**EGRE 364 – Microcomputer System**

**Laboratory Number 8**

**Light up a LED via EXTI**

**Lab Section: Tuesday 1PM**

***Lab conducted on: November 12th , 2019***

***Report Submitted on: November 22th , 2018***

**Papa Beye & Omar Amr**

**Major: CPE & CPE**

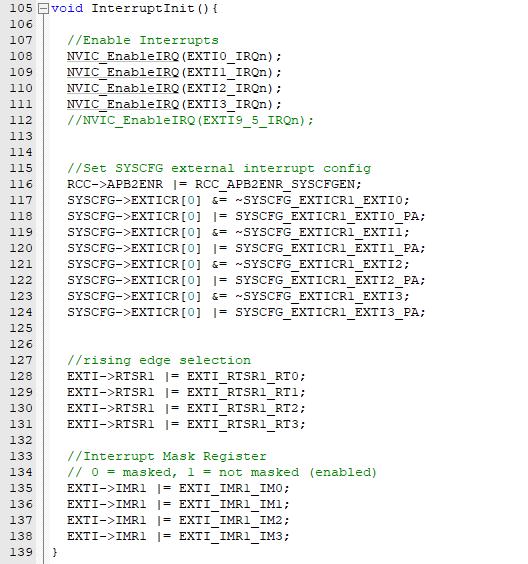
**Introduction**

The purpose of this lab was to utilize the GPIO Output Data Register for the two LEDs from the first lab, red and green, and interrupts to be able to repeatedly blink the red LED and use an external interrupt to toggle the green LED simultaneously using C language in the Keil µVision software development environment. To achieve this objective, we used GPIO registers, internal interrupt SysTick, and initialized external interrupts to be able to set up the interrupt handlers to complete the objective of using interrupts to read push buttons to complete a task simultaneously with the blinking red LED. The main objective of this lab was to program the microcontroller to flash the red LED periodically and use EXTI to read a button push to toggle the green LED and to do something cool. We decide to control the speed of the flashing red LED using two button and two external interrupts to increase or decrease the flashing period. We used the previously utilized joystick buttons on board the STM32L4 microcontroller. We used the center button (PA0) to toggle the green LED and the left button (PA1) to increase rate of the flashing red LED and the right button (PA2) to decrease the rate. Using C allowed us to approach the lab with a higher level of abstraction to complete our implementation of external interrupts. In this lab, we initialized GPIO ports and manipulated the ODR values and set & enable certain registers to initialize the external interrupts. The configuration used for this lab can be see below in the C code. We implemented a delay function to control the period of the flashing red LED by utilizing the System Timer, which uses the SysTick interrupt that triggers every 1 ms and allowing us to accurately set the delay between flashes of the red LED to 1 sec. In this lab, we used the practiced methods and material that we learned in class about internal and external interrupts and overall microcontroller processes from Lab 1.

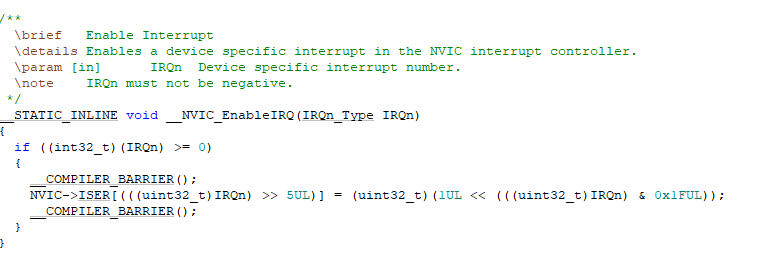
**Functionality and Correctness**

The STM32L4 discovery board has two LEDs connected to PE8 & PB2 pins of the board processor. Similar to lab 1, we enabled the clock of both ports, set the mode of the pins as outputs, and set the value of the output of each pin using the ODR to toggle the LED on and off. We used the delay function shown below to create a delay right after we toggle the value of the ODR value of the red LED (PB2). Our delay used the SysTick interrupts ability to interrupt every 1 ms, allowing us to delay for 1000 ms, or 1 sec, for the periodic flashing of the red LED. Using a global variable, we used two other external interrupts from the left button (EXTI1) and the right button (EXTI2) to alter the delay variable to simulate a quicker or slower flashing rate. In order to the 3 external interrupts, we must initialize them beforehand. First, like in the SysTick initialization, we set enable the interrupt number of EXTI0, EXTI1, EXTI2 by using the given function NVIC\_EnableIRQ(), which is detailed below. Next, we set the SYSCFG external interrupt configuration for EXTI0, 1, 2 by clearing their respective bit in the control register and then set the bit. In order to cause the external interrupt with the press of the button, transition from ‘0’ to ‘1’, we need to set the external interrupts as rising edge by setting the EXTI rising edge trigger selection register for each respective bit. Finally, we needed set each respective bit of EXTI 0, 1, 2 of the EXTI interrupt mask register. The initialization documented here is detailed below. After initialization, we were able to write the interrupt handler, which are called by the NVIC when the external interrupt is triggered. In each interrupt handler, we had to clear the corresponding pending bit to allow future interrupts by writing the pending to a “1”. Doing so, we were to complete tasks inside each interrupt handler, such as increasing or decreasing the delay or toggling the green LED.

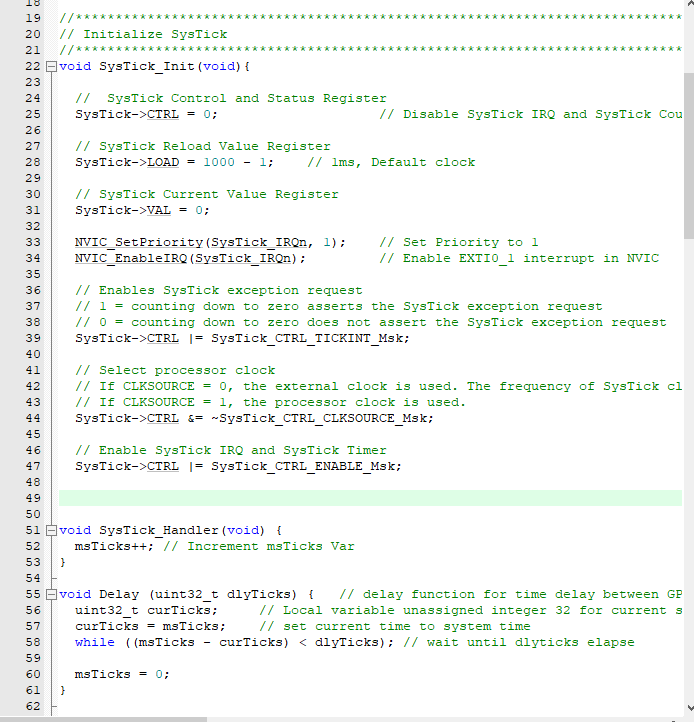
**C Code to configure EXTI 0,1,2,3**

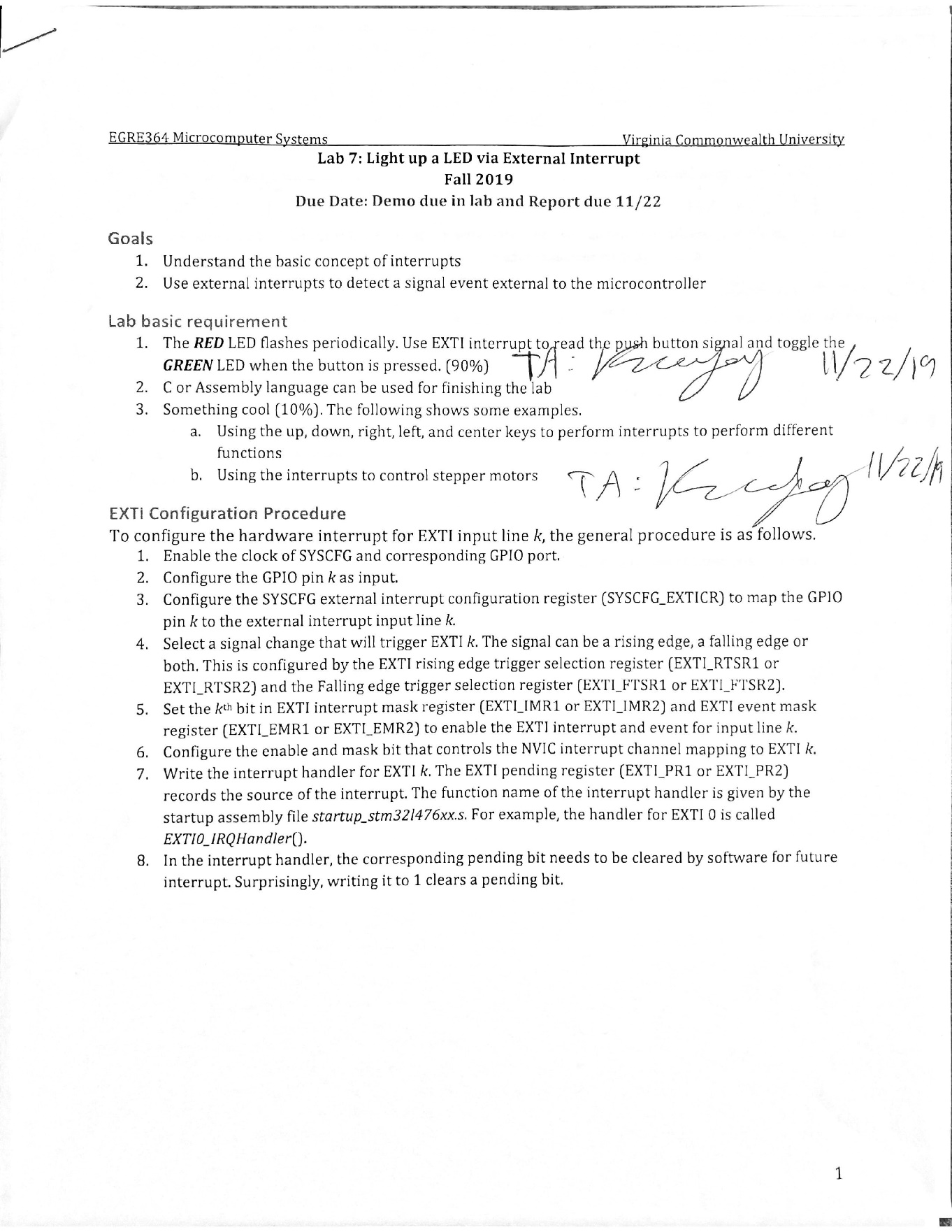
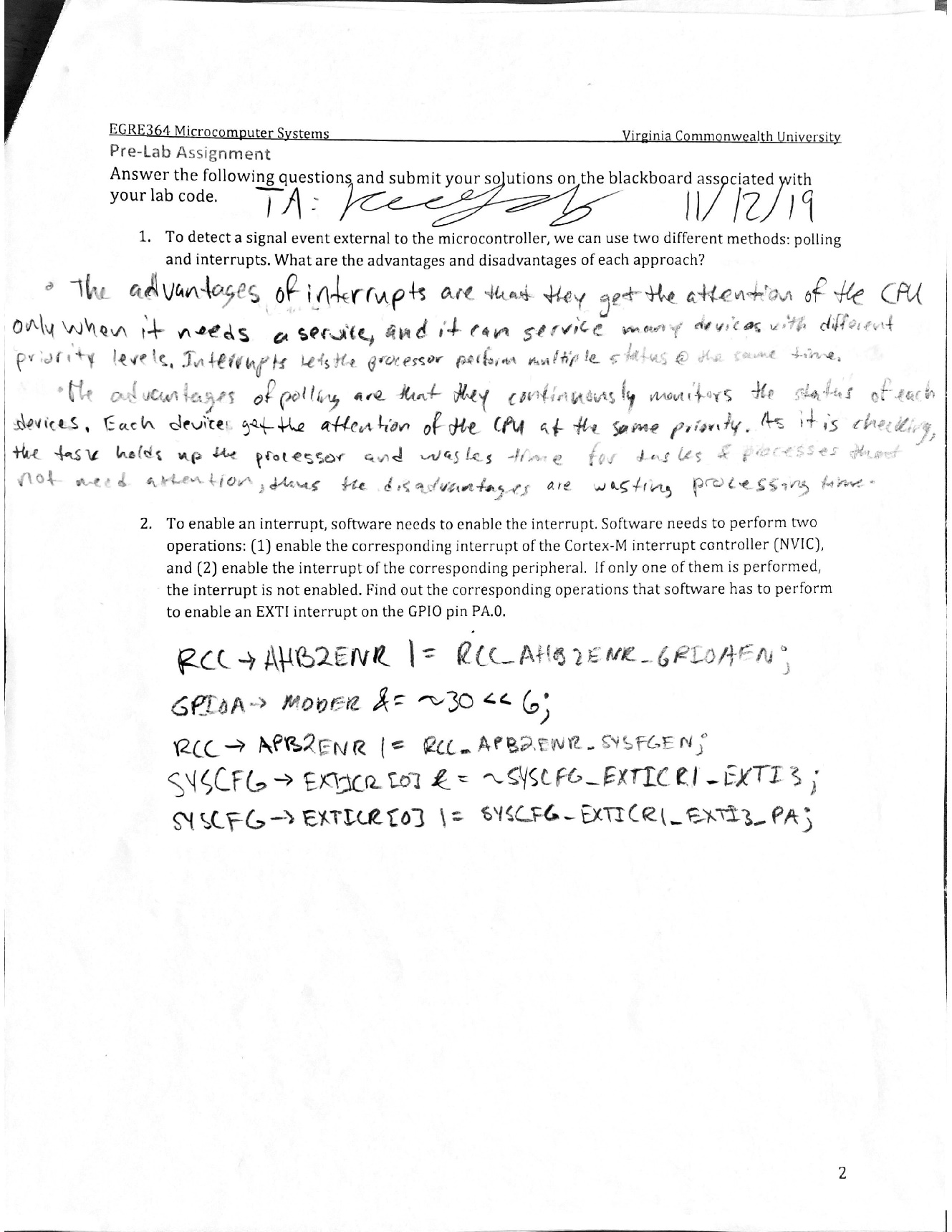


**Details of EnableIRQ function**

****

**Delay Code using SysTicks**

****

**Pre-Lab and Demo**

Omar Amr Papa Beye

**Post-Lab**

1. If the interrupt pending flag is not clear, set to “1”, then the microcontroller will think that another interrupt is being called and will repeatedly call the interrupt handler, causing an infinite loop of the interrupt handler and the processor getting hung on.

**Conclusion**

In this lab, we became more comfortable with interfacing with practical GPIO initialization and utilization alongside external interrupts to occur while flashing the red LED. In this lab, we learned about initializing and utilizing external interrupts to complete tasks and how to apply it with our STM32L4 board. We were able to achieve the objective of flashing the red LED and toggling the green LED with an external interrupt; we also implemented two other external interrupts using the two side buttons on the joystick to increase or decrease the rate/delay of the flashing LED. We used previous knowledge of implementing the flashing green and red LED from lab 1 to complete this lab. We learned a lot and gained more experience about how to use external interrupts and how they work on the STM32L4 board. We didn’t face any issues but there was a learning curve that allowed us to learn and understanding how initialized and use external interrupts in our lab to properly complete our objective.