Assignment 2: Digital I/O with TIMERB

Team members:

Priya Wilkhu
Elizabeth Sotomayer
Matthew Sharp
Bekhzodbek Moydinboyev

I. Problem description

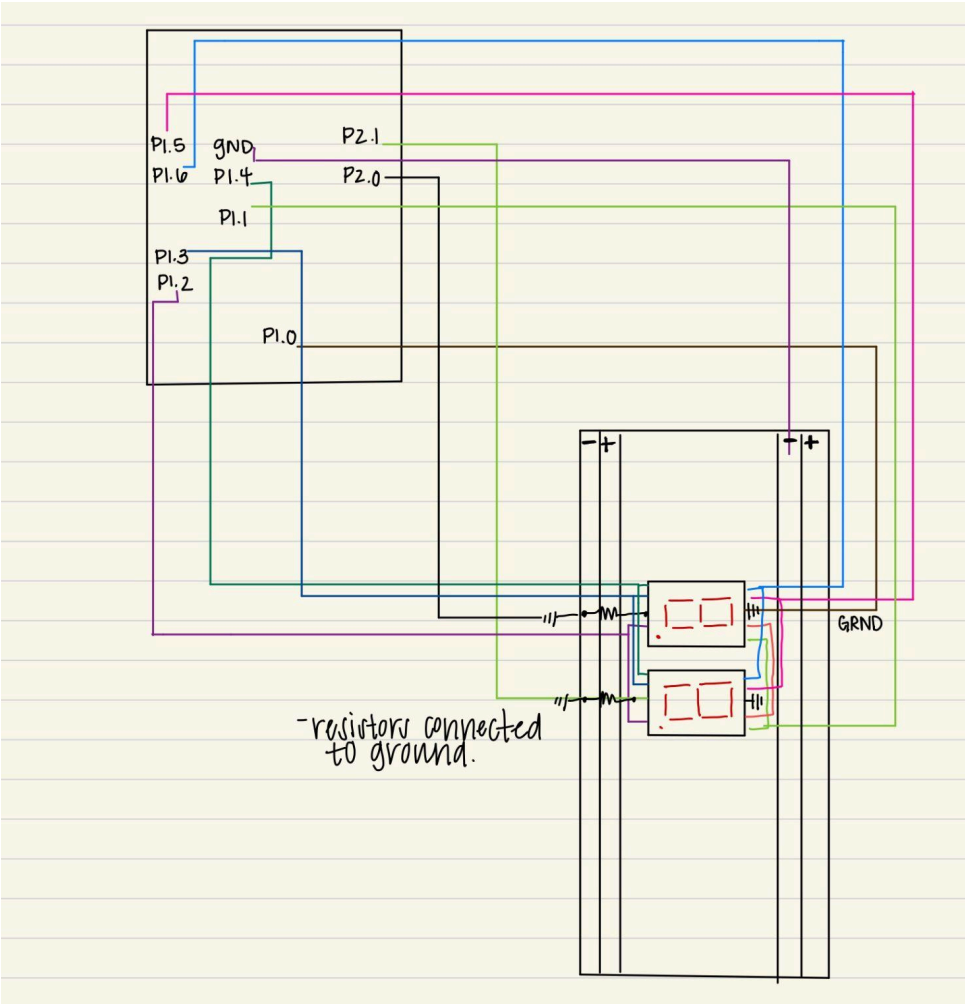
The objective of this assignment is to design and implement a digital counter using Digital I/O and TIMERB, which counts from 00 to 59 and displays the count on two seven-segment displays. The counter increments every second, requiring precise timing control. Given the constraint of using only nine I/O pins, efficient utilization of hardware and software resources is essential. The TIMERB module will be configured to generate a one-second delay, ensuring accurate timing for the counter. The project involves calculating the appropriate timer values to achieve the desired one-second intervals, integrating Digital I/O operations to drive the seven-segment displays, and documenting the entire process. The final deliverables include a detailed project report and a video demonstration of the working system. This assignment requires collaborative teamwork and practical application of embedded system concepts, as outlined in Module 3's interactive lecture videos.

Video:

https://youtu.be/JjYLVobS7XY?si=riUmxbAgGE_u2i-9

II. Wiring Diagram

Using a common cathode



Code for each number on displays

```
; bitmapping: 0GFE DCBA
; 0: 0011 1111 = 0x3F
; 1: 0000 0110 = 0x06
; 2: 0101 1011 = 0x5B
; 3: 0100 1111 = 0x4F
; 4: 0110 0110 = 0x66
; 5: 0110 1101 = 0x6D
; 6: 0111 1101 = 0x7D
; 7: 0000 0111 = 0x07
; 8: 0111 1111 = 0x7F
; 9: 0110 1111 = 0x6F
```

III. Pseudocode

init:
    Allocate storage for each segment display value 0-9
    Setup timer B
    Enable interrupts and clear flag

main:
    Initalize port 1.0-1.7 and 2.0,2.1 as output
    Unlock GPIO
    Start register R5 and R7 at 0
    Load 0 digit to displays

loop:
    Call display digits function
    Jump back to loop label

Display Digits:
    Load digit pattern from R5 to register R6
    Enable only display 2
    Send pattern to port 1
    Call Delay
    Load pattern from R7 to R8
    Enable only display 1
    Send pattern to port 1
    Call delay
    Return to loop

Delay:
    Load high value to R4

Delay_loop:
    Decrement R4
    Jump back to Delay_loop til R4=0

ISR:

    Clear the CCIFG flag
    Increment display 2 register
    Compare R5 with value 10
     If not equal, jump to no_tens_increment function
            Else, load 0 to R5 to start over seconds
    Increment display 1 register
    Compare with value 6

Jump to no_tens_increment if display 1 != 6
Else, move 0 to display 1 to reset counter
no_tens_increment:
Return from interrupt

IV. Code

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;-------------------------------------------------------------------------------
        .cdecls C,LIST,"msp430.h"       ; Include device header file
;-------------------------------------------------------------------------------
        .def    RESET                   ; Export program entry-point to
                                        ; make it known to linker.
;-------------------------------------------------------------------------------
        .text                           ; Assemble into program memory.
        .retain                         ; Override ELF conditional linking
                                        ; and retain current section.
        .retainrefs                     ; And retain any sections that have
                                        ; references to current section.
;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
;-------------------------------------------------------------------------------
; Setup LED1 (used in original timer setup, not needed for seven-segment display)
;-------------------------------------------------------------------------------
; Setup Timer B0
;-------------------------------------------------------------------------------
        mov.w   #TBCLR, &TB0CTL         ; Clear Timer & Dividers
        bis.w   #TBSSEL__ACLK, &TB0CTL  ; Select ACLK as Timer Source
        bis.w   #MC__UP, &TB0CTL        ; Choose UP Counting
        mov.w   #32768, &TB0CCR0        ; Initialize CCR0 to 32768
        bis.w   #CCIE, &TB0CCTL0        ; Enable Capture/Compare IRQ
        bic.w   #CCIFG, &TB0CCTL0       ; Clear Interrupt Flag
        nop
        bis.w   #GIE, SR                ; Enable Maskable Interrupts
        nop
;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------
main:
        bic.b #0FFh, &P1OUT             ; Clear P1 output
```

```
        bis.b #0FFh, &P1DIR             ; Set P1 direction to output
        bic.b #0FFh, &P2OUT             ; Clear P2 output
        bis.b #03h, &P2DIR              ; Set P2.0 and P2.1 to output
        bic.b #LOCKLPM5, &PM5CTL0     ; Unlock GPIO
        mov.b #0, R5                    ; Start display 2 at 0
        mov.b #0, R7                    ; Start display 1 at 0

loop:
        call #DisplayDigits             ; Display current values on displays
        jmp loop                        ; Loop forever

DisplayDigits:
        mov.b digits(R5), R6            ; Load digit pattern into R6
        mov.b #01h, &P2OUT              ; Enable Display 2 only
        mov.b R6, &P1OUT                ; Send pattern to P1
        call #DelayLong                 ; Long delay
        mov.b digits(R7), R6            ; Load digit pattern into R6
        mov.b #02h, &P2OUT              ; Enable Display 1 only
        mov.b R6, &P1OUT                ; Send pattern to P1
        call #DelayLong                 ; Long delay
        ret

DelayLong:
        mov.w   #0FFh, R4               ; Load long delay
d_loop_long:
        dec.w   R4
        jnz     d_loop_long
        ret

digits:
        .byte 0x3F                      ; Digit '0'
        .byte 0x06                      ; Digit '1'
        .byte 0x5B                      ; Digit '2'
        .byte 0x4F                      ; Digit '3'
        .byte 0x66                      ; Digit '4'
        .byte 0x6D                      ; Digit '5'
        .byte 0x7D                      ; Digit '6'
        .byte 0x07                      ; Digit '7'
        .byte 0x7F                      ; Digit '8'
        .byte 0x6F                      ; Digit '9'

;-------------------------------------------------------------------------------
; Interrupt Service Routine for Timer B0 CCR0
;-------------------------------------------------------------------------------
```

```
ISR_TB0_CCR0:
        bic.w   #CCIFG, &TB0CCTL0       ; Clear the CCIFG flag
        inc.b   R5                      ; Increment display 2 register
        cmp.b   #10, R5                 ; Compare with value 10
        jne     no_tens_increment       ; If not equal, jump
        mov.b   #0, R5                  ; Else, start back at 0
        inc.b   R7                      ; Increment display 1 register
        cmp.b   #6, R7                  ; Compare with value 6
        jne     no_tens_increment       ; Jump if display 1 != 6
        mov.b   #0, R7                  ; Else, refresh to 0
no_tens_increment:
        reti                            ; Return from interrupt

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
        .sect   ".reset"                ; MSP430 RESET Vector
        .short  RESET
        .sect   ".int43"                ; Timer0_B3 CCIFG0 Vector
        .short  ISR_TB0_CCR0

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
        .global __STACK_END
        .sect   .stack
```