MECHTREON 3TA4 Lab 2
Reation Time Tester

Introduction

This lab introduces you to more details about the software and hardware development tools you will use this semester to work with the STM32L476G Discovery board and STM32L476 Cortex-M4 MCU. In this lab you will be asked to implement a reaction time tester. The purpose is to see how fast you can push a button after an LED light flashes and to display that time on an LCD screen. You will interface the MCU with the joystick and the LCD unit on the Discovery board.

Prelab [5 pts]

1. This lab assumes you are familiar with the material required in the prelab reading for Lab1.

2. Reading Chapter 11 Interrupts (particularly sec 11.8) and Chapter 15 General Purpose Timers of "Embedded Systems with ARM Cortex-M Microcontrollers in Assembly and C" by Yifeng Zhu.

3. Read up on Finite State Machines (FSMs) in this `IEEE Software Article`.

4. Read about the external interrupts of the STM32L476 MCU in Chapter 13 and 14 of the `MCU's Reference Manual`.

5. Read about the `overview of the HAL drivers` for STM32L4 in Chapter 2 of the `HAL library user manual`. Especially pay attention to Section 2.11, `the HAL system peripheral handlling`, in which how the GPIO and EXTI are handled by the HAL drivers are talked about . Also read about the `HAL System Driver` in Chapter 5.

6. Read and try to understand the codes provided by the dedicated BSP drivers for STM32L476G-DISCO boards (`STM32Cube_FW_14_V1.8.0_Drivers\BSP\STM32L476G-Discovery\`). The BSP drivers are for controlling the components, such as LEDs, joystick and LCD module, on the boards. Pay attention to files `stm32l476g_discovery_glass_lcd.c and stm32l476g_discovery.c`.

7. Draw a Finite State Machine (FSM) to implement the behavior described in the Requirements section below. Consider using a table too to describe the system behavior for different external inputs to make sure you have completely and consistently specified the behavior.

8. Start to modify the Lab2 starter code based upon the prelab readings and the lab Procedure requirements below. Upload the modified files and FSM (in a document or picture file) to your folder on Avenue. One copy per group please.

Note: To receive full marks for the prelab, upload your modified lab 2 code to your folder on Avenue before 2nd lab session begins.

Hardware

1. **LCD**

   The LCD module mounted on the STM32L476G-DISCO board has 24 segments with 4 commons. It is capable tof displaying 6 letters, but with the function `BSP_LCD_GLASS_ScrollSentence()`, which is provided the BSP drivers, users can display long sentences.

2. **Timer Configureation**

   The STM32L476 board comes with serveral timers, among them, TIM2, TIM3, TIM4 and TIIM5 are general purpose timers. They are all attached to the APB1 bus. These four timers frequencies are determined by the APB1 bus frequency.

   Please read Section 6.2 of the `MCU's Reference Manual` for STM32L476-DISCO boards to understand the clock tree of the board, and read Section 6.2.15 to unserstand its timers' clocks.

   In our starter kit project, the MSI is choosed to be the clock source for the board and it's frequency is set to 4MHz. The AHB bus' prescaler and APB1 bus' prescaller are both set to 1, therefore the frequency of the APB1 bus is also 4MHz. The timers attached to this bus all have original frequency of 4MHz.

   The timer's frequency can be further slowed down by chooseing and setting a timer's prescaler.

   A timer can generate interrupts on different events. For example, when the timer counter overflow it may generate an update interrupt, when the timer counter counts to a specific value stored in the output compare register, it may genterate an outpout compare interrupt. A counter has four independent channels. Each channel can work independently for certain type of tasks. For example, one timer can independently generate four ouptput compare interrupts at different frequencies.

   For more details about the features and functions of the timers, please read Chapter 31 of the `MCU's Reference Manual` for the STM32L476-DISCO boards.

3. **EEPROM**

   The STM32L476G-DISCO board does not have internal EEPROM, but the EEPROM can be emulated using FLASH memory. Look through the example code for STM32L476G_EVAL board at `STM32Cube_FW_14_V1.8.0_Projects\STM32L476_EVEAL\Applications\EEPROM\EEPROM_Emulation\Src\eeprom.c` to see how EEPROM is emulated using FLASH, and also read the `EEPROM Emulation Manual`. However, the EEPROM emulation liabray file used in our lab starter project is modified fom a library file for STM32F4 (because when we first tested the the STM32L4 board in 2016, the emulation file for STM32L4 was not available). Please carefully study how EEPROM emulation works since related questions may be asked in a quiz.

4. **Random Number Generator (RNG)**

   This lab requires your application program to have a random waiting time at certain stage, so you may need to use a random number. The MCU of STM32L476G-DISCO boards has a true random number generator, which can generate random numbers of 32 bits. Please read the appropriate chapters in both the `MCU's Reference Manual` and the `HAL library user manual` for the freatures of RNG and for how to use the RNG. Figure out your solution to use these 32 bits numbers for your needs (You do not need a number is such a big range).

Procedure

NOTE: If you eventually can not get your user application fully function, you may still get certain points by completing steps in this section. Remember to demonstrate to TAs at the steps to get points.

1. Download the starter package (lab2starter.zip) and unpackage it. Place the project folder in the directory structure as suggested in Lab 1, i.e., the project folder and the the firmware package folder are in the same directory, as this is the lab starter project's default setting.
2. Read through and understand the sample code.
3. Build the project.
4. Program the STM32L476G Discovery board using the ST-Link programmer.
5. Push the Reset button. This starts running the code on the STM32L476 MCU.
6. [**5 pts.**] Configure the joystick and make it to trigger interrupts when press different directions. You will use some of these EXTIs to control your user application.
7. [**5 pts.**] Configure your timer. Make your timer to trigger an interrupt at 1 Hz frequency, and make LED blink through timer interrupt. (You may need more than one timers in your user application).
8. [**5 pts.**] Write your testing code to test your emulated EEPROM. Make writing into and reading from EEPROM work. You will need to use EEPROM to save timing record.
9. [**5 pts.**] Write your testing code to generate random numbers ranging ROUGHLY from 0 to 4000. Later on you can easily covert these numbers to 0 to about 4 seconds of time.
10. [**40 pts.**] Complete your user application according to following lab requirements.

Requirements

1. All timing must be done with interrupt-driven hardware timers and not with software wait-loops. DO NOT use the Delay function, your program will not work correctly if you do not set the interrupts' priorities in a appropriate order.
2. All programs must be in C.
3. Your program should implement the following behaviour:

- Upon a reset signal, flash the LEDs on the development board at a few Hz and wait for a user button (The selection, or a direction of the joystick, for example) press.
- At the user button press, turn the LEDs off for about 2 seconds. Figure out a way to make the waiting time random, so that it is harder to predict. You are required to use the RNG available on the MCU.
- After the variable waiting time, turn on the LEDs. At the same time, start a timer and display the timer count in millisecond on the LCD, and wait for a user button press. (your code should detect the cheating condition of a pressed button at t=0 and respond by returning to the flashing LEDs).
- At the button press, measure the time in millisecond from the moment when LEDs became on to the moment when the button is pressed. Display the timing result on LCD. If the timing result is the best, save it in the emulated EEPROM.
- Return to the flashing LED state when a user button is pressed at any time/state.
- When a user button is pressed, display the the saved fastest time on LCD.

Assignment

1. [**60 pts. (The total 60 pts in the Procedure section)**] A functional program that implements all the requirements correctly.
2. [**15 pts.**] A heavily commented C code. Clear and concise comments are very important for others to understand your code and even for you later on when you need to remember what you did.
3. [**20 pts.**] Motivate your design and implementation decisions to your TA:
    - How would you measure time-base accuracy?
    - Why did we make you use interrupts rather than wait-loops for timing?
    - The scheme you used to detect the pushbutton state (e.g. polling loop, interrupt).
    - Why did we store the best test record in the emulated EEPROM? What's special about EEPROM?
    - Other design aspects of the assignment.