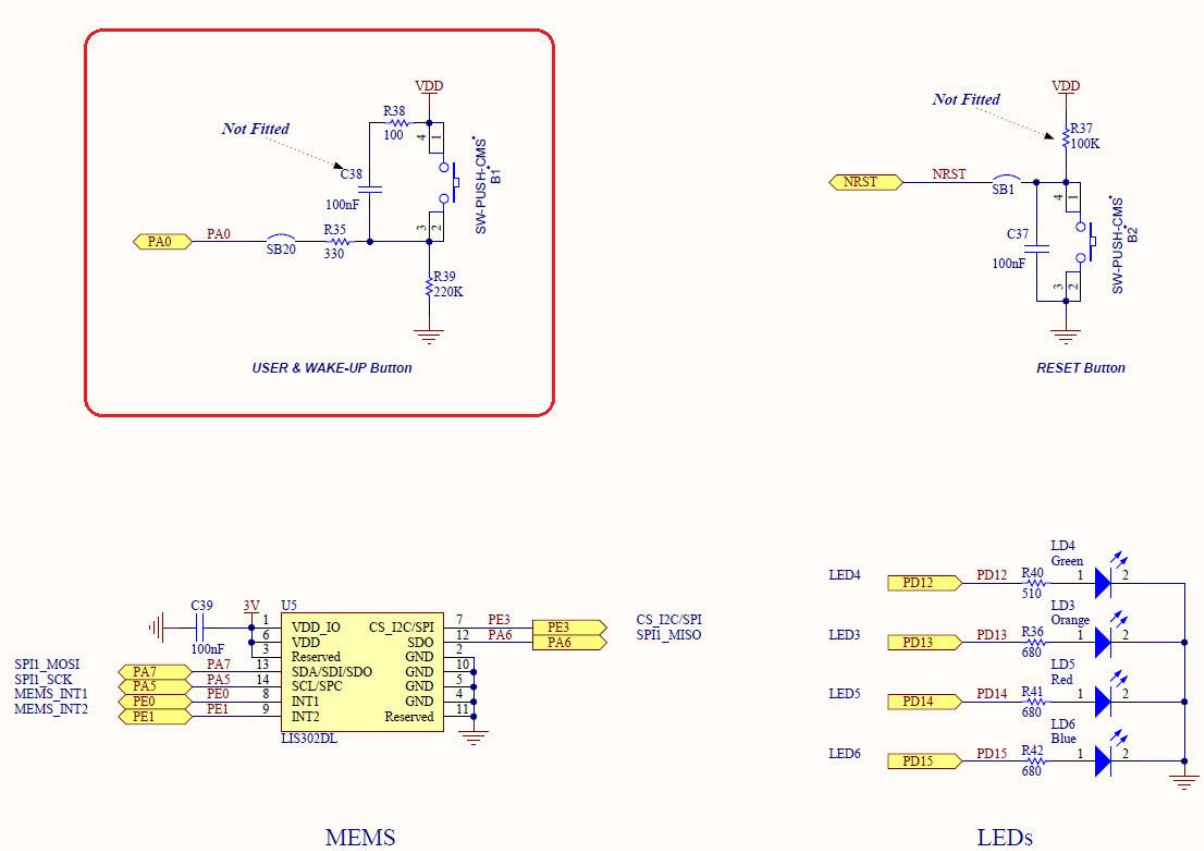
GL Starter Kit
LED’s and Button

Программируемая кнопка

Сначала определим, к какому выводу на плате подключена кнопка. Для этого открываем схему платы, где показано, к каким выводам подключены кнопки и светодиоды на плате.



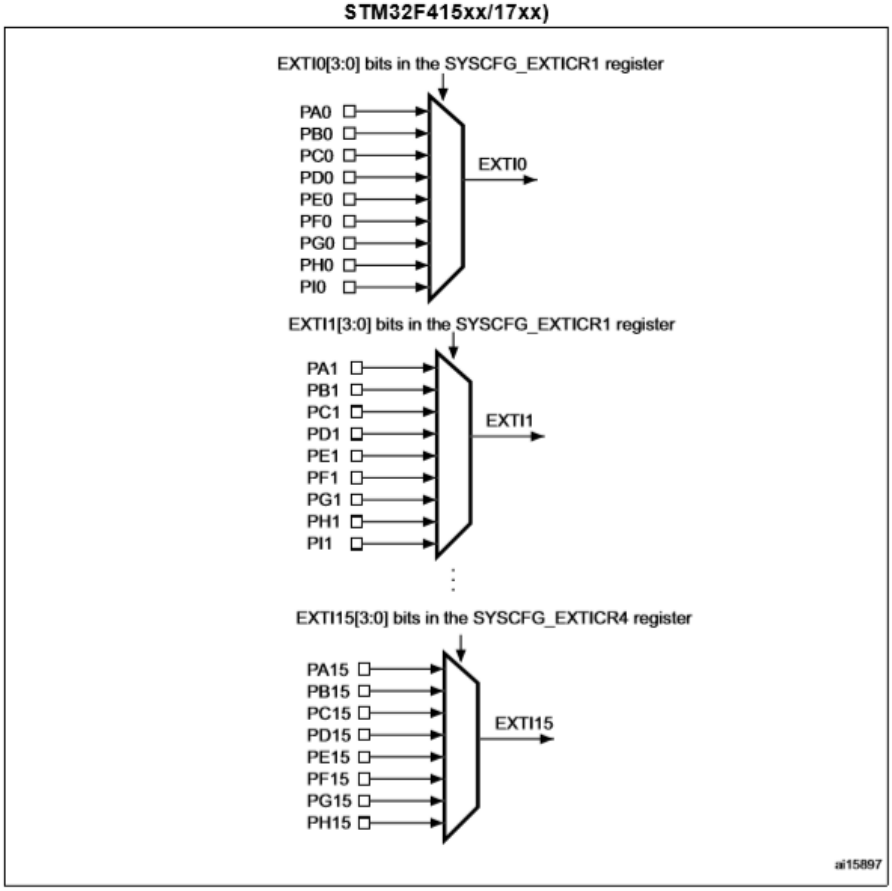
Из схемы видно, что пользовательская кнопка подсоединена к выводу "PA0" и подтянута к "GND" внешним резистором. При нажатии на кнопку на выводе будет появляться логическая единица.

Внешние прерывания

Внешние прерывания — прерывания, происходящие при сигналах внешних устройств. Именно внешние прерывания позволяют оперативно обрабатывать сигналы внешних устройств, например, реагировать на нажатие кнопки, сигналы датчиков, и так далее.

Контроллеры STM32, имеют в своем составе контроллер внешних прерываний EXTI. Всего у микроконтроллеров серии STM32F4 внешние прерывания EXTI0-EXTI15 могут быть настроены на работу от изменения уровня на соответствующей ножке контроллера. В контроллерах STM32 мы вольны сами выбрать ножки, на которые будут настроены прерывания EXTI. Рассмотрим следующее:

Figure 42. External interrupt/event GPIO mapping (STM32F405xx/07xx and STM32F415xx/17xx)



Здесь мы видим 16 линий, подключенных через мультиплексоры к одноименным пинам всех портов. То есть одновременно мы можем обработать 16 ножек контроллера, но, как видно из мультиплексной организации, все они должны быть с разными номерами. То есть мы можем обработать одновременно ножки PA1 и PC2, но не можем обработать PA1 и PC1.

Для прерывания EXTI0 мы можем выбрать нулевой бит порта (PA0, PB0, PC0 и так далее), для EXTI1 первый бит, и так далее. Остальные прерывания EXTI16-EXTI22, подключены в следующем порядке:

The seven other EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is connected to the USB OTG FS Wakeup event
- EXTI line 19 is connected to the Ethernet Wakeup event
- EXTI line 20 is connected to the USB OTG HS (configured in FS) Wakeup event
- EXTI line 21 is connected to the RTC Tamper and TimeStamp events
- EXTI line 22 is connected to the RTC Wakeup event

То есть мы можем ещё обработать внешние события от программируемого детектора напряжений, от будильника RTC, от "пробуждений" USB и Ethernet и т.д.

Обработчики прерываний

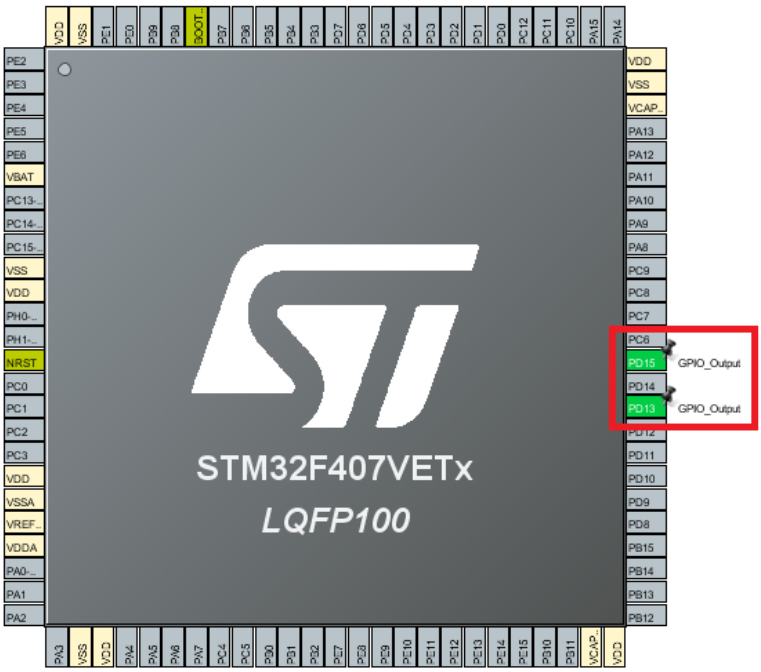
STM32F4 имеет 7 обработчиков прерываний для выводов GPIO. Они находятся в таблице ниже:

Irq	Handler	Description
EXTI0_IRQn	EXTI0_IRQHandler	Handler for pins connected to line 0
EXTI1_IRQn	EXTI1_IRQHandler	Handler for pins connected to line 1
EXTI2_IRQn	EXTI2_IRQHandler	Handler for pins connected to line 2
EXTI3_IRQn	EXTI3_IRQHandler	Handler for pins connected to line 3
EXTI4_IRQn	EXTI4_IRQHandler	Handler for pins connected to line 4
EXTI9_5_IRQn	EXTI9_5_IRQHandler	Handler for pins connected to line 5 to 9
EXTI15_10_IRQn	EXTI15_10_IRQHandler	Handler for pins connected to line 10 to 15

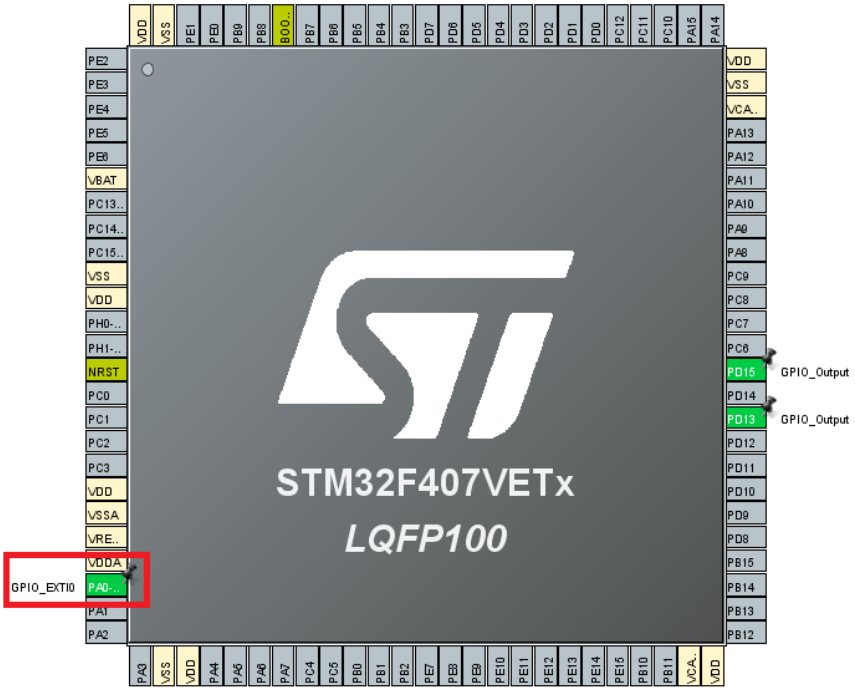
В этой таблице показано, какой IRQ необходимо установить для NVIC [Nested Vector Interrupt Controller] (первый столбец) и имена функций для обработки прерываний (второй столбец). Также можно заметить, что только строки от 0 по 4 имеют собственный обработчик IRQ. Строки 5-9 имеют одинаковый обработчик прерываний, это справедливо и для 10-15.

Выполнение тестовой задачи

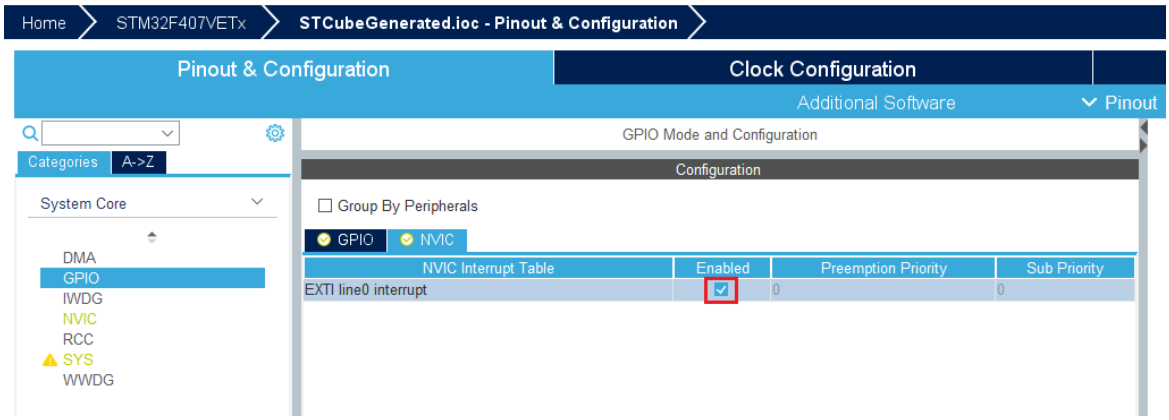
По аналогии с лабораторной работой #1 создаем для нашей платы новый проект.
В CubeMX настраиваем интересующие нас светодиоды: синий (PD15) и оранжевый (PD13) устанавливаем в GPIO_Output.



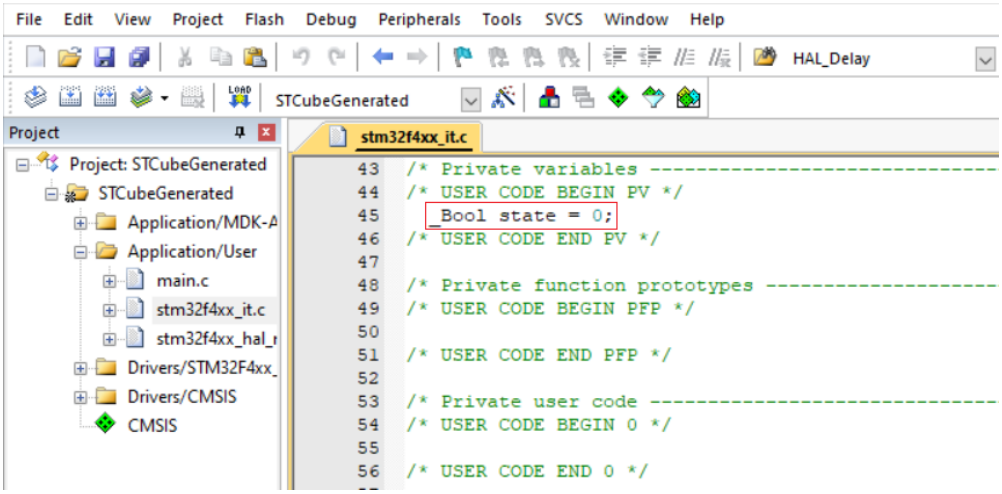
Далее установим PA0-WKUP в GPIO_EXTI0.



В таблице настроек контроллера прерываний NVIC, включим прерывания EXTI0.

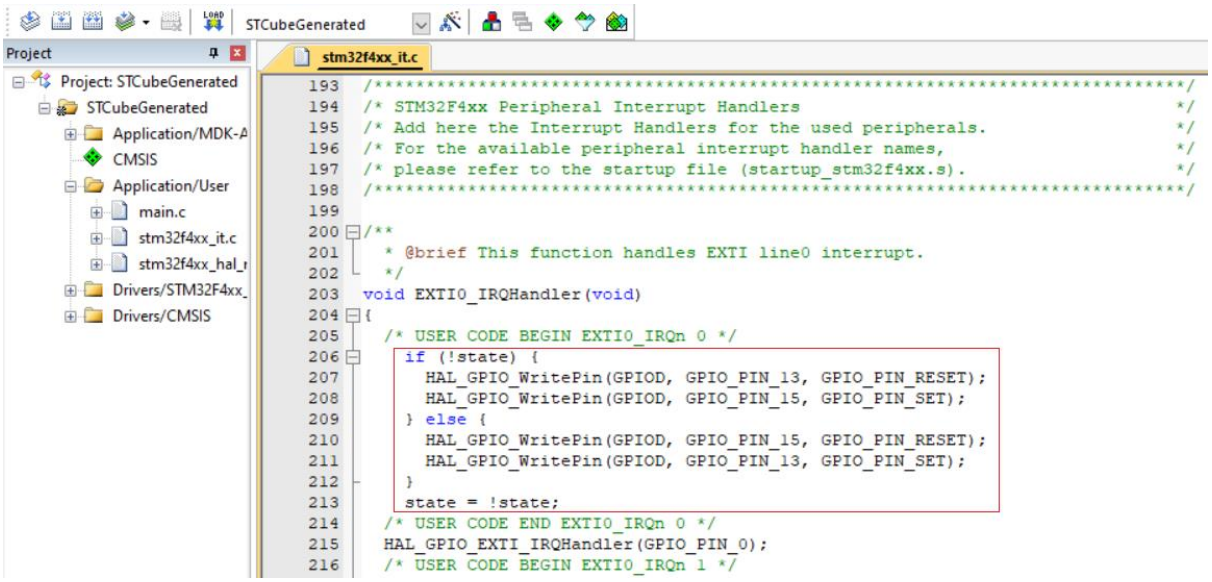


Генерируем наш код и открываем новый проект.
Находим файл *stm32f4xx_it.c*
Для сохранения информации о нажатии кнопки введем переменную *_Bool state = 0;*



Теперь в обработчик прерывания *void EXTI0_IRQHandler(void)* добавим код для изменения состояния светодиодов при нажатии на кнопку.

```
if (!state) {
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);
} else {
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
}
state = !state;
```



Собираем наш проект и прошиваем плату. На выходе мы получаем прошивку, где последовательные нажатия на программируемую кнопку (синюю) зажигают то синий, то оранжевый светодиод.

Ремарка

Если вы будете использовать задержку:

HAL_Delay(500);

При попытке вызвать ее в void EXTI0_IRQHandler(void) вы можете обнаружить, что поведение программы не соответствует ожидаемому. Одним из решений будет изменение приоритета прерываний. Как вариант, можете добавить в мейн следующие строки:

NVIC_SetPriority(SysTick_IRQn, 1);
NVIC_SetPriority(EXTI0_IRQn, 2);

Соответствие PD к цвету светодиода, как вариант, можете посмотреть на схеме в начале документа, или в предыдущей лабе.

Варианты заданий

Модифицируйте проект согласно с вашим вариантом. Вариант определяется как остаток от деления номера в списке группы на 5 (n%5).

0. Нажатие на кнопку зажигает или гасит сразу все светодиоды.
1. Нажатие на кнопку подает сигнал SOS «три точки — три тире — три точки» синим светодиодом.
2. Нажатие кнопки зажигает по одному светодиоду по часовой стрелке начиная с зеленого светодиода (PD в порядке возрастания). Если горят все светодиоды, нажатие кнопки начинает процесс с начала.
3. Нажатие на кнопку попарно зажигает противоположные светодиоды.
4. Нажатие кнопки поочередно зажигает светодиоды против часовой стрелки начиная с синего (PD в порядке убывания). Одновременно должен гореть только один светодиод.