REQUIREMENTS NOT MET

N/A

PROBLEMS ENCOUNTERED

1. Prelab question 9 gave me too long of a time period at first, this was because I was using "CLK_PSADIV_2_gc" instead of "TC_CLKSEL_DIV2_gc"
2. Part 4 was a bit tough because my counter was stuck in an infinite loop. This is because I did not use "TC0_OVFIF_bp", but instead, used my own definition for bit 1.
3. Generating bounce in the tactile switches for part 4 was a bit tough because the switches were still new, but after wearing them down a bit by pressing them over and over, I finally got them to bounce a bit.

FUTURE WORK/APPLICATIONS

The topics in this lab can be used to implement timers, develop software using input and output ports, and how to adjust clock cycles for measuring time.

PRE-LAB EXERCISES

i. Which configuration register allows the utilization of an I/O port pin configured as an input? Which configuration registers allow the utilization of an I/O port pin configured as an output?

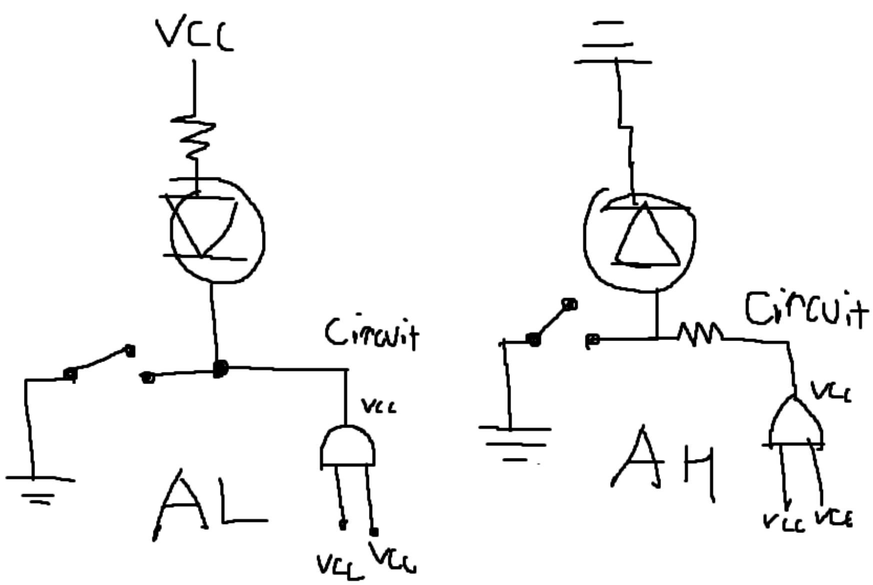
> **The "in" register of a port allows you to read the port as input.**
> **The "out" register of a port allows you to output vales at certain pins.**
> **The "outset" register writes a "high" signal to whichever bits are set to "1"**
> **The "outclr" register writes a "low" signal to whichever bits are set to "1"**
> **The "outtgl" register toggles a signal to whichever bits are set to "1"**

ii. What is the purpose of the SET/CLR/TGL variants of the DIR and OUT registers?

> **The purpose is to allow you to adjust the signal levels of individual pins without Affecting other pins**

iii. Are the LEDs on the OOTB Switch & LED Backpack active-high, or active-low? Draw a schematic diagram for a single LED circuit with the same activation level used on the backpack, as well as one with the opposite activation level. Also, draw a schematic diagram for a single-pole, single-throw (SPST) switch circuit, using the same pull-up or pull-down resistor condition utilized on the backpack, as well as another switch circuit using the opposite configuration.

> **Led's are active-low.**



iv. Which I/O ports are utilized for the DIP switches and LEDs on the OOTB Switch & LED Backpack?

> **The led circuits utilize port C**
> **The switch circuit utilizes port A**

v. Would it be possible to interface the OOTB µPAD with an external input device consisting of 24 inputs? If so, describe how many I/O ports would be necessary. If not, explain why.

> **The OOTB has 3 ports with 8 pins each. All ports are configurable as inputs. Therefore, it should be possible to connect a 24-pin input device.**

vi. Assuming a system clock frequency of 2 MHz, a prescaler value of 8, and a desired period of 72 ms, calculate a theoretically-corresponding timer/counter period value two separate times: once using a form of dimensional analysis, providing explanation(s) when appropriate, and another time using the general formula provided within The Most Common Use Case for Timer/Counters.

"cycle" = system clock cycle
"tick" = counter cycle

$$SCF = \frac{2\times10^6 \text{ cycle}}{\text{second}}$$

$$Pre = \frac{8 \text{ cycles}}{\text{Tick}}$$

$$D = 72 \text{ ms}$$

$$\frac{2\times10^6 \text{ cycle}}{\text{second}} \rightarrow \frac{2\times10^6 \text{ cycle}}{10^3 \text{ ms}} = \frac{2\times10^4 \text{ cycle}}{10 \text{ ms}}$$

$$\frac{2\times10^4 \text{ cycle}}{10 \text{ ms}} \times \frac{72}{72} \Rightarrow \frac{144\times10^4 \cancel{\text{cycles}}}{720 \text{ ms}} \times \frac{1 \text{ Tick}}{8 \cancel{\text{cycles}}}$$

$$\frac{18\times10^4 \text{ ticks}}{720 \text{ ms}} = \boxed{\frac{18000 \text{ Ticks}}{72 \text{ ms}}}$$

$$Per = 72\left(\frac{2\times10^6 \text{ cycles}}{10^3 \text{ ms}} \Big/ \frac{8 \text{ cycles}}{\text{Tick}}\right)$$

$$= 72\left(\frac{2\times10^6 \cdot \text{Ticks}}{8\cdot10^3 \text{ ms}}\right) = 9\left(\frac{2\times10^6 \text{ Ticks}}{10^3 \text{ ms}}\right)$$

$$= \frac{18\times10^4 \text{ Ticks}}{10 \text{ ms}} = \boxed{\frac{18000 \text{ Ticks}}{10 \text{ ms}}}$$

vii. Assuming a system clock frequency of 2 MHz, is a period of two seconds achievable when using a 16-bit timer/counter prescaler value of one? If not, determine if there exists any prescaler value that allows for this period under the assumed circumstances, and if there does, list such a value.

n bit count field

Period P

$P_{TC} = MP$

$$P = \frac{1\text{ Tick}}{1\text{ cycle}} \times \frac{2\times10^6\text{ cycles}}{\text{second}} = \frac{2\times10^6\text{ Ticks}}{\text{Second}}$$

$T = 2^n P$

$$2\text{ seconds} = 2^n\text{ ticks}\left(\frac{\text{second}}{2\times10^6\text{ ticks}}\right)$$

$$4\ \cancel{\text{second}} = 2^n\ \cancel{\text{ticks}}\left(\frac{\cancel{\text{second}}}{10^6\ \cancel{\text{ticks}}}\right)$$

$4\times10^6 = 2^n$

$n = 22$ $\boxed{\text{No}}$

$$P = \frac{1\text{ Tick}}{n\text{ cycle}} \times \frac{2\times10^6\text{ cycles}}{\text{second}} = \frac{2\times10^6\text{ Ticks}}{n\text{ second}}$$

$$2\text{ seconds} = 2^{16}\text{ ticks}\left(\frac{n\text{ second}}{2\times10^6\text{ ticks}}\right)$$

$61.04 = \boxed{n \Rightarrow 62}$

$$T = 2^{16}\text{ ticks} \times \left(\frac{\text{second}}{32258.01\text{ ticks}}\right)$$

$$P = \frac{1\text{ tick}}{62\text{ cycles}}\left(\frac{2\times10^6\text{ cycles}}{\text{sec}}\right) = \frac{32258.01\text{ ticks}}{\text{second}}$$

viii. What is the maximum time value (to the nearest millisecond) representable by a timer/counter, if the relevant system clock frequency is 2 MHz? What about for a system clock frequency of 32.768 kHz?

$$T = 2^{16}\,\text{ticks}\left(\frac{1\,\text{sec}}{2\times10^{6}\,\text{ticks}}\right)$$

$$\underline{T = 33\,\text{ms}}$$

$$T = 2^{16}\,\text{Ticks}\left(\frac{1\,\text{sec}}{32768\,\text{ticks}}\right)$$

$$\underline{T = 2\,\text{seconds}}$$

ix. Create an assembly program to perform the same procedure as in § 3.2 but utilize a prescaler value of two. Perform everything else described in the section for this new context, i.e., experimentally determine which whole-number digital period value provides a corresponding period with the least amount of error, provide an appropriate screenshot of the relevant waveform with the minimal amount of error, including its precise frequency, and provide within the caption of the relevant screenshot the whole-number value that resulted in a minimal amount of error. Finally, describe and explain why there may be any differences between the two contexts, i.e., between using a prescaler value of sixteen, the value in exercise vi, and a prescaler value of two.

The program is in section 3, labeled "prelab question number 9"

x. Create an assembly program to keep track of elapsing minutes with a timer/counter, i.e., design a "watch" that only has a "minute-hand". (Hint: Instead of attempting to configure the period of the timer/counter to directly correspond to sixty seconds, configure the period to correspond to one second, and then keep track of how many times this timer/counter overflows [or underflows, if you wish to configure the timer/counter to count down].)

The program is in section 3, labeled "prelab question number 10"

xi. It is stated above that, in the relevant context, it should not be necessary to debounce (nor wait for the release of) either of tactile switch S1 or tactile switch S2 located on the OOTB MB. Why is this so?

Because whether our program goes back to the "edit" loop is not dependent on whether the button is released or not, bouncing does not matter.

xii. Provide a scenario in which the above program would experience unintended behavior due to tactile switch bouncing.

Scenario:
You press the "edit" button to add a frame to your animation

Because of bouncing, the frame is added multiple times when you meant to add it once.

PSEUDOCODE/FLOWCHARTS

SECTION 1

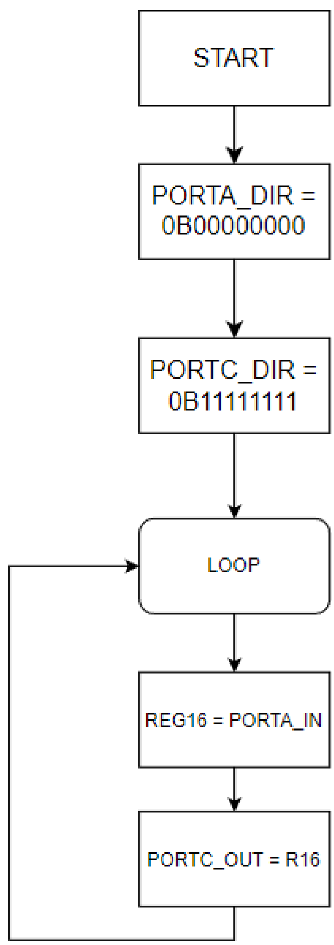


Figure 1:Flowchart for "lab2_1.asm". This copies the switch circuit register values to the output registers for the LED's

SECTION 2

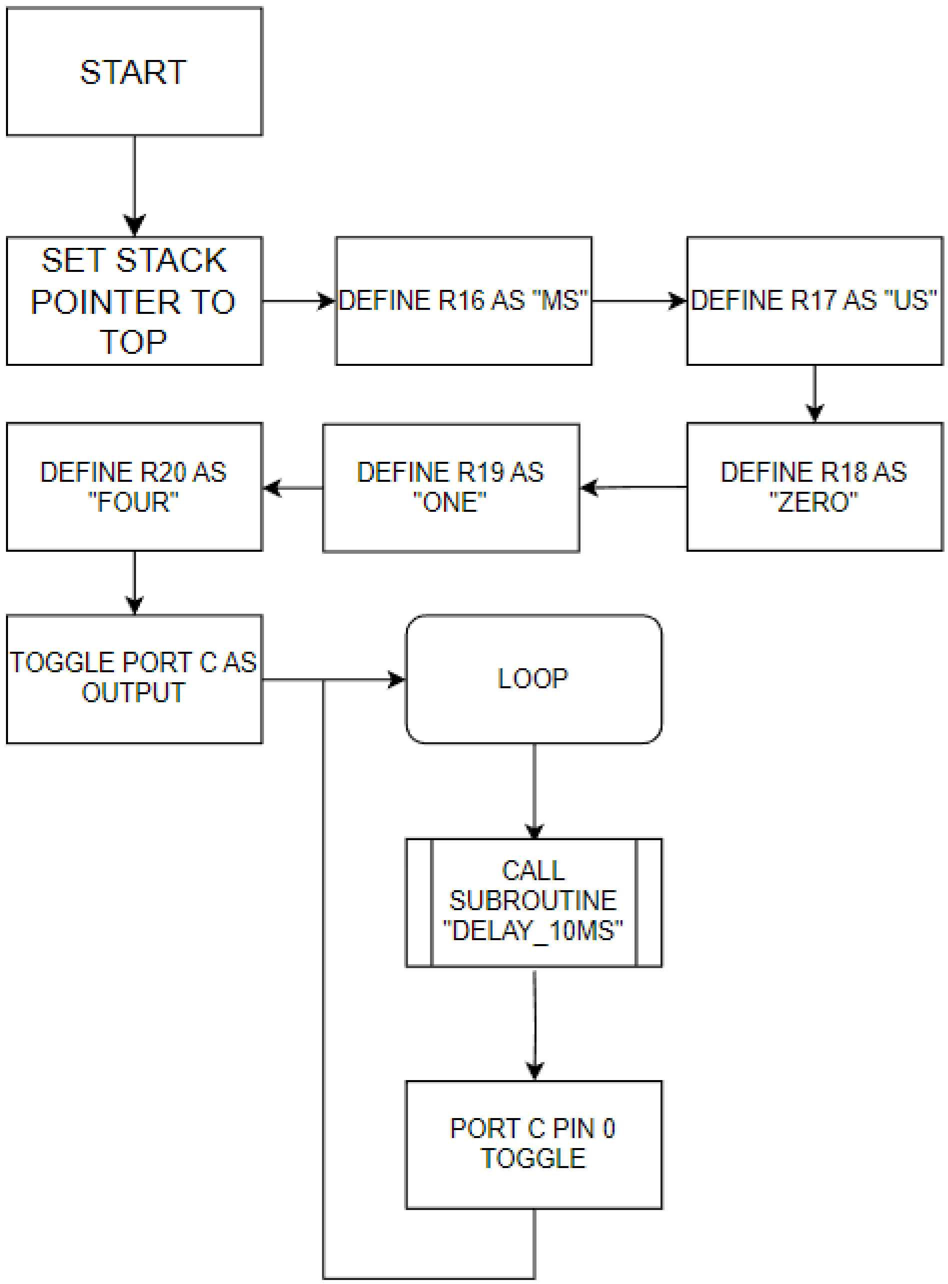


Figure 2: Flowchart for "lab2_2.asm" with "delay_10ms" being the only subroutine. This toggles pin 0 of port C every 10ms, giving us a 10ms PWM signal.

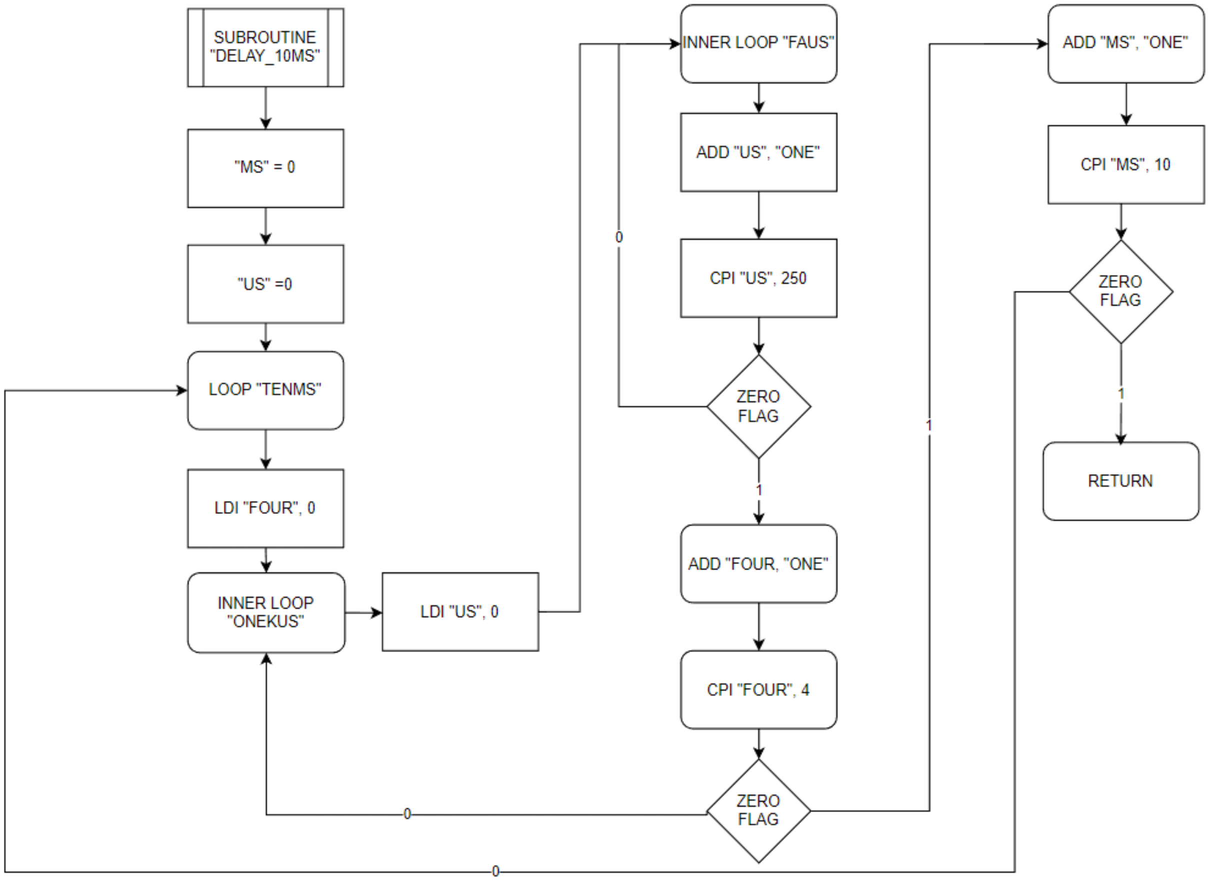


Figure 3: Flowchart for "delay_10ms" subroutine.
"FAUS" creates a delay of 250us by executing 500 single cycle instructions.
"ONEKUS" creates a 1000us delay by running "FAUS" 4 times.
"TENMS" creates a 10000us delay by running "ONEKUS" 10 times.
Giving us a delay of 10ms.

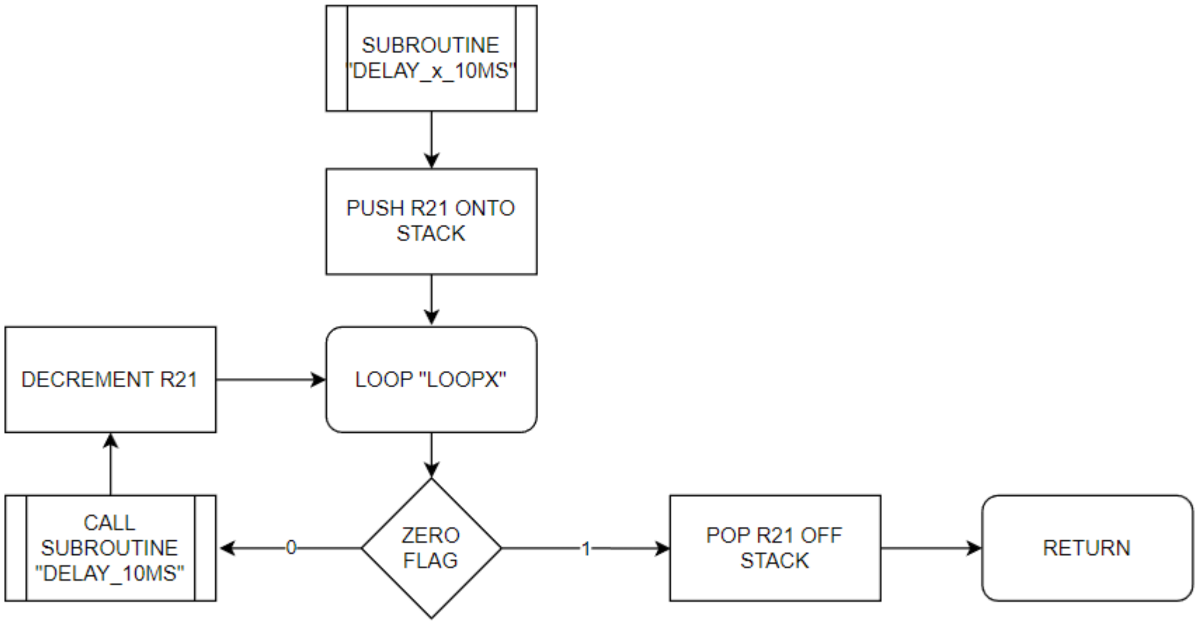


Figure 4: Flowchart for "delay_X_10ms".
R21 is the multiple of 10ms delays.

SECTION 3

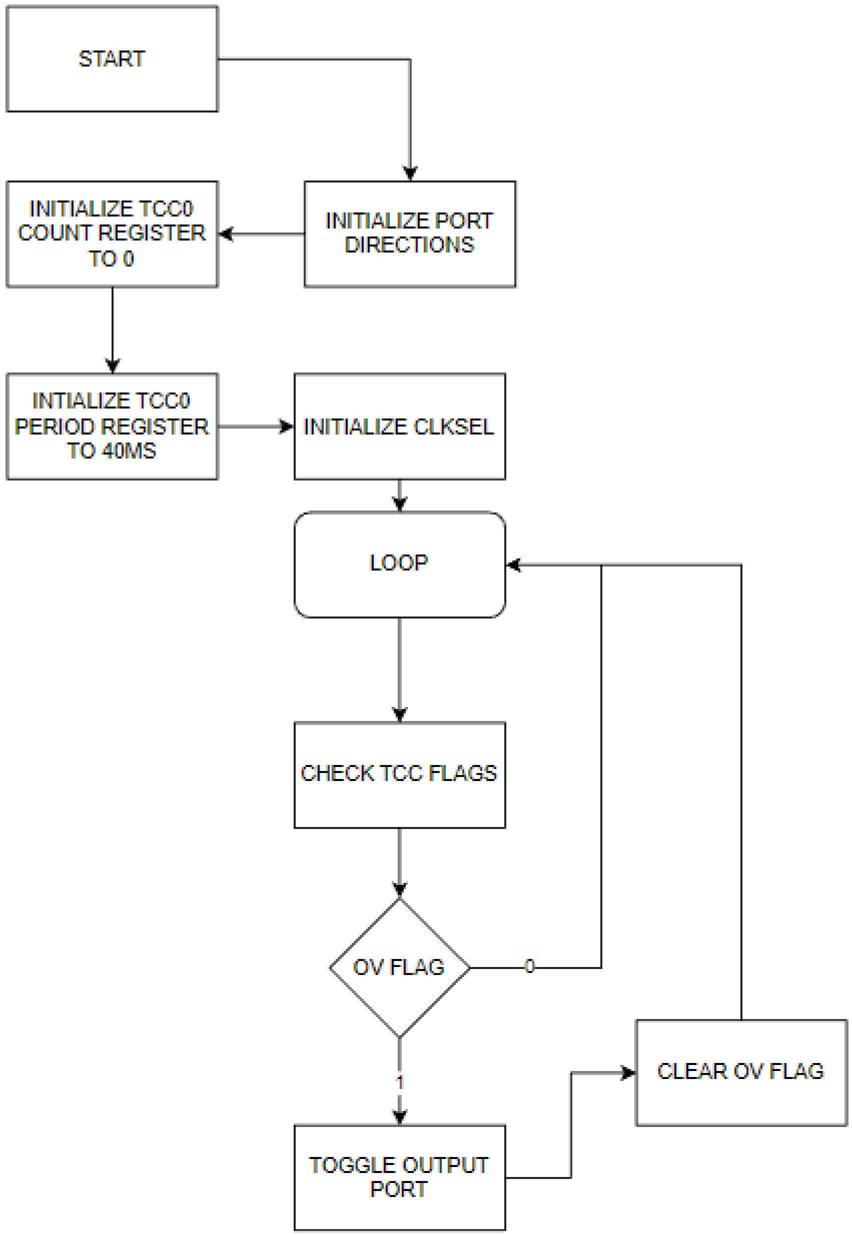


Figure 5: flowchart for "lab2_prelabquestionix.asm".
This triggers an output signal every 40ms
by checking the overflow flag on timer/counter number 0

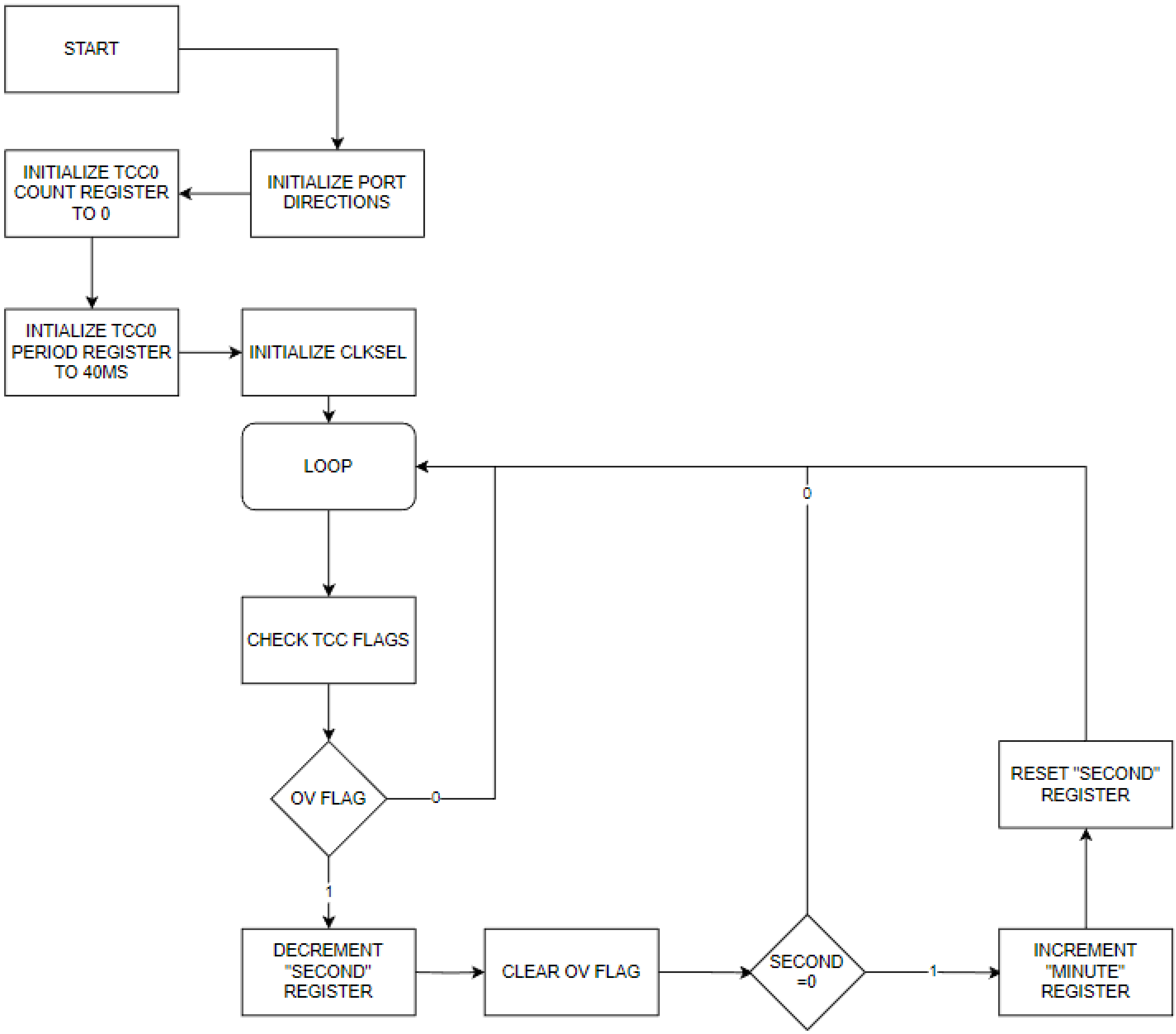


Figure 6: flowchart for "lab2_prelabquestionx.asm". This counts how many minutes have elapsed since program began execution.

Section 4

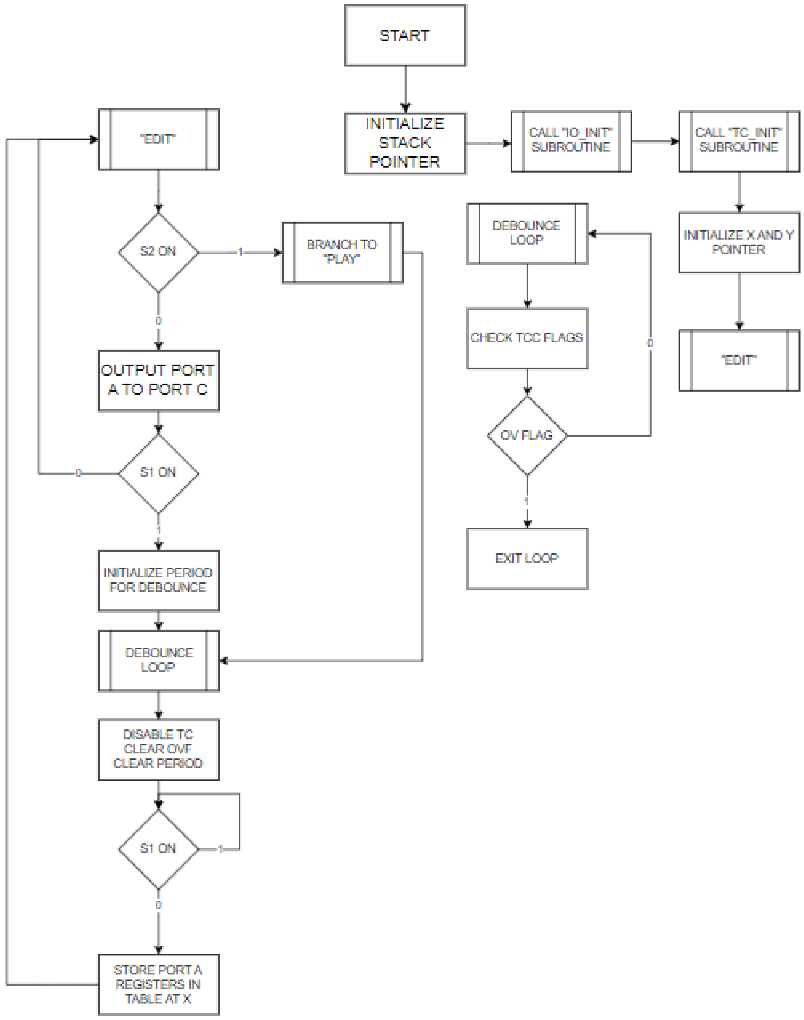


Figure 7: flowchart for "lab2_4.asm". These flowcharts depict the behavior for the debounce loop and the edit loop

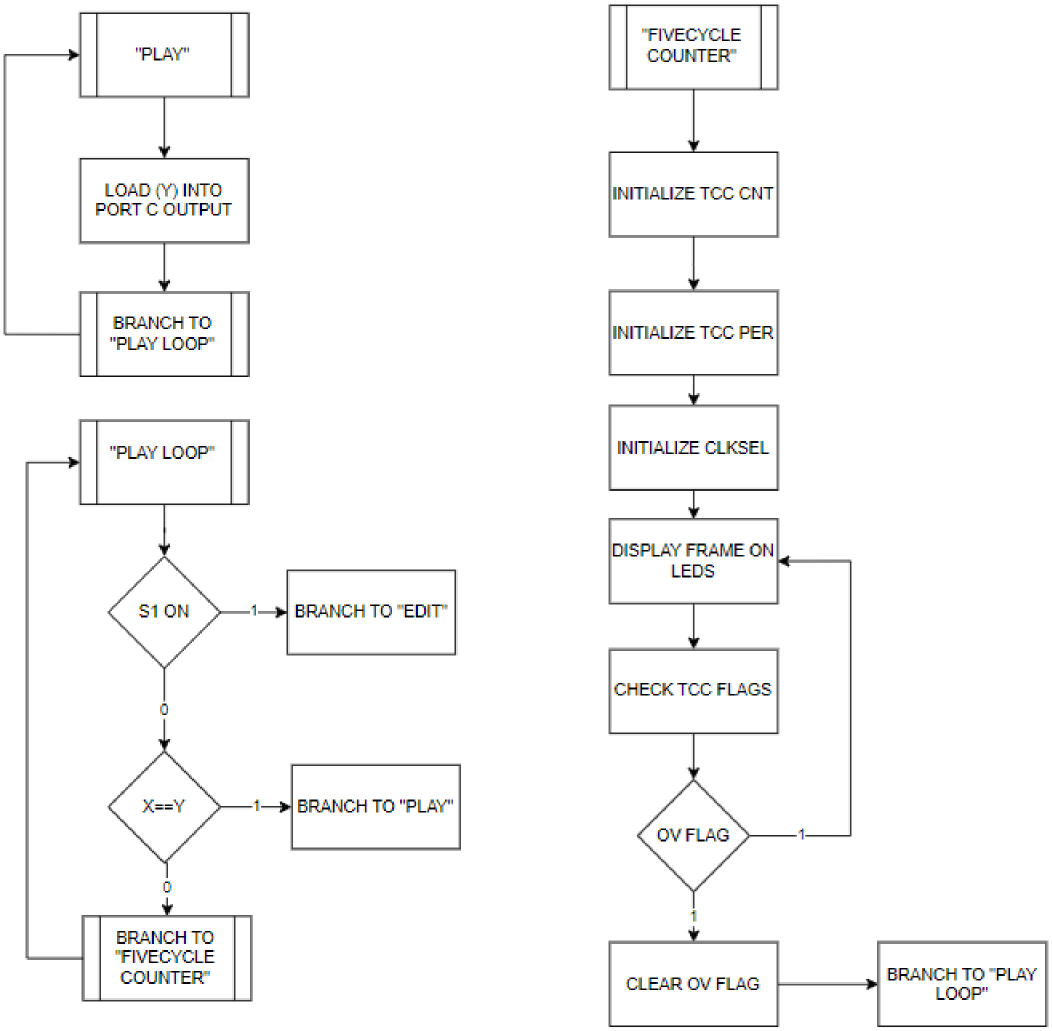


Figure 8: Second flowchart for "lab2_4.asm". These flowcharts depict the behavior of the play loop and the 5-cycle timer.

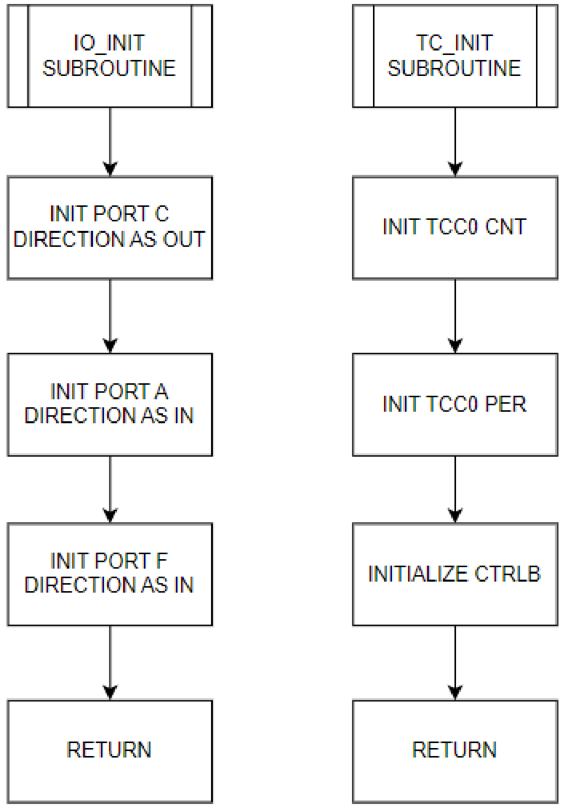


Figure 9: Third flowchart for "lab2_4.asm". These flowcharts depict the behavior of the IO initialization subroutine and the TC initialization subroutine.

PROGRAM CODE

SECTION 1

```
;Lab 2, Section 1
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: Allows control of LED's through switch circuit
;***************INCLUDES*************************************
.include "ATxmega128a1udef.inc"
;***************END OF INCLUDES******************************

;*********************************EQUATES********************************
.EQU INPUT = 0B00000000
.EQU OUTPUT = 0B11111111
;*******************************END OF EQUATES*******************************

;*********************************DEFS********************************
;*******************************END OF DEFS*******************************

;***********MEMORY CONFIGURATION*************************

;***********END OF MEMORY CONFIGURATION***************
;***********MAIN PROGRAM*******************************
.CSEG
.org 0x0100
MAIN:
;set port directions
LDI R16, INPUT
STS PORTA_DIR , R16
LDI R16, OUTPUT
STS PORTC_DIR , R16

;loop for actual led and switch circuits
LOOP:
       ;copy load value from switch registers into led registers
       LDS R16, PORTA_IN
       STS PORTC_OUT,R16
RJMP LOOP
;***********END MAIN PROGRAM******************************
```

Section 2

Original code to create software delay of 10ms

```
;Lab 2, Section 2
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: Implements software delays
;***************INCLUDES*************************************
.include "ATxmega128a1udef.inc"
;***************END OF INCLUDES******************************

;*********************************EQUATES********************************
.EQU sramend = 0x3fff ;top of stack
.EQU srambegin = 0x2000 ;bottom of stack
.EQU input = 0b00000000
.EQU output = 0b11111111
;*******************************END OF EQUATES*******************************

;*********************************DEFS*******************************
.DEF ms_r16 = r16
.DEF us_r17 = r17
.DEF zero_r18 = r18
.DEF one_r19 = r19
.DEF four_r20 = r20
;*******************************END OF DEFS*******************************

;***********MAIN PROGRAM*******************************
.CSEG
.org 0x0100
MAIN:
;initialize stack pointer
ldi r16, low(sramend)
out CPU_SPL, r16
ldi r16, high(sramend)
out CPU_SPH, r16
;set port directions
LDI R22, output
STS PORTC_DIR , R22
;initialize registers
ldi zero_r18,0
ldi one_r19,1
ldi four_r20,4
;loop to call subroutine
LOOP:
	rcall delay_10ms
	STS PORTC_OUTTGL,R22
RJMP LOOP
;***********END MAIN PROGRAM*******************************

;********************SUBROUTINES*************************************
; Subroutine Name: delay_10ms
; performs a series of instructions for 10ms
; Inputs: none
; Ouputs: none
; Affected: r16, r17,r19,r20
delay_10ms:
	ldi ms_r16,0
	ldi us_r17,0
```

```
        tenms:
                ldi four_r20, 0
                        ;1ms
                        onekus:
                                ldi us_r17,0
                                        ;250us
                                        faus:
                                                add us_r17,one_r19
                                                cpi us_r17,250
                                        brne faus
                                add four_r20,one_r19
                                cpi four_r20,4
                        ;branch if 1ms
                        brne onekus
                add ms_r16,one_r19
                ;branch if 10 ms
                cpi ms_r16,10
        brne tenms
ret
```

Code after adjustment for imprecision

```
;Lab 2, Section 2
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: Implements software delays
;***************INCLUDES************************************
.include "ATxmega128a1udef.inc"
;***************END OF INCLUDES*****************************

;*********************************EQUATES*******************************
.EQU sramend = 0x3fff ;top of stack
.EQU srambegin = 0x2000 ;bottom of stack
.EQU input = 0b00000000
.EQU output = 0b11111111
;*******************************END OF EQUATES*******************************

;*********************************DEFS*******************************
.DEF ms_r16 = r16
.DEF us_r17 = r17
.DEF zero_r18 = r18
.DEF one_r19 = r19
.DEF four_r20 = r20
;*******************************END OF DEFS*******************************

;***********MAIN PROGRAM*******************************
.CSEG
.org 0x0100
MAIN:
;initialize stack pointer
ldi r16, low(sramend)
out CPU_SPL, r16
ldi r16, high(sramend)
out CPU_SPH, r16
;set port directions
LDI R22, output
STS PORTC_DIR , R22
;initialize registers
ldi zero_r18,0
ldi one_r19,1
ldi four_r20,4
;loop to call subroutine
LOOP:
       rcall delay_10ms
       STS PORTC_OUTTGL,R22
RJMP LOOP
;***********END MAIN PROGRAM******************************

;********************SUBROUTINES**************************************
; Subroutine Name: delay_10ms
; performs a series of instructions for 10ms
; Inputs: none
; Ouputs: none
; Affected: r16, r17,r19,r20
delay_10ms:
       ldi ms_r16,0
       ldi us_r17,0
       tenms:
              ldi four_r20, 0
                     ;1ms
                     onekus:
                            ldi us_r17,0
```

```
                                    ;250us
                                    faus:
                                           add us_r17,one_r19
                                           cpi us_r17,253
                                    brne faus
                             add four_r20,one_r19
                             cpi four_r20,2
                      ;branch if 1ms
                      brne onekus
               add ms_r16,one_r19
               ;branch if 10 ms
               cpi ms_r16,10
        brne tenms
ret
```

Including "delay_x_10ms" subroutine. Delays for .04s.

```
;Lab 2, Section 2
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: Implements software delays
;**************INCLUDES************************************
.include "ATxmega128a1udef.inc"
;**************END OF INCLUDES*****************************

;*********************************EQUATES*******************************
.EQU sramend = 0x3fff ;top of stack
.EQU srambegin = 0x2000 ;bottom of stack
.EQU input = 0b00000000
.EQU output = 0b11111111
;******************************END OF EQUATES*******************************

;*********************************DEFS*******************************
.DEF ms_r16 = r16
.DEF us_r17 = r17
.DEF zero_r18 = r18
.DEF one_r19 = r19
.DEF four_r20 = r20
.DEF multiple_r21 = r21
;******************************END OF DEFS*******************************

;***********MAIN PROGRAM*******************************
.CSEG
.org 0x0100
MAIN:
;initialize stack pointer
ldi r16, low(sramend)
out CPU_SPL, r16
ldi r16, high(sramend)
out CPU_SPH, r16
;set port directions
LDI R22, output
STS PORTC_DIR , R22
;initialize registers
ldi zero_r18,0
ldi one_r19,1
ldi four_r20,4
;loop to call subroutine
LOOP:
       ldi multiple_r21,1
       rcall delay_x_10ms
       STS PORTC_OUTTGL,R22
RJMP LOOP
;***********END MAIN PROGRAM*****************************

;********************SUBROUTINES*************************************
; Subroutine Name: delay_10ms
; performs a series of instructions for 10ms
; Inputs: none
; Ouputs: none
; Affected: r16, r17,r19,r20
delay_10ms:
       ldi ms_r16,0
       ldi us_r17,0
       tenms:
              ldi four_r20, 0
                     ;1ms
```

```
                      onekus:
                             ldi us_r17,0
                                    ;250us
                                    faus:
                                           add us_r17,one_r19
                                           cpi us_r17,253
                                    brne faus
                             add four_r20,one_r19
                             cpi four_r20,2
                      ;branch if 1ms
                      brne onekus
               add ms_r16,one_r19
               ;branch if 10 ms
               cpi ms_r16,10
        brne tenms
ret
; Subroutine Name: delay_x_10ms
; delays a select multiple of 10ms
; Inputs: r21
; Ouputs: none
; Affected: r16,r17,r19,r20,r21
delay_x_10ms:
push r21
        loopx:
               cpi multiple_r21,0
               breq exit
               call delay_10ms
               dec multiple_r21
        rjmp loopx
exit:
pop r21
ret
```

Section 3

```
;Lab 2, Section 3
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: Implements software delay using timer
;**************INCLUDES************************************
.include "ATxmega128a1udef.inc"
;**************END OF INCLUDES*****************************

;*********************************EQUATES*******************************
.EQU input = 0b00000000
.EQU output = 0b11111111
;.EQU div2 = 0b00000010
.EQU prescalar = 8
.EQU sysclk = 2000000
.EQU desiredperiod = .04 ;40ms
.EQU reciprocal = 1/.04
.EQU offset = 175 ;correcting for imprecision
;*******************************END OF EQUATES*******************************

;*********************************DEFS********************************

;*******************************END OF DEFS*******************************

;**********MAIN PROGRAM*******************************
.CSEG
.org 0x0100
MAIN:

;initialize port c for output
ldi r16, output
sts PORTC_DIR, r16

;initialize count register
ldi r16,0
sts TCC0_CNT, r16
sts TCC0_CNT+1,r16

;**********************NOTES**********************
;if we want to achieve a period of 40 ms with a prescalar of 8
;and a frequency of 2mhz, that equates to:
;ticks = (2000000cycles/second)/(8cycles/tick)*.04seconds = 10000 ticks
;ticks = (systemclock/prescalar) / (1/desiredperiod)
;it also may be a good idea to add a number that corrects
;for any imprecision
;**********************END OF NOTES*****************

;initialize period register
ldi r16,low(((sysclk/prescalar)/reciprocal)+offset)
sts TCC0_PER, r16
ldi r16,high(((sysclk/prescalar)/reciprocal)+offset)
sts TCC0_PER+1,r16

;initialize clksel
ldi r16, CLK_PSADIV_8_gc
sts TCC0_CTRLA,r16

;toggle output port
loop:
       lds r17,TCC0_INTFLAGS
       ;check ov flag
```

```
       andi r17,0b00000001
       cpi r17,1
       ;branch if we have overflow
       breq toggleoutput
       ;else
       rjmp loop
       ;if we have an overflow
       toggleoutput:
              ;toggle outputs
              ldi r17,output
              sts PORTC_OUTTGL, r17
              ;clear ov flag
              ldi r17, 0b00000001
              sts TCC0_INTFLAGS,r17
rjmp loop
end:
rjmp end

;***********END MAIN PROGRAM******************************
```

Section 3, prelab question 9

```
;Lab 2, prelab question ix
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: Implements software delay using timer with prescalar
;value of 2
;**************INCLUDES************************************
.include "ATxmega128a1udef.inc"
;***************END OF INCLUDES******************************

;*********************************EQUATES*******************************
.EQU input = 0b00000000
.EQU output = 0b11111111
;.EQU div2 = 0b00000010
.EQU prescalar = 2
.EQU sysclk = 2000000
.EQU desiredperiod = .04 ;40ms
.EQU reciprocal = 1/.04
.EQU offset = 255 ;correcting for imprecision
;*******************************END OF EQUATES*******************************

;*********************************DEFS*******************************

;*******************************END OF DEFS******************************

;***********MAIN PROGRAM******************************
.CSEG
.org 0x0100
MAIN:

;initialize port c for output
ldi r16, output
sts PORTC_DIR, r16

;initialize count register
ldi r16,0
sts TCC0_CNT, r16
sts TCC0_CNT+1,r16

;**********************NOTES**********************
;if we want to achieve a period of 40 ms with a prescalar of 2
;and a frequency of 2mhz, that equates to:
;ticks = (2000000cycles/second)/(2cycles/tick)*.04seconds = 40000 ticks
;ticks = (systemclock/prescalar) / (1/desiredperiod)
;it also may be a good idea to add a number that corrects
;for any imprecision
;**********************END OF NOTES****************

;initialize period register
ldi r16,low(((sysclk/prescalar)/reciprocal)+offset)
sts TCC0_PER, r16
ldi r16,high(((sysclk/prescalar)/reciprocal)+offset)
sts TCC0_PER+1,r16

;initialize clksel
ldi r16, CLK_PSADIV_2_gc
sts TCC0_CTRLA,r16

;toggle output port
loop:
	lds r17,TCC0_INTFLAGS
	;check ov flag
```

```
        andi r17,0b00000001
        cpi r17,1
        ;branch if we have overflow
        breq toggleoutput
        ;else
        rjmp loop
        ;if we have an overflow
        toggleoutput:
                ;toggle outputs
                ldi r17,output
                sts PORTC_OUTTGL, r17
                ;clear ov flag
                ldi r17, 0b00000001
                sts TCC0_INTFLAGS,r17
rjmp loop
end:
rjmp end

;***********END MAIN PROGRAM*******************************
```

A screenshot of the waveform is located in the appendix in figure 15

When the prescalar value is set to 2, the offset needs to be higher in order to obtain the same time period.
This is because when the prescalar value decreases, there are more clock cycles required in order to count the same amount of time.
While this does allow more precise time measurement, you need more bits and a higher count value in order to measure the same amount of time.

Section 3, prelab question 10

```
;Lab 2, prelab question x
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: counts how many minutes have elapsed since start of program

;**************INCLUDES************************************
.include "ATxmega128a1udef.inc"
;**************END OF INCLUDES******************************

;********************************EQUATES*******************************
.EQU prescalar = 64
.EQU sysclk = 2000000
.EQU desiredperiod = 1 ;1000ms
.EQU reciprocal = 1/1
.EQU offset = 590 ;correcting for imprecision
;******************************END OF EQUATES*******************************

;********************************DEFS*******************************
.DEF second_r18 = r18
.DEF minute_r19 = r19
.DEF one_r20 = r20
;******************************END OF DEFS******************************

;**********MAIN PROGRAM******************************
.CSEG
.org 0x0100
MAIN:

;initialize registers
ldi second_r18,60
ldi minute_r19,0
ldi one_r20,1

;initialize count register
ldi r16,0
sts TCC0_CNT, r16
sts TCC0_CNT+1,r16

;**********************NOTES**********************
;if we want to achieve a period of 40 ms with a prescalar of 8
;and a frequency of 2mhz, that equates to:
;ticks = (2000000cycles/second)/(64cycles/tick)*1seconds = 31250 ticks
;ticks = (systemclock/prescalar) / (1/desiredperiod)
;it also may be a good idea to add a number that corrects
;for any imprecision
;**********************END OF NOTES*****************

;initialize period register
ldi r16,low(((sysclk/prescalar)/reciprocal)+offset)
sts TCC0_PER, r16
ldi r16,high(((sysclk/prescalar)/reciprocal)+offset)
sts TCC0_PER+1,r16

;initialize clksel
ldi r16, TC_CLKSEL_DIV64_gc
sts TCC0_CTRLA,r16

loop:
	lds r17,TCC0_INTFLAGS
	;check ov flag
```

```
	andi r17,0b00000001
	cpi r17,1
	;branch if we have overflow
	breq decrementsecond
	;else
	rjmp loop
	decrementsecond:
		;decrement "second" register
		dec second_r18
		;clear ov flag
		ldi r17, 0b00000001
		sts TCC0_INTFLAGS,r17
		;see if we hit a minute
		cpi second_r18, 0
		brne loop
		add minute_r19,one_r20
rjmp loop
end:
rjmp end

;***********END MAIN PROGRAM*******************************
```

Section 4

```
;*****************************************************************************
;  File name: lab2_4.asm
;  Author: Christopher Crary
;  Last Modified By: Steven Miller
;  Last Modified On: 3 june 2023
;  Purpose: To allow LED animations to be created with the OOTB uPAD,OOTB SLB, and OOTB MB.
;*****************************************************************************

;*******INCLUDES*************************************
.include "ATxmega128a1udef.inc"
;*******END OF INCLUDES******************************

;*******DEFINED SYMBOLS******************************
.equ ANIMATION_START_ADDR    =      0x2000
.equ ANIMATION_SIZE                 =      0x1fff
.EQU sramend = 0x3fff
.EQU allones = 0b11111111
.EQU allzeroes = 0b00000000
.EQU s2bit = 3
.EQU s1bit = 2
.EQU sysclk = 2000000
.EQU debouncereciprocal =1/.01
.EQU animationreciprocal = 1/.2
.EQU offset = 0
.EQU animationoffset = -192;to make it 5hz
.EQU prescalar = 1024
;*******END OF DEFINED SYMBOLS**********************

;*******MEMORY CONSTANTS****************************
; data memory allocation
.dseg
.org ANIMATION_START_ADDR
ANIMATION:
.byte ANIMATION_SIZE
;*******END OF MEMORY CONSTANTS*********************

;*******MAIN PROGRAM********************************
.cseg
.org 0x0000
       rjmp MAIN

.CSEG
.org 0x0100
MAIN:
; initialize the stack pointer
       ldi r16, low(sramend)
       sts CPU_SPL,r16
       ldi r16, high(sramend)
       sts CPU_SPH,r16
; initialize relevant I/O modules (switches and LEDs)
       rcall IO_INIT

; initialize (but do not start) the relevant timer/counter module(s)
       rcall TC_INIT

; Initialize the X and Y indices to point to the beginning of the
; animation table. (Although one pointer could be used to both
; store frames and playback the current animation, it is simpler
; to utilize a separate index for each of these operations.)
; Note: recognize that the animation table is in DATA memory
       ldi XL, low(ANIMATION_START_ADDR)
       ldi XH, high(ANIMATION_START_ADDR)
```

```
       ldi YL, low(ANIMATION_START_ADDR)
       ldi YH, high(ANIMATION_START_ADDR)

; begin main program loop

; "EDIT" mode
EDIT:

; Check if it is intended that "PLAY" mode be started, i.e.,
; determine if the relevant switch has been pressed.
       lds r16, PORTF_IN
; If it is determined that relevant switch was pressed,
; go to "PLAY" mode.
       sbrs r16, s2bit
       rjmp play

; Otherwise, if the "PLAY" mode switch was not pressed,
; update display LEDs with the voltage values from relevant DIP switches
; and check if it is intended that a frame be stored in the animation
; (determine if this relevant switch has been pressed).
       lds r16, PORTA_IN
       sts PORTC_OUT, r16

; If the "STORE_FRAME" switch was not pressed,
; branch back to "EDIT".
       lds r17, PORTF_IN
       sbrc r17, s1bit
       rjmp edit

; Otherwise, if it was determined that relevant switch was pressed,
; perform debouncing process, e.g., start relevant timer/counter
; and wait for it to overflow. (Write to CTRLA and loop until
; the OVFIF flag within INTFLAGS is set.)

       ;load period register
       ldi r16,low(((sysclk/prescalar)/debouncereciprocal)+offset)
       sts TCC0_PER, r16
       ldi r16,high(((sysclk/prescalar)/debouncereciprocal)+offset)
       sts TCC0_PER+1,r16
       ldi r16,TC_CLKSEL_DIV1024_gc
       sts TCC0_CTRLA,r16

       ;debouncing
       debounceloop:
              lds r17,TCC0_INTFLAGS
              ;check ov flag
              ;branch if we have overflow
              sbrs  r17,TC0_OVFIF_bp
              rjmp debounceloop

; After relevant timer/counter has overflowed (i.e., after
; the relevant debounce period), disable this timer/counter,
; clear the relevant timer/counter OVFIF flag,
; and then read switch value again to verify that it was
; actually pressed. If so, perform intended functionality, and
; otherwise, do not; however, in both cases, wait for switch to
; be released before jumping back to "EDIT".
       ;disable TC
       ldi r17, 0b00000000
       sts TCC0_CTRLA,r17
       ;clear ov flag
       ldi r17, 0b00000001
       sts TCC0_INTFLAGS,r17
       ;clear period register
```

```
        ldi r17, 0b00000000
        sts TCC0_PER,r17
        sts TCC0_PER+1,r17

; Wait for the "STORE FRAME" switch to be released
; before jumping to "EDIT".
STORE_FRAME_SWITCH_RELEASE_WAIT_LOOP:
        ;read switch again
        lds r17, PORTF_IN
        sbrs r17, s1bit
        rjmp store_frame_switch_release_wait_loop
        storeframe:
                ;store port A registers in X
                lds r16, PORTA_IN
                st x+, r16
                rjmp edit



; "PLAY" mode
PLAY:

; Reload the relevant index to the first memory location
; within the animation table to play animation from first frame.
        ldi YL,low(animation_start_addr)
        ldi YH,high(animation_start_addr)
        ld r20, y
        sts PORTC_OUT,r20

PLAY_LOOP:

; Check if it is intended that "EDIT" mode be started
; i.e., check if the relevant switch has been pressed.`
        lds r17, PORTF_IN
; If it is determined that relevant switch was pressed,
; go to "EDIT" mode.
        sbrs r17, s1bit
        rjmp edit
; Otherwise, if the "EDIT" mode switch was not pressed,
; determine if index used to load frames has the same
; address as the index used to store frames, i.e., if the end
; of the animation has been reached during playback.
; (Placing this check here will allow animations of all sizes,
; including zero, to playback properly.)
; To efficiently determine if these index values are equal,
; a combination of the "CP" and "CPC" instructions is recommended.
        ld r20, y
        comparexylower:
                ;compare lower bytes of x and y
                mov r16,xl
                mov r17, yl
                cp r16,r17
                breq comparexyhigher
                rjmp fivecyclecounter
                comparexyhigher:
                        ;compare higher bytes of x and y
                        mov r16,xh
                        mov r17, yh
                        cp r16,r17
                        breq play
                        rjmp fivecyclecounter


; If index values are equal, branch back to "PLAY" to
```

```
; restart the animation.


; Otherwise, load animation frame from table,
; display this "frame" on the relevant LEDs,
; start relevant timer/counter,
; wait until this timer/counter overflows (to more or less
; achieve the "frame rate"), and then after the overflow,
; stop the timer/counter,
; clear the relevant OVFIF flag,
; and then jump back to "PLAY_LOOP".

fivecyclecounter:
       ;initialize count
       ldi r16,0
       sts TCC0_CNT, r16
       sts TCC0_CNT+1,r16
       ;load period register
       ldi r16,low(((sysclk/prescalar)/animationreciprocal)+animationoffset)
       sts TCC0_PER, r16
       ldi r16,high(((sysclk/prescalar)/animationreciprocal)+animationoffset)
       sts TCC0_PER+1,r16
       ;initialize CLKSEL
       ldi r16,TC_CLKSEL_DIV1024_gc
       sts TCC0_CTRLA,r16
       ;load animation frames
       sts PORTC_OUT,r20
       loadanimationframe:
              lds r17,TCC0_INTFLAGS
              ;check ov flag
              ;branch if we have overflow
              sbrs  r17,TC0_OVFIF_bp
              rjmp loadanimationframe
              ;clear OVF
              ;sts PORTC_OUT,r20
              ldi r17, 0b00000001
              sts TCC0_INTFLAGS,r17
              adiw y,1
              rjmp play_loop


; end of program (never reached)
DONE:
       rjmp DONE
;*******END OF MAIN PROGRAM *************************

;*******SUBROUTINES**********************************

;***************************************************
; Name: IO_INIT
; Purpose: To initialize the relevant input/output modules, as pertains to the
;                application.
; Input(s): N/A
; Output: N/A
;***************************************************
IO_INIT:
; protect relevant registers
       push r16
; initialize the relevant I/O
       ;set port C as output
       ldi r16, allones
       sts PORTC_DIRSET,r16
       ;set port A as input
       ldi r16, allones
```

```
       sts PORTA_DIRCLR,r16
       ;set port F as input
       ldi r16, allones
       sts PORTF_DIRCLR,r16
; recover relevant registers
       pop r16
; return from subroutine
       ret
;****************************************************
; Name: TC_INIT
; Purpose: To initialize the relevant timer/counter modules, as pertains to
;                 application.
; Input(s): N/A
; Output: N/A
;****************************************************
TC_INIT:
; protect relevant registers
       push r16
; initialize the relevant TC modules
       ldi r16, allzeroes
       sts TCC0_CNT, r16
       sts TCC0_CNT+1,r16
       sts TCC0_PER, r16
       sts TCC0_PER+1, r16
       ;initialize CTRLB
       ldi r16, allzeroes
       sts TCC0_CTRLB,r16
       ;clear OVF
       ldi r17, 0b00000001
       sts TCC0_INTFLAGS,r17
; recover relevant registers
       pop r16
; return from subroutine
       ret

;*******END OF SUBROUTINES**************************
```

APPENDIX

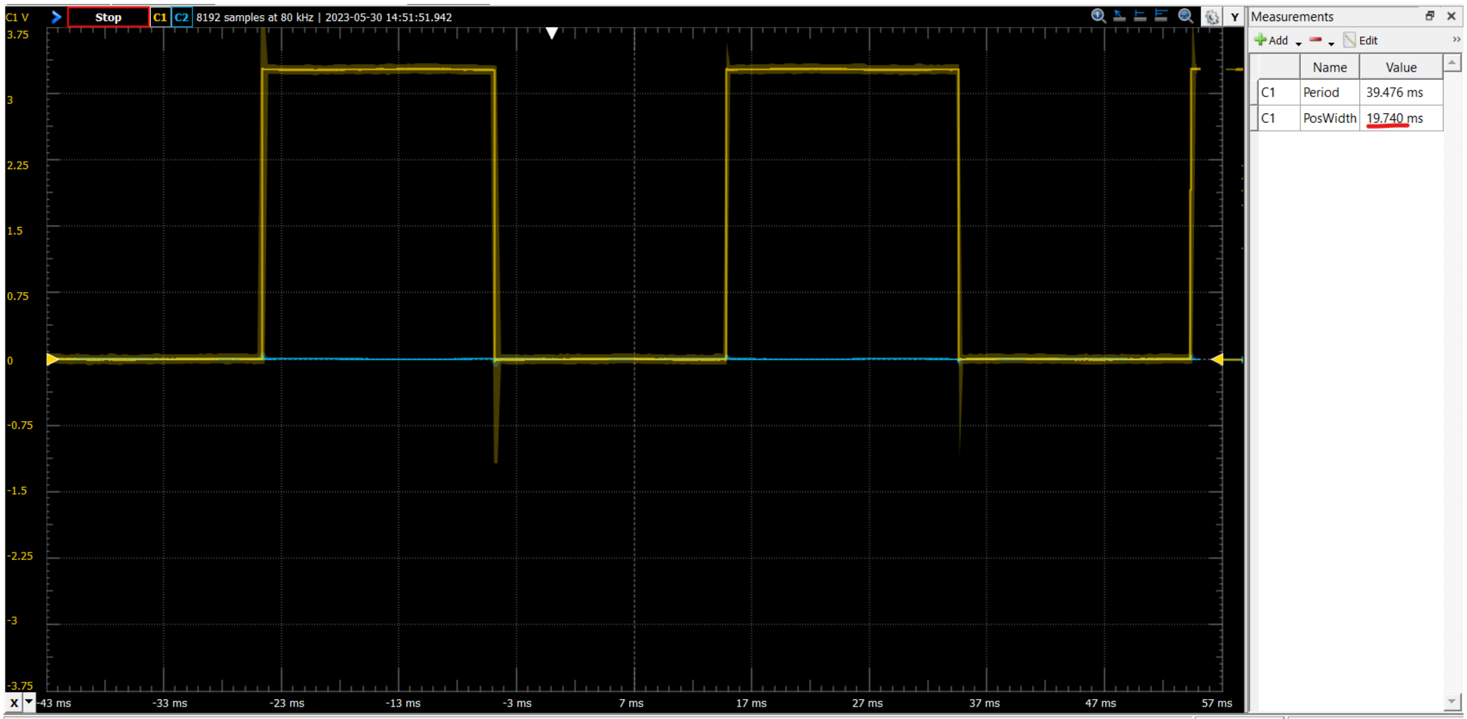


Figure 10: Software Delay created using "delay_10ms" without adjustment. The red underlined is the length of the delay.

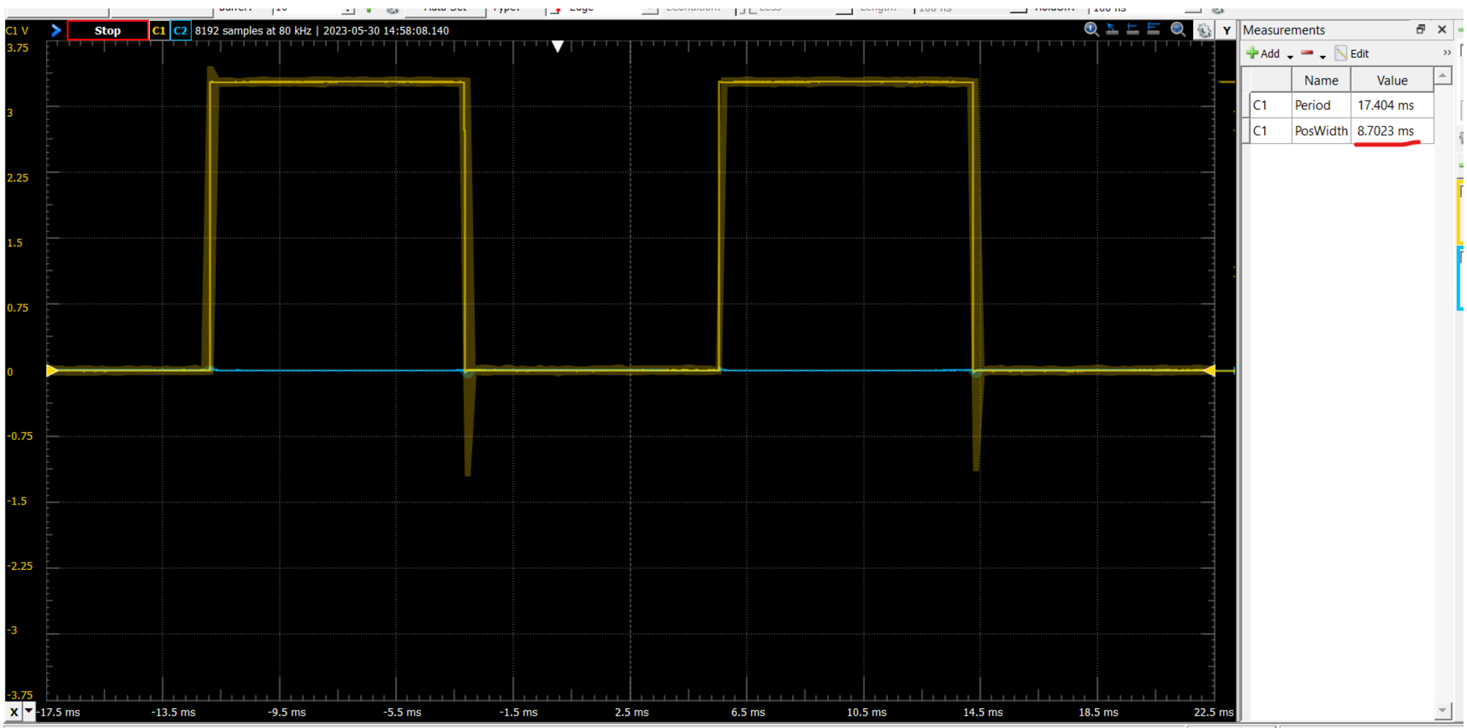


Figure 11: Software Delay created using "delay_10ms" after reducing number of times "ONEKUS" runs "FAUS" from 4 to 2

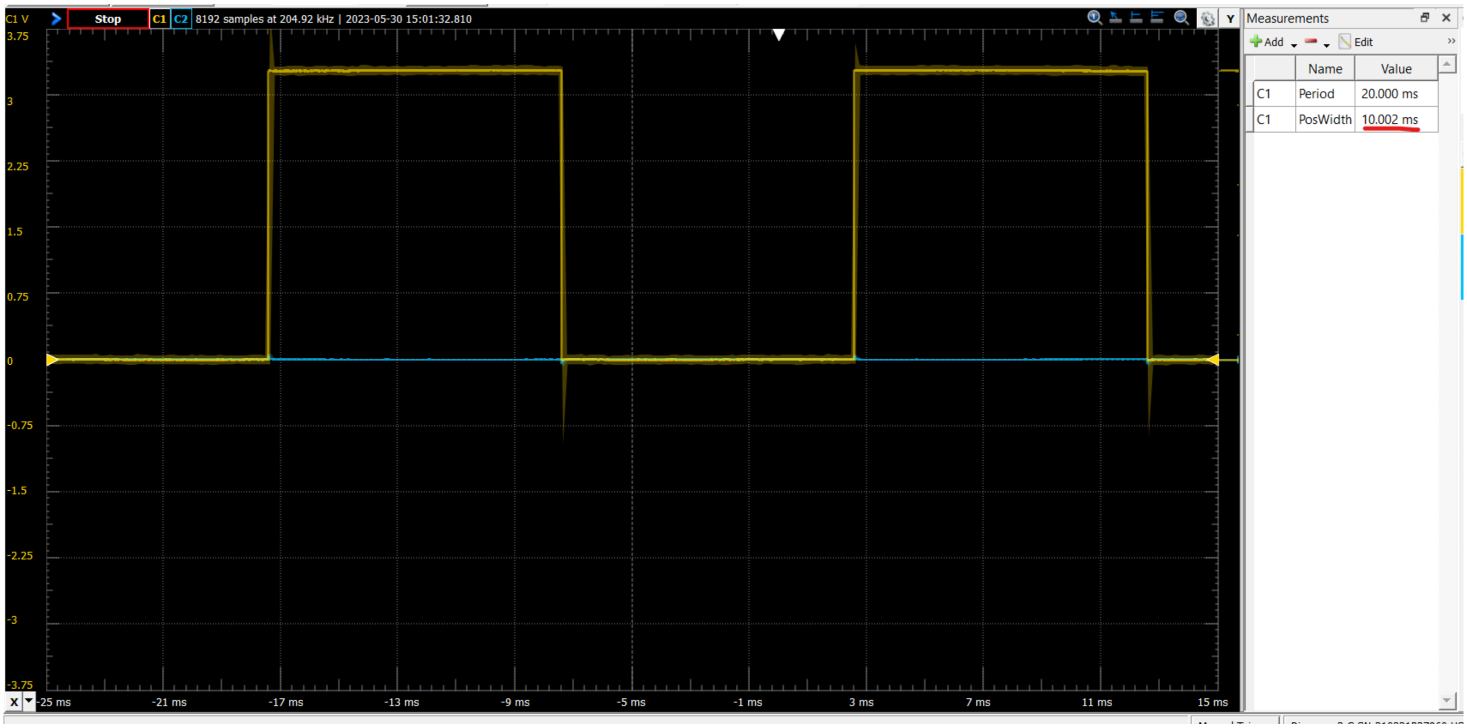


Figure 12: Software Delay created using "delay_10ms" subroutine after reducing number of times "ONEKUS" runs "FAUS" from 4 to 2, while also increasing number of times "FAUS" iterates from 250 to 253.

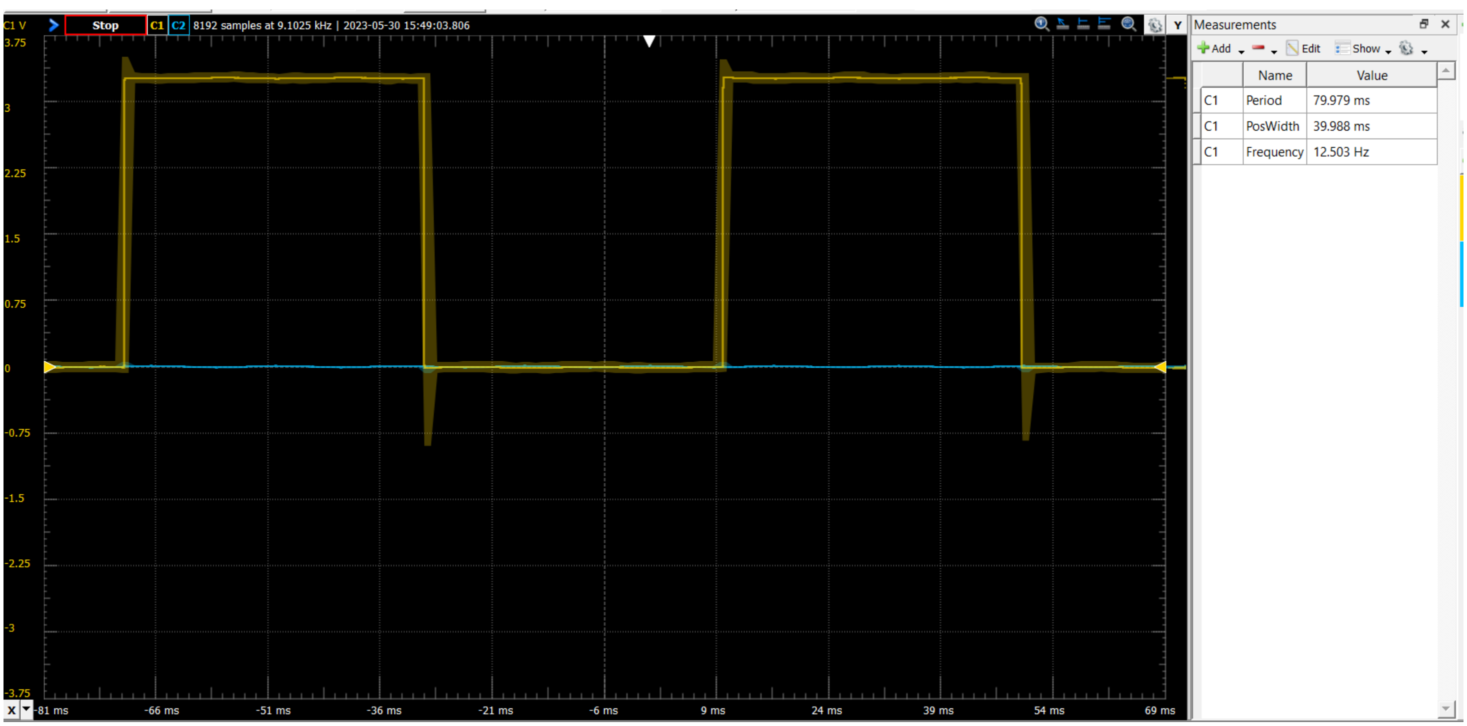


Figure 13: .04 second delay using "delay_x_10ms" subroutine

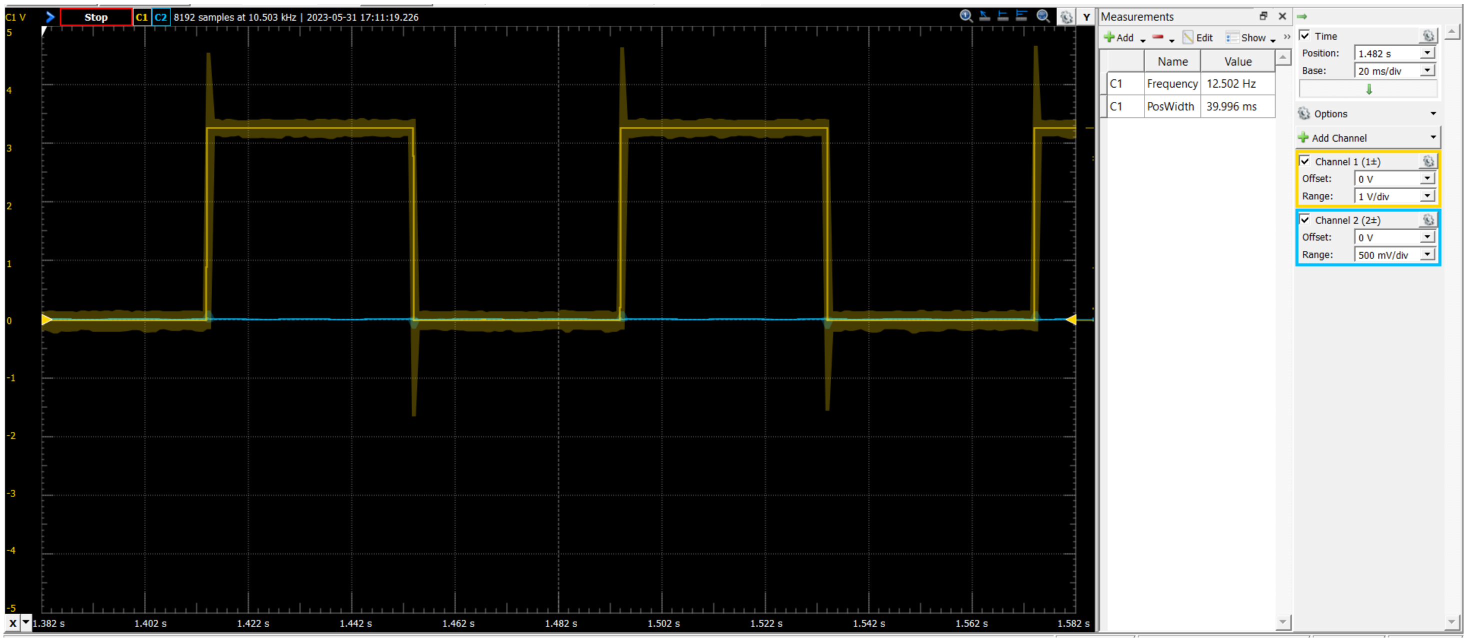


Figure 14: .04 second delay using timer/counter number 0. With a pre-scalar value of 8 An offset of 175 had to be added to account for imprecision.

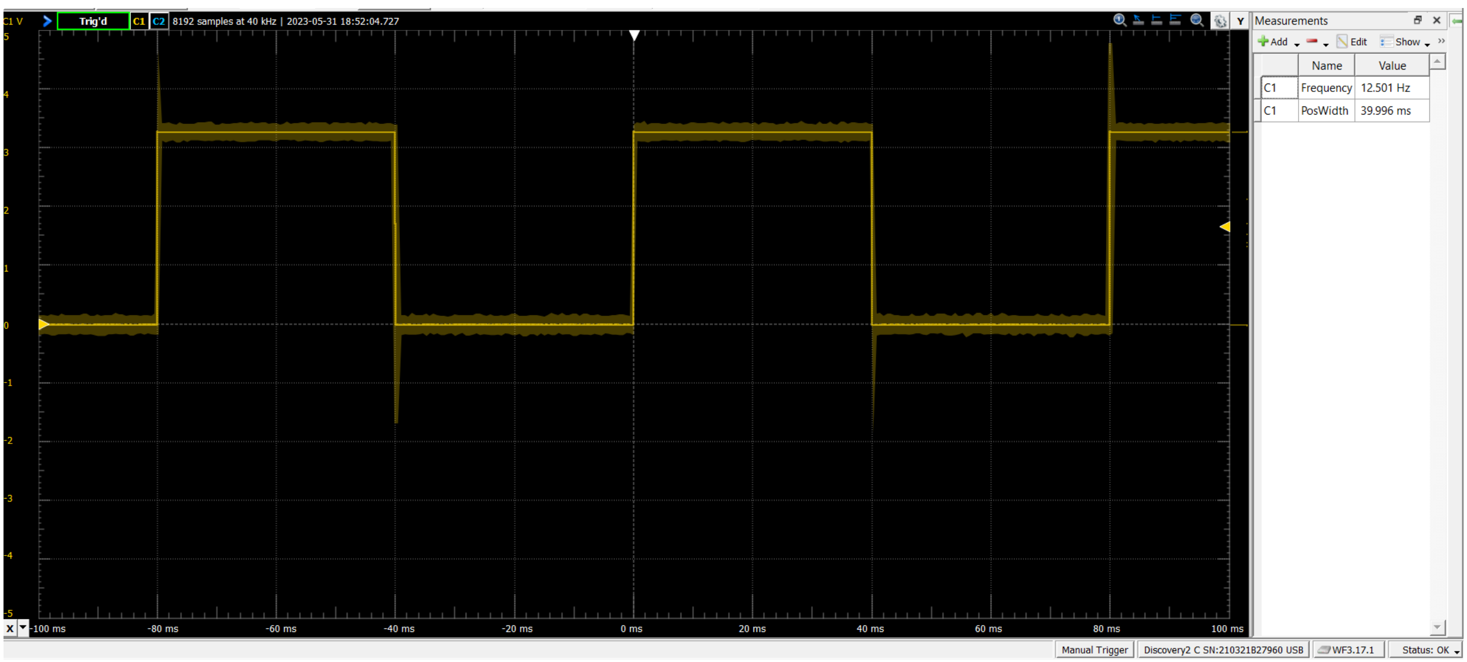


Figure 15: .04 second delay using timer/counter number 0 when pre-scalar value is set to 2. The offset is set to 760.

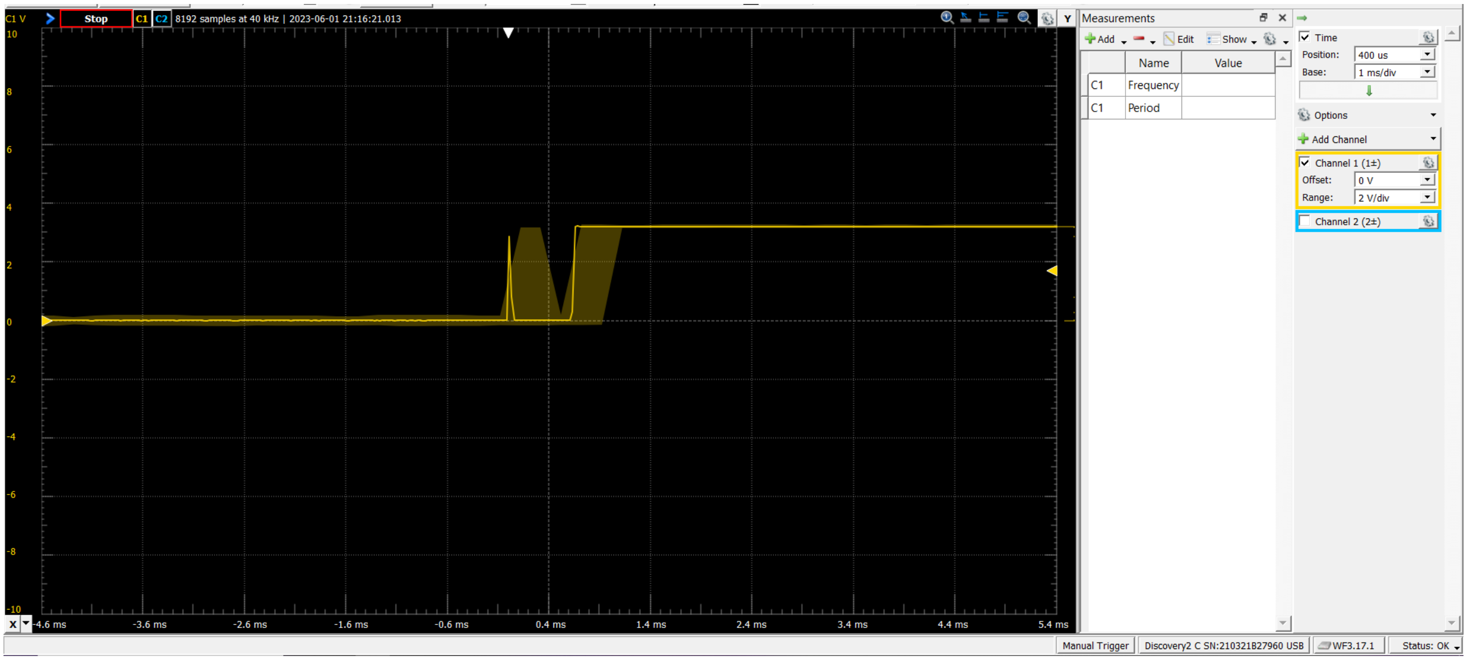


Figure 16: Bouncing on switch S2 when pressed.

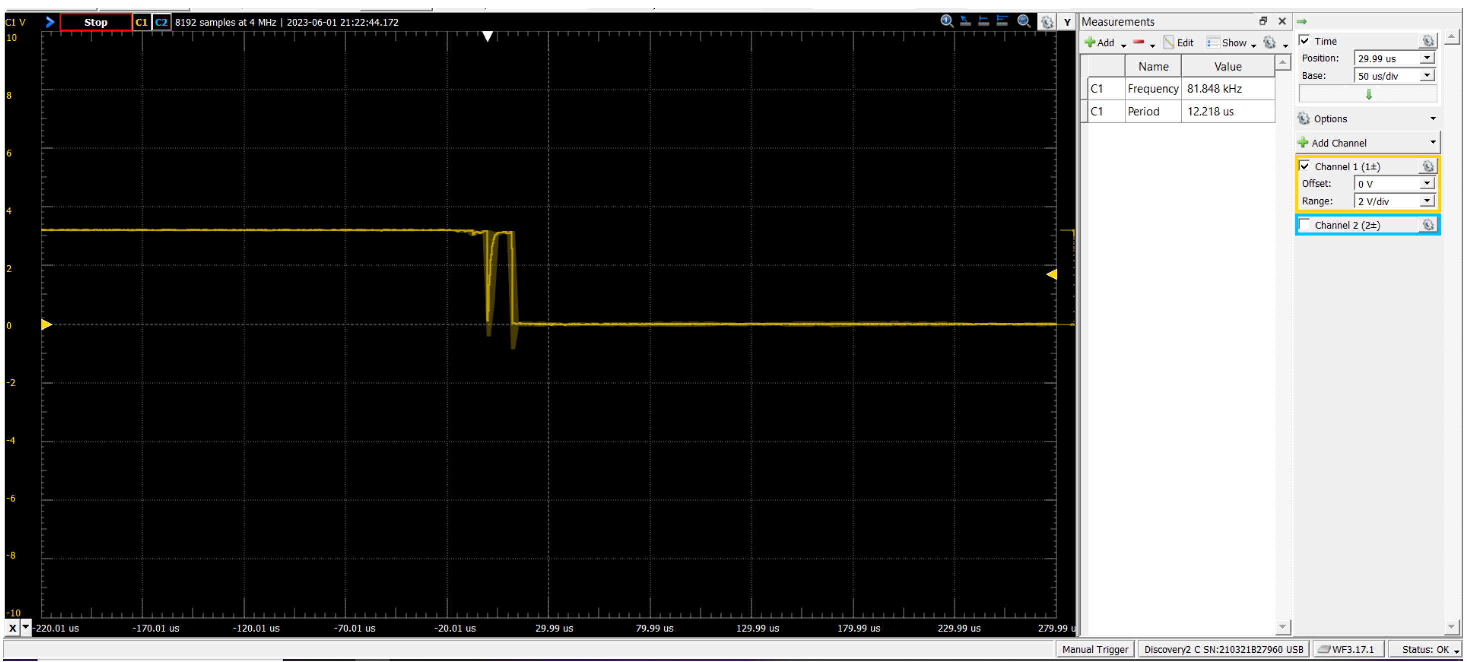


Figure 17: Bouncing on switch S2 when released.