Lab Report 2

Nathan Tipton, Partner: Tarrin Rasmussen

September 27, 2016

Contents

1 Objectives

The purpose of this lab is to practice connecting external peripherals to the microcontroller and use delay loops.

2 Overview

For this lab, we will be writing a ten bit binary counter in assembly. The counter is required to increments at a frequency of 2 Hz.We will display the counter on an LED bar graph. Three buttons will be used to start, stop and reset the counter. GPIO ports will be used to interface the LED display and switches with our microcontroller. Delay timing will be verified using a logic analyzer.

3 Procedure

3.1 LED Display

1. The LED display and bar resistor were connected to the breadboard.
2. Port A: pins [7:2] and Port B: pins [3:0] were used to interface between the microcontroller and LED display.
3. From the menu, select Project ->Build Target. As long as there weren't any errors, your project is ready to run!

3.1.1 Results

Connecting the LED display and bar resistor was fairly straight forward. The only problem we ran into was our unfamiliarity of the bar resistor. We did not know what it was exactly. After examining the datasheet for the bar resistor we were able to understand how to connect in with the LED display. We realized we needed one extra 220 Ohm resistor inorder for each LED on the display to have one.

3.2 Buttons

The LED display will be controlled by three buttons that perform the following actions: RESET, START, and STOP.

1. A button for the RESET was connected to the breadboard. This button was connected to Port C: Pin [4].
2. The two internal buttons on the microcontroller will be used for STOP and START. Port F: pins [4,0] were used for these buttons.

3. Select Debug -> Start/Stop Debug Session and your debugger will automatically launch. In the debugger you have access to all of the memory on your device. You can step through the code using F10/F11 or you can let it run by pressing F5.

4. Select View -> System Viewer -> GPIO -> GPIOF. This is the port which the LEDs on your physical

3.2.1 Results

We ran into some difficulties with the setup of our buttons. We didn't fully understand the GPIOLOCK and GPIOCR registers initally. We were trying to read from the PF0 pin for our start button and we did not know why it wasn't working. After reading the datasheet several times we came to understand that you can't just unlock the port. After unlocking the port you still need to configure the pin in the GPIOCR register inorder for a value to be read.

3.3 Code

3.3.1 Requirements

1. The counter will count at 2 HZ as verified by the logic analyzer.

2. When the stop button is pressed, the counter will pause with the current value still visible.

3. When the start button is pressed, the counter will resume from the current count.

4. When the reset button is pressed, the counter will start counting from zero.

3.3.2 Results

First part of the code is initializing and configuring all the registers for our Ports: A, B, C, and F. For our counter, we decided to have two nested loops incrementing the data for Ports A and B. In each loop the buttons would be checked and the proper actions initiated when pressed. Figure 3 shows our code planned out. While implementing the code we had some trouble using the branch with the link register. We could not get the syntax for the instruction correct. After going through the textbook, we were able to troubleshoot and correct our mistakes.

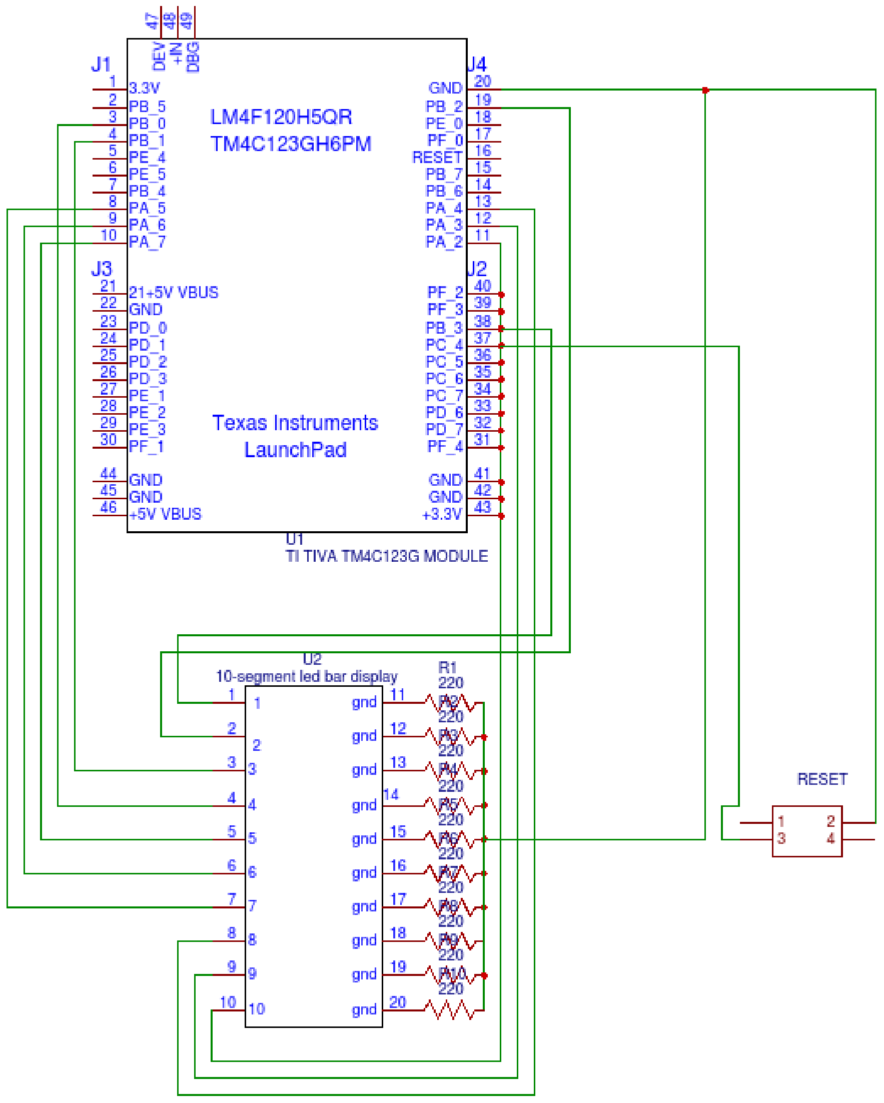


Figure 1: Schematic for counter

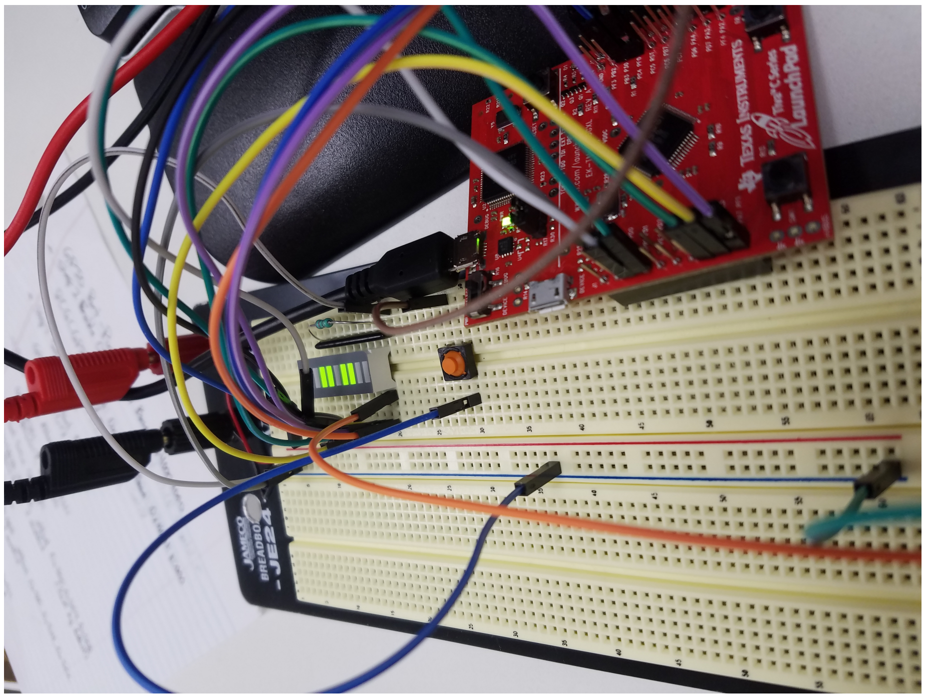


Figure 2: Physical Setup

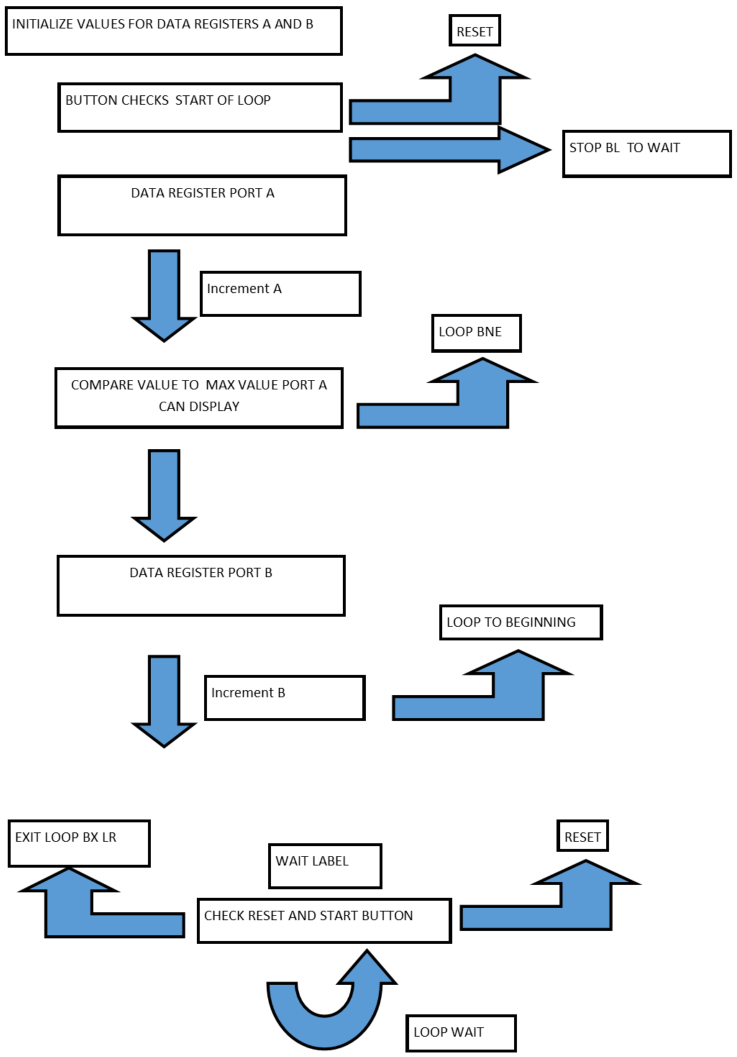


Figure 3: Flowchart for counter code.

```
 Start

            MOV32 R0, #0x40004000 ;Port A Base
            MOV32 R2, #0x400FE608 ;Clock Base

        ;Enable Clock
            LDR R1, [R2]
            ORR R1, #0x27 ;0b00111111
            STR R1, [R2]

            MOV32 R3, #2 ;Clock init
dummy SUBS R3, #1
            BNE dummy

            ;
            ; Setup Port A [7:2]
            ;

        ;Set Direction
            LDR R1, [R0, #0x400]
            ORR R1, #0xFC ;#0b11111100
            STR R1, [R0, #0x400]

        ;Disable Alternate
            LDR R1, [R0, #0x420]
            AND R1, #0x3 ;#0b00000011
            STR R1, [R0, #0x420]

            ;Set Drive Strength
            LDR R1, [R0, #0x508]
            ORR R1, #0xFC ;#0b11111100
            STR R1, [R0, #0x508]

            ;Pin Function
            LDR R1, [R0, #0x50C]
            ORR R1, #0xFC ;#0b11111100
            STR R1, [R0, #0x50C]

            ;Enable Pin
            LDR R1, [R0, #0x51C]
            ORR R1, #0xFC ;#0b11111100
            STR R1, [R0, #0x51C]

            ;Write to Pin
            LDR R1, [R0, #0x3F0]
            ORR R1, #0xFF ;#0b11111111
            STR R1, [R0, #0x3F0]

            ;
            ; Setup Port B [3:0]
            ;

            MOV32 R0, #0x40005000 ;Port B Base

        ;Set Direction
            LDR R1, [R0, #0x400]
            ORR R1, #0x0F ;#0b00001111
            STR R1, [R0, #0x400]

        ;Disable Alternate
            LDR R1, [R0, #0x420]
            AND R1, #0xF0 ;#0b11110000
            STR R1, [R0, #0x420]

            ;Set Drive Strength
```

```
    LDR R1, [R0, #0x508]
    ORR R1, #0x0F ;#0b00001111
    STR R1, [R0, #0x508]

    ;Pin Function
    LDR R1, [R0, #0x50C]
    MOV R1, #0xFF ;#0b11111111
    STR R1, [R0, #0x50C]

    ;Enable Pin
    LDR R1, [R0, #0x51C]
    ORR R1, #0x0F ;#0b00001111
    STR R1, [R0, #0x51C]

    ;Write to Pin
    LDR R1, [R0, #0x3C]
    MOV R1, #0x0F ;#0b00001111
    STR R1, [R0, #0x3C]

    ;
    ; Setup Port C [4]
    ;

    MOV32 R0, #0x40006000 ;Port C Base

;Set Direction
    LDR R1, [R0, #0x400]
    AND R1, #0xEF ;#0b11101111
    STR R1, [R0, #0x400]

;Disable Alternate
    LDR R1, [R0, #0x420]
    AND R1, #0xEF ;#0b11101111
    STR R1, [R0, #0x420]

    ;Set Drive Strength
    LDR R1, [R0, #0x508]
    ORR R1, #0x10 ;#0b00010000
    STR R1, [R0, #0x508]

    ;Pin Function
    LDR R1, [R0, #0x510]
    ORR R1, #0x10 ;#0b00010000
    STR R1, [R0, #0x510]

    ;Enable Pin
    LDR R1, [R0, #0x51C]
    ORR R1, #0x10 ;#0b00010000
    STR R1, [R0, #0x51C]

    ;Write to Pin
    ;LDR R1, [R0, #0x3F0]
    ;ORR R1, #0xFF ;#0b11111111
    ;STR R1, [R0, #0x3F0]

    ;
    ; Setup Port F [4&0]
    ;

    MOV32 R0, #0x40025000 ;Port F Base

    ;Unlock Port
    MOV32 R1, #0x4C4F434B ;Unlock Code
    STR R1, [R0, #0x520]
    LDR R1, [R0, #0x524]
    ORR R1, #0xFF ;0b11111111
    STR R1, [R0, #0x524]
    ;Set Direction
        LDR R1, [R0, #0x400]
        AND R1, #0xEE ;#0b11101110
        STR R1, [R0, #0x400]

    ;Disable Alternate
        LDR R1, [R0, #0x420]
        AND R1, #0xEE ;#0b11101110
        STR R1, [R0, #0x420]

        ;Set Drive Strength
        LDR R1, [R0, #0x508]
        ORR R1, #0x11 ;#0b00010001
        STR R1, [R0, #0x508]

        ;Pin Function
        LDR R1, [R0, #0x510]
        ORR R1, #0x11 ;#0b00010001
        STR R1, [R0, #0x510]

        ;Enable Pin
        LDR R1, [R0, #0x51C]
        ORR R1, #0x11 ;#0b00010001
        STR R1, [R0, #0x51C]

        ;Write to Pin
        ;LDR R1, [R0, #0x3F0]
        ;ORR R1, #0xFF ;#0b11111111
        ;STR R1, [R0, #0x3F0]

        ;
        ; Counter
        ;

        MOV32 R0, #0x40004000 ;Port A Base
        MOV32 R1, #0x40005000 ;Port B Base
        MOV32 R6, #0x40006000 ;Port C Base
        MOV32 R8, #0x40025000 ;Port F Base

reset MOV R2, #0xFB ;#0b11111011
        STR R2, [R0, #0x3F0]
        MOV R5, #0x0F ;#0b00001111
        STR R5, [R1, #0x3C]
        MOV32 R3, #610000

loop LDR R7, [R6, #0x40] ;Check Reset Button
        CMP R7, #0
        BEQ reset

        LDR R7, [R8, #0x40] ;Check Stop Button
        CMP R7, #0
        BLEQ wait

        SUBS R3, #1 ;Delay Loop
        BNE loop

        STR R2, [R0, #0x3F0] ;Increment A
        MOV32 R3, #610000
        SUB R2, #4
        CMP R2,#−5;

        BNE loop ; Increment B
        MOV R2, #0xFB ;#0b11111011
        SUB R5, #1
        STR R5, [R1, #0x3C]
        B loop
```

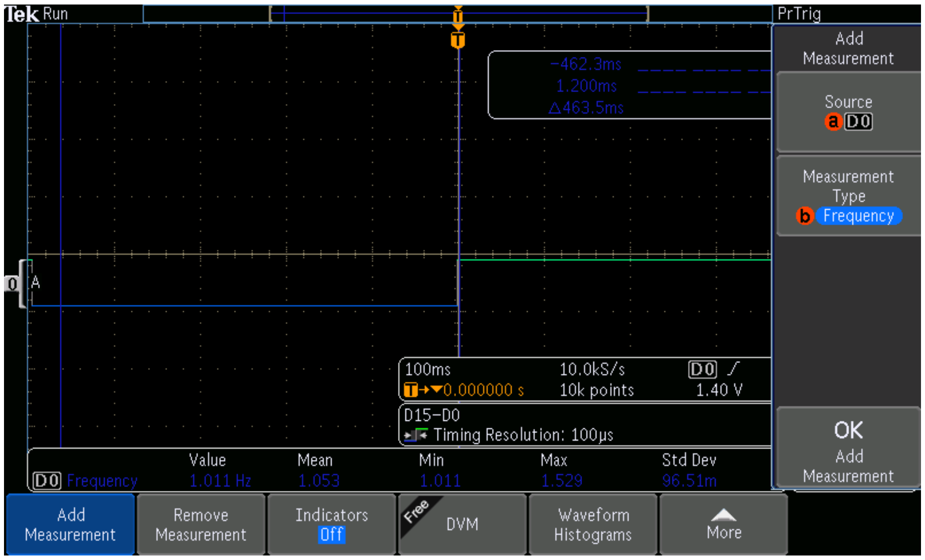
```
wait LDR R7, [R6, #0x40] ;Check Reset Button
             CMP R7, #0
             BEQ reset
             LDR R7, [R8, #0x4] ;Check Start Button
             CMP R7, #1
             BEQ wait
             BX LR


        B Start
```

3.4 Frequency

3.4.1 Requirements

The required frequency was 2 Hz.

1. The delay loop was modified to obtain the required frequency.
2. Logic Analyzer probe was set up to measure the first LED in the LED display.
3. Measurement of frequency was selected on the Logic Analyzer.
4. Measured value was 1.011 Hz.



−462.3ms
1.200ms
Δ463.5ms

100ms 10.0kS/s D0 1.40 V
T→0.000000 s 10k points
D15–D0
Timing Resolution: 100µs

	Value	Mean	Min	Max	Std Dev
D0 Frequency	1.011 Hz	1.053	1.011	1.529	96.51m

Figure 4: Frequency measurement

3.4.2 Results

We measured the frequency from the first LED on the display. However, the LED only display every other cycle. If we take the frequency of the LED as shown in Figure 4 and multiply by 2 to represent every cycle, we will obtain the required frequency of 2 Hz.

4 Conclusion

We had some trouble at the start of the lab. The debugger wasn't working for us, making it difficult to understand what our code was doing. This slowed us down but we eventually worked through it. We learned a great deal about creating loops and different branch conditions. Branching with the Link Register is going to be a very valuable tool which I understand better now because of this lab. We also understand the GPIOLOCK and GPIOCR register more fully. After this lab, I will be studying the different registers that you have to configure for the ports. These registers were the parts of the lab that I struggled with the most. I want to better understand the purpose of each register and how to configure it properly.