Reykjavík University

Embedded System Programming

T-738-EMBE

Assignment 1.2: Timer

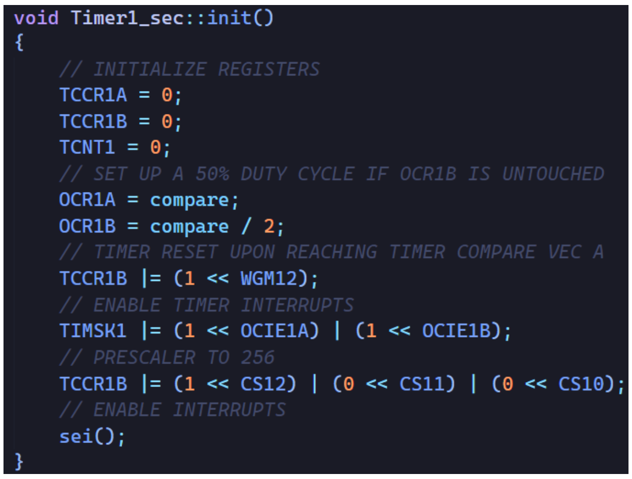
31. August 2021

Student:
Sindri Þór Jónsson

Instructor:
Torfi Þórhallsson

1 Part

Create an application to blink an LED at 1000 ms intervals using a simple timer driver that hides the timer control registers. To control the LED, you can re-use the Digital_out driver from assignment A1.2. Add an argument to the init function to set the timer interval to a different value (in milliseconds).
A timer driver application to blink an LED at a 1000 millisecond interval, the driver from Assignment 1.2 is used to write out to the inbuilt LED on the micro-controller. A initializing function "init()" sets the prescaler and compare register for the LED blinking interrupt:



```
void Timer1_sec::init()
{
    // INITIALIZE REGISTERS
    TCCR1A = 0;
    TCCR1B = 0;
    TCNT1 = 0;
    // SET UP A 50% DUTY CYCLE IF OCR1B IS UNTOUCHED
    OCR1A = compare;
    OCR1B = compare / 2;
    // TIMER RESET UPON REACHING TIMER COMPARE VEC A
    TCCR1B |= (1 << WGM12);
    // ENABLE TIMER INTERRUPTS
    TIMSK1 |= (1 << OCIE1A) | (1 << OCIE1B);
    // PRESCALER TO 256
    TCCR1B |= (1 << CS12) | (0 << CS11) | (0 << CS10);
    // ENABLE INTERRUPTS
    sei();
}
```

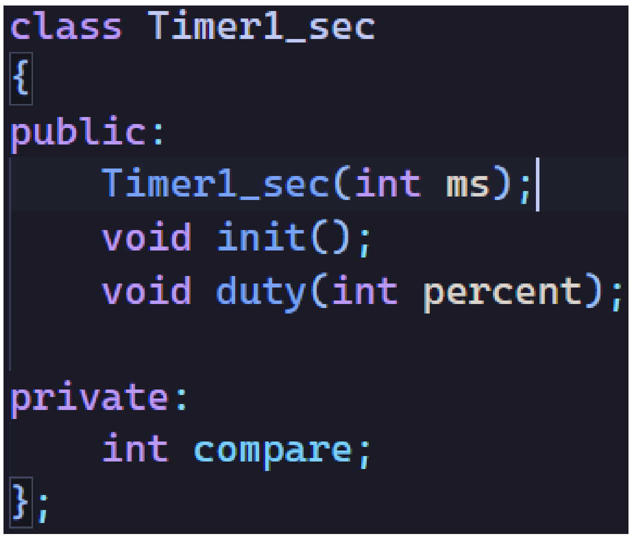
In this implementation the prescaler is set to 1024 and the compare variable is the timing for the interrupt vector. The class constructor takes in the amount of milliseconds (int) and toggles the LED when the selected amount is reached (1000ms).
What is the maximum length of the timer interval that the driver supports in this implementation?
Since the Timer1 register is a 16bit the maximum length of the timer interval of the driver in this implementation is found by:

$$ms = (2^{16}/(\frac{16000}{1024})) - 1 = 4193ms$$

This means that the maximum length timer of this driver is 4193ms.

2 Part

Modify the timer class to allow the application to blink with a duty cycle other than 50%.
The function called "duty" is added to the class which takes in a integer representing the duty cycle percentage of the LED. If this function does not get called the default duty cycle of 50% will be used, which i think is a neater approach.



```
class Timer1_sec
{
public:
    Timer1_sec(int ms);
    void init();
    void duty(int percent);

private:
    int compare;
};
```

The function sets the ratio of the two compare register to match the input duty cycle. Interrupt routines are then called on each compare register. This class member function allows the duty cycle to be changed while the timer is running.

With the duty cycle set to 20%, reduce the timer period until unit you no longer notice the light blinking. At what timer frequency does the LED appear to change from blinking to a stable intensity?
With a duty cycle of 20% at around 50Hz i can no longer notice that the LED is blinking. The LED appears to be at a stable intensity.

How does the intensity of the LED vary when you change the duty cycle?
With the frequency where the blinking isn't noticeable anymore the duty cycle changes result in a brightness change in the LED.

Low duty cycle: The LED is dimly lit
High duty cycle: The LED shines brightly