

Lab 1 - Microcontrollers

Sheil Sarda sheils@seas.upenn.edu

1. LED

1.1.1 Ranges of Resistance for LED

[Digikey datasheet for 160-1128-ND LTL-4224 5mm red LED](#)

From Page 3:

Forward Voltage	V_F	2.0	2.6	V	$I_F = 20\text{mA}$
-----------------	-------	-----	-----	---	---------------------

Calculate the smallest resistor that you can safely use with this LED in series with 5V power.

Continuous Forward Current (I_f)	20 mA
Battery Voltage (V)	5 V
Forward Voltage (V_f)	2-2.6 V
Max Voltage across resistor	$5 - 2 = 3\text{V}$
Resistance (R)	$3\text{V} / 20\text{mA} = 150 \text{ Ohms}$

1.1.2 Optimal Resistance for Dim Light

Resistance for Dim Light = 10 KOhms

Battery Voltage (V)	5 V
Forward Voltage (V_f)	2-2.6 V
Max Voltage across resistor	$5 - 2 = 3\text{V}$
Resistance (R)	10 KOhms
Current (I_f)	$3\text{V} / 10 \text{ KOhms} = 0.3 \text{ mA}$

1.1.3 Two LEDs in Parallel

LEDs chosen

LED Type	Forward Voltage	Current
160-1128-ND LTL-4224 5mm red LED	2V	20 mA
160-1132-ND LTL-4238 5mm green LED	2.1V	20 mA

Based on the specifications, I would expect the Red LED to be slightly brighter than the green LED because of the lower forward voltage drop.

Measured voltage at A: **1.90 V**. The Red LED is slightly brighter.

1.1.4 Two LEDs in Series

Measured voltage at A: **3.81V**.

Voltage between the 2 LEDs: **1.95V**

This is expected based on the datasheets, because due to current being the same for both LEDs in series, causing the voltage drop in the middle of the two to be approximately half of the voltage at A.

1.1.5 Optimal LED for a Table Lamp

American Bright Optoelectronics Corporation AB-EZP10A-B3-K13

Cost: \$1.91

Size / Dimension: 0.118" L x 0.118" W (3.00mm x 3.00mm)

Features:

- Luminous intensity remains constant for $10.4V < V_{in} < 24V$
- Can actively modulate the power usage by decreasing the input current to avoid overheating.
- The input current, light output, and color temperature can stay consistent despite the exterior voltage changes or fluctuations.
- Uses conventional 12 V / 24 VDC voltage sources

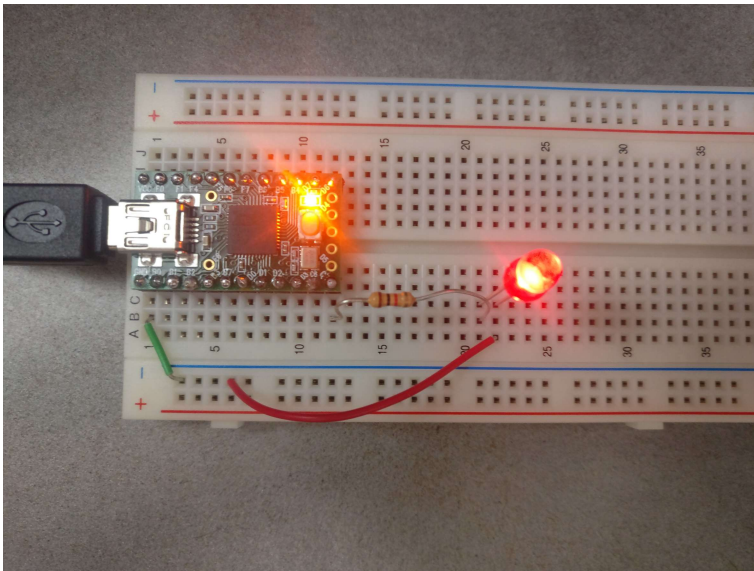
2. Introduction to Teensy

1.2.1 Modified Registers

- Data Direction Register `DDRC`
- GPIO Pinout `PORTC`

1.2.2 Circuit Diagram

Circuit Diagram



Choice of Hardware

- Used 1kOhm current limiting resistor
- Used ground pin from the Teensy (which comes from laptop)
- Used Red LED since it was easiest to detect whether its on / off

1.2.3 Variable Frequency

Frequency appears to be continually on at around 60Hz

C Code

```
#include "teensy_general.h" // includes the resources included in the teensy_general.h file

#define FREQ_HZ 60 // variable for frequency

int main(void)
{
    /* insert your hardware initialization here */

    teensy_clockdivide(0); //set the clock speed
    set(DDRC, 7);

    for(;;){
        toggle(PORTC, 7);
        teensy_led(TOGGLE); // switch the led state
        teensy_wait(1000/FREQ_HZ); // wait
    }
    return 0; /* never reached */
}
```

1.2.4 Variable Duty Cycle

```
#include "teensy_general.h" // includes the resources included in the teensy_general.h file

#define FREQ_HZ 10 // variable for frequency
#define DUTY_CYCLE 50 // variable for duty cycle percentage

int main(void)
{
```

```

teensy_clockdivide(0); //set the clock speed
set(DDRC, 7);

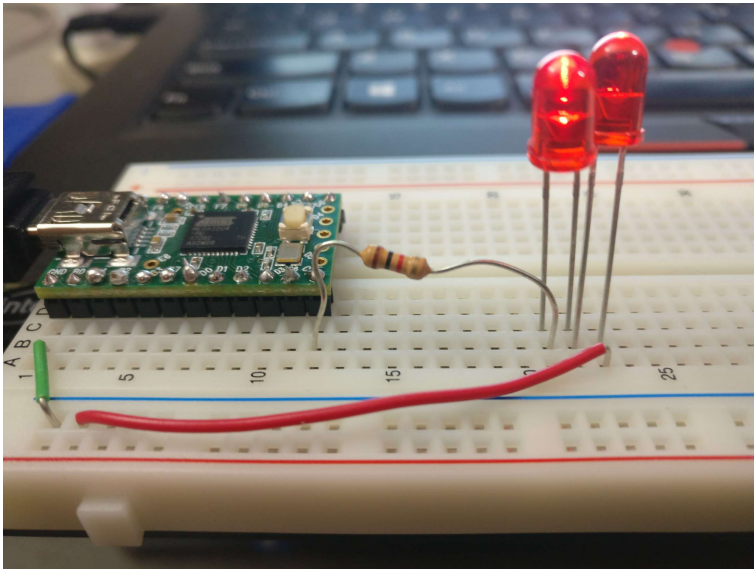
for(;;){
    /* ON Part of the Duty Cycle */
    toggle(PORTC, 7);
    teensy_wait(10 * DUTY_CYCLE / FREQ_HZ);          // wait

    /* OFF Part of the Duty Cycle */
    clear(PORTC, 7);
    teensy_wait(10 * (100 - DUTY_CYCLE) / FREQ_HZ);    // wait
}
return 0;      /* never reached */
}

```

1.2.5 Maximum LEDs from a GPIO Pin

Circuit Diagram



Limitations

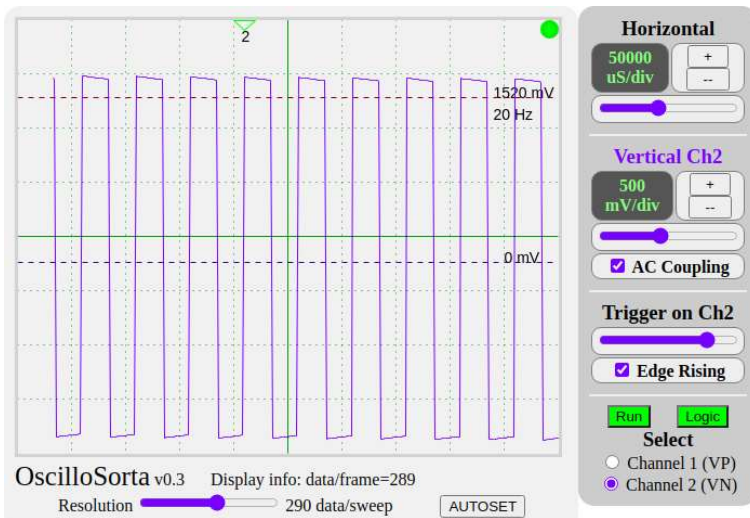
Since each red LED uses 2V, and each GPIO pin of the Teensy outputs 5V and 20 mA we can connect up to 2 LEDs at visible brightness (since $2V + 2V < 5V$).

The GPIO output voltage of the Teensy is the limiting factor.

3. Timers

1.3.1 Use the timer to get 20Hz

Oscilloscope Output



C Code

```
#include "teensy_general.h" // includes the resources included in the teensy_general.h file

#define FREQ_HZ 20 // variable for frequency
#define PRESCALAR 256 // prescaler used
#define SYS_CLOCK 16e6 // clock speed (16 Mhz)
int main(void)
{
    /* insert your hardware initialization here */

    teensy_clockdivide(0); //set the clock speed
    set(DDRC, 7);

    set(TCCR1B, CS12); // set 256 prescaler

    TCNT1 = 0x00;
    long cutoff = SYS_CLOCK/(2*FREQ_HZ*PRESCALAR);

    for(;;){

        if(TCNT1 > cutoff){
            toggle(PORTC, 7);
            teensy_led(TOGGLE);
            TCNT1 = 0x00;
        }
    }
    return 0; /* never reached */
}
```

1.3.2 Adjust the Pre-scaler

Adjusted the pre-scaler in the code snippet for 1.3.1.

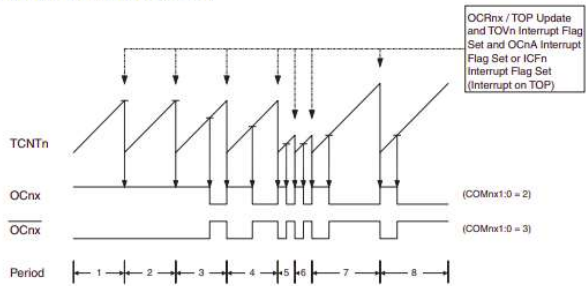
The default system clock frequency is 16MHz.

1.3.3 PWM functions of timer

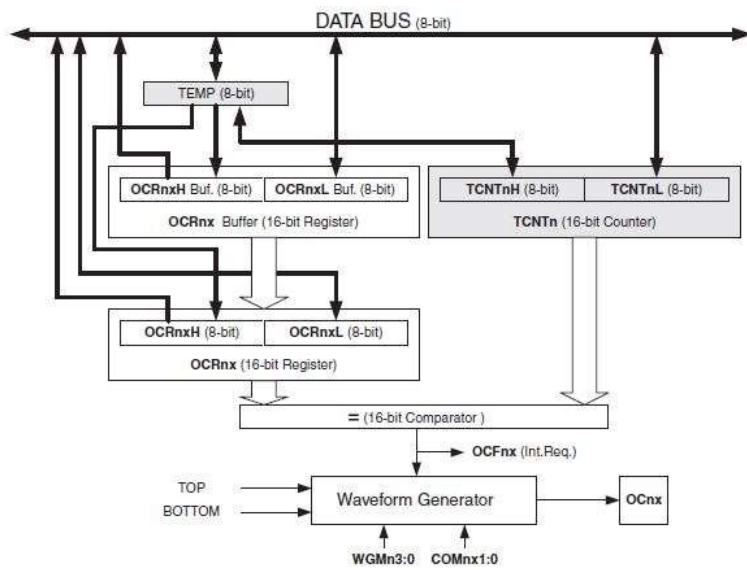
Do the same thing you did in 1.2.4

Timer options used

Figure 14-7. Fast PWM Mode, Timing Diagram

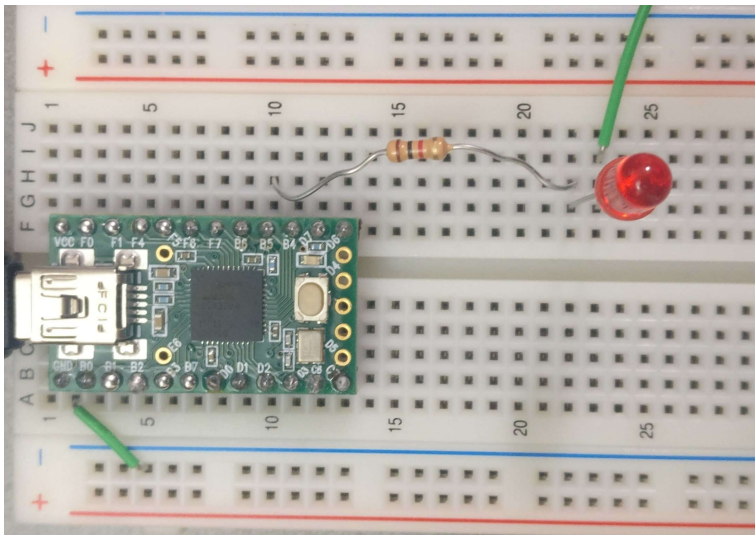


- Output compare is set on the compare match between TCNT1 and OCR1A
- Mode 14 on Timer 1 to count up to ICR1
- Toggling Port B5 at Output Compare Match
- Set Compare match registers ICR1 and OCR1A
- 256x prescaler
- Set OC1A on compare match when up-counting and clear on compare match

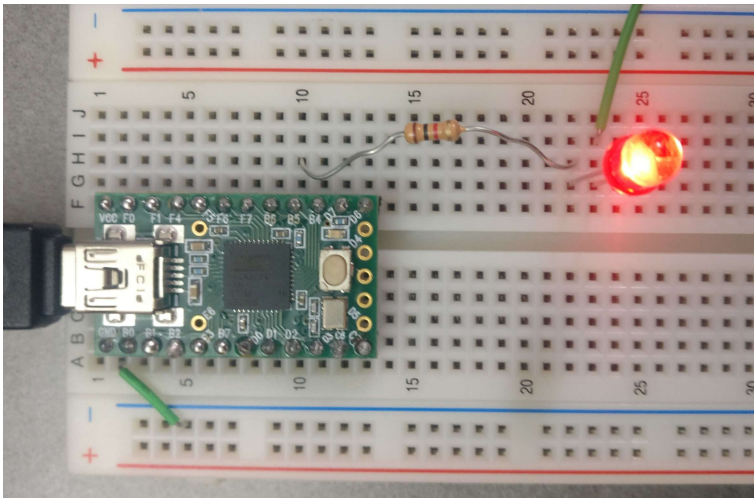


Duty Cycles

0% cycle



100% cycle



C Code

```
#include "teensy_general.h" // includes the resources included in the teensy_general.h file

#define FREQ_HZ    1    // variable for frequency
#define PRESCALAR  256  // prescaler used
#define SYS_CLOCK  16e6 // clock speed (16 Mhz)
#define DUTY_CYCLE 1    // duty cycle

int main(void)
{
    set(TCCR1B, CS12); // set 256 prescaler
    teensy_clockdivide(0); //set the clock speed

    set(DDRB, 5); // B5 is output compare pin

    // (mode 14) UP to ICR1, PWM mode
    set(TCCR1B, WGM13); set(TCCR1B, WGM12); set(TCCR1A, WGM11);

    // clear B5 at OC (to start off part of duty cycle)
    set(TCCR1A, COM1A1);

    // set compare match register
    ICR1 = SYS_CLOCK/(FREQ_HZ*PRESCALAR);
    // set duty cycle PWM
    OCR1A = ICR1*DUTY_CYCLE;

    while(1);
    return 0;    /* never reached */
}
```

Video Demonstration

4. Practice with Loops

1.4.1 Make the External LED Pulse

Part A

The pulse should start at 0 intensity, take 1 second to increase in intensity until it is full brightness, then 1 second to decrease in brightness and repeat.

```

#include "teensy_general.h" // includes the resources included in the teensy_general.h file

#define FREQ_HZ    50    // variable for frequency
#define PRESCALAR  256   // prescalar used
#define SYS_CLOCK  16e6  // clock speed (16 Mhz)

int main(void)
{
    set(TCCR1B, CS12); // set 256 prescalar
    teensy_clockdivide(0); //set the clock speed

    set(DDRB, 5); // B5 is output compare pin

    // (mode 14) UP to ICR1, PWM mode
    set(TCCR1B, WGM13); set(TCCR1B, WGM12); set(TCCR1A, WGM11);

    // toggle B5 at OC
    set(TCCR1A, COM1A1);

    /* Set OC1A on compare match when upcounting. Clear OC1A
     * on compare match when down-counting.
     */
    set(TCCR1B, WGM13); set(TCCR1B, WGM12);

    // set compare match register
    ICR1 = SYS_CLOCK/(FREQ_HZ*PRESCALAR);

    double arr[] = {0, 0.05, 0.2, 0.5, 1, 1}; // duty cycles
    int len = (sizeof(arr) / sizeof(double));
    int delay_time = 1000/len; // how much time to spend at each duty cycle

    while(1){
        // rising
        for(int i = 0; i < len; ++i){
            OCR1A = ICR1*arr[i];
            _delay_ms(delay_time);
        }
        // falling
        for(int i = len-1; i >= 0; --i){
            OCR1A = ICR1*arr[i];
            _delay_ms(delay_time);
        }
    }

    return 0;    /* never reached */
}

```

Part B

- Make the time to increase and the time to decrease a variable.
- Make a video with the following configuration:
 - 0.3 second to full intensity
 - 0.7 seconds to 0 intensity

Video of Asymetric Pulses

C Code

```

#include "teensy_general.h" // includes the resources included in the teensy_general.h file

#define FREQ_HZ    50    // variable for frequency
#define PRESCALAR  256   // prescalar used
#define SYS_CLOCK  16e6  // clock speed (16 Mhz)

```



```

#define RISE_TIME 300 // time in ms to full intensity
#define FALL_TIME 700 // time in ms to 0 intensity

int main(void)
{
    set(TCCR1B, CS12); // set 256 prescaler
    teensy_clockdivide(0); //set the clock speed

    set(DDRB, 5); // B5 is output compare pin

    // (mode 14) UP to ICR1, PWM mode
    set(TCCR1B, WGM13); set(TCCR1B, WGM12); set(TCCR1A, WGM11);

    // toggle B5 at OC
    set(TCCR1A, COM1A1);

    /* Set OC1A on compare match when upcounting. Clear OC1A
     * on compare match when down-counting.
     */
    set(TCCR1B, WGM13); set(TCCR1B, WGM12);

    // set compare match register
    ICR1 = SYS_CLOCK/(FREQ_HZ*PRESCALAR);

    double arr[] = {0, 0.05, 0.2, 0.5, 1, 1};
    int len = (sizeof(arr) / sizeof(double));

    // time to spend at each duty cycle
    int rise_ms = RISE_TIME/len;
    int fall_ms = FALL_TIME/len;

    while(1){
        // rising
        for(int i = 0; i < len; ++i){
            OCR1A = ICR1*arr[i];
            _delay_ms(rise_ms);
        }
        // falling
        for(int i = len-1; i >= 0; --i){
            OCR1A = ICR1*arr[i];
            _delay_ms(fall_ms);
        }
    }

    return 0; /* never reached */
}

```

Circuit Diagram


```

* `RISE_TIME`, `FALL_TIME`, `MAX_INTENSITY`,
* `FREQ_HZ`, `PRESCALAR` and `SYS_CLOCK`
* Assumes LED is plugged into Port `B5`
* Uses Timer 1 for PWM functionality
*/
void pulse_led(){
    set(TCCR1B, CS12); // set 256 prescalar
    teensy_clockdivide(0); //set the clock speed

    set(DDRB, 5); // B5 is output compare pin

    // (mode 14) UP to ICR1, PWM mode
    set(TCCR1B, WGM13); set(TCCR1B, WGM12); set(TCCR1A, WGM11);

    // toggle B5 at OC
    set(TCCR1A, COM1A1);

    /* Set OC1A on compare match when upcounting. Clear OC1A
    * on compare match when down-counting.
    */
    set(TCCR1B, WGM13); set(TCCR1B, WGM12);

    // set compare match register
    ICR1 = SYS_CLOCK/(FREQ_HZ*PRESCALAR);

    int len = (sizeof(intensity) / sizeof(double));

    // time to spend at each duty cycle
    int rise_ms = RISE_TIME/(len*LERP);

    int lerp_i = 0;
    double lerp_intensity;

    while(1){
        for(int i = 0; i < len; ++i){
            // LERP between values
            if(i < len - 1){

                for(lerp_i = 0; lerp_i < LERP; ++lerp_i){

                    lerp_intensity = (intensity[i+1] - intensity[i])*(lerp_i/LERP);
                    lerp_intensity += intensity[i];

                    OCR1A = ICR1*lerp_intensity*MAX_INTENSITY;
                    _delay_ms(rise_ms);
                }
            } else { // last value in array
                OCR1A = ICR1*intensity[i]*MAX_INTENSITY;
                _delay_ms(rise_ms);
            }
        }
    }
}

int main(void)
{
    pulse_led();
    return 0; /* never reached */
}

```

1.4.3 Extra Credit Weaker Heartbeat

- Heartbeat stays at a constant frequency, but maximum intensity slowly fades as if the heartbeat is getting weaker
- Have it take 20 beats before it is reduced to 0 intensity
- Show a TA your LED blinking; answer his/her questions, and get signed off

C Code

[illegible]

```

        _delay_ms(rise_ms);

    }
} else { // last value in array
    OCR1A = ICR1*intensity[i]*max_intensity;
    _delay_ms(rise_ms);
}
}
if(beat_count > TOTAL_BEATS) break;
beat_count++;
}
}

int main(void)
{
    pulse_led();
    return 0; /* never reached */
}

```

5. Retrospective

- Spent approximately 20 hours on this lab.
- Spent 10+ hours on the Timers and Loops part of the lab combined
- Additionally, spent 5 hours on the Extra Credit