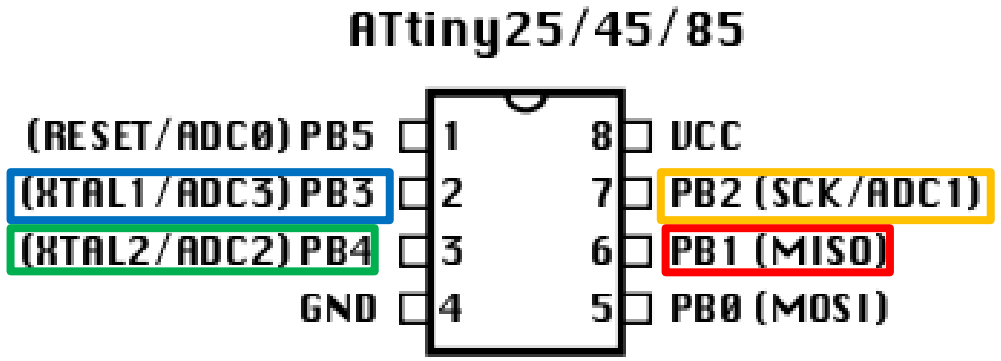
eLantern Project

Hardware and Software

Andrew Krock

May 20,2015

Pinouts

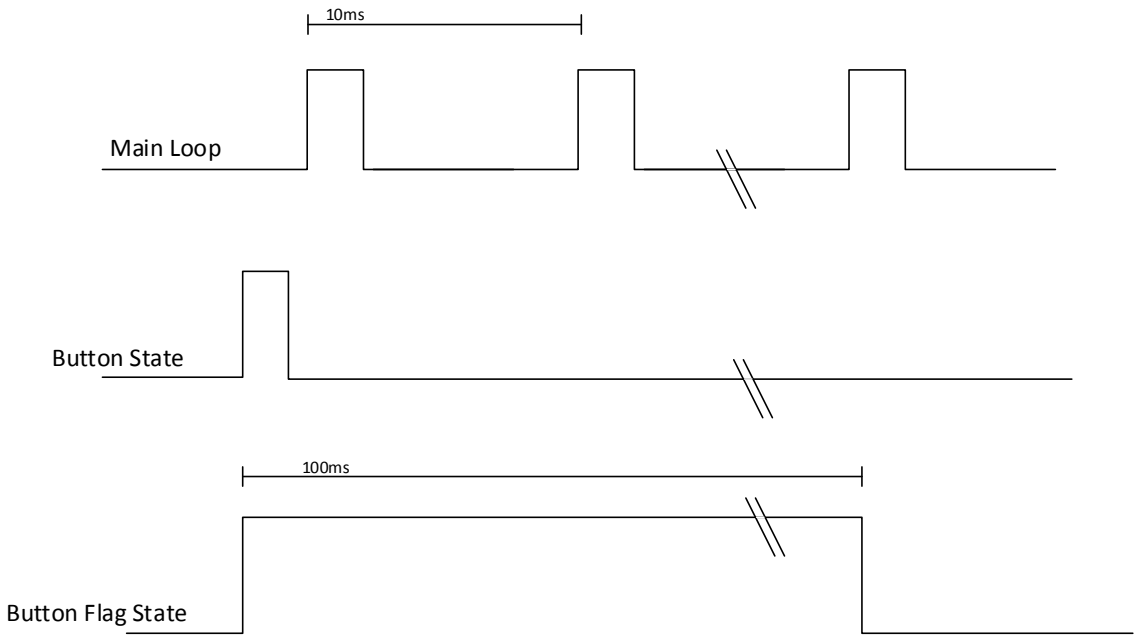



```
//Light Pin
#define LIGHT_PIN                   (1 << PORTB1)

//Buttion Pin
#define BUTTON_PIN                  (1 << PORTB3)
```

```
#define SOFTUART_RXPIN    PINB
#define SOFTUART_RXDDR    DDRB
#define SOFTUART_RXBIT    PB4

#define SOFTUART_TXPORT   PORTB
#define SOFTUART_TXDDR    DDRB
#define SOFTUART_TXBIT    PB2
```

Main Loop

```
int main(void){
        timer_init();
        interrupt_init();
        button_init();
        light_init();
        softuart_init();
        while(1){
                if(get_ticks() > eLanternServicePeriod){        //Does these jobs every period defined
                        eLanternServicePeriod += 10;    //Update the service period
                        button_status();
                        light_status();
                        print_light_state();
                }
/*              if(get_ticks() > testPeriod){
                        testPeriod += 1000;
                        softuart_putchar('1');
                }*/
        }
        return 0;
}
```



MCU Hardware Configuration

```
//Initializes timer to CTC for 1 ms period
void timer_init(){
        cli();
        //Start moving over to timer1
        TCCR1 = (CTC1x)|(PWM1Ax)|(COM1A1x)|(CS11x)|(CS10x);
        OCR1C = 250;
        TIMSK = (TOIE1x); //Overflow int
        OCR1A = 0;
        //To do PWM set OCR1A to a value
        sei();
}
```

```
//Initializes the interrupt vector
void interrupt_init(){
        cli();
        GIMSK = (PCIEx);
        PCMSK = (PCINT3x);
        sei();
}
```

ISR Code

```
ISR(PCINT0_vect){
        if(button_state == DEBOUNCING){
                button_flag = 0;
        }
        else{
                button_flag = 1;
        }
}
```

```
//Interrupts every 1 ms and adds a tick
ISR(TIMER1_OVF_vect){
        ticks ++;
        debounce_timer ++;
        sleep_timer ++;
        select_timer ++;
        runtime_timer ++;
        fade_timer ++;
}
```

Modules

- timer.c/.h
  - File handles timer init
  - File also keeps track of tick values
- button_state.c/.h
  - File handles button and interrupt init
  - File also handles the debouncing of the piezo
- light_state.c/.h
  - File handles light pin init
  - File has the state machine that controls the state of the light

timer.c/.h

```
/*-------------------------------------------------------
* Author :          Andrew Krock
* Filename :        timer.h
* Date Created :    Monday March 23, 2015 07:59:34 PM
* Last Edited :     Saturday May 09, 2015 01:15:44 PM
* Description :
-------------------------------------------------------*/

#ifndef TIMER_H
#define TIMER_H

void timer_init();
unsigned int get_ticks();
unsigned int get_debounce();
unsigned int get_sleep();
unsigned int get_select();
unsigned int get_runtime();
unsigned int get_fade();

extern unsigned int ticks;
extern unsigned int debounce_timer;
extern unsigned int sleep_timer;
extern unsigned int select_timer;
extern unsigned int runtime_timer;
extern unsigned int fade_timer;

#endif //TIMER_H
```

button_state.c/.h

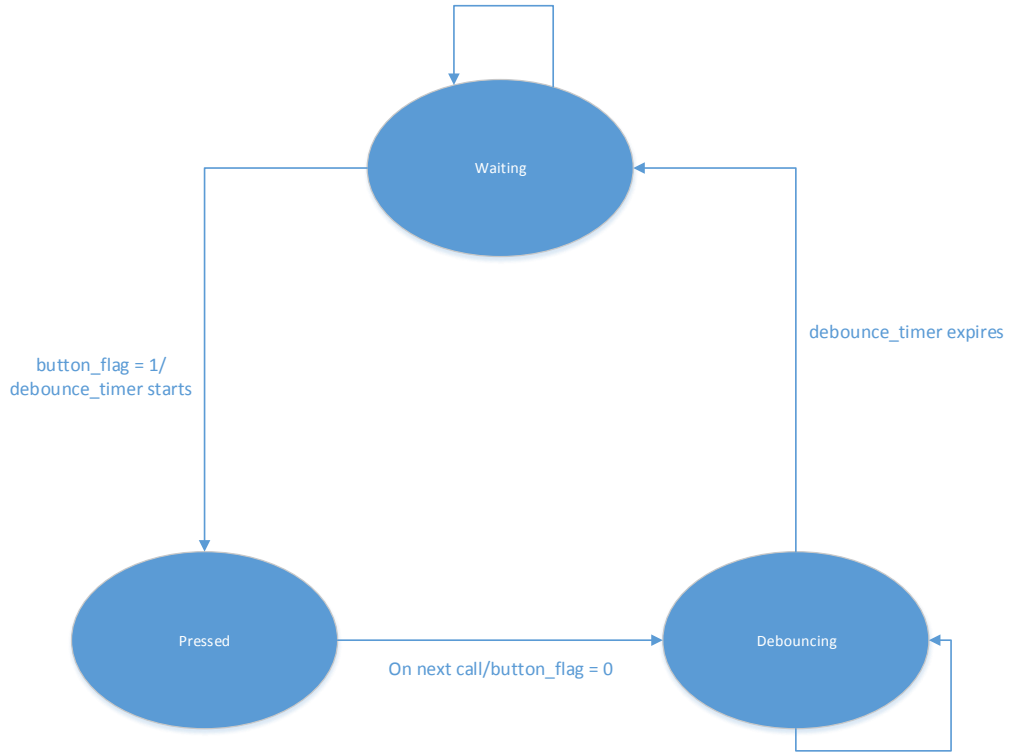
```
/*-------------------------------------------------------
* Author :          Andrew Krock
* Filename :        button_state.h
* Date Created :    Thursday March 26, 2015 01:34:01 PM
* Last Edited :     Thursday May 14, 2015 03:39:58 PM
* Description :
-------------------------------------------------------*/

#ifndef BUTTON_STATE_H
#define BUTTON_STATE_H

void button_init();
void interrupt_init();
void button_status();

extern unsigned int button_flag;
#endif //BUTTON_STATE_H
```

light_state.c/.h

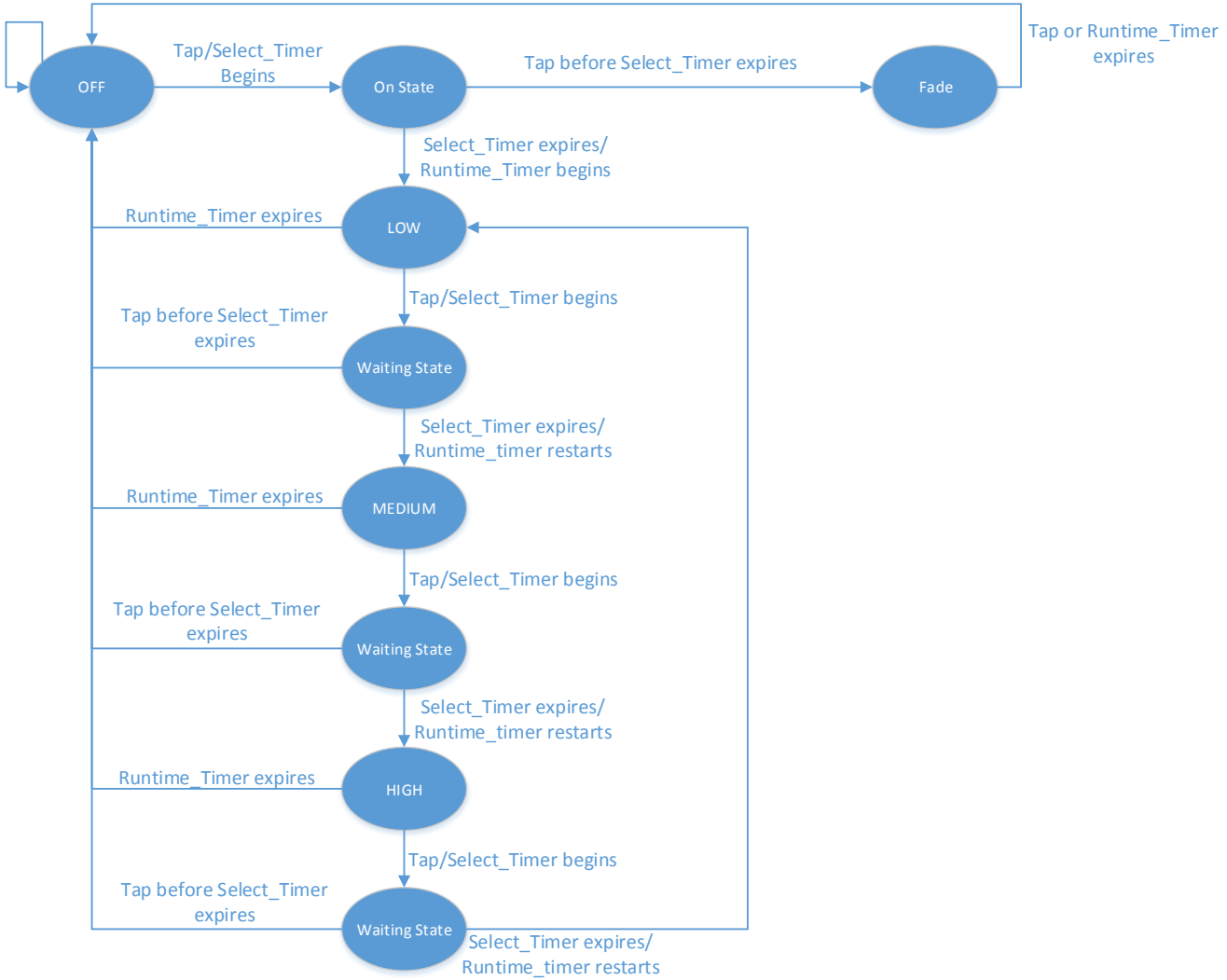
```
/*-------------------------------------------------------
* Author :          Andrew Krock
* Filename :        light_state.h
* Date Created :    Thursday March 26, 2015 08:51:44 PM
* Last Edited :     Thursday May 14, 2015 03:40:00 PM
* Description :
-------------------------------------------------------*/

#ifndef LI_H
#define LI_H

void light_init();
void light_status();
void print_light_state();

#endif //LI_H
```

Debounce State Machine

```
//Debounce switch statement
void button_status(){
        switch(button_state){
                case WAITING:
                        if(button_flag == 1){
                                debounce_timer = 0;
                                button_state = PRESSED;
                        }
                        break;
                case PRESSED:
                        button_flag = 0;
                        button_state = DEBOUNCING;
                        break;
                case DEBOUNCING:
                        if(get_debounce() > DEBOUNCE_TIME){
                                button_state = WAITING;
                        }
                        break;
                default:
                        break;
        }
}
```



Application State Machine



Application State Machine

```
//Finite state machine that controls what the
//light is doing
void light_status(){
        switch(light_state){
                case OFF:
                        OCR1A = OFF;
                        if(button_flag == 1){
                                light_state = ON_STATE;
                                select_timer = 0;
                        }
                        //if(get_sleep() > SLEEP_TIME){
                        //      set_sleep_mode(SLEEP_MODE_PWR_DOWN);
                        //      sleep_mode();
                        //}
                        break;
                case ON_STATE:
                        if(button_flag == 1 && get_select() < SELECT_TIME){
                                OCR1A = OFF;
                                fade_timer = 0;
                                runtime_timer = 0;
                                light_state = FADE;
                        }
                        else if(get_select() > SELECT_TIME){
                                light_state = LOW;
                                runtime_timer = 0;
                        }
                        break;
                case LOW:
                        OCR1A = LIGHT_LOW;
                        if(button_flag == 1){
                                light_state = WAITING_STATE_1;
                                select_timer = 0;
                        }
                        if(get_runtime() > RUNTIME){
                                light_state = OFF;
                        }
                        break;
                case WAITING_STATE_1:
                        if(button_flag == 1 && get_select() < SELECT_TIME){
                                light_state = OFF;
                        }
                        else if(get_select() > SELECT_TIME){
                                runtime_timer = 0;
                                light_state = MEDIUM;
                        }
                        break;
                case MEDIUM:
                        OCR1A = LIGHT_MEDIUM;
```

Next Steps

- Clean up Application State Machine
  - Make a function that would count taps
- Make exit of FADE state two taps and not one
- Add the ability to change the speed of the fading
- Continual clean up of code