Miller, Steven
Class #: 11318
Anthony Stross
June 6, 2023

REQUIREMENTS NOT MET

N/A

PROBLEMS ENCOUNTERED

N/A

FUTURE WORK/APPLICATIONS

The content in this lab can be used in future applications to eliminate switch bouncing in the most efficient way possible, implementing interrupts, and using bit masks.

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.

Configure an interrupt source

Set interrupt condition and interrupt level(low, medium, or high)

Set the PMIC

Enable global interrupts

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.

When an interrupt is detected, a flag is raised in the PMIC. When this occurs, the program memory address for the most recent instruction is added to the stack. The PC is loaded with the program memory address of the interrupt vector. Assuming the programmer loaded the vector with the ISR, the ISR is executed.
When the ISR is finished executing, the PMIC EX flags are cleared, and the PC is loaded with the most recent 2 bytes on the stack (our program instruction when the interrupt was called).

PSEUDOCODE/FLOWCHARTS

SECTION 1

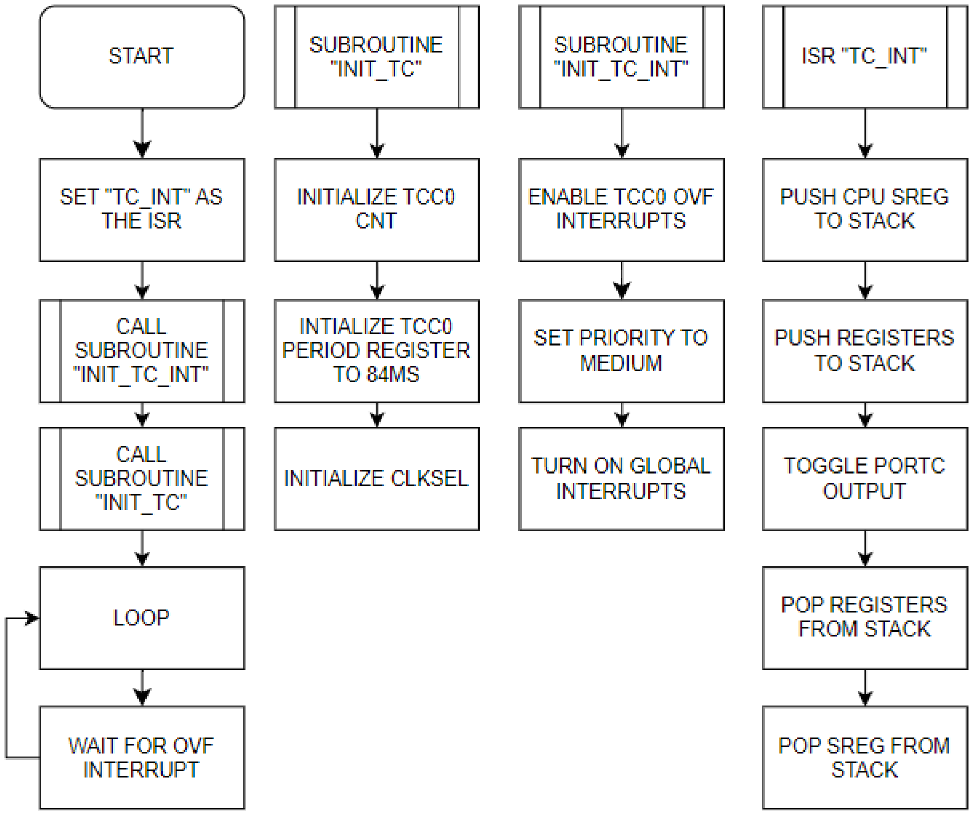


Figure 1: Flowchart for "lab3_1.asm"
The TC interrupt is enabled first, then the TC.
Some useless code is then executed, waiting for an interrupt.

Section 2a

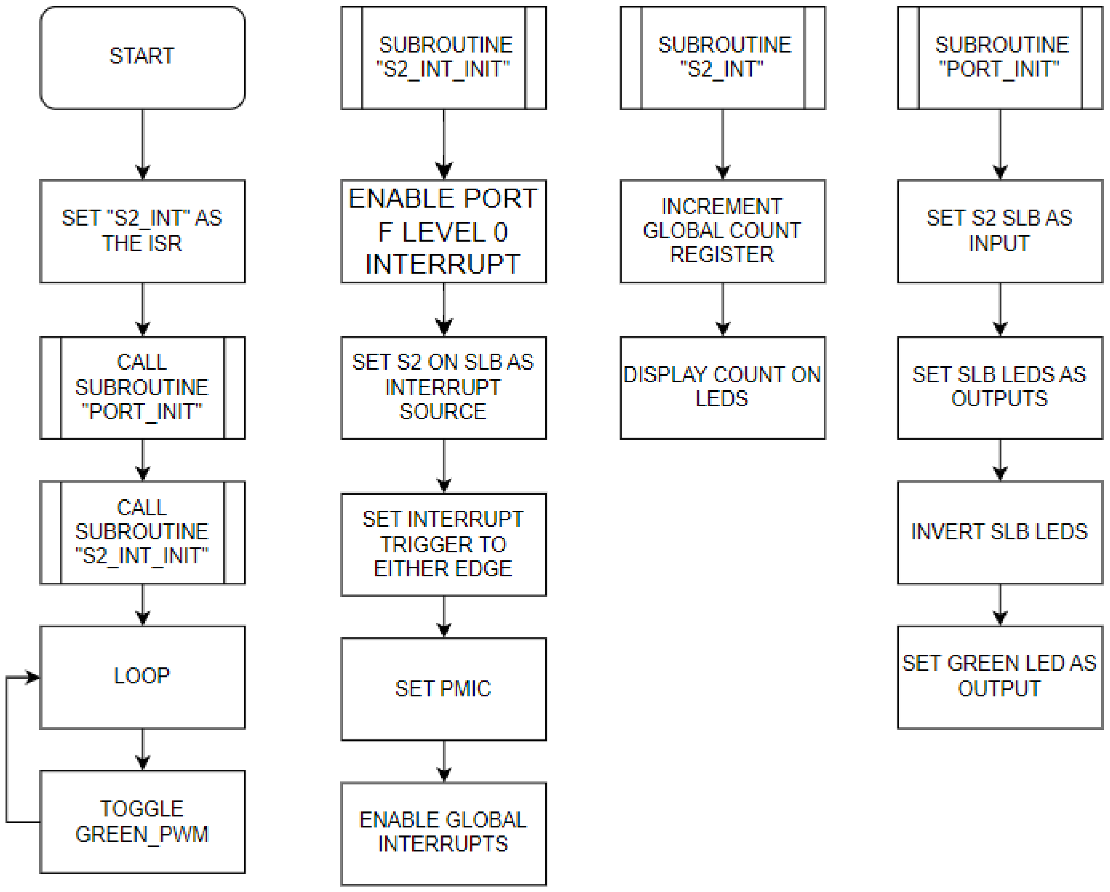


Figure 2: Flowcharts for “lab3_2a.asm”

Section 2b

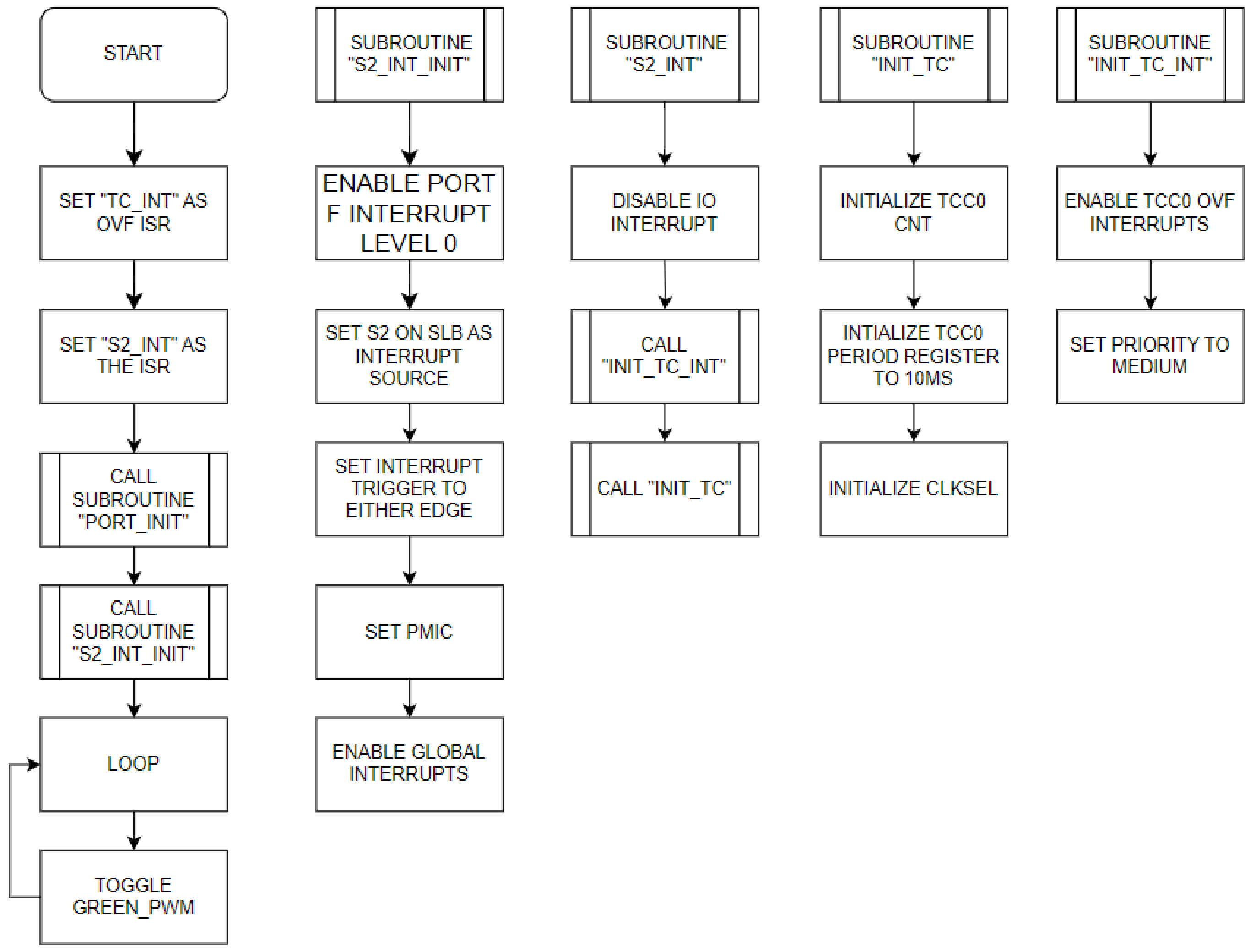


Figure 3: Flowchart for “lab3_2b.asm”

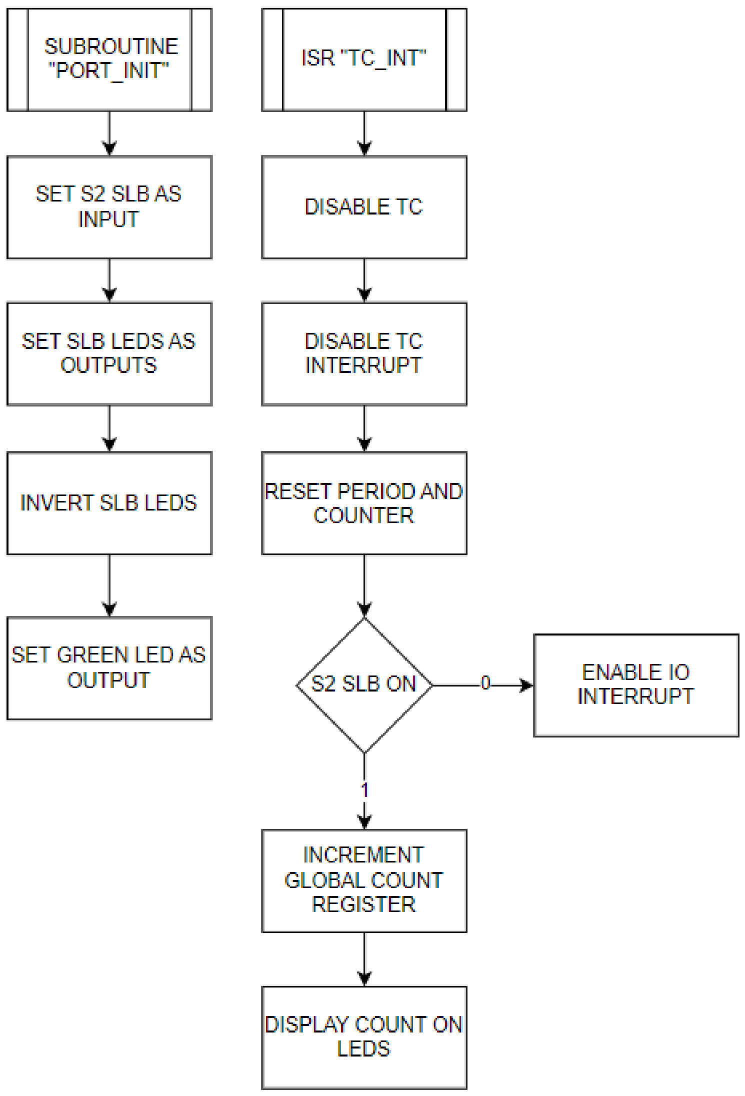


Figure 4: Second flowchart for “lab3_2b.asm”

PROGRAM CODE

SECTION 1

```
;Lab 3, Section 1
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: triggers an overflow interrupt every 84ms
.include "ATxmega128a1udef.inc"
;***************END OF INCLUDES******************************

;*********************************EQUATES********************************
.EQU input = 0b00000000
.EQU output = 0b11111111
.EQU prescalar = 1024
.EQU sysclk = 2000000
.EQU reciprocal = 1/.084 ;idk how to spell reciprocal
.EQU offset =-11 ;correcting for imprecision
.equ stack_init = 0x3FFF
;*******************************END OF EQUATES*******************************

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

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

;***********MAIN PROGRAM*******************************
.CSEG
.org 0x0000
       rjmp main

.CSEG
.org TCC0_OVF_vect
rjmp TC_INT

.CSEG
.org 0x0200
MAIN:
       ;initialize stack pointer
       ldi r16, low(stack_init)
       out CPU_SPL, r16
       ldi r16, high(stack_init)
       out CPU_SPH, r16
rcall init_tc_int
rcall init_tc

;toggle output port
loop:
       ;wait for interrupt
       nop
       rjmp loop

rjmp loop
end:
rjmp end

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

;*****************************************************
; Name: INIT_TC
; Purpose: To initialize the relevant timer/counter modules, as pertains to
;                application.
; Input(s): N/A
; Output: N/A
;****************************************************
INIT_TC:
;initialize count register
ldi r16,0
sts TCC0_CNT, r16
sts TCC0_CNT+1,r16

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

;initialize clksel
ldi r16, TC_CLKSEL_DIV1024_gc
sts TCC0_CTRLA,r16
ldi r16, input
sts TCC0_CTRLB,r16

ret
;****************************************************
; Name: INIT_TC_INT
; Purpose: To initialize the OVF interrupt
; Input(s): N/A
; Output: N/A
;****************************************************
INIT_TC_INT:
;store registers
push r16
;initialize port c for output
ldi r16, output
sts PORTC_DIR, r16
;enable tcc0 ovf interrupts, set priority to medium
ldi r16, 0b00000010
sts TCC0_INTCTRLA,r16
;enable global interrupts
sts PMIC_CTRL,r16
sei
pop r16
ret
;****************************************************
; Name: TC_INT
; Purpose: The TC interrupt service routine
; Input(s): TCC0_INTFLAGS, CPU_SREG
; Output: PORTC_OUTTGL
;****************************************************
TC_INT:
;push cpu sreg to stack
push r20
lds r20,CPU_SREG
push r20
;push registers to stack
push r16
;toggle port c output
ldi r16,0b11111111
sts PORTC_OUTTGL,r16
;pop registers and sreg from stack
pop r16
pop r20
sts CPU_SREG, r20
pop r20
reti
```

Section 2a

```
;Lab 3, Section 2a
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: triggers an interrupt every time s2 on the SLB is pressed

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

;*********************************EQUATES*******************************
.EQU input = 0b00000000
.EQU bit3 = 0b00001000
.EQU output = 0b11111111
.equ stack_init = 0x3FFF
;*******************************END OF EQUATES*******************************

;*********************************DEFS*******************************
.def global_r20 = r20
;*******************************END OF DEFS*******************************

;***********MAIN PROGRAM*******************************
.CSEG
.org 0x0000
       rjmp main

.CSEG
.org PORTF_INT0_vect
       rjmp S2_INT

.CSEG
.ORG 0x0200
main:
       ;set stack pointer
       ldi r16, low(stack_init)
       out CPU_SPL, r16
       ldi r16, high(stack_init)
       out CPU_SPH, r16
       ;initialization subroutines
       rcall port_init
       rcall s2_int_init
       ldi r16, 0b00100000
       loop:
              sts PORTD_OUTTGL,r16
       rjmp loop

end:
       rjmp end
       ;INPUTS: SWITCHES
       ;OUTPUTS: LEDS
;*****************************************************
; Name:PORT_INIT
; Purpose: TO INITIALIZE INPUT AND OUTPUT PORTS
; Input(s): S2_SLB (PORTF_PIN3)
; Output: SLB_LEDS (PORTC)
;*****************************************************
PORT_INIT:
       ;save registers
       push r16
       ;set s2 slb as input
       ldi r16, bit3
       sts PORTF_DIRCLR,r16
```

```
	;set slb_leds as outputs
	ldi r16,output
	sts PORTC_DIRSET,r16
	;invert SLB LEDS by using a mask
	ldi r16,0xff
	sts PORTCFG_MPCMASK,r16
	ldi r16,0b01000000
	sts PORTC_PIN0CTRL,r16
	;set green led as output
	ldi r16,0b00100000
	sts PORTD_DIRSET,r16
	ldi r16, 0b11011111
	sts PORTD_OUT,r16
	;restore from stack
	pop r16
RET
;****************************************************
; Name:S2_INT_INIT
; Purpose: TO INITIALIZE S2 INTERRUPTS
; Input(s): N/A
; Output: N/A
;****************************************************
S2_INT_INIT:
	;save registers
	push r16
	;set s2 on slb as interrupt source
		;sets interrupt level as medium
	ldi r16, 0b00000010
	sts PORTF_INTCTRL,r16
		;sets s2 slb as interrupt source
	ldi r16, bit3
	sts PORTF_INT0MASK,r16
	;set interrupt trigger to rising edge
	ldi r16, 0b00000001
	sts PORTF_PIN3CTRL,r16
	;set PMIC
	ldi r16, 0b00000010
	sts PMIC_CTRL, r16
	;enable global interrupts
	sei
	;restore from stack
	pop r16
RET
;****************************************************
; Name:S2_INT
; Purpose: THE S2 ISR
; Input(s): N/A
; Output: SLB_LEDS (PORTC)
;****************************************************
S2_INT:
	;save registers from stack
	push r16
	lds r16, CPU_SREG
	push r16
	;increment global count register
	inc global_r20
	;display count on leds
	sts PORTC_OUT,global_r20
	;restore registers
	pop r16
	sts CPU_SREG, r16
	pop r16
RETI
```

Section 2b

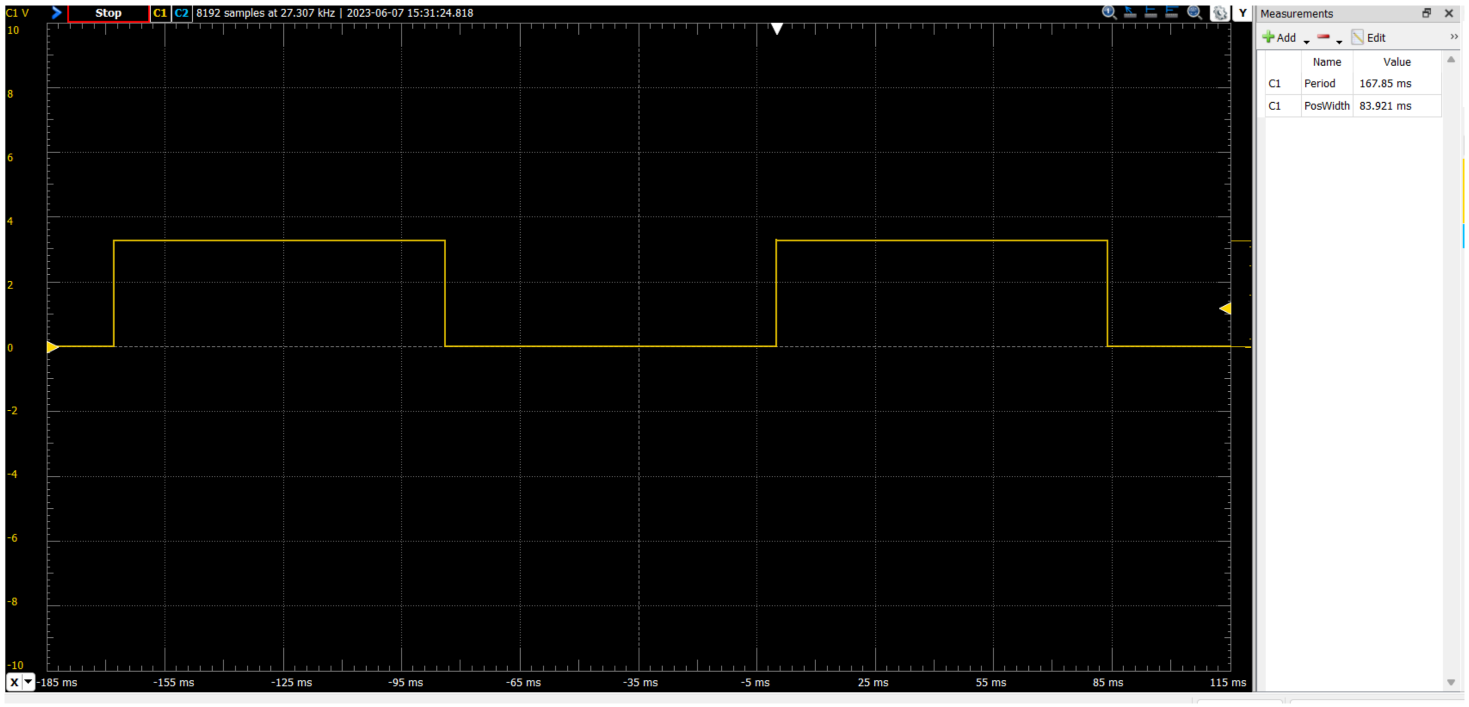
```
;Lab 3, Section 2b
;Name: Steven Miller
;Class #: 11318
;PI Name: Anthony Stross
;Description: displays binary number in register 21 on the leds

;***************INCLUDES************************************
.include "ATxmega128a1udef.inc"
;***************END OF INCLUDES******************************
;*********************************EQUATES********************************
.EQU input = 0b00000000
.EQU bit3 = 0b00001000
.EQU output = 0b11111111
.EQU stack_init = 0x3FFF
.EQU prescalar = 1024
.EQU sysclk = 2000000
.EQU reciprocal = 1/.01 ;idk how to spell reciprocal
.EQU offset =0 ;correcting for imprecision
;*******************************END OF EQUATES*******************************

;*********************************DEFS********************************
.def global_r21 = r21
;*******************************END OF DEFS*******************************

;***********MAIN PROGRAM******************************
.CSEG
.org 0x0000
       rjmp main
;set interrupt vectors
.CSEG
.org TCC0_OVF_vect
       rjmp TC_INT
.ORG PORTF_INT0_vect
       rjmp S2_INT

.CSEG
.ORG 0x0200
main:
       ;set stack pointer
       ldi r16, low(stack_init)
       out CPU_SPL, r16
       ldi r16, high(stack_init)
       out CPU_SPH, r16
       ;initialization subroutines
       rcall port_init
       rcall s2_int_init
       ldi r16, 0b00100000
       loop:
              sts PORTD_OUTTGL,r16
       rjmp loop

end:
       rjmp end

;***********END MAIN PROGRAM******************************
;*****************************************************
; Name:PORT_INIT
; Purpose: TO INITIALIZE INPUT AND OUTPUT PORTS
; Input(s): S2_SLB (PORTF_PIN3)
; Output: SLB_LEDS (PORTC)
; Registers affected: PORTF_DIR,PORTCFG_MPCMASK,PORTC_DIR,PORTC_PIN0CTRL,PORTD_DIR
;*****************************************************
```

```
PORT_INIT:
	;save registers
	push r16
	;set s2 slb as input
	ldi r16, bit3
	sts PORTF_DIRCLR,r16
	;set slb_leds as outputs
	ldi r16,output
	sts PORTC_DIRSET,r16
	;invert SLB LEDS by using a mask
	ldi r16,0xff
	sts PORTCFG_MPCMASK,r16
	ldi r16,0b01000000
	sts PORTC_PIN0CTRL,r16
	;set green led as output
	ldi r16,0b00100000
	sts PORTD_DIRSET,r16
	ldi r16, 0b11011111
	sts PORTD_OUT,r16
	;restore from stack
	pop r16
RET
;*****************************************************
; Name:S2_INT_INIT
; Purpose: TO INITIALIZE S2 INTERRUPTS
; Input(s): N/A
; Output: N/A
; Registers affected: PORTF_INTCTRL, PORTF_INT0MASK,PMIC_CTRL,PORTF_PIN3CTRL
;*****************************************************
S2_INT_INIT:
	;save registers
	push r16
	;set s2 on slb as interrupt source
		;sets interrupt level as medium
	ldi r16, 0b00000010
	sts PORTF_INTCTRL,r16
		;sets s2 slb as interrupt source
	ldi r16, bit3
	sts PORTF_INT0MASK,r16
	;set interrupt trigger to either edge
	ldi r16, 0b00000000
	sts PORTF_PIN3CTRL,r16
	;set PMIC
	ldi r16, 0b00000010
	sts PMIC_CTRL, r16
	;enable global interrupts
	sei
	;restore from stack
	pop r16
RET
;*****************************************************
; Name:S2_INT
; Purpose: THE S2 ISR
; Input(s): N/A
; Output: SLB_LEDS (PORTC)
; Registers affected: PORTF_INTCTRL
;*****************************************************
S2_INT:
	;save registers from stack
	push r20
	lds r20, CPU_SREG
	push r20
	push r16
	;disable io interrupts
```

```
       ldi r16, 0b00000000
       sts PORTF_INTCTRL,r16
       rcall init_tc_int
       rcall init_tc
       pop r16
       pop r20
       sts CPU_SREG,r20
       pop r20
RETI

;****************************************************
; Name: INIT_TC
; Purpose: To initialize the relevant timer/counter modules, as pertains to
;               application.
; Input(s): N/A
; Output: N/A
; Registers affected: TCC0_CNT,TCC0_PER,TCC0_CTRLA,TCC0_CTRLB
;****************************************************
INIT_TC:
push r16
;initialize count register
ldi r16,0
sts TCC0_CNT, r16
sts TCC0_CNT+1,r16

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

;initialize clksel
ldi r16, TC_CLKSEL_DIV1024_gc
sts TCC0_CTRLA,r16
ldi r16, input
sts TCC0_CTRLB,r16
pop r16

ret
;****************************************************
; Name: INIT_TC_INT
; Purpose: To initialize the OVF interrupt
; Input(s): N/A
; Output: N/A
; Registers affected: TCC0_CTRLA
;****************************************************
INIT_TC_INT:
;store registers
push r16
;enable tcc0 ovf interrupts, set priority to medium
ldi r16, 0b00000010
sts TCC0_INTCTRLA,r16
pop r16
ret
;****************************************************
; Name: TC_INT
; Purpose: The TC interrupt service routine
; Input(s): TCC0_INTFLAGS, CPU_SREG
; Output: PORTC_OUTTGL
; Registers affected: TCC0_CNT,TCC0_PER,TCC0_CTRLA,TCC0_CTRLB,TCC0_INTCTRLA
;****************************************************
TC_INT:
;push cpu sreg to stack
push r20
```

```
lds r20,CPU_SREG
push r20
;push registers to stack
push r16
;disable TC
ldi r16, 0b00000000
sts TCC0_CTRLA, r16
;disable tc interrupt
ldi r16, 0b00000000
sts TCC0_INTCTRLA,r16
;reset period and counter
sts TCC0_PER, r16
sts TCC0_PER+1, r16
sts TCC0_CNT,r16
sts TCC0_CNT+1,r16

;get s2 slb switch status
lds r16, PORTF_IN

;increment if on
sbrc r16,3
rjmp enable
;increment global count register
inc global_r21
;display count on leds
sts PORTC_OUT,global_r21

;enable io interrupt
enable:
ldi r16, 0b00000010
sts PORTF_INTCTRL,r16
ldi r16, 0B00000001
sts PORTF_INTFLAGS,r16

;pop registers and sreg from stack
pop r16
pop r20
sts CPU_SREG, r20
pop r20
reti
```

APPENDIX



8192 samples at 27.307 kHz | 2023-06-07 15:31:24.818

	Name	Value
C1	Period	167.85 ms
C1	PosWidth	83.921 ms

Figure 5: Waveform using interrupt enabled TC