# REQUIREMENTS NOT MET

N/A. All Requirements are met in this lab.

# PROBLEMS ENCOUNTERED

For part 1, the main problems encountered were understanding the functions and implementation of the port keywords. This was eventually solved by watching the supporting video and reading the microprocessor manual to be better equipped with the understanding and structure nature of the ports of the DIP switches and LEDs.

For part 2, the main problems encountered were understanding the function and implementation of the stack. This issue was solved by more research into the manuals. In addition, the structure for the subroutines were listed but never given an example so it became a makeshift process of making one to fit its requirements. There was another issue being that the delay on successfully run when adding an additional counter that branch to repeat the previous counter but would logically in my perspective do nothing to aid it.

For part 3, the main problems encountered were figuring out the number of loops to run the 10ms delay as the ideal calculated value taken from the equations did not actually produce the desired resulted active simulation. I believe it is due to propagating errors that occur with the remaining programing code. This was resolved with some plugging and chugging of looping interval range till the desired delay is met. Similarly, the same thing was done to get the desired frequency of 5Hz. Because of the occurring errors, the exercise of retrieving the minutes is quite unstable.

For part 4, the main problems encountered were figuring out the implementation of the debounce on the tactical switch as there was mainly conceptual pieces of how it works for me to work from. Some trouble was encountered with obtaining the include files to be placed in the Atmel project, this was solved when it was later demoed in class.

# FUTURE WORK/APPLICATIONS

This lab was a good introduction into input and output ports. This lab is to be the expansion of more complex assembly programs, able to give the users' information and allow the users to provide inputs to the microprocessor. If given more time, the code of the lab could have been more organized and have a much neater layout to further reduce the likelihood of mistakes and further enhance the understanding of the program. A more extensive denounce would have been implemented and many more different LEDs could have been placed into effects as the 8-bit input from the switch could have been translated to the RBG LEDs to provide an better animation. Additionally, I could have used better instructions to make the code run more efficiently or learn to write more complex programs.

# PRE-LAB EXERCISES

Part 1: Introduction to I/O

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?

Each port has one data direction register (DIR), when the DIRn bit in the DIR register is set to 1 then pin n is configured as an output while if the DIRn bit in the DIR register is set to 0 then the pin is configured as an input.

ii. What is the purpose of the DIRSET/DIRCLR configuration registers? Why do they exist, instead of just the DIR register? How about the OUTSET/OUTCLR registers, in regard to the OUT register?

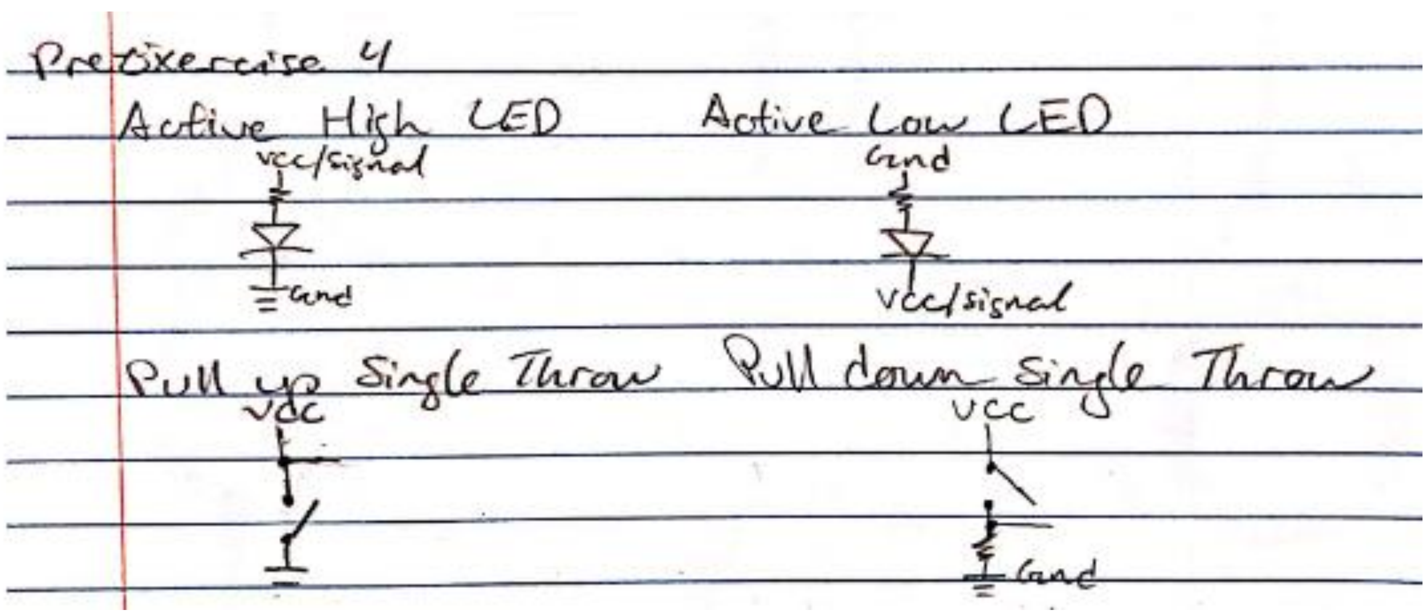
The purpose of the DIRSET/DIRCLR configuration registers is to set/clear, respectably, the corresponding bit in DIR. They exist to modify only individual pins you choose, changing some specify bits, to read/write and so that when reading it will return the value of the DIR register. Similarly, the OUTSET/OUTCLR registers does the same thing but with the OUT register.

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

The I/O ports which are utilized for the DIP switches is port A while the I/O ports which are utilized for the LEDs is port C.

iv. 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.

The LEDs on the OOTB Switch & LED Backpack are active-low.




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.

It would be possible to interface the OOTB µPAD with an external input device consisting of 24 inputs where it would be using all the female connectors on it being set as output, thus using all four ports or 24 pins.

Part 3: Introduction To Timer/Counters

vi. Assuming a system clock frequency of 2 MHz, a prescaler value of 256, and a desired period of 50 milliseconds, 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.

Dimensional Analysis

f=2MHz -> Period=0.5us -> 0.5*10^-6s        period= 50ms -> 50*10^-3s

2MHz tick per 50ms at 1 tick every 256 scaled tick.

2e6 SC ticks/50e-3 * 1 TC tick/256 SC tick = 390.625 tick

Formula

PER = D(SCF/PRE) = 50ms(2MHz/256) = 50e-3(2e6 SC ticks/256) = 390.625 ticks.

PER being the digital timer/counter period value, SCF being the relevant system clock frequency, PRE being the relevant timer/counter prescaler value, and D being the specified period of time, in terms of seconds.

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.

F=2MHz -> P=0.5us        16-bit -> 64k tick -> 32.768ms        prescaler = 1

Period = 64k *1*0.5us = 0.032s        The 2 seconds period is not achieved with a prescaler of 1.

For a 2 second period to be achieved, a prescaler, closest value, of 64 is needed.

2s = 64k *n*0.5us => n=62.5

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?

Assuming using a 16-bit counter, the max number of period tick is 64k. Therefore, using prescaler of 1 to maximize and a system clock frequency of 2 MHz, solve for the desired period:

PER = D(SCF/PRE) => 64k tick = D(2MHz/1) => 64k tick = D(2e6/1) => D= 33ms max time value.

Assuming using a 16-bit counter, the max number of period tick is 64k. Therefore, using prescaler of 1 to maximize and a system clock frequency of 32.768 kHz, solve for the desired period:

PER = D(SCF/PRE) => 64k tick = D(2MHz/1) => 64k tick = D(32.768e3/1) => D= 2e3 ms max time value.

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 256 and a prescaler value of two.

See part 3 code, DELAY_50MS.

As seen in the screenshot: 7 and 8, it shows that the period tick value with a prescaler value of 256 is smaller than the period with a prescaler value of 2. This is because, based on the equation PER = D(SCF/PRE), the prescaler is the denominator value so increasing it will reduce the overall fraction causing smaller value rather when the prescaler is decreasing which will increase overall period tick value.

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.

See part 3 code, EXERCISE10.

Part 4: LED Animation Creator

xi. It is stated above that, in the relevant context, it should not be necessary to debounce either of tactile switch S1 or tactile switch S2 located on the OOTB MB (or OOTB EBIBB, if using a previous version of the lab kit) – Why is this so?

It should not be necessary to debounce either of tactile switch S1 or tactile switch S2 located on the OOTB MB as after it is determined that the tactile switch has been pressed the next process should begin regardless of the state of switch occurring again. For instance, the when S2 is pressed the PLAY function should begin so if S1 still bounces it would not impact the PLAY mode of the program since it is independent of S1 at that point onward.

# PSEUDOCODE/FLOWCHARTS

## SECTION X (1, 2, etc.)

Part 1: Introduction to I/O

Johnny Li
Lab 2
EEL3744

Lab 2 Part 1

Loop for each pin on switch
if (closed) {
LED ON }
else if (open) {
LED OFF }
Loop

Part 2: Software Delays

Lab 2 Part 2

Toggle LED every 40ms.

Main: ldi PortC to output
sts PortC to be off initially.

Loop: rcall rountine
sts PortC Toggle
rjmp Loop

rountine: Load registers with values and decrement.
when equal to 0 Exit

Part 3: Introduction To Timer/Counters

Lab 2 Part 3

Toggle LED every 50ms with Timer.

Main: Initialize stack.

Set port C as output and off initially.

-Setup 16-bit counter, use TCC0 by default

-Set range of 0 to 255.

-Load Period

-Set Prescaler

Loop: rcall Delay

Toggle LED on/off

rjmp Loop

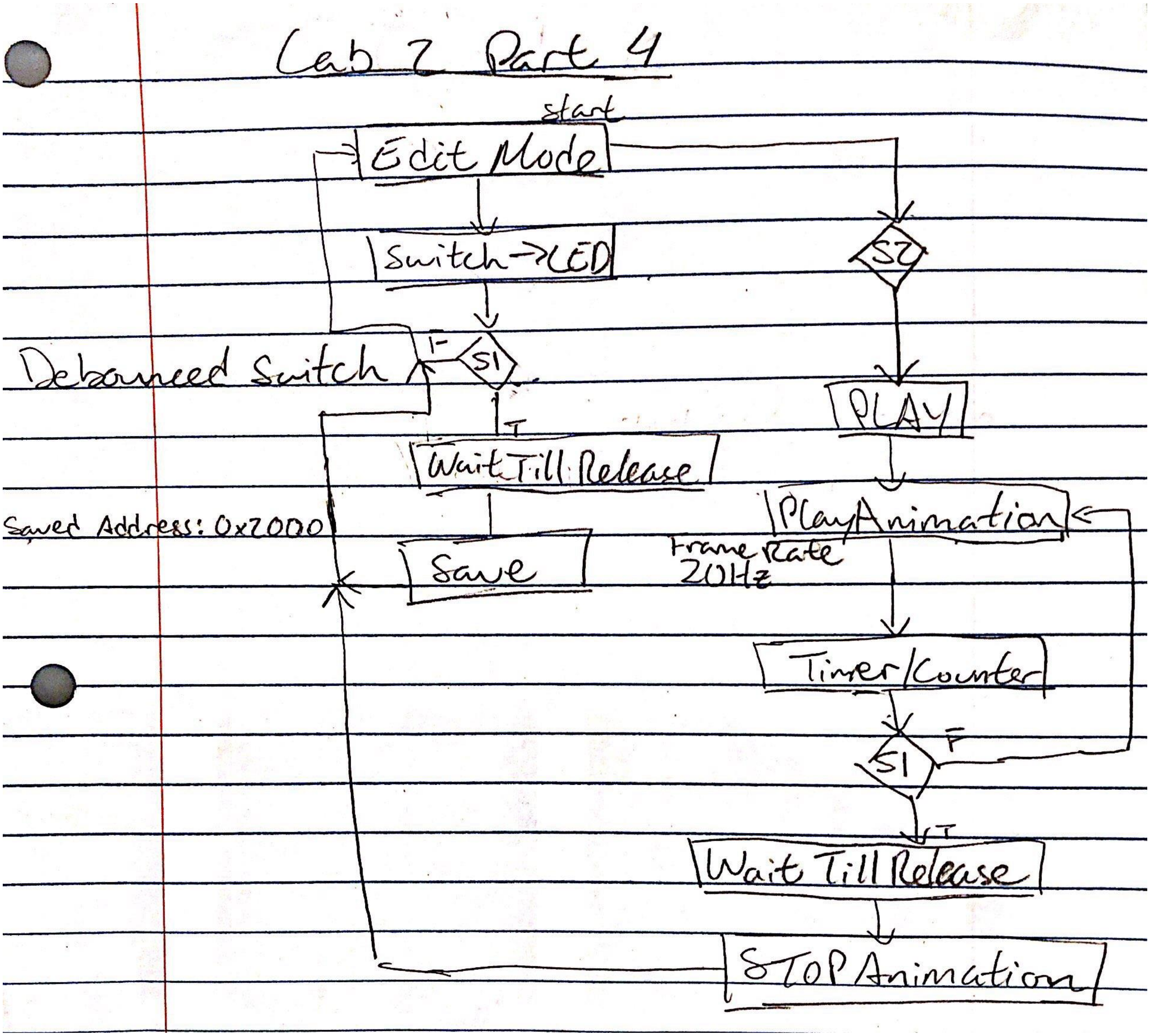
Delay: ldr r17, CPU_SREG ;load flag

sbrs r17,6 ;check for overflow

ret

rjmp Delay

Part 4: LED Animation Creator



# PROGRAM CODE

## SECTION X (1, 2, etc.)

Part 1: Introduction to I/O

```
;Lab 2 Part 1
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: I/O, Timing
;********************************MAIN PROGRAM**********************************
       .cseg
.org 0x0000
       rjmp MAIN

.org 0x100
MAIN:
       ldi r16, inlow       ;load 0 to register
    sts PORTA_DIR,r16       ;set portA as input

       ldi r16, outhigh     ;load 1 to register
       sts PORTC_OUT,r16    ;set portC initally off
       sts PORTC_DIR,r16    ;set portC as output

Loop:
       lds r16, PORTA_IN    ;load port A value to r16
    sts PORTC_OUT, r16      ;store switch value to LED portout

    rjmp Loop
;*****************************END**********************************************
```

Part 2: Software Delays

```
;Lab 2 Part 2
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: Software Delays
;********************************MAIN PROGRAM**********************************
       .cseg
.org 0x0000
       rjmp MAIN
.org 0x100
MAIN:
       ;Stack initialization
       ldi YL, low(stackaddress)
       out CPU_SPL,YL
       LDI YL, high(stackaddress)
       out    CPU_SPH,YL

       ldi r16, outhigh     ;load 1 to register
       sts PORTC_OUT,r16    ;set portC initally off
       sts PORTC_DIR,r16    ;set portC as output

Loop:
       rcall DELAY_X_10MS ;number of delays

       rcall DELAY_10MS     ;start subrountine
```

```
      ldi r16, 0x00 ;load true value to r16
   sts PORTC_OUT, r16      ;store toggle LED portout

      rcall DELAY_10MS     ;start subrountine

      ldi r16, outhigh     ;load true value to r16
   sts PORTC_OUT, r16      ;store toggle LED portout

   rjmp Loop

.org 0x300    ;put this here only to know the address of subroutine
;********************SUBROUTINES**************************************
; Subroutine Name: DELAY_10MS
; Delay program form 10ms
; Inputs: N/A
; Ouputs: N/A
; Affected: R16,R17,R18,R19
      ldi    r19, 0x01     ;bigger counter
DELAY_10MS:
      ldi r18, 0x00 ;end value
      ldi r16, 0xFF ;max value
      mov r17, r20  ;number of loop cycle

work:
      inc r18              ;counter
      cp r16,r18    ;compare if counter reach max value,
      breq ending   ;branch to end

ending:
      cpi r17, 0x00 ;check if zero
      breq done     ;done
      dec r17       ;contiune subrountine
      rjmp work

done:
      dec r19       ;bigger counter
      cpi r19, 0x00 ;check if zero
      brne DELAY_10MS      ;done

      ret    ;return stack pointer

.org 0x400    ;put this here only to know the address of subroutine
;********************SUBROUTINES**************************************
; Subroutine Name: DELAY_X_10MS
; Number of delays to be run
; Inputs: N/A
; Ouputs: N/A
; Affected: R20
DELAY_X_10MS:
      ldi r20, 11*0x09     ;Number of delays to be run
      ret
;*****************************END**********************************************
```

## Part 3: Introduction To Timer/Counters

```
;Lab 2 Part 3
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: Introduction To Timer/Counters
```

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

.org 0x100
MAIN:
       ;Stack initialization
       ldi YL, low(stackaddress)
       out CPU_SPL,YL
       LDI YL, high(stackaddress)
       out    CPU_SPH,YL

       ldi r16, outhigh     ;load 1 to register
       sts PORTC_OUT,r16    ;set portC initally off
       sts PORTC_DIR,r16    ;set portC as output

       ldi ZL, 0x87
       ldi ZH, 0x01
       sts TCC0_PER, ZL            ;load period tick
       sts TCC0_PER+1, ZH

       ;Ex10
       ldi r20,0     ;number of minutes
       ldi r21,0     ;number of runs 50ms -> 1s = 20 runs
       ldi r22,0     ;number of seconds

Loop:
       rcall DELAY_50MS     ;start subrountine

       ;Ex10
       inc r21 ;increment run

       ldi r18,1     ;reset flag
       sts TCC0_INTFLAGS, r18

       ldi r18,0     ;reset count
       sts TCC0_CNT, r18
       sts TCC0_CNT+1, r18

       ldi r16, 0x00 ;load true value to r16
       sts PORTC_OUT, r16   ;store toggle LED portout

       rcall DELAY_50MS     ;start subrountine

       ;Ex10
       inc r21 ;increment run
       cpi r21,20    ;1 seconds reached
       BREQ second

;Ex10
RETURN:
       cpi r22,60    ;1 min reached
       BREQ minute

       ldi r18,1     ;reset flag
       sts TCC0_INTFLAGS, r18

       ldi r18,0     ;reset count
       sts TCC0_CNT, r18
       sts TCC0_CNT+1, r18
```

```
	ldi r16, outhigh	;load true value to r16
	sts PORTC_OUT, r16	;store toggle LED portout

	rjmp Loop

;Ex10
second:
	inc r22		;increment second counter
	ldi r21, 0	;reset run counter
	rjmp RETURN

minute:
	inc r20		;increment minute counter
	ldi r22,0	;reset second counter
	rjmp return

.org 0x300	;put this here only to know the address of subroutine
;*********************SUBROUTINES***************************************
; Subroutine Name: DELAY_50MS
; Delay program form 50ms
; Inputs: N/A
; Ouputs: N/A
; Affected: R17,R18
DELAY_50MS:
	ldi r18, 6
	sts TCC0_CTRLA, r18		;increment and set prescaler 256

	lds r17, TCC0_INTFLAGS	;load flag
	sbrs r17, 0	;check if flag is triggered
	rjmp DELAY_50MS	;continue delay

	ret	;return stack pointer
;*****************************END*************************************************
```

## Part 4: LED Animation Creator

```
;Lab 2 Part 4
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: LED Animation Creator
;********************************MAIN PROGRAM**********************************
	.cseg

	; upon system reset, jump to main program (instead of executing
	; instructions meant for interrupt vectors)
	.org RESET_addr		; RESET_addr = 0x000
	rjmp MAIN

	; place the main program somewhere after interrupt vectors (ignore for now)
	.org MAIN_PROGRAM_START_ADDR		; MAIN_PROGRAM_START_ADDR = 0xFD
MAIN:
	; initialize the stack pointer
	INIT_STACK_POINTER	; macro defined in "atxmega128a1u_extra.inc"

	; 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
        ;initialize X
        ldi XL, low(ANIMATION_START_ADDR)
        ldi XH, high(ANIMATION_START_ADDR)
        ;initialize y
        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
        ; (determine if the relevant switch has been pressed)
        ;read S2
        lds r16, PORTF_IN
        sbrs r16, 3   ;check if 3rd bit is set thus S2 is pressed

        ; if it is determined that relevant switch was pressed,
        ; go to "PLAY" mode
        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    ;load port A value to r16
    sts PORTC_OUT, r16      ;store switch value to LED portout

        lds r16, PORTF_IN
        sbrc r16, 2   ;check if 2nd bit is set thus S1 is not pressed

        ; if the "STORE_FRAME" switch was not pressed,
        ; branch back to "EDIT"
        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)
Timer:
        ldi r18, 6
        sts TCC0_CTRLA, r18         ;increment and set prescaler 256

        lds r17, TCC0_INTFLAGS      ;load flag
        sbrs r17, 0   ;check if flag is triggered
        rjmp Timer    ;continue delay

        ; 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"
        ldi r18, 0    ;disable timer
        sts TCC0_CTRLA, r18

        ldi r18,0     ;reset count
```

```
       sts TCC0_CNT, r18
       sts TCC0_CNT+1, r18

       ldi r18,1     ;reset flag
       sts TCC0_INTFLAGS, r18
       ////////////////////////////////////////////////////////////////////
       lds r16, PORTF_IN
       sbrc r16, 2   ;check if 2nd bit is clear set thus S1 is not pressed
       rjmp EDIT     ;return to edit

       ; wait for the "STORE FRAME" switch to be released
       ; before jumping to "EDIT"
STORE_FRAME_SWITCH_RELEASE_WAIT_LOOP:
       lds r17, PORTA_IN    ;store port A input to
       st x+, r17    ;output table.

WAIT_LOOP:
       lds r16, PORTF_IN
       sbrc r16, 2   ;check if 2nd bit is clear set thus S1 is not pressed
       rjmp EDIT     ;return to edit
       rjmp WAIT_LOOP       ;wait till debounced is over

       ; "PLAY" mode
PLAY:
       ; reload the relevant index to the first memory location
       ; within the animation table to play animation from first frame
       ;reload y to memory location
       ldi YL, low(ANIMATION_START_ADDR)
       ldi YH, high(ANIMATION_START_ADDR)
       rjmp PLAY_LOOP

PLAY_LOOP:
       ; check if it is intended that "EDIT" mode be started
       ; i.e., check if the relevant switch has been pressed
       lds r16, PORTF_IN
       sbrs r16, 2   ;check if 2nd bit is set thus S1 is pressed

       ; if it is determined that relevant switch was pressed,
       ; go to "EDIT" mode
       rjmp EDIT     ;return to 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
       cp XL,YL      ;compare lower half
       cpc XH, YH    ;compare upper half

       ; if index values are equal, branch back to "PLAY" to
       ; restart the animation
       breq PLAY

       ; 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"
        ld r19, Y+    ;load table
        sts PORTC_OUT, r19   ;store switch value to LED portout

TimerPLAY:
        ldi r18, 6
        sts TCC0_CTRLA, r18         ;increment and set prescaler 256

        lds r17, TCC0_INTFLAGS      ;load flag
        sbrs r17, 0   ;check if flag is triggered
        rjmp TimerPLAY       ;continue delay

        ldi r18, 0    ;disable timer
        sts TCC0_CTRLA, r18

        ldi r18,0     ;reset count
        sts TCC0_CNT, r18
        sts TCC0_CNT+1, r18

        ldi r18,1     ;reset flag
        sts TCC0_INTFLAGS, r18

        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
        ; Switch
        ldi r16, inlow       ;load 0 to register
     sts PORTA_DIR,r16       ;set portA as input
        ; LED
        ldi r16, outhigh     ;load 1 to register
        sts PORTC_OUT,r16    ;set portC initally off
        sts PORTC_DIR,r16    ;set portC as output
        ; S1
        ldi r16, 0x01<<2
        sts PORTF_DIRCLR, r16
        ; S2
        ldi r16, 0x01<<3
        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

	; initialize the relevant TC modules
	ldi ZL, 0x87
	ldi ZH, 0x01
	sts TCC0_PER, ZL		;load period tick
	sts TCC0_PER+1, ZH

	; recover relevant registers

	; return from subroutine
	ret
;*****************************END OF SUBROUTINES*******************************
;*****************************END OF "lab2_4.asm"*******************************
```

# APPENDIX

Part 1: INTRODUCTION TO I/O

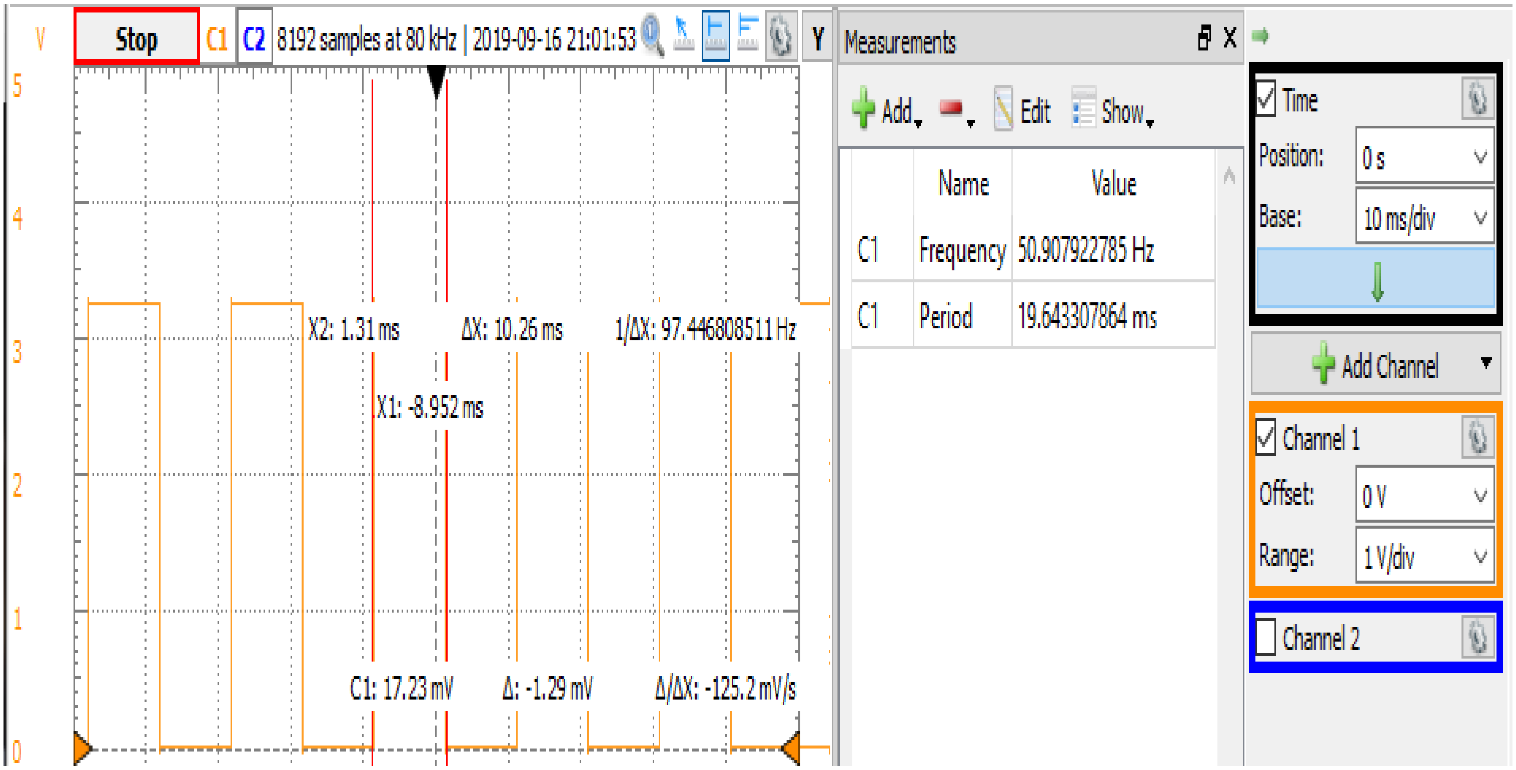
```
;*********************************INCLUDES************************************
.include "ATxmega128a1udef.inc"
;******************************END OF INCLUDES*******************************
;******************************DEFINED SYMBOLS*******************************
.equ outhigh=0xff   ;set as input or high value
.equ inlow=0x00 ;set as output or low value
;**************************END OF DEFINED SYMBOLS****************************
;******************************MEMORY CONSTANTS******************************
;***************************END OF MEMORY CONSTANTS**************************
```

Screenshot 1: Memory Configuration of part 1

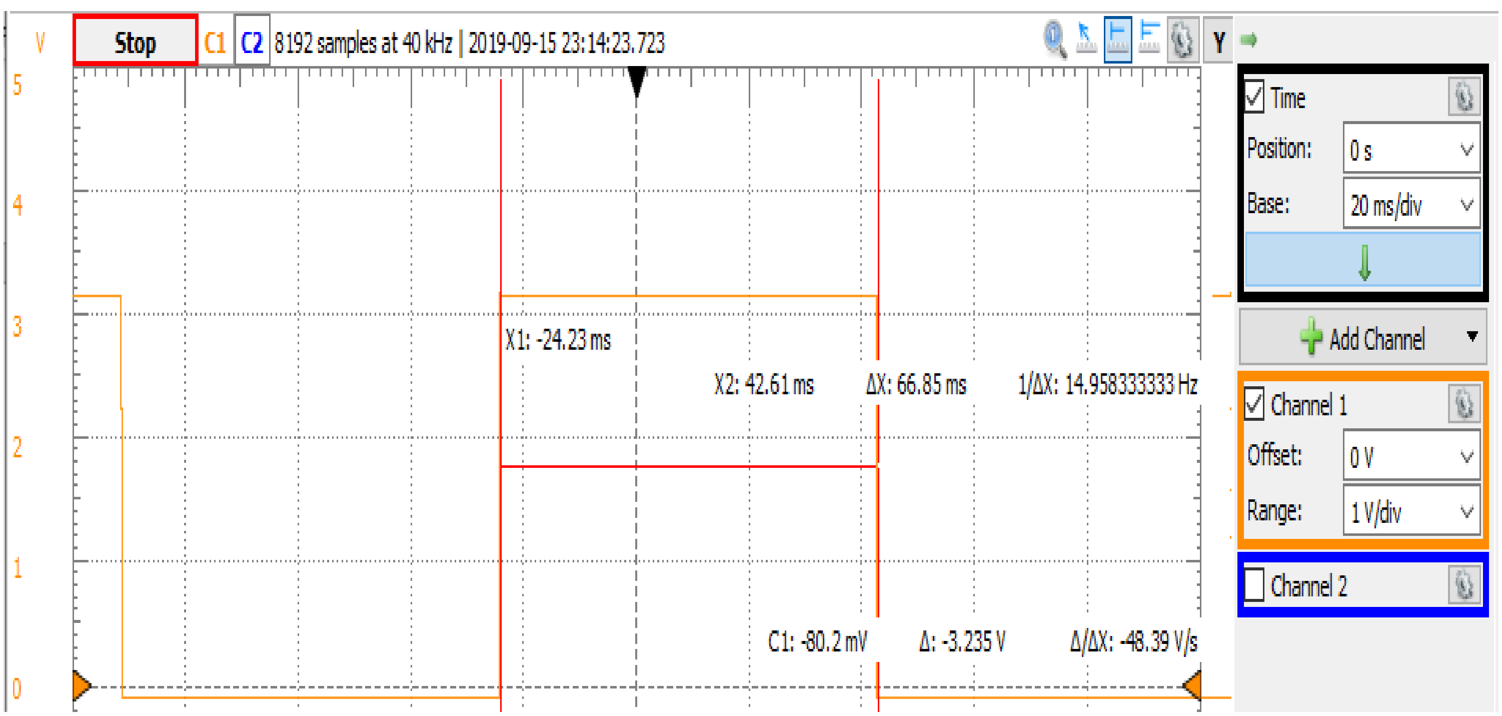
Part 2: Software Delays

```
;*********************************INCLUDES************************************
.include "ATxmega128a1udef.inc"
;******************************END OF INCLUDES*******************************
;******************************DEFINED SYMBOLS*******************************
.equ outhigh=0xff   ;set as input or high value
.equ stackaddress=0x3FFF    ;stack starting address
;**************************END OF DEFINED SYMBOLS****************************
;******************************MEMORY CONSTANTS******************************
;***************************END OF MEMORY CONSTANTS**************************
```

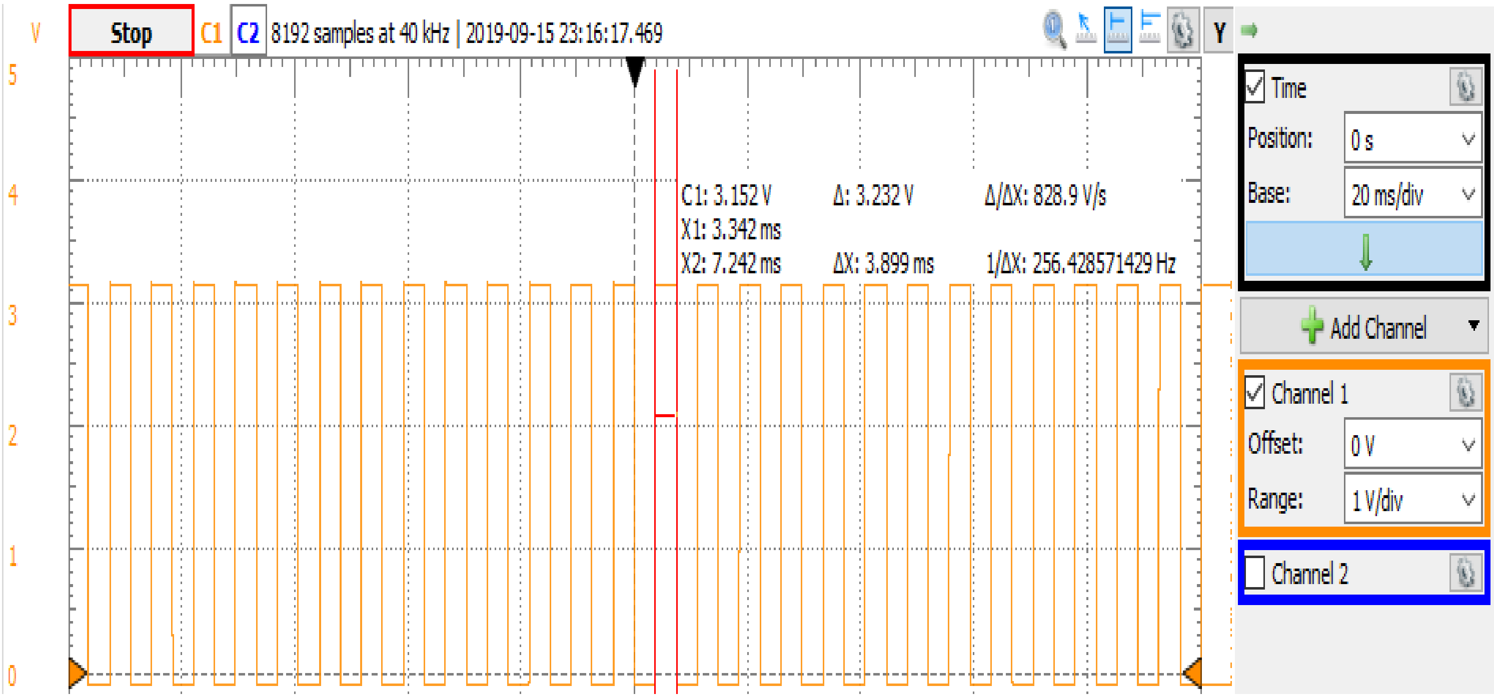
Screenshot 2: Memory Configuration of part 2



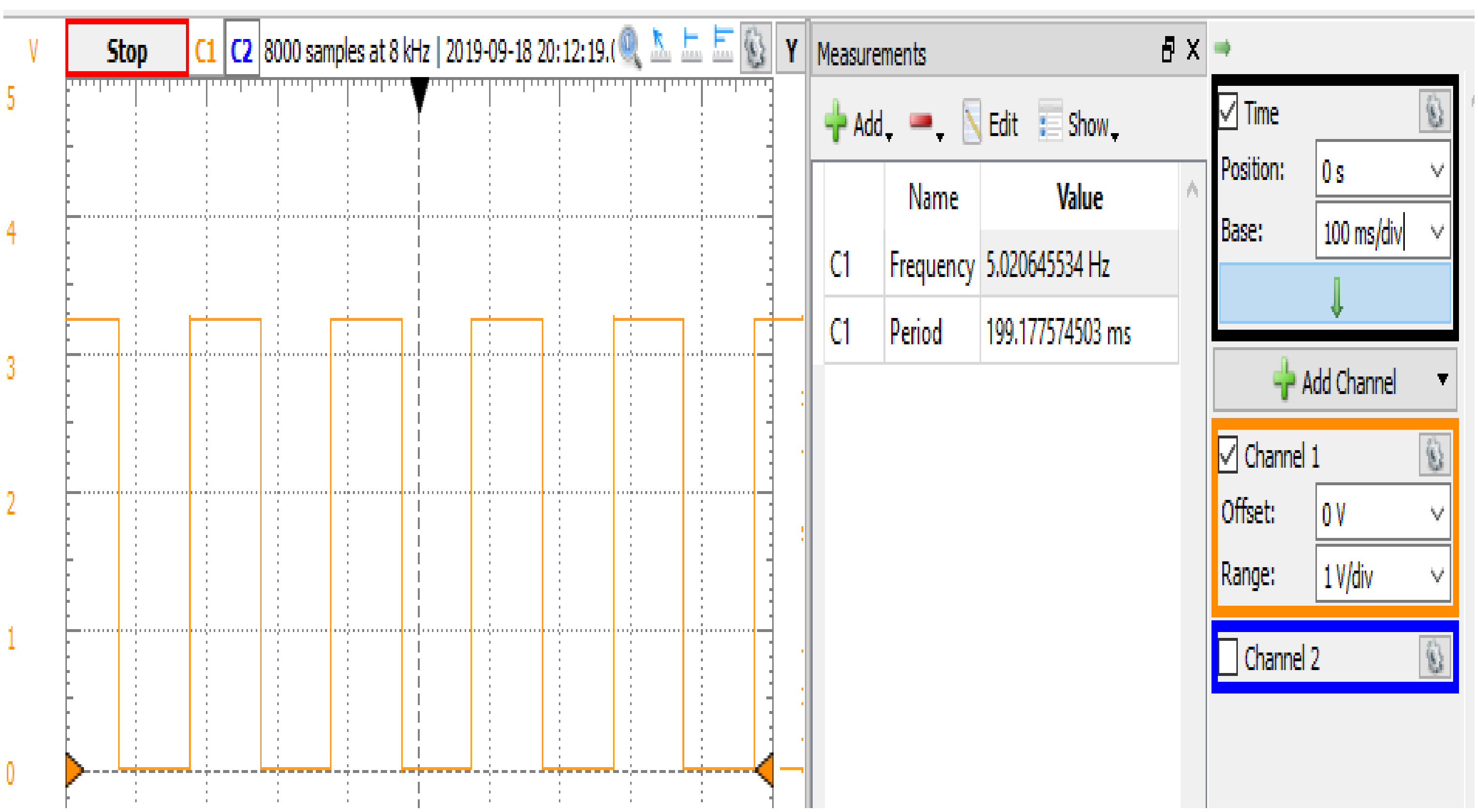

Screenshot 3: Software Delay of part 2, about 10ms. Half the period is on with a delay of 10ms till it becomes off and delay again for 10ms to be on again, totaling a 20ms period.




Screenshot 4: Software Delay Trail 1, overtime.




Screenshot 5: Software Delay Trail 2, undertime.



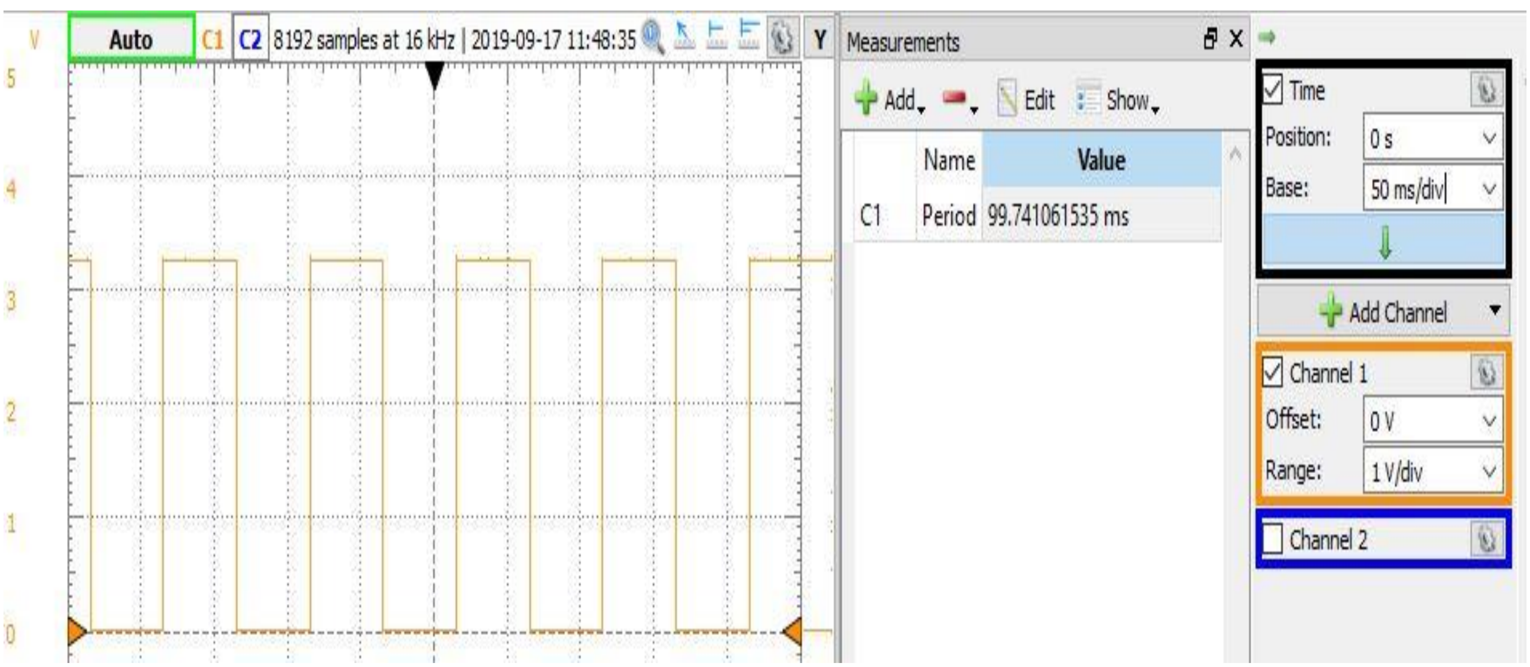

Screenshot 6: Software Delay of part 2, about 5Hz. Half the frequency is on having the toggle frequency is 10 Hz, i.e., that the overall waveform has a frequency of 5 Hz.

Part 3: Introduction To Timer/Counters

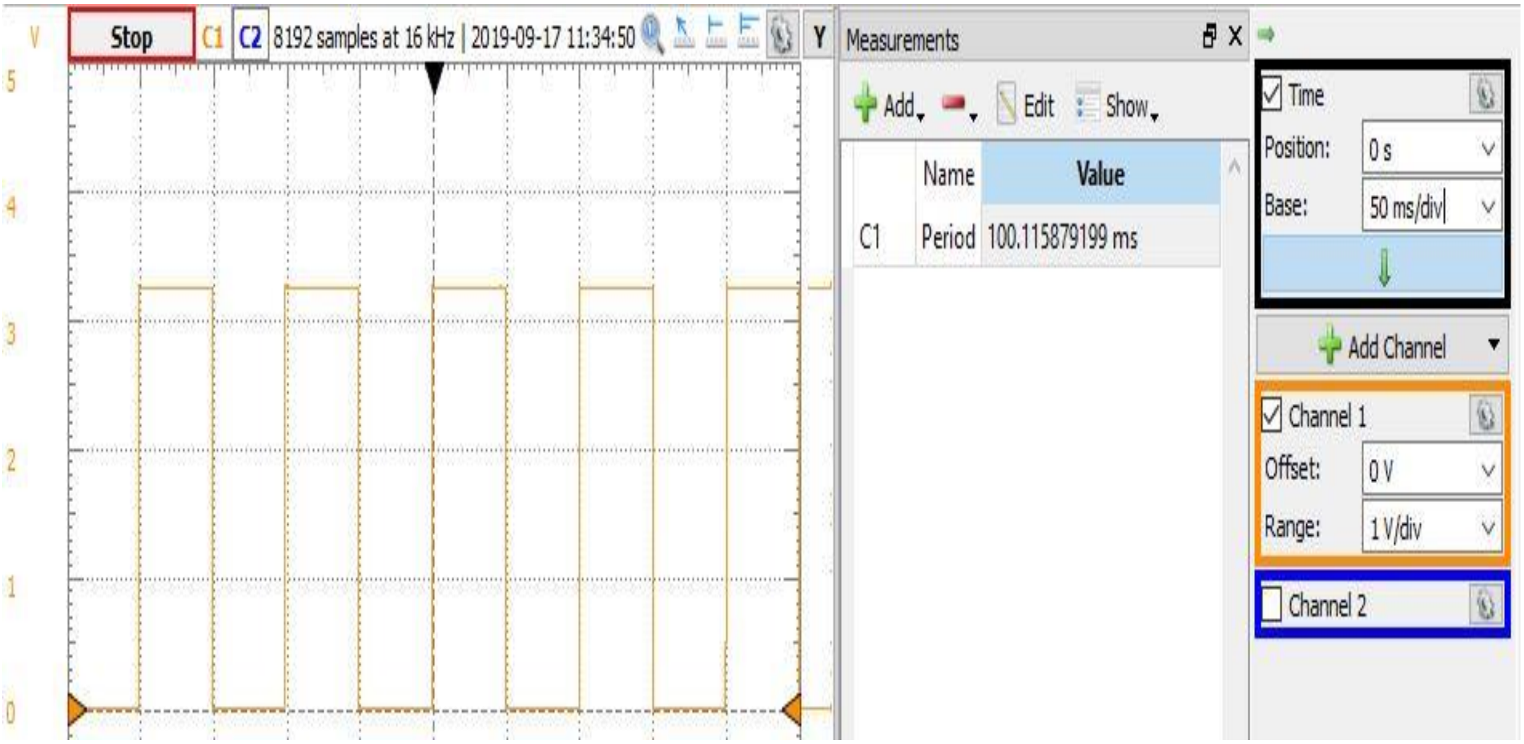
lab2_3.asm | lab2_4.asm

```
;*********************************INCLUDES*************************************
.include "ATxmega128a1udef.inc"
;******************************END OF INCLUDES**********************************
;******************************DEFINED SYMBOLS**********************************
.equ outhigh=0xff   ;set as input or high value
.equ stackaddress=0x3FFF    ;stack starting address
;***************************END OF DEFINED SYMBOLS******************************
;******************************MEMORY CONSTANTS*********************************
;***************************END OF MEMORY CONSTANTS*****************************
```

Screenshot 7: Memory Configuration of part 3




Screenshot 8: Timer Delay with a prescaler of 256 of part 3, about 50ms. Half the period is on with a delay of 50ms till it becomes off and delay again for 50ms to be on again, totaling a 100ms period.




Screenshot 9: Timer Delay with a prescaler of 2 of part 3, about 50ms. Half the period is on with a delay of 50ms till it becomes off and delay again for 50ms to be on again, totaling a 100ms period.

Part 4: LED Animation Creator

lab2_4.asm

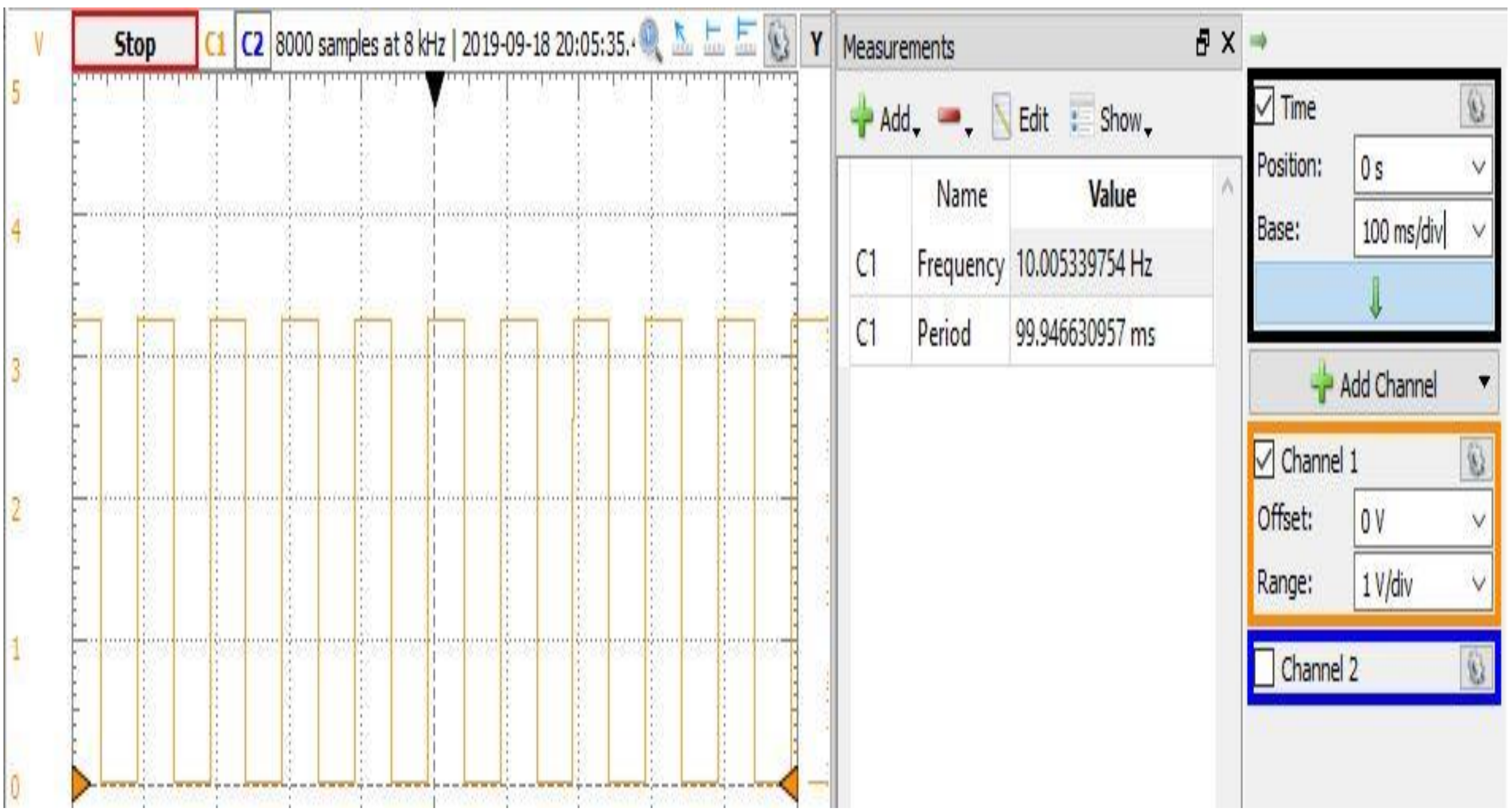
```
;*****************************************************************************
;  File name: lab2_4.asm
;  Author: Christopher Crary
;  Last Modified By: Christopher Crary
;  Last Modified On: 06 September 2019
;  Purpose: To allow LED animations to be created with the OOTB µPAD,
;           OOTB SLB, and OOTB MB (or EBIBB, if a previous version of the kit
;           is used).
;
;           NOTE: The use of this file is NOT required! This file is just given
;           as an example for how to potentially write code more effectively.
;*****************************************************************************
;*********************************INCLUDES************************************
; the inclusion of the following file is REQUIRED for our course, since
; it is intended that you understand concepts regarding how to specify an
; "include file" to an assembler. (this is also included within "lab2_4.inc"
; provided below, so it is not necessary here, but it is done for good
; measure.)
.include "ATxmega128a1udef.inc"

; certain aspects of this overall program are placed into the following
; "include" file, for the purpose of potentially increasing both the
; readability of the application code and the "modularity" of the application.
; the use of this (or any) "include" file is NOT required for our course!
.include "lab2_4.inc"
;*****************************END OF INCLUDES*********************************
;*****************************DEFINED SYMBOLS*********************************
.equ outhigh=0xff   ;set as input or high value
.equ inlow=0x00 ;set as output or low value
;*****************************END OF DEFINED SYMBOLS**************************
```

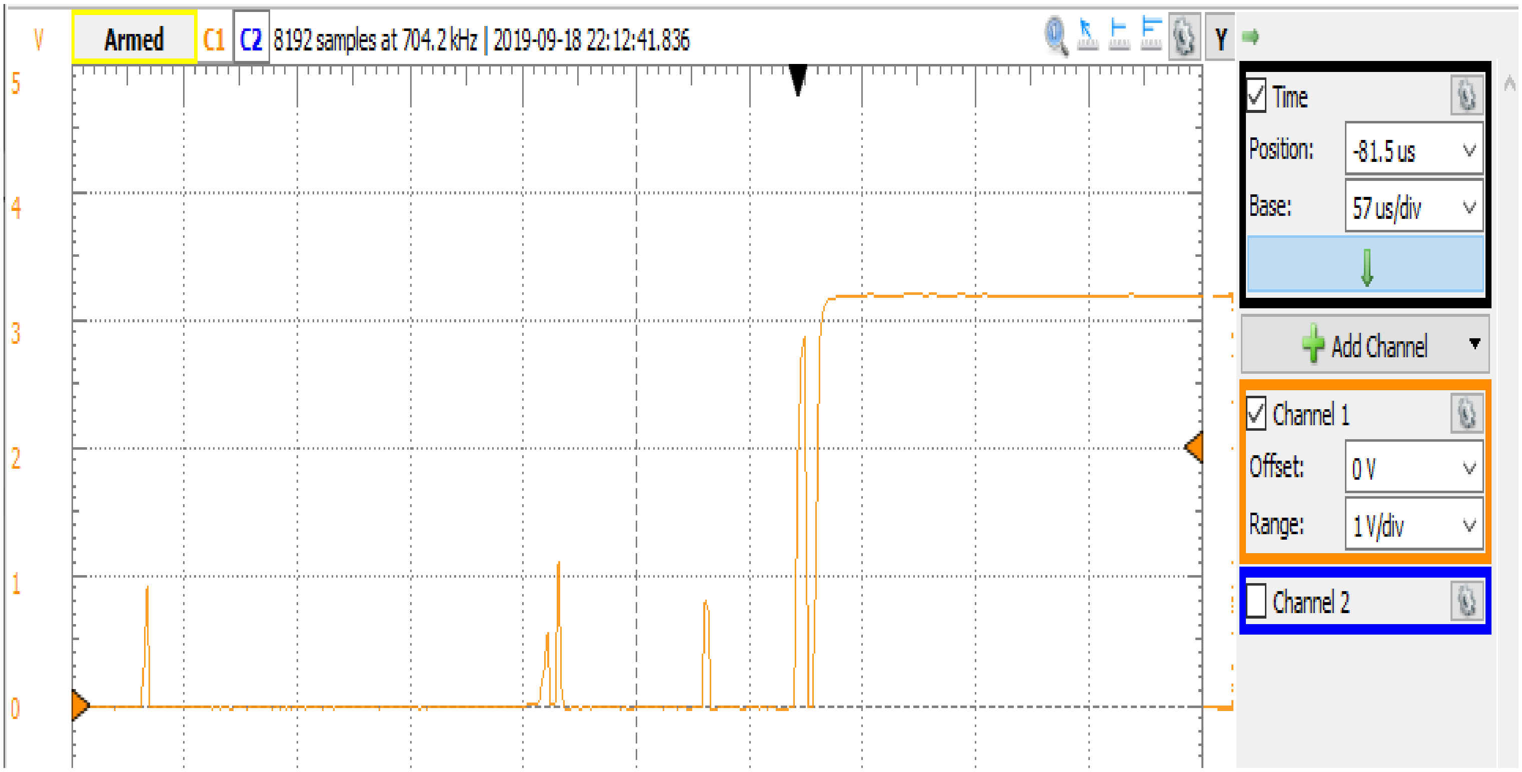
```
;*************************END OF DEFINED SYMBOLS*******************************
;*****************************MEMORY CONSTANTS*********************************
    ; data memory allocation
    .dseg

    .org ANIMATION_START_ADDR       ; ANIMATION_START_ADDR = 0x2000
ANIMATION:
    .byte ANIMATION_SIZE            ; ANIMATION_SIZE = 0x2000
```

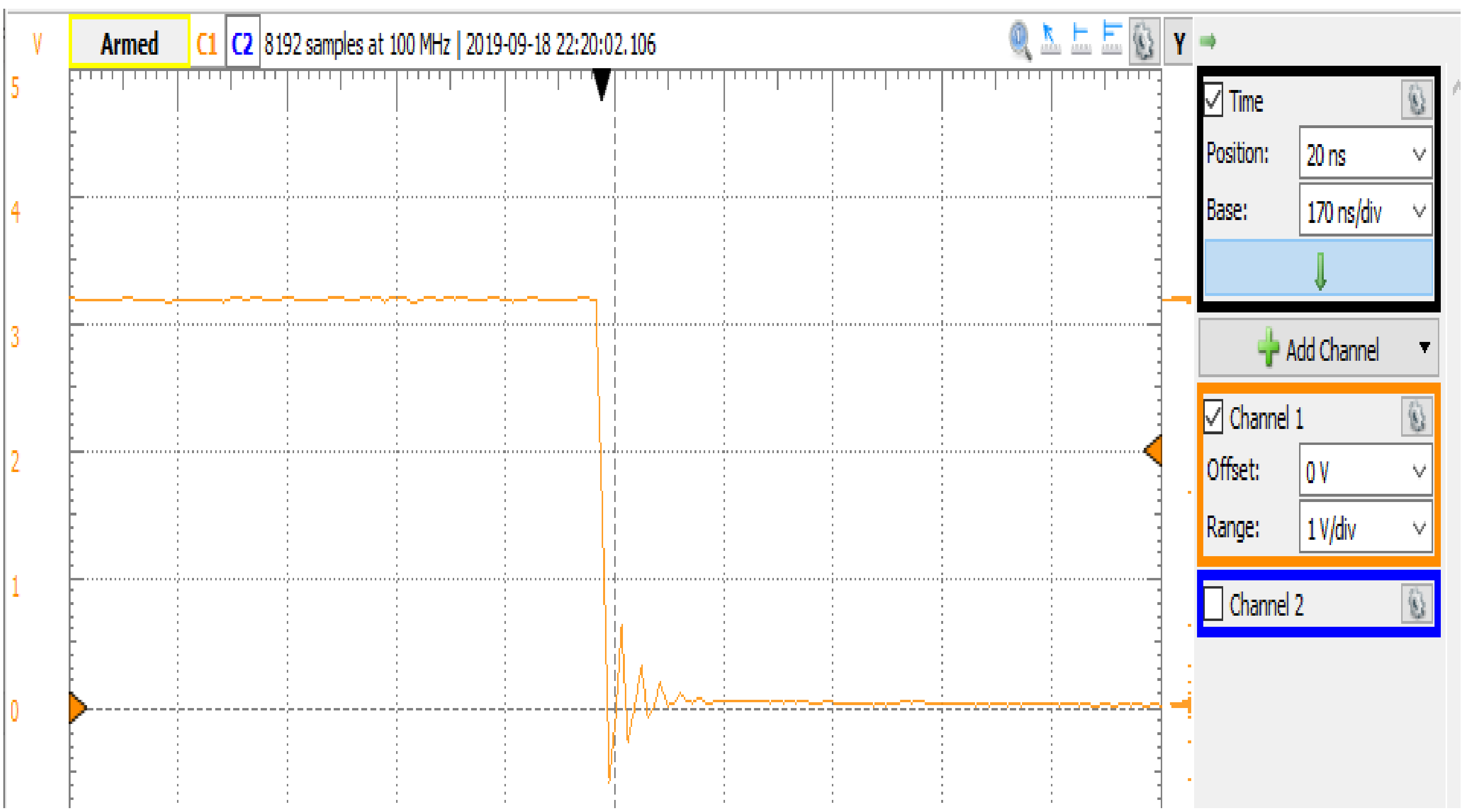
Screenshot 10: Memory Configuration of part 4




Screenshot 11: Animation frequency, having the toggle frequency is 20 Hz, i.e., that the overall waveform has a frequency of 10 Hz.




Screenshot 12: Tactile switch bouncing when pressed.




Screenshot 13: Tactile switch bouncing when released.