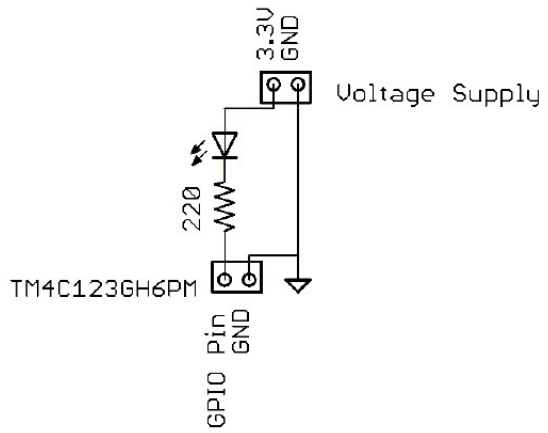
Xuecong Fan A01972388
Robert McKay
Sep 25, 2017

Lab 2 Report

Introduction:

For this lab, we would be written a ten-bit binary counter in assembly that increments at a frequency of 2 Hz. The value of the counter would be displayed on the LED bar graph. We use three buttons which are start, stop and reset to count. Our program would interface with an LED display and switches using GPIO ports. For the timing requirements would be verified using logic analyzer.

Procedure:



Picture 1

For LED display would be connected to microcontroller bread board. One ground pin is connected microcontroller and the voltage source to the breadboard. The ground could be same. Use picture 1 to see how they are connect on the board.

To programmed this lab section in the keil. First, we programmed initializing the microcontroller. Then, we programmed a count function with a delay. For the counter it would count at 2Hz +/- 5% as verified by the logic analyzer. We need stop button to pause with the current value, start button to resume from current count and reset button to start counting from zero for programmed.

Figure 1,2,3 under the results are the code for this lab section.

Figure 4 under the results is the logic analyzer output.

Figure 5under the results is schematic by handwrite.

Result:

```
;
;Summary--Designing a 10-bit counter with start, stop, and reset
;
;Inputs: PA4 (RESET), PF0 (START), PF4 (STOP)
;Outputs: PB0-PB7, PA2, PA3 (LEDs for counting)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Pseudocode
;
;begin
;   count
;   if (stop)
;       stop
;   if (reset)
;       reset
;   loop back to count
;end
;
;begin stop
;   if (start)
;       count
;       loop back to stop
;end stop
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

        ALIGN
Start
    ldr R1, =0x400FE608 ;Load the clock address and prepare to turn it on
    mov R0, #0x23; 0010 0011, to open ports A, B, and F
    str R0, [R1]; write this to turn on the clock

    ldr R1, =0x40004000 ;get address of port A
    ldr R2, =0x40005000 ;get address of port B
    ldr R3, =0x40025000 ;get address of port F

    ;mov32 R0, #0x4C4F4348 ;unlock code, used to open port F
    ;str R0, [R3, #520]

    mov R0, #0xFF ;1111 1111, used to enable pins
    str R0, [R1, #0x524] ;Lets us use pins in port A by writing 1s in GPIOCR
    str R0, [R2, #0x524] ;Lets us use pins in port B by writing 1s in GPIOCR
    ;mov R0, #0xFF ;might be necessary to only enable the pins we want
    ;str R0, [R3, #0x524] ;Lets us use pins PF0 and PF4 by writing 1s in GPIOCR
    mov32 R0, #0x4C4F4348 ;Unlock code
    mov R4, #0xFF
    str R0, [R3, #0x520] ;stores the unlock code to manipulate F
    str R4, [R3, #0x524] ;changes GPIOCR in F to FF

    mov R0, #0xFF ;will be used to configure all ports in B
    str R0, [R2, #0x400] ;configures pins in port B to be output
    mov R0, #0xC ;0000 1100
    str R0, [R1, #0x400] ;configures PA2 and PA3 to be output
    mov R0, #0x0 ;0000 0000
    str R0, [R3, #0x400] ;configures port F to be input

    mov R0, #0x11 ;0001 0001
    str R0, [R3, #0x510] ;Sets pull up resistors for PF0 and PF4
```

Figure 1: part 1 for code

```
    str R0, [R3, #0x510] ;Sets pull up resistors for PF0 and PF4
    mov R0, #0x10 ;0001 0000, sets pull up resistor for PA4
    str R0, [R1, #0x510] ;

    mov R0, #0x1C ;0001 1100
    str R0, [R1, #0x51C] ;Digital enable for pins in A (PA2-PA4)
    mov R0, #0xFF
    str R0, [R2, #0x51C] ;Digital enable for pins in B
    mov R0, #0x11
    str R0, [R3, #0x51C] ;Digital enable for pins in F

    mov R0, #0x0 ;0000 0000
    str R0, [R1, #0x420] ;clear alternate functions in A
    str R0, [R2, #0x420] ;clear alternate functions in B
    str R0, [R3, #0x420] ;clear alternate functions in F

    ;set active low for all pins
    mov R0, #0xFF
    str R0, [R2, #0x3FC];set all pins for B
    mov R0, #0x1C
    str R0, [R1, #0x3FC];set PA2-PA4
    mov R0, #0x11
    str R0, [R3, #0x3FC];set PF0 and PF4

reset   mov R0, #0x0
        ldr R1, =0x40005000 ;get base address for port b
        mvn R0, R0 ;invert the bits
        str R0, [R1, #0x3FC] ;display the count on port B LEDs (First 8)
        mov R2, R0 ;get a new register so you don't mess with count
        lsr R2, #6 ;do a shift to get the most significant 2 bits lined up with
            PA6 and PA7
        ldr R1, =0x40004000 ;get base address for port a
        str R2, [R1, #0x3FC] ;display the count on port A LEDs (Last 2)

count   ldr R1, =0x40005000 ;get base address for port b
        add R0, #0x1 ;increment the count
        mvn R0, R0 ;invert the bits
        str R0, [R1, #0x3FC] ;display the count on port B LEDs (First 8)
        mov R2, R0 ;get a new register so you don't mess with count
        lsr R2, #6 ;do a shift to get the most significant 2 bits lined up with
            PA6 and PA7
        ldr R1, =0x40004000 ;get base address for port a
        str R2, [R1, #0x3FC] ;display the count on port A LEDs (Last 2)
        mvn R0, R0 ;invert the bits to start counting again

        mov32 R3, #0x5CC0E ;Sets the value of iterations
delay   sub R3, #0x1 ;decrement the max value
        nop
        mov R4, #0x10 ;sets the value that would correspond with the stop button
            being pressed
        ldr R1, =0x40025000 ;We're going to check port F for the values
        ldr R5, [R1, #0x3FC] ;Loads the value of port F
        cmp R4, R5 ;checks the stop
        beq stop ;if the button is pressed, it goes.
        ldr R5, =0x40004000 ;loads port A, we'll be looking at PA4
        ldr R4, [R5, #0x3FC] ;loads the info we want, PA4 included
        and R4, #0x10 ;masks the bits from PA
        cmp R4, #0x0 ;checks if reset is being pressed
        beq reset
        cmp R3, #0x0 ;checks to see if we're ready to count again
        bne delay
        b count
```

Figure 2: part 2 for code

```
stop    ldr R5, [R1, #0x3FC] ;loads port F in preparation to check start
        mov R4, #0x01 ;moves the value for start
        cmp R4, R5 ;checks the start value
        beq count
        ldr R5, =0x40004000 ;loads port A, we'll be looking at PA4
        ldr R4, [R5, #0x3FC] ;loads the info we want, PA4 included
        and R4, #0x10 ;masks the bits from PA
        cmp R4, #0x0 ;checks if reset is being pressed
        beq reset
        b stop

        ALIGN
        END
```

Figure 3: part 3 for code

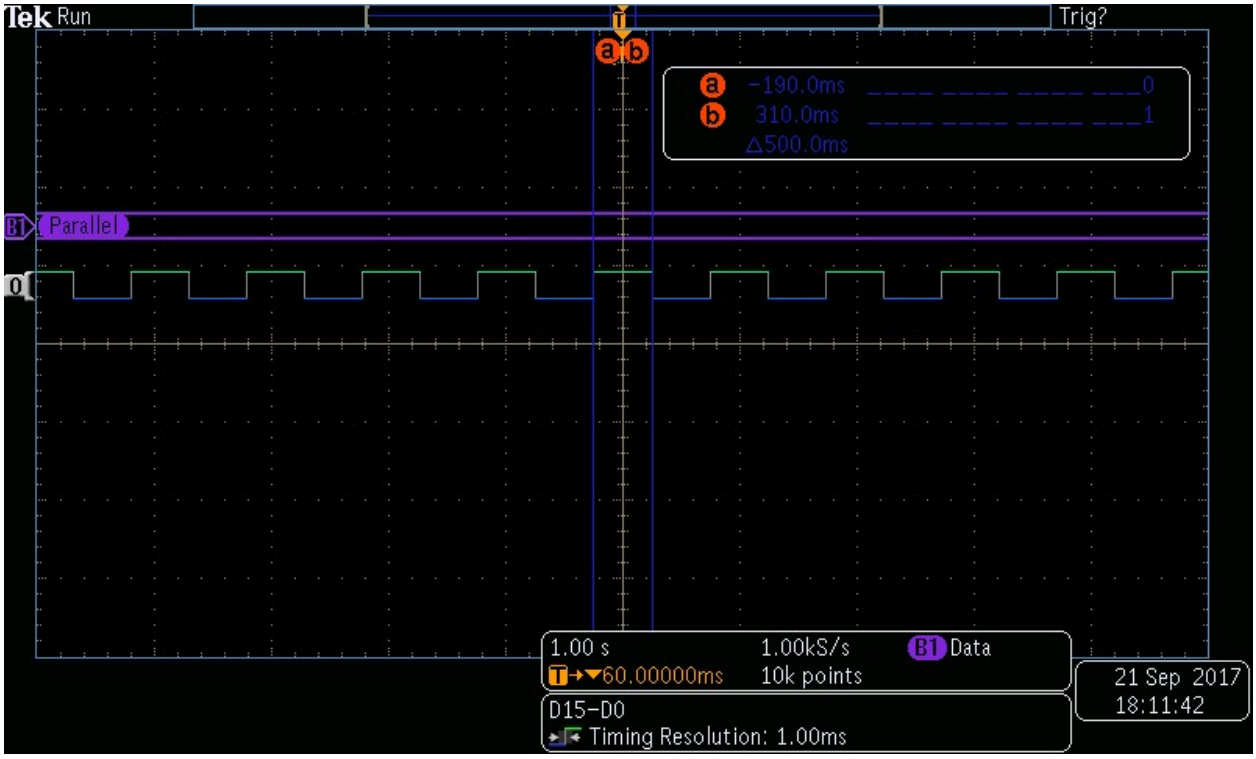


Figure 4

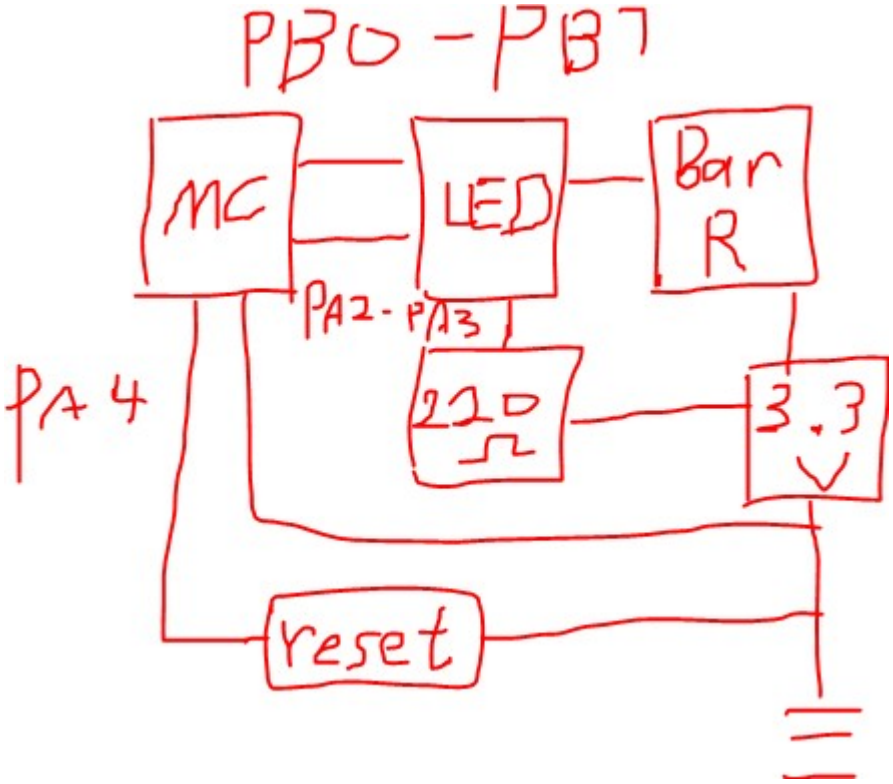


Figure 5

Issues:

This code part is the hardest part for the lab section, because we need debugged long time when we did this lab. Also, since we programmed the code we have to use the textbook and other resource to search the correct way for the code.

Conclusions:

For this lab we can learned how to programmed code and how to connected physical board to our code. Know where to search the information we need for how to write the code and how to set up the physical board.