Lab 6 Report

Dallin Peterson

A02366595

Caleb Hamblin

A02420019

Objectives

The purpose of this lab is to gain experience using the SysTick Timer and interrupts**.**

Procedure

We prepared for the lab by designing a schematic for the GPIO pins we would use to connect to the LCD and the three push buttons. We then created pseudocode to initialize the SysTick Timer, NVIC register, and EXTI interrupts which were connected to three push buttons. After, we made the pseudocode that described the behavior of the LCD timer when the buttons were pressed. The green button enabled the timer counting, the yellow button stopped the timer counting, and the red button reset the timer to 00:00.0.

To configure the SysTick timer, we connected SysTick to the 16MHz HSI clock and then we disabled the SysTick IRQ. We then changed the SysTick control clock source bit to '1' to be connected to the processor clock. We set the reload value register to 1600000-1 which we hypothesized would provide a 0.1 second SysTick clock period. After, we reset the SysTick current-value register (SYSTICK_VAL) to '0,' and the priority to the highest value corresponding to the lowest priority. Finally, we set the TICKINT bit to high and SysTick IRQ to high to activate the SysTick timer.

To configure the EXTI to connect to the push-buttons, we connected the EXTIs to enable the clock and then connected EXTI 8, 9, and 10 to GPIOB pins 8, 9, and 10. We configured the triggers to detect a falling edge for a button press. Then, we disabled the interrupt mask in IMR1 for EXTI 8, 9, and 10 which enabled the interrupts. We set the priority of EXTI9_5 to '2,' and the EXTI15_10 priority to '1.' Finally, we enabled the EXTIs 8, 9, and 10 in the NVIC register.

In the SysTick handler, if the global integer "timerEnable" was 1, we incremented T which represented a tenth of a second. If T was 9, then we incremented S2, which represented the one's place of seconds, by 1 and set T to 0. This same logic was used hierarchically for S1, M2, and M1, so that the timer logic would increment appropriately for the ten's place of seconds, the one's place of minutes, and the ten's place of minutes. These integer values were placed in a global array called "time" in this order: {M1, M2, ':', S1, S2, '.', T}. We added 48 to M1, M2, S1, S2, and T to convert the numbers to the ASCII table form. Finally, we called the function

“LCD_DisplayString,” which handles printing a string on the LCD, with the “time” array and ‘0’ as the arguments.

There are two button-press interrupt handlers: one for EXTIs 9-5 and one for EXTIs 15-10. Since we had our green and yellow buttons connected to GPIOB pins 8 and 9 respectively, we handled the logic for those two pins inside the EXTI9_5_IRQHANDLER function. If the green button was pressed, timerEnable would be set to one. If the yellow button was pressed, timerEnable would be set to 0. We then cleared the EXTI interrupts 8 and 9. In EXTI15_10_IRQHandler, we handled the logic for the pressed red button, which set M1, M2, S1, S2, and T to zero. Then, we cleared the EXTI interrupt 10.

The GPIO configuration for pins 8, 9, 10 ten was input mode on the MODER, and pull up mode on the PUPDR. These instructions are inside the function “buttonGPIOInit” function. We kept the same configuration of GPIO pins for the LCD.

In main.c, we called all of the initialization functions and then entered into a “while(1)” infinite loop.

We experienced issues with the timing of the clock. The timer was incrementing too slow, so to solve this problem, we assumed that the system clock was 4 MHz instead of 16 MHz. If that was true, the SysTick reload value should be 399999 and not 1599999. We tested out this hypothesis and the timer lined up with a timer on our iPhones. We checked that the 16 MHz HIS clock was on and found that it was turned on. For that reason, we are not sure why the microcontroller behaved as though it had a 4 MHz clock. However, we found a solution to the timing problem.

Results

We were able to successfully display a timer on an LCD; the timer was controlled by three pushbuttons. These buttons called external interrupts to modify the timer behavior. We had some confusion regarding the system clock. We tried to set up the 16 MHz HSI clock, but when calculating the reload value for the SysTick handler we had to treat the clock like it was at 4 MHz. We also conducted a rough test of the accuracy of the timer. We started our timer and another stopwatch and compared the results after one minute. The stopwatch read 01:00.0 and our timer read 01:00.5, showing a 0.83% error.

Figures

Figure 1. Timer accuracy test.

Conclusion

In this lab, we figured out how to configure and alter the SysTick interrupt and external interrupts using C code. We also gained experience in troubleshooting delay issues and correctly entering/exiting interrupt handlers. We found that utilizing SysTick interrupts is an effective way to generate a consistent clock.

Appendix

Interrupts.c

```
#include "interrupts.h"

#include "stm32l476xx.h"

#include "LCD.h"

//#include "startup_stm321476xx.s"

//#include


#define pb8 (1<<8)

#define pb9 (1<<9)

#define pb10 (1<<10)


volatile int timerEnable = 0; //if 0, stop. if 1, go.
```

```
volatile int T; //tenths of a second
volatile int S1; //1s place of seconds
volatile int S2; //10s place of seconds
volatile int M1; //1s place of minutes
volatile int M2; //10s place of minutes
unsigned char time[7];

void SysTick_Init(uint32_t ticks) {
   SysTick->CTRL = 0; //disable systick irq and counter

   SysTick->LOAD = ticks - 1;

   NVIC_SetPriority (SysTick_IRQn, (1<<__NVIC_PRIO_BITS) - 1); //set sys tick to lowest priority

   //reset systick counter value
   SysTick ->VAL = 0;

   //select processor clock
   SysTick->CTRL |= SysTick_CTRL_CLKSOURCE_Msk; //systick control register bit 2 goes high, for processor clock

   SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk; //counter counts down to 0 and makes excpetoin

   SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk; //enable
}
```

```
void EXTI_Init(void) {
    //enables clock to sysconfig
    RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;



    //connect exti 8 9 10 to gpiob 8 9 10
    SYSCFG->EXTICR[2] &= ~SYSCFG_EXTICR3_EXTI8;
    SYSCFG->EXTICR[2] |= SYSCFG_EXTICR3_EXTI8_PB;
    SYSCFG->EXTICR[2] &= ~SYSCFG_EXTICR3_EXTI9;
    SYSCFG->EXTICR[2] |= SYSCFG_EXTICR3_EXTI9_PB;
    SYSCFG->EXTICR[2] &= ~SYSCFG_EXTICR3_EXTI10;
    SYSCFG->EXTICR[2] |= SYSCFG_EXTICR3_EXTI10_PB;

//disable rising edge on exti 8 9 10
    EXTI->RTSR1 &= ~EXTI_RTSR1_RT8;
    EXTI->RTSR1 &= ~EXTI_RTSR1_RT9;
    EXTI->RTSR1 &= ~EXTI_RTSR1_RT10;

       //set exti 8 9 10 to falling edge trigger
    EXTI->FTSR1 |= EXTI_FTSR1_FT8;
    EXTI->FTSR1 |= EXTI_FTSR1_FT9;
    EXTI->FTSR1 |= EXTI_FTSR1_FT10;

    //enable exti interrupts on 8 9 10
    EXTI->IMR1 |= EXTI_IMR1_IM8;
    EXTI->IMR1 |= EXTI_IMR1_IM9;
    EXTI->IMR1 |= EXTI_IMR1_IM10;
```

```
    //set exti 8 9 10 priority to 1
    NVIC_SetPriority(EXTI9_5_IRQn, 2);
    NVIC_SetPriority(EXTI15_10_IRQn, 1);

    //enable interrups exti9 to 5 and exti 15 - 10
    NVIC_EnableIRQ(EXTI9_5_IRQn);
    NVIC_EnableIRQ(EXTI15_10_IRQn);
}

void buttonGPIOInit(void) {
    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN;

    GPIOB->MODER &= 0xFFC0FFFF; //clear pins 8 9 10, input mode
    GPIOB->PUPDR &= 0xFFC0FFFF; //clear pins 8 9 10
    GPIOB->PUPDR |= 0x00150000; //set pins 8 9 10 to pull up mode.

}

void SysTick_Handler(void) {

    if(timerEnable) {
        if(T == 9) {
            T = 0;
            if(S2 == 9) {
                S2 = 0;
                if(S1 == 5) {
```

```
                S1 = 0;
                if(M2 == 9) {
                   M2 = 0;
                   if(M1 == 9) {
                      M1 = 0;
                   } else M1++;
                } else M2++;
             } else S1++;
          } else S2++;
       } else T++;

    } //end if timer enable

    time[0] = M1 + 48; //convert integer to character
    time[1] = M2 + 48;
    time[2] = ':';
    time[3] = S1 + 48;
    time[4] = S2 + 48;
    time[5] = '.';
    time[6] = T + 48;
    LCD_DisplayString(0, time);



}
void EXTI9_5_IRQHandler(void)
{
```

```
    //if green button pressed, activate timer count
    if(!((GPIOB->IDR & pb8) == pb8))
        timerEnable = 1;

    //if yellow button pressed, stop timer count
    if(!((GPIOB->IDR & pb9) == pb9))
        timerEnable = 0;
    EXTI->PR1 |= EXTI_PR1_PIF8;
    EXTI->PR1 |= EXTI_PR1_PIF9;
}
void EXTI15_10_IRQHandler(void) {
    //if red button pressed, reset timer
    M1 = 0;
    M2 = 0;
    S1 = 0;
    S2 = 0;
    T = 0;
    EXTI->PR1 |= EXTI_PR1_PIF10;
};
```