Peer Review #2

This review is for Lab 3: Interrupt Service Routines in C. The goal of this lab is to become familiar with the interrupt system, and once again go over the setting up of I/O pins in the uVision environment. This lab also uses those tools to assist in creation of a counter that on the outside looks like the same as Lab 2, however, uses much less data by using interrupts instead of constant polling of switches. Used in this lab includes: A breadboard (with studio), a TM32 NUCLEO-L432KC Board, wires, and a .c file. The objective of this lab was to create a loop that would start/stop counting if a button was pushed, and go up/down if another button was pushed. We were also to use the Logic Analyzer in Waveforms to verify the functionality of our program, which helps to become more familiar with the various tools given to us in Waveforms.

**The program written obtains the following steps:**

1. Establish variables used, pins set up, and interrupts setup
2. Enter an infinite loop where we only call the two functions: Delay and Count
3. If an interrupt is trigged, go to that interrupts handler
   1. If it was EXTI1 (button 1 was pressed) then start or stop the count
   2. If it was EXTI2 (button 2 was pressed) then change the direction of the count
4. Then go back to the beginning of the infinite loop

My code worked very well and made the overall lab go quickly. The connections I made match the connections given to use in Table 1.

Table

Description automatically generated

Table 1

After setting up the lab and testing it out, I had a few minor problems with the original code I brought into lab with me. My main problem was I had misunderstood the assignment of buttons 1 and 2. Originally, I had that button 1 would start the program, and set the count to count up. Then button 2 would only set the count to count down. This was an easy fix that I applied in the lab by adding a few statements. Then setting up the Logic Analyzer was simple as long as I pay close attention to the wire being connected, since the wires are so close together on the DIO port, it is easy to mix them up and have a simple error in the lab. The figure in figure 1 shows the DIO ports on Waveforms while my program is running, and figure 2 shows the logic analyzer capture my results for this lab.

**The code used for this lab is shown on the next page.**

**A screenshot of a game

Description automatically generated with medium confidence**

Figure 2

A picture containing application

Description automatically generated

Figure 3

/\*--------------------------------------

Gabriel Emerson

ELEC 3040

Lab 3

8/30/21

---------------------------------------\*/

#include "stm32l432xx.h" /\* Microcontroller information \*/

/\* Define global variables \*/

static int counter;

static int counter2;

static int run;

static int up;

/\*---------------------------------------------------\*/

/\* Initialize GPIO pins used in the program \*/

/\* PA1 = push button \*/

/\* PA2 = push button \*/

/\* PB3 = blue LED, PB4 = green LED \*/

/\*---------------------------------------------------\*/

static void PinSetup () {

/\* Configure PA1 as input pin to read switch \*/

RCC->AHB2ENR |= 0x01; /\* Enable GPIOA clock (bit 0) \*/

GPIOA->MODER &= ~(0x03FFFC3C); /\* General purpose input mode \*/

GPIOA->MODER |= (0x01555400); /\* General purpose output mode\*/

/\* Configure PB4,PB3 as output pins to drive LEDs \*/

RCC->AHB2ENR |= 0x02; /\* Enable GPIOB clock (bit 1) \*/

GPIOB->MODER &= ~(0x000003C0); /\* Clear PB4-PB3 mode bits \*/

GPIOB->MODER |= (0x00000140); /\* General purpose output mode for PB4-PB3\*/

}

static void interrupt\_setup() {

/\*\* RCC->APB2ENR |= 0x01; // Turn on clock for SYSCFG

SYSCFG->EXTICR[0] &= ~SYSCFG\_EXTICR1\_EXTI1; //Clear EXTI1 bit in config register

SYSCFG->EXTICR[0] |= SYSCFG\_EXTICR1\_EXTI1\_PA; //Select PA1 as interrupt source

SYSCFG->EXTICR[0] |= SYSCFG\_EXTICR1\_EXTI2\_PA; //Select PA2 as interrupt source

EXTI->RTSR1 |= EXTI\_RTSR1\_RT1; // Rising edge triggered for EXTI1

EXTI->RTSR1 |= EXTI\_RTSR1\_RT2; // Rising edge triggered for EXTI2

EXTI->PR1 = EXTI\_PR1\_PIF1; // Clear EXTI1 pending bit

EXTI->PR1 = EXTI\_PR1\_PIF2; // Clear EXTI2 pending bit

EXTI->IMR1 |= EXTI\_IMR1\_IM1; // Enable EXTI1

EXTI->IMR1 |= EXTI\_IMR1\_IM2; // Enable EXTI2

NVIC\_ClearPendingIRQ(EXTI1\_IRQn); // Clear NVIC pending bit

NVIC\_ClearPendingIRQ(EXTI2\_IRQn); // Clear NVIC pending bit

NVIC\_EnableIRQ(EXTI1\_IRQn); // Enables IRQ

NVIC\_EnableIRQ(EXTI2\_IRQn); // Enables IRQ

\*/

/\*\* I think these would do the same thing but give actual numbers

======================================================================= \*/

EXTI->RTSR1 |= 0x0006; // line2-1 are rising edge triggered

EXTI->IMR1 |= 0x0006; //line 2-1 are not masked; therefore they're enabled

EXTI->PR1 |= 0x0006; //clear pending for EXTI2-1

EXTI->PR1 |= 0x0006; //clear pending for PA2-1

NVIC\_EnableIRQ(EXTI1\_IRQn);

NVIC\_EnableIRQ(EXTI2\_IRQn);

SYSCFG->EXTICR[0] &= 0xF00F;

\_\_enable\_irq();

}

/\*----------------------------------------------------------\*/

/\* Delay function - do nothing for about 1 seconds \*/

/\*----------------------------------------------------------\*/

// Delay is exactly 0.5s

static void delay() {

int i,j,n;

for (i=0; i<20; i++) { //outer loop

for (j=0; j<8225; j++) { //inner loop

n = j; //dummy operation for single-step test

} //do nothing

}

}

static void count() {

if (run == 1) {

counter2 = (counter2 < 9) ? (counter2 + 1) : 0; //Infinite counter, Keeps from going over 10

if (up == 1) {

counter = (counter < 9) ? (counter + 1) : 0;

}

else {

counter = (counter > 0) ? (counter - 1) : 9;

}

// Clear, and grab the bit for LED's in PA12-5

GPIOA->ODR &= 0x0000; //clear the bits

GPIOA ->ODR |= counter << 5;

GPIOA ->ODR |= counter2 << 9;

}

}

//Interrupt called when PA1 is pressed

void EXTI1\_IRQHandler() {

\_\_disable\_irq();

if (run == 0) {

run = 1;

}

else {

run = 0;

}

GPIOB->ODR ^= 0x0008; //Set PB3=1 to turn on blue LED (in BSRR lower half)

EXTI->PR1 |= 0x0002;

NVIC\_ClearPendingIRQ(EXTI1\_IRQn);

\_\_enable\_irq();

}

//Interrupt called when PA2 is pressed

void EXTI2\_IRQHandler() {

\_\_disable\_irq();

if (up == 1) {

up = 0;

}

else {

up = 1;

}

GPIOB->ODR ^= 0x0010; //Set PB4 = 1 to turn ON green LED (in BSRR lower half)

EXTI->PR1 |= 0x0004;

NVIC\_ClearPendingIRQ(EXTI2\_IRQn);

\_\_enable\_irq();

}

int main() {

PinSetup();

interrupt\_setup();

counter = 0;

counter2 = 0;

run = 0;

up = 0;

GPIOA->ODR &= 0x0000;

while(1) {

count();

delay(); // Delay for half a second

}

}