Lab 4
Pengzhao Zhu
Section: 112D

# Initial Pseudocode (Lab 4)

## Part A Pseudocode:

Set up r23 to be 0x00 for the 32Mhz subroutine
rcall CLK  (to set up 32Mhz clock)

Initialize Stack Pointer to 0x3FFF

Set Blue LED to be output

Remap PORTD  ; move location of 0C02 from Px2 to Px6

Set the PER register of PORTD to 0xFF  ;0xFF for 0-255 of RGB LED

Set Timer clock on Port D to be CLK/1024

Use CTRLB to enable Compare/Capture C and enable single-slope PWM

Set TCD0_CCC to 0x0E

Invert the signal outputted to PORTD pin 6 using PORTD_PIN6CTRL (because LED is low-true)

DONE:
    rjmp DONE

CLK (32 MHZ subroutine)

push r16

set OSC_CTRL to be the 32 MHZ oscillator

NSTABLE:

Check if 32MHZ oscillator is stable

If stable, go to STABLE

If not stable, go back to NSTABLE

STABLE:

Write IOREG (0xD8) to CPU_CCP to enable change

Select the 32 MHZ oscillator

Write IOREG (0XD8) to CPU_CCP to use prescaler

Use r23 initialized outside the subroutine to set it up so it remains 32Mhz

pop r16

ret

## Part C Pseudocode:

```
.org 0x0000
        rjmp MAIN
```

set up ISR name to jump to when encountered PORTF_INTO_vect

```
.org 0x100
```

MAIN:

Set up r17 to be a counter register for how many time the ISR has been executed

Set up r23 to be 0x00 for the 32Mhz subroutine

rcall CLK

Initialize Stack Pointer to 0x3FFF

Set 8 LED on PORTC to be output

Turn the 8 active low LED off

Enable low level interrupt for INT0

Set PF2 (Tactile switch S1) as the source for INT0

Set PF2 (Tactile switch S1) as input

Configure Pin2 of PortF to be falling edge trigger using PORTF_PIN2CTRL

Enable low level interrupt in the PMIC

sei

ldi r16, 0x70  ; to toggle the LED in an infinite loop

LOOP:

```
        sts PORTD_OUTTGL, r16
```

rjmp LOOP

ISR: ;ISR to output count to 8 LED

push r19

push r18

put CPU_SREG into r18

push r18

increment r17

Takes one's complement of r17 because LED are active low

sts PORTC_OUT, r17

Takes one's complement of r17 to ensure correct count

Clear interrupt flag using PORTF_INTFLAGS. Write a one to it

pop r18

Put value of r18 back to CPU_SREG

Pop r18

Pop r19

reti

CLK (32 MHZ subroutine)

push r16

set OSC_CTRL to be the 32 MHZ oscillator

NSTABLE:

Check if 32MHZ oscillator is stable

If stable, go to STABLE

If not stable, go back to NSTABLE

STABLE:

Write IOREG (0xD8) to CPU_CCP to enable change

Select the 32 MHZ oscillator

Write IOREG (0XD8) to CPU_CCP to use prescaler

Use r23 initialized outside the subroutine to set it up so it remains 32Mhz

pop r16

ret

## Part D Pseudocode:

```
.org 0x0000
        rjmp MAIN
set up ISR name to jump to when encountered PORTF_INTO_vect
set up TIMER_ISR name to jump to when encountered TCD0_OVF_vect
.equ debounce_timer= (32000000*.005)/1024
.org 0x100
Set up r23 to use in the 32Mhz subroutine
Set r17 to be a register for how many times the ISR has been executed
rcall CLK
Initialize Stack Pointer to 0x3FFF
Set 8 LED on PORTC to be output
Turn the 8 active low LED off
rcall DEBOUNCE (to set up interrupt)
ldi r16, 0x70    ; to toggle the LED in an infinite loop
LOOP:
        sts PORTD_OUTTGL, r16
        rjmp LOOP

DEBOUCNE:
Push r16
```

Enable low level interrupt using PORTF_INTCTRL

Set PF2 (Tactile switch S1) as the source for INT0

Set PF2 (Tactile switch S1) as input

Configure the source to be falling edge trigger using PORTF_PIN2CTRL

Enable low level interrupt in the PMIC

Sei

Pop r16

Ret

ISR:

Push r18

Put CPU_SREG into r18

Push r18

Push r16

Setting the CNT back to zero

Set the PER for 5ms (math before the main code)

Configure timer clock for CLK/1024

Enable low level timer interrupt

Disable PORTF external interrupt

Pop r16

Pop r18

Value in r18 back to CPU_SREG

Pop r18

Reti

Timer_ISR:

Push r18

Value in CPU_SREG t or18

Push r18

Push r16

Disable Timer

Disable Timer Interrupt

Check if switch is set

brts NOT_ACTIVE

increment r17

Take one's complement of r17

Value in r17 to PORTC_OUT

Take one's complement of r17

NOT_ACTIVE:

Enable PORTF external interrupt

Clear the Interrupt Flag

Pop r16

Pop r18

Value in r18 to CPU_SREG

Pop r18

reti

CLK (32 MHZ subroutine)

push r16

set OSC_CTRL to be the 32 MHZ oscillator

NSTABLE:

Check if 32MHZ oscillator is stable

If stable, go to STABLE

If not stable, go back to NSTABLE

STABLE:

Write IOREG (0xD8) to CPU_CCP to enable change

Select the 32 MHZ oscillator

Write IOREG (0XD8) to CPU_CCP to use prescaler

Use r23 initialized outside the subroutine to set it up so it remains 32Mhz

pop r16

ret

## Part E Pseudocode:

```
.org 0x0000
        rjmp MAIN
```

set up ISR name to jump to when encountered PORTF_INTO_vect

set up TIMER_ISR name to jump to when encountered TCC0_OVF_vect

set up TIMER_BLINK_ISR name to jump to when encountered TCD0_OVF_vect

.equ debounce_timer= (32000000*.005)/1024

Put values of RGB into a Table in Program Memory

.org 0x100

Set up r23 to use in the 32Mhz subroutine

Set r17 to be a register for how many times the ISR has been executed

rcall CLK

Initialize Stack Pointer to 0x3FFF

Set 8 LED on PORTC to be output

Turn the 8 active low LED off

Check Value r17

If r17 = 0 or 4 {

Branch to LED_OFF

}

else if r17= 1 or 5 {

Branch to INCREDIBLE HULK

}

else if r17 = 2 or 6 {

Branch to HOLIDAY

}

else if r17=3 or 7 {
Branch to UF

}

INCREDIBLE HULK:

Push necessary register

Set up LED it blinks between INCREDIBLE HULK PURPLE and BLUE

Pop necessary register

ret

HOLIDAY:

Push necessary register

Set up LED so it blinks between HOLIDAY RED and HOLIDAY GREEN

Pop necessary register

Ret

UF:

Push necessary register

Set up LED so it blinks between UF ORANGE and UF BLUE

Pop necessary register

Ret

ISR:  ; to trigger when S1 is pressed

Initialize debounce timer for .5 ms

Enable Timer interrupt

Disable Port Interrupt

Reti

TIMER_ISR:

Disable Timer

Disable Timer Interrupt

Enable Port Interrupt

Reti

TIMER_BLINK_ISR:

Disable Timer

Disable Timer interrupt

Move to the next value in table because .1 second has been pressed already

Reti

CLK (32 MHZ subroutine)

push r16

set OSC_CTRL to be the 32 MHZ oscillator

NSTABLE:

Check if 32MHZ oscillator is stable

If stable, go to STABLE

If not stable, go back to NSTABLE

STABLE:

Write IOREG (0xD8) to CPU_CCP to enable change

Select the 32 MHZ oscillator

Write IOREG (0XD8) to CPU_CCP to use prescaler

Use r23 initialized outside the subroutine to set it up so it remains 32Mhz

```
pop r16
ret
```