TA Andrew Gerber 9/12/2023

Lab 1 Report

Benjamin Meads
A02323437
Nick Templeton
A02268123

Objectives

Become familiar with the process of building, downloading, debugging, running, and modifying assembly programs on a microcontroller using an IDE. In this lab we ran, modified, and debugged our first assembly program on the STM32L476 microcontroller. Our goal was to turn our user programmable LED2 to flash on and off as fast as possible.

Procedure

We started the new Keil project and got it all ready and configured for our STM board, and prepared the IDE for its use in the rest of the procedure. We walked through the assembly code to make sure we understood what was happening, doing the relevant research as needed. We modified the code so the pin for the user controlled LED was oscillating between High and Low. We did this by creating a basic loop. Where after we set the registers and load them into memory. We loop back to the beginning to set them again.

At first we achieved a change rate of 250kHz change rate. Which oscillate so fast to where the LED looked dimmer than it was in its usual High state. We then removed our lines of code where we loaded the registers, and instead just changed and stored the value. We used the Logic analyzer to find how quickly the oscillation was occurring. We achieved a new high-score of 574.7kHz.

Results

Our final results included placing our modification assembly instructions inside a quick loop that just called itself and looped continuously. We used AND instructions to set the value to zero, as well as OR instructions to set the value to one. We first achieved a frequency of 412kHz. But after removing our load instruction, we brought that frequency up to 581.8kHz.

Figures

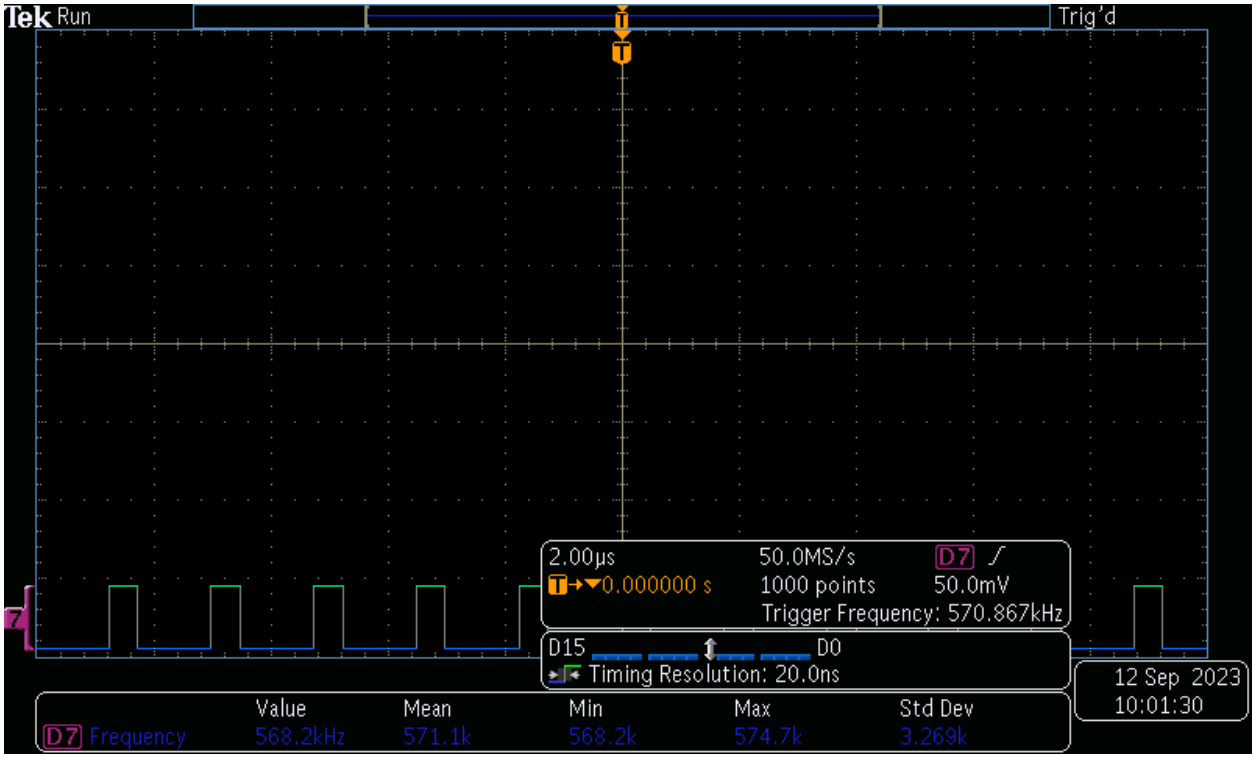

Figure 1: Oscilloscope showing a sample period of 200ns

Conclusion

This lab gave us a great intro into microcontrollers and allowed us to modify and implement our first assembly coded file onto our microcontroller. We were able to practice reading assembly and learning what exactly is being done within the processor. And finally, we were in the lead when we left with the highest frequency in the class with 574.7kHz.

```
;******************** (C) Yifeng ZHU ***************************************
; @file    main.s
; @author  Yifeng Zhu
; @date    May-17-2015
; @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
; @attension
;           This code is provided for education purpose. The author shall not be
;           held liable for any direct, indirect or consequential damages, for any
;           reason whatever. More information can be found from book website:
;           http:;www.eece.maine.edu/~zhu/book
;*****************************************************************************


    INCLUDE core_cm4_constants.s
    INCLUDE stm32l476xx_constants.s

    AREA    main, CODE, READONLY
    EXPORT  __main
    ENTRY

__main  PROC

    ; Enable the clock to GPIO Port A
    LDR r0, =RCC_BASE ;creates space for clock
    LDR r1, [r0, #RCC_AHB2ENR] ;enables the clock (read)
    ORR r1, r1, #RCC_AHB2ENR_GPIOAEN ;enables the clock (modify)
    STR r1, [r0, #RCC_AHB2ENR] ;enables the clock (write)

    ; MODE: 00: Input mode, 01: General purpose output mode
    ;       10: Alternate function mode, 11: Analog mode (reset state)

    ; Turn on the LED
    LDR r0, =GPIOA_BASE ;setting up memory space for GPIOA
    LDR r1, [r0, #GPIO_MODER] ;reading in GPIO Moder offset
    AND r1, r1, #0xFFFFF3FF ;setting PIN 5 to 00
    ORR r1, r1, #0x00000400 ;setting PIN 5 to 01
    STR r1, [r0, #GPIO_MODER] ;store the value in r1. Into memory location r0 + GPIO_MODR (offset)
L3
        ;LDR r1, [r0, #GPIO_ODR] ;Load r1 with the contends of Output data register     ;removed
        ORR r1, r1, #0x00000020 ;setting PIN 20 to HIGH
        STR r2, [r0, #GPIO_ODR];store contents of r1 into memory location r0 + GPIO_ODR
        ;////////////////////////////////////////////////////////////////////////////////////////////////////
        ;LDR r1, [r0, #GPIO_ODR] ;Load r1 with the contends of Output data register
        AND r1, r1, #0xFFFFFFDF ;setting PIN 20 to LOW
        STR r3, [r0, #GPIO_ODR];store contents of r1 into memory location r0 + GPIO_ODR
    B L3
    ;////////////////////////////////////////////////////////////////////////////////////////////////////////////
stop    B       stop

    ENDP

    ALIGN

    END
```