Долгое удержание кнопки

Действие по простому нажатию кнопки реализуется довольно просто — добавляем сглаживающую фронт RC-цепочку (антидребезг), настраиваем прерывание и кодируем обработчик прерывания `EXTI9_5_IRQHandler`. Но что делать, если мы хотим повесить несколько действий на одну и ту же кнопку?

Возьмите свой телефон и найдите там кнопку выключения экрана. В большинстве случаев она реализует несколько действий — длительное удержание выключает телефон. В нашем случае с помощью удержания кнопки можно перейти, например, к настройки времени, а простое одиночное нажатие будет отображать либо время, либо температуру.

Решений может быть несколько. Самый тупой способ — это внутри прерывания замерять время зажатия кнопки.

```
// delay for 3 seconds
#define BUTTON_LONG_PRESS_DELAY       3000
// ...
void EXTI9_5_IRQHandler(void) {
    button_timer_start(); // reset counter and start the timer
    while((GPIOA->IDR & GPIO_IDR_IDR5) != 0);
    button_timer_stop();  // stop the timer
    if (button_timer_get_value() < BUTTON_LONG_PRESS_DELAY) {
        // short press action here
    } else {
        // long press action here
    }
    // clean pending but
}
```

Это очень плохое решение. Здесь сразу же нарушается первое правило работы с прерываниями: код должен выполняться как можно быстрее. Пока нажата кнопка всё устройство будет парализовано. Если скажем другой таймер отвечает за обновление показаний на дисплее (в нашем случае это не так), то прерывание от него не произойдёт до тех пор, пока вы не отпустите кнопку. Это не желательное поведение. Ситуация, однако, может быть более критической.

Разбираясь с модулем `EXTI` мы показали, что прерывание может происходить либо по переднему, либо по заднему, либо по обоим фронтам сразу. Т.е. зная что прерывание произошло по переднему фронту, можно совершить одно действие, а по заднему другое. К сожалению, напрямую проверить по какому фронту произошло прерывание нельзя — нет специального регистра, который бы сигнализировал об этом. Но можно использовать знание того, как работает схема.

RC-цепочка надёжно справляется с дребезгом контактов. Это значит, что при нажатии на линии гарантировано высокий уровень, т.е. считав входной регистр и обнаружив там `1` мы с уверенность можем сказать, что прерывание произошло по переднему фронту. Соответственно, обнаружив там `0` поймём что это был задний фронт.

Реализуем это программно.

```
void EXTI9_5_IRQHandler(void) {
    if ((GPIOA->IDR & GPIO_IDR_IDR5) == GPIO_IDR_IDR5) { // rising edge
        button_timer_start();
    } else { // falling egde
        button_timer_stop();
        if (button_timer_get_value() < BUTTON_LONG_PRESS_DELAY) {
            // short press action here
        } else {
            // long press action here
        }
    }
    // Clean pending bit
}
```

На забудьте разрешить прерывание по заднему фронту в настройка EXTI.

Осталось лишь подумать, как реализовать таймер. Попробуйте реализовать данную функцию самостоятельно, с помощью `TIM1`.

Подсказка

Рассчитайте предделитель таймера так, что бы увеличение счётчика происходило раз в 1 мс. Всю инициализацию поместите в `button_timer_start()`. Функция `button_timer_start()` должна сбросить состояние счётчика в ноль, а затем запустить таймер. Функция `button_timer_stop()` просто останавливает таймер. Последние две функции полезно сделать в виде макроса, либо указать им модификатор `inline`.

Если у вас не получилось настроить таймер самостоятельно, то воспользуйтесь примерами кода от компании ST и проанализируйте что вы сделали не так. Примеры лежат в архиве с библиотекой, который можно скачать на странице микроконтроллера **stm32f103c8**. Если даже не помог пример из библиотеки, реализацию можно скопировать из репозитория курса на github.com: **CMSIS**.

Назад | **Оглавление** | **Дальше**