ECEN-361 Lab-02:Clks, Timers, Interrupts

# NAME: \_Michael Munoz\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction and Objective of the Lab

The objectives of this lab are as follows:

* Part 1: Load theLED-D1 Blinky working with a simple timer-based interrupt.

Add two more timers to blink LED\_D2 and LED\_D3 with differing rates.

* Part 2: Reconfigure the timer clock to see the effects of changing the clock source and parameters.
* Part 3: Use a built-in timer to count the time of an external event (button push)

This will be done with a fun Reaction timer.

For each of the parts, follow the instructions, then fill in answers to the questions. Expected answers are indicated in the boxes with red text/spaces to fill in answers.

The submission for this lab is simply the repository that you’ll modify. Your modifications get pushed back to github.com. Your responses, as recorded in this file, will be checked along with your running project.

# Part 1: Instructions

## Clone the repo for this file

By now, you should’ve completed the assignment ([HERE](https://byui.instructure.com/courses/251502/files/113360890?module_item_id=31336740)) about getting on and using GitHub.

Start by accepting the assignment in GitHub Classroom for ECEN-361-Lab-02. Clone the repo.

## Add the repo project to your workspace.

Import with File/Import and point to the directory of the newly cloned project:

Clean Project then Build Project:

A screenshot of a computer

Description automatically generated

There should be no errors or warnings.

## Run the project.

The project should simply blink the D1\_LED once per second.

No seven-segment display.

## Create 2 more timer interrupts that blink as follows:

D2\_LED: Once every 500 mS.

D3\_LED: Once every 250 mS.

Do this by using the GUI (click on the MX -- .ioc file). Note that two of the timers are already taken:

* DON’T USE TIM17 – it’s dedicated to displaying the seven-segment lights
* DON’T’USE TIM16 – Note that it’s doing D1 at 1 second.

Note that three things have to happen to make a timer-based interrupt work:

1. Timer has to be initialized (this code is generated by the GUI – see   
   **MX\_TIM17\_Init();**
2. Timer has to be started – You put this in main.c see:   
   **HAL\_TIM\_Base\_Start\_IT(&htim17);**
3. ISR has to be defined -- You put this in main.c see:

HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)

When completed answer the following questions:

## Part 1 Questions (2pts)

Note the speed of D1/D2/D3 – they should seem like a 3-bit binary counter.

How fast does D1 turn on/off?

\_\_1Hz\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Do they all toggle at exactly the same time?

\_All of the LEDs do not toggle at exactly the same time, EVERY time, but there is a particular moment when all three LEDs toggle OFF at the exact same time, and that is approximately 3secs after displaying a binary 7, when toggling off to display a binary 0.\_\_\_\_\_\_\_\_

## Part 2: Changing the clock tree

Steps:

1. Open the ioc Configuration GUI
2. Change the APB1 and ABP2-Prescalers to “/8” (Changing both of them guarantees that whatever timer you chose will be affected.)

A computer screen shot of a diagram

Description automatically generated

1. Compile and re-run and observe the behavior of the LEDs

## Part 2 Questions (2pts)

What has happened to the speed of the timers?

\_\_The speed of the timers has decreased dramatically.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is the new frequency of LED D1?:

\_250mHz\_\_\_\_\_\_\_\_\_\_\_\_\_

Why didn’t the Seven-Segment Light update rate change?

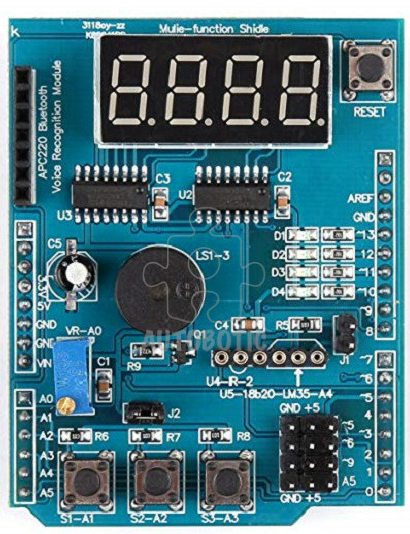
\_\_The seven segment display rate of change did update, however we cannot perceive the change due to the frequency being higher than we can notice with the naked eye. The timer used for the seven segment display is TIM17 which is tied to APB2, which was also prescaled by a factor of 8, indicated the clock speed changed.\_

## Part 3: Reaction Timer

In addition to performing useful tasks at set intervals, timers can also be used to measure elapsed time of an event. The events can be triggered by software, or by a hardware input.

For this part of the lab, we’ll make a small “reaction timer” that measures how fast your hand/eye coordination can be, in milliseconds.

We’ll define the buttons and display as shown:



**Reaction Time**

**In MilliSeconds**

FASTEST

STOP

START

START button: Initiates a random wait. After the random wait, all the SevenSeg lights go on

As soon as the lights go on, a timer starts counting milliseconds

STOP button: Stops the millisecond reaction timer and shows it on the display

FASTEST button: Extra Credit – This button shows the fastest speed.

Code for this part is organized in the **ReactionTester.c** source file and **main.c**. Fill in between the comments:

/\* Student Start HERE \*/

/\* Student End HERE \*/

Read thru the comments in the code. Most of the structure is in place, and you should only have to modify places between Student\_Start / Student\_End.

Note that for the reaction timer to be accurate, if you changed the prescaler above in Part2, you’ll need to reset it back to the default of no-prescale, X1.

## Extra-Credit Opportunities (5pts for any of the following)

* In the current code, there’s no penalty for “Cheating” by pushing the stop button before all the “Go” lights turn on. Implement some sort of indicator that the Stop button was pushed prematurely.
* Change the “Go” lights to be all of the D1..4 LEDs instead of display all ‘8888’ on the SevenSegments.

I made all four LEDs, D1-D4 turn on when the GO lights are turned on alongside the seven segment display displaying “8888”.

* Make the final reaction time flash on/off

If you do any of these items – just mention what and how it worked, below: