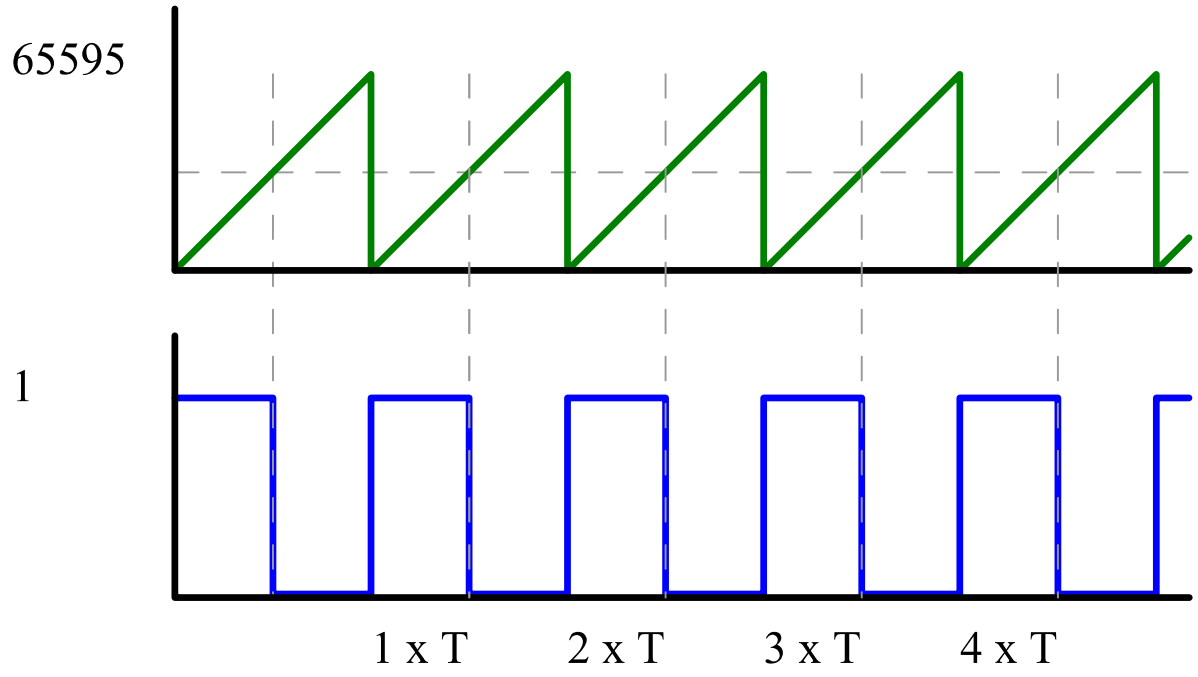
Широтно-импульсная модуляция

Мы уже говорили раньше, что таймеры могут работать в разных режимах. Один из них позволит нам управлять яркостью светодиода. Дабы разобраться, какой режим нам нужен, давайте порассуждаем.

Человеческий глаз — штука сложная. В нём есть палочки и колбочки, а яркость, с которой мы наблюдаем объекты, зависит от количества упавших на них фотонов. Особенность заключается в том, что «оцифровка» количества фотонов происходит не сразу. Они работают как сумматор, т. е. накапливают «заряд», и через какой-то промежуток времени показания снимаются. Это называют инертностью человеческого зрения. Другими словами, если объект будет мерцать быстрее, чем происходит снятие показаний — мы просто не заметим мерцания. Зачем нам это нужно? Всё просто! Время «регистрации» фиксировано (меняется от человека к человеку), а значит, на колбочку/палочку может упасть фиксированное (если мы берем конкретный источник света) количество фотонов. Если половину этого времени светодиод будет гореть, а вторую половину нет — то на колбочку/палочку попадет в два раза меньше фотонов, чем в том случае, если светодиод будет гореть постоянно.

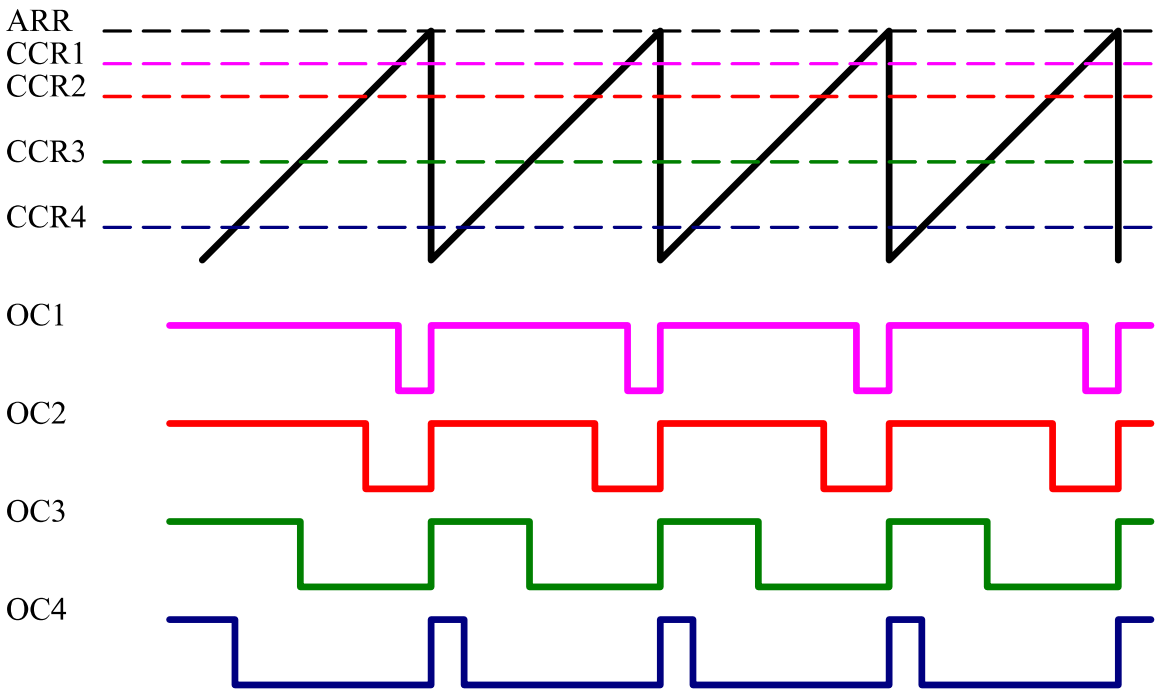
Примерная частота, при которой среднестатистический человек не заметит мерцания — 50 Гц [1].

Такой принцип используется не только для регулировки яркости светодиодов, но и при управлении некоторыми электродвигателями. Модуляция возможна разная, но самая популярная из них — это ШИМ (широтно-импульсная модуляция, с англ. PWM — Pulse Width Modulation). Частота (период) фиксирована, а вот заполнение (англ. duty) может меняться.



В реальности наши органы чувств имеют логарифмическую характеристику чувствительности, будь то слух или зрение. Это позволяет им работать в огромном диапазоне интенсивностей (закон Фехнера — Вебера). В связи с этим, линейно меняя заполнение, нам не удастся «плавно» менять яркость. Впрочем, мы пренебрежем этим знанием и реализуем зажигание и угасание светодиода по линейному закону (оставив возможность реализации «того, как правильно» на вашу совесть).

В разделе «Таймеры. Обзор» мы рассматривали возможности таймеров, и как не сложно догадаться, нам потребуется режим генерации ШИМ. Согласно схеме светодиод подключен к ножке PA3. Её следует настроить как альтернативный выход с подтяжкой. Настройка таймера в начале не многим отличается от того, что мы уже делали с `TIM4`, нам просто не нужно использовать прерывание. У таймера `TIM2` имеется четыре канала, которые могут работать независимо друг от друга.



Частота для каждого канала будет одна и та же, а вот заполнение можно сделать разным. Более того, генерацию можно сделать инвертированной.

К ножке PA3 подведён 4 канал, соответственно его-то и нужно использовать. Выбрать его для генерации ШИМ нужно обратившись к регистру `CCER`, установив туда `1` в бит `CC4E`. Второй бит из этого регистра, `CC4P`, отвечает за активный уровень — запись `0` делает активным уровнем высокий (т.е. до того как счётчик досчитает до значения в `ARR`), а `1` низкий.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

Все каналы могут работать как на вход (англ. capture), так и на выход (англ. compare). Так как мы собираемся генерировать ШИМ, нужно настроить нужный канал через регистры `CCMR1` (1 и 2 канал) и `CCMR2` (3 и 4 канал).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Записав `00` в `CC4S` канал настраиваемся на выход.

Записав в `OC4M` последовательность `110` активируется прямая модуляция (PWM mode 1), для получения инвертированной (PWM mode 2) придётся записать `111`.

```
#define PERIOD(f)       (1e6 / f)
```

```
#define PRESCALER       (SystemCoreClock / 1e6)
#define FREQ            1000

void led_pwm_init(void) {
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
    RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;

    // PA3 as output, alternative, push-pull
    GPIOA->CRL &= ~GPIO_CRL_CNF3;
    GPIOA->CRL |= GPIO_CRL_CNF3_1;
    GPIOA->CRL &= ~GPIO_CRL_MODE3;
    GPIOA->CRL |= GPIO_CRL_MODE3_1;

    // TIM2, timer 2, channel 4
    TIM2->CR1 &= ~TIM_CR1_DIR;          // upcounting
    TIM2->CR1 &= ~TIM_CR1_CMS;          // edge-align mode
    TIM2->PSC = PRESCALER;              // 64
    TIM2->ARR = PERIOD(FREQ) - 1;       // 1 kHz
    TIM2->CCR4 = PERIOD(FREQ) / 2 - 1; // 50% by default
    TIM2->CCER |= TIM_CCER_CC4E | TIM_CCER_CC4P; // 4 channel, low level
    TIM2->BDTR |= TIM_BDTR_MOE;         // use as output
    TIM2->CCMR2 &= ~TIM_CCMR2_CC4S;     // output
    TIM2->CCMR2 = TIM_CCMR2_OC4M_2 | TIM_CCMR2_OC4M_1; // PWM1 mode
    TIM2->CR1 |= TIM_CR1_CEN;           // enable timer
}
```

Напишем функцию изменения яркости светодиода. Пусть для простоты будет 256 (`uint8_t`) градаций яркости.

```
void led_dim(const uint8_t brightness) {
    TIM2->CCR4 = brightness * (PERIOD(FREQ) - 1) / 255;
}
```

Протестируем функцию, и добавим её вызов в функцию `main()` . Светодиод долен плавно загораться и тухнуть.

```
int main(void) {
    led_pwm_init();

    while(1) {
        for (uint32_t i = 0; i < 256; i++) {
            led_dim(i);
            delay(1000);
        }
        for (uint32_t i = 254; i > 0; i--) {
            led_dim(i);
            delay(1000);
        }
    }
}
```

Код урока можно найти на github: **CMSIS**.

Назад | Оглавление | Дальше

1. Согласно санитарным нормам, частота источников света должна быть больше 300 Гц. ↩