 

***MIDDLE EAST TECHNICAL UNIVERSITY***

***ELECTRICAL-ELECTRONICS ENGINEERING DEPARTMENT***

EE447 Experıment 3 prelımınary work

**Student Names:** Arda ÜNVER - Mustafa YILMAZ

**Student ID:** 2444081 - 2305746

**Submission Date:** 26.11.2023

Before starting the preliminary work, our aim was to implement the code given in the 6th page of the manual. The code is changed, to observe different colors at different times. In the main function, the init\_func () is called, initially. The init\_func () function, configures the necessary ports as outputs, enables the clock for the Port F, enables digital functionality, sets the necessary variables for the SysTick. Then, the main function enters the infinite loop (while (1)). Main code enters the ISR (Interrupt Service Routine) as the counter value (STCURRENT) of SysTick hits 0. In the ISR (SysTick\_Handler), three bits of GPIOF\_DATA are changed, while the third, second, and first bit correspond to the colors Blue, Green, Red, respectively. If all these bits, that are configured as outputs in the init\_func (), are set to 1, LED is lit Green, Blue, and Red, at the same time, resulting a White color. In other words, in order to make the LED turn cyan (combination of blue and green), GPIOF\_DATA must set to 12, which corresponds to 1100 in binary. While Green and Blue are lit, Red (the first index) is not. This exercise helped us to obtain a better understanding regarding the dynamics of the SysTick Process and how to implement the procedure in C language.

**1. (10%) Write a C function that which sends a GPIO port the necessary signals to demonstrate the Full Step Mode in both directions (clockwise or counterclockwise).**

For Question 1, the procedure is highly familiar. Previously, there were 3 output pins corresponding to 3 colors. and we were able to turn each color on, periodically. Working principles of a Step Motor resembles this. There are 4 input pins for a Step Motor. When one pin is set to high, one transistor gets activated, providing energy to the motor coil, that it's connected. At each step, one motor coil should be energized. When the motor coils are energized, one at a time and one after another, the motor rotates a constant number of degrees. When the procedure is reversed (reversing the order in which the motor coils are energized), the motor rotates in the reversed direction. The main code starts with the initial configuration of the registers and the clock. SysTick->LOAD holds the number of cycles passed before the system enters the SysTick Interrupt. A variable with the name "step" is initialized to zero before the while (1) loop. Step is incremented at each loop cycle.

0 <= Step < 5,000,000 ...................... GPIOB\_DATA is set to 0001 ---> rotation is set to 0 (CW) ----> Systick\_Handler ()

5,000,000 <= Step < 10,000,000 ............. GPIOB\_DATA is set to 1000 ---> rotation is set to 0 (CCW) ---> Systick\_Handler ()

Step = 10,000,001 .......................... Step is set to 0

Systick\_Handler () ----> rotation = 0 ----> GPIOB\_DATA is doubled (LSL) ----> if GPIOB\_DATA is 1000, set it back to 0001

|----> rotation = 1 ----> GPIOB\_DATA is halved (LSR) ----> if GPIOB\_DATA is 0001, set it back to 1000

**2. (10%) Now, you will design a system that has two inputs from push buttons and provides a step to the stepper motor upon input. One button is to provide a step for clockwise rotation and the other is for counterclockwise rotation. You will use 4 buttons of the 4x4 Keypad Module introduced in Experiment-2. Draw the necessary connections between TM4C123G, ULN2003A’s board, 4x4 Keypad Module and stepper motor.**

**3. (10%) According to your hardware design in step-2, write a C program that, in an infinite loop, gives a step upon the release of one button and gives a step in the opposite direction upon the release of the other button. You may assume that the other button is never pushed until the pressed button is released. The response of the motor should be after the button release. You should be aware of bouncing inherent in the buttons.**

**4. (5%) At this stage, you will design a system that has 4 inputs from push buttons to control a stepper motor. One button is for speeding up, one is for slowing down, the other two are for directions. You will use 4 buttons of the 4x4 Keypad Module introduced in Experiment-2. Draw the necessary connections between TM4C123G, ULN2003A’s board, 4x4 Keypad Module and stepper motor.**

**5. (65%) According to your hardware design in part-4, write a C program that, in an infinite loop, drives a stepper motor speed and direction of which can be controlled by external push buttons. 8 You may assume that no button is never pushed until a pressed button is released. The controls should be applied upon releasing the corresponding button. You should be aware of bouncing inherent in the buttons. For this item, please explain how you attack the problem and draw a flowchart of your algorithm.**

Port B is used for Step Motor PB0,PB1,PB2,PB3 as outputs

(connect 12V to VBus, GND to GND)

Port E is used for Push Buttons PE0,PE1,PE2,PE3 as inputs

(connect VCC to +3.3V, GND to GND)

rotation = -1 means CW rotation (triggered with PushButton1)

rotation = 0 means idle mode (no rotation)

rotation = 1 means CCW rotation (triggered with PushButton2)

Initial value of SysTick->LOAD determines the initial speed.

Change of GPIOB->DATA can be observed at

memory location of 0x400053FC.

GPIOB->DATA will be either one of the following: 1,2,4,8, indicating the motor coils.

Remark: You can also call for another function in SysTick\_Handler.

void SysTick\_Handler (void){

button\_pressed(rot);

}

void button\_pressed(rot){

....

Write your code here

....

}

Change of GPIOE->DATA can be observed at

memory location of 0x400243FC.

GPIOE->DATA is 0x0F by default.

When PushButton1 is pressed, GPIOE->DATA = 0x0E.

When PushButton1 is pressed, GPIOE->DATA = 0x0D.

SPEED DOWN by increasing the period of the SysTick Interrupt.

SPEED UP by decreasing the period of the SysTick Interrupt.