SANTA CLARA UNIVERSITY
Electrical and Computer Engineering Department

ELEN 120 – Embedded Computing Systems

Lab 5 – Interrupts and Timers

Sal Martinez and Dylan Thornburg

Andrew Wolfe

Assignment: In this assignment, you will analyze and build a program that uses interrupts and a program that uses a timer. This requires initializing all of the control registers for an interrupt and writing the interrupt service routine for the first 2 projects and configuring the timer hardware and writing the interrupt service routine for the third. Luckily, I've already done all of that for you. All you need to do is understand it and do it again.

Learning Objective: Learn how to detect and respond to internal and external interrupts and to program the internal timer modules.

Lab Procedure:

Before problem 1 – make a copy of ExtInt. Compile it and run it on the Discovery board. Each time to press the "up" joystick, it should toggle the red LED. If you download it and run it without the debugger, you need to press the black reset button before it will work. Review this code with the debugger, if necessary, to understand it. If you determine that your prelab answers were wrong – provide corrected answers in your lab report.

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.

Demo to the TA. Turn in your source code.

Main:

```
        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_GPIOEEN
                bl              portclock_en                    ; enable port E clock
                ldr             r0,=RCC_AHB2ENR_GPIOBEN
                bl              portclock_en                    ; enable port B clock


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

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

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



endless  b              endless
                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

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



                ALIGN
                AREA    myData, DATA, READWRITE

                ALIGN
```

```
        END
```

Jstick.s:

```
        INCLUDE core_cm4_constants.s        ; Load Constant Definitions
        INCLUDE stm32l476xx_constants.s

        AREA    main, CODE, READONLY

porta_init      PROC        ;Initialize port A for the joystick demo
                EXPORT      porta_init
                ldr         r2,=(RCC_BASE+RCC_AHB2ENR)          ;Turn on port A clock (bit
0)
                ldr         r1,[r2]
                orr         r1,#RCC_AHB2ENR_GPIOAEN
                str         r1,[r2]
                ldr         r2,=(GPIOA_BASE+GPIO_MODER)         ;clear bits 0-7 and 10-11 in
GPIOA_MODER
                ldr         r1,[r2]
                ldr         r0,=0x000000cff
                bic         r1,r0
                str         r1,[r2]
                ldr         r2,=(GPIOA_BASE+GPIO_PUPDR)         ;set field 0-3 and 5 in
GPIOA_PUPR to 10
                ldr         r1,[r2]
                ldr         r0,=0x000000455
                bic         r1,r0
                lsl         r0,#1
                orr         r1,r0
                str         r1,[r2]
                bx          lr
                ENDP

read_jstick     PROC
                EXPORT read_jstick
                ldr         r2,=(GPIOA_BASE+GPIO_IDR)
                ldr         r0,[r2]
                and         r0,#0x0000002f
                bx          lr
                ENDP


;Interrupt Support Code

exti0_init      PROC        ;initialize the external interrupt detector for PA.3
                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,#0x00000007                                  ;This is the
default anyway
                str         r1,[r2]
                ldr         r2,=(EXTI_BASE+EXTI_RTSR1)  ;enable rising edge trigger for EXTI3

                ldr         r1,[r2]
                orr         r1,#EXTI_RTSR1_RT0
                str         r1,[r2]
                ldr         r2,=(EXTI_BASE+EXTI_FTSR1)  ;disable falling edge trigger for EXTI3

                ldr         r1,[r2]
                bic         r1,#EXTI_FTSR1_FT0              ;also the default
                str         r1,[r2]
                ldr         r2,=(EXTI_BASE+EXTI_IMR1)   ;enable EXTI3 interrupt (unmask)
                ldr         r1,[r2]
                orr         r1,#EXTI_IMR1_IM0
                str         r1,[r2]
                ldr         r2,=(NVIC_BASE+NVIC_ISER0)  ;enable the EXTI3 interrupt in NVIC_ISER0
                ldr         r1,=(1<<6)
                str         r1,[r2]
                bx          lr
                ENDP

exti3_init      PROC        ;initialize the external interrupt detector for PA.3
                EXPORT      exti3_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 EXTI3
                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 EXTI3

                ldr         r1,[r2]
                orr         r1,#EXTI_RTSR1_RT3
                str         r1,[r2]
                ldr         r2,=(EXTI_BASE+EXTI_FTSR1)  ;disable falling edge trigger for EXTI3

                ldr         r1,[r2]
                bic         r1,#EXTI_FTSR1_FT3              ;also the default
                str         r1,[r2]
                ldr         r2,=(EXTI_BASE+EXTI_IMR1)   ;enable EXTI3 interrupt (unmask)
                ldr         r1,[r2]
                orr         r1,#EXTI_IMR1_IM3
                str         r1,[r2]
                ldr         r2,=(NVIC_BASE+NVIC_ISER0)  ;enable the EXTI3 interrupt in NVIC_ISER0
                ldr         r1,=(1<<9)
                str         r1,[r2]
                bx          lr
                ENDP
                ALIGN
```

```
END
```

Joystick.h:

```
IMPORT  porta_init
IMPORT          read_jstick
IMPORT          exti3_init
IMPORT          exti0_init

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.**

Demo to the TA. Turn in your source code. Explain in your report how you make sure the interrupt came from pin 5 and not pins 6-9.

Explanation: We made a conditional branch to avoid pins 6-9. If it had pin 5 (0x20), it skipped the instructions:

```
push  {lr}
bl              green_tog
pop             {lr}
mov             r1,#EXTI_PR1_PIF5
```

And just continued to store the value in r2.

Main:

```
;******************** (C) Andrew Wolfe ********************************************
; @file    main_hw_proto.s
; @author  Andrew Wolfe
; @date    August 18, 2019
; @note
;         This code is for the book "Embedded Systems with ARM Cortex-M
;         Microcontrollers in Assembly Language and C, Yifeng Zhu,
;         ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara University
;**********************************************************************************


        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_GPIOEEN
                bl          portclock_en                    ; enable port E clock


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

                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

                bl          porta_init                      ;initialize port A for this program
                bl          exti5_init                      ;initialize exti5 interrupt
                bl          exti3_init                      ;initialize exti3 interrupt



endless  b              endless
                ENDP


EXTI9_5_IRQHandler PROC
                EXPORT      EXTI9_5_IRQHandler
                ldr         r2, =(EXTI_BASE+EXTI_PR1)
                ldr r3, [r2]
                tst     r3, #0x20
                beq cond
                push    {lr}
                bl          green_tog
                pop         {lr}
                mov         r1,#EXTI_PR1_PIF5
cond    str             r1,[r2]
                dsb
                bx          lr
                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
```

```
                ALIGN
                AREA    myData, DATA, READWRITE

                ALIGN


        END
```

Jstick.s:

```
        INCLUDE core_cm4_constants.s        ; Load Constant Definitions
        INCLUDE stm32l476xx_constants.s

        AREA    main, CODE, READONLY

porta_init      PROC        ;Initialize port A for the joystick demo
                EXPORT      porta_init
                ldr         r2,=(RCC_BASE+RCC_AHB2ENR)          ;Turn on port A
clock (bit 0)
                ldr         r1,[r2]
                orr         r1,#RCC_AHB2ENR_GPIOAEN
                str         r1,[r2]
                ldr         r2,=(GPIOA_BASE+GPIO_MODER)         ;clear bits 0-7 and
10-11 in GPIOA_MODER
                ldr         r1,[r2]
                ldr         r0,=0x000000cff
                bic         r1,r0
                str         r1,[r2]
                ldr         r2,=(GPIOA_BASE+GPIO_PUPDR)         ;set field 0-3 and 5
in GPIOA_PUPR to 10
                ldr         r1,[r2]
                ldr         r0,=0x000000455
                bic         r1,r0
                lsl         r0,#1
                orr         r1,r0
                str         r1,[r2]
                bx          lr
                ENDP

read_jstick     PROC
                EXPORT read_jstick
                ldr         r2,=(GPIOA_BASE+GPIO_IDR)
                ldr         r0,[r2]
                and         r0,#0x0000002f
                bx          lr
                ENDP


;Interrupt Support 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_EXTICR1)    ;select PA.5 and the
trigger for EXTI5
                ldr     r1,[r2]
                bic     r1,#0x00000070
                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 EXTI5 interrupt (unmask)
                ldr     r1,[r2]
                orr     r1,#EXTI_IMR1_IM5
                str     r1,[r2]
                ldr     r2,=(NVIC_BASE+NVIC_ISER0)  ;enable the EXTI5 interrupt in
NVIC_ISER0
                ldr     r1,=(1<<23)
                str     r1,[r2]
                bx      lr
                ENDP

exti3_init      PROC    ;initialize the external interrupt detector for PA.3
                EXPORT  exti3_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 EXTI3
                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 EXTI3

                ldr     r1,[r2]
                orr     r1,#EXTI_RTSR1_RT3
                str     r1,[r2]
                ldr     r2,=(EXTI_BASE+EXTI_FTSR1)  ;disable falling edge trigger for EXTI3

                ldr     r1,[r2]
                bic     r1,#EXTI_FTSR1_FT3              ;also the default
                str     r1,[r2]
                ldr     r2,=(EXTI_BASE+EXTI_IMR1)   ;enable EXTI3 interrupt (unmask)
                ldr     r1,[r2]
                orr     r1,#EXTI_IMR1_IM3
```

```
                str             r1,[r2]
                ldr             r2,=(NVIC_BASE+NVIC_ISER0)  ;enable the EXTI3 interrupt in
NVIC_ISER0
                ldr             r1,=(1<<9)
                str             r1,[r2]
                bx              lr
                ENDP
                ALIGN

                END
```

Jstick.h:

```
                IMPORT  porta_init
                IMPORT          read_jstick
                IMPORT          exti3_init
                IMPORT          exti5_init

                END
```

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 ground as well.)

Demo to the TA. Take a picture of the scope. Turn in your source code and the picture.

Timer is a program that turns a light on for 1 second and turns it off for one second before turning it back on. It has a period of 2 seconds and frequency of .5hz (.495hz exact on the oscilloscope). Our goal is to achieve a 1 KHz wave. We did this by using a much smaller prescale and reloading value which was calculated from this equation:

c/(2xy)=z
c is clock speed of processor (in our case 4000000hz or 4 mHz)
Z is the goal clock speed (for our assignment it was 1000hz or 1Khz)
X and y are the prescalar and reload values but for the sake of this equation can really be whatever so long as the equation works out. To be safe, just never have x and y be zero.
The two values we chose were 999 (really 1000) and 1 (really 2).

This gave us a 995 Hz wave. Due to error propagation, we assumed that the value was close enough to be considered 1 KHz.

CODE (only timer was modified):

```
        INCLUDE core_cm4_constants.s        ; Load Constant Definitions
        INCLUDE stm32l476xx_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 1s 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
```