**ECE0202: Embedded Systems and Interfacing**

**Lab 6: Real-Time Clock using SysTick (in C)**

**Due: April 25th 11:59pm**

**Objectives**

* **Use the SysTick timer interrupts to periodically read registers.**
* **Program and use the Real-Time Clock (RTC) module.**

**Deliverables**

* **(40 points) Code that uses SysTick to generate an interrupt every 1 ms. In the SysTick Handler, read the RTC clock and display the current time (hours, minutes, and seconds) on the OLED every 1 s.**
* **(10 points) Answers to the pre-lab questions.**
* **(50 points) A demonstration of the operational system to an instructor or TA.** **Indicate how each group member participates and contributes to the lab at the end of the lab report.**
* **10 points extra credit: Set an alarm and flash the on-board LEDs when the alarm goes off.**

**Please submit your code as \*.c files**

David Anderson and Jingtong Hu – 10/23/2019

**Getting Started**

Watch YouTube tutorial:

• SysTick, https://www.youtube.com/watch?v=aLCUDv\_fgoU (11 minutes)

Download the C lab template. Write your code in the “main. c” file.

You need to call a few deep-level system configuration functions in order to start and configure the clocks and power distribution for the system. Your program will need to call functions from the included files in the following order:

1. System\_Clock\_Init
2. I2C\_GPIO\_init()
3. I2C\_Initialization(I2C1)
4. ssd1306\_Init()
5. RTC\_Clock\_Init
6. Following these function calls, follow the steps to configure the RTC module according to the section below.
7. Following the configuration of RTC, follow the instructions below to configure the SysTick module and interrupt the handler.

**Display the time using RTC periodically**

Use the SysTick module to periodically generate interrupts that will read the Real-Time Clock (RTC). The SysTick interrupt should be generated every 1 ms. After each second, which means that every time the total amount of SysTick interrupts is 1000, the OLED should display the current time (hours, minutes, and seconds using 24h format) that is stored in the RTC module. For instance, if the time in the RTC is 13:12:56, display 131256 on the OLED screen. The illustration of the OLED is shown as follows.

|  |  |
| --- | --- |
|  | * 0.96” OLED display module * Driver chip: SSD 1306 * Resolution: 128×64 pixels * Working voltage: 2.7V – 5.5V * Power consumption: 0.04W during normal operation * I2C Interface, default I2C address: 0x78 * SCA and SDA have already been pulled up to Vcc on the board. |

**RTC Configuration**

Before you use the RTC to provide the calendar time and date, you need to configure the RTC module first. The steps to initialize the RTC are:

1. Initialize the clock for RTC. You need to select LSE as the clock source. This is done by calling the RTL\_Clock\_Init() function that is defined in the RTC.c code.
2. Disable the RTC register write protection. First, write “CA”, then “53” into the WPR register.
3. Enter initialization mode by setting the “init” bit in the ISR register.
4. Wait for confirmation of the initialization by waiting for the INITF bit to be set in the ISR register.
5. Set the RTC to an initial time value. In this lab, we do not care about the date, so you only need to set the value in the time register (TR). In this lab, set the initial time as 13:59:30.
6. Exit initialization mode by clearing the “init” bit of the ISR register.
7. Enable write protection for the RTC by writing “FF” into the WPR register.

**SysTick Configuration**

In this lab, in order to generate a SysTick interrupt with a period of 1 ms, we need to configure the clock for the system timer (SysTick) first. By calling the System\_Clock\_Init() function that is defined in SysClock.c, the clock for SysTick is configured to be 80 MHz

You need to generate SysTick interrupts with a period of 1 ms. In order to do that, you need to configure the related registers. The steps are as follows:

1. Disable the SysTick IRQ by clearing the Tickint bit in the CTRL register.
2. Set the SysTick reload value register (LOAD). The frequency of the clock is 1ms. The formula calculating the value to load looks like this:
3. Reset the SysTick counter value register (VAL) by writing a value of 0 to it.
4. Set SysTick->CTRL to use the internal processor clock rather than the external clock.
5. Set SysTick->CTRL to enable interrupts (the TICKINT field).
6. Set SysTick->CTRL to enable the timer via the ENABLE field.

**Writing the SysTick Interrupt Handler**

After you configure and enable the SysTick interrupt, every time a SysTick interrupt signal is detected, the processor will be forced to respond to this interrupt. It will auto-stack the related registers, execute the interrupt handler, then auto-unstack the values stored on the stack and return to executing the main program thread. As a programmer, you only need to be in charge of writing the SysTick interrupt handler, which means you need to tell the processor what to do after a SysTick interrupt is generated. The SysTick interrupt handler is called SysTick\_Handler().

**Using the** **DisplayString function in C**

You will need to display characters on the OLED screen. The OLED is connected with the Discovery board via I2C. The circuit is as follows.

A picture containing text, electronics

Description automatically generated

The template in this lab already includes functions to achieve I2C communication and SSD 1306 driver configuration. The only thing you need to do is to call the following functions in your DisplayString(char\* message) function.

ssd1306\_Fill(White);

ssd1306\_SetCursor(2,0);

ssd1306\_WriteString(message, Font\_11x18, Black);

ssd1306\_UpdateScreen();

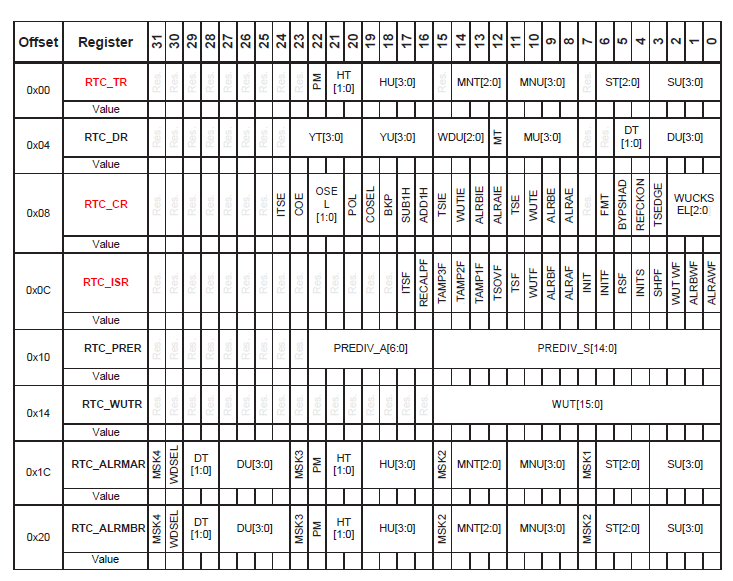
Following the function call, the characters will be displayed on the OLED screen.

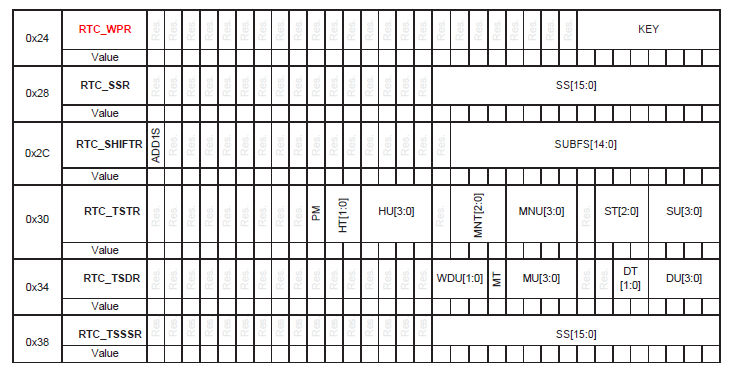
Remember that the number you read from the RTC Time Register will be an int, and the OLED displays character data! You will need to convert the ints into character data in order to display it properly.

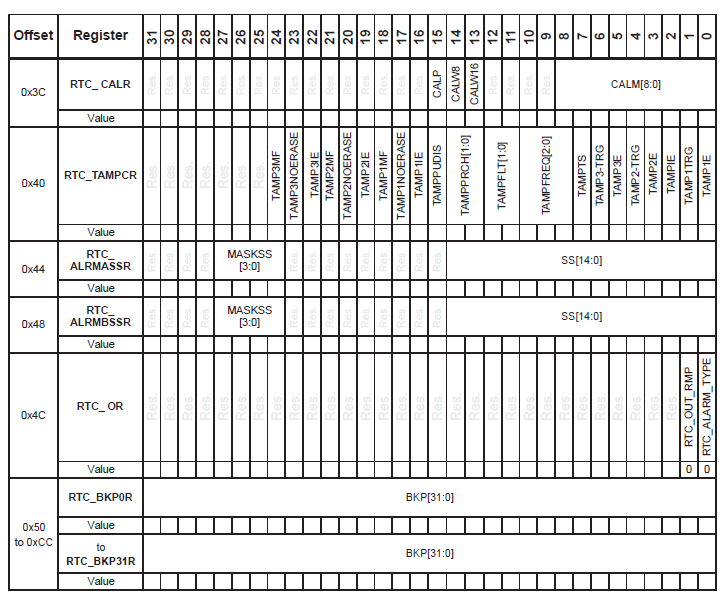
This lab requires to read the RTC clock and display the current time every 1 s, so you may need to write a delay() function.

**Pre-Lab Questions**

The registers related to the RTC module are shown below. The registers that we care about in this lab are RTC\_TR, RTC\_ISR, and RTC\_WPR. The other registers, however, may be used for extra credit and for other related tasks.







**Participation and Contribution**

Please indicate the participation and contribution for each group member using the following table.

|  |  |
| --- | --- |
| Name | Participation and Contribution |
|  |  |
|  |  |