Lin, Jake
Stony Brook ID 114565529
Lab 9
Software Width Pulse Modulation
Section L04
Bench #10

```
;
; period_factor.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
 reti ;External Intr0 vector
 reti ;External Intr1 vector
RESET:

 sbi VPORTA_DIR, 7 ; set PA7 = output.
 sbi VPORTA_OUT, 7 ; set /SS of DOG LCD = 1 (Deselected)
 sbi VPORTD_DIR, 0
 rcall init_lcd_dog ; init display, using SPI serial interface
ldi r16, 0x00;load 0s
 out VPORTC_DIR, r16;make portc into input
 cbi VPORTB_DIR, 5; make port b at bit 5 into a input
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
ldi r18,0; this is used to ensure we dont type more than 3 digits
ldi r21, 0
loop:
rcall poll; polls the flip flop to see when to accept new key code
rcall retrieve_data; returns the key value that you pressed in r16
;make sure r16 isn't any of these characters
cpi r16, $42
breq loop
cpi r16, $46
breq loop
cpi r16, $45
breq loop
cpi r16, $44
breq loop
;is it enter?
cpi r16, $43 ; is the key code enter?
brne is_clear; if it isnt then is it clear?
ldi r18, 0; allows you to enter 3 digits again
inc r21
cpi r21, 2; sees if is second enter

brne loop; if its not then you can poll for new values
ldi YL, LOW(dsp_buff_1); make y point to the third digit
ldi YH, HIGH(dsp_buff_1)
;does the first line
adiw Y, 12; makes y point to the first 0
```

```
ld r16, Y+; loads the first value there
ld r17, Y+; second
ld r19, Y ;rightmost value loaded
subi r16, $30; finds their decimal equivalent
subi r17, $30
subi r19, $30
cpi r16, 2; is the first char 2 or greater?
brge clear1; if so then you must clear the whole display and start over
ldi r20, $64; if not then load 100 into r20
mul r16, r20; multiply first term by 100
mov r16, r0; moves the first term into r16
ldi r20, $0A; loads a ten
mul r17, r20; multiplies ten
mov r17, r0; load the product into r17
add r16, r17
add r16, r19
cpi r16, 101; see if the result is greater than 100
brlo PC+2; if so then you must clear
rjmp clear1
;does the next line, r16 must be fixed
adiw Y, 15
ld r21, Y+; load the left most value
ld r17, Y+; load the next value to the right
ld r19, Y; load the next value to the right
subi r21, $30; find their decimal equivalents
subi r17, $30
subi r19, $30
cpi r21, 2
brge clear1
mul r21, r20; do the same steps to get the multiplying factor
mov r21, r0
mul r17, r20
mov r17, r0
add r19, r17
add r19, r21
cpi r19, 101; see if the result is greater than 100
brlo PC+2; if so then you must clear

rjmp clear1
rcall change_led
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
cpi r20, 1; is it the second line?
breq clear2

clear1:; clears the first line
ldi YL, LOW(dsp_buff_1); make Y point to the beginning of sram
ldi YH, HIGH(dsp_buff_1)
adiw Y, 14
clr r21;
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
clear2:; clears the second line
ldi YL, LOW(dsp_buff_1); make Y point to the beginning of sram
ldi YH, HIGH(dsp_buff_1)
adiw Y, 31
ldi r17, '1'
st Y, r17
ldi r17, '0'
sbiw Y, 1
st Y, r17
sbiw Y, 1
st Y, r17
rcall update_lcd_dog; update the lcd
ldi r18,0
rjmp loop; clear completed
is_char: ;its a random character
cpi r18, 3; dont allow new characters to be inputted if theres 3 digits already
breq go_to_beg
ldi YL, LOW(dsp_buff_1); make Y point to the beginning of sram
ldi YH, HIGH(dsp_buff_1)
cpi r21, 0; compares if the row is the first
breq PC+2; if so then you skip so that you increment Y less
adiw Y, 17
adiw Y, 13
ld r17, Y; load the right most value from Y
sbiw Y, 1; move Y down
st Y, r17; store the right most value to the middle one and so on
adiw Y, 2
cpi r18, 0; takes care of the 001 case

brne PC+2
st Y, r17
ld r17, Y
sbiw Y, 1
st Y+, r17
st Y, r16
rcall update_lcd_dog; update the lcd
inc r18
go_to_beg:
jmp loop; stored and shifed the text completed
 .include "lcd_dog_asm_driver_avr128.inc"
;10000
change_led:
ldi r17, 100; load 100 into r17
sub r17, r16; subtract 100 by r16 to get low count inserted into r17
;this is the low count
mul r17, r19; multiplies the low count by what ever the factor is
```

```
mov r17, r0;
mov r18, r1;
;this is the high count
mul r16, r19; multiplies the high count by the factor
mov r16, r0
mov r19, r1
mov r20, r16
mov r23, r19
mov r21, r17
mov r22, r18


again:
;push r16; high count right bits
;push r19; high count left bits
cpi r19, 0; check if the first half is 0
brne PC+3;
cpi r16, 0
breq pwm_low; if it is that means that the duty cycle is 0, the second half
sbi VPORTD_OUT, 0; if it isn't that means the there is some positive pulse to be made,
   so first ; do that to the output pin
pwm_high:; keep outputting the 1

subi r16, 1; subtracts 1 from the first 4bits
sbci r19, 0; subtract from the higher register if there is a carry needed
cpi r16, 0; sees if the two registers equal 0, set Z to 1
brne PC+2
cpi r19, 0
brne pwm_high;
;pop r19
;pop r16
;push r17; low count right bits
;push r18; low count left bits
cpi r18, 0
brne PC+3
cpi r17, 0; finished with the high, check if r17 is 0,
breq again; if it is , then there is no low
cbi VPORTD_OUT, 0; if it isnt, there is some low
pwm_low:
subi r17, 1; subtracts 1 from the first 4bits
sbci r18, 0; subtract from the higher register if there is a carry needed
cpi r17, 0; sees if the two registers equal 0, set Z to 1
brne PC+2
cpi r18, 0
brne pwm_low;
;pop r18
;pop r17
in r16, VPORTC_IN
andi r16, 0b11110000; used to compare if the user types in clear midway
cpi r16, 0b00110000
breq PC+6;; go to clear the code and enable re-enter
```

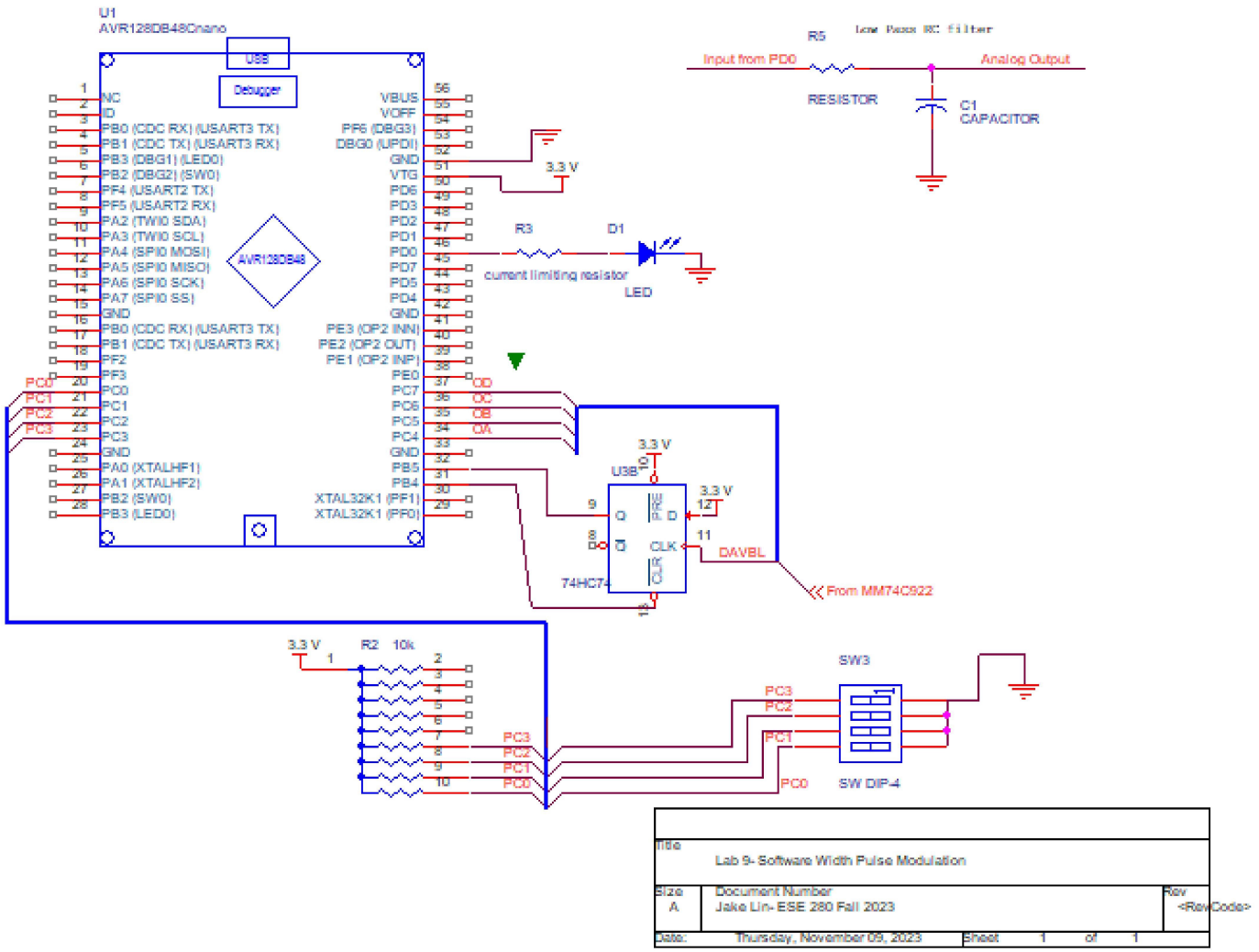
```
mov r16, r20; move the original values back into the registers
mov r19, r23
mov r17, r21
mov r18, r22
rjmp again

jmp clear1

;
;load message subroutine
load_msg:; this is 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 end_load; 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
end_load:
ret
;poll subroutine
poll:
 sbi VPORTB_DIR, 4; make port b at bit 4 into an output
 cbi VPORTB_OUT, 4; make port b at bit 4 into 0 to clear first
 sbi VPORTB_OUT, 4; make the bit 1 to deassert
 find_0:
 sbic VPORTB_IN, 5; skip if you found a 0
 rjmp find_0 ; loop if not
 find_1:
 sbis VPORTB_IN, 5
 rjmp find_1;found 1 so now the data from the keypad is ready and can be accepted from↵
     flip flop
 in r16, VPORTC_IN
 lsr r16 ;move r16 value to the right four decimals
 lsr r16
 lsr r16
 lsr r16
 ret
;retrieve data subroutine
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%T Multiply = 001                ", 9
segtable: .db $31, $34,$37,$41,$32,$35,$38,$30,$33,$36,$39,$42,$46,$45,$44,$43;
```



Title: Lab 9- Software Width Pulse Modulation		
Size: A	Document Number: Jake Lin- ESE 280 Fall 2023	Rev: <RevCode>
Date: Thursday, November 09, 2023	Sheet 1 of 1	