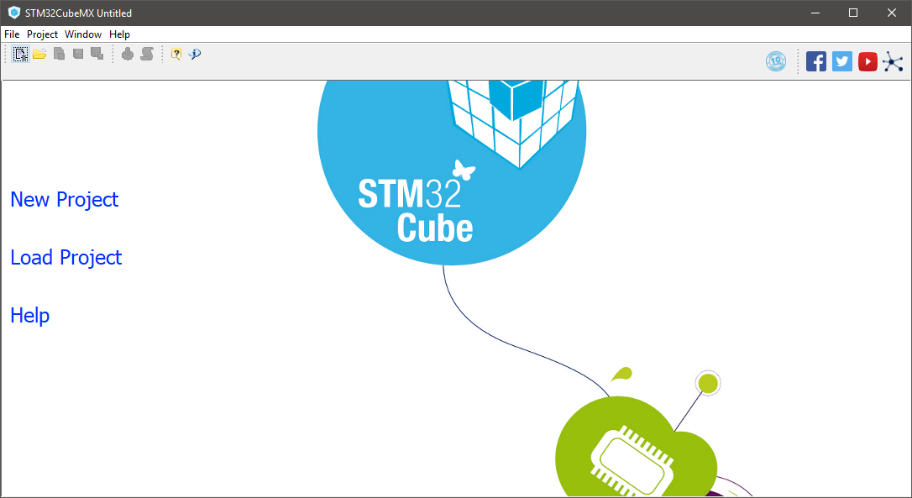
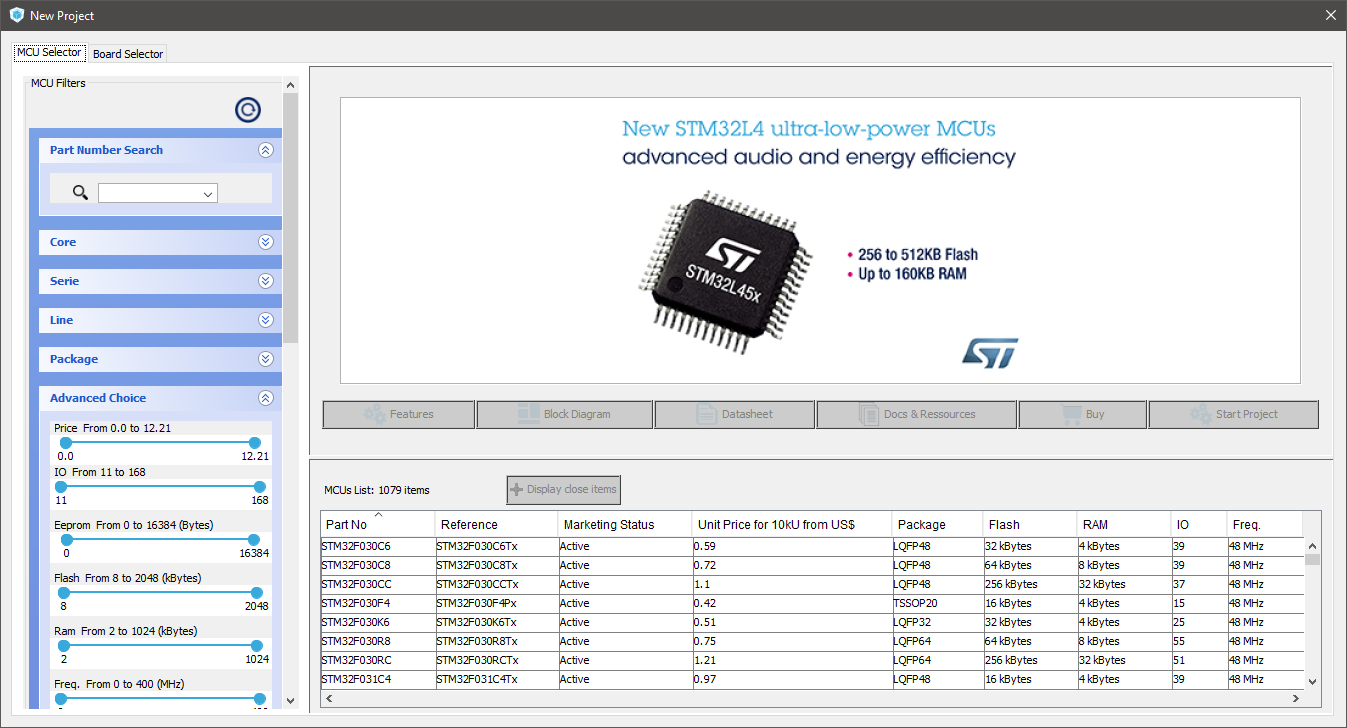
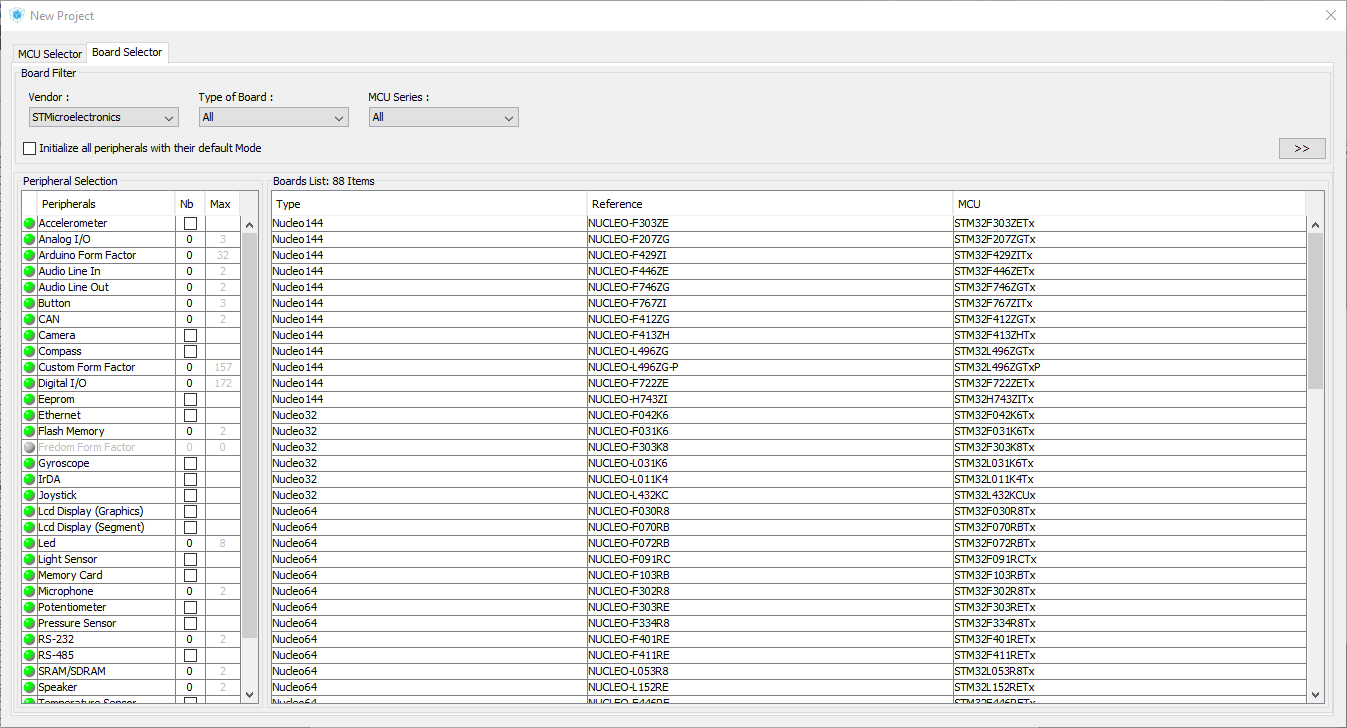
Создание проекта в CubeMX для STM32F401 (Nucleo)

Пример настройки и использования GPIO + EXTI

*Примечание: перед началом работы скачайте и установите утилиту CubeMX c сайта ST:* [*https://goo.gl/y69SeR*](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”.



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:

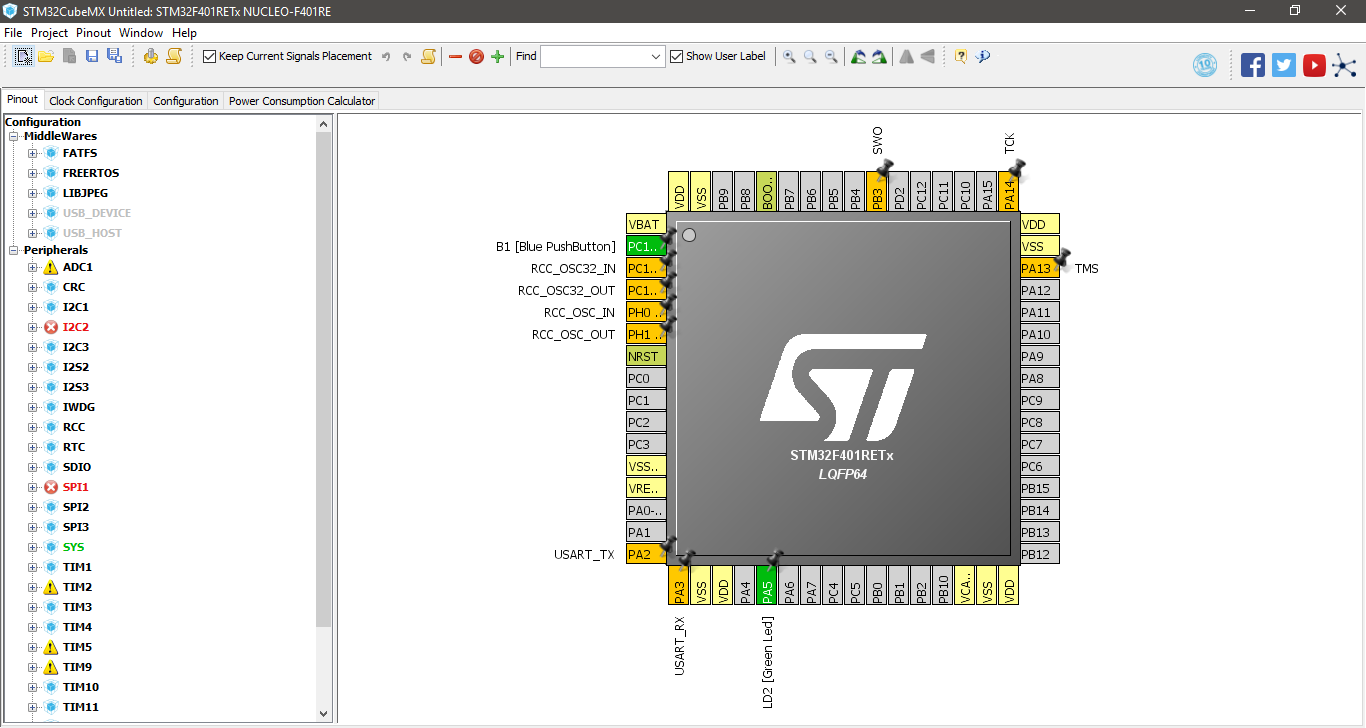
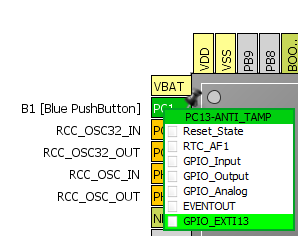
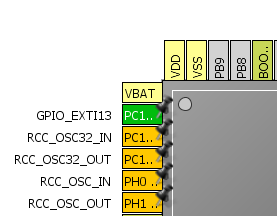
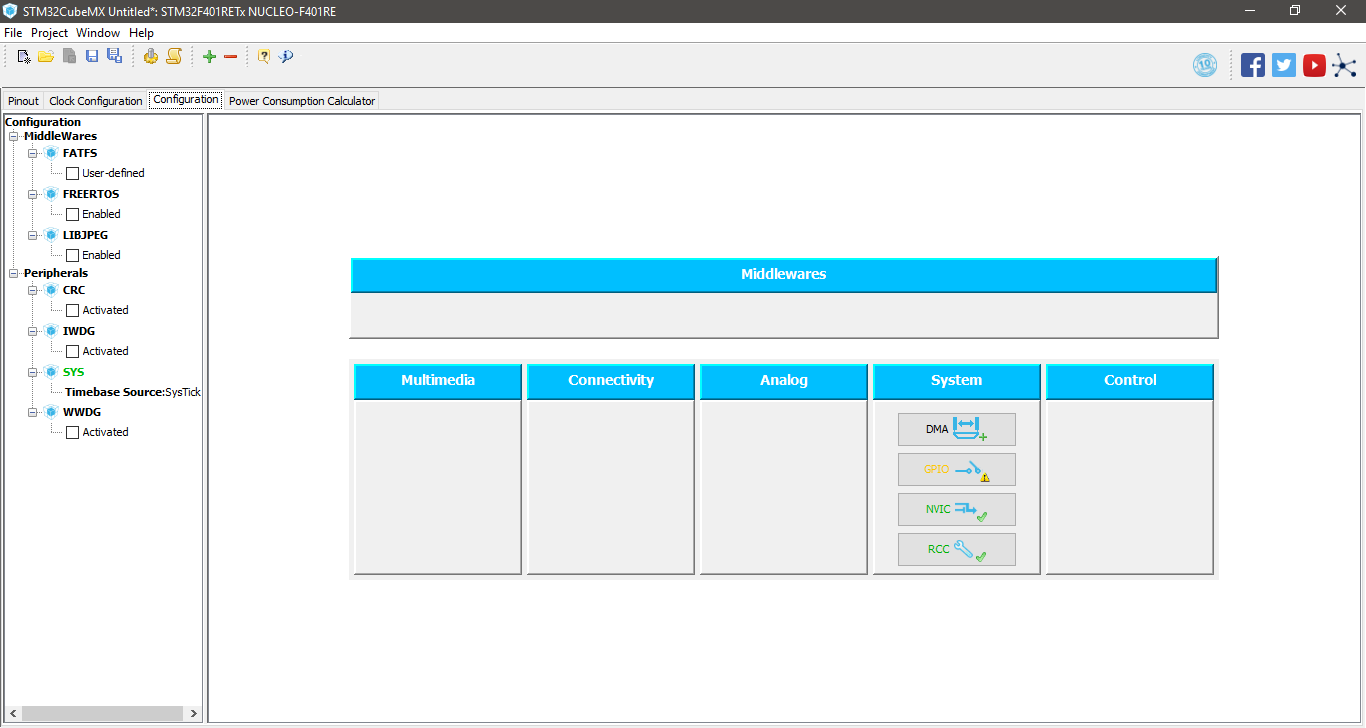
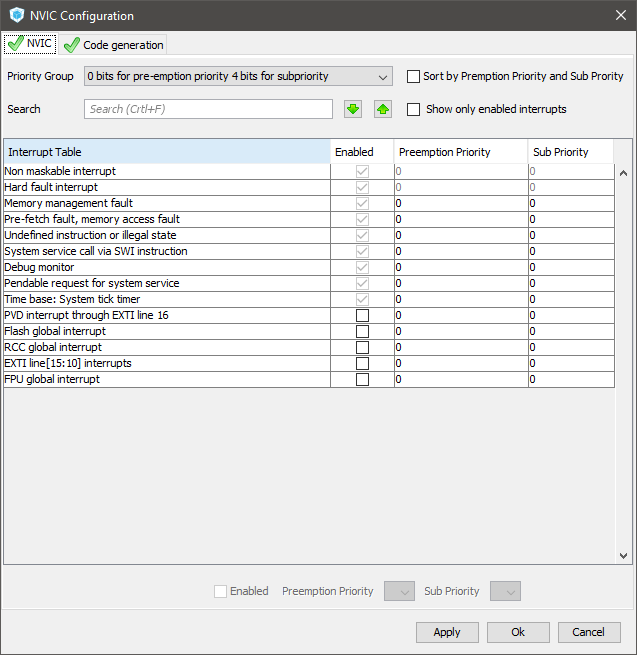
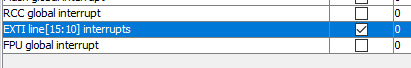
Откроется окно для выбора функционала контроллера. С помощью него можно настраивать, какие функции и выводы контроллера будут использованы в проекте.

Рис. 1 - Вкладка «Pinout»

Некоторые выводы контроллера заняты, так как на выбранной плате к ним подключена периферия. «Пользовательские» выводы выделены зеленым цветом. Название периферии указано рядом с выводом. 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, в группе «System» нам доступны настройки контроллера ПДП (DMA), портов ввода-вывода (GPIO), контроллера прерываний (NVIC), 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:

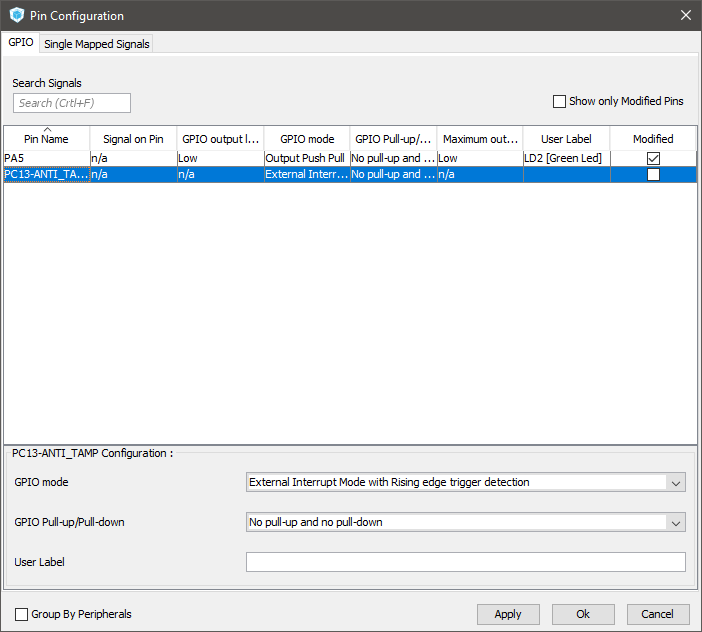
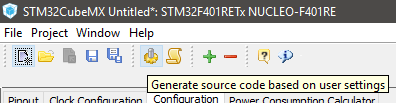
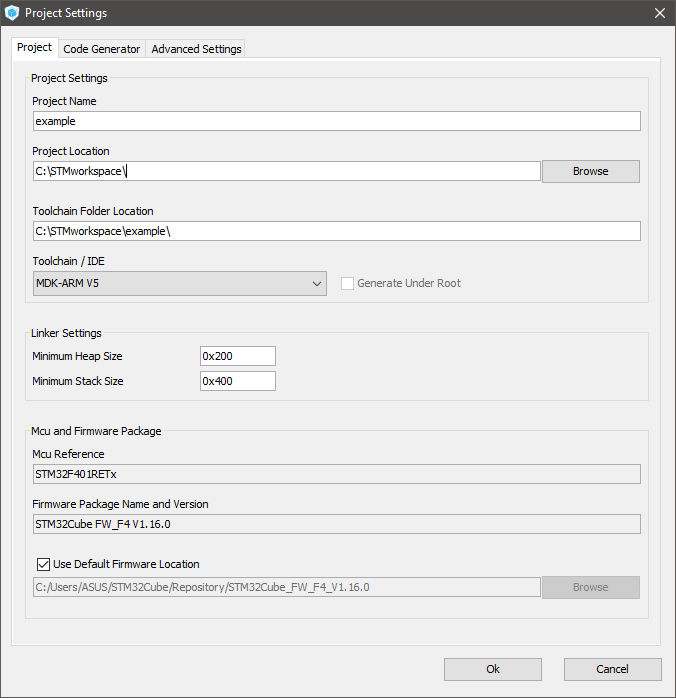
Также, зайдем в настройки GPIO. В этом окне можно настроить режимы работы портов ввода-вывода, а в частности – настроить, по какому фронту сработает наше прерывание. По умолчанию установлено срабатывание по переднему фронту. Давайте его таким и оставим:

Рис 2 – Окно настройки GPIO

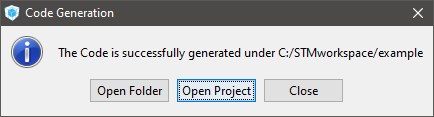
Не забываем нажать «ОК».

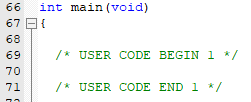
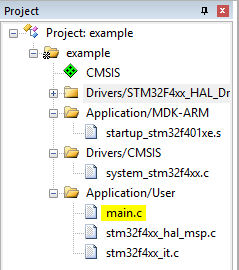
На этом настройка нашего проекта в Cube завершена. Для продолжения необходимо запустить генератор кода. Нажмем на кнопку с шестерней в верхней части интерфейса:

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

При желании место размещения проекта можно изменить, нажав «Browse».

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:

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.