(Interrupts)

(Curtis Robinson, Khondakar Mujtaba)

(ELEN 120)

(Wednesday 2:15)

Objective: Throughout this lab, our goal is to analyze and build a program that uses interrupts and a program that uses a timer. This task requires initializing all of the control registers for an interrupt and writing the interrupt service routine for the first 2 sections. This also requires configuring the time hardware and writing the interrupt service routine for the third section.

Procedure:

Problem 1:

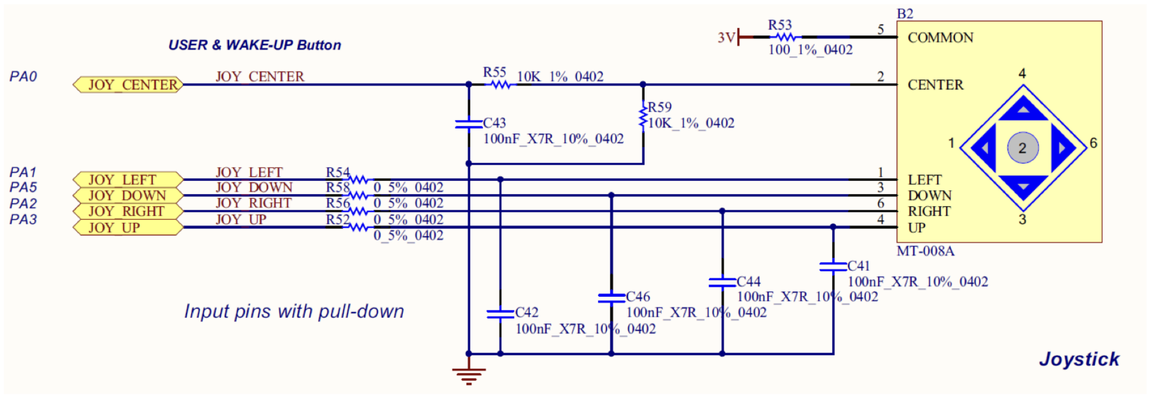
First, we need to create a new project and add it to the ExtInt Files. Then, fill in the enable commands and other necessary GPIO and Interrupt instructions to enable the green LED when the interrupt EXT0(center button) toggles.

Problem 2:

Now add the code that toggles the green LED when the EXT5 (down button) is toggled. (This one is more tricky because pins 5-9 share the same interrupt vector and some of the bits are in different registers).

Problem 3:

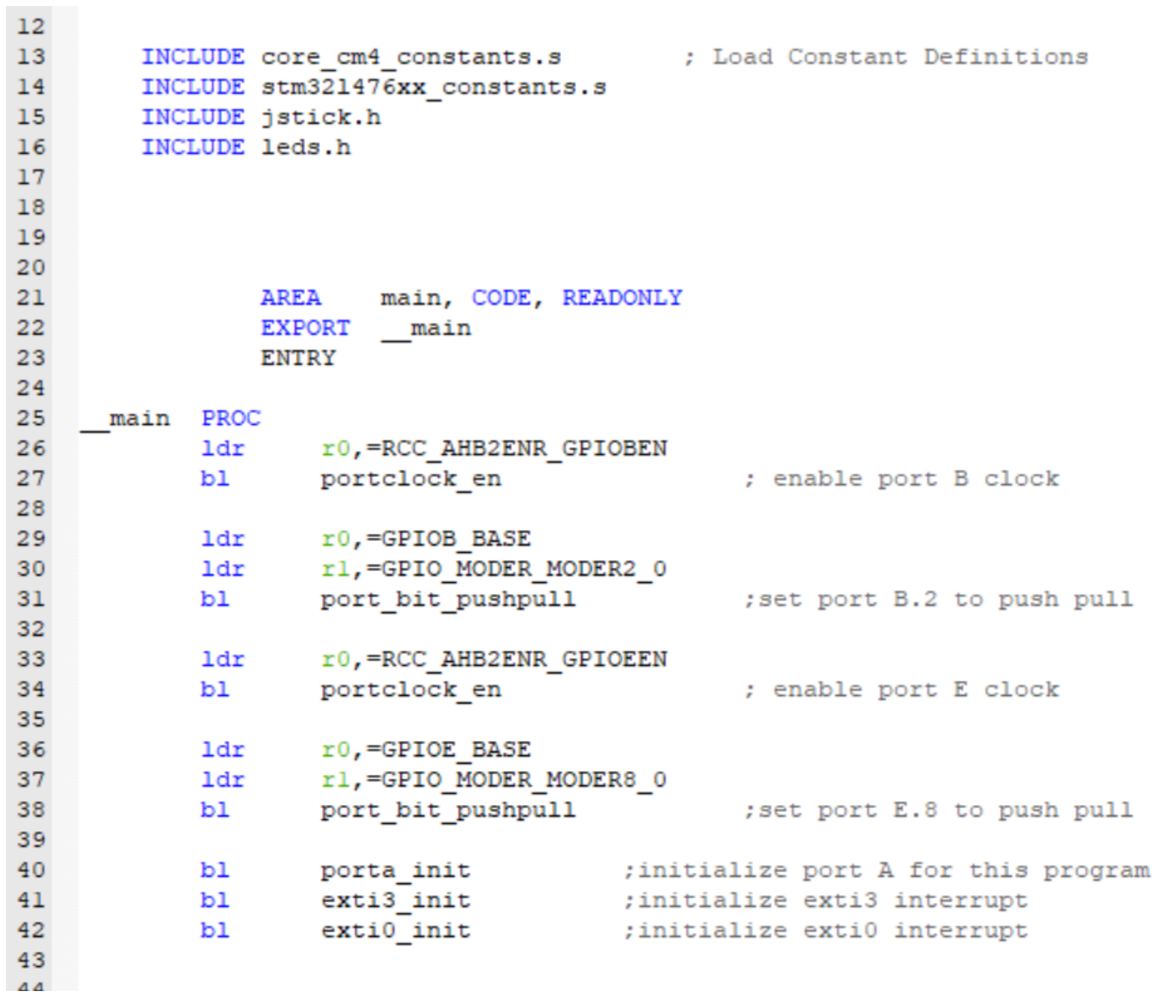
Now we want to manipulate the code so that a 1kHz square wave is displayed on Port B Pin 2. Connect an oscilloscope and measure the output signal.



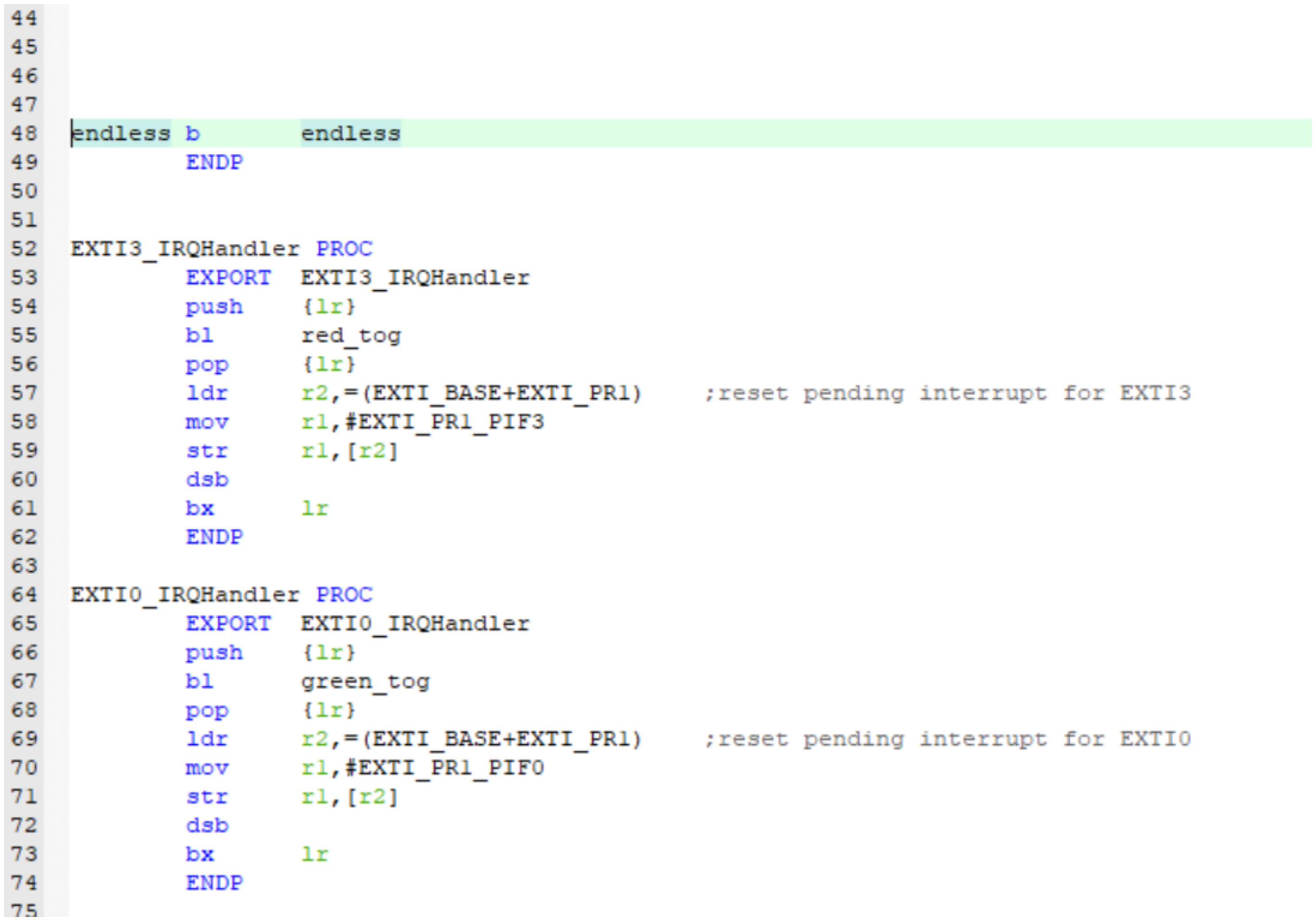
Problem 1:

Now, create a new project and add in the ExtInt files. Add additional code to this project so that the green LED toggles when interrupt EXT0 (the center button) is pressed.

Main.s:

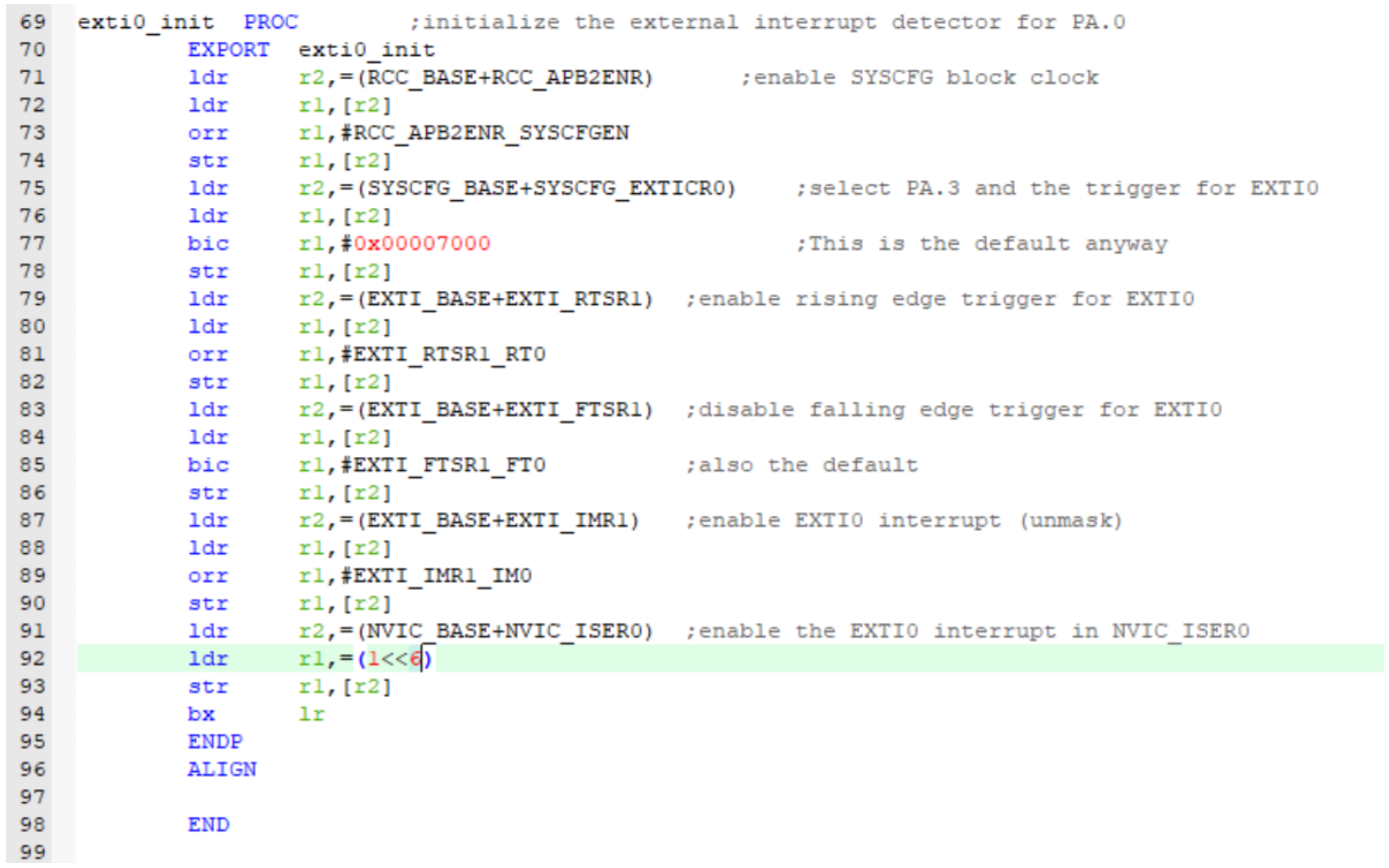


```

    INCLUDE core_cm4_constants.s        ; Load Constant Definitions
    INCLUDE stm32l476xx_constants.s
    INCLUDE jstick.h
    INCLUDE leds.h




            AREA    main, CODE, READONLY
            EXPORT  __main
            ENTRY

__main  PROC
        ldr     r0,=RCC_AHB2ENR_GPIOBEN
        bl      portclock_en                ; enable port B clock

        ldr     r0,=GPIOB_BASE
        ldr     r1,=GPIO_MODER_MODER2_0
        bl      port_bit_pushpull           ;set port B.2 to push pull

        ldr     r0,=RCC_AHB2ENR_GPIOEEN
        bl      portclock_en                ; enable port E clock

        ldr     r0,=GPIOE_BASE
        ldr     r1,=GPIO_MODER_MODER8_0
        bl      port_bit_pushpull           ;set port E.8 to push pull

        bl      porta_init          ;initialize port A for this program
        bl      exti3_init          ;initialize exti3 interrupt
        bl      exti0_init          ;initialize exti0 interrupt

```



```
endless b       endless
        ENDP


EXTI3_IRQHandler PROC
        EXPORT  EXTI3_IRQHandler
        push    {lr}
        bl      red_tog
        pop     {lr}
        ldr     r2,=(EXTI_BASE+EXTI_PR1)     ;reset pending interrupt for EXTI3
        mov     r1,#EXTI_PR1_PIF3
        str     r1,[r2]
        dsb
        bx      lr
        ENDP

EXTI0_IRQHandler PROC
        EXPORT  EXTI0_IRQHandler
        push    {lr}
        bl      green_tog
        pop     {lr}
        ldr     r2,=(EXTI_BASE+EXTI_PR1)     ;reset pending interrupt for EXTI0
        mov     r1,#EXTI_PR1_PIF0
        str     r1,[r2]
        dsb
        bx      lr
        ENDP
```

Jstick.s New code:

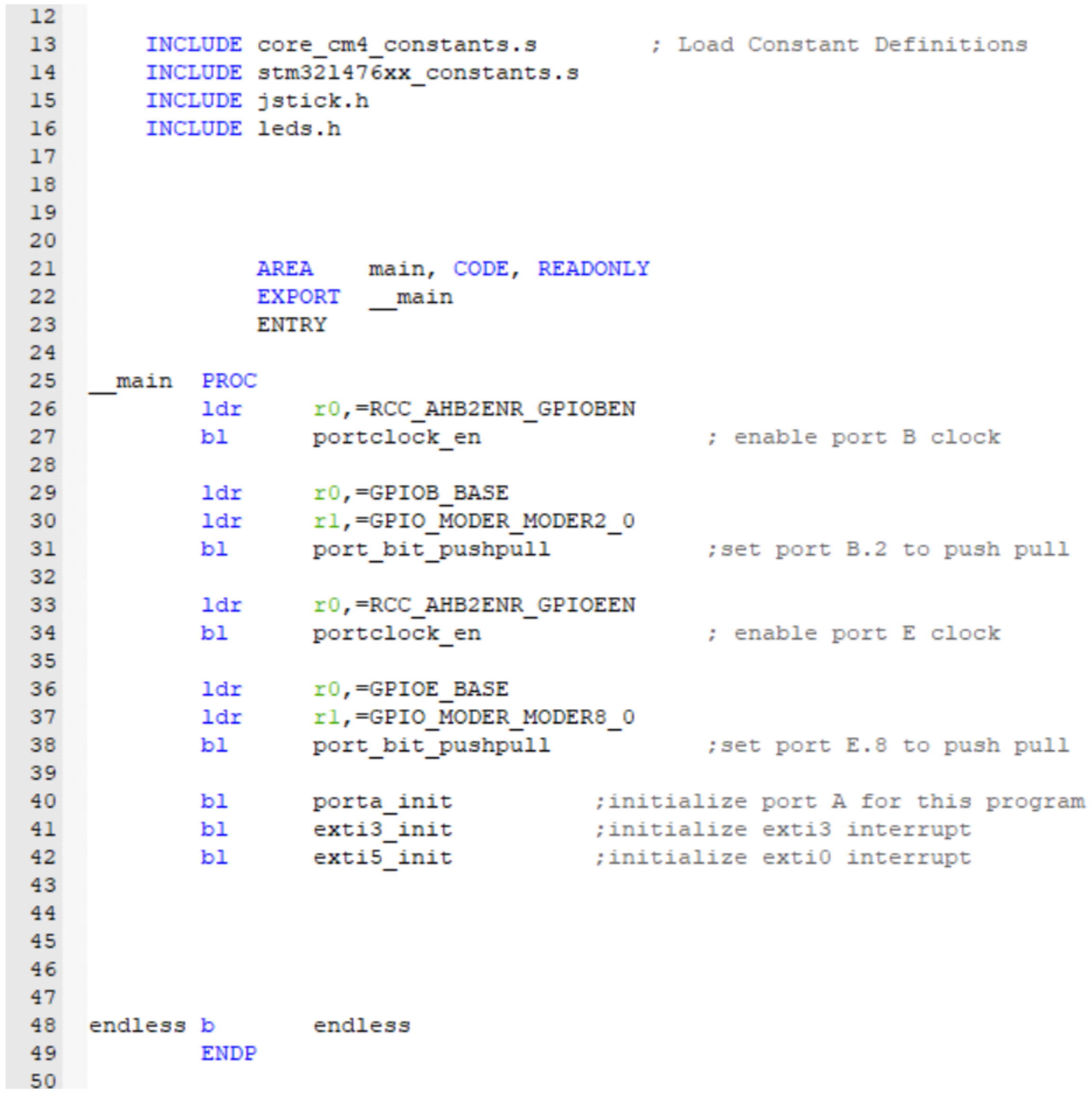


```
exti0_init  PROC        ;initialize the external interrupt detector for PA.0
        EXPORT  exti0_init
        ldr     r2,=(RCC_BASE+RCC_APB2ENR)      ;enable SYSCFG block clock
        ldr     r1,[r2]
        orr     r1,#RCC_APB2ENR_SYSCFGEN
        str     r1,[r2]
        ldr     r2,=(SYSCFG_BASE+SYSCFG_EXTICR0)    ;select PA.3 and the trigger for EXTI0
        ldr     r1,[r2]
        bic     r1,#0x00007000                      ;This is the default anyway
        str     r1,[r2]
        ldr     r2,=(EXTI_BASE+EXTI_RTSR1)  ;enable rising edge trigger for EXTI0
        ldr     r1,[r2]
        orr     r1,#EXTI_RTSR1_RT0
        str     r1,[r2]
        ldr     r2,=(EXTI_BASE+EXTI_FTSR1)  ;disable falling edge trigger for EXTI0
        ldr     r1,[r2]
        bic     r1,#EXTI_FTSR1_FT0          ;also the default
        str     r1,[r2]
        ldr     r2,=(EXTI_BASE+EXTI_IMR1)   ;enable EXTI0 interrupt (unmask)
        ldr     r1,[r2]
        orr     r1,#EXTI_IMR1_IM0
        str     r1,[r2]
        ldr     r2,=(NVIC_BASE+NVIC_ISER0)  ;enable the EXTI0 interrupt in NVIC_ISER0
        ldr     r1,=(1<<6)
        str     r1,[r2]
        bx      lr
        ENDP
        ALIGN

        END

```

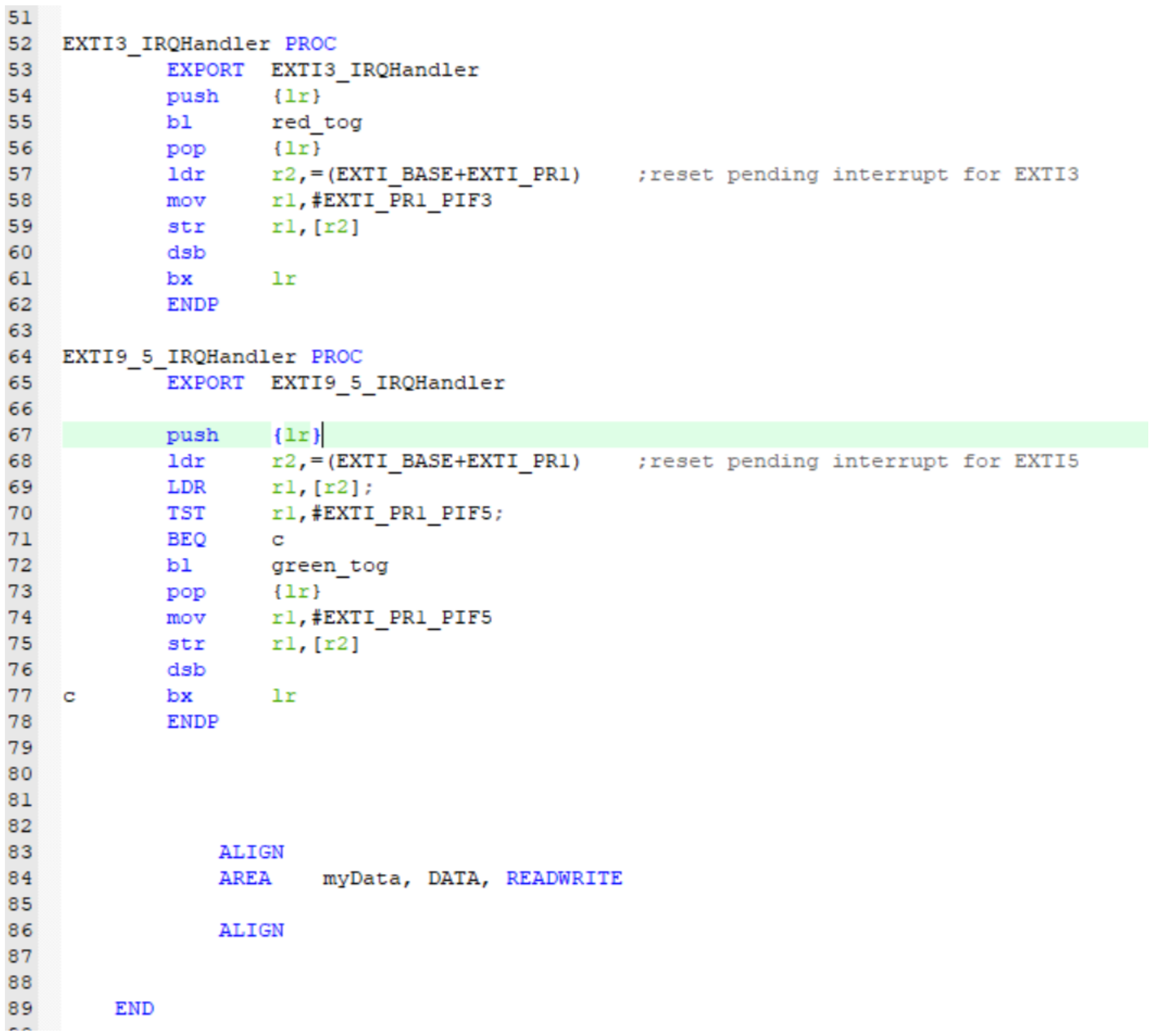
Problem 2:

Now, Copy your last project and modify it so that the green LED toggles when interrupt EXT5 (the down button) is pressed *instead* of the center button. This is a little bit harder since pins 5-9 share one interrupt vector and some of the bits are in different registers. When the interrupt is triggered, you need to make sure it came from pin 5 and not pins 6-9.

Main.s:

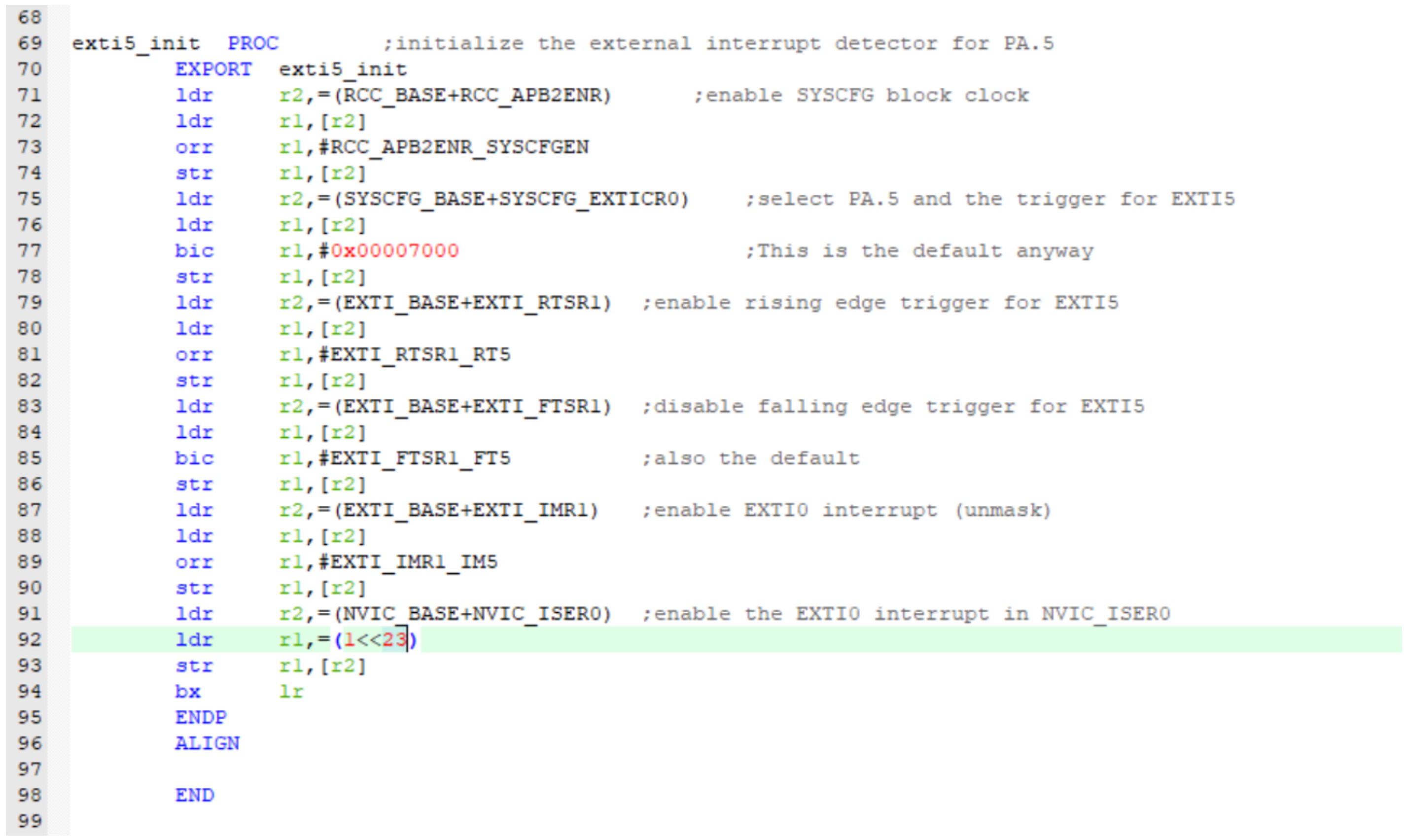


```

    INCLUDE core_cm4_constants.s        ; Load Constant Definitions
    INCLUDE stm32l476xx_constants.s
    INCLUDE jstick.h
    INCLUDE leds.h




            AREA    main, CODE, READONLY
            EXPORT  __main
            ENTRY

__main  PROC
        ldr     r0,=RCC_AHB2ENR_GPIOBEN
        bl      portclock_en                ; enable port B clock

        ldr     r0,=GPIOB_BASE
        ldr     r1,=GPIO_MODER_MODER2_0
        bl      port_bit_pushpull           ;set port B.2 to push pull

        ldr     r0,=RCC_AHB2ENR_GPIOEEN
        bl      portclock_en                ; enable port E clock

        ldr     r0,=GPIOE_BASE
        ldr     r1,=GPIO_MODER_MODER8_0
        bl      port_bit_pushpull           ;set port E.8 to push pull

        bl      porta_init          ;initialize port A for this program
        bl      exti3_init          ;initialize exti3 interrupt
        bl      exti5_init          ;initialize exti0 interrupt





endless b       endless
        ENDP

```



```

EXTI3_IRQHandler PROC
        EXPORT  EXTI3_IRQHandler
        push    {lr}
        bl      red_tog
        pop     {lr}
        ldr     r2,=(EXTI_BASE+EXTI_PR1)    ;reset pending interrupt for EXTI3
        mov     r1,#EXTI_PR1_PIF3
        str     r1,[r2]
        dsb
        bx      lr
        ENDP

EXTI9_5_IRQHandler PROC
        EXPORT  EXTI9_5_IRQHandler

        push    {lr}
        ldr     r2,=(EXTI_BASE+EXTI_PR1)    ;reset pending interrupt for EXTI5
        LDR     r1,[r2];
        TST     r1,#EXTI_PR1_PIF5;
        BEQ     c
        bl      green_tog
        pop     {lr}
        mov     r1,#EXTI_PR1_PIF5
        str     r1,[r2]
        dsb
c       bx      lr
        ENDP




            ALIGN
            AREA    myData, DATA, READWRITE

            ALIGN


    END
```

Jstick.s new code:



```
exti5_init  PROC        ;initialize the external interrupt detector for PA.5
        EXPORT  exti5_init
        ldr     r2,=(RCC_BASE+RCC_APB2ENR)      ;enable SYSCFG block clock
        ldr     r1,[r2]
        orr     r1,#RCC_APB2ENR_SYSCFGEN
        str     r1,[r2]
        ldr     r2,=(SYSCFG_BASE+SYSCFG_EXTICR0)    ;select PA.5 and the trigger for EXTI5
        ldr     r1,[r2]
        bic     r1,#0x00007000                      ;This is the default anyway
        str     r1,[r2]
        ldr     r2,=(EXTI_BASE+EXTI_RTSR1)  ;enable rising edge trigger for EXTI5
        ldr     r1,[r2]
        orr     r1,#EXTI_RTSR1_RT5
        str     r1,[r2]
        ldr     r2,=(EXTI_BASE+EXTI_FTSR1)  ;disable falling edge trigger for EXTI5
        ldr     r1,[r2]
        bic     r1,#EXTI_FTSR1_FT5          ;also the default
        str     r1,[r2]
        ldr     r2,=(EXTI_BASE+EXTI_IMR1)   ;enable EXTI0 interrupt (unmask)
        ldr     r1,[r2]
        orr     r1,#EXTI_IMR1_IM5
        str     r1,[r2]
        ldr     r2,=(NVIC_BASE+NVIC_ISER0)  ;enable the EXTI0 interrupt in NVIC_ISER0
        ldr     r1,=(1<<23)
        str     r1,[r2]
        bx      lr
        ENDP
        ALIGN

        END
```

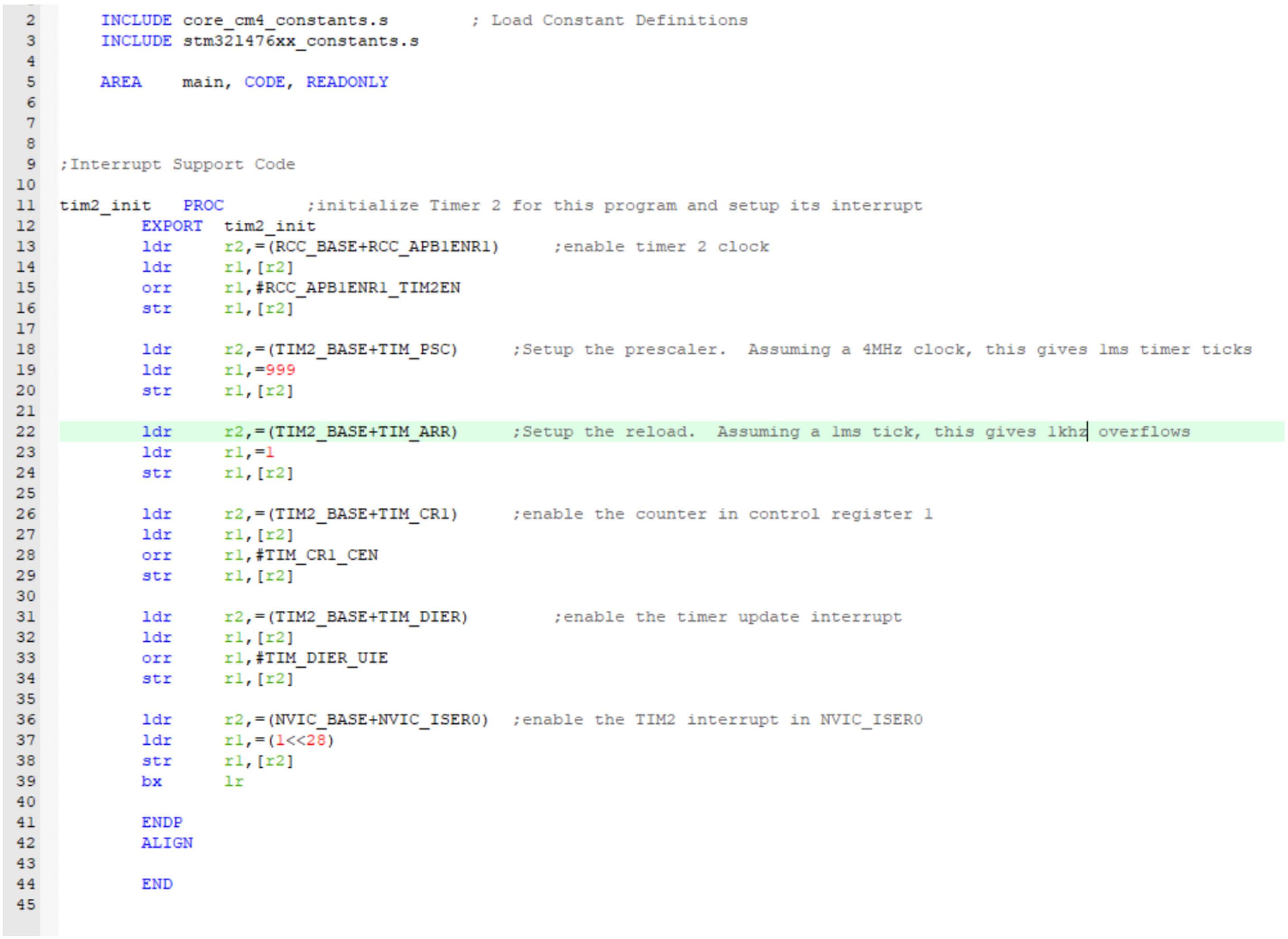
In order to make sure that we are only receiving an input from Pin 5 and not 6-9, we create a mask to clear any of the incoming bits that come from the pin 5-9, effectively leaving only the input of pin 5 alone. Then the program can act on the input of pin 5.

Problem 3:

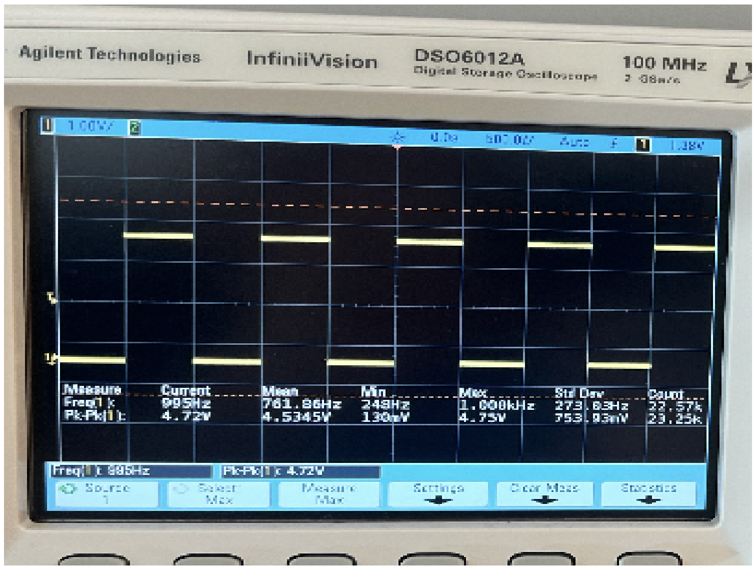
Copy and run my project called "timer". Describe what it does in your lab report. Now modify it so that it generates a 1KHz square wave on the Port B Pin 2 output.

Now –connect the output to an oscilloscope to view the output signal. Connect that pin to the oscilloscope and demo the square wave. (you need to connect the scope to the ground as well

This code makes the red led blink once a second. It also generates a square wave out of Pin 2.



```
    INCLUDE core_cm4_constants.s        ; Load Constant Definitions
    INCLUDE stm321476xx_constants.s

    AREA    main, CODE, READONLY



;Interrupt Support Code

tim2_init   PROC        ;initialize Timer 2 for this program and setup its interrupt
        EXPORT  tim2_init
        ldr     r2,=(RCC_BASE+RCC_APB1ENR1)     ;enable timer 2 clock
        ldr     r1,[r2]
        orr     r1,#RCC_APB1ENR1_TIM2EN
        str     r1,[r2]

        ldr     r2,=(TIM2_BASE+TIM_PSC)     ;Setup the prescaler.  Assuming a 4MHz clock, this gives 1ms timer ticks
        ldr     r1,=999
        str     r1,[r2]

        ldr     r2,=(TIM2_BASE+TIM_ARR)     ;Setup the reload.  Assuming a 1ms tick, this gives 1khz overflows
        ldr     r1,=1
        str     r1,[r2]

        ldr     r2,=(TIM2_BASE+TIM_CR1)     ;enable the counter in control register 1
        ldr     r1,[r2]
        orr     r1,#TIM_CR1_CEN
        str     r1,[r2]

        ldr     r2,=(TIM2_BASE+TIM_DIER)        ;enable the timer update interrupt
        ldr     r1,[r2]
        orr     r1,#TIM_DIER_UIE
        str     r1,[r2]

        ldr     r2,=(NVIC_BASE+NVIC_ISER0)  ;enable the TIM2 interrupt in NVIC_ISER0
        ldr     r1,=(1<<28)
        str     r1,[r2]
        bx      lr

        ENDP
        ALIGN

        END
```



```
Measure    Current   Mean        Min      Max        Std Dev     Count
Freq(1):   995Hz     761.86Hz    248Hz    1.000kHz   273.83Hz    22.57k
Pk-Pk(1):  4.72V     4.5345V     130mV    4.75V      753.93mV    23.25k

Freq(1): 995Hz     Pk-Pk(1): 4.72V
```