Objectives

- To execute the instructions of Lab 1 (ECE 3710 Lab 1.pdf) from the course wiki*.
- To become familiar with the process of building, downloading, debugging and running programs on the microcontroller using an IDE and verifying behavior using the logic analyzer:
    - Tektronix MDO3014 Mixed Domain Oscilloscope
    - Tiva C Series TM4C123GH6PM Microcontroller

Overview

"In this lab we will be running our first assembly program on the microcontroller and de-bugging another program that copies data in a couple ways. All assembly programs may be found on the course wiki*." (ECE 3710 Lab 1.pdf)

Preparation

- Come to lab with the microcontroller board and its USB cable.
- Be prepared to use most of the instructions that have been learned in class.
- Read V1.Ch2.3-7 and V2.Ch1.3 of course textbooks.
- Read through the two Keil uVision tutorials from Circuits Today posted on the course wiki*.

* https://spaces.usu.edu/display/ece3710/Home

Procedure

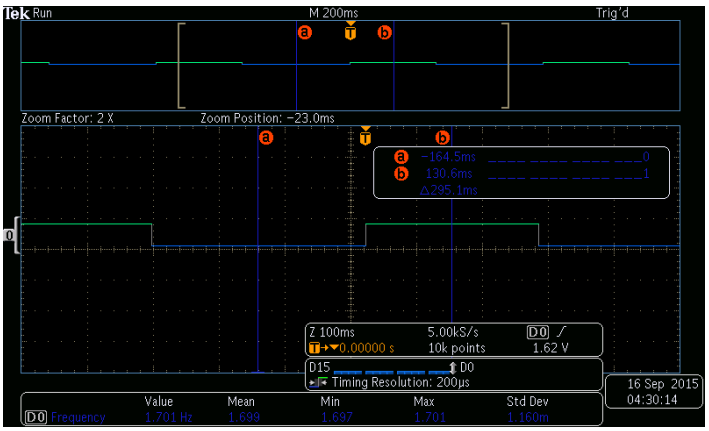
Running Blinky on the Microcontroller Board

Observe and note what is happening on the board.

We see that the light D1 alternates at a constant rate between the red light and the blue light.

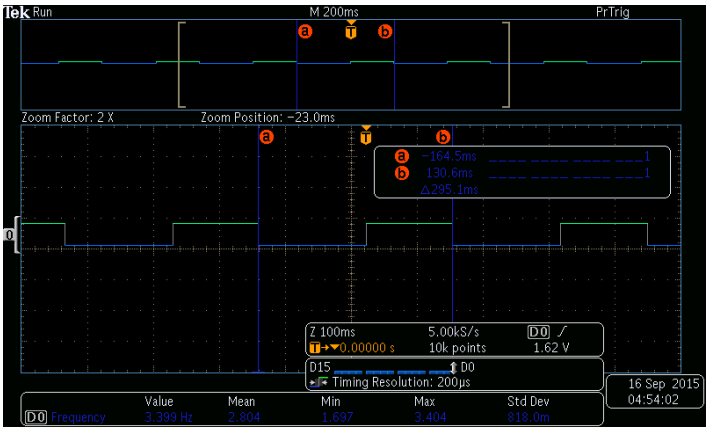
Use the logic analyzer to measure the frequency of the blinking and modify `Blinky.asm` so your board blinks twice as fast.

To increase the rate at which the light D1 alternates between red and blue, lines 41 and 52 of the `Blinky.asm` file were modified to increase the value of the data stored in R4 by 2 rather than 1 with each loop iteration.



BLINKY.asm Before Frequency Change

$$f = \frac{1}{T} \quad T = 587.89 \text{ ms} \quad f = 1.701 \text{ Hz}$$



BLINKY.asm After Frequency Change

$$f = \frac{1}{T} \quad T = 294.20 \text{ ms} \quad f = 3.399 \text{ Hz}$$

Commenting Code

BLINKY.asm

```
        THUMB
        AREA    |.text|, CODE, READONLY, ALIGN=2
        EXPORT  Start


GPIO_CLK        DCD 0x400FE018   ; address for GPIO Clock
CLKVAL          DCD 0x0020       ; value for enabling gpios clock
UNLOCK          DCD 0x4C4F434B   ; gpio unlock code.
GPIOF           DCD 0x40025000   ; base address for one of the GPIOs
GPIOF_PINS      DCD 0xE          ; value for configuring the gpio
                                 ; LIST the values toggled by the setup value.
                                 ; i.e. gpio enable, pull up register, etc.
                                 ; Table 10-1. GPIO Pins With Non-Zero Reset Values (pg. 648)
                                 ; 0xE = 1110
                                 ; GPIOAFSEL=1 GPIODEN=1 GPIOPDR=1 GPIOPUR=0
                ALIGN

Start
        mov32 R0, #0x400FE108 ; Enable GPIO Clock
        mov R1, #0x20
        str R1, [R0]

        mov32 R0, #0x40025000 ; GPIOF address

        ;unlock GPIOF
        mov32 R1, #0x4C4F434B ; GPIO Unlock code.
        str R1, [R0,#0x520]

        mov R1, #0x1F
        str R1, [R0,#0x524]   ; GPIOCR
        mov R1, #0x11
        str R1, [R0,#0x510]
        mov R1, #0x0F
        str R1, [R0,#0x400]   ; GPIODIR
        mov R1, #0x1F
        str R1, [R0,#0x51C]   ; digital enable
```

```
loop
        ;; GPIO Port F: GPIODATA (pg. 659 - MDS)
        ;; User Switches and RGB User LED (pg. 9 - BUG)
        ;; DATA register alternates between values 0x12 and 0x14
        ;; Bits 1 and 2 alternate between 0 and 1; i.e. RED and BLUE
        ;; 0x12 = RED, 0x14 = BLUE
        MOV32 R1, #0x02       ; load value for turning on LED color RED
        STR R1, [R0,#0x38]    ; write the above value to GPIOC ODR register

        MOV R4, #0            ;initial value for iteration loop
        MOV32 R5, #0xFFFFF    ;number of iterations for delay loops
delay1
        ADD R4, #1            ; increment by one which sets the default base clock speed
        CMP R4, R5            ;check number of iterations
        BLE delay1            ;continue if iterated less than 0xFFFFF + 1 times, otherwise repeat delay loop

        MOV32 R1, #0x04       ; load value for turning on LED color BLUE
        STR R1, [R0, #0x38]     ;write the above value to GPIOC ODR

        MOV R4, #0              ;initial value for iteration loop
        ;                       ; **** the tm4c123gh6pm has 16 MHz clock.
        ;;                       how long should the loop take with that clock?
        ;;                       one clock cycle/machine cycle
        ;;                       loop contains 15 lines of machine code
        ;;                       the period of 16 Mhz is 62.5 ns; i.e. time of one clock cycle
        ;;                       therefore 15 lines of machine code would take 937.5 ns (15 * 62.5 ns) to execute
delay2
        ADD R4, #1              ; increment by one which sets the default base clock speed
        CMP R4, R5              ; check number of iterations
        BLE delay2              ; continue if iterated less than 0xFFFFF + 1 times, otherwise repeat delay loop

        MOV32 R1, #0x08         ; load value for turning on LED color GREEN
        STR R1, [R0, #0x38]     ;write the above value to GPIOC ODR

        B loop                  ;do it all over again, forever

        ALIGN
        END
```

Code Debugging

From the file hello_students.asm, where is the message “Hello Students!” located in memory and where is it being written?

RO: Source Addr. (ROM)	R1: Destination Addr. (RAM)	R2: Data	R2: ASCII	R3: Counter
0x0000.0118	0x2000.03ED	0x48	H	4
0x0000.0119	0x2000.03EE	0x65	e	
0x0000.011A	0x2000.03EF	0x6C	l	
0x0000.011B	0x2000.03F0	0x6C	l	
0x0000.011C	0x2000.03F1	0x6F	o	3
0x0000.011D	0x2000.03F2	0x20		
0x0000.011E	0x2000.03F3	0x53	S	
0x0000.011F	0x2000.03F4	0x74	t	
0x0000.0120	0x2000.03F5	0x75	u	2
0x0000.0121	0x2000.03F6	0x64	d	
0x0000.0122	0x2000.03F7	0x65	e	
0x0000.0123	0x2000.03F8	0x6E	n	
0x0000.0124	0x2000.03F9	0x74	t	1
0x0000.0125	0x2000.03FA	0x73	s	
0x0000.0126	0x2000.03FB	0x21	!	
0x0000.0127	0x2000.03FC	0x00	NUL	

HELLO_STUDENTS.asm (original)

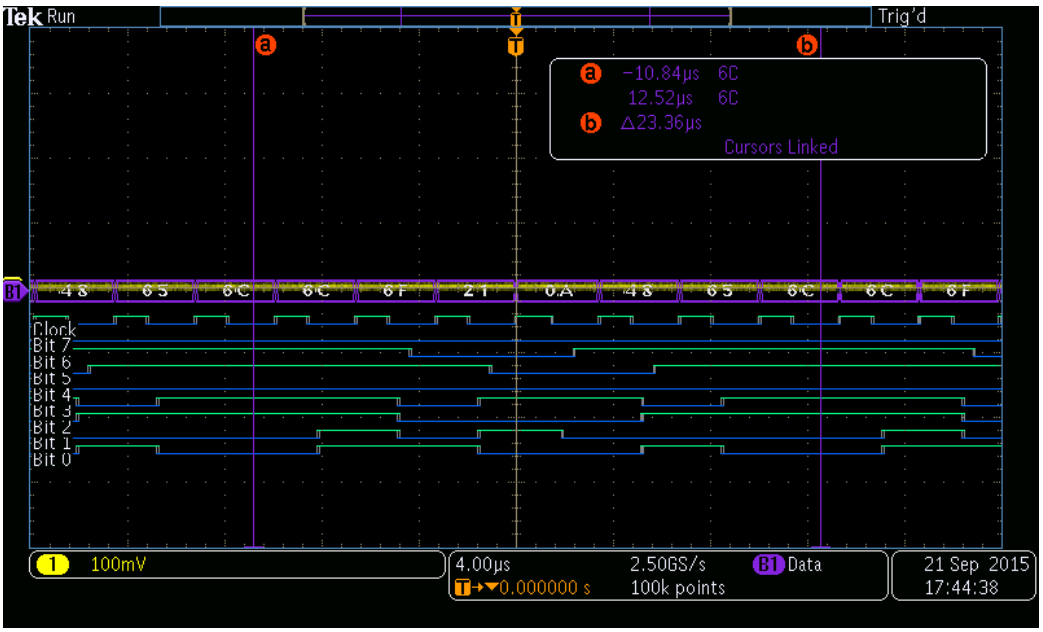
```
   AREA main, CODE, READONLY, ALIGN=2
   THUMB
   EXPORT Start

message  DCB "Hello Students!",0 ; message stored readonly

         ALIGN ;pg149
Start    LDR R0, =message ; load address of the message

         MOV R1, SP ; load memory location to store

load     LDR R2,[R0] ; load a word of the message
         ADD R0, R0, #4 ; adds 4 to the pointer
         MOV R3, #4 ; used as a counter

    SUB R3, R3, #1 ; decrements counter
    STR R2,[R1,#4] ; store the word to memory, inc R1 by
4

         CMP R3, #0 ; check for null
         BNE load ; repeat if not null terminated

loop     B loop

   ALIGN
   END
```

HELLO_STUDENTS.asm (modified)

```
   AREA main, CODE, READONLY, ALIGN=2
   THUMB
   EXPORT Start

message  DCB "Hello Students!",0 ; message stored readonly

         ALIGN ;pg149
Start    LDR R0, =message ; load address of the message

         MOV R1, SP ; load memory location to store
         SUB R1, #19 ; offset memory location to store
         MOV R3, #4 ; used as a counter

load     LDR R2,[R0] ; load a word of the message
         STR R2,[R1] ; store the word to memory
         ADD R0, R0, #4 ; adds 4 to the src addr reg
         ADD R1, R1, #4 ; adds 4 to the dst addr reg
         SUB R3, R3, #1 ; decrements counter
         CMP R3, #0 ; check for null
         BNE load ; repeat if not null terminated

loop     B loop

   ALIGN
   END
```

Logic Analyzer

After flashing the program ‘logic analyzer.c’ to the microcontroller, configure the logic analyzer to capture the ASCII character \n (0x0A).



Data	ASCII
0x48	H
0x65	e
0x6C	l
0x6C	l
0x6F	o
0x21	!
0x0A	\n