Lin, Jake
Stony Brook ID 114565529
Lab 11
PWM_100_Levels_TCA0
Section L04
Bench #10

```
; PWM_100_levels_TCA0.asm
;
; Created: 11/5/2023 7:04:45 PM
; Author : Jake
;

 .CSEG
 ; interrupt vector table, with several 'safety' stubs
 rjmp RESET ;Reset/Cold start vector
  .equ PERIOD_EXAMPLE_VALUE = 100       ; 1% resolution for duty cycle setting
 reti ;External Intr0 vector
 reti ;External Intr1 vector

 .org PORTE_PORT_vect
jmp take_input      ;vector for all PORTE pin change IRQs

RESET:
 sbi VPORTA_DIR, 7 ; set PA7 = output.
 sbi VPORTA_OUT, 7 ; set /SS of DOG LCD = 1 (Deselected)
 sbi VPORTD_DIR, 0
 sbi VPORTD_DIR, 1
 rcall init_lcd_dog ; init display, using SPI serial interface

ldi r16, 0x00;load 0s
 out VPORTC_DIR, r16;make portc into input
 cbi VPORTE_DIR, 0; make port b at bit 5 into a input
 ;used r16
 ldi r16, 0x02
 sts PORTE_PIN0CTRL, r16; make pin 0 in porte detect rising edge

 ;write the prelimaries
initial:
line1:
ldi ZL, LOW(text1<<1); makes z point to the first text at position 1
ldi ZH, HIGH(text1<<1)
rcall load_msg;its the initial display being executed
rcall update_lcd_dog; print the skeleton


;Route WO0 to PD0 instead of its default pin PA0
    ldi r16, 0x03       ;mux TCA0 WO0 to PD0
    sts PORTMUX_TCAROUTEA, r16

    ;Set CTRLB to use CMP0 and single slope PWM
    ldi r16, TCA_SINGLE_CMP0EN_bm | TCA_SINGLE_WGMODE_SINGLESLOPE_gc ;CMP0EN and
      single slope PWM
    sts TCA0_SINGLE_CTRLB, r16

    ldi r16, LOW(PERIOD_EXAMPLE_VALUE)      ;set the period
    sts TCA0_SINGLE_PER, r16
    ldi r16, HIGH(PERIOD_EXAMPLE_VALUE)
    sts TCA0_SINGLE_PER + 1, r16
    ;Set clock and start timer/counter TCA0

    ldi r16, 0
    ldi r18,0; this is used to ensure we dont type more than 3 digits
    sei

main_loop:
rjmp main_loop


;subroutines start below
 .include "lcd_dog_asm_driver_avr128.inc"

;load message subroutine
load_msg:; this is only called when trying to initialize the lcd
ldi YL, LOW(dsp_buff_1); make y point to the beginning of sram
ldi YH, HIGH(dsp_buff_1)
lpm r16, Z+; take the first value that z is pointing to and load to r16
cpi r16, 1; compare to 1
breq load_msg2; if it is 1, then begin reading the first line
load_msg2:
lpm r16, Z+; load whats following, 1,2, or3 to r16
cpi r16, 9; compare if that value is 9, which is shouldnt
breq add_space; if it is 9 then you have reached the end of the line and you read next
   line
st Y+, r16; store all the values from the line into the sram
rjmp load_msg2; go back load_ms2 if the line has not finished reading
add_space:
ldi r16, 32
ldi r17, ' '
addspace:
st Y+, r17
dec r16
cpi r16, 0
brne addspace
ret


take_input:
push r16
push r19
push r17
cli ;dont allow new values to influence interrupt
ldi r16, PORT_INT0_bm; gets the bit mask
sts PORTE_INTFLAGS, r16; makes the bit in the flag register a 0 again
 in r16, VPORTC_IN

 cpi r18, 4
 brne PC+8
 cpi r16, 0b00110001
 breq clear
 pop r17
 pop r19
 pop r16
 sei
 reti

 lsr r16 ;move r16 value to the right four decimals
 lsr r16
 lsr r16
 lsr r16
 rcall retrieve_data; get the equivalent from the tablelookup

;make sure r16 isn't any of these characters
cpi r16, $42
breq return_mainloop
cpi r16, $46
breq return_mainloop
cpi r16, $45
breq return_mainloop
cpi r16, $44
breq return_mainloop
jmp PC+4
return_mainloop:
sei
reti

;is it enter?
cpi r16, $43 ; is the key code enter?
brne is_clear; if it isnt then is it clear?
pop r19
pop r16
cpi r18, 3
brne return_mainloop
inc r18
ldi YL, LOW(dsp_buff_1+12); make y point to the third digit
ldi YH, HIGH(dsp_buff_1+12)
ld r16, Y+; loads the left position Y
ld r17, Y+; move to the next
ld r19, Y
subi r16, $30; finds the decimal equivalent
subi r17, $30
subi r19, $30
cpi r16, 2
brge clear
ldi r20, $64; multiplies by 100 the left most value
mul r16, r20
mov r16, r0
ldi r20, $0A; multiplies by 10 the middle value
mul r17, r20
mov r17, r0
add r16, r17
add r16, r19; adds the values together
```

```
cpi r16, 101
brge clear
mov r19, r16
pop r17

sts TCA0_SINGLE_CMP0BUF, r16    ;use TCA0_SINGLE_CMP0BUF for double buffering
ldi r16, 0
sts TCA0_SINGLE_CMP0BUF + 1, r16
ldi r16, TCA_SINGLE_CLKSEL_DIV64_gc | TCA_SINGLE_ENABLE_bm
sts TCA0_SINGLE_CTRLA, r16

sei
reti


is_clear: ;is it clear?
cpi r16, $41; is the key code clear?
brne is_char; if it isnt then it has to be a character
clear:
ldi r16, '0'; load 0 in
st Y, r16; store 0 in the sram
sbiw Y,1
st Y, r16; store another _ afterwards
sbiw Y,1
st Y, r16; store a 0 at the last position
ldi r18,0
rcall update_lcd_dog; update the lcd
pop r19
pop r16
pop r17
sei
reti; clear completed


is_char: ;its a random character
cpi r18, 3; dont allow new characters to be inputted if theres 3 digits already
breq return_mainloop
ldi YL, LOW(dsp_buff_1+13); make Y point to the beginning of sram
ldi YH, HIGH(dsp_buff_1+13)
ld r17, Y
sbiw Y, 1
st Y, r17
adiw Y, 2
ld r17, Y
sbiw Y, 1
st Y+, r17
st Y, r16

rcall update_lcd_dog; update the lcd
inc r18
pop r19
pop r16
pop r17
sei
reti ; return to mainloop

 retrieve_data:
andi r16, 0x0F; mask the values so that only the first four bits are contained
cpi r16, 16 ;result 0-15?
brlo lookup ;yes look up the keycode
 ldi r16, 0
 clc
 ret lookup:
 ldi ZH, high (segtable*2)
ldi ZL, low(segtable*2)
 ldi r17, $00
 add ZL, r16
 adc ZH, r17
 lpm r16, Z
 sec
ret

;A is clear, C is enter
//segtable: .db $01, $04,$07,$0A,$02,$05,$08,$00,$03,$06,$09,$0B,$0F,$0E,$0D,$0C
text1: .db 1, "Duty Cycle= 000%", 9
segtable: .db $31, $34,$37,$41,$32,$35,$38,$30,$33,$36,$39,$42,$46,$45,$44,$43;
```