 80% of the MR0 register, that causes 80% Duty Cycle.

- Which match register should be used for P30 (P1_2) pin. **(?)** _____ *0,5 pts*
- Write a formula to calculate the match register for P30 (P1_2) pin **(*?)**. *In the **PWM_Write** function, value parameter will be given between 0 and 100 and according to that value, your formula should calculate the correct Match Register value.* _____ *1 pt*

When a Match Register value is changed, you should enable the use of updated PWM match values.

- Enable PWM Match Register Latch. **(*?)** _____ *0,5 pts*

In order to check the all the code, you can built your code and change *PWM_Write(0);* to *PWM_Write(100);* and run the code (100 means 100% Duty Cycle for PWM. Change this value to observe what changes on the LED).

9) Changing Cycle Rate of the PWM *2 pts*

When High Cycle Rate is used, the ON and OFF state of the LED can not be measured by the human eye (The blink of the LED cannot be seen). Therefore, in order to see the blink action on the LED, Low Cycle Rate should be used.

In this Lab, LED will blink and the brightness of the LED will be changed. Therefore, Cycle Rate should be a changeable. In **Library/PWM.c** file:

- Write a formula that changes the MR0 register value for a given parameter. (The value parameter will be given between 0 and 1000 and for example value = 20 means giving pulse every 20 ms) **(*?)** _____ *1 pt*

When a Match Register value is changed, you should enable the use of updated PWM match values.

- Enable PWM Match Register Latch for MR0. **(*?)** _____ *1 pt*

10) Changing LED Blinks with Joystick *4 pts*

In this section, you will write a code for performing these task:

- When code start, LED should be turned off.
- When you press the UP button of the Joystick, LED will be turned on (use 100% Duty Cycle)
- When you press the CENTER button of the Joystick, LED will be turned on (use 50% Duty Cycle)
- When you press the DOWN button of the Joystick, LED will be turned off.
- When you press the LEFT button of the Joystick, LED will blink (2 times in a second).
- When you press the RIGHT button of the Joystick, LED will blink (10 times in a second).

Hint: You can use **Joystick_Left_Pressed**, **Joystick_Down_Pressed …** from **Joystick.h** file. Also you can change the period of the PWM with **PWM_Cycle_Rate** method. **PWM_Cycle_Rate** method gets period as millisecond. **PWM_Cycle_Rate(20)** changes the period of the PWM pulse to 20 millisecond.