Assignment 2.1

Steinarr Hrafn Höskuldsson

August 30, 2022

Part 1

I created timer_msec.h and timer_msec.cpp as can be seen in appendix A.

The `init()` method sets the timer up to fire every second. Running the main.cpp program given in the assignment results in the LED blinking at a frequency of 0.5Hz as expected.

The `init(int period_ms)` method sets the timer up to fire every period_ms milliseconds. The maximum length of interval the timer supports is $\lfloor 2^{16} * 1024/16000 \rfloor = 4194$ milliseconds.

Part 2

I modified the example to include a `duty_cycle` option. The code can be seen in Appendix A. with the duty cycle set to 20% I reduced the timer period until I stopped noticing the light blinking. I found it to be around 26 or 27 ms. 28 ms was visibly flickering and 25 ms looked solid.

Part 3

I added a method to the timer class called `set` that changes the duty cycle. I the used the while loop present in the `main` function to vary the duty cycle. Running the program I observed the LED fading from very dim to very bright over the course of 5 seconds before abruptly turning off and repeating the fade in.

A Code

```
class Timer_msec
{
public:
    Timer_msec();
    void init();
    void init(int period_ms);
    void init(int period_ms, float duty_cycle);
    void set(float duty_cycle);
};
```

Listing 1: timer_msec.h

```
#include <timer_msec.h>
#include <avr/interrupt.h>

Timer_msec::Timer_msec(){}

void Timer_msec::init(){
    // could be:
    // init(1000);
    // instead, the basics, from slides L2.2:
```

```
    // this code sets up timer1 for a 1s @ 16Mhz Clock (mode 4)
    // counting 16000000/1024 cycles of a clock prescaled by 1024
    TCCR1A = 0;                             // set timer1 to normal operation (all
    bits in control registers A and B set to zero)
    TCCR1B = 0;                             //
    TCNT1 = 0;                              //  initialize counter value to 0
    OCR1A = 16000000 / 1024 - 1;            // assign target count to compare
    register A (must be less than 65536)
    TCCR1B |= (1 << WGM12);                 // clear the timer on compare match A
    TIMSK1 |= (1 << OCIE1A);                // set interrupt on compare match A
    TCCR1B |= (1 << CS12) | (1 << CS10); // set prescaler to 1024 and start the
    timer
    sei();                                  // enable interrupts
}

void Timer_msec::init(int period_ms){
    // could be:
    //init(period_ms, 0.5);
    // instead, the solution to part 1:

    // period_ms has to be 4194 or smaller since  (2**16 -1)*1.024/16.000.000 =
    int max_period_ms = 4194;
    if (period_ms > max_period_ms)
    {
        period_ms = max_period_ms;
    }
    // total amount of clock pulses (also convert ms to seconds)
    uint32_t total = (uint32_t)16000 *period_ms;
    // target after taking prescaler into account
    uint16_t target = total / 1024 - 1;

    // this code sets up timer1 for a 1s @ 16Mhz Clock (mode 4)
    // counting 16000000/1024 cycles of a clock prescaled by 1024
    TCCR1A = 0;                             // set timer1 to normal operation (all
    bits in control registers A and B set to zero)
    TCCR1B = 0;                             //
    TCNT1 = 0;                              //  initialize counter value to 0
    OCR1A = target;                         // assign target count to compare
    register A (must be less than 65536)
    TCCR1B |= (1 << WGM12);                 // clear the timer on compare match A
    TIMSK1 |= (1 << OCIE1A);                // set interrupt on compare match A
    TCCR1B |= (1 << CS12) | (1 << CS10); // set prescaler to 1024 and start the
    timer
    sei();                                  // enable interrupts
}

void Timer_msec::init(int period_ms, float duty_cycle)
{
    // period_ms has to be 4194 or smaller since  (2**16 -1)*1.024/16.000.000 =
    int max_period_ms = 4194;
    if (period_ms > max_period_ms)
    {
        period_ms = max_period_ms;
    }
    // total amount of clock pulses (also convert ms to seconds)
    uint32_t total = (uint32_t)16000 * period_ms;
    // target after taking prescaler into account
    uint16_t target = total/1024 - 1;

    // this code sets up timer1 for a 1s @ 16Mhz Clock (mode 4)
```

```
    // counting 16000000/1024 cycles of a clock prescaled by 1024
    TCCR1A = 0;                          // set timer1 to normal operation (all
    bits in control registers A and B set to zero)
    TCCR1B = 0;                          //
    TCNT1 = 0;                           //  initialize counter value to 0
    OCR1A = target;                      // assign target count to compare
    register A (must be less than 65536)
    OCR1B = OCR1A * duty_cycle;
    TCCR1B |= (1 << WGM12);              // clear the timer on compare match A
    TIMSK1 |= (1 << OCIE1A);             // set interrupt on compare match A
    TIMSK1 |= (1 << OCIE1B);             // set interrupt on compare match B
    TCCR1B |= (1 << CS12) | (1 << CS10); // set prescaler to 1024 and start the
    timer
    sei();                               // enable interrupts
}

void Timer_msec::set(float duty_cycle){
    OCR1B = OCR1A * duty_cycle;
}
```

Listing 2: timer_msec.cpp

```
#include <digital_out.h>
#include <timer_msec.h>
#include <avr/interrupt.h>

#include <avr/delay.h>

Digital_out led(5);
Timer_msec timer;


int main()
{
  led.init();

  // part 1:
  // timer.init();
  // timer.init(500);

  // part 2:
  timer.init(20, 0.20);

  sei(); // enable interrupts

  // do nothing forever
  while (1){}

  // for part 3 comment out the above while loop

  // loop through 0.0 - 1.0
  int duty_cycle = 0;
  while (true)
  {
    duty_cycle += 5; // increase by five  percent per loop
    if (duty_cycle > 100){
      duty_cycle = 0;
    }
    // change duty cycle
    timer.set(duty_cycle/100.);
    // delay 250 means full range takes 20*250 = 5000 milliseconds.
```

```
    _delay_ms(200);

  }
}

ISR(TIMER1_COMPA_vect)
{
  // action on compare match A

  // for part 1 use:
  // led.toggle();

  // for part 2 and 3 it should be
  led.set_hi();
}

ISR(TIMER1_COMPB_vect)
{
  // action on compare match B
  led.set_lo();
}
```

Listing 3: main.cpp