Lab 3 Summary

Schuster, Mark

Section 1540

10/14/2017

**B)** Pre-Lab Answers:

1. **1**, as you may use three of the four compare registers, one to handle each LED’s individual duty cycle.
2. This would simple elongate the period of the PWM cycle. Henceforth, left at the same compare value the current duty cycle would be reduced even further.

**C)** Problems encountered:

No major problems or difficulties, just did a lot of reading.

**D)** Future Work/Applications:

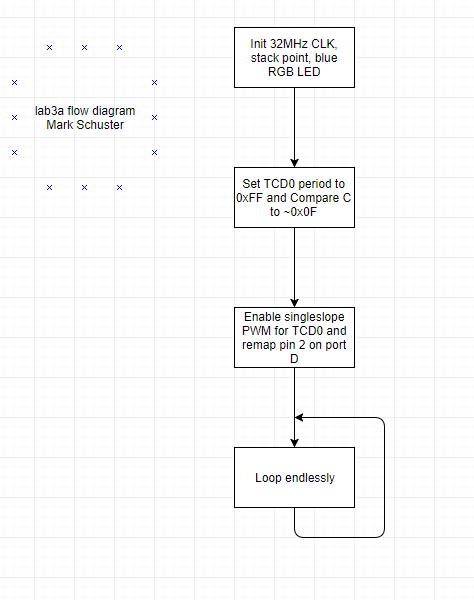
I really enjoyed using playing with PWM and I look forward to using it for serial comms.

**E)** Schematics:

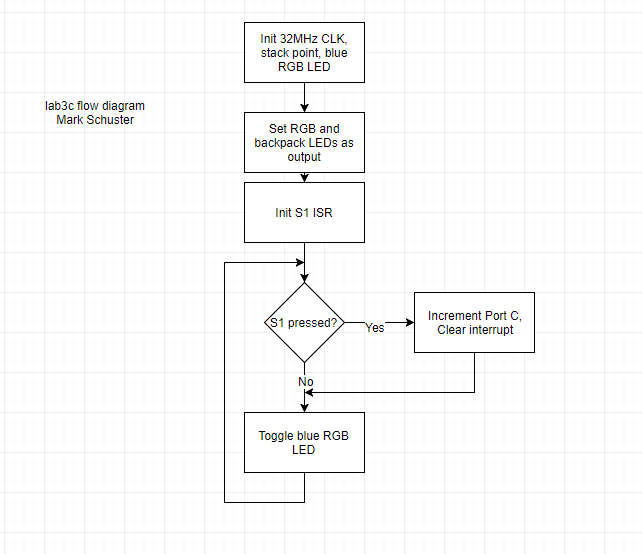
Not applicable for this lab.

**F)** Pseudocode/Flowcharts:

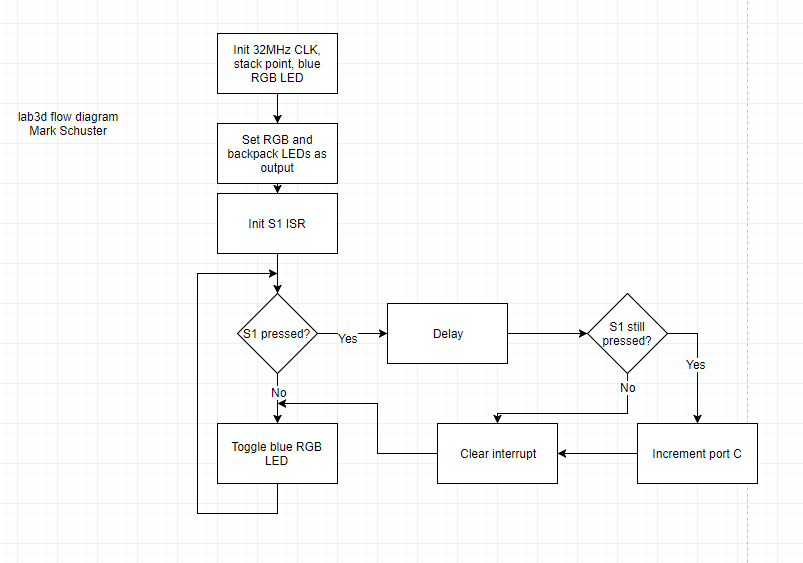
Lab3a flow diagram:



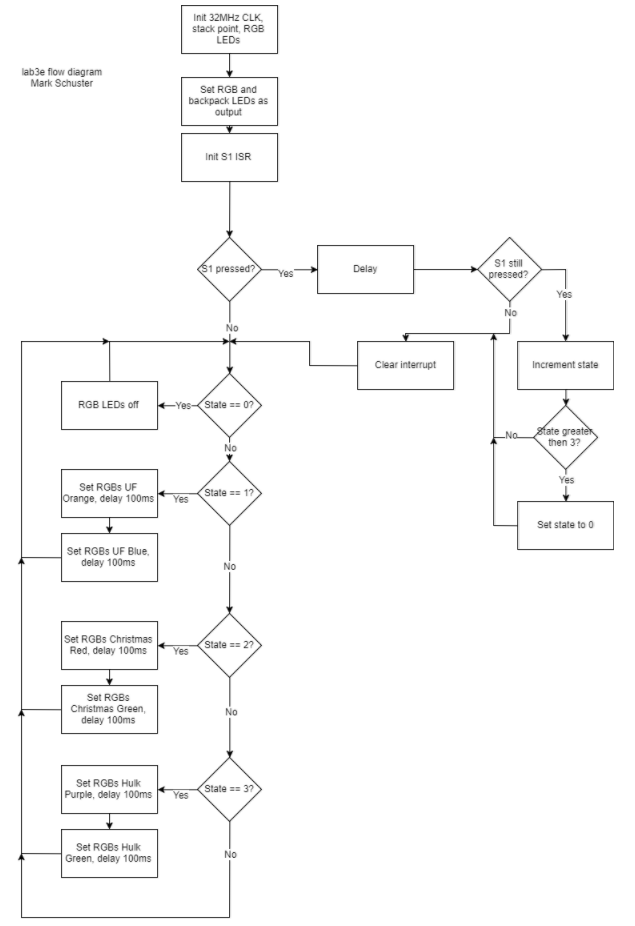
Lab3c flow diagram:



Lab3d flow diagram:



Lab3e flow diagram:



**G)** Program Code:

lab3a.asm:

; Lab 3 part A

; Name: Mark L. Schuster

; Section #: 1540

; TA Name: Christopher Crary

; Description: Uses a timer to enable PWM

; on the blue LED of the RGB LED.

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

; ~~~ Used in USE32MHzCLK ~~~ ;

.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.

.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.

.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.

.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.

.equ CLKSEL = 1 ; Value to select the 32MHz CLK.

.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.

; ~~~ Used in SETTIMER\_PWM ~~~ ;

.equ TCDSEL = 0b0001 ; Value to set the prescaler of the TC to be 1024 time the period of the system CLK.

.equ TCDPER = 0x00FF ; Value of the TC period.

.equ RGBVAL = 0x0F

.equ TCDCMP = (0xFF - RGBVAL)

.equ TCDCMPAINT = 0b00010000 ; Value to init the TC compare A reg.

.equ TCDCMPA\_INTFLAGLOC = 6 ; Location of the compare interrupt flag in the TC's interrupt flags reg.

.equ TCD\_ENPORTD\_SINGLESLOPE = 0b01000011 ; Sets the PWM mode of the TC to single slope.

; ~~~ Used in MAIN ~~~ ;

.equ PDDIRSET = 0b01000000 ; Value to set the dir of port D.

.equ REMAPTOBLUE = 0b00000100 ; Value to remap the output of the TC handling the PWM.

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org 0x200

init:

sei

ldi clkPrescaler, CLKPS ; Load the prescaler value and call USE32MHzCLK.

rcall USE32MHzCLK

rcall SETTIMER\_PWM ; Init the PWM TC.

ldi r16, PDDIRSET ; Set the direction of the blue RGB LED to output.

sts PORTD\_DIRTGL, r16

ldi r16, REMAPTOBLUE ; Remap the TC's compare C from bit 2 to bit 6.

sts PORTD\_REMAP, r16

ldi r16, 0xFF ; Init the RGB LED to off.

sts PORTD\_OUT, r16

loop:

rjmp loop

.org 0x300

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USE32MHzCLK

; Sets the external 32MHz as the active clock for the device

; Inputs: r17 as the desired prescaler for the clock

; Outputs: None

; Affected: r16, r17

USE32MHzCLK:

push r16 ; Preserve the values of r16, r17.

push r17 ;

ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.

sts OSC\_CTRL, r16 ;

checkReady:

lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly

andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.

cpi r16, CLKEN ;

breq clockSel ; If it is move on, if not loop continuously.

rjmp checkReady ;

clockSel:

ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler

sts CPU\_CCP, r16 ; to be written to.

sts CLK\_PSCTRL, clkPrescaler ;

sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control

ldi r16, CLKSEL ; to be set to output the 32 MHz.

sts CLK\_CTRL, r16 ;

pop r17 ; Restore the values of r16 and r17.

pop r16 ;

ret ; return.

.org 0x350

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: SETTIMER\_PWM

; Initialized TCC2 by setting its period and compare A value.

; Inputs: None

; Outputs: None

; Affected: r16,

SETTIMER\_PWM:

push r16 ; Preserve r16.

ldi r16, TCDSEL ; Enable the TC and set its period to be that of the system CLK.

sts TCD0\_CTRLA, r16

ldi r16, TCD\_ENPORTD\_SINGLESLOPE;

sts TCD0\_CTRLB, r16

ldi r16, low(TCDPER) ; Load the period of the TC into the TC's period regs and load the same

sts TCD0\_PER, r16

ldi r16, high(TCDPER)

sts TCD0\_PER+1, r16

ldi r16, low(TCDCMP) ; Load the value to be compared that will

sts TCD0\_CCC, r16 ; determine the duty cycle.

ldi r16, high(TCDCMP)

sts TCD0\_CCC+1, r16

pop r16 ; Restore r16.

ret ; Return.

Lab3c.asm:

; Lab 3 part C

; Name: Mark L. Schuster

; Section #: 1540

; TA Name: Christopher Crary

; Description: Sets up an interrupt triggered by button S1

; that increments the value outputted to the LEDs

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

; ~~~ General ~~~ ;

.equ STACKINIT = 0x3FFF

; ~~~ Used in USE32MHzCLK ~~~ ;

.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.

.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.

.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.

.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.

.equ CLKSEL = 1 ; Value to select the 32MHz CLK.

.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.

; ~~~ Used in SETTIMER\_PWM ~~~ ;

.equ TCDSEL = 0b0001 ; Value to set the prescaler of the TC to be 1024 time the period of the system CLK.

.equ TCDPER = 0x00FF ; Value of the TC period.

.equ RGBVAL = 0x0F

.equ TCDCMP = (TCDPER - RGBVAL)

.equ TCDCMPAINT = 0b00010000 ; Value to init the TC compare A reg.

.equ TCDCMPA\_INTFLAGLOC = 6 ; Location of the compare interrupt flag in the TC's interrupt flags reg.

.equ TCD\_ENPORTD\_SINGLESLOPE = 0b01000011 ; Sets the PWM mode of the TC to single slope.

.equ REMAPTOBLUE = 0b00000100 ; Remaps the TC's 3rd compare reg from the 2nd bit to the 6th bit.

; ~~~ Used in INIT\_BUTTON\_S1\_INT ~~~ ;

.equ INT0\_LOW\_EN = 0b0001 ; Set the interrupt’s priority to low.

.equ BUTTON\_S1\_INT\_TRIGG = 0b0100 ; Set the interrupt to be triggered by S1.

.equ CLEAR\_INT0\_FLAG = 0b01 ; Value to clear the interrupt’s flag.

; ~~~ Used in MAIN ~~~ ;

.equ S1\_DIR\_CLR = 0b0100 ; Value to set the direction of S1 to input.

.equ RGB\_BLUE\_DIR\_SET = 0b01000000 ; Value to set the direction of the blue RGB LED to output.

.equ LEDS\_DIR\_OUT = 0xFF ; Value to set the direction of the backpack LEDs to output.

.equ LED\_COUNT\_INIT = 0xFF ; Initial value of the count.

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org 0x200

init:

ldi clkPrescaler, CLKPS ; Standard inits of the CLK and stack ptr.

rcall USE32MHzCLK

ldi XL, low(STACKINIT)

out CPU\_SPL, XL

ldi XL, high(STACKINIT)

out CPU\_SPH, XL

ldi r16, S1\_DIR\_CLR ; Set S1 as input.

sts PORTF\_DIRCLR, r16

ldi r16, RGB\_BLUE\_DIR\_SET ; Set the blue RGB LED as output.

sts PORTD\_DIRSET, r16

ldi r16, LEDS\_DIR\_OUT ; Set the backpack LEDs as output, and

sts PORTC\_DIRSET, r16 ; init their value to off.

sts PORTC\_OUT, r16

rcall INIT\_BUTTON\_S1\_INT ; Init S1's interrupt.

sei ; Enable interrupts.

ldi r17, LED\_COUNT\_INIT ; Set the LED count to zero. For active low

; LEDs this would be NOT 0x00 or 0xFF.

loop:

sts PORTD\_OUTTGL, r16 ; Toggle backpack LEDs and loop infinitely.

rjmp loop

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INTERUPT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: ISR\_BUTTON\_PRESSED

; Sets up an interrupt to be triggered by the S1 button to

; increment a count and output it to the LEDs.

; Inputs: None

; Outputs: None

; Affected: r16, r17

.org PORTF\_INT0\_vect

rjmp ISR\_BUTTON\_PRESSED

.org 0x400

ISR\_BUTTON\_PRESSED:

ldi r16, CLEAR\_INT0\_FLAG ; Clear the interrupt flag.

sts PORTF\_INTFLAGS, r16

dec r17 ; If so, decrement r17 which is equivalent to

sts PORTC\_OUT, r17 ; incrementing the count displayed by the LEDs

; as they are active low.

reti ; Return.

.org 0x300

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USE32MHzCLK

; Sets the external 32MHz as the active clock for the device

; Inputs: r17 as the desired prescaler for the clock

; Outputs: None

; Affected: r16, r17

USE32MHzCLK:

push r16 ; Preserve the values of r16, r17.

push r17 ;

ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.

sts OSC\_CTRL, r16 ;

checkReady:

lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly

andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.

cpi r16, CLKEN ;

breq clockSel ; If it is move on, if not loop continuously.

rjmp checkReady ;

clockSel:

ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler

sts CPU\_CCP, r16 ; to be written to.

sts CLK\_PSCTRL, clkPrescaler ;

sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control

ldi r16, CLKSEL ; to be set to output the 32 MHz.

sts CLK\_CTRL, r16 ;

pop r17 ; Restore the values of r16 and r17.

pop r16 ;

ret ; return.

.org 0x350

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: SETTIMER\_PWM

; Initialized TCD0 by setting its period and compare A value.

; Inputs: None

; Outputs: None

; Affected: r16,

SETTIMER\_PWM:

push r16 ; Preserve r16.

ldi r16, TCDSEL ; Enable the TC and set its period to be that of the system CLK.

sts TCD0\_CTRLA, r16

ldi r16, TCD\_ENPORTD\_SINGLESLOPE;

sts TCD0\_CTRLB, r16

ldi r16, low(TCDPER) ; Load the period of the TC into the TC's period regs and load the same

sts TCD0\_PER, r16

ldi r16, high(TCDPER)

sts TCD0\_PER+1, r16

ldi r16, low(TCDCMP) ; Load the value to be compared that will

sts TCD0\_CCC, r16 ; determine the duty cycle.

ldi r16, high(TCDCMP)

sts TCD0\_CCC+1, r16

pop r16 ; Restore r16.

ret ; Return.

.org 0x450

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: INIT\_BUTTON\_S1\_INT

; Set up an interrupt for button S1.

; Inputs: None

; Outputs: None

; Affected: r16,

INIT\_BUTTON\_S1\_INT:

push r16 ; Preserve r16.

ldi r16, INT0\_LOW\_EN ; Set the interrupts priority

sts PORTF\_INTCTRL, r16 ; to low.

ldi r16, BUTTON\_S1\_INT\_TRIGG ; Set the bitmask to trigger the

sts PORTF\_INT0MASK, r16 ; interrupt on S1's bit.

ldi r16, PORT\_ISC\_FALLING\_gc ; Set the interrupt to trigger on

sts PORTF\_PIN2CTRL, r16 ; a falling edge as S1 is active low.

ldi r16, PMIC\_LOLVLEN\_bm ; Enable low priority interrupts.

sts PMIC\_CTRL, r16

pop r16 ; Restore r16.

ret ; Return.

lab3d.asm:

; Lab 3 part D

; Name: Mark L. Schuster

; Section #: 1540

; TA Name: Christopher Crary

; Description: Sets up an interrupt triggered by button S1

; that increments the value outputted to the LEDs

; except with debouncing for S1.

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

; ~~~ General ~~~ ;

.equ STACKINIT = 0x3FFF

; ~~~ Used in USE32MHzCLK ~~~ ;

.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.

.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.

.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.

.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.

.equ CLKSEL = 1 ; Value to select the 32MHz CLK.

.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.

; ~~~ Used in SETTIMER\_PWM ~~~ ;

.equ TCDSEL = 0b0001 ; Value to set the prescaler of the TC to be 1024 time the period of the system CLK.

.equ TCDPER = 0x00FF ; Value of the TC period.

.equ RGBVAL = 0x0F

.equ TCDCMP = (TCDPER - RGBVAL)

.equ TCDCMPAINT = 0b00010000 ; Value to init the TC compare A reg.

.equ TCDCMPA\_INTFLAGLOC = 6 ; Location of the compare interrupt flag in the TC's interrupt flags reg.

.equ TCD\_ENPORTD\_SINGLESLOPE = 0b01000011 ; Sets the PWM mode of the TC to single slope.

.equ REMAPTOBLUE = 0b00000100 ; Remaps the TC's 3rd compare reg from the 2nd bit to the 6th bit.

; ~~~ Used in DEBOUNCE\_S1 ~~~ ;

.equ TCC0SEL = 0b0111 ; Set the counter to 1024 times the sys CLK's period.

.equ TCC0PER = 0x0080 ; Delay 0x40 ticks for debouncing.

.equ TCC0DISABLE = 0b0000 ; Value to disable the TC.

; ~~~ Used in INIT\_BUTTON\_S1\_INT ~~~ ;

.equ INT0\_LOW\_EN = 0b0001 ; Set the interrupt’s priority to low.

.equ BUTTON\_S1\_INT\_TRIGG = 0b0100 ; Set the interrupt to be triggered by S1.

.equ CLEAR\_INT0\_FLAG = 0b01 ; Value to clear the interrupt’s flag.

; ~~~ Used in MAIN ~~~ ;

.equ S1\_DIR\_CLR = 0b0100 ; Value to set the direction of S1 to input.

.equ RGB\_BLUE\_DIR\_SET = 0b01000000 ; Value to set the direction of the blue RGB LED to output.

.equ LEDS\_DIR\_OUT = 0xFF ; Value to set the direction of the backpack LEDs to output.

.equ LED\_COUNT\_INIT = 0xFF ; Initial value of the count.

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org 0x200

init:

ldi clkPrescaler, CLKPS ; Standard inits of the CLK and stack ptr.

rcall USE32MHzCLK

ldi XL, low(STACKINIT)

out CPU\_SPL, XL

ldi XL, high(STACKINIT)

out CPU\_SPH, XL

ldi r16, S1\_DIR\_CLR ; Set S1 as input.

sts PORTF\_DIRCLR, r16

ldi r16, RGB\_BLUE\_DIR\_SET ; Set the blue RGB LED as output.

sts PORTD\_DIRSET, r16

ldi r16, LEDS\_DIR\_OUT ; Set the backpack LEDs as output, and

sts PORTC\_DIRSET, r16 ; init their value to off.

sts PORTC\_OUT, r16

rcall INIT\_BUTTON\_S1\_INT ; Init S1's interrupt.

sei ; Enable interrupts.

ldi r17, LED\_COUNT\_INIT ; Set the LED count to zero. For active low

; LEDs this would be NOT 0x00 or 0xFF.

loop:

sts PORTD\_OUTTGL, r16 ; Toggle backpack LEDs and loop infinitely.

rjmp loop

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INTERUPT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: ISR\_BUTTON\_PRESSED

; Sets up an interupt to be triggered by the S1 button to

; increment a count and output it to the LEDs.

; Inputs: None

; Outputs: None

; Affected: r16, r17

.org PORTF\_INT0\_vect

rjmp ISR\_BUTTON\_PRESSED

.org 0x300

ISR\_BUTTON\_PRESSED:

rcall DEBOUNCE\_S1 ; Handle the bouncing of the button by waiting.

push r16 ; Preserve r16.

lds r16, PORTF\_IN ; Check the switch to make sure it's still

sbrc r16, 2 ; being pressed.

rjmp endInt ; If not, exit the ISR.

dec r17 ; If so, decrement r17 which is equivalent to

sts PORTC\_OUT, r17 ; incrementing the count displayed by the LEDs

; as they are active low.

endInt:

ldi r16, CLEAR\_INT0\_FLAG ; Clear the interrupt flag.

sts PORTF\_INTFLAGS, r16

pop r16 ; Restore r16.

reti ; Return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USE32MHzCLK

; Sets the external 32MHz as the active clock for the device

; Inputs: r17 as the desired prescaler for the clock

; Outputs: None

; Affected: r16, r17

USE32MHzCLK:

push r16 ; Preserve the values of r16, r17.

push r17

ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.

sts OSC\_CTRL, r16

checkReady:

lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly

andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.

cpi r16, CLKEN

breq clockSel ; If it is move on, if not loop continuously.

rjmp checkReady

clockSel:

ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler

sts CPU\_CCP, r16 ; to be written to.

sts CLK\_PSCTRL, clkPrescaler

sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control

ldi r16, CLKSEL ; to be set to output the 32 MHz.

sts CLK\_CTRL, r16

pop r17 ; Restore the values of r16 and r17.

pop r16

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: SETTIMER\_PWM

; Initialized TCD0 by setting its period and compare A value.

; Inputs: None

; Outputs: None

; Affected: r16,

SETTIMER\_PWM:

push r16 ; Preserve r16.

ldi r16, TCDSEL ; Enable the TC and set its period to be that of the system CLK.

sts TCD0\_CTRLA, r16

ldi r16, TCD\_ENPORTD\_SINGLESLOPE;

sts TCD0\_CTRLB, r16

ldi r16, low(TCDPER) ; Load the period of the TC into the TC's period regs and load the same

sts TCD0\_PER, r16

ldi r16, high(TCDPER)

sts TCD0\_PER+1, r16

ldi r16, low(TCDCMP) ; Load the value to be compared that will

sts TCD0\_CCC, r16 ; determine the duty cycle.

ldi r16, high(TCDCMP)

sts TCD0\_CCC+1, r16

pop r16 ; Restore r16.

ret ; Return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: INIT\_BUTTON\_S1\_INT

; Set up an interupt for button S1.

; Inputs: None

; Outputs: None

; Affected: r16,

INIT\_BUTTON\_S1\_INT:

push r16 ; Preserve r16.

ldi r16, INT0\_LOW\_EN ; Set the interrupt’s priority

sts PORTF\_INTCTRL, r16 ; to low.

ldi r16, BUTTON\_S1\_INT\_TRIGG ; Set the bitmask to trigger the

sts PORTF\_INT0MASK, r16 ; interrupt on S1's bit.

ldi r16, PORT\_ISC\_FALLING\_gc ; Set the interrupt to trigger on

sts PORTF\_PIN2CTRL, r16 ; a falling edge as S1 is active low.

ldi r16, PMIC\_LOLVLEN\_bm ; Enable low priority interrupts.

sts PMIC\_CTRL, r16

pop r16 ; Restore r16.

ret ; Return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: DEBOUNCE\_S1

; Checks for bouncing on S1.

; Inputs: None

; Outputs: None

; Affected: r16

DEBOUNCE\_S1:

push r16 ; Preserve r16.

ldi r16, TCC0SEL ; Enable the TC and set its period to be 1024 times that of the system CLK.

sts TCC0\_CTRLA, r16

ldi r16, low(TCC0PER) ; Load the period of the TC into the TC's period regs.

sts TCC0\_PER, r16

ldi r16, high(TCC0PER) ;

sts TCC0\_PER+1, r16

checkS1Loop:

lds r16, TCC0\_INTFLAGS ; Wait until the TC's overflow flag is triggered

sbrs r16, 0

rjmp checkS1Loop

ldi r16, 0x01 ; Once broken from the loop, clear the flag, reset the

sts TCC0\_INTFLAGS, r16 ; count, and disable the timer.

ldi r16, 0x00

sts TCC0\_CNT, r16

ldi r16, TCC0DISABLE

sts TCC0\_CTRLA, r16

pop r16 ; Restore r16.

ret ; Return.

lab3e.asm(Not sure what happened with the formatting☹):

**; Lab 3 part E**

**; Name: Mark L. Schuster**

**; Section #: 1540**

**; TA Name: Christopher Crary**

**; Description: Switches pairs of colors outputted**

**; by the RGB LED by pressing the button S1.**

**.nolist ; Included for fun.**

**.include "ATxmega128A1Udef.inc" ;**

**.list ;**

**; ~~~ General ~~~ ;**

**.equ STACKINIT = 0x3FFF**

**; ~~~ Used in USE32MHzCLK ~~~ ;**

**.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.**

**.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.**

**.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.**

**.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.**

**.equ CLKSEL = 1 ; Value to select the 32MHz CLK.**

**.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.**

**; ~~~ Used in INIT\_RGB ~~~ ;**

**.equ RGB\_DIRSET = 0b01110000**

**; ~~~ Used in SETTIMER\_PWM ~~~ ;**

**.equ TCDSEL = 0b0001 ; Value to set the prescaler of the TC to be 1024 time the period of the system CLK.**

**.equ TCDPER = 0x00FF ; Value of the TC period.**

**.equ TCDCMPAINT = 0b00010000 ; Value to init the TC compare A reg.**

**.equ TCDCMPA\_INTFLAGLOC = 6 ; Location of the compare interrupt flag in the TC's interrupt flags reg.**

**.equ TCD\_ENPORTD\_SINGLESLOPE = 0b01110011 ; Sets the PWM mode of the TC to single slope.**

**.equ RGB\_REMAP = 0b00000111 ; Value to remap pins 0, 1, & 2 to 4, 5, & 6 respectively.**

**; ~~~ Used in DELAY\_100ms ~~~ ;**

**.equ TC\_SEL = 0b0111 ; Value to set the prescaler of the TC to be 1024 time the period of the system CLK.**

**.equ TC\_PER = 0x0C35 ; Value of the TC period.**

**.equ TC\_DISABLE = 0b0000**

**; ~~~ Used in DEBOUNCE\_S1 ~~~ ;**

**.equ TCC0SEL = 0b0111 ; Set the counter to 1024 times the sys CLK's period.**

**.equ TCC0PER = 0x0080 ; Delay 0x80 ticks for debouncing.**

**.equ TCC0DISABLE = 0b0000 ; Value to disable the TC.**

**; ~~~ Used in INIT\_BUTTON\_S1\_INT ~~~ ;**

**.equ INT0\_LOW\_EN = 0b0001 ; Set the interrupt’s priority to low.**

**.equ BUTTON\_S1\_INT\_TRIGG = 0b0100 ; Set the interrupt to be triggered by S1.**

**.equ CLEAR\_INT0\_FLAG = 0b01 ; Value to clear the interrupt’s flag.**

**; ~~~ Used in ISR\_BUTTON\_PRESSED ~~~ ;**

**.equ NUM\_STATES = 4 ; Number of states in the state machine.**

**; ~~~ Used in MAIN ~~~ ;**

**.equ S1\_DIR\_CLR = 0b0100 ; Value to set the direction of S1 to input.**

**.equ INIT\_STATE = 0x00 ; Initial state of the state machine.**

**.equ OFF\_STATE = 0x00 ; State for setting the RGB to off.**

**.equ UF\_STATE = 0x01 ; State for setting the RGB to UF colors.**

**.equ CHRISTMAS\_STATE = 0x02 ; State for setting the RGB to Christmas colors.**

**.equ HULK\_STATE = 0x03 ; State for setting the RGB to Hulk colors.**

**.equ COPS\_STATE = 0x04 ; State for setting the RGB to police colors.**

**.equ RGB\_OFF\_VAL = 0x000000 ; RGB Value for off.**

**.equ RGB\_UF\_ORNG = 0xFA4616 ; RGB Value for UF orange.**

**.equ RGB\_UF\_BLUE = 0x0021A5 ; RGB Value for UF blue.**

**.equ RGB\_CHRISTMAS\_RED = 0xC21F1F ; RGB Value for Christmas red.**

**.equ RGB\_CHRISTMAS\_GRN = 0x3C8D0D ; RGB Value for Christmas green.**

**.equ RGB\_HULK\_PRPL = 0x8A2C9A ; RGB Value for Hulk purple.**

**.equ RGB\_HULK\_GRN = 0x49FF07 ; RGB Value for Hulk green.**

**.equ RGB\_COPS\_BLUE = 0x000080 ; RGB Value for police blue.**

**.equ RGB\_COPS\_RED = 0x720027 ; RGB Value for police red.**

**.org 0x0000**

**rjmp init ; Start at 0x0000 and jump to program.**

**.org 0x200**

**init:**

**ldi clkPrescaler, CLKPS ; Standard inits of the CLK, the RGB, stack ptr.**

**rcall USE32MHzCLK**

**rcall INIT\_RGB**

**ldi XL, low(STACKINIT)**

**out CPU\_SPL, XL**

**ldi XL, high(STACKINIT)**

**out CPU\_SPH, XL**

**ldi r16, S1\_DIR\_CLR ; Set the direction of S1's direction to input.**

**sts PORTF\_DIRCLR, r16**

**rcall INIT\_BUTTON\_S1\_INT ; Setup the interrupt for S1 being pressed.**

**sei ; Enable interrupts**

**ldi r19, INIT\_STATE ; Set the initial state.**

**loop:**

**cpi r19, OFF\_STATE ; Case statement to handle state machine.**

**breq RGB\_OFF**

**cpi r19, UF\_STATE**

**breq RGB\_UF**

**cpi r19, CHRISTMAS\_STATE**

**breq RGB\_CHRISTMAS**

**cpi r19, HULK\_STATE**

**breq RGB\_HULK**

**; cpi r19, COPS\_STATE**

**; breq RGB\_COPS**

**rjmp loop ; Catch just in case.**

**RGB\_OFF:**

**rcall SET\_RGB\_OFF ; Turn of RGB.**

**rjmp loop**

**RGB\_UF:**

**ldi r16, ~byte3(RGB\_UF\_ORNG) ; Oscillate between UF colors, waiting 1ms on each.**

**ldi r17, ~byte2(RGB\_UF\_ORNG)**

**ldi r18, ~byte1(RGB\_UF\_ORNG)**

**rcall SET\_RGB**

**rcall DELAY\_100ms**

**ldi r16, ~byte3(RGB\_UF\_BLUE)**

**ldi r17, ~byte2(RGB\_UF\_BLUE)**

**ldi r18, ~byte1(RGB\_UF\_BLUE)**

**rcall SET\_RGB**

**rcall DELAY\_100ms**

**rjmp loop**

**RGB\_CHRISTMAS:**

**ldi r16, ~byte3(RGB\_CHRISTMAS\_RED) ; Oscillate between Christmas colors, waiting 1ms on each.**

**ldi r17, ~byte2(RGB\_CHRISTMAS\_RED)**

**ldi r18, ~byte1(RGB\_CHRISTMAS\_RED)**

**rcall SET\_RGB**

**rcall DELAY\_100ms**

**ldi r16, ~byte3(RGB\_CHRISTMAS\_GRN)**

**ldi r17, ~byte2(RGB\_CHRISTMAS\_GRN)**

**ldi r18, ~byte1(RGB\_CHRISTMAS\_GRN)**

**rcall SET\_RGB**

**rcall DELAY\_100ms**

**rjmp loop**

**RGB\_HULK:**

**ldi r16, ~byte3(RGB\_HULK\_PRPL) ; Oscillate between Hulk colors, waiting 1ms on each.**

**ldi r17, ~byte2(RGB\_HULK\_PRPL)**

**ldi r18, ~byte1(RGB\_HULK\_PRPL)**

**rcall SET\_RGB**

**rcall DELAY\_100ms**

**ldi r16, ~byte3(RGB\_HULK\_GRN)**

**ldi r17, ~byte2(RGB\_HULK\_GRN)**

**ldi r18, ~byte1(RGB\_HULK\_GRN)**

**rcall SET\_RGB**

**rcall DELAY\_100ms**

**rjmp loop**

**; Included for fun!**

**;RGB\_COPS:**

**; ldi r16, ~byte3(RGB\_COPS\_BLUE); Oscillate between cop colors, waiting 1ms on each.**

**; ldi r17, ~byte2(RGB\_COPS\_BLUE)**

**; ldi r18, ~byte1(RGB\_COPS\_BLUE)**

**; rcall SET\_RGB**

**; rcall DELAY\_100ms**

**; ldi r16, ~byte3(RGB\_COPS\_RED)**

**; ldi r17, ~byte2(RGB\_COPS\_RED)**

**; ldi r18, ~byte1(RGB\_COPS\_RED)**

**; rcall SET\_RGB**

**; rcall DELAY\_100ms**

**; rjmp loop**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INTERUPT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; Subroutine Name: ISR\_BUTTON\_PRESSED**

**; Sets up an interrupt to be triggered by the S1 button to**

**; increment a count and output it to the LEDs.**

**; Inputs: None**

**; Outputs: None**

**; Affected: r16, r17**

**.org PORTF\_INT0\_vect**

**rjmp ISR\_BUTTON\_PRESSED**

**.org 0x300**

**ISR\_BUTTON\_PRESSED:**

**rcall DEBOUNCE\_S1 ; Handle debouncing by waiting.**

**push r16 ; Preserve r16.**

**lds r16, PORTF\_IN ; Check if S1 is still being pressed.**

**sbrc r16, 2**

**rjmp endInt ; If not, exit the ISR.**

**inc r19 ; If so, proceed to the next state.**

**cpi r19, NUM\_STATES**

**brlt endInt**

**ldi r19, OFF\_STATE**

**endInt:**

**ldi r16, CLEAR\_INT0\_FLAG ; If so, clear the inter**

**sts PORTF\_INTFLAGS, r16**

**pop r16**

**reti**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; Subroutine Name: USE32MHzCLK**

**; Sets the external 32MHz as the active clock for the device**

**; Inputs: r17 as the desired prescaler for the clock**

**; Outputs: None**

**; Affected: r16, r17**

**USE32MHzCLK:**

**push r16 ; Preserve the values of r16, r17.**

**push r17 ;**

**ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.**

**sts OSC\_CTRL, r16 ;**

**checkReady:**

**lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly**

**andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.**

**cpi r16, CLKEN ;**

**breq clockSel ; If it is move on, if not loop continuously.**

**rjmp checkReady ;**

**clockSel:**

**ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler**

**sts CPU\_CCP, r16 ; to be written to.**

**sts CLK\_PSCTRL, clkPrescaler ;**

**sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control**

**ldi r16, CLKSEL ; to be set to output the 32 MHz.**

**sts CLK\_CTRL, r16 ;**

**pop r17 ; Restore the values of r16 and r17.**

**pop r16 ;**

**ret ; return.**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; Subroutine Name: SETTIMER\_PWM**

**; Initialized TCD0 by setting its period and compare A value.**

**; Inputs: None**

**; Outputs: None**

**; Affected: r16,**

**SETTIMER\_PWM:**

**push r16 ; Preserve r16.**

**ldi r16, TCDSEL ; Enable the TC and set its period to be that of the system CLK.**

**sts TCD0\_CTRLA, r16**

**ldi r16, TCD\_ENPORTD\_SINGLESLOPE; Set the TC's PWM mode to "single slope".**

**sts TCD0\_CTRLB, r16**

**ldi r16, low(TCDPER) ; Load the period of the TC into the TC's period regs.**

**sts TCD0\_PER, r16**

**ldi r16, high(TCDPER)**

**sts TCD0\_PER+1, r16**

**pop r16 ; Restore r16.**

**ret ; Return.**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; Subroutine Name: DELAY\_100ms**

**; Delays 1ms**

**; Inputs: None**

**; Outputs: None**

**; Affected: r16**

**DELAY\_100ms:**

**push r16 ; Preserve r16.**

**ldi r16, TC\_SEL ; Enable the TC and set its period to be that of the system CLK.**

**sts TCC1\_CTRLA, r16**

**ldi r16, low(TC\_PER) ; Load the period of the TC into the TC's period regs.**

**sts TCC1\_PER, r16**

**ldi r16, high(TC\_PER)**

**sts TCC1\_PER+1, r16**

**DELAY\_100ms\_loop: ; Loop checking the TC's interrupt flags until**

**lds r16, TCC1\_INTFLAGS ; the overflow flag is set.**

**sbrs r16, 0**

**rjmp DELAY\_100ms\_loop**

**sts TCC1\_INTFLAGS, r16**

**ldi r16, TC\_DISABLE ; Break from the loop and disable the TC.**

**sts TCC1\_CTRLA, r16**

**pop r16 ; Restore r16.**

**ret ; Return.**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; Subroutine Name: INIT\_RGB**

**; Sets up the RGB LED**

**; Inputs: None**

**; Outputs: None**

**; Affected: r16**

**INIT\_RGB:**

**rcall SETTIMER\_PWM ; Set the TC used to operate on the RGB.**

**push r16 ; Preserve r16.**

**ldi r16, RGB\_DIRSET ; Set port D (RGB port) to output.**

**sts PORTD\_DIRSET, r16**

**ldi r16, RGB\_REMAP ; Remap bits 6-4 to TCD0's compare regs A, B, and C.**

**sts PORTD\_REMAP, r16**

**pop r16 ; Restore r16.**

**ret ; Return.**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; Subroutine Name: SET\_RGB**

**; Sets the value of the RGB LED**

**; Inputs: r16: Value of Red, r17: Value of Green, r18: Value of Blue**

**; Outputs: None**

**; Affected: r16, r17, r18**

**SET\_RGB:**

**push r16 ; Preserve r16.**

**sts TCD0\_CCA, r16 ; Set each of the compare regs to**

**ldi r16, 0x00 ; their respective RGB values.**

**sts TCD0\_CCA+1, r16**

**sts TCD0\_CCB, r17**

**sts TCD0\_CCB+1, r16**

**sts TCD0\_CCC, r18**

**sts TCD0\_CCC+1, r16**

**pop r16 ; Restore r16.**

**ret ; Return.**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; Subroutine Name: SET\_RGB\_OFF**

**; Turns the RGB off.**

**; Inputs: None**

**; Outputs: None**

**; Affected: r16**

**SET\_RGB\_OFF:**

**push r16 ; Preserve r16.**

**ldi r16, 0xFF ; Set r16 to a value outside**

**sts TCD0\_CCA, r16 ; the TC's period.**

**sts TCD0\_CCA+1, r16**

**sts TCD0\_CCB, r16 ; This prevents the RGB from**

**sts TCD0\_CCB+1, r16 ; ever being on.**

**sts TCD0\_CCC, r16**

**sts TCD0\_CCC+1, r16**

**pop r16 ; Restore r16.**

**ret ; Return.**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; Subroutine Name: INIT\_BUTTON\_S1\_INT**

**; Set up an interrupt for button S1.**

**; Inputs: None**

**; Outputs: None**

**; Affected: r16,**

**INIT\_BUTTON\_S1\_INT:**

**push r16 ; Preserve r16.**

**ldi r16, INT0\_LOW\_EN ; Set the interrupt’s priority**

**sts PORTF\_INTCTRL, r16 ; to low.**

**ldi r16, BUTTON\_S1\_INT\_TRIGG ; Set the bitmask to trigger the**

**sts PORTF\_INT0MASK, r16 ; interrupt on S1's bit.**

**ldi r16, PORT\_ISC\_FALLING\_gc ; Set the interrupt to trigger on**

**sts PORTF\_PIN2CTRL, r16 ; a falling edge as S1 is active low.**

**ldi r16, PMIC\_LOLVLEN\_bm ; Enable low priority interrupts.**

**sts PMIC\_CTRL, r16**

**pop r16 ; Restore r16.**

**ret ; Return.**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; Subroutine Name: DEBOUNCE\_S1**

**; Checks for bouncing on S1.**

**; Inputs: None**

**; Outputs: None**

**; Affected: r16**

**DEBOUNCE\_S1:**

**push r16 ; Preserve r16.**

**ldi r16, TCC0SEL ; Enable the TC and set its period to be 1024 times that of the system CLK.**

**sts TCC0\_CTRLA, r16**

**ldi r16, low(TCC0PER) ; Load the period of the TC into the TC's period regs.**

**sts TCC0\_PER, r16**

**ldi r16, high(TCC0PER) ;**

**sts TCC0\_PER+1, r16**

**checkS1Loop:**

**lds r16, TCC0\_INTFLAGS ; Wait until the TC's overflow flag is triggered**

**sbrs r16, 0**

**rjmp checkS1Loop**

**ldi r16, 0x01 ; Once broken from the loop, clear the flag, reset the**

**sts TCC0\_INTFLAGS, r16 ; count, and disable the timer.**

**ldi r16, 0x00**

**sts TCC0\_CNT, r16**

**ldi r16, TCC0DISABLE**

**sts TCC0\_CTRLA, r16**

**pop r16 ; Restore r16.**

**ret ; Return.**

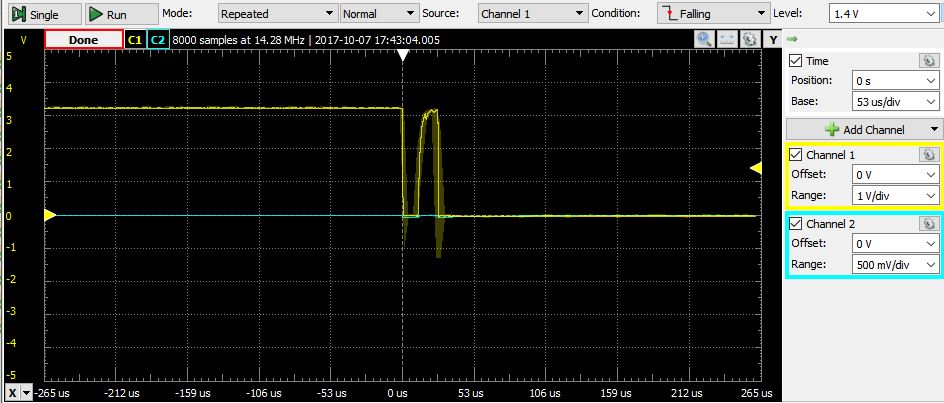
**H)** Appendix:

Files:

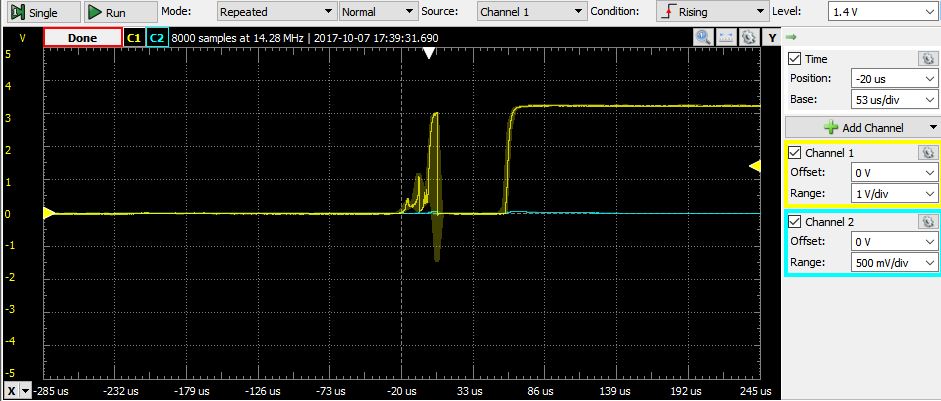
* Lab3.pdf
* Lab3a.asm
* Lab3c.asm
* Lab3d.asm
* Lab3e.asm

Screenshots:

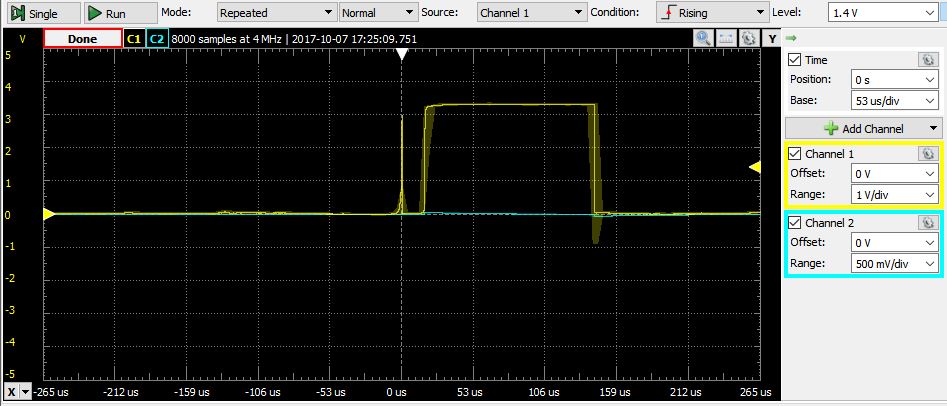
S1 pressed:



S1 released:



Switch S3(1) closed:



Switch S3(1) opened:

