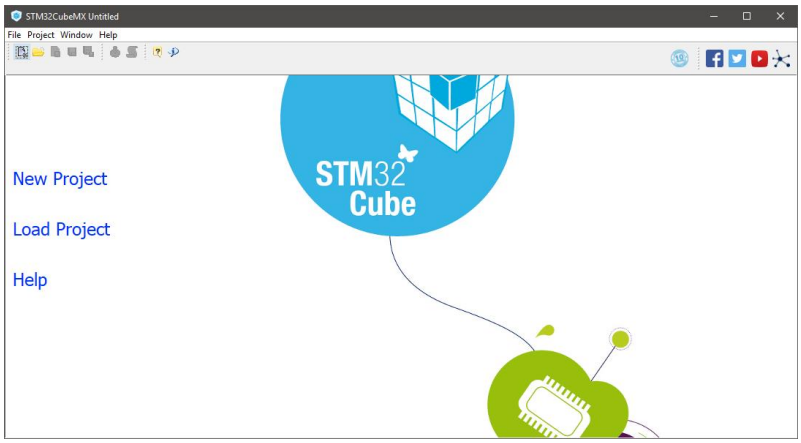
Создание проекта в CubeMX для STM32F401 (Nucleo)

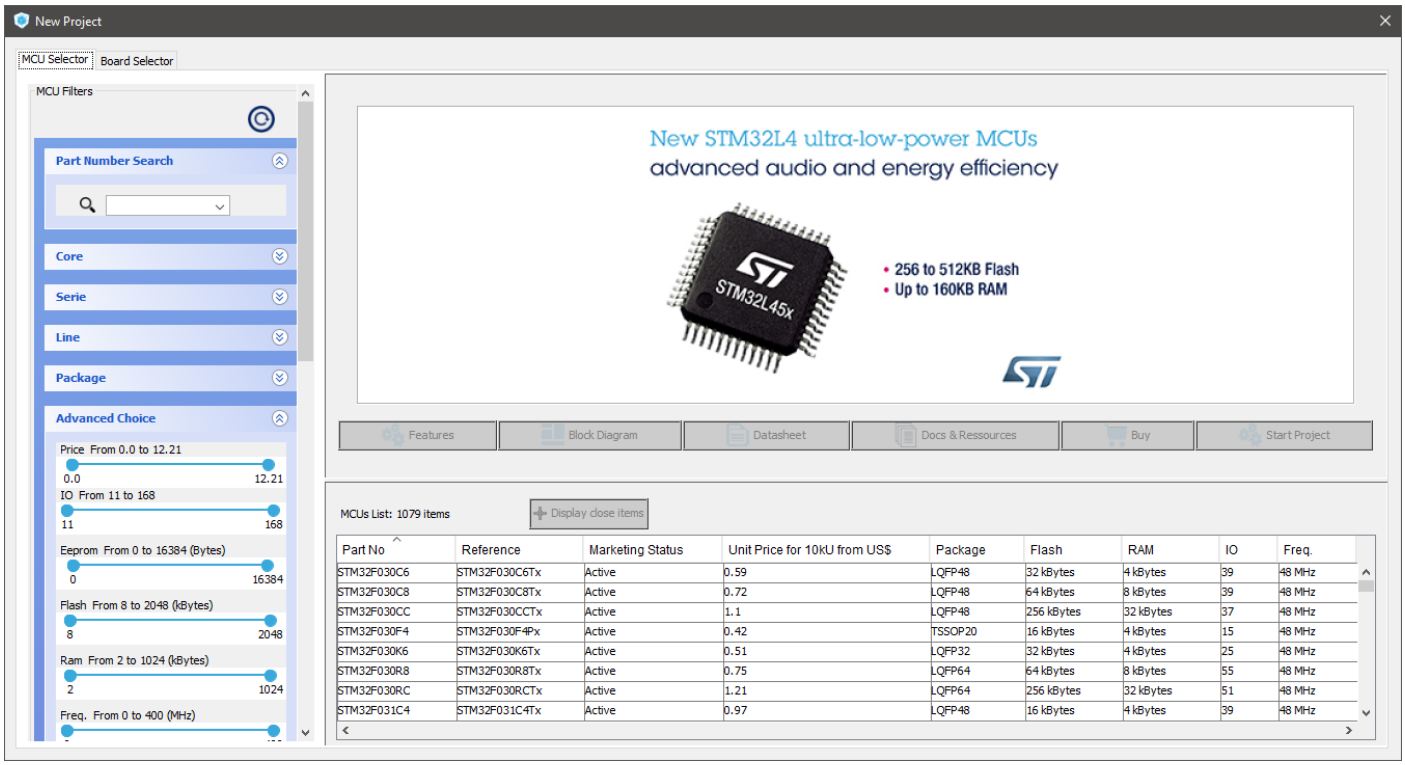
Пример настройки и использования GPIO + EXTI

Примечание: перед началом работы скачайте и установите утилиту CubeMX с сайта ST: https://goo.gl/y69SeR. Предполагается, что у вас установлена и работает IDE Keil v5 для STM32.

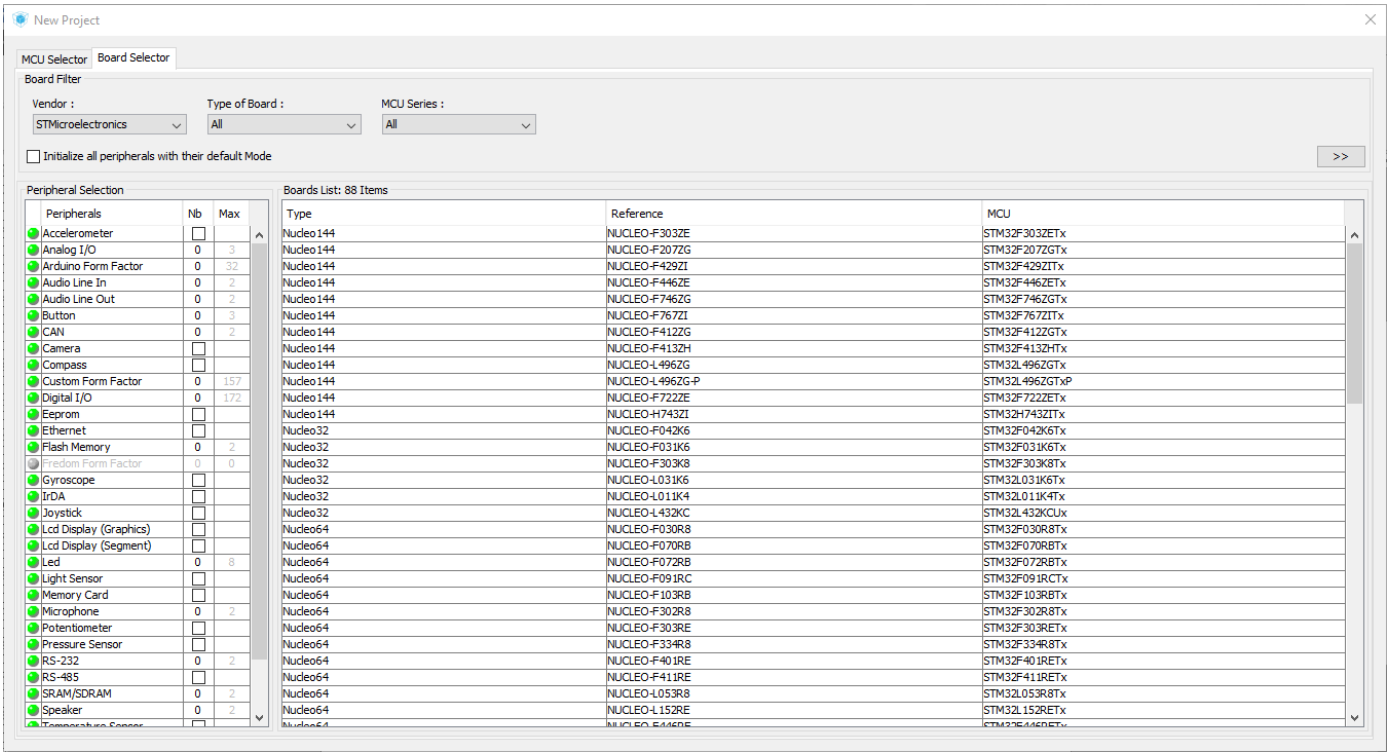
1. Запустите утилиту нажатием на ярлык или пункт меню Пуск.
2. После загрузки программы выберите “New Project”:
3. На следующем этапе вы должны выбрать конкретную модель контроллера, с которой собираетесь работать. Модель указана на корпусе микросхемы.



Для удобства пользователя имеется возможность выбора демонстрационной платы. В этом варианте Cube самостоятельно создаст код для инициализации всей периферии на плате, выберет правильные настройки тактирования и т.д. Воспользуемся этим вариантом. Перейдите на вкладку “Board Selector”.

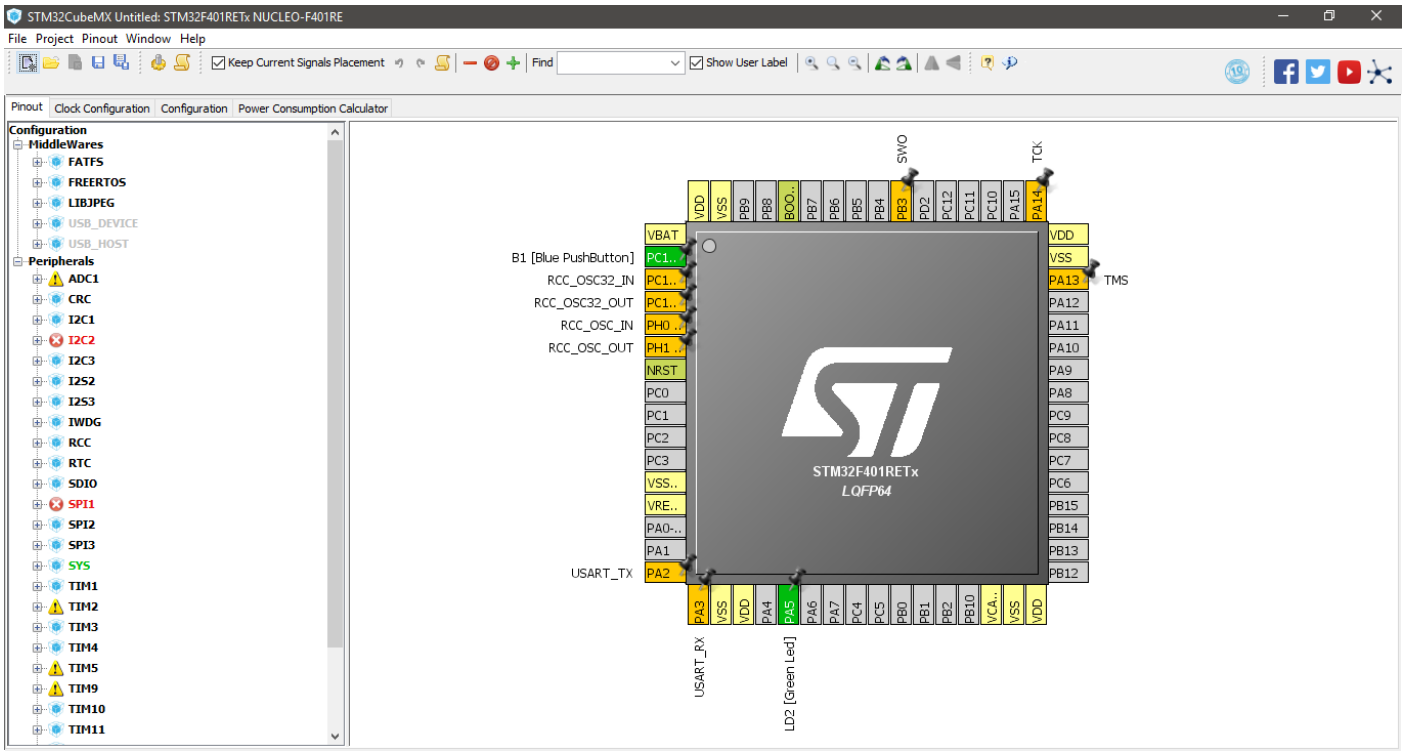


Part No	Reference	Marketing Status	Unit Price for 10kU from US$	Package	Flash	RAM	IO	Freq.
STM32F030C6	STM32F030C6Tx	Active	0.59	LQFP48	32 kBytes	4 kBytes	39	48 MHz
STM32F030C8	STM32F030C8Tx	Active	0.72	LQFP48	64 kBytes	8 kBytes	39	48 MHz
STM32F030CC	STM32F030CCTx	Active	1.1	LQFP48	256 kBytes	32 kBytes	37	48 MHz
STM32F030F4	STM32F030F4Px	Active	0.42	TSSOP20	16 kBytes	4 kBytes	15	48 MHz
STM32F030K6	STM32F030K6Tx	Active	0.51	LQFP32	32 kBytes	4 kBytes	25	48 MHz
STM32F030R8	STM32F030R8Tx	Active	0.75	LQFP64	64 kBytes	8 kBytes	55	48 MHz
STM32F030RC	STM32F030RCTx	Active	1.21	LQFP64	256 kBytes	32 kBytes	51	48 MHz
STM32F031C4	STM32F031C4Tx	Active	0.97	LQFP48	16 kBytes	4 kBytes	39	48 MHz

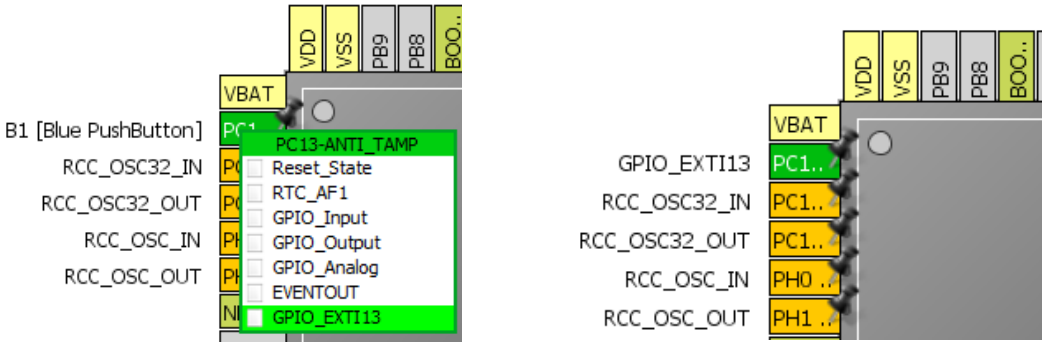


В списке плат выберите ту, которая была выдана преподавателем. Название платы указано на упаковке и на наклейке на самой плате. В качестве примера выберем NUCLEO-F401RE:

Откроется окно для выбора функционала контроллера. С помощью него можно настраивать, какие функции и выводы контроллера будут использованы в проекте.

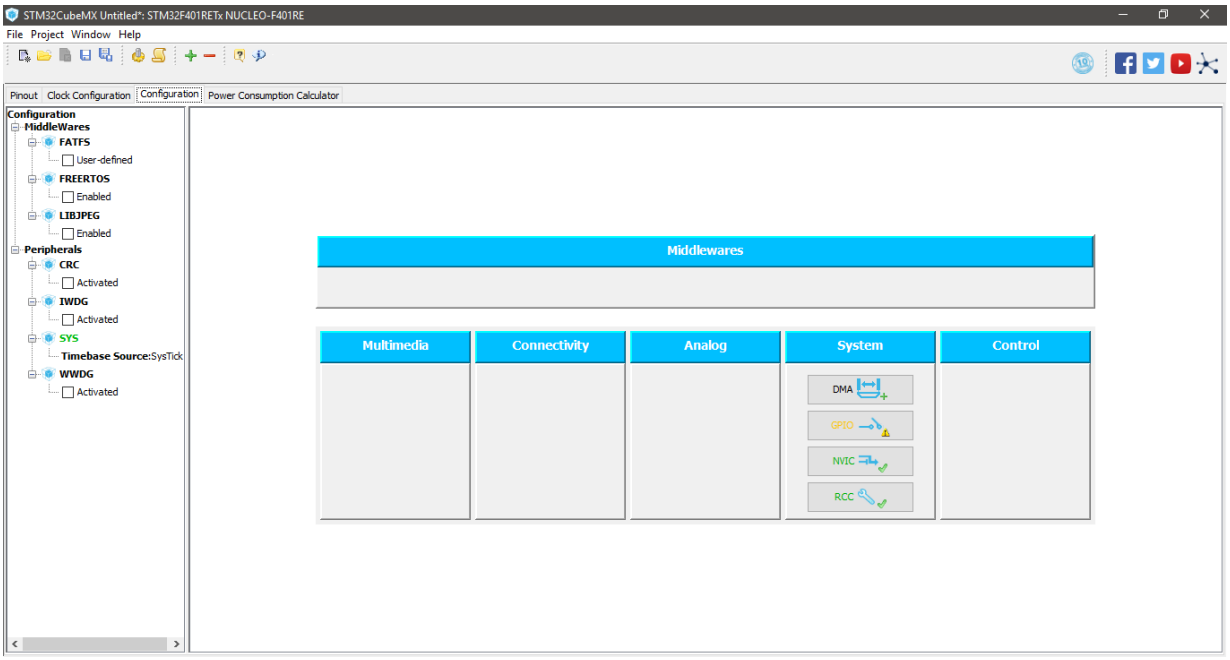


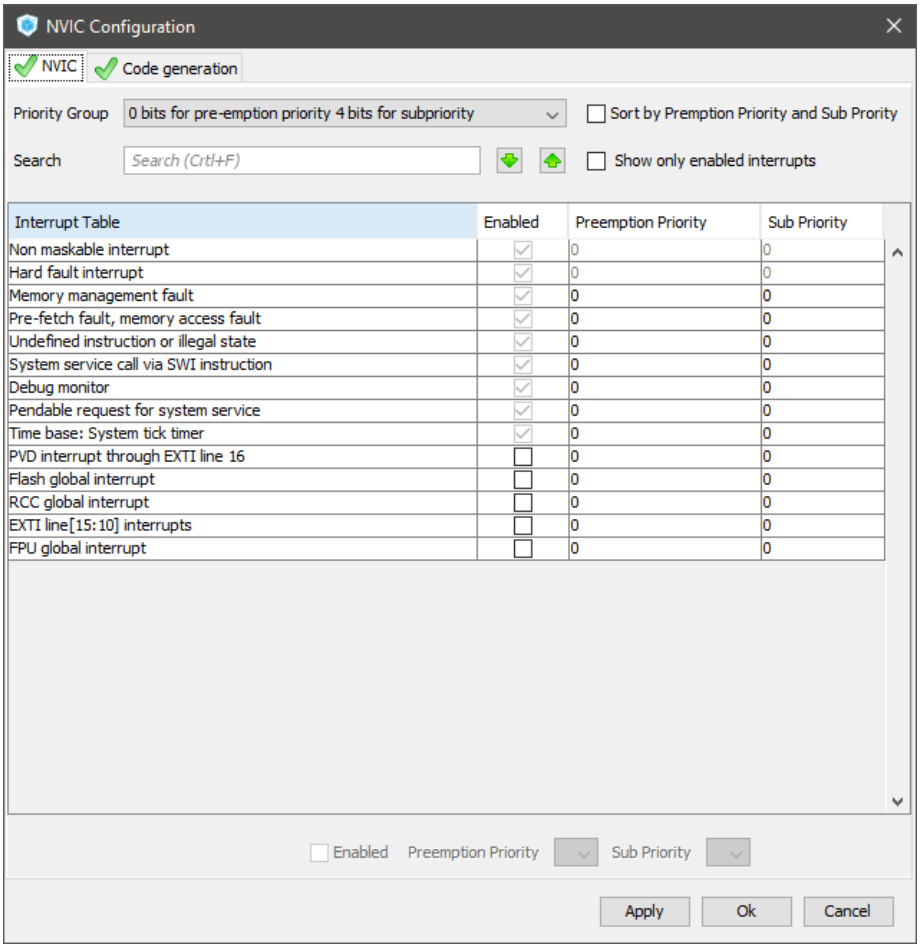
Некоторые выводы контроллера заняты, так как на выбранной плате к ним подключена периферия. Исходя из названия можно определить, что к PC13 подключена синяя кнопка на плате. Нажмем левой кнопкой на вывод и выберем режим работы – источник прерывания по линии прерываний 13. Увидим, что подпись вывода изменится.



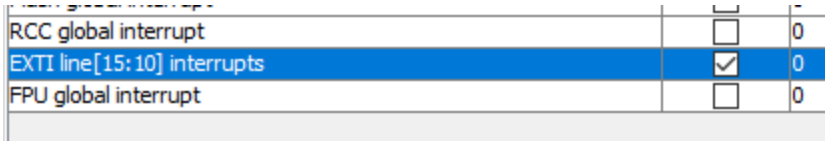
В верхней части интерфейса программы находятся вкладки “Pinout”, на которой мы сейчас находимся, “Clock configuration” для настройки тактирования, “Configuration” для подробных настоек функций, выбранных на вкладке Pinout и “Power Consumption Calculator” для расчета потребляемой мощности контроллера. Перейдем на вкладку “Configuration”.

Как видно, нам доступны настройки контроллера ПДП, портов ввода-вывода, контроллера прерываний и настройки питания и прочих системных параметров. Для включения прерывания по событию «нажатие кнопки» нажмем на кнопку настроек контроллера прерываний (NVIC).

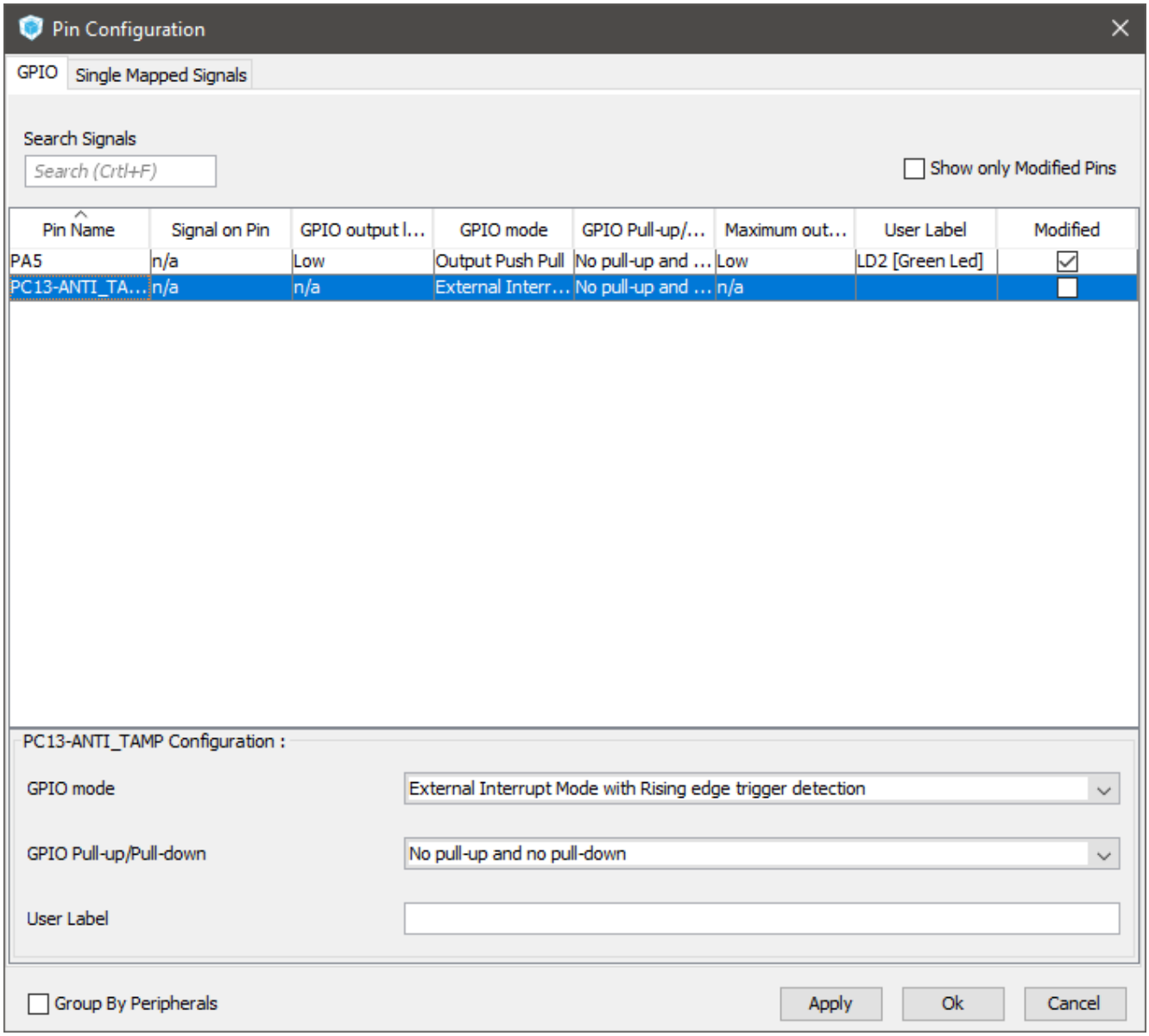




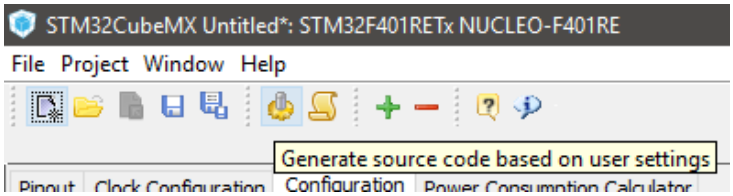
В данном окне можно включать/выключать различные прерывания, а также устанавливать их приоритет. Включим прерывание по внешнему событию EXTI:



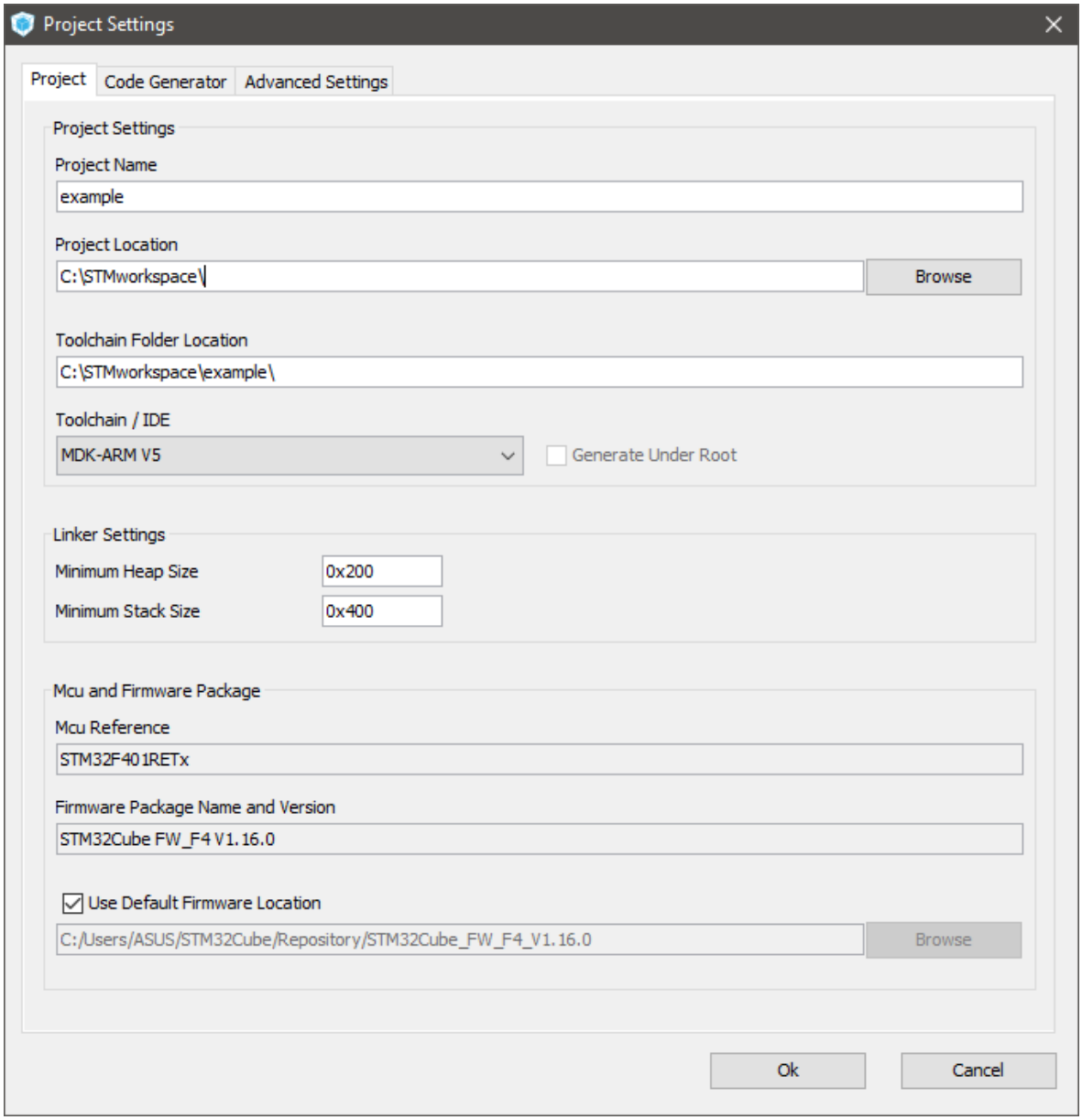
Также, зайдем в настройки GPIO. В этом окне можно настроить режимы работы портов ввода-вывода, а в частности – настроить, по какому фронту сработает наше прерывание. По умолчанию установлено срабатывание по переднему фронту. Давайте его таким и оставим:



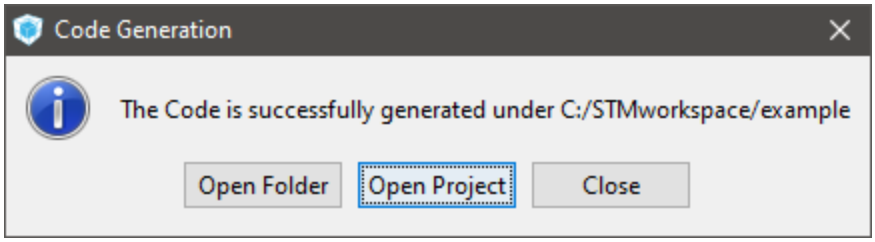
На этом настройка нашего проекта в Cube завершена. Для продолжения необходимо запустить генератор кода. Нажмем на кнопку с шестерней в верхней части интерфейса:



Откроется окно настроек проекта. Придумаем нашему проекту имя и выберем среду для дальнейшей разработки – MDK-ARM v5 (Keil v5, например).

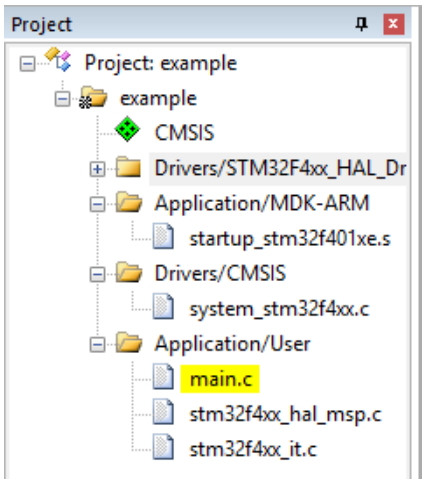


После этого Cube автоматически сгенерирует проект со всей заданной ранее конфигурацией, но без какой-либо логики работы или функционала. Выберем пункт “Open Project”, после чего проект откроется в среде Keil.



Cube добавил все необходимые для работы файлы в дерево файлов проекта. Найдем среди них main.c, с которого начнем реализацию функционала. Вы вольны добавлять код в любое место любого файла, но Cube ожидает от пользователя ввод модификаций в специально обозначенных местах, чтобы при повторной генерации кода изменения не были стерты. Вот так обозначаются эти места:

```
66  int main(void)
67  {
68
69    /* USER CODE BEGIN 1 */
70
71    /* USER CODE END 1 */
```



Ради примера, давайте реализуем смену состояния светодиода по фронту сигнала с кнопки. Вся периферия уже настроена для нас, остается только написать функционал. Cube генерирует код на основе библиотеки HAL, потому продолжим программирование именно на ней. Добавим в main.c следующий код:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
    if(GPIO_Pin == GPIO_PIN_13){
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
    }
}
```

Сгенерированный Cube код позаботится об обработке прерывания и установке необходимых флагов, после чего вызовет HAL_GPIO_EXTI_Callback. Так как обработчик прерывания один на все линии, необходимо проверить, было ли это событие от кнопки. Для этого в HAL предусмотрены макросы, такие как GPIO_PIN_13. На этом реализация функционала завершена, можно скомпилировать код, отправить его на контроллер и проверить его работу.