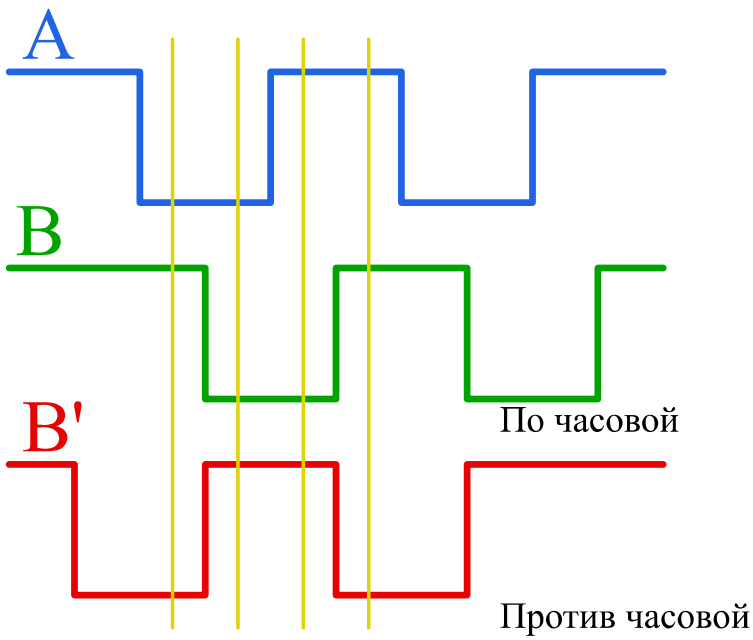
Работа с энкодером. Простое использование

Датчик угла поворота (ДУП, или англ. *encoder*) — это электронно-механическое устройство, преобразующее поворот в цифровой или аналоговый сигнал. Такие устройства могут быть основаны на разных физических принципах, но концепция у них одна. В целом их можно поделить на две большие группы: абсолютные (значение которых известно относительно «нуля») и инкрементальные (или накопительные, т. е. их положение относительно, и можно говорить только о направлении и скорости вращения). В нашем устройстве присутствует датчик угла поворота второго типа. Некоторые из ДУП могут иметь встроенную в вал кнопку (как в нашем устройстве), механика которой ничем не отличается от обычной.



A	B	B'
0	1	0
0	0	1
1	0	1
1	1	0

В начальном состоянии выводы подтянуты к высокому состоянию при помощи резистора. Когда вал начинают вращать по направлению часовой стрелки, сигнал **A** переходит в низкое состояние, а **B** делает это после небольшой задержки. Если крутить вал в противоположную стрелке сторону, то ситуация изменится на обратную – запаздывать будет сигнал **A**. Временные интервалы зависят от скорости вращения.

> Как и любая механическая система, датчики угла поворота страдают так называемым «дребезгом» контактов. Однако внутри самого микроконтроллера уже имеются резисторы подтяжки порядка МОм, поэтому впаивать резисторы нет особой надобности (можно сэкономить), плюс сам МК позволяет отфильтровать дребезг.

В микроконтроллерах stm32 некоторые из таймеров имеют особый режим работы (англ. encoder mode), позволяющий реализовать работу с энкодером аппаратно.

Таймер позволяет отслеживать либо фронты **A** (`TI1`), либо фронты **B** (`TI2`), либо оба сразу. В зависимости от выбранного режима изменение счётчика будет меняться либо на 2, либо на 4 (по количеству фронтов, передних и задних).

Режима работы с энкодером описан в документе AN4013, а так же в Reference Manual, пункт «**15.3.12 Encoder interface mode**».

Приступим к реализации и для начала создадим прототипы функций, которые понадобятся для работы.

```
// encoder.h
void encoder_init(void);
uint16_t encoder_get_value(void);
```

В целом логика настройки всегда остаётся примерно одной и той же: включаем тактирование порта и периферии; настраиваем ножки; настраиваем периферию; включаем периферию. Последовательность лучше не нарушать, в противном случае периферия скорее всего работать не будет.

```
#define MAX_COUNT_VALUE             128

void encoder_init(void) {
    // Включаем тактирование порта A и TIM3
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
    RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;

    // Настраиваем ножки PA6 и PA7 на вход, альтернативная функция, подтяжка к питанию
    // PA6
    GPIOA->CRL &= ~GPIO_CRL_MODE6;
    GPIOA->CRL &= ~GPIO_CRL_CNF6_0;
    GPIOA->CRL |= GPIO_CRL_CNF6_1;
    GPIOA->ODR |= GPIO_ODR_ODR6; // pull-up
    // PA7
    GPIOA->CRL &= ~GPIO_CRL_MODE7;
    GPIOA->CRL &= ~GPIO_CRL_CNF7_0;
    GPIOA->CRL |= GPIO_CRL_CNF7_1;
    GPIOA->ODR |= GPIO_ODR_ODR7;

    // Настраиваем таймер TIM3
    TIM3->CR1 &= ~TIM_CR1_CMS;    // сброс выравнивания, edge-aligned mode
    TIM3->CR1 &= ~TIM_CR1_DIR;    // счёт вверх
    TIM3->ARR = MAX_COUNT_VALUE; // устанавливаем период
    TIM3->PSC = 0;                // нет предделителя
    // Включаем режим энкореда, отслеживание только фронта TI1
    TIM3->SMCR &= ~TIM_SMCR_SMS;
    TIM3->SMCR |= TIM_SMCR_SMS_0;
    TIM3->CCMR1 &= ~TIM_CCMR1_CC1S; // очистка битов CC1S
    TIM3->CCMR1 &= ~TIM_CCMR1_CC2S; // очистка битов CC2S
    TIM3->CCMR1 |= TIM_CCMR1_CC1S_0 | TIM_CCMR1_CC2S_0;
    // устанавливаем полярность
    TIM3->CCER &= ~TIM_CCER_CC1P; // CP1 возростание
    TIM3->CCER &= ~TIM_CCER_CC2P; // CP2 возростание
    // включаем таймер
    TIM3->CR1 |= TIM_CR1_CEN;
}
```

Таймер настроен на режим работы с энкодером, теперь при при повороте состояние регистра `CNT` будет меняться на `2` (так как мы отслеживаем только `TI1`) от `0` до `128`. Допишем функцию возврата значения счётчика:

```
uint16_t encoder_get_value(void) {
    return TIM3->CNT;
}
```

Для проверки работы можно запустить следующий код из главного цикла:

```
led_toggle();
delay(encoder_get_value() * 10000);
```

Светодиод будет мигать с разной задержкой, в зависимости от содержания регистра `CNT`. Код можно найти на github: **CMSIS**.

Перед переходом к следующему разделу подумайте, как и где вы будите использовать данные функции. Должен ли энкодер менять одну переменную или придётся менять несколько? Чем такая реализация плоха?

Назад | **Оглавление** | **Дальше**