# REQUIREMENTS NOT MET

N/A

# PROBLEMS ENCOUNTERED

One problem I encountered was that the program would unexpectedly jump into the overflow interrupt vector immediately after returning from INIT_TCC0 in lab3_2b.asm. After troubleshooting, I realized I had forgotten to include a pop instruction to balance the earlier push, which caused the stack pointer to misalign and return to the wrong address. Another issue arose in the debounce implementation, where the counter sometimes failed to increment; this was due to using the wrong branch instruction (SBRS instead of SBRC) when checking the switch input. Correcting both mistakes resolved the issues, and all aspects of the required pre-lab functions now work as intended.

# FUTURE WORK/APPLICATIONS

With more time and resources, this lab could be extended to use interrupt-driven debouncing for multiple switches simultaneously, allowing for more complex user input. The timer configuration could also be adapted to measure precise press durations, enabling different actions for short versus long button presses. Additionally, integrating external peripherals such as displays or communication modules would demonstrate how interrupt-based control scales to larger embedded applications.

# PRE-LAB EXERCISES

i. Assuming that no interrupt has been previously configured, devise and describe a generalized series of steps for configuring any interrupt within the ATxmega128A1U, i.e., not just an interrupt within the TC system.

a. Assuming that no interrupt has been previously configured, a generalized series of steps for configuring any interrupt within the ATxmega128A1U would go as follows: First you need to identify the interrupt source and choose the peripheral or pin that can generate an interrupt such as a Timer/Counter overflow. Then you need to configure the peripheral for its task, like setting the direction and input sense of a GPIO pin. Next, you need to enable the specific interrupt inside the peripheral and set its priority level (TCC0_INTCTRLA, PORTx_INTCTRL). After that, you need to enable the corresponding interrupt level in the PMIC (PMIC_LOLVLEN_bm → PMIC_CTRL) and enable global interrupts next by setting the I bit in the status register (sei). Finally you need to place a jump instruction at the interrupt vector pointing to the the relevant interrupt service routine. In the ISR is where you will write what you want to happen when an interrupt occurs (make sure to save the SREG register, clear the interrupt flag, and return with reti).

ii. Explain what happens in hardware (in other words, without the programmer's intervention) when the processor detects and then services (and returns from) an interrupt. Be as specific as possible, referencing certain registers when appropriate. You can assume that the reti instruction does not count as programmer intervention. You may provide a flowchart as a response, if desired.

a. The following explains what happens in hardware when the processor detects and then services an interrupt: First an interrupt request occurs where a peripheral sets its interrupt flag, such as the TCC0_INTFLAGS or PORTx_INTFLAGS. If the interrupt is enabled in the peripherals interrupt control register and its priority is enabled in the PMIC_CTRL register, then the request is passed to the CPU. The processor then check if the I bit is set in the status register (CPU_SREG); if the I bit is set the request is acknowledged by the CPU, and the CPU completes the current instruction before servicing the interrupt. The program counter is automatically loaded with the address of the interrupt vector associated with the request (which should have a jump instruction to the ISR), and program execution continues at that ISR location. The ISR executes user-defined instructions, the first of which should be to preserve the status register and clear the interrupt flag. When the ISR ends with reti, the program counter is restored to the instruction following the one that was interrupted. The PMIC also restores its previous state so the system continues with the correct interrupt priority level.

# PSEUDOCODE/FLOWCHARTS

## SECTION 1

**Lab3_1.asm:**

```
;define constants
;assembler directives

.cseg
.org 0
Rjmp main

.org TCC0 OVF interrupt vector
Rjmp TCC0_OVF_ISR

.org 0x100
Main:
;init stack pointer

Rcall INIT_IO
Rcall INIT_TCC0
Rcall INTR_INIT

;infinite loop waiting for interrupts
Rjmp MAIN

;subroutines

;NAME: INIT_IO
;PURPOSE: Initiliaze relevant I/O modules:
```

```
        -
;Input(s):
;Output(s):
;Registers Affected:
INIT_IO:
;push registers
;run subroutine

;initialize pin Pc0 direction to out
;initialize pin Pc0 out reg to be off on the LED (high)

;pop registers

ret




;NAME: INIT_TCC0
;PURPOSE: Initiliaze TCC0:
        -
;Input(s):
;Output(s):
;Registers Affected:



INIT_TC:
;push registers

;run subroutine
;initialize timer count to 0
;initialize ctrla to off
```

```
;initialize period to correspond to 37ms (PRE = 64 / PER = 1157)

;initialize prescaler to 64 and turn on timer

;pop registers
ret



;NAME: INTR_INIT
;PURPOSE: Initialize interrupt signals signals for relevant application:
        -
;Input(s):
;Output(s):
;Registers Affected:



INTR_INIT:
;push registers

;run subroutine

;initialize relevant TC interrupt signals (TCC0_INTCTRLA) (low level)

;enable low-level interrupts globally (PMIC_CTRL)

;enable the global interrupt bit (sei)

;pop registers

ret
```

```
;interrupt service routines:



;Name: TCC0_OVF_ISR
;Purpose:
;Input(s)
;Output(s)
;Registers Affected :



TCC0_OVF_ISR:
;preserve status register

;clear the flag

;toggle pin0 of port c

;recover the status register

Reti
```

**Lab3_2a.asm:**

```
;define constants
;assembler directives

.cseg
.org 0
Rjmp main

.org PORTx_INT0 interrupt vector
Rjmp PORTx_INT0_ISR

.org 0x100
Main:
;init stack pointer

Rcall INIT_IO
Rcall INIT_TCC0
Rcall INTR_INIT

LOOP:
;toggle BLUE_PWM as quick as possible
Rjmp LOOP

;subroutines

;NAME: INIT_IO
;PURPOSE: Initiliaze relevant I/O modules:
        -
;Input(s):
;Output(s):
;Registers Affected:
```

```
INIT_IO:
;push registers
;run subroutine

;initialize PORT C direction to out
;initialize PORT C out reg to be off on the LED (high)

;Initialize s1 on the slb to be an input

;configure Blue_PWM as an output (start off)

;pop registers

ret



;NAME: INTR_INIT
;PURPOSE: Initialize interrupt signals signals for relevant application:
        -
;Input(s):
;Output(s):
;Registers Affected:



INTR_INIT:
;push registers

;run subroutine

;configure port interrupt for S1
;set the input/sense to a falling edge (switch pressed = low) (PINnCTRL)
```

```
;Enable s1 as an interrupt source in PORTx_INT0MASK
;Enable INT0 as low level in PORTx_INTCTRL



;enable low-level interrupts globally (PMIC_CTRL)

;enable the global interrupt bit (sei)

;pop registers

ret



;interrupt service routines:



;Name: PORTx_INT0_ISR
;Purpose:
;Input(s)
;Output(s)
;Registers Affected :



PORTx_INT0_ISR:
;preserve status register

;clear the flag

;increment global register
;output counter value to LEDS
```

```
;recover the status register

Reti
```

**Lab3_2b.asm:**

```
;define constants
;assembler directives

.cseg
.org 0
Rjmp main

.org TCC0_OVF_vect
Rjmp TCC0_OVF_ISR

.org PORTF_INT0_vect
Rjmp PORTx_INT0_ISR

.org 0x100
Main:
;init stack pointer

Rcall INIT_IO
Rcall INIT_TCC0
Rcall INTR_INIT
```

```
LOOP:
;toggle BLUE_PWM as quick as possible
Rjmp LOOP

;subroutines

;NAME: INIT_IO
;PURPOSE: Initiliaze relevant I/O modules:
;Input(s):
;Output(s):
;Registers Affected:

INIT_IO:
;push registers
;run subroutine

;initialize PORT C direction to out
;initialize PORT C out reg to be off on the LED (high)

;Initialize s1 on the slb to be an input

;configure Blue_PWM as an output (start off)

;pop registers

Ret

INIT_TCC0:
;push registers
```

```
;run subroutine
;initialize timer count to 0
;initialize ctrla to off
;initialize period to correspond to ~14ms (PRE = 64 / PER = 450)

;enable TCC0_OVFIF interrupt at low level
;clear overflow flag

;pop registers
ret



;NAME: INTR_INIT
;PURPOSE: Initialize interrupt signals signals for relevant application:

;Input(s):
;Output(s):
;Registers Affected:



INTR_INIT:
;push registers

;run subroutine

;configure port interrupt for S1
;Enable s1 as an interrupt source in PORTx_INT0MASK
;Enable INT0 as low level in PORTx_INTCTRL
;set the input/sense to a falling edge (switch pressed = low) (PINnCTRL)
```

```
;enable low-level interrupts globally (PMIC_CTRL)

;enable the global interrupt bit (sei)

;pop registers

ret



;interrupt service routines:



;Name: PORTx_INT0_ISR
;Purpose:
;Input(s)
;Output(s)
;Registers Affected :



PORTx_INT0_ISR:
;preserve status register

; disable PF2 as INT0 source during debounce (clear PF2 bit in PORTF_INT0MASK)
;clear the flag

;reset and start TCC0
;cnt = 0
;clear OVF flag
;start timer
```

```
;recover the status register

Reti

;Name: TCC0_OVF_ISR
;Purpose:
;Input(s)
;Output(s)
;Registers Affected :


TCC0_OVF_ISR:
;preserve status register

;clear the flag

;stop timer since debounce period is complete
;clear OVF flag
;cnt = 0

;read s1 pin
;if PF2 still pressed (logic 0)
        ;inc counter (r17)
        ;com r17 display on leds and com again
;else
        ;do nothing
;re-enable switch interrupt
;clr INT0IF flag
```

;set pf2 bit in PORTF_INT0MASK

;recover the status register

Reti

# PROGRAM CODE

## SECTION 2

## Lab3_1.asm

```
/*
 * lab3_1.asm
 *
 *  Created: 9/26/2025 9:35:08 PM
 *   Author: arist
 */
 ; Lab 3, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Configure TCC0 to overflow every 37 ms and toggle a Port C LED
;              inside the overflow ISR. Used to verify interrupt handling
;              and timer configuration.

;define constants
;assembler directives
.include "ATxmega128A1Udef.inc"

.cseg
.org 0
rjmp MAIN

.org TCC0_OVF_vect
rjmp TCC0_OVF_ISR

.org 0x100
Main:
;init stack pointer
ldi r16, low(0x3FFF)
sts CPU_SPL, r16
ldi r16, high(0x3FFF)
sts CPU_SPH, r16

Rcall INIT_IO
Rcall INIT_TCC0
Rcall INTR_INIT

;infinite loop waiting for interrupts
LOOP:
Rjmp LOOP

;subroutines

/*****************************************************************************
* Name:     INIT_IO
* Purpose:  Initialize Port C pin 0 as output and turn the LED off.
* Inputs:   None
* Outputs:  None
* Affected: PORTC_DIRSET, PORTC_OUTSET
*****************************************************************************/


INIT_IO:
```

```
;push registers
push r16

;run subroutine

;initialize pin Pc0 direction to out
;initialize pin Pc0 out reg to be off on the LED (high)
ldi r16, (1<<0)
sts PORTC_OUTSET, r16
sts PORTC_DIRSET, r16


;pop registers
pop r16

ret



/*****************************************************************************
* Name:     INIT_TCC0
* Purpose:  Configure Timer/Counter C0 to overflow every ~37 ms (PER = 1182, clk/64).
* Inputs:   None
* Outputs:  None
* Affected: TCC0_CNT, TCC0_PER, TCC0_CTRLA
*****************************************************************************/



INIT_TCC0:
;push registers
push r16

;run subroutine

;initialize timer count to 0
ldi r16, 0
sts TCC0_CNT, r16
sts TCC0_CNT + 1, r16
;initialize ctrla to off
ldi r16, TC_CLKSEL_OFF_gc
sts TCC0_CTRLA, r16

;initialize period to correspond to 37ms (PRE = 64 / PER = 1157)
ldi r16, low(1182)
sts TCC0_PER, r16
ldi r16, high(1182)
sts TCC0_PER + 1, r16


;initialize prescaler to 64 and turn on timer
ldi r16, TC_CLKSEL_DIV64_gc
sts TCC0_CTRLA, r16

;pop registers
pop r16

ret


/*****************************************************************************
```

```
* Name:     INTR_INIT
* Purpose:  Enable low-level interrupts and configure PMIC for TCC0 overflow interrupt.
* Inputs:   None
* Outputs:  None
* Affected: PMIC_CTRL, TCC0_INTCTRLA
*****************************************************************************/


INTR_INIT:
;push registers
push r16


;run subroutine

;initialize relevant TC interrupt signals (TCC0_INTCTRLA) (low level)
ldi r16, TC_OVFINTLVL_LO_gc
sts TCC0_INTCTRLA, r16

;enable low-level interrupts globally (PMIC_CTRL)
ldi r16, PMIC_LOLVLEN_bm
sts PMIC_CTRL, r16

;enable the global interrupt bit (sei)
sei

;pop registers
pop r16

ret


;interrupt service routines:


/*****************************************************************************
* Name:     TCC0_OVF_ISR
* Purpose:  ISR for Timer/Counter C0 overflow. Clears OVF flag and toggles PC0 LED.
* Inputs:   None
* Outputs:  None
* Affected: PORTC_OUTTGL, TCC0_INTFLAGS
*****************************************************************************/



TCC0_OVF_ISR:
;preserve status register
push r16
lds r16, CPU_SREG
push r16


;clear the TCC0 OVFIF int flag
ldi r16, TC0_OVFIF_bm
sts TCC0_INTFLAGS, r16


;toggle pin0 of port c
ldi r16, (1<<0)
sts PORTC_OUTTGL, r16
```

```
;recover the status register
pop r16
sts CPU_SREG, r16
pop r16

reti
```

# Lab3_2a.asm

```
/*
 * lab3_2a.asm
 *
 *  Created: 9/28/2025 4:51:39 PM
 *   Author: arist
 */
;Lab3_2a.asm:
;define constants
;assembler directives

; Lab 3, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Configure PF2 (S1 switch) to trigger an interrupt on a falling edge.
;              In the ISR, increment a counter and display the value in binary on
;              Port C LEDs. Meanwhile, the main loop toggles the Blue PWM LED
;              as quickly as possible to demonstrate asynchronous operation.


.cseg
.org 0
Rjmp main

.org PORTF_INT0_vect ;interrupt vector
Rjmp PORTF_INT0_ISR

.org 0x100
Main:
;init stack pointer
ldi r16, low(0x3FFF)
sts CPU_SPL, r16
ldi r16, high(0x3FFF)
sts CPU_SPH, r16

;global count register
clr r17

Rcall INIT_IO
Rcall INTR_INIT

LOOP:
;toggle BLUE_PWM as quick as possible
ldi r16, (1<<6)
sts PORTD_OUTTGL, r16
Rjmp LOOP

;subroutines

/*****************************************************************************
```

```
* Name:     INIT_IO
* Purpose:  Configure Port C as LED outputs, PF2 (S1) as input, and PD4-PD6 as RGB outputs.
* Inputs:   None
* Outputs:  None
* Affected: PORTC_DIRSET/OUTSET, PORTF_DIRCLR, PORTD_DIRSET/OUTSET
*****************************************************************************/


INIT_IO:
;push registers
push r16
;run subroutine

;initialize PORT C direction to out
;initialize PORT C out reg to be off on the LED (high)
ldi r16, 0xFF
sts PORTC_OUTSET, r16
sts PORTC_DIRSET, r16


;Initialize s1 on the slb to be an input
ldi    r16, (1<<2)
sts PORTF_DIRCLR, r16

;configure three color LEDS as outputs (start off)
ldi r16, (0b01110000)
sts PORTD_OUTSET, r16
sts PORTD_DIRSET, r16


;pop registers
pop r16

ret


/*****************************************************************************
* Name:     INTR_INIT
* Purpose:  Configure Port F INT0 for falling-edge on PF2 (S1). Enable global interrupts.
* Inputs:   None
* Outputs:  None
* Affected: PORTF_INT0MASK, PORTF_INTCTRL, PORTF_PIN2CTRL, PMIC_CTRL
*****************************************************************************/



INTR_INIT:
;push registers
push r16

;run subroutine

;configure port interrupt for S1



;Enable s1 as an interrupt source in PORTx_INT0MASK
ldi r16, (1<<2)
sts PORTF_INT0MASK, r16

;Enable INT0 as low level in PORTx_INTCTRL
```

```
ldi r16, PORT_INT0LVL_LO_gc
sts PORTF_INTCTRL, r16

;set the input/sense to a falling edge (switch pressed = low) (PINnCTRL)
ldi r16, PORT_ISC_FALLING_gc ; default to PORT_OPC_TOTEM_gc = 000
sts PORTF_PIN2CTRL, r16

;enable low-level interrupts globally (PMIC_CTRL)
ldi r16, 0x01 ; PMIC_LOLVLEN_bm
sts PMIC_CTRL, r16

;enable the global interrupt bit (sei)
sei

;pop registers
pop r16

ret


;interrupt service routines:


/*****************************************************************************
* Name:     PORTF_INT0_ISR
* Purpose:  ISR for S1 switch press. Increments global counter and displays it on LEDs.
* Inputs:   None
* Outputs:  Counter value displayed on Port C LEDs (active-low).
* Affected: r17 (counter), PORTC_OUT, PORTF_INTFLAGS
*****************************************************************************/



PORTF_INT0_ISR:
;preserve status register
push r16
lds r16, CPU_SREG
push r16

;clear the flag
ldi r16, (1<<0) ; PORT_INT0IF_bm
sts PORTF_INTFLAGS, r16

;increment global register
inc r17
com r17
;output counter value to LEDS
sts PORTC_OUT, r17

com r17

;recover the status register
pop r16
sts CPU_SREG, r16
pop r16

Reti
```

# Lab3_2b.asm

```
/*
 * lab3_2b.asm
 *
 *  Created: 9/28/2025 8:45:34 PM
 *   Author: arist
 */
 ;define constants
;assembler directives

; Lab 3, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Configure PF2 (S1 switch) with interrupt-based debounce using TCC0.
;              On the first falling edge, disable PF2 interrupts and start TCC0.
;              When TCC0 overflows (~14 ms), check if PF2 is still pressed. If so,
;              increment the counter and update the Port C LEDs. Finally,
;              re-enable PF2 interrupts. The main loop continues toggling the Blue
;              PWM LED asynchronously.

.include "ATxmega128A1Udef.inc"

.cseg
.org 0
Rjmp main

.org TCC0_OVF_vect
Rjmp TCC0_OVF_ISR

.org PORTF_INT0_vect
Rjmp PORTF_INT0_ISR

.org 0x100
Main:
;init stack pointer
ldi r16, low(0x3FFF)
sts CPU_SPL, r16
ldi r16, high(0x3FFF)
sts CPU_SPH, r16

Rcall INIT_IO
Rcall INIT_TCC0
Rcall INTR_INIT

clr r17

LOOP:
;toggle BLUE_PWM as quick as possible
ldi r16, (1<<6)
sts PORTD_OUTTGL, r16
rjmp LOOP

;subroutines

/*****************************************************************************
* Name:     INIT_IO
* Purpose:  Configure Port C as LED outputs, PF2 (S1) as input, and PD4–PD6 as RGB outputs.
* Inputs:   None
```

```
* Outputs:  None
* Affected: PORTC_DIRSET/OUTSET, PORTF_DIRCLR, PORTD_DIRSET/OUTSET
*************************************************************************************/

INIT_IO:
;push registers
push r16
;run subroutine

;initialize PORT C direction to out
;initialize PORT C out reg to be off on the LED (high)
ldi r16, 0xFF
sts PORTC_OUTSET, r16
sts PORTC_DIRSET, r16


;Initialize s1 on the slb to be an input
ldi    r16, (1<<2)
sts PORTF_DIRCLR, r16

;configure three color LEDS as outputs (start off)
ldi r16, (0b01110000)
sts PORTD_OUTSET, r16
sts PORTD_DIRSET, r16


;pop registers
pop r16

ret

/*************************************************************************************
* Name:     INIT_TCC0
* Purpose:  Configure Timer/Counter C0 as debounce timer (~14 ms at clk/64).
*           Timer remains stopped until enabled by PF2 interrupt.
* Inputs:   None
* Outputs:  None
* Affected: TCC0_CNT, TCC0_PER, TCC0_CTRLA, TCC0_INTCTRLA, TCC0_INTFLAGS
*************************************************************************************/


INIT_TCC0:
;push registers
push r16

;run subroutine
;initialize timer count to 0
ldi r16, 0
sts TCC0_CNT, r16
sts TCC0_CNT + 1, r16
;initialize ctrla to off
ldi r16, TC_CLKSEL_OFF_gc
sts TCC0_CTRLA, r16

;initialize period to correspond to ~14ms (PRE = 64 / PER = 450)
ldi r16, low(450)
sts TCC0_PER, r16
ldi r16, high(450)
sts TCC0_PER + 1, r16

;enable TCC0_OVFIF interrupt at low level
```

```
ldi r16, TC_OVFINTLVL_LO_gc
sts TCC0_INTCTRLA, r16

;clear overflow flag
ldi r16, TC0_OVFIF_bm
sts TCC0_INTFLAGS, r16

;pop registers
pop r16

ret


/*****************************************************************************
* Name:     INTR_INIT
* Purpose:  Configure PF2 (S1) interrupt on falling edge and enable low-level interrupts.
* Inputs:   None
* Outputs:  None
* Affected: PORTF_INT0MASK, PORTF_INTCTRL, PORTF_PIN2CTRL, PMIC_CTRL
*****************************************************************************/



INTR_INIT:
;push registers
push r16

;run subroutine

;configure port interrupt for S1
;Enable s1 as an interrupt source in PORTx_INT0MASK
ldi r16, (1<<2)
sts PORTF_INT0MASK, r16
;Enable INT0 as low level in PORTx_INTCTRL
ldi r16, PORT_INT0LVL_LO_gc
sts PORTF_INTCTRL, r16
;set the input/sense to a falling edge (switch pressed = low) (PINnCTRL)
ldi r16, PORT_ISC_FALLING_gc
sts PORTF_PIN2CTRL, r16

;enable low-level interrupts globally (PMIC_CTRL)
ldi r16, PMIC_LOLVLEN_bm
sts PMIC_CTRL, r16

;enable the global interrupt bit (sei)
sei

;pop registers
pop r16

ret


;interrupt service routines:


/*****************************************************************************
* Name:     PORTF_INT0_ISR
* Purpose:  ISR on first falling edge of S1. Disables PF2 interrupts, clears flag,
*           resets and starts debounce timer TCC0.
* Inputs:   None
```

```
* Outputs:  None
* Affected: PORTF_INT0MASK, PORTF_INTFLAGS, TCC0_CNT, TCC0_INTFLAGS, TCC0_CTRLA
*****************************************************************************/

PORTF_INT0_ISR:
;preserve status register
push r16
in r16, CPU_SREG
push r16

; disable PF2 as INT0 source during debounce (clear PF2 bit in PORTF_INT0MASK)
;clear the flag
ldi r16, 00
sts PORTF_INT0MASK, r16

ldi r16, PORT_INT0IF_bm
sts PORTF_INTFLAGS, r16

;reset and start TCC0
;cnt = 0
;clear OVF flag
;start timer
clr r16
sts TCC0_CNT, r16
sts TCC0_CNT + 1, r16

ldi r16, TC0_OVFIF_Bm
sts TCC0_INTFLAGS, r16

ldi r16, TC_CLKSEL_DIV64_gc
sts TCC0_CTRLA, r16


;recover the status register
pop r16
out CPU_SREG, r16
pop r16

reti

/*****************************************************************************
* Name:     TCC0_OVF_ISR
* Purpose:  ISR when debounce timer expires. If PF2 is still pressed (logic low),
*           increment counter and update LEDs. Then re-enable PF2 interrupt.
* Inputs:   None
* Outputs:  Counter value displayed on Port C LEDs (active-low).
* Affected: r17 (counter), PORTC_OUT, PORTF_INTFLAGS, PORTF_INT0MASK, TCC0_INTFLAGS
*****************************************************************************/


TCC0_OVF_ISR:
;preserve status register
push r16
in r16, CPU_SREG
push r16


;stop timer since debounce period is complete
ldi r16, TC_CLKSEL_OFF_gc
sts TCC0_CTRLA, r16
;clear OVF flag
```

```
ldi r16, TC0_OVFIF_bm
sts TCC0_INTFLAGS, r16
;cnt = 0
clr r16
sts TCC0_CNT, r16
sts TCC0_CNT + 1, r16



;read s1 pin
lds r16, PORTF_IN
sbrc r16, 2
rjmp NO_PRESS

;if PF2 still pressed (logic 0)
	;inc counter (r17)
	inc r17
	;com r17 display on leds and com again
	com r17
	sts PORTC_OUT, r17
	com r17
;else
NO_PRESS:
	;do nothing

;re-enable switch interrupt
;clr INT0IF flag
ldi r16, PORT_INT0IF_bm
sts PORTF_INTFLAGS, r16
;set pf2 bit in PORTF_INT0MASK
ldi r16, (1<<2)
sts PORTF_INT0MASK, r16

;recover the status register
pop r16
out CPU_SREG, r16
pop r16


reti
```

# 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:

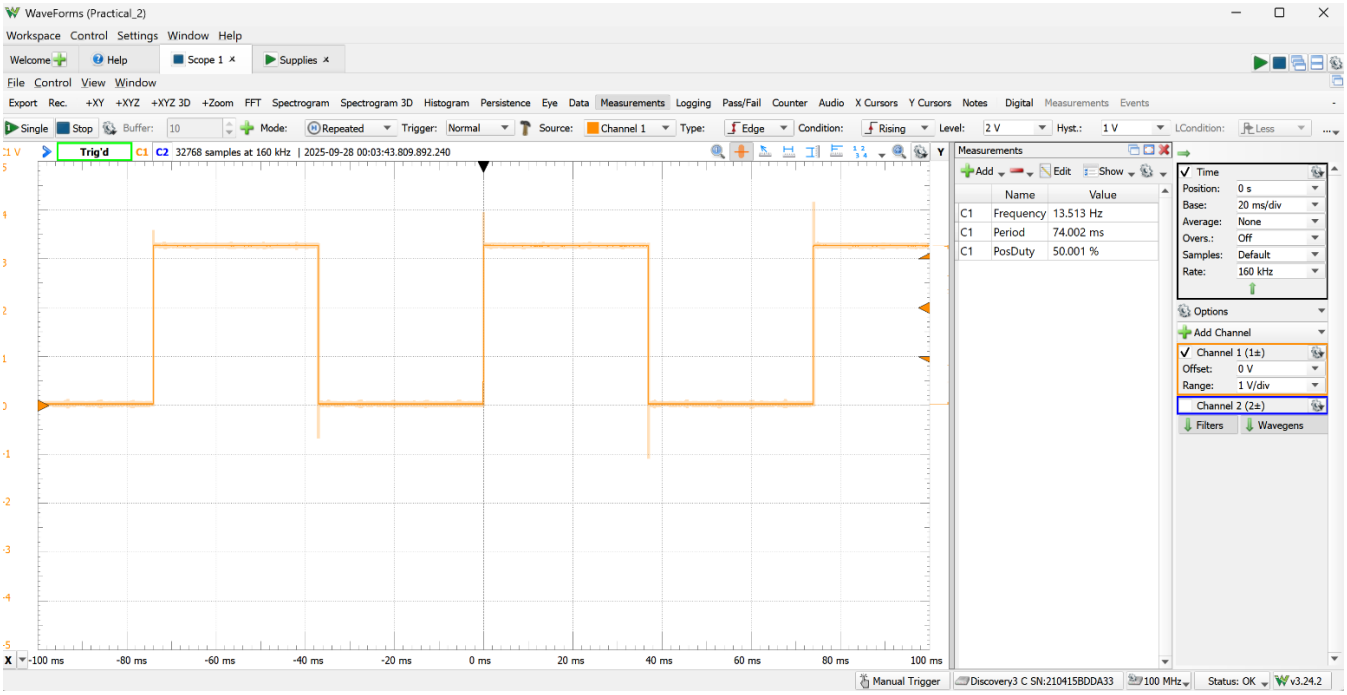


*Figure 1: Screenshot of pin toggling at 37ms (~74ms in Picture since toggle shows double the period).*