Objectives

- To execute the instructions of Lab 2 (ECE 3710 Lab 2.pdf) from the course wiki*.
- To gain experience interfacing the microcontroller with external peripherals using GPIO. In addition, to learn how to use delay loops and verify their accuracy using a logic analyzer.

Overview

"For this lab, you will be writing a ten bit binary counter in assembly that increments at a frequency of 2 Hz (0.5 seconds). The value of the counter will be displayed on an LED bar graph. Three buttons will be used to start, stop and reset the counter. The program will be run on your microcontroller and it will interface with an LED display and switches using GPIO ports. All timing requirements will be verified using a logic analyzer." (ECE 3710 Lab 2.pdf)

Preparation

- Come to lab with the microcontroller board and breadboard.
- Be prepared to use most of the instructions that have been learned in class.
- Read V1.Ch2.2 and V1.Ch4.2.2 of course textbooks.
- Obtain the following items from the department lab store:
    - 10 LED Bar Graph
    - Push Button
    - Bar Resistor - 220 Ohm
    - Standard Resistor - 220 Ohm
    - Ribbon Cable
    - Leads for the Voltage Source

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

Procedure

LED Display and Buttons

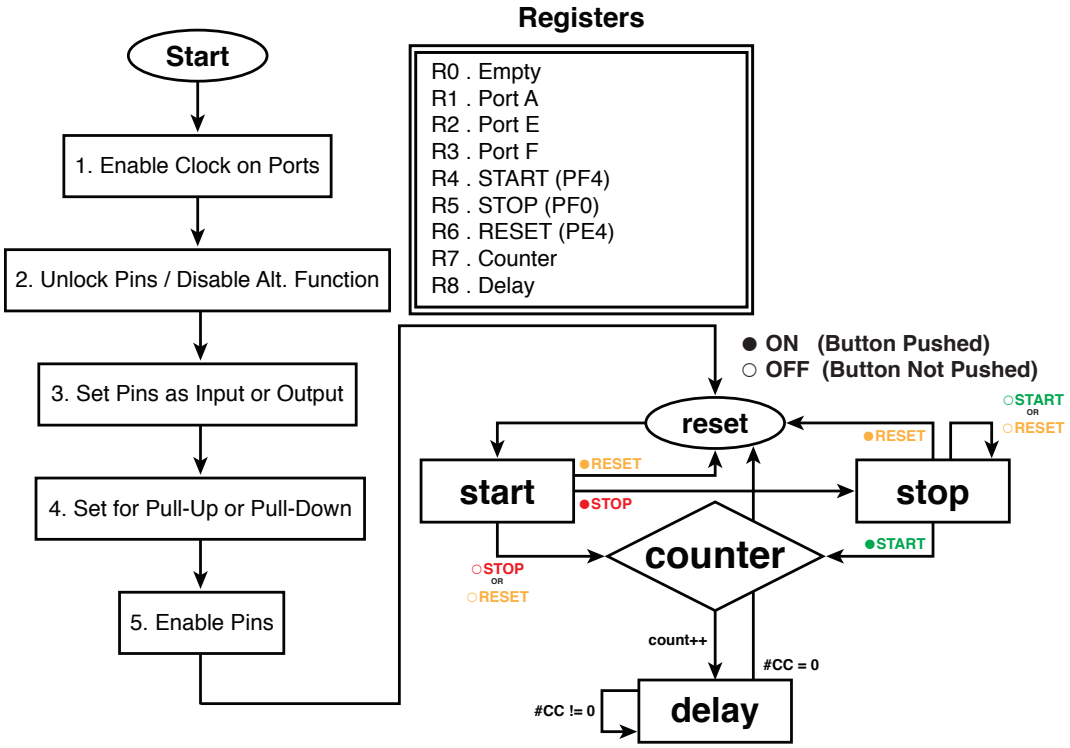
See file ECE 3710 Lab 2.pdf for setup details.

Software Requirements

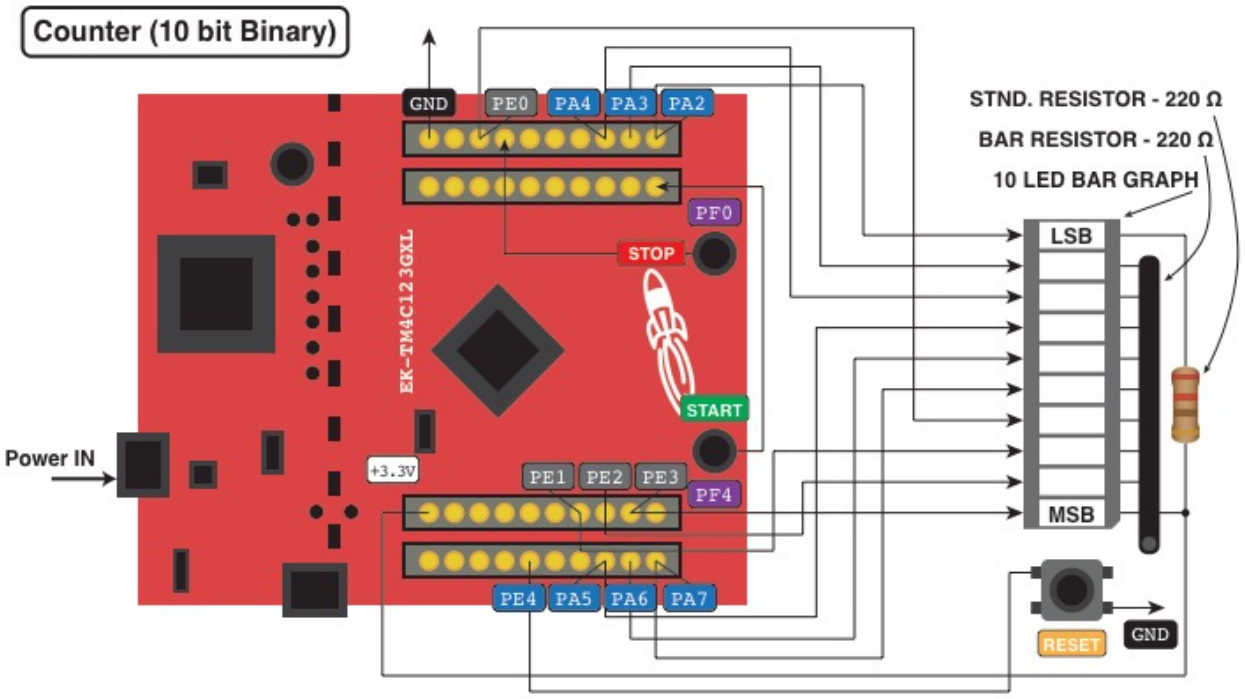
1. The counter will count at 2 Hz +/- 5% as verified by the logic analyzer.
2. When the stop button is pressed, the counter will pause with the current value still visible.
3. When the start button is pressed, the counter will resume from the current count.
4. When the reset button is pressed, the counter will start counting from zero.

Documentation

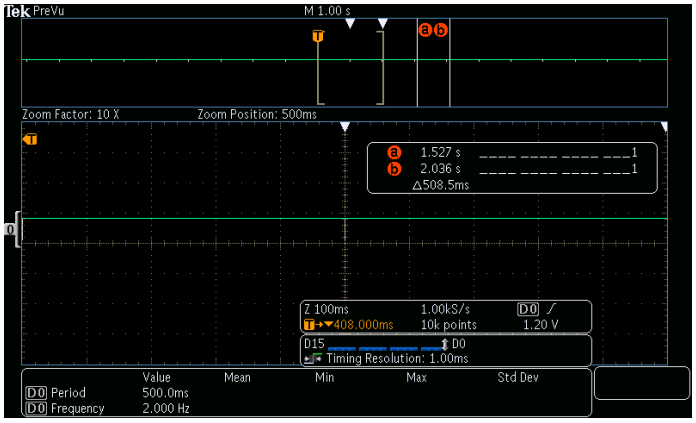
Software Design

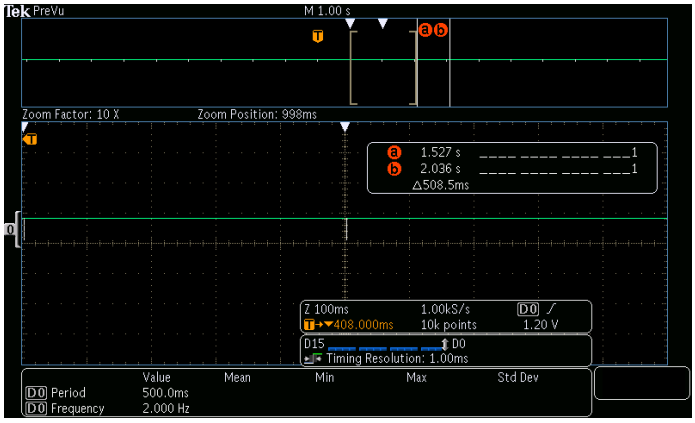


Hardware Schematic



Logic Analyzer Analysis





The counter counts at 500.0 ms per count which is within the specifications of the software requirements.

Program Code

```
       THUMB
       AREA    |.text|, CODE, READONLY, ALIGN=2
       EXPORT  Start
        ; Program Written by Joel Meine

Start
        ; 1. Enable Clock on Ports
        LDR R1,=0x400FE108                      ; Clock Base Address : RCGC2 (pg. 462 - MDS)
        MOV R0,#0x31                            ; Enable Clock on PF, PE, PA (0x31=0b0011.0001)
        STR R0,[R1]

        ; begin >> GPIO Port F (PF)
        LDR R3,=0x40025000                      ; Port Base Address (pg. 656 - MDS)
          ; 2. Unlock Pins / Disable Alt. Function
        LDR R0,=0x4C4F434B                      ; Unlock Code (pg. 681 - MDS)
        STR R0,[R3,#0x520]                      ; GPIO Lock : GPIOLOCK (pg. 681 - MDS)
        MOV R0,#0x11                            ; Unlock Pins - PF4 (START), PF0 (STOP) (0x11=0b0001.0001)
        STR R0,[R3,#0x524]                      ; GPIO Commit : GPIOCR (pg. 682 - MDS)
          ; 3. Set Pins as Input or Output
        MOV R0,#0xEE                            ; Enable as Input - PF4 (START), PF0 (STOP) (0xEE=0b1110.1110)
        STR R0,[R3,#0x400]                      ; GPIO Direction : GPIODIR (pg. 660 - MDS)
          ; 4. Set for Pull-Up or Pull-Down
        MOV R0,#0x11                            ; Set for Pull-Up Resistor (0x11 = 0b0001.0001)
        STR R0,[R3,#0x510]                      ; GPIO Pull-Up Select : GPIOPUR (pg. 674 - MDS)
          ; 5. Enable Pins
        MOV R0,#0x11                            ; Enable Pins - PF4 (START), PF0 (STOP) (0x11 = 0b0001.0001)
        STR R0,[R3,#0x51C]                      ; GPIO Digital Enable : GPIODEN (pg. 679 - MDS)
        ; end >> GPIO Port F (PF)

        ; begin >> GPIO Port E (PE)
        LDR R2,=0x40024000                      ; Port Base Address (pg. 656 - MDS)
          ; 2. Unlock Pins / Disable Alt. Function
        LDR R0,=0x4C4F434B                      ; Unlock Code (pg. 681 - MDS)
        STR R0,[R2,#0x520]                      ; GPIO Lock : GPIOLOCK (pg. 681 - MDS)
        MOV R0,#0x1F                            ; Unlock Pins - PE4 (RESET), PE3, PE2, PE1, PE0 (0x11=0b0001.1111)
        STR R0,[R2,#0x524]                      ; GPIO Commit : GPIOCR (pg. 682 - MDS)
          ; 3. Set Pins as Input or Output
        MOV R0,#0x0F                            ; Enable as Input - PE4 (RESET)
                                                ; Enable as Output - PE3, PE2, PE1, PE0 (0x0F=0b0000.1111)
        STR R0,[R2,#0x400]                      ; GPIO Direction : GPIODIR (pg. 660 - MDS)
          ; 4. Set for Pull-Up or Pull-Down
```

```
        MOV R0,#0x10                            ; Set for Pull-Up Resistor - PE4 (RESET) (0x10=0b0001.0000)
        STR R0,[R2,#0x510]                      ; GPIO Pull-Up Select : GPIOPUR (pg. 674 - MDS)
        MOV R0,#0x0F                            ; Set for Open-Drain - PE3, PE2, PE1, PE0 (0x0F=0b0000.1111)
        STR R0,[R2,#0x50C]                      ; GPIO Open Drain Select : GPIODR (pg. 673 - MDS)
          ; 5. Enable Pins
        MOV R0,#0x1F                            ; Enable Pins - PE4 (RESET), PE3, PE2, PE1, PE0 (0x11=0b0001.1111)
        STR R0,[R2,#0x51C]                      ; GPIO Digital Enable : GPIODEN (pg. 679 - MDS)
        ; end >> GPIO Port E (PE)

        ; begin >> GPIO Port A (PA)
        LDR R1,=0x40004000                      ; Port Base Address (pg. 656 - MDS)
          ; 2. Unlock Pins / Disable Alt. Function
        LDR R0,=0x4C4F434B                      ; Unlock Code (pg. 681 - MDS)
        STR R0,[R1,#0x520]                      ; GPIO Lock : GPIOLOCK (pg. 681 - MDS)
        MOV R0,#0xFC                            ; Unlock Pins - PA7, PA6, PA5, PA4, PA3, PA2 (0xFC=0b1111.1100)
        STR R0,[R1,#0x524]                      ; GPIO Commit : GPIOCR (pg. 682 - MDS)
          ; 3. Set Pins as Input or Output
        MOV R0,#0xFC                            ; Enable as Output - PA7, PA6, PA5, PA4, PA3, PA2 (0xFC=0b1111.1100)
        STR R0,[R1,#0x400]                      ; GPIO Direction : GPIODIR (pg. 660 - MDS)
          ; 4. Set for Pull-Up or Pull-Down
        MOV R0,#0xFC                    ; Set for Open-Drain - PA7, PA6, PA5, PA4, PA3, PA2 (0xFC=0b1111.1100)
        STR R0,[R1,#0x50C]                      ; GPIO Open Drain Select : GPIODR (pg. 673 - MDS)
          ; 5. Enable Pins
        MOV R0,#0xFC                            ; Enable Pins - PA7, PA6, PA5, PA4, PA3, PA2 (0xFC=0b1111.1100)
        STR R0,[R1,#0x51C]                      ; GPIO Digital Enable : GPIODEN (pg. 679 - MDS)
        ; end >> GPIO Port A (PA)

;       R0 - Empty
;       R1 - Port A
;       R2 - Port E
;       R3 - Port F
;       R4 - START (PF4)
;       R5 - STOP (PF0)
;       R6 - RESET (PE4)
;       R7 - Counter
;       R8 - Delay

reset                                           ; Returns program to initial state.
        LDR R8,=0xF2340                         ; Number of Clock Cycles in Delay Loop >> 2 Hz +- 5% or 500 ms
                                                ; >> (1.9 Hz - 2.1 Hz) or (475 ms - 525 ms)

start                                           ; Checks if RESET or STOP buttons have been pushed.
        LDR R6,[R2,#0x3FC]                      ; Check state of RESET button; i.e. ON (0) or OFF (1).
        LSR R6,#4                               ; Isolate RESET bit.
        CMP R6,#0                               ; Is RESET button pushed?
        MOVEQ R7,#0x3FF                         ; If yes, then set counter to initial value and return to 'reset'.
        BEQ reset                               ; ...
        LDR R5,[R3,#0x3FC]                      ; Check state of STOP button; i.e. ON (0) or OFF (1).
        AND R5,#0x01                            ; Isolate STOP state.
        CMP R5,#0                               ; Is STOP button pushed?
        BEQ stop                                ; If yes, then go to 'stop'.
        B counter                               ; Proceed to 'counter'.

stop                                            ; Checks if START or RESET buttons have been pushed.
        LDR R4,[R3,#0x3FC]                      ; Check state of START button; i.e. ON (0) or OFF (1).
        AND R4,#0x10                            ; Isolate START state.
        CMP R4,#0                               ; Is START button pushed?
        BEQ counter                             ; If yes, then proceed to 'counter'.
        LDR R6,[R2,#0x3FC]                      ; Check state of RESET button; i.e. ON (0) or OFF (1).
        LSR R6,#4                               ; Isolate RESET bit.
        CMP R6,#0                               ; Is RESET button pushed?
        MOVEQ R7,#0x3FF                         ; If yes, then set counter to initial value and return to 'reset'.
        BEQ reset                               ; ...
        LDR R5,[R3,#0x3FC]                      ; Check state of STOP button; i.e. ON (0) or OFF (1).
        B stop                                  ; Return to 'stop'.
```

```
counter                                         ; Counts from 0 to 1023 (Active-Low).
        SUB R7,#1                               ; Increment count.
        CMP R7,#0                               ; Is count equal to the maximum count value?
        MOVEQ R7,#0x3FF                         ; If yes, then set counter to initial value.
        LSL R0,R7,#2                            ; Shift Bits for Writing Only to PA:7-2 -- #2 means skipping PA:1-0.
        STR R0,[R1,#0x3FC]                      ; ...
        LSR R0,R0,#8                    ; Shift Bits for Writing Only to PE:3-0 -- #8 means storing on PE:3-0.
        STR R0,[R2,#0x3FC]                      ; ...

delay
        SUBS R8,#1                              ; Decrement number of clock cycles.
        BNE delay                               ; Repeat 'delay' until clock cycle count reaches zero.
        B reset                                 ; Return to 'reset'.

      ALIGN
      END
```

Commentary

- Lab 2 was completed with considerable complications and severe delays due to unforeseen errors in the hardware setup. Per the results during the simulation, as I firmly thought from the beginning the code itself was not in error. The error and lack of understanding came from the hardware setup where it wasn't clear how the LED display was to receive power or where the bar resistor plugged in relative to all other components.
- Relatively speaking, even after convincing myself that the hardware setup was right and the code was perfectly fine, the main breakthrough in finally getting the LED display to display the count was merely due to an apparent misunderstanding of how the program is flashed onto the microcontroller. I had expected that once the program was flashed onto the board that I would immediately see the LED display light up and start counting after pushing the assigned START button. My assumption was incorrect as I learned that first the microcontroller's onboard RESET button had to be pressed first before the program would run at all.
- The code is optimized to the furthest extent my present experience can yield. However, there are two considerations of the code that still does not fully satisfy what I was ideally looking to achieve.
    - One, in the "start" and "stop" states, I reused the RESET button check code in both states. Ideally it would have been better to only have to declare the RESET button check code once, but after numerous attempts and variations of the code to accomplish that task, I ultimately quit just to get the behavior working correctly.
    - Two, the responsiveness of the button pushes is mostly decent but dependent on how long the button is pushed. If pushed quickly, more often than not the program isn't reliably responsive. If pushed slowly, then the button pushes are very responsive and reliable.
    - I would be very interested in receiving the insight of a colleague if they can identify how to correct the two points I commented on earlier.