REQUIREMENTS NOT MET

N/A

PROBLEMS ENCOUNTERED

In all of the programs, I had to adjust the PER value compared to the theoretical calculation in order to achieve the correct measured timing on the scope. I also had a lot of issues using the DAD oscilloscope and needed to revisit the Waveforms tutorial to properly set up the time base and capture valid signals. In addition, I often needed to carefully review the XMEGA manuals to fully understand how to configure and use the clock and timer registers correctly. I also had to debug my code for a very long time since I did not debug as I coded.

FUTURE WORK/APPLICATIONS

The work in this lab provides a foundation that can be extended by introducing hardware interrupts, which will be the focus of the next lab. By layering interrupts on top of the timing and polling techniques used here, tasks like LED animation playback and watch counting could be handled automatically by the hardware instead of continuous software loops. This would not only make the programs more efficient but also open the door to building more advanced applications such as real-time clocks/alarms or responsive user interfaces.

PRE-LAB EXERCISES

Part 1:

i. Which configuration register allows the utilization of a subset of I/O port pins to be configured as inputs (without affecting other pins)? Which configuration register allowed the utilization of a subset of I/O port pins to be configured as an output (without affecting other pins)?

   a. The PORTx_DIRCLR configuration register allows the utilization of a subset of I/O port pins to be configured as inputs (without affecting other pins).
   b. The PORTx_DIRSET configuration register allows the utilization of a subset of I/O port pins to be configure as an output (without affecting other pins).

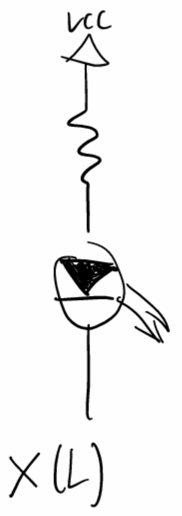
ii. What is the purpose of the SET/CLR/TGL variants of the DIR and OUT registers?

   a. The purpose of the SET/CLR/TGL variants of the DIR and OUT registers is to allows you to alter just a subset of bits which you intend to change and leave the rest of them unaffected. This enables a safer way to modify bits since it automatically preserves the rest of the bits.

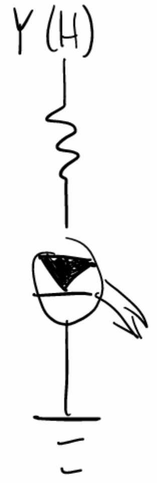
iii. Are the LEDs on the OOTB Switch & LED Backpack active-high, or active-low? Draw a schematic diagram for a single LED circuit with the same activation level used on the backpack, as well as one with the opposite activation level. Also, draw a schematic diagram for a single-pole, single-throw (SPST)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

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

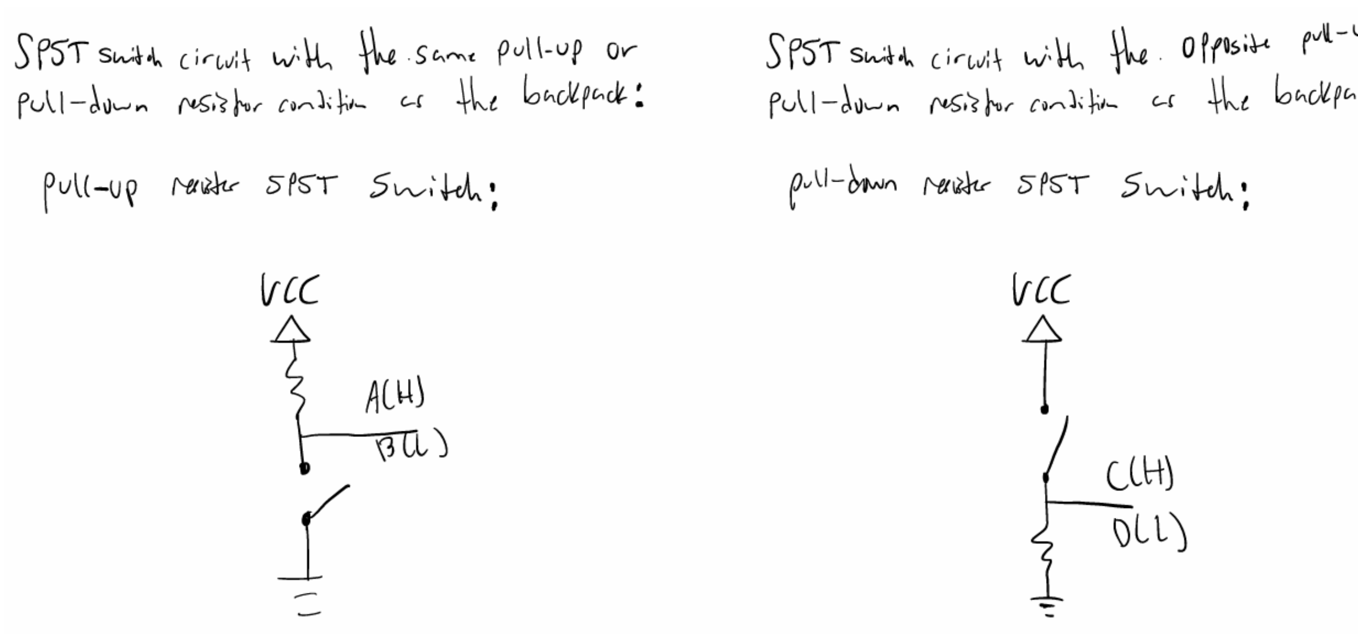
LED circuit with same activation level as the backpack (Active-Low):



LED Circuit with opposite activation as the backpack (Active-High):



   b.

c. 

SPST switch circuit with the same pull-up or pull-down resistor condition as the backpack:

pull-up resistor SPST Switch:

SPST switch circuit with the opposite pull-
pull-down resistor condition as the backpa

pull-down resistor SPST Switch:

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

a. The A I/O port is utilized for the DIP switches on the OOTB Switch & LED Backpack.
b. The C I/O port is utilized for the LEDs on the OOTB Switch & LED Backpack

v. Would it be possible to interface the OOTB µPAD with an external input device consisting of 22 inputs? If so, describe how many I/O ports would be necessary. If not, explain why.

a. In general, it would be possible to interface the OOTB µPAD with an external input device consisting of 22 inputs since it provides 78 Programmable I/O pins. 22 inputs would require three input ports, since each port contains eight pins, allowing for 24 inputs. Since there are at least three ports available on the µPAD, it is possible.

vi. Assuming a system clock frequency of 2 MHz, a prescaler value of 256, and a desired timer/clock period of 55 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

Sys Clock Frequency: 2MHz

Prescaler: 256

Desired Period: 55ms

Calculate timer/counter period

Dimensional Analysis:

$$\underbrace{\frac{2e^4 \cancel{\text{SC ticks}}}{10\text{ ms}}}_{\text{System clock frequency}} \times \frac{1\text{ TC tick}}{\underbrace{256}_{\text{prescaler}} \cancel{\text{SC ticks}}} = \frac{\frac{2e^4}{256}\text{ TC ticks}}{10\text{ ms}} = \frac{78.125\text{ TC ticks}}{10\text{ ms}} \cdot \underbrace{\frac{55\text{ ms}}{1}}_{\text{Desired period}} = 429.6875\text{ ticks}$$

$$\approx \underbrace{430}_{\text{whole number ticks}}\text{ ticks}$$

General Formula:

$$\text{Per} = D\left(\frac{SCF}{PRE}\right) \qquad \text{Per} = 55\text{ms}\left(\frac{2\text{MHz}}{255}\right) = 55\cdot10^{-3}\left(\frac{2\cdot10^6}{256}\right) = 429.6875$$

$$\approx 430\text{ ticks}$$

a.

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.

SCF: 2MHz

Desired period: 2s

16 bit timer counter, prescaler = 1

$$\text{Max per} = 2^{16} = 65{,}536$$

$$65{,}356 = D \cdot 2 \cdot 10^6$$

$$\underline{D = 0.032768\,s}$$ much less than 2 seconds

$$65{,}356 = 2 \cdot \left(\frac{2 \cdot 10^6}{X}\right)$$

$$0.016384 = \frac{1}{X} \rightarrow x = 61{,}035$$

↑ any prescaler above 61.035 will work

Using PRE = 64 →

$$\text{Per} = 2\left(\frac{2 \cdot 10^6}{64}\right)$$

$$\text{Per} = 62{,}500$$

a.

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 37.444 kHz?

Max time val (to nearest millisecond) representable by a timer/counter, if the relevant system clock frequency is 2 MHz?

What about a sys clock Freq of 37.444 kHz

Assuming 16-bit timer/counter, $F_{clk} = 2\,MHz$:

$$T_{max} = \frac{2^{16}}{f_{clk}} = \frac{65{,}536}{f_{clk}} \qquad \underline{f_{clk} = 2 \cdot 10^6\ Hz} \rightarrow \frac{65{,}536}{2 \cdot 10^6} = 0.032768 \text{ seconds}$$

$$\approx 33 \text{ milliseconds}$$

If $f_{clk} = 37.444\ kHz$:

$$T_{max} = \frac{65{,}536}{37.444 \cdot 1000} = 1.75024 \text{ seconds}$$

$$\approx 1{,}750 \text{ milliseconds}$$

a.

ix. Create an assembly program to perform the same procedure as in § 3.2 but utilize a prescaler value of 8. 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, the value in exercise vi, and a prescaler value of 8.

a. Using a prescaler of 8 results in a timer tick of 4 µs, compared to 128 µs with a prescaler of 256. Because the tick interval is smaller, the PER register can be set with finer resolution, yielding a closer match to the desired 55 ms half-period. This reduces the error. The program `lab2_3a.asm` was written to implement this case using prescaler = 8. The experimentally determined PER value that provided the least error was 14,071, producing a measured half-period of 55.00 ms, which is essentially error-free compared to the expected value of 55.00 ms. By comparison, the prescaler = 256 case (PER = 439) produced a half-period of 55.035 ms, with an error of 0.07%.
b. See the **lab2_3a** pseudocode and program code in the relevant sections.
c. See the appendix for relevant screenshots of the waveforms.

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].

a. See the **lab2_3Watch** pseudocode and program code in the relevant sections.
b. See the appendix for relevant screenshots for this exercise.

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

a. It should not be necessary to debounce (nor wait for the release of) the tactile switch S2 on the OOTB MB since that switch is used to stop the animation and return to EDIT mode; if there is bouncing and this instruction repeats, it will have no impact on the function of the program since it will just stay in EDIT mode.
b. It should not be necessary to debounce (nor wait for the release of) the tactile switch S1 on the SLB since this switch is only used to enter PLAY mode; once again, if there is bouncing and this instruction repeats, it will not have an impact on the function of the program since it will just stay in PLAY mode.

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

a. The program is designed so that pressing S2 on the SLB backpack stores the current frame into memory, so if S2 bounces when you press it, it could store that one frame several times in memory, causing your animation to repeat frames you did not intend to repeat.

PSEUDOCODE/FLOWCHARTS

SECTION 1

Part 1 Pseudocode

Lab2_1

```
;assembler directives

;define constants


;start of program

MAIN:

;initialize necessary ports for switches/LEDs

;clear PORTA (switches) direction to inputs

;initialize PORTC (LEDs) to output 1 (off)

;set PORTC (LEDs) direction to outputs


;start of infinite loop to read switches

LOOP:

;read PORTA

;turn on corresponding bits in the LEDS in PORTC


;Jump back to LOOP to repeat forever
```

Part 2:

Lab2_2

```
;assembler directives

;define constants



;Initialize the stack pointer to 0x3FFF (low byte first)
        ;load low byte into CPU_SPL
        ;load high byte into CPU_SPH


;start program
;MAIN:
;toggle output of I/O port pin
;Call Delay_10MS
;repeat



Delay_10MS:
;push registers


;
;run subroutine
;cycle needed is 10 milliseconds / 0.5 microseconds = 20,000 cycles
;innerLoopVal * OuterLoopVal * cycle instructions = 20,000
;load innerLoopVal into r16
;load outerLoopVal into r17


Outer loop:
        ;load inner counter                              1 cycle
Inner loop:
        ;Decrement innerLoop                                  1 cycle
        ;branch to outer loop once inner cycles have completed   2 cycles to branch / 1 cycle to fall through
        ;decrement outer loop                                  1 cycle
        ;exit full loop when outer cycles have completed       2 cycles to branch / 1 cycle to fall through
                ;n = num inner loop, m = num outer loops
                ;total clock cycles is roughly M * (3n + 3) cycles
                        ;so if I make m = 200, and n = 32 there should be around 20,000 cycles


;pop registers
;Return to main


DELAY_X_10MS:
;push registers (use r18 for X parameter)


;load X into R18


;loop X times
DelayXloop:
        ;Branch to DONE if R18 is zero
        ;Call Delay_10MS
        ;decrement r18
        ;jump to DelayXLoop


DONE:
;Pop registers
;Return to main
```

Part 3:

Part 3.1

Lab2_3

```
;assembler directives

;define constants

;fcpu

;PRE

;D

:recipD


.org 0x00

        rjmp MAIN


.org 0x100

;start program

MAIN:

;initialize stack to 0x3FFF


;initialize portC_0 direction to out

;set portC_Out to high


;initialize timer/counter register to 0

;set time period register (low byte first) to correspond to 55ms

;PER = dur *  (fclk / PRE)

; PER = fcpu / PRE / recipD


;clear pending flags

;start TCC0 with prescaler 256
```

```
WAIT_OVF:
;load FLAG register into r16
;If OVFIF (bit 0) is clear jump to WAIT_OVF

;else clear OVFIF
;Toggle probe pin
; go to WAIT_OVF
```

Part 3a

Lab2_3a

```
;assembler directives
;define constants
;fcpu
;PRE
;D
:recipD

.org 0x00
        rjmp MAIN

.org 0x100
;start program
MAIN:
;initialize stack to 0x3FFF

;initialize portC_0 direction to out
;set portC_Out to high

;initialize timer/counter register to 0
```

```
;set time period register (low byte first) to correspond to 55ms
;PER = dur *  (fclk / PRE)
; PER = fcpu / PRE / recipD = 13750

;clear pending flags
;start TCC0 with prescaler 8

WAIT_OVF:
;load FLAG register into r16
;If OVFIF (bit 0) is clear jump to WAIT_OVF

;else clear OVFIF
;Toggle probe pin
; go to WAIT_OVF
```

PART 3b – Watch

Lab2_3Watch

```
;assembler directives
;define constants
;create addresses for seconds and minutes counter and allocate a byte

.org 0
rjmp MAIN

.org 0x100
MAIN:

;initialize stack
```

```
;initialize timer / call TC_INIT to make the timer period 1 second


;start loop


minLoop:
        ;load flag register into general purpose register
        ;if OVFIF (bit 0) is clear than jmp back to minLoop
        ;else seconds++
        ;clear OVFIF
        ;compare seconds to 60
        ;if equal minutes++ and clr seconds
        ;jump to minLoop



; 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
; recover relevant registers
; return from subroutine
        ret
```

USED THE SKELETON FOR PART 4

PROGRAM CODE

SECTION 2

Part 1:

Lab2_1

```
;
; lab2.asm
;
; Created: 9/16/2025 8:28:10 PM
; Author : arist
;

;Lab 2, Section 22
;Name: Arion Stern
;Class #: 11303
;PI Name: Ian Santamauro
; Description: Program to continually read DIP switch values
;              from PORTA and output them to the corresponding
;              LEDs on PORTC of the OOTB Switch & LED Backpack.

; Replace with your application code

.org 0x0000
    rjmp MAIN

.org 0x100
MAIN:
;initialize necessary ports for switches/LEDs
;clear PORTA (switches) direction to inputs
ldi r16, 0xFF
sts PORTA_DIRCLR, r16

;initialize PORTC (LEDs) to output 1 (off)
sts PORTC_OUTSET, r16

;set PORTC (LEDs) direction to outputs
sts PORTC_DIRSET, r16


;start of infinite loop to read switches
LOOP:
;read port A
lds r17, PORTA_IN
;turn on corresponding bits in the LEDS
sts PORTC_OUt, r17

;Jump back to LOOP to repeat forever
rjmp LOOP
```

Part 2:

Lab2_2

```
/*
 * lab2_2.asm
 *
 *  Created: 9/17/2025 11:11:28 PM
 *   Author: arist
 */

; Lab 2, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Program implementing a 10 ms software delay subroutine
;              (and multiples), verified using the DAD Scope.
;              Includes toggling PORTC pin 0

 .include "ATxmega128A1Udef.inc"


;assembler directives
;define constants


.org 0x00
       rjmp MAIN





.org 0x100
;start program
MAIN:

;Initialize the stack pointer to 0x3FFF (low byte first)
       ;load low byte into CPU_SPL
       ldi r16, low(0x3FFF)
       out CPU_SPL, r16
       ;load high byte into CPU_SPH
       ldi r16, high(0x3FFF)
       out CPU_SPH, r16

;set direction and out register for output pin (PortC pin 0)
ldi r16, (1<<0)
sts PORTC_DIRSET, r16
sts PORTC_OUTSET, r16

;toggle output of I/O port pin
ToggleLoop:
       sts PORTC_OUTTGL, r16
       ;Call Delay_10MS
       ;rcall delay_10MS

       ;load X into r18 and call delay_X_10ms
       ldi r18, 4
       rcall Delay_X_10MS
       ;repeat
       rjmp ToggleLoop

;****************************************************
; Name: DELAY_10MS
; Purpose: Generate a 10 millisecond delay using
;          nested software loops (at 2 MHz clock).
; Input(s): None
; Output(s): None
; Registers Affected: r16, r17 (protected with push/pop)
;****************************************************

Delay_10MS:
;push registers
       push r16
       push r17
;
;run subroutine
;cycle needed is 10 milliseconds / 0.5 microseconds = 20,000 cycles
;innerLoopVal * OuterLoopVal * cycle instructions = 20,000
;load innerLoopVal into r16
;load outerLoopVal into r17


ldi r17, 200

Outerloop:
       ;load inner counter                                    1 cycle
       ldi r16, 33
InnerLoop:
       ;Decrement innerLoop                            1 cycle
       dec r16
       ;branch to inner loop until inner cycles have completed2 cycles to branch / 1 cycle to fall
through
       brne InnerLoop
       ;decrement outer loop                                          1 cycle
       dec r17
       ;exit full loop when outer cycles have completed       2 cycles to branch / 1 cycle to fall
through
       brne OuterLoop


;n = num inner loop, m = num outer loops
;total clock cycles is roughly M * (3n + 3) cycles
;so if I make m = 200, and n = 32 there should be around 20,000 cycles


;pop registers
       pop r17
       pop r16

       ret

       ;19.373 at 200, 32 = 3.135% error
       ;19.956 at 200, 33 = 0.22% error

;****************************************************
; Name: DELAY_X_10MS
; Purpose: Delay for X multiples of 10 ms by calling
;          DELAY_10MS repeatedly.
; Input(s): r18 = number of 10 ms intervals
; Output(s): None
; Registers Affected: r18 (protected with push/pop)
;****************************************************

DELAY_X_10MS:
;push registers (use r18 for X parameter)
push r18

;loop X times
DelayXloop:
       ;Branch to DONE if R18 is zero
       tst r18
       BREQ DONE
       ;Call Delay_10MS
       rCall DELAY_10MS
       ;decrement r18
       dec r18
       ;jump to DelayXLoop
       rjmp DelayXLoop

DONE:
;Pop registers
       pop r18
;Return to main
       ret
```

Part 3:

Lab2_3

```
/*
 * lab2_3.asm
 *
 *  Created: 9/20/2025 7:24:58 PM
 *   Author: arist
 */

; Lab 2, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Program using Timer/Counter TCC0 with prescaler 256
;              to toggle PORTC pin 0 every 55 ms. Tested with
;              different period values to minimize timing error.

;assembler directives
;define constants
.equ fcpu = 2000000
.equ pre = 256
.equ D = 55/1000
.equ recipD = 1000/55


.org 0x00
       rjmp MAIN


.org 0x100
;start program
```

```
MAIN:

;initialize stack to 0x3FFF
ldi r16, low(0x3FFF)
out CPU_SPL, r16

ldi r16, high(0x3FFF)
out CPU_SPH, r16


;initialize portC_0 direction to out
;set portC_Out to low
ldi r16, 1
sts PORTC_OUTCLR, r16
sts PORTC_DIRSET, r16


;initialize timer/counter register to 0
;set time period register (low byte first) to correspond to 55ms
;PER = dur *  (fclk / PRE)
; PER = fcpu / PRE / recipD = 429 because decimal is truncated
sts TCC0_CTRLA, r16

sts TCC0_CNT, r16
sts TCC0_CNT+1, r16

;ldi r16, low(fcpu/PRE/recipD)
ldi r16, low(439)
sts TCC0_PER, r16

;ldi r16, high(fcpu/PRE/recipD)
ldi r16, high(439)
sts TCC0_PER+1, r16

;clear pending flags
;start TCC0 with prescaler 256
ser r16
sts TCC0_INTFLAGS, r16

ldi r16, TC_CLKSEL_DIV256_gc
sts TCC0_CTRLA, r16

WAIT_OVF:
;load FLAG register into r16
;If OVFIF (bit 0) is clear jump to WAIT_OVF
lds r16, TCC0_INTFLAGS
sbrs r16, TC0_OVFIF_bp
rjmp WAIT_OVF

;else clear OVFIF
ldi r17, TC0_OVFIF_bm
sts TCC0_INTFLAGS, r17

;Toggle probe pin
ldi r18, 1
sts PORTC_OUTTGL, r18

; go to WAIT_OVF
rjmp WAIT_OVF
```

Lab2_3a

```
/*
 * lab2_3a.asm
 *
 *  Created: 9/20/2025 11:40:29 PM
 *   Author: arist
 */

; Lab 2, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Program using Timer/Counter TCC0 with prescaler 8
;              to toggle PORTC pin 0 every 55 ms. Used to compare
;              timing accuracy with the prescaler 256 implementation.


;assembler directives
;define constants
.equ fcpu = 2000000
.equ pre = 256
.equ D = 55/1000
.equ recipD = 1000/55


.org 0x00
       rjmp MAIN


.org 0x100
;start program
MAIN:

;initialize stack to 0x3FFF
ldi r16, low(0x3FFF)
out CPU_SPL, r16

ldi r16, high(0x3FFF)
out CPU_SPH, r16


;initialize portC_0 direction to out
;set portC_Out to low
ldi r16, 1
sts PORTC_OUTCLR, r16
sts PORTC_DIRSET, r16


;initialize timer/counter register to 0
;set time period register (low byte first) to correspond to 55ms
;PER = dur *  (fclk / PRE)
; PER = fcpu / PRE / recipD = 13750
sts TCC0_CTRLA, r16

sts TCC0_CNT, r16
sts TCC0_CNT+1, r16
```

```
;ldi r16, low(fcpu/PRE/recipD)
ldi r16, low(14060)
sts TCC0_PER, r16

;ldi r16, high(fcpu/PRE/recipD)
ldi r16, high(14060)
sts TCC0_PER+1, r16

;clear pending flags
;start TCC0 with prescaler 8
ser r16
sts TCC0_INTFLAGS, r16

ldi r16, TC_CLKSEL_DIV8_gc
sts TCC0_CTRLA, r16

WAIT_OVF:
;load FLAG register into r16
;If OVFIF (bit 0) is clear jump to WAIT_OVF
lds r16, TCC0_INTFLAGS
sbrs r16, TC0_OVFIF_bp
rjmp WAIT_OVF

;else clear OVFIF
ldi r17, TC0_OVFIF_bm
sts TCC0_INTFLAGS, r17

;Toggle probe pin
ldi r18, 1
sts PORTC_OUTTGL, r18

; go to WAIT_OVF
rjmp WAIT_OVF
```

Lab2_3Watch

```
/*
 * lab2_3Watch.asm
 *
 *  Created: 9/21/2025 12:51:41 AM
 *   Author: arist
 */

 ; Lab 2, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Program using Timer/Counter TCC0 with prescaler 64
;              to implement a simple minute-tracking watch. The
;              program counts seconds and increments a minute
;              counter stored in data memory.

;PART 3.2 - Watch
;assembler directives
;define constants
.include "ATxmega128A1Udef.inc"

;.equ seconds = 0x2000
;.equ minutes = 0x2002

;create addresses for seconds and minutes counter and allocate a byte
.dseg
.org 0x2000
Seconds: .byte 1
Minutes: .byte 1


.cseg

.org 0
rjmp MAIN

.org 0x100
MAIN:


;initialize stack
ldi r16, low(0x3FFF)
out CPU_SPL, r16

ldi r16, high(0x3FFF)
out CPU_SPH, r16

clr r17
clr r18
sts seconds, r17
sts minutes, r18

; initialize relevant I/O modules (switches and LEDs)
       rcall IO_INIT


;initialize timer / call TC_INIT to make the timer period 1 second
       rcall TC_INIT
```

```
;start loop


minLoop:
       ;load flag register into general purpose register
       lds r16, TCC0_INTFLAGS
       sbrs r16, TC0_OVFIF_bp
       ;if OVFIF (bit 0) is clear than jmp back to minLoop
       rjmp minLoop

       ;clear OVFIF
       ldi r16, (TC0_OVFIF_bm)
       sts TCC0_INTFLAGS, r16
       ;else seconds++
       inc r17


       ldi r16, 1
       sts PORTC_OUTTGL, r16
       ;compare seconds to 60
       CPI r17, 60
       BRNE not60
       ;if equal minutes++ and clr seconds
       clr r17
       inc r18
       ldi r16, (1<<1)
       sts PORTC_OUTTGL, r16

       not60:
       ;store seconds and minutes
       sts Seconds, r17
       sts Minutes, r18
       ;jump to minLoop
              rjmp minLoop


;****************************************************
; Name: IO_INIT
; Purpose: Initialize relevant I/O modules.
;          Configures PC0 and PC1 as outputs for
;          toggling each second and each minute.
; Input(s): None
; Output(s): None
; Registers Affected: r16
;****************************************************
IO_INIT:
; protect relevant registers
       push r16


; initialize the relevant I/O
       ldi r16, 1
       sts PORTC_OUTCLR, r16
       sts PORTC_DIRSET, r16

       ; IO_INIT: make PC1 an output too
       ldi r16, (1<<0)|(1<<1)
       sts PORTC_DIRSET, r16
       sts PORTC_OUTSET, r16
```

```
; recover relevant registers
       pop r16

; return from subroutine
       ret


;****************************************************
; Name: TC_INIT
; Purpose: Configure Timer/Counter TCC0 to generate
;          an overflow every 1 second.
; Input(s): None
; Output(s): None
; Registers Affected: r16
;****************************************************
TC_INIT:
; protect relevant registers
       push r16


; initialize the relevant TC modules
;clear timer counter registers
ldi r16, 0
sts TCC0_CTRLA, r16

sts TCC0_CNT, r16
sts TCC0_CNT + 1, r16

;set period to have second value
; per = Fclk / PRE / (1/D) = 62500

ldi  r16, low(31951)
sts TCC0_PER, r16
ldi  r16, high(31951)
sts TCC0_PER+1, r16

;clear OVFIF FLAG
ldi r16, 1
sts TCC0_INTFLAGS, r16


;start timer: prescaler 64

ldi r16, TC_CLKSEL_DIV64_gc
sts TCC0_CTRLA, r16


; recover relevant registers
pop r16

; return from subroutine
       ret
```

Part 4:

Lab2_4

```
/*
 * lab2_4.asm
 *
 *  Created: 9/21/2025 2:17:42 PM
 *   Author: arist
 */

; Lab 2, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: LED animation creator program. Allows frame storage in EDIT mode
                    ;  (with debounced S2 on SLB), plays stored frames sequentially at 5 Hz in
                    ;  PLAY mode (triggered by S1 on SLB), and stops playback when S2 on the Memory Base
is pressed.


 ;****************************************************
;  File name: lab2_f25_4_skeleton.asm
;  Author: Christopher Crary
;  Last Modified By: Dr. Schwartz
;  Last Modified On: 21 Aug 2025
;  Purpose: To allow LED animations to be created with the OOTB uPAD,
;                    OOTB SLB, and OOTB MB.
;
;  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.
.include "ATxmega128a1udef.inc"
;*******END OF INCLUDES******************************

;*******DEFINED SYMBOLS******************************
.equ ANIMATION_START_ADDR   =      0x2000 ;useful, but not required
.equ ANIMATION_SIZE                =      0x2000 ;useful, but not required
.equ ledPortOUT = PORTC_OUT
.equ ledPortDIR = PORTC_DIR
.equ dipPortIN = PORTA_IN
.equ tacSLBPortIN = PORTF_IN
.equ sw_EDITbm = (1<<3) ; port f slb s2
.equ sw_PLAYbm = (1<<2) ; port f slb s1

.equ MBPortIN = PORTE_IN
.equ mS1bm = (1<<0)
.equ mS2bm = (1<<1) ; port E MB animation stop return to EDIT

.equ dClk = 440 ;pre = 64

;*******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
; upon system reset, jump to main program (instead of executing
; instructions meant for interrupt vectors)
.org 0x00
       rjmp MAIN

; place the main program somewhere after interrupt vectors (ignore for now)
.org 0x100          ; >= 0xFD
MAIN:
; initialize the stack pointer
ldi r16, low(0x3FFF)
out CPU_SPL, r16
ldi r16, high(0x3FFF)
out 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

;X for storing (points to next free slot)
;Y for playing back (points to frame currently being played)
ldi XL, low(ANIMATION_START_ADDR)
ldi XH, high(ANIMATION_START_ADDR)

movw Y, X


; 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
andi r16, sw_PLAYbm

; If it is determined that relevant switch was pressed,
; go to "PLAY" mode.
BREQ 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 r16, PORTF_IN
andi r16, sw_EDITbm
BRNE 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.)
ldi r16, TC0_OVFIF_bm
sts TCC0_INTFLAGS, r16

ldi r16,  TC_CLKSEL_DIV64_gc
sts TCC0_CTRLA, r16

DB_WAIT:
lds r17, TCC0_INTFLAGS
sbrs r17, TC0_OVFIF_bp
rjmp DB_WAIT




; 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".

;reset OVFIF
ldi r16, (TC0_OVFIF_bm)
sts TCC0_INTFLAGS, r16
;stop debounce timer
ldi r16, 0x00
sts TCC0_CTRLA, r16

lds r16, PORTF_IN
andi r16, sw_EDITbm
BRNE EDIT

;store current DIP val to animation SRAM
lds r18, PORTA_IN
st X+, r18
```

```
; Wait for the "STORE FRAME" switch to be released
; before jumping to "EDIT".
STORE_FRAME_SWITCH_RELEASE_WAIT_LOOP:
       lds r16, PORTF_IN
       andi r16, sw_EDITbm
       breq STORE_FRAME_SWITCH_RELEASE_WAIT_LOOP

       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)


PLAY_LOOP:

; Check if it is intended that "EDIT" mode be started
; i.e., check if the relevant switch has been pressed.`
lds r16, PORTE_IN
andi r16, ms2bm


; If it is determined that relevant switch was pressed,
; go to "EDIT" mode.
BREQ 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
cpc XH, YH

; 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 r18, Y+
sts PORTC_OUT, r18

ldi r16, TC1_OVFIF_bm
sts TCC1_INTFLAGS, r16
```

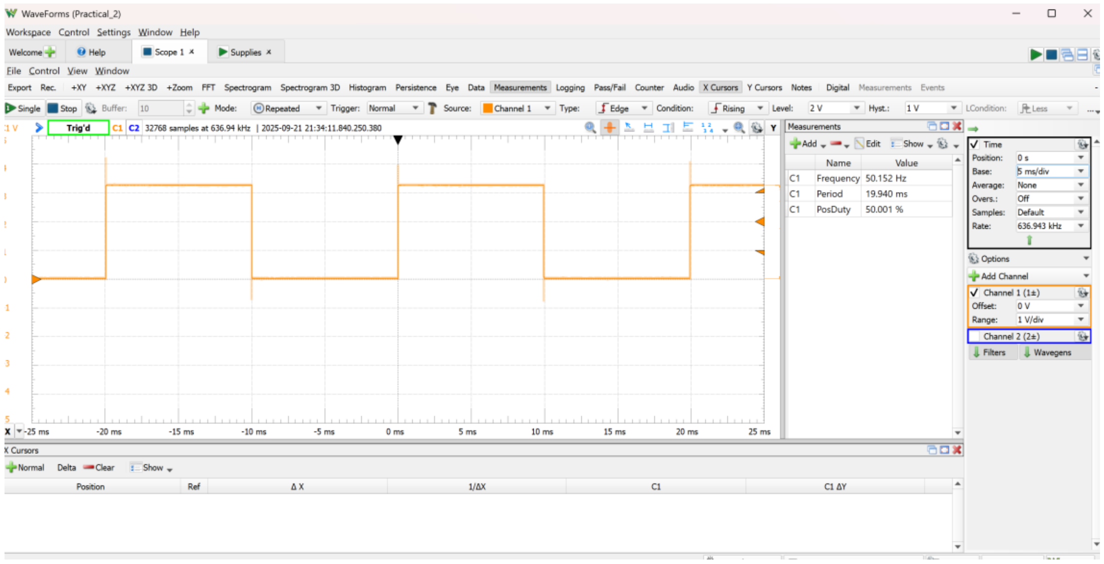
```
ldi r16, TC_CLKSEL_DIV64_gc
sts TCC1_CTRLA, r16


WAIT_FRAME:
       lds r17, TCC1_INTFLAGS
       sbrs r17, TC1_OVFIF_bp
       rjmp WAIT_FRAME

       ;stop and clear timer
       ldi r16, TC1_OVFIF_bm
       sts TCC1_INTFLAGS, r16

       ldi r16, 0x00
       sts TCC1_CTRLA, r16

       rjmp PLAY_LOOP


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

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

;***************************************************
; Name: IO_INIT
; Purpose: Initialize all relevant I/O modules:
;          - Configure PORTC (LEDs) as outputs (off)
;          - Configure PORTA (DIP switches) as inputs
;          - Configure SLB switches (EDIT=S2, PLAY=S1) as inputs
;          - Configure MB switch (stop return to EDIT) as input
; Input(s): None
; Output(s): None
; Registers Affected: r16
;***************************************************
IO_INIT:
; protect relevant registers
push r16

; initialize the relevant I/O

;leds to output off
ldi r16, 0xFF
sts PORTC_OUT, r16
sts PORTC_DIR, r16

;dip to IN
sts PORTA_DIRCLR, r16

; SLB tac to IN
ldi r16, (sw_EDITbm | sw_PLAYbm)
sts PORTF_DIRCLR, r16

ldi r16, (mS1bm | mS2bm)
sts PORTE_DIRCLR, r16


; recover relevant registers
```

```
       pop r16

; return from subroutine
       ret
;***************************************************
; Name: TC_INIT
; Purpose: Initialize Timer/Counter modules:
;          - TCC0 → debounce timer (~450 cycles at prescaler 64)
;          - TCC1 → frame timer (~6390 cycles at prescaler 64,
;                    5 Hz frame rate)
; Input(s): None
; Output(s): None
; Registers Affected: r16
;***************************************************
TC_INIT:
; protect relevant registers
push r16

; initialize the relevant TC modules

;TCC0 for DEBOUNCE TIMER
ldi r16, 0x00
sts TCC0_CTRLA, r16

sts TCC0_CNT, r16
sts TCC0_CNT+1, r16

ldi r16, low(450)
sts TCC0_PER, r16
ldi r16, high(450)
sts TCC0_PER+1, r16

ldi r16, TC0_OVFIF_bm
sts TCC0_INTFLAGS, r16

;TCC1 for FRAME TIMER

ldi r16, 0x00
sts TCC1_CTRLA, r16

sts TCC1_CNT, r16
sts TCC1_CNT+1, r16

ldi r16, low(6390)
sts TCC1_PER, r16
ldi r16, high(6390)
sts TCC1_PER+1, r16

ldi r16, TC1_OVFIF_bm
sts TCC1_INTFLAGS, r16



; recover relevant registers
pop r16

; return from subroutine
       ret

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

APPENDIX

Supporting ASM/C Code and Additional Information

- Included/Referenced Files and Headers:
  - ATxmega128a1udef.inc (standard device definition file provided by Microchip)
- Additional Screenshots/Images:



WaveForms (Practical_2)

Mode: Repeated | Trigger: Normal | Source: Channel 1 | Type: Edge | Condition: Rising | Level: 2 V | Hyst.: 1 V

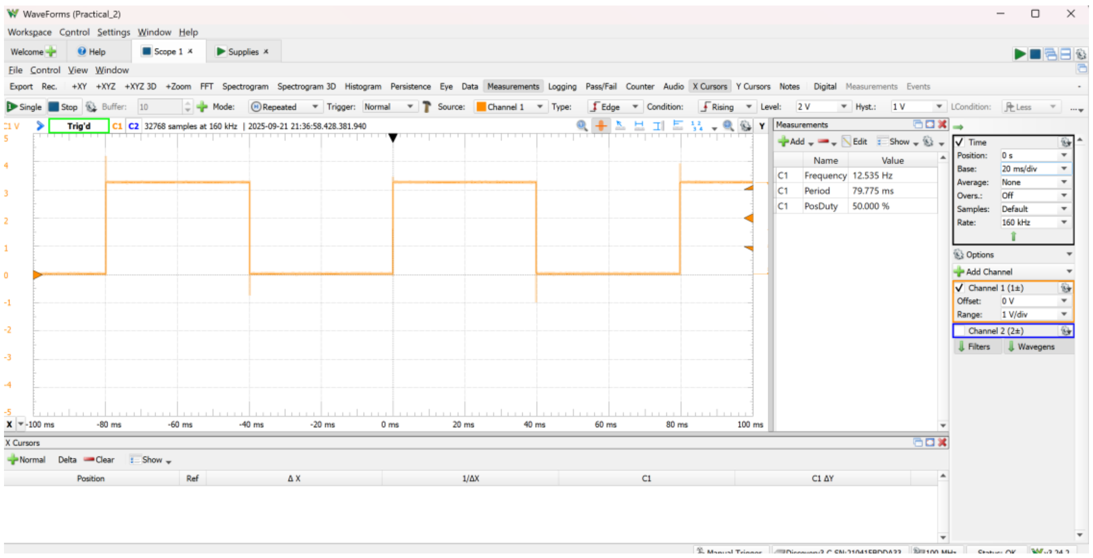
Trig'd | 32768 samples at 636.94 kHz | 2025-09-21 21:34:11.840.250.380

	Name	Value
C1	Frequency	50.152 Hz
C1	Period	19.940 ms
C1	PosDuty	50.001 %

Time — Position: 0 s; Base: 5 ms/div; Average: None; Overs.: Off; Samples: Default; Rate: 636.943 kHz

Channel 1 (1±) — Offset: 0 V; Range: 1 V/div

Figure 1: Scope capture of PORTC pin 0 toggling using the software delay subroutine (Delay_10MS). Measured period = 19.94 ms, expected = 20.0 ms. Error = 0.3%.



WaveForms (Practical_2)

Mode: Repeated | Trigger: Normal | Source: Channel 1 | Type: Edge | Condition: Rising | Level: 2 V | Hyst.: 1 V

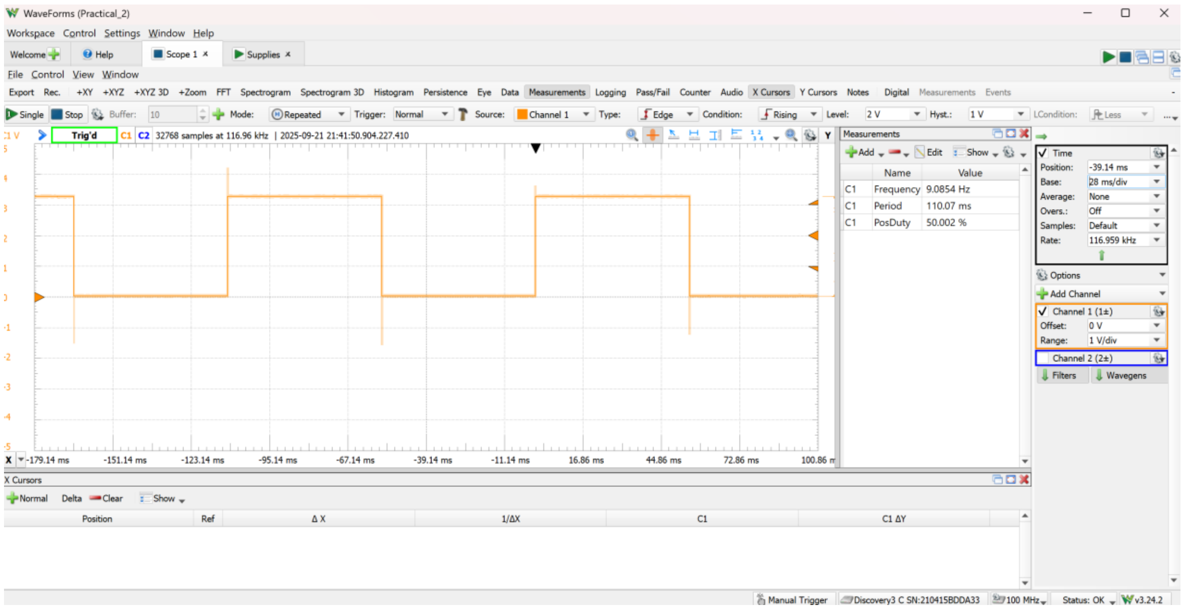
Trig'd | 32768 samples at 160 kHz | 2025-09-21 21:36:58.428.381.940

	Name	Value
C1	Frequency	12.535 Hz
C1	Period	79.775 ms
C1	PosDuty	50.000 %

Time — Position: 0 s; Base: 20 ms/div; Average: None; Overs.: Off; Samples: Default; Rate: 160 kHz

Channel 1 (1±) — Offset: 0 V; Range: 1 V/div

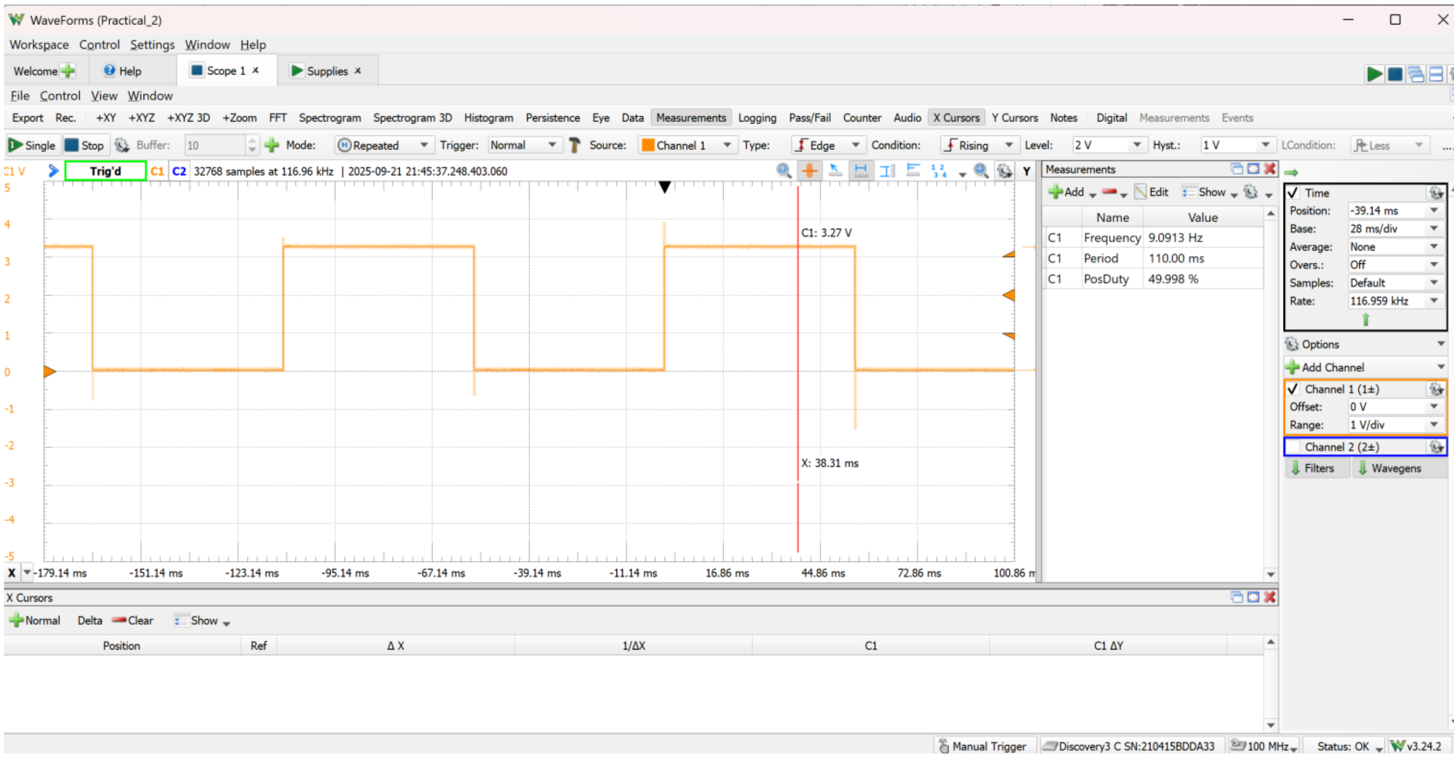
Figure 2: Scope capture of PORTC pin 0 toggling at 12.5 Hz using the software delay subroutine (Delay_X_10MS, X=4). Measured period = 79.775 ms, expected = 80.0 ms. Error = 0.28%.



	Name	Value
C1	Frequency	9.0854 Hz
C1	Period	110.07 ms
C1	PosDuty	50.002 %

Time Position: -39.14 ms; Base: 28 ms/div; Average: None; Overs.: Off; Samples: Default; Rate: 116.959 kHz
Channel 1 (1±) Offset: 0 V; Range: 1 V/div
Mode: Repeated; Trigger: Normal; Source: Channel 1; Type: Edge; Condition: Rising; Level: 2 V; Hyst.: 1 V

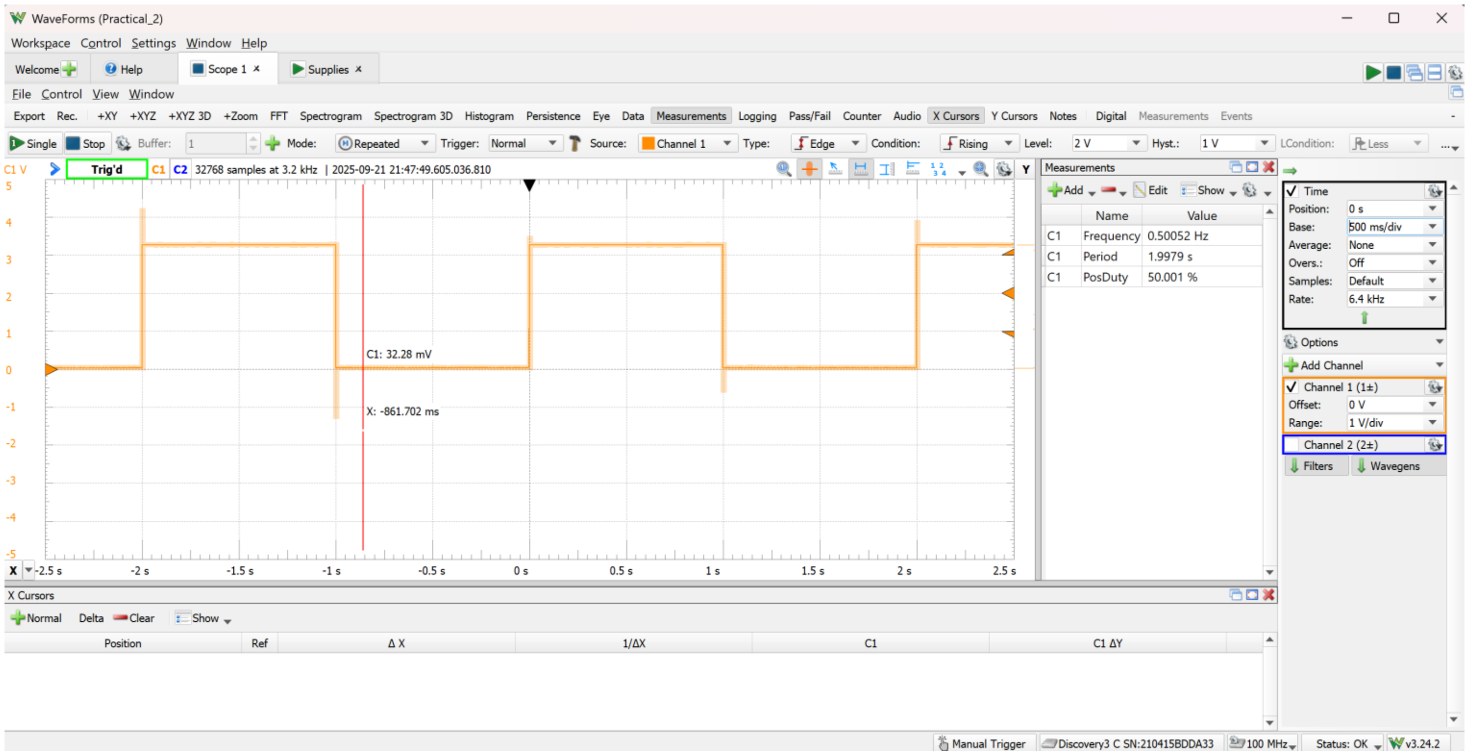
Figure 3: Scope capture of PORTC pin 0 toggling with Timer/Counter TCC0, prescaler = 256, PER = 439. Measured half-period = 55.035 ms, expected = 55.00 ms. Error = 0.06%.



	Name	Value
C1	Frequency	9.0913 Hz
C1	Period	110.00 ms
C1	PosDuty	49.998 %

Time Position: -39.14 ms; Base: 28 ms/div; Average: None; Overs.: Off; Samples: Default; Rate: 116.959 kHz
Channel 1 (1±) Offset: 0 V; Range: 1 V/div
Mode: Repeated; Trigger: Normal; Source: Channel 1; Type: Edge; Condition: Rising; Level: 2 V; Hyst.: 1 V

Figure 4: Scope capture of PORTC pin 0 toggling with Timer/Counter TCC0, prescaler = 8, PER = 14,071. Measured half-period = 55.00 ms, expected = 55.00 ms. Error = 0%. The smaller prescaler provides finer PER resolution, resulting in improved timing accuracy compared to prescaler = 256



Mode: Repeated | Trigger: Normal | Source: Channel 1 | Type: Edge | Condition: Rising | Level: 2 V | Hyst.: 1 V

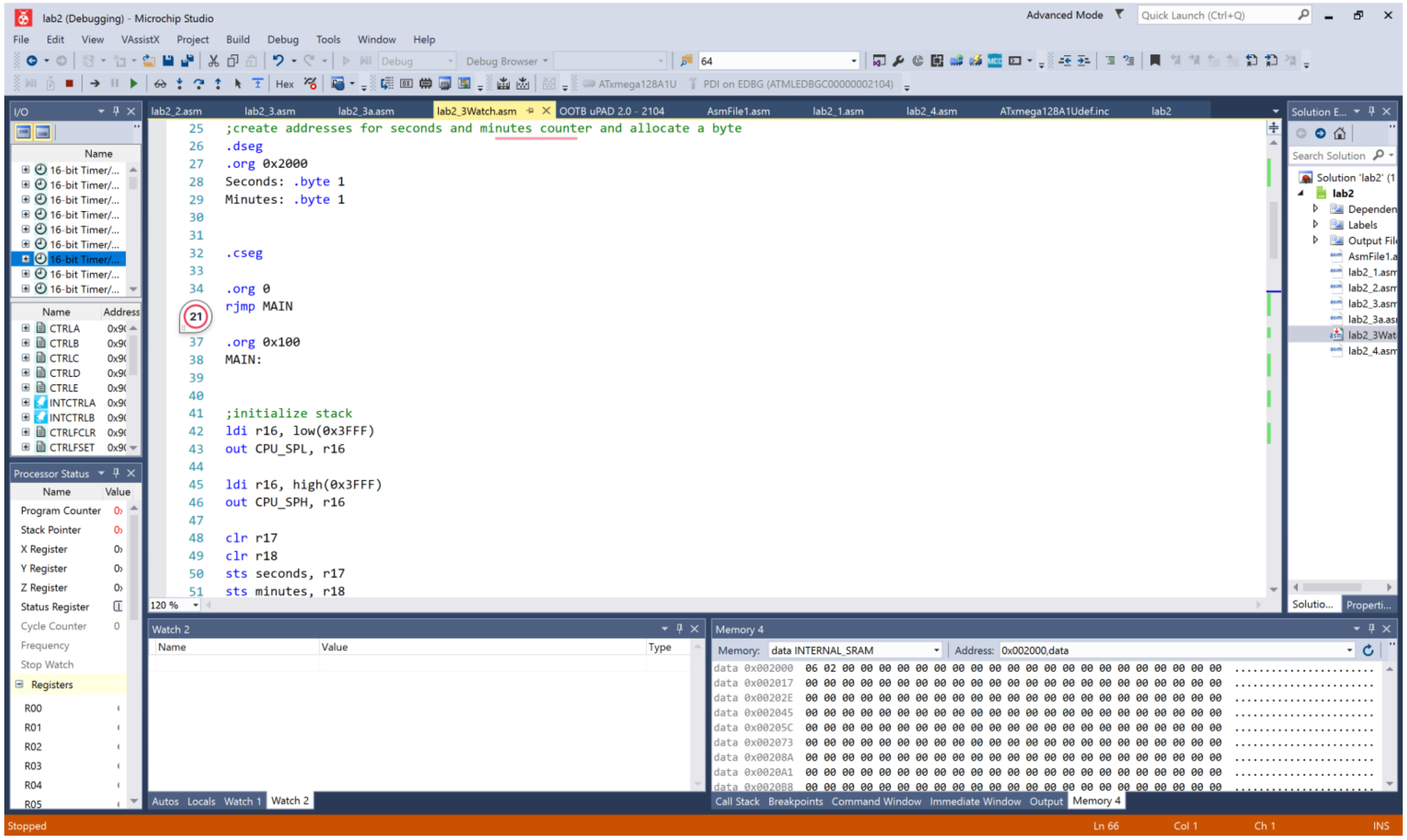
32768 samples at 3.2 kHz | 2025-09-21 21:47:49.605.036.810

	Name	Value
C1	Frequency	0.50052 Hz
C1	Period	1.9979 s
C1	PosDuty	50.001 %

Time — Position: 0 s; Base: 500 ms/div; Average: None; Overs.: Off; Samples: Default; Rate: 6.4 kHz

Channel 1 (1±) — Offset: 0 V; Range: 1 V/div

Figure 5: Scope capture of PORTC pin 0 toggling once per second (2 seconds because of toggle) using Timer/Counter TCC0 (prescaler = 64). Measured period = 1.9979 s, expected = 2.000 s. Error = 0.105. The minute-hand output on PORTC pin 1 toggles once per 60 seconds (not shown).

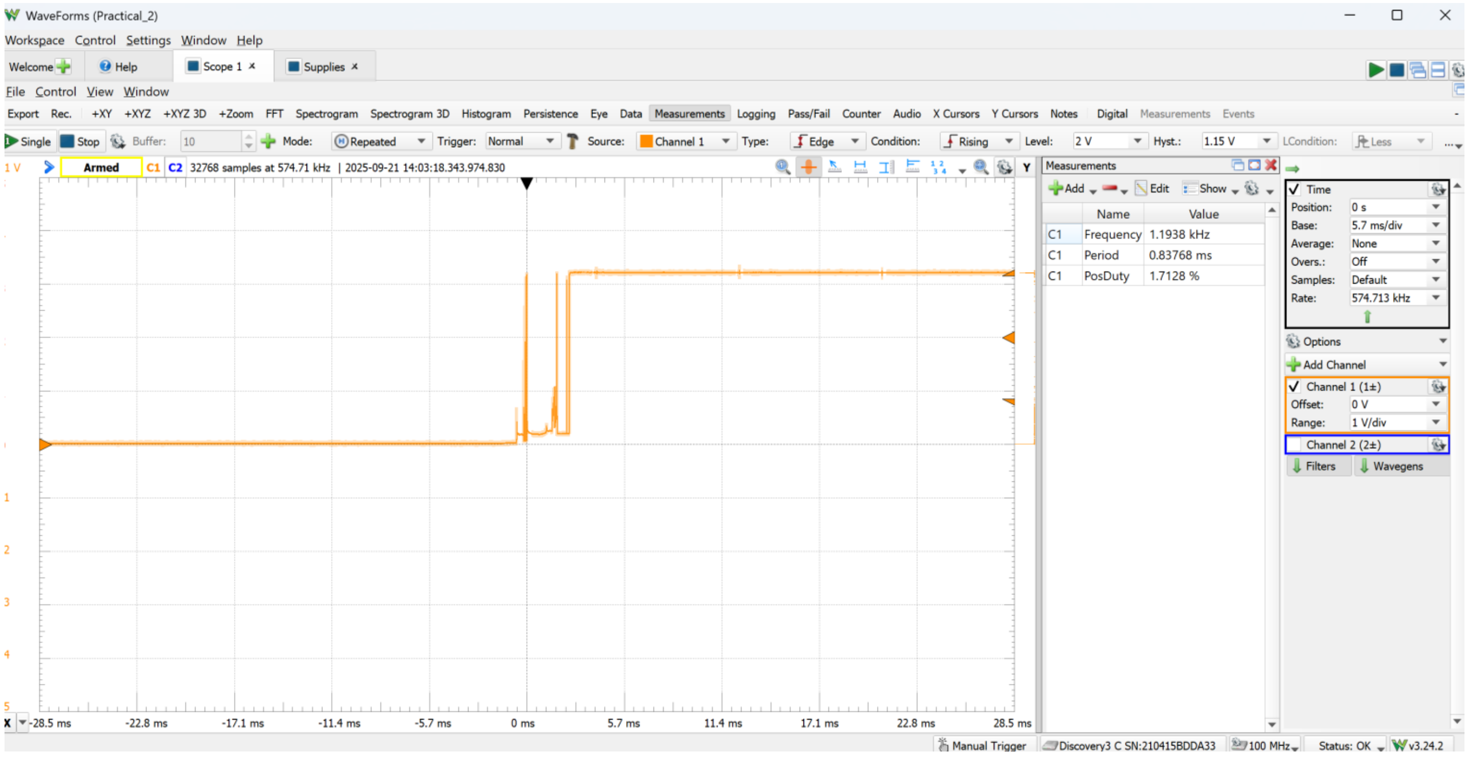

lab2 (Debugging) - Microchip Studio
lab2_2.asm lab2_3.asm lab2_3a.asm lab2_3Watch.asm OOTB uPAD 2.0 - 2104 AsmFile1.asm lab2_1.asm lab2_4.asm ATxmega128A1Udef.inc lab2
ATxmega128A1U
PDI on EDBG (ATMLEDBGC00000002104)
Watch 2
Memory 4
Stopped
Ln 66 Col 1 Ch 1 INS

```
;create addresses for seconds and minutes counter and allocate a byte
.dseg
.org 0x2000
Seconds: .byte 1
Minutes: .byte 1


.cseg

.org 0
rjmp MAIN

.org 0x100
MAIN:


;initialize stack
ldi r16, low(0x3FFF)
out CPU_SPL, r16

ldi r16, high(0x3FFF)
out CPU_SPH, r16

clr r17
clr r18
sts seconds, r17
sts minutes, r18
```

Memory: data INTERNAL_SRAM Address: 0x002000,data

```
data 0x002000  06 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .......................
data 0x002017  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .......................
data 0x00202E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .......................
data 0x002045  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .......................
data 0x00205C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .......................
data 0x002073  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .......................
data 0x00208A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .......................
data 0x0020A1  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .......................
data 0x0020B8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .......................
```

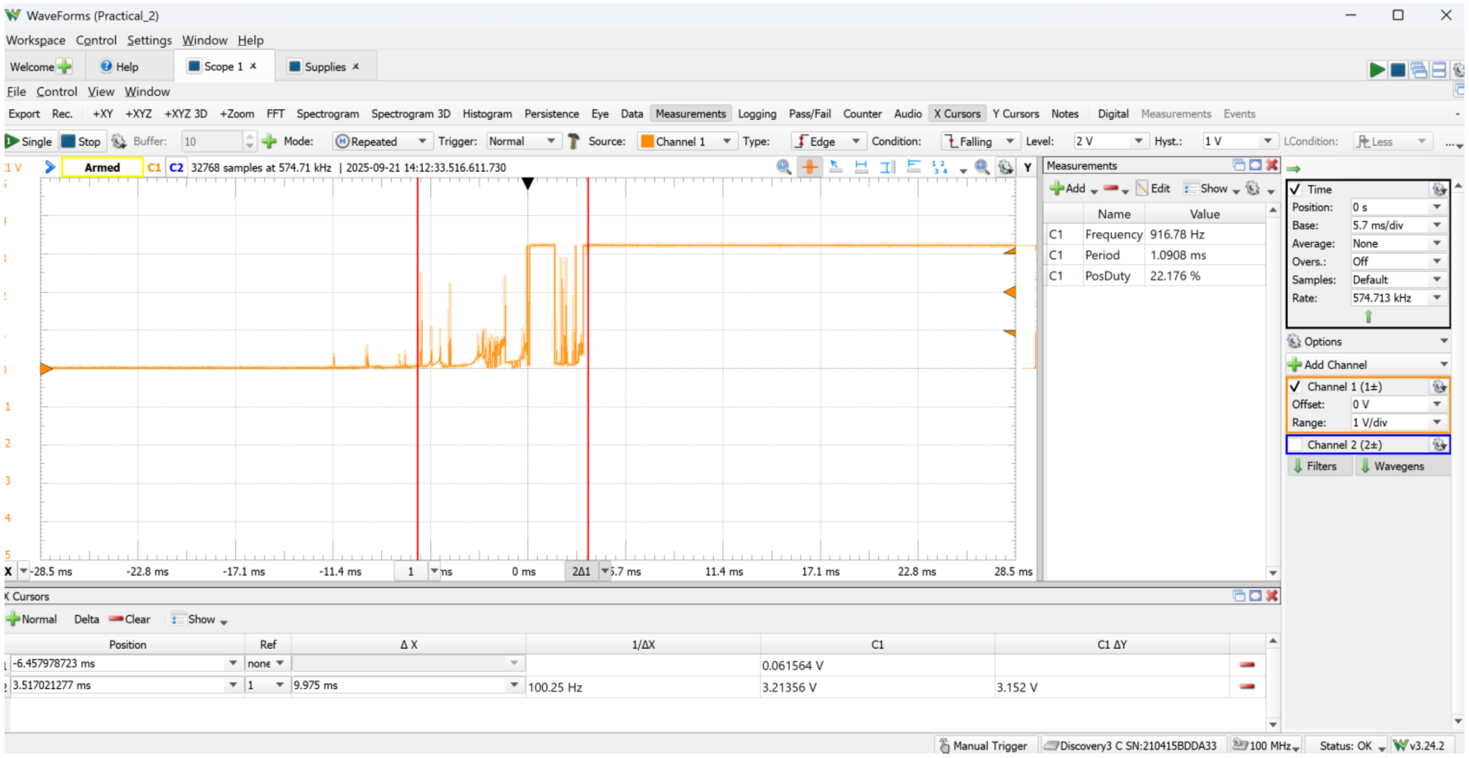
Figure 6: View of memory window showing the program keeping track of minutes and seconds


WaveForms (Practical_2)
Workspace Control Settings Window Help
Welcome Help Scope 1 Supplies
File Control View Window
Export Rec. +XY +XYZ +XYZ 3D +Zoom FFT Spectrogram Spectrogram 3D Histogram Persistence Eye Data Measurements Logging Pass/Fail Counter Audio X Cursors Y Cursors Notes Digital Measurements Events
Single Stop Buffer: 10 Mode: Repeated Trigger: Normal Source: Channel 1 Type: Edge Condition: Rising Level: 2 V Hyst.: 1.15 V LCondition: Less
Armed C1 C2 32768 samples at 574.71 kHz | 2025-09-21 14:03:18.343.974.830
-28.5 ms -22.8 ms -17.1 ms -11.4 ms -5.7 ms 0 ms 5.7 ms 11.4 ms 17.1 ms 22.8 ms 28.5 ms
Time Position: 0 s Base: 5.7 ms/div Average: None Overs.: Off Samples: Default Rate: 574.713 kHz
Options Add Channel
Channel 1 (1±) Offset: 0 V Range: 1 V/div
Channel 2 (2±)
Filters Wavegens
Manual Trigger Discovery3 C SN:210415BDDA33 100 MHz Status: OK v3.24.2

Measurements

	Name	Value
C1	Frequency	1.1938 kHz
C1	Period	0.83768 ms
C1	PosDuty	1.7128 %

Figure 7: Scope capture of tactile switch S2 (SLB, Store Frame) press, showing mechanical bouncing. Debouncing with Timer/Counter TCC0 ensures only one frame is stored per press.


WaveForms (Practical_2)
Workspace Control Settings Window Help
Welcome Help Scope 1 Supplies
File Control View Window
Export Rec. +XY +XYZ +XYZ 3D +Zoom FFT Spectrogram Spectrogram 3D Histogram Persistence Eye Data Measurements Logging Pass/Fail Counter Audio X Cursors Y Cursors Notes Digital Measurements Events
Single Stop
Armed
-28.5 ms -22.8 ms -17.1 ms -11.4 ms 0 ms 5.7 ms 11.4 ms 17.1 ms 22.8 ms 28.5 ms
Manual Trigger Discovery3 C SN:210415BDDA33 100 MHz Status: OK v3.24.2

Buffer: 10 | Mode: Repeated | Trigger: Normal | Source: Channel 1 | Type: Edge | Condition: Falling | Level: 2 V | Hyst.: 1 V | LCondition: Less

C1 C2 32768 samples at 574.71 kHz | 2025-09-21 14:12:33.516.611.730

Measurements

	Name	Value
C1	Frequency	916.78 Hz
C1	Period	1.0908 ms
C1	PosDuty	22.176 %

Time — Position: 0 s; Base: 5.7 ms/div; Average: None; Overs.: Off; Samples: Default; Rate: 574.713 kHz

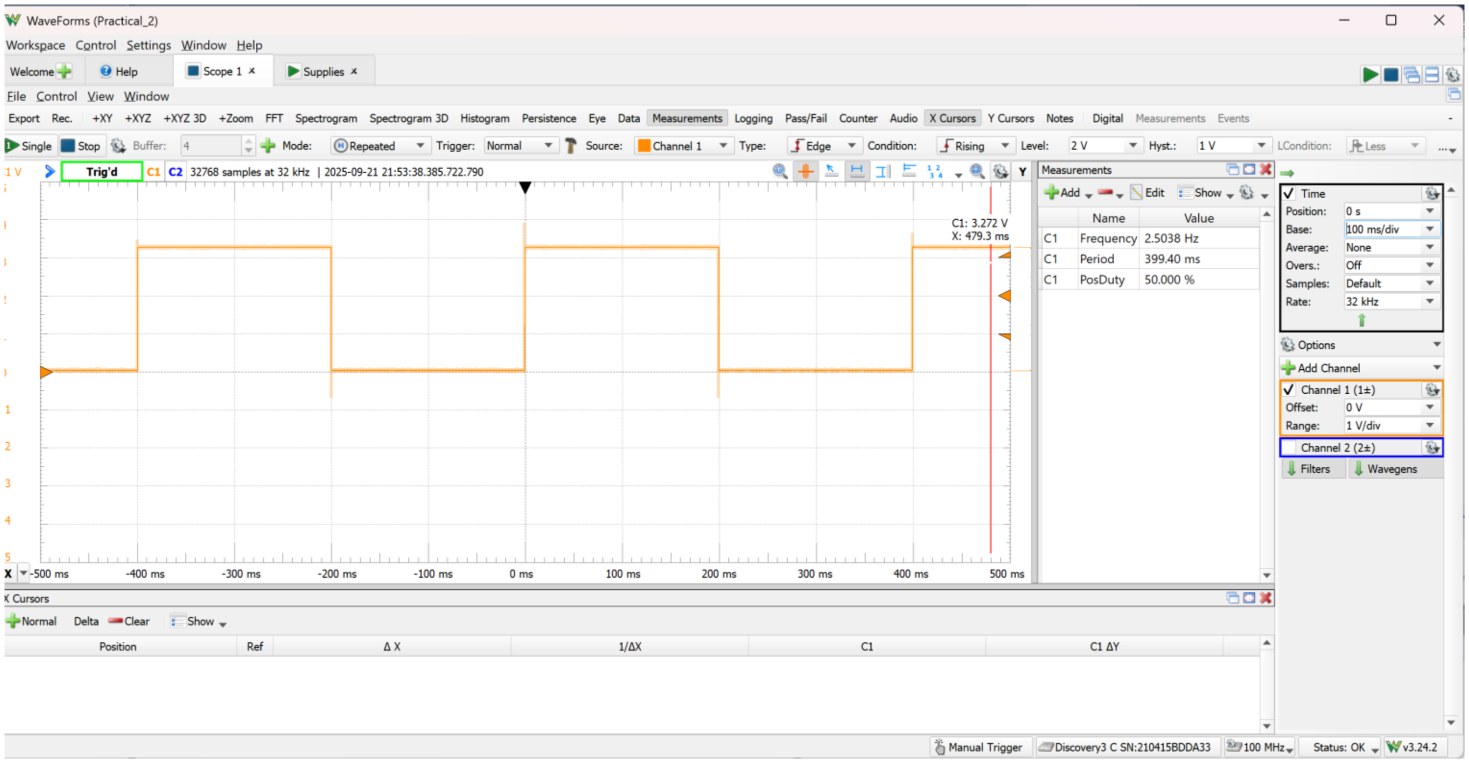
Channel 1 (1±) — Offset: 0 V; Range: 1 V/div

Channel 2 (2±)

X Cursors

	Position	Ref	Δ X	1/ΔX	C1	C1 ΔY
1	-6.457978723 ms	none			0.061564 V	
2	3.517021277 ms	1	9.975 ms	100.25 Hz	3.21356 V	3.152 V

Figure 8: Scope capture of tactile switch S2 (SLB, Store Frame) release, showing mechanical bouncing. Debouncing with Timer/Counter TCC0 ensures only one frame is stored per press.


WaveForms (Practical_2)
Workspace Control Settings Window Help
Welcome Help Scope 1 Supplies
File Control View Window
Export Rec. +XY +XYZ +XYZ 3D +Zoom FFT Spectrogram Spectrogram 3D Histogram Persistence Eye Data Measurements Logging Pass/Fail Counter Audio X Cursors Y Cursors Notes Digital Measurements Events
Single Stop
Trig'd
C1: 3.272 V
X: 479.3 ms
-500 ms -400 ms -300 ms -200 ms -100 ms 0 ms 100 ms 200 ms 300 ms 400 ms 500 ms
Manual Trigger Discovery3 C SN:210415BDDA33 100 MHz Status: OK v3.24.2

Buffer: 4 | Mode: Repeated | Trigger: Normal | Source: Channel 1 | Type: Edge | Condition: Rising | Level: 2 V | Hyst.: 1 V | LCondition: Less

C1 C2 32768 samples at 32 kHz | 2025-09-21 21:53:38.385.722.790

Measurements

	Name	Value
C1	Frequency	2.5038 Hz
C1	Period	399.40 ms
C1	PosDuty	50.000 %

Time — Position: 0 s; Base: 100 ms/div; Average: None; Overs.: Off; Samples: Default; Rate: 32 kHz

Channel 1 (1±) — Offset: 0 V; Range: 1 V/div

Channel 2 (2±)

X Cursors

Position	Ref	Δ X	1/ΔX	C1	C1 ΔY

Figure 9: Scope capture of LED animation playback at 5 Hz (2.5 since the toggle reduces frequency by ½) using Timer/Counter TCC1. Measured period = 399.4 ms, expected = 400.0 ms. Error = 0.15%.

Measurement Table:

Section	**Expected Period**	**Measured Period**	**Error (%)**	**Notes**
2 – Software Delay (Delay_10MS)	20.0 ms	19.94 ms	0.30%	Single 10 ms delay subroutine
2 – Software Delay (Delay_X_10MS, X=4)	80.0 ms	79.775 ms	0.28%	Toggle at ~12.5 Hz
3.1 – Timer/Counter (Prescaler 256, PER=439)	55.00 ms	55.035 ms	0.06%	TCC0 half-period
3.1a – Timer/Counter (Prescaler 8, PER=14,071)	55.00 ms	55.00 ms	0.00%	Finer PER resolution vs. prescaler 256
3.2 – Watch (Prescaler 64, PER=31,951)	2.000 s (toggle = 2s)	1.9979 s	0.105%	PORTC pin 0 toggles once every 2s (1s per overflow)
4 – Switch Bounce (S2 press)	N/A	N/A	N/A	Shows mechanical bouncing on press
4 – Switch Bounce (S2 release)	N/A	N/A	N/A	Shows mechanical bouncing on release
4 – Animation Playback (Prescaler 64, PER=6390)	400.0 ms	399.4 ms	0.15%	Playback at ~5 Hz (toggle halves freq → 2.5 Hz)