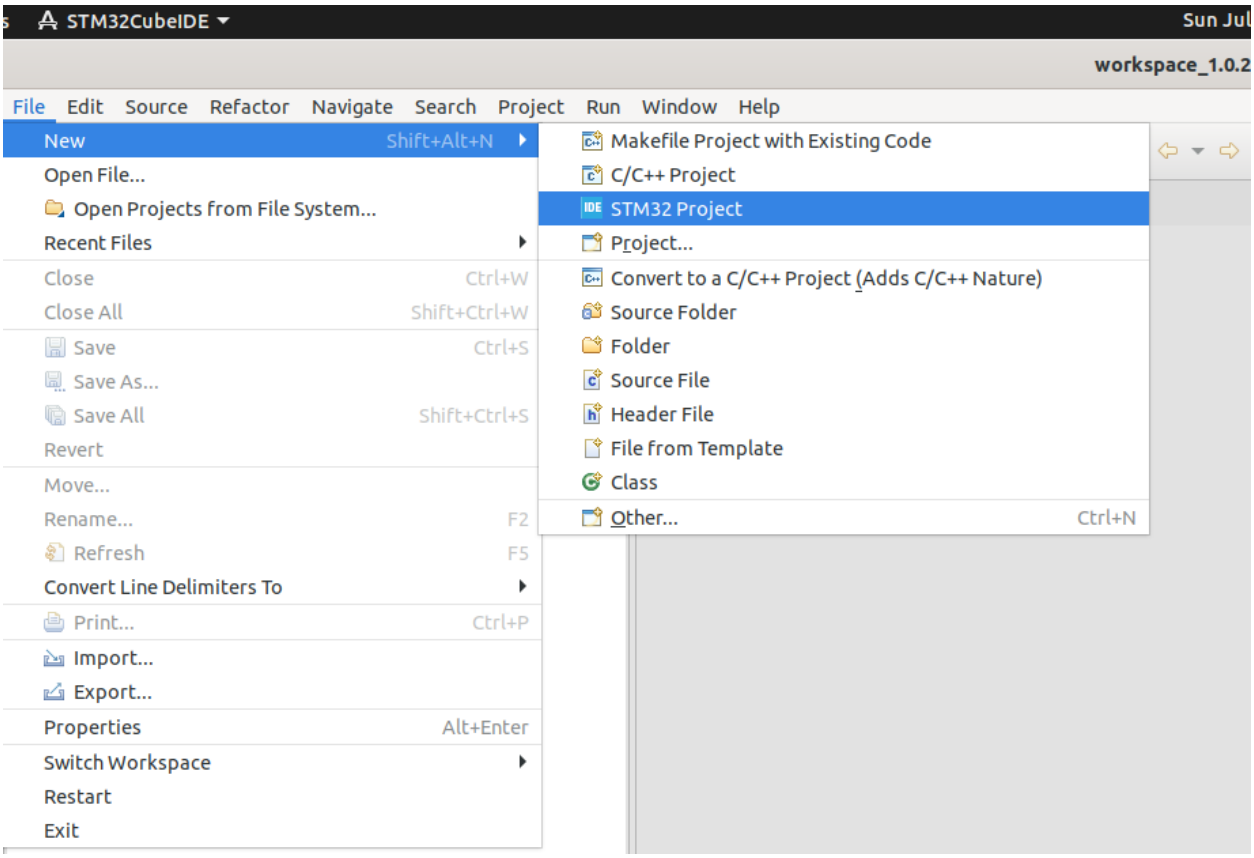
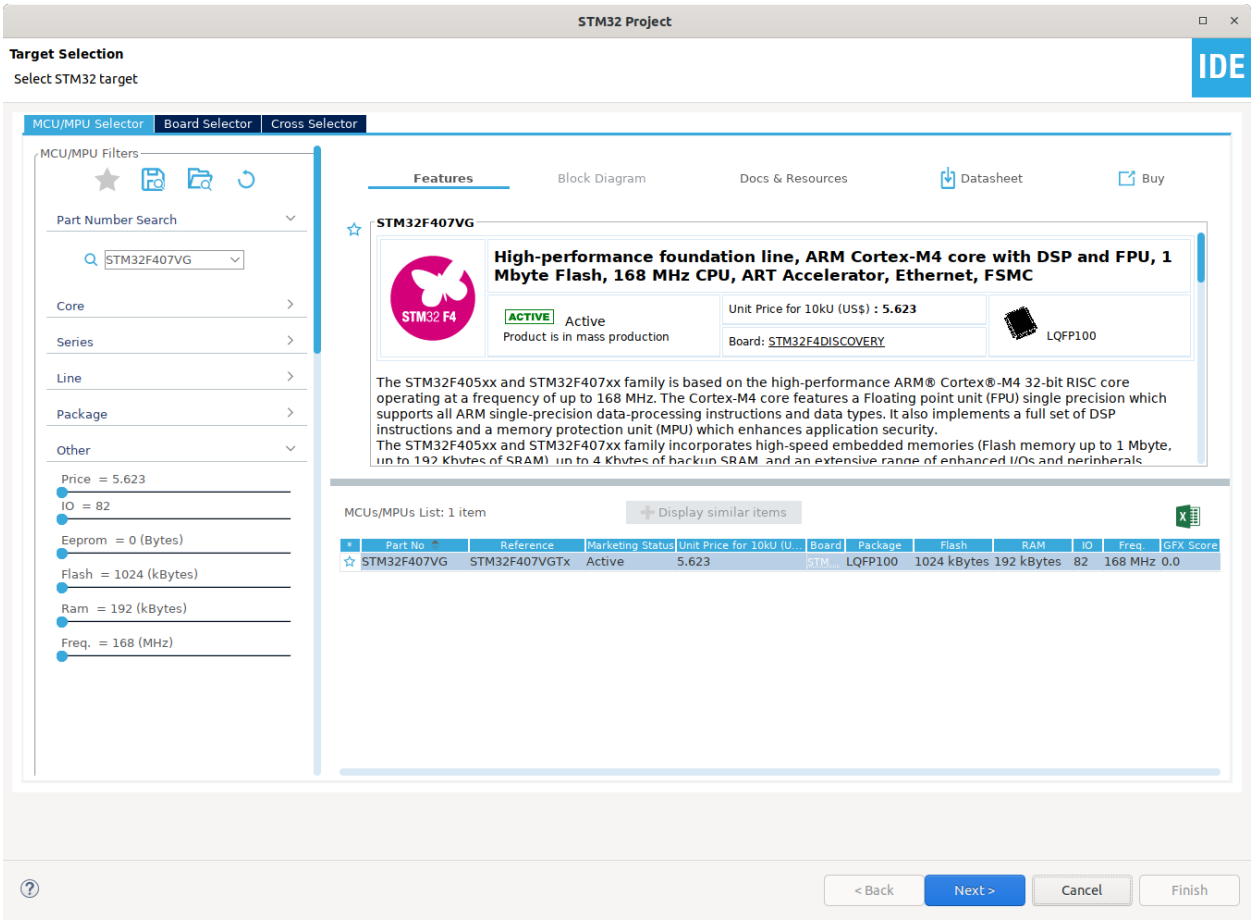
**STM32CubeIDE + GL Starter Kit**
**Getting started**

## 1. Создание нового проекта

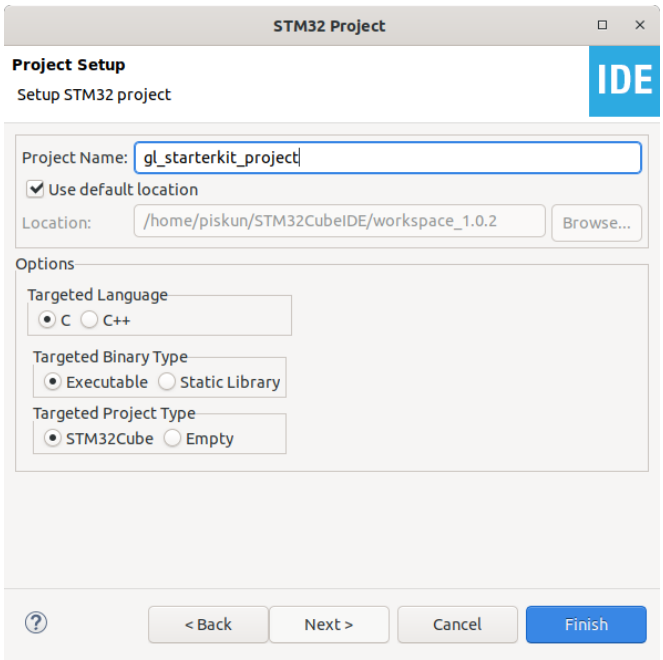
В STM32CubeIDE *File -> New -> STM32 Project*



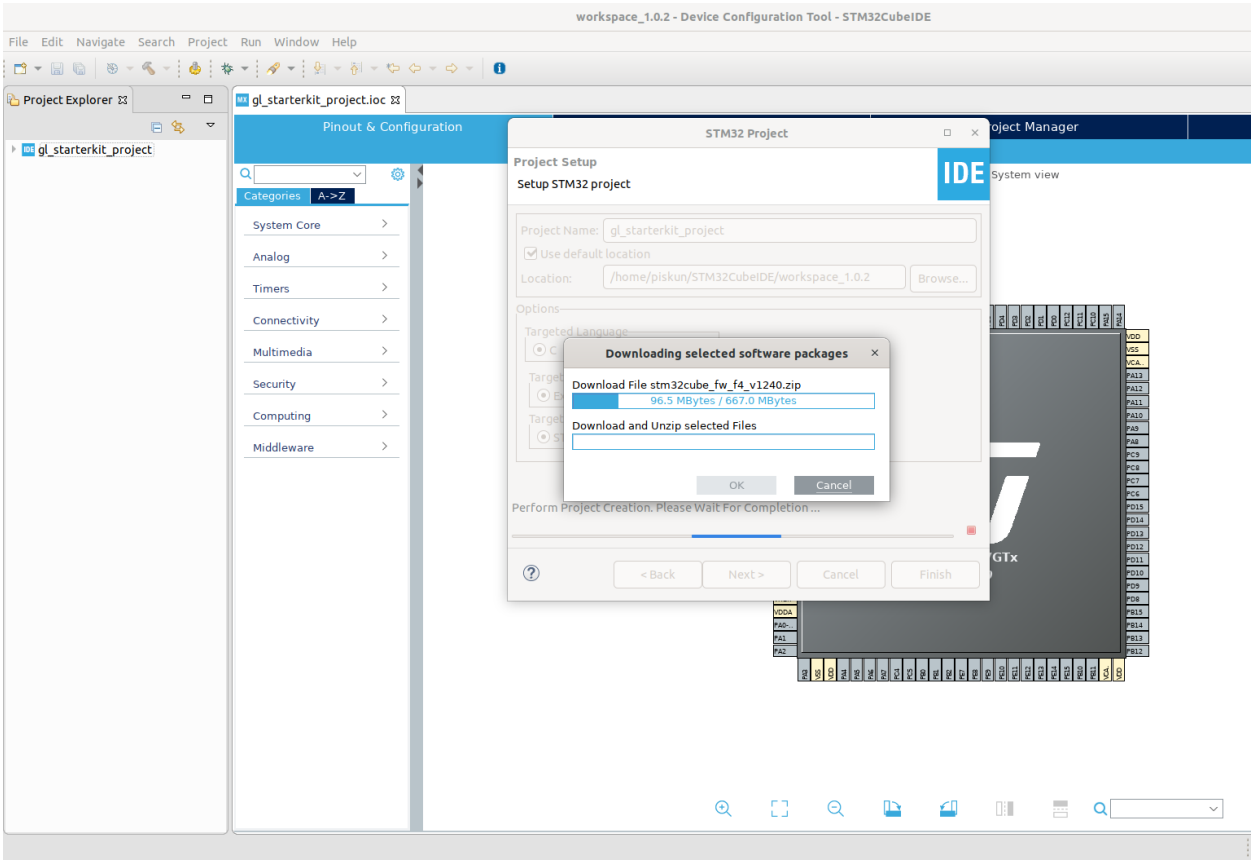

В *Target Selector* во вкладке *MCU/MPU Selector* выбрать *STM32F407VG*



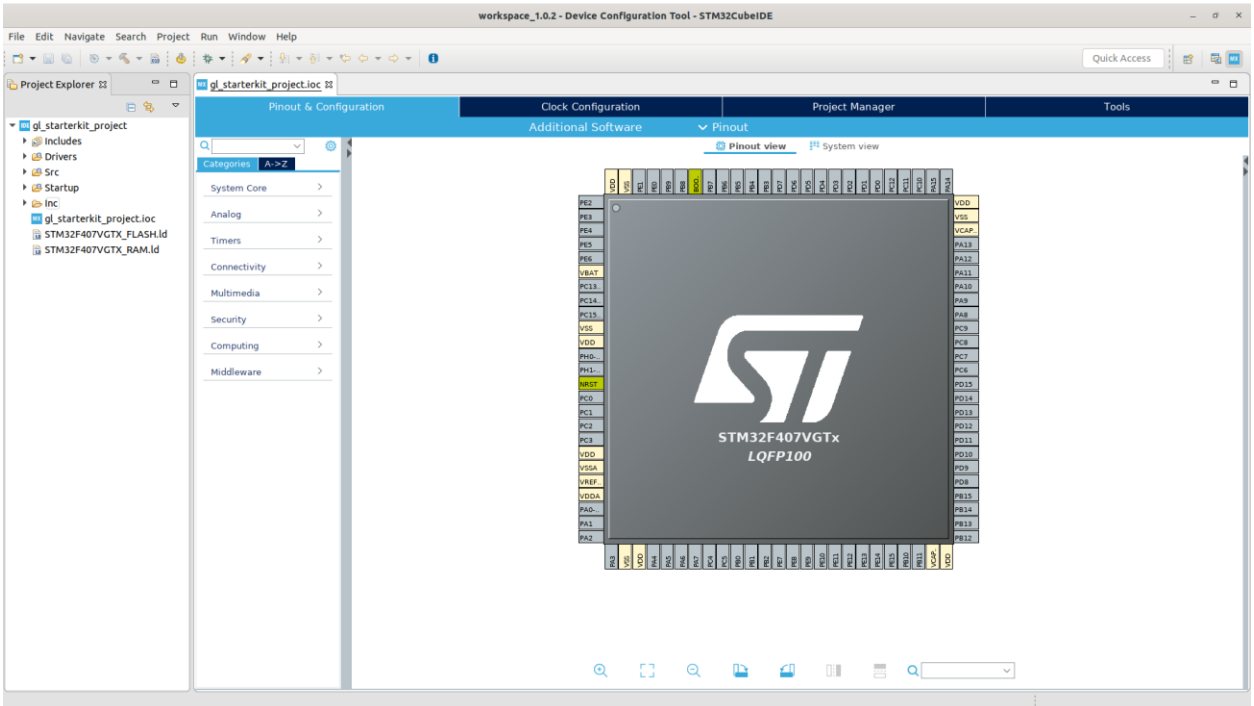

Ввести имя проекта



При создании первого проекта для микроконтроллера серии STM32F4 будет загружен Firmware Package (HAL, скрипты линкера, стартап файлы и т.п. ).

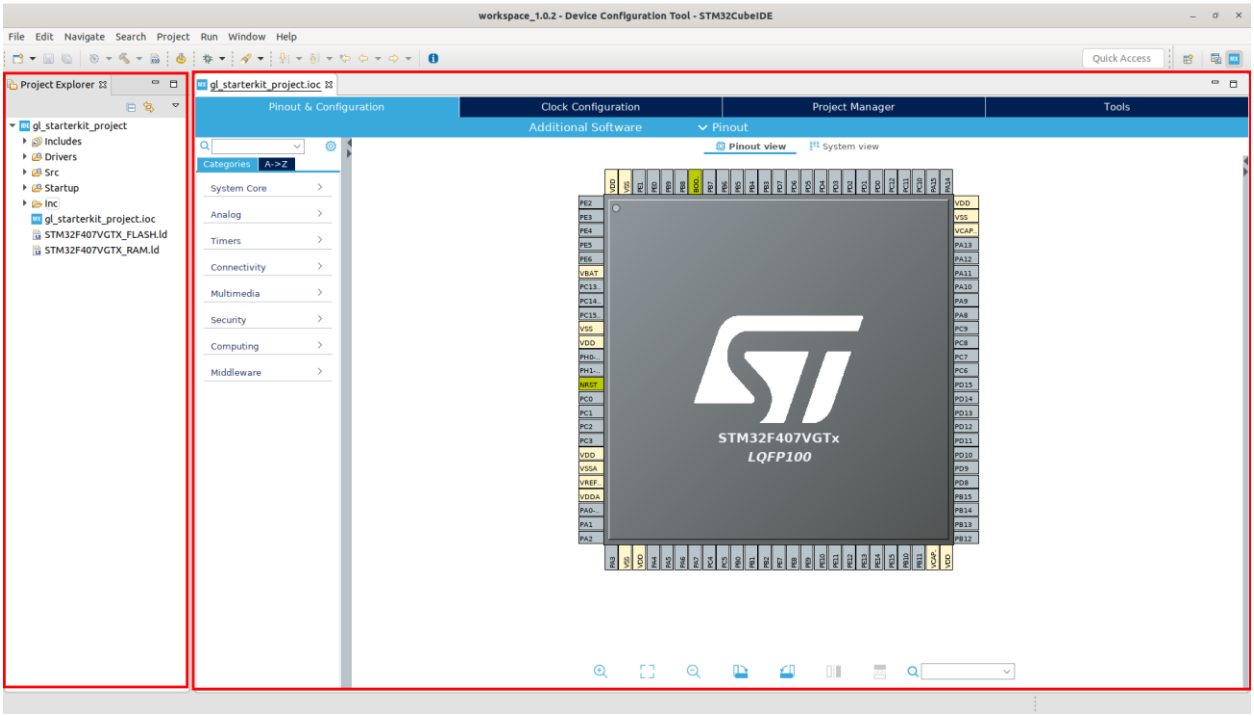



После завершения создания нового проекта IDE будет иметь следующий вид



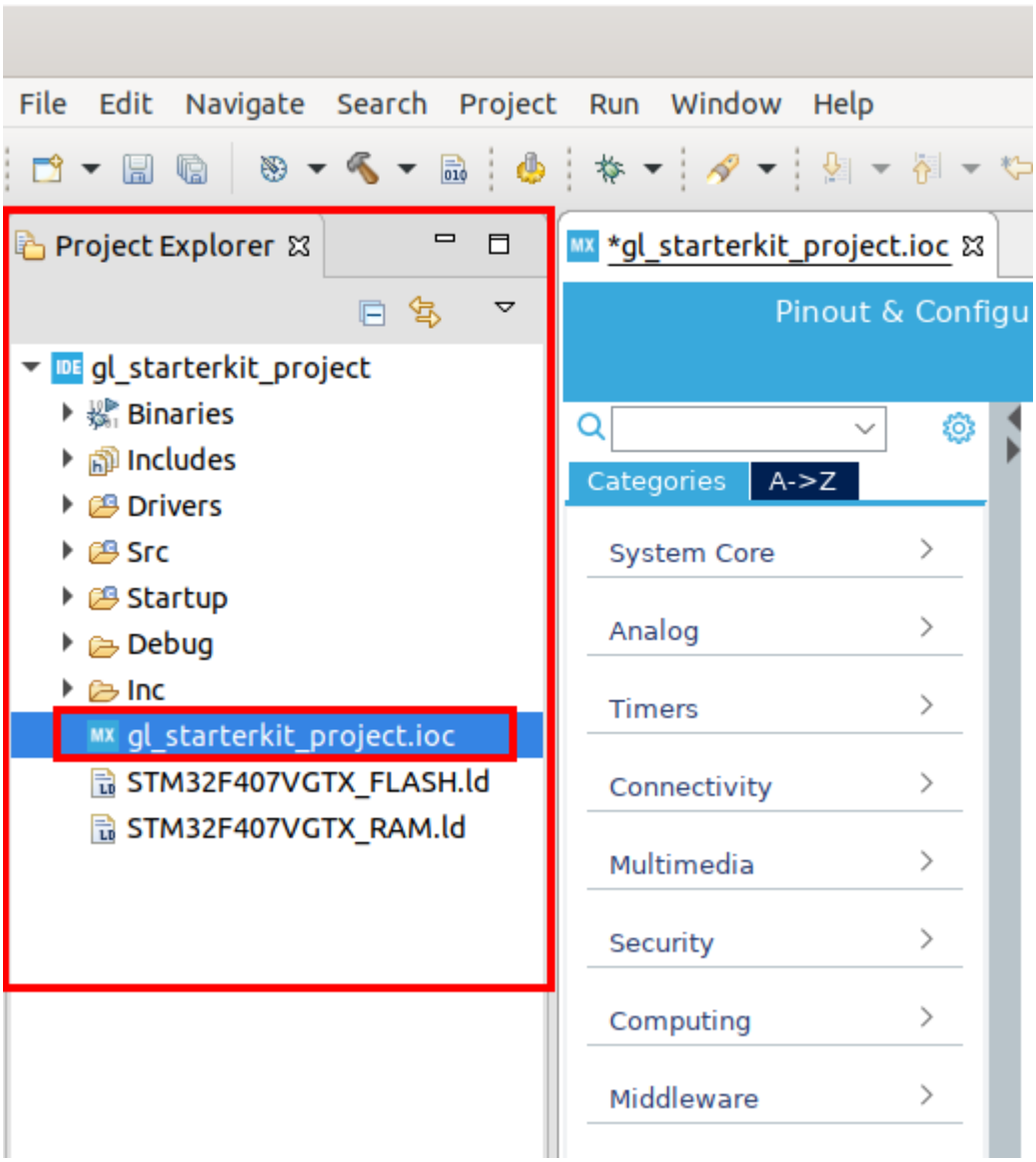
Главное окно IDE разделено на две части:

- *Project Explorer* (слева) - навигация по файлам проекта
- *Device Configuration Tool* (справа) - конфигурирование контроллера - тактирование, порты ввода/вывода, периферия, опции кодогенерации и т.д.




**2. Обзор Device Configuration Tool**

Для того, чтобы открыть окно Device Configuration Tool нужно дважды кликнуть по файлу с расширение *.ioc*



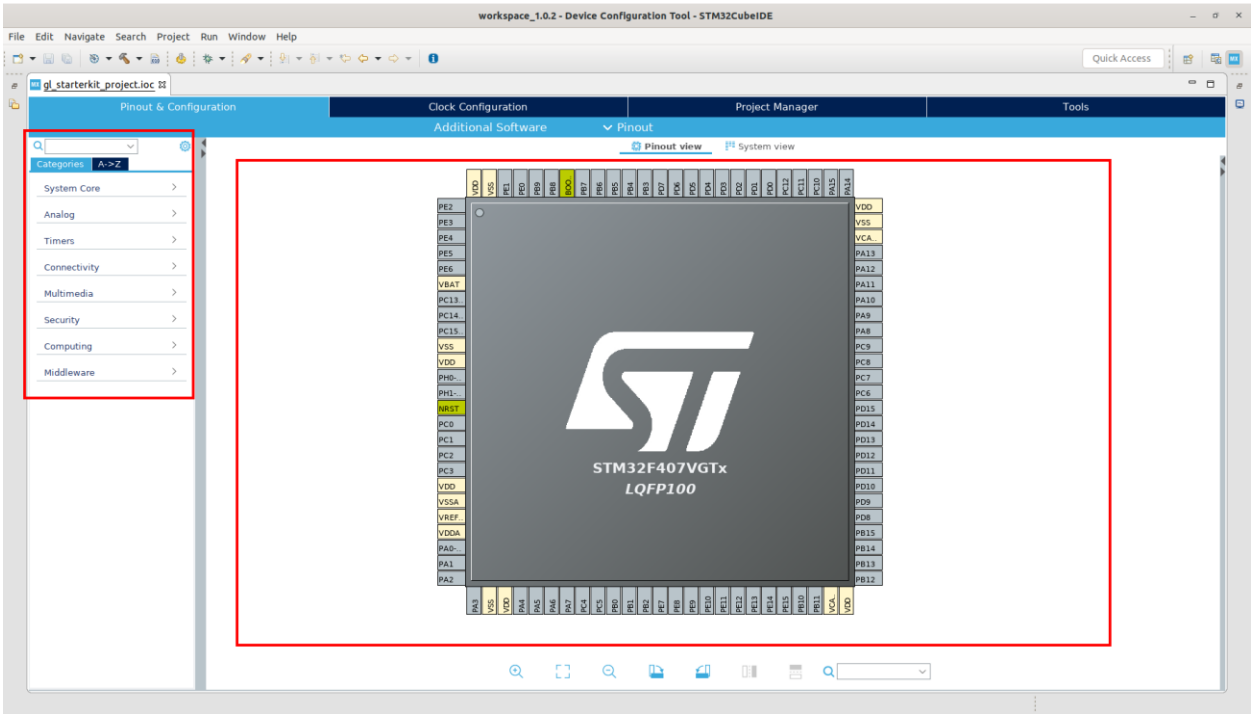

Device Configuration Tool состоит из четырех вкладок:

- *Pinout & Configuration* - настройка портов ввода/вывода и периферии;
- *Clock Configuration* - настройка тактирования - выбор тактового генератора, настройка PLL, частот шин, периферии;
- *Project Manager* - можно задать минимальный размер stack/heap, используемую версию Firmware Package, а также опции кодогенерации.
- *Tools* - можно приблизительно рассчитать потребление тока контроллером.

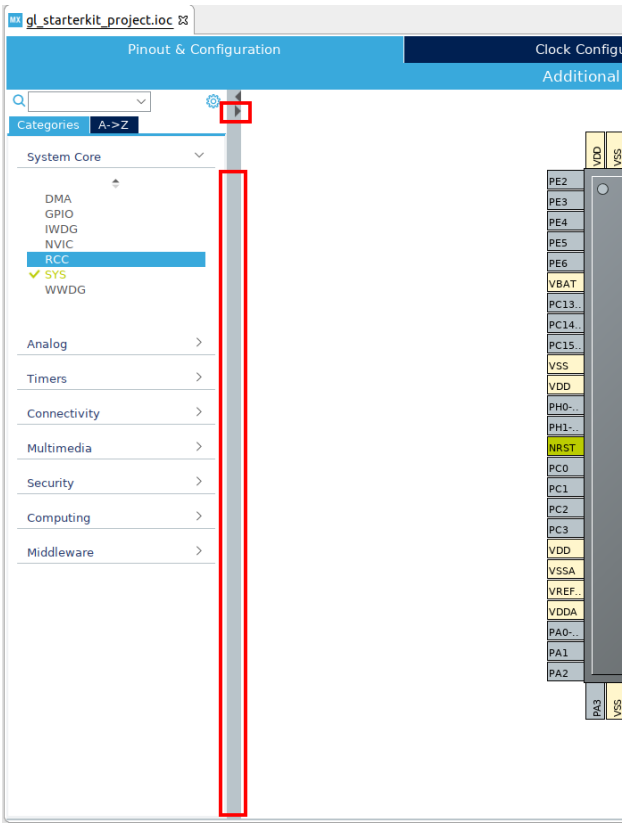
### 2.1 Pinout & Configuration

По умолчанию окно разделено на две части:

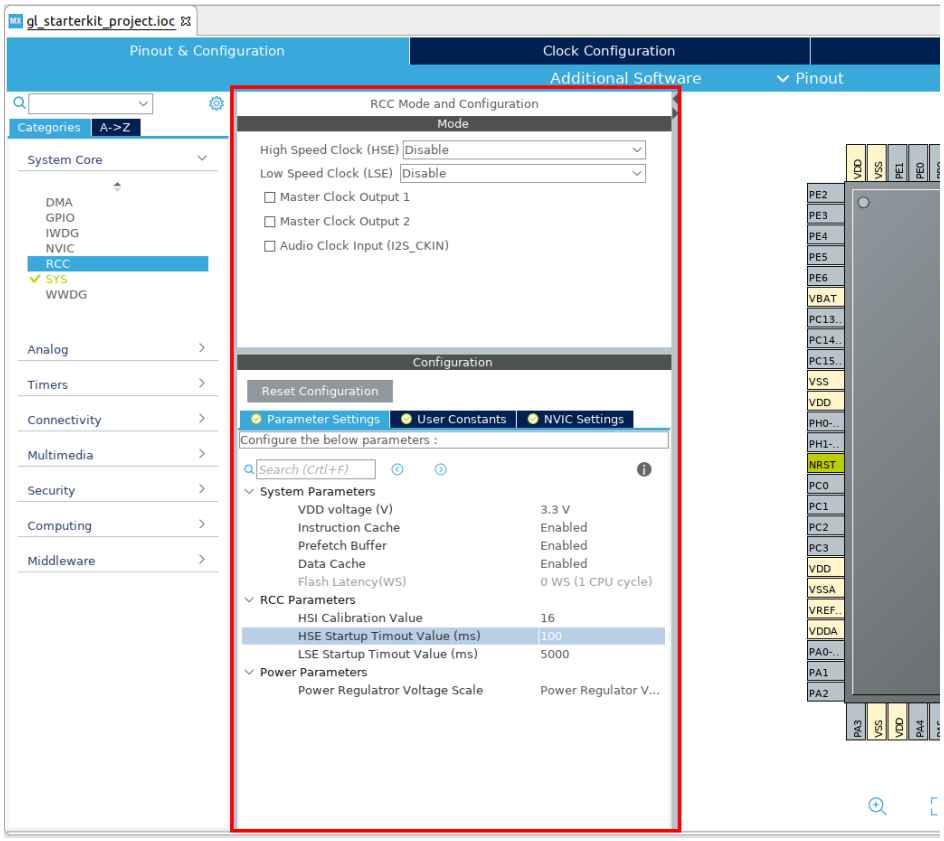
- *Peripherals Configuration* - периферия контроллера, сгруппированная по категориям (*Categories*). Также можно переключить список на отображение в алфавитном порядке (*A->Z*);
- *Pinout view* - отображение контроллера в выбранном корпусе (LQFP100) со всеми пинами (портами ввода/вывода).



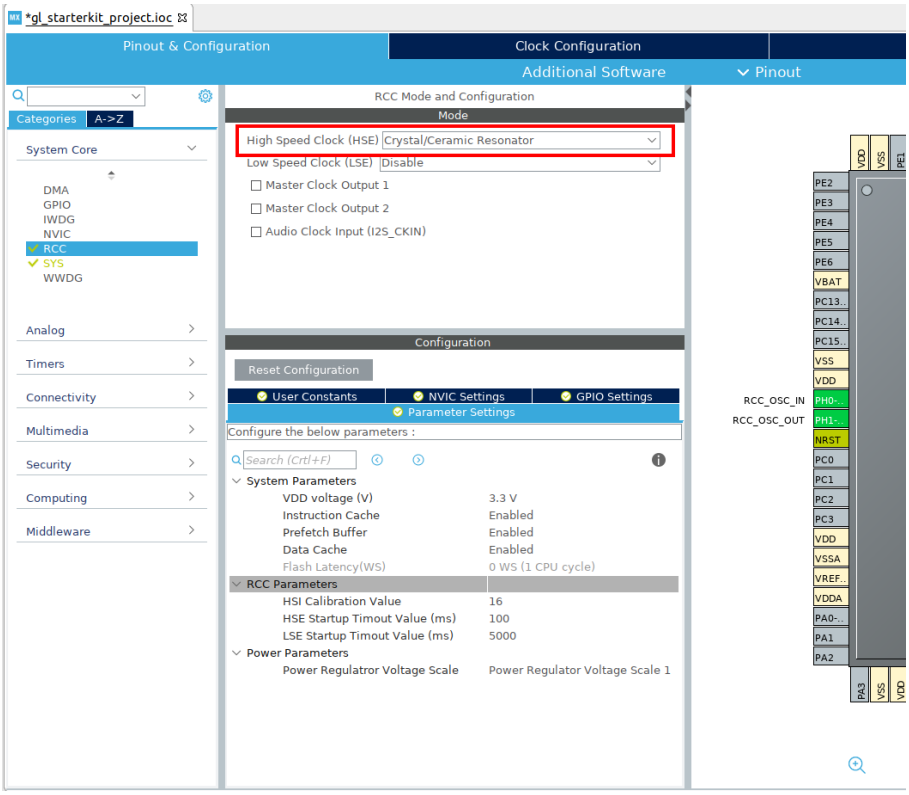

После включения/сброса MCU тактируется от внутреннего RC-генератора (HSI - High Speed Internal oscillator), который не отличается высокой стабильностью частоты. StarterKit имеет внешний кварц на 8 МГц. Переключим источник тактовых импульсов на HSE (High Speed External oscillator) - внешний кварцевый резонатор. В *Peripherals Configuration* выберем *System Core -> RCC.* По умолчанию окно настройки периферии скрыто. Для того, чтобы его открыть, нужно кликнуть на вертикальную полосу-разделитель между окнами *Peripherals Configuration* и *Pinout view* либо нажать на треугольник на той же полосе-разделителе



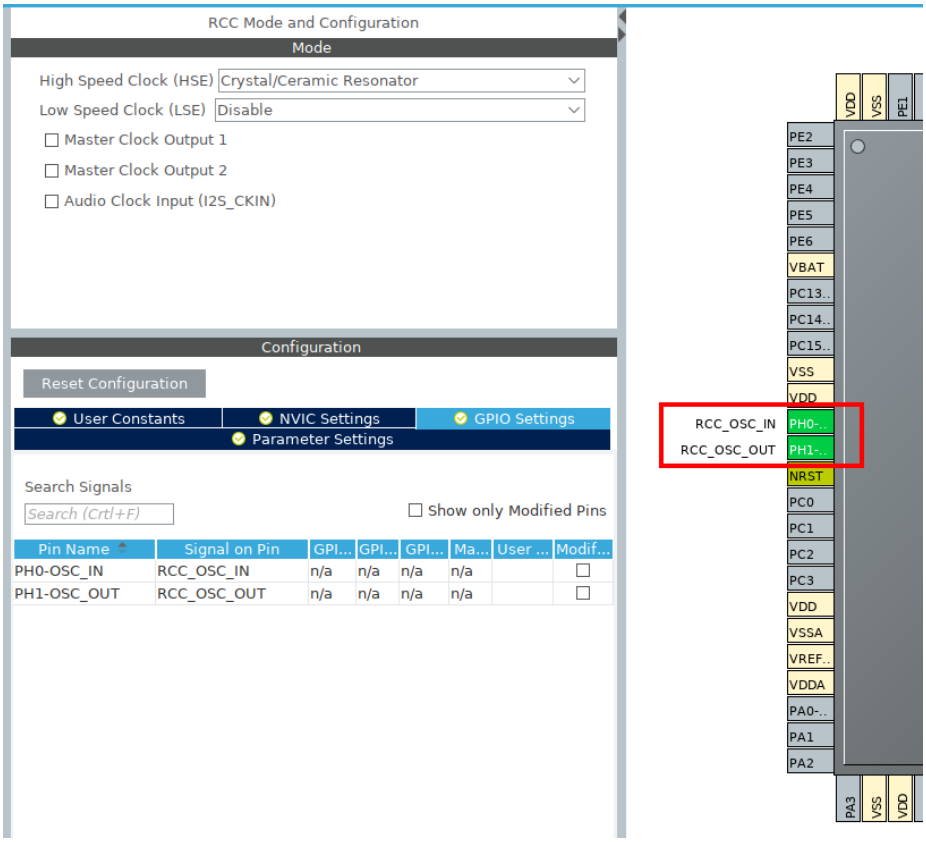

Окно настройки выбранной периферии выглядит следующим образом



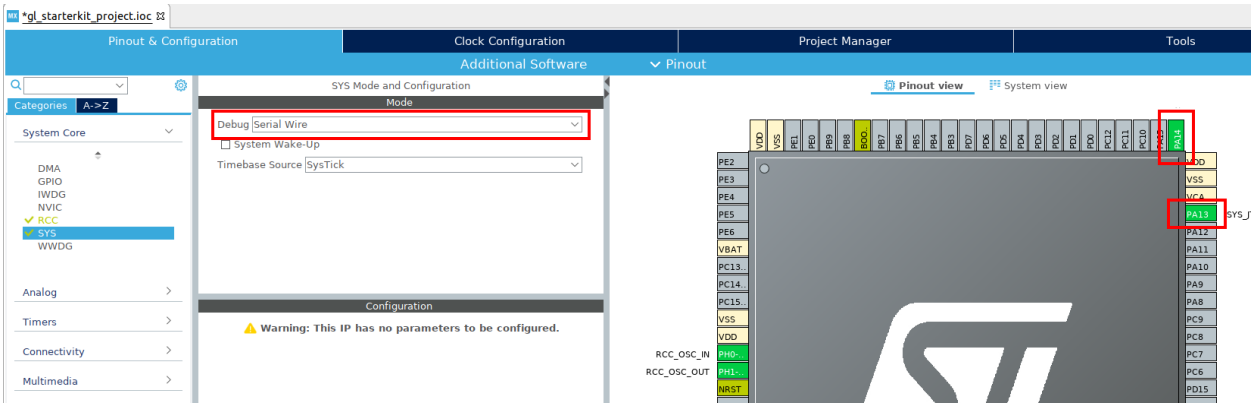
Включим тактирование от HSE. Для этого в выпадающем списке *High Speed Clock (HSE)* выберем *Crystal/Ceramic Resonator*




После этих действий в *Pinout View* пины PH0 и PH1 (OSC_IN и OSC_OUT соответственно) стали зелеными т.е. эти пины стали задействованы



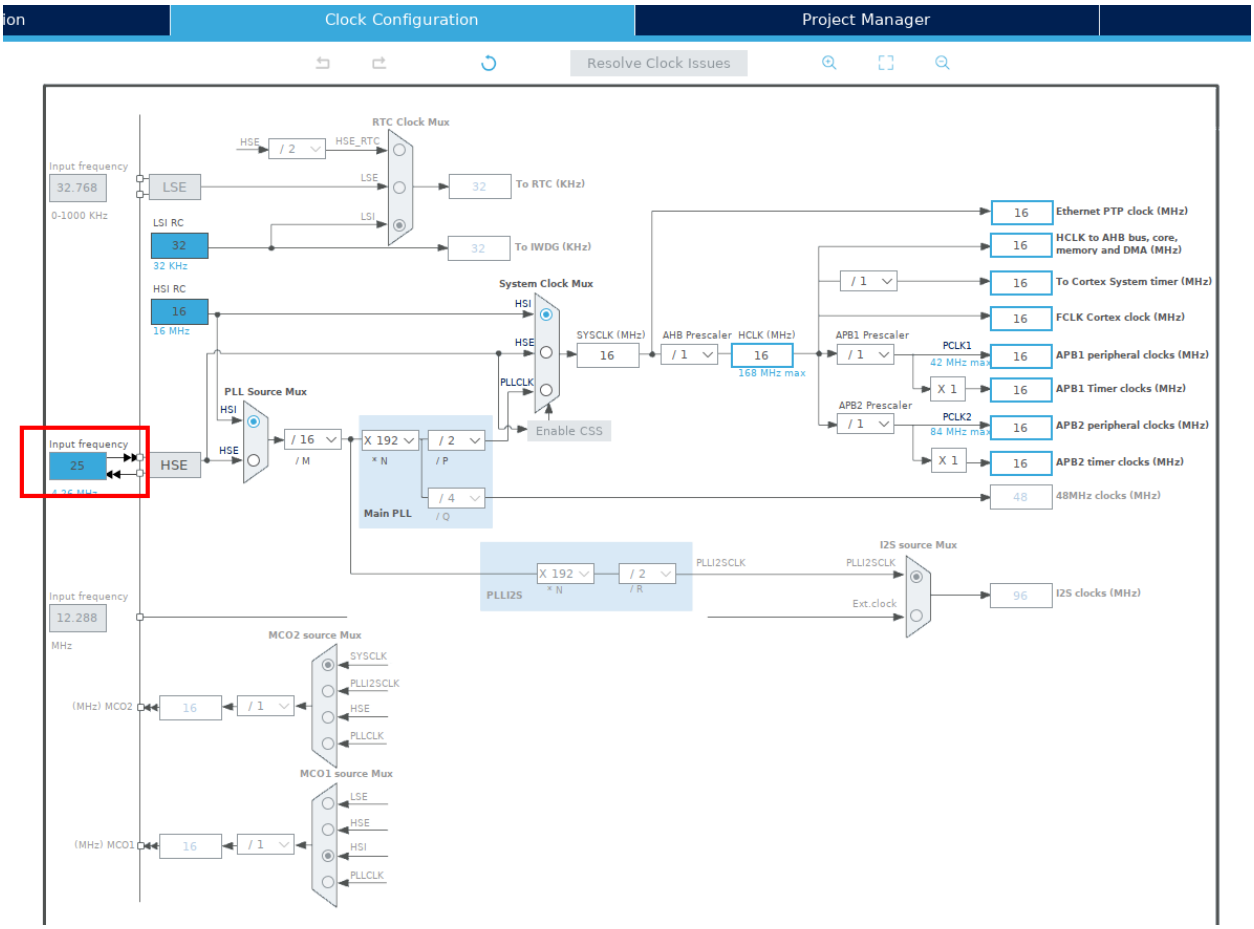

Далее необходимо разрешить отладку по SWD (последовательный отладочный интерфейс). Для этого в *Peripherals Configuration* переходим в категорию *SYS.* В выпадающем списке *Debug* выбираем *Serial Wire*



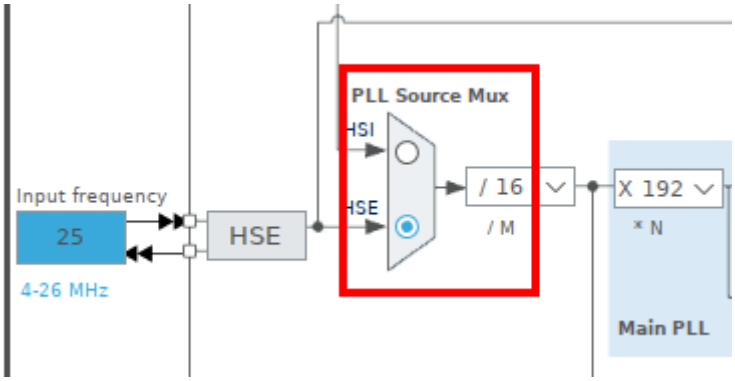

Так же видно, что в *Pinout View* пины PA13 (SWDIO) и PA14 (SWCLK) стали задействованы

### 3.1. Clock Configuration

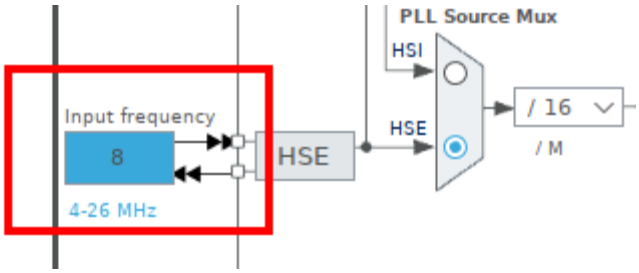
Показывает схему тактирования выбранного MCU




Так как мы включили тактирование от HSE, то блок *HSE* стал активным. Но источником тактовых импульсов по прежнему является HSI. Для переключения на HSE в блоке *PLL Source Mux* выберем HSE (просто кликнуть на “кружочек” напротив HSE)



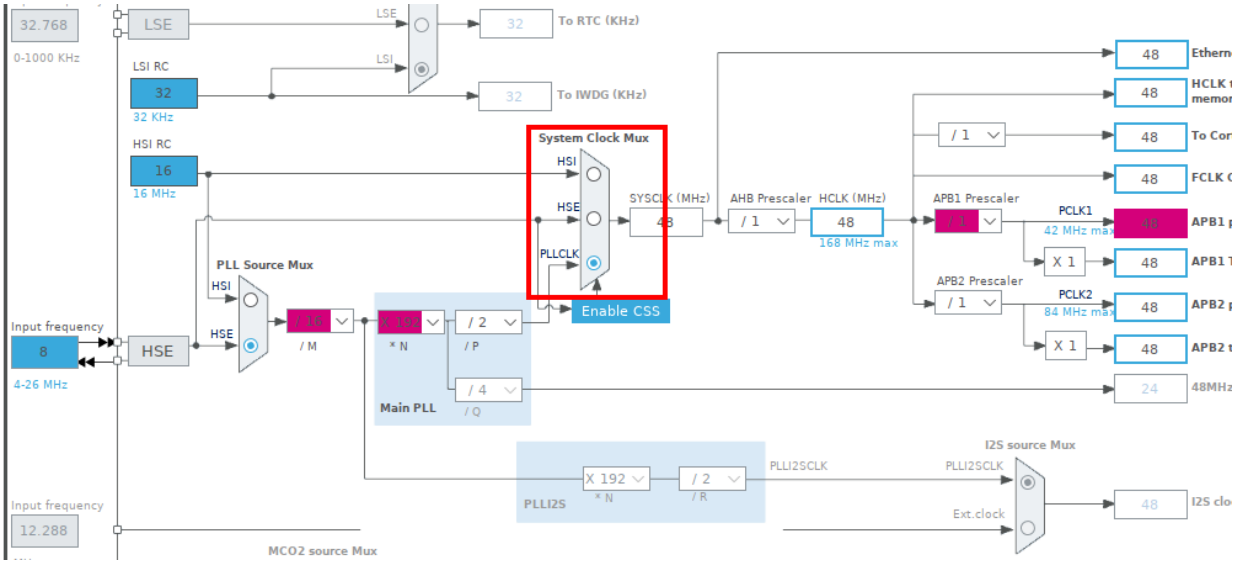

Как упоминалось ранее, на StarterKit стоит кварц на 8 МГц. *Исправим Input frequency*, т.к. по умолчанию стоит 25 МГц



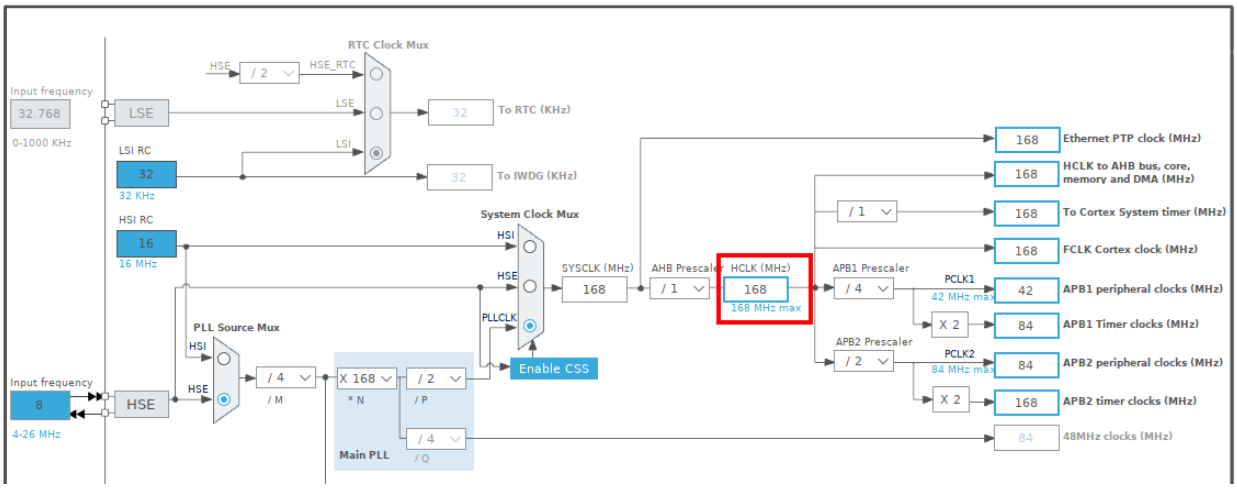

MCU имеет 3 шины:

- AHB - Advanced High-performance Bus. На этой шине работает ядро (core), DMA, память
- APB1/APB2 - Advanced Peripheral Bus. На этих шинах работают таймеры и почти вся периферия.

Максимальная частота AHB - 168 МГц. Для повышения частоты тактового генератора (HSE = 8МГц) задействуем PLL (Phase-Locked Loop). Для этого *System Clock Mux* выберем *PLLCLK.*




Красные значения делителей и частот означают, что частоты, после выбранных коэффициентов деления получились либо ниже допустимого предела, либо выше. Коэффициенты деления можно подобрать вручную. А можно эту задачу передать *Clock Configuration*. Так, например, если мы хотим получить частоту AHB = 168МГц, можно в *HCLK* ввести *168* и нажать *Enter*. Коэффициенты деления будут пересчитаны автоматически



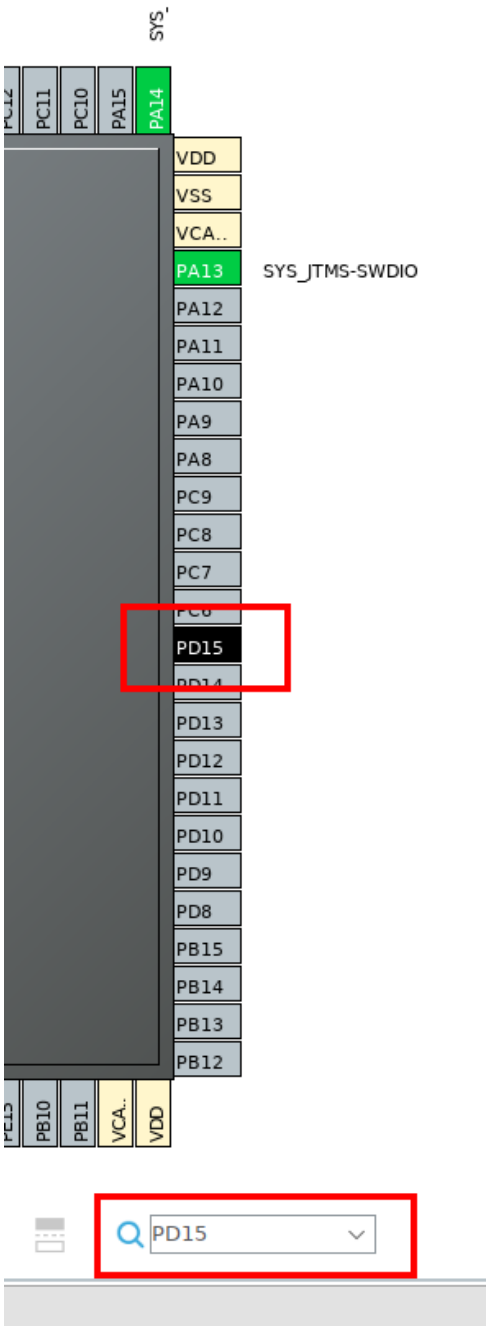

Настройка тактирования окончена. Теперь попробуем выполнить первую задачу - помигать светодиодом.

**4. Мигание светодиодом**

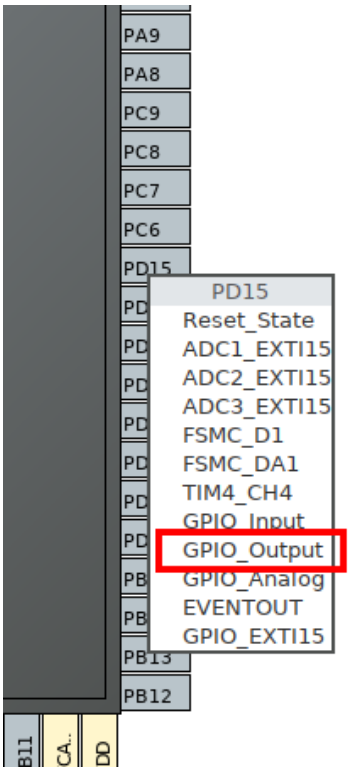
StarterKit имеет 4 светодиода:

- PD12 - зеленый;
- PD13 - оранжевый;
- PD14 - красный;
- PD15 - синий.

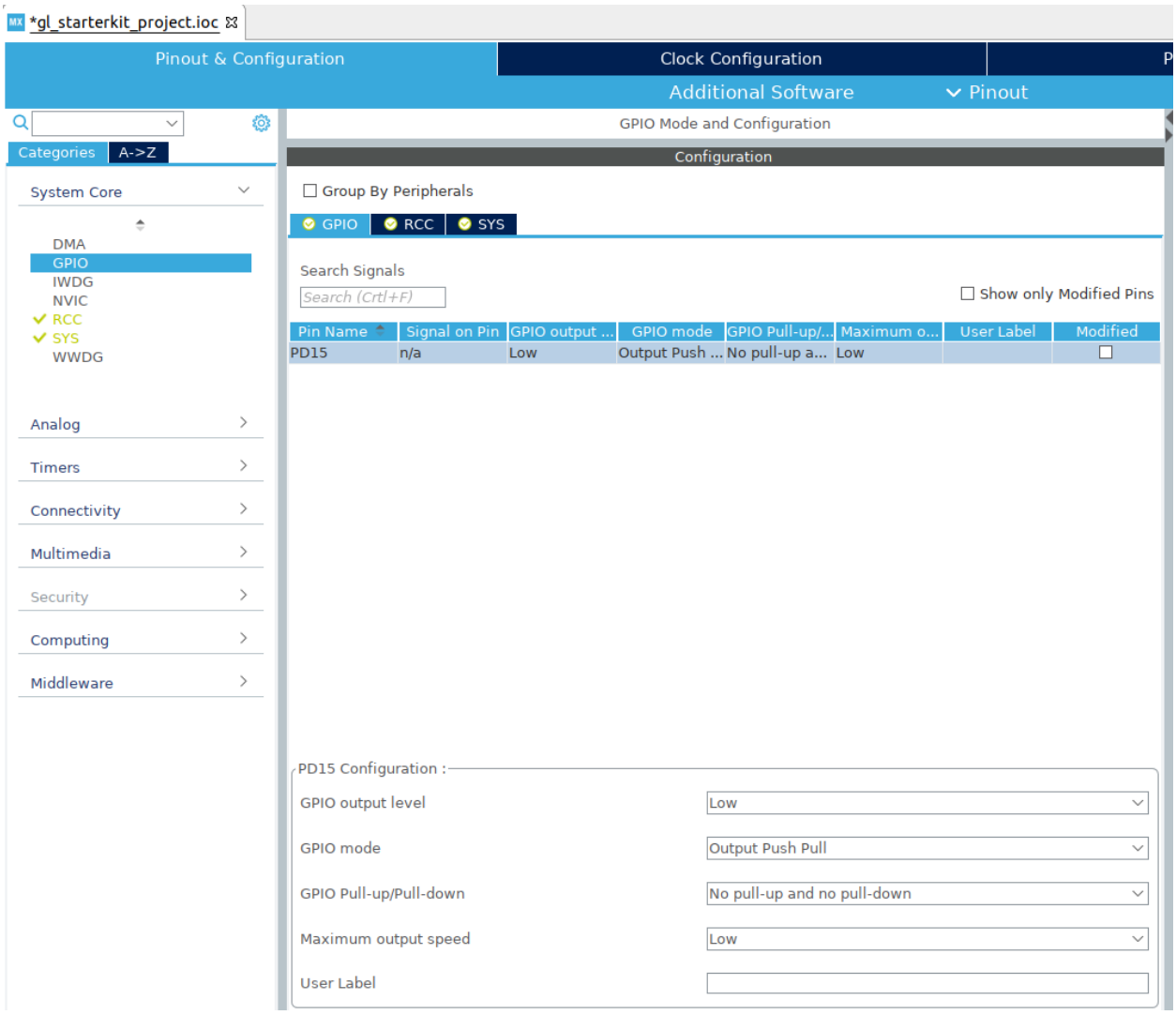
Помигаем, к примеру, синим светодиодом. Для этого нужно настроить пин #15 порта PORTD (PD15). Переходим на вкладку *Pinout & Configuration.* В *Pinout view* нужно найти пин PD15. Для этого можно воспользоваться поиском. В окне поиска введем *PD15*



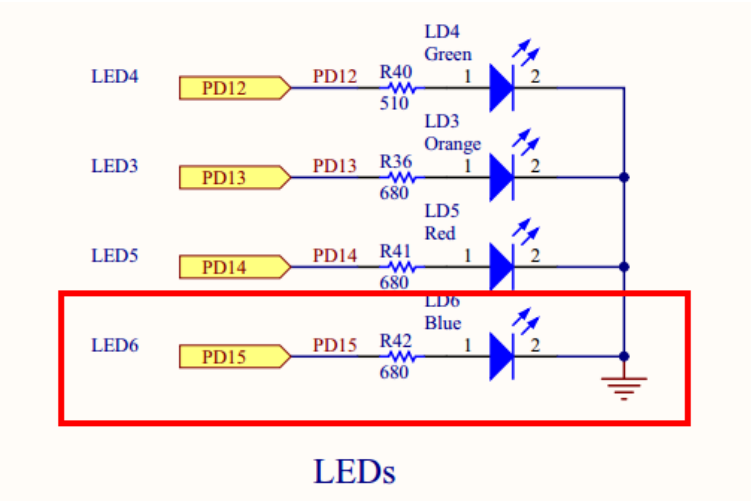

Для управления светодиодом пин PD15 должен быть настроен на выход. Для этого в *Pinout view* кликнем на *PD15* и из списка выберем *GPIO_Output*



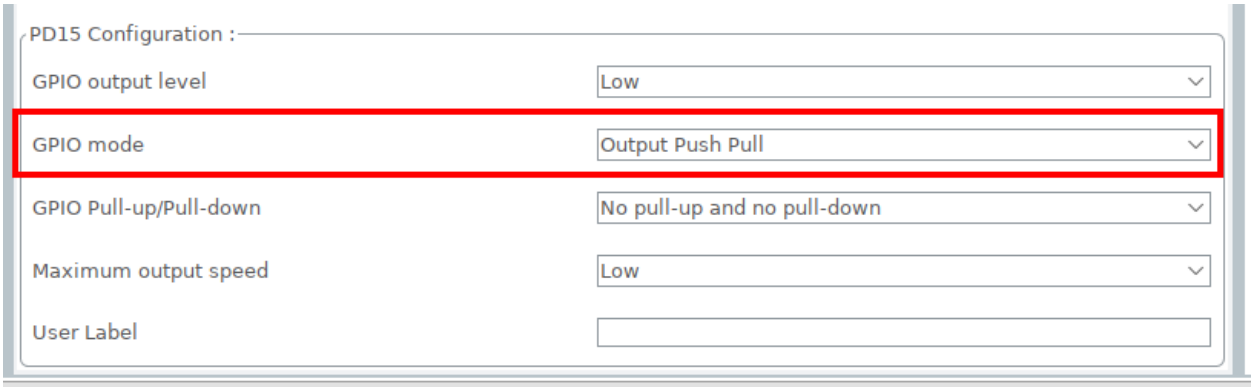

Далее переходим в *Peripheral Configuration* в категорию *System Core* в раздел *GPIO*. В списке появился PD15. После того, как мы выделим этот пин в списке, у нас появятся поля для настройки.




По схеме синий светодиод, подключенный к PD15, не содержит подтягивающего резистора

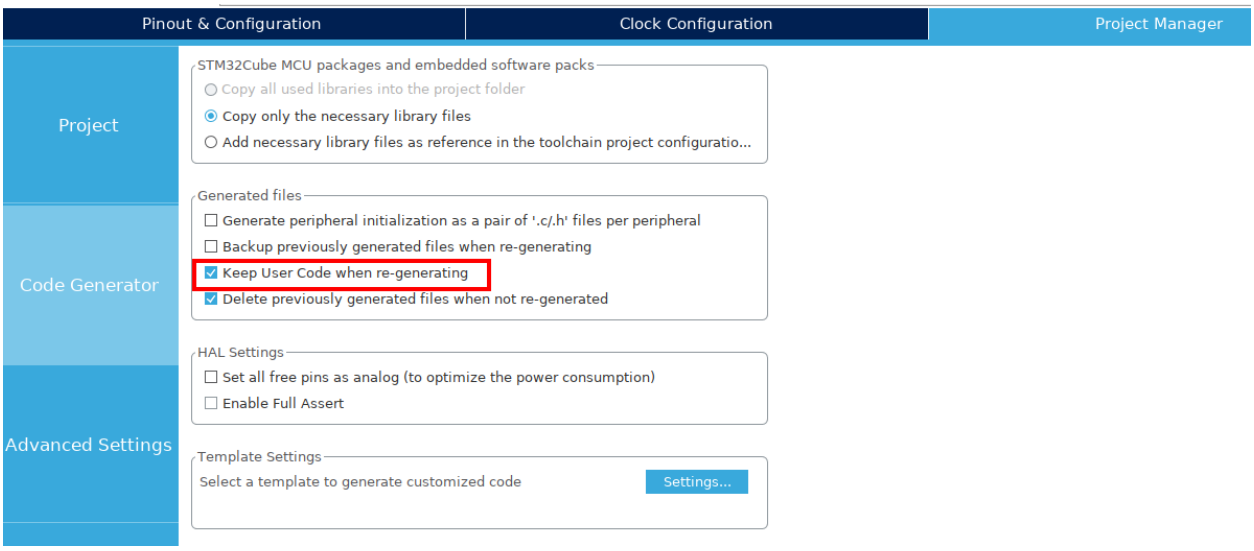


Поэтому режим работы пина (GPIO mode) должен быть *push-pull*

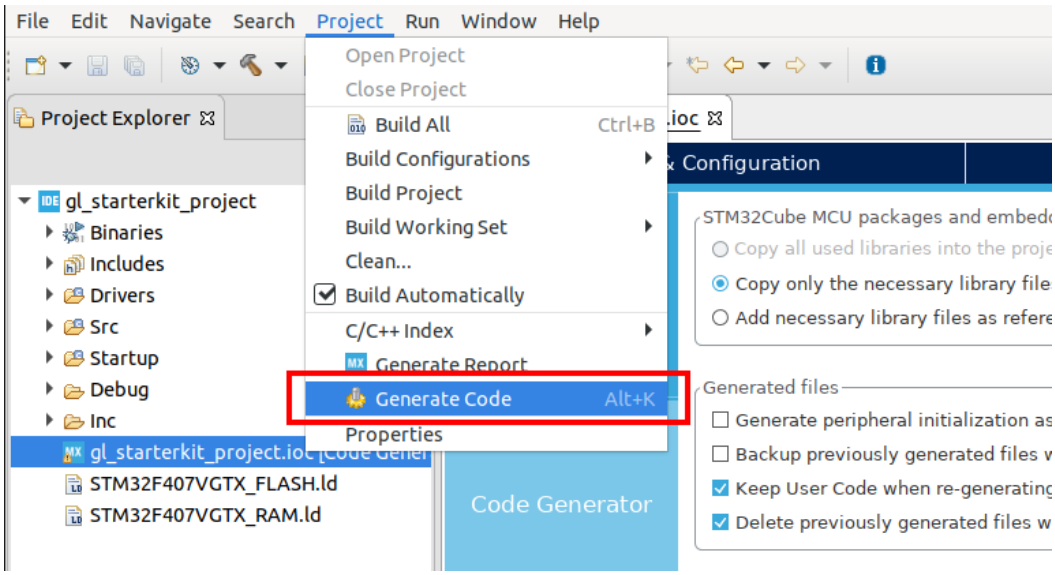



Остальный настройки можно оставить без изменений.

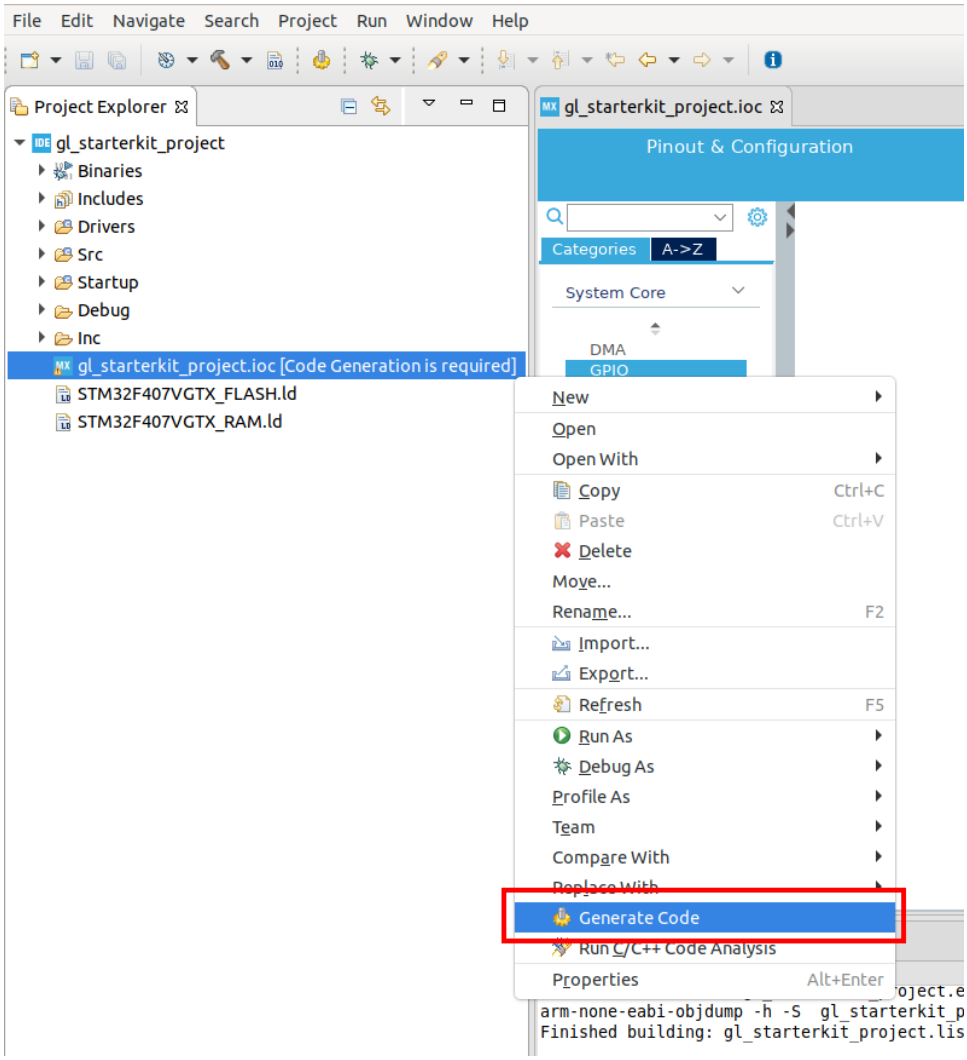
Теперь все готово для генерации кода. Для этого в *Device Configuration Tool* перейдем на вкладку *Project Manager* в *Code Generator* и поставим галочку напротив *Keep User Code when re-generating*



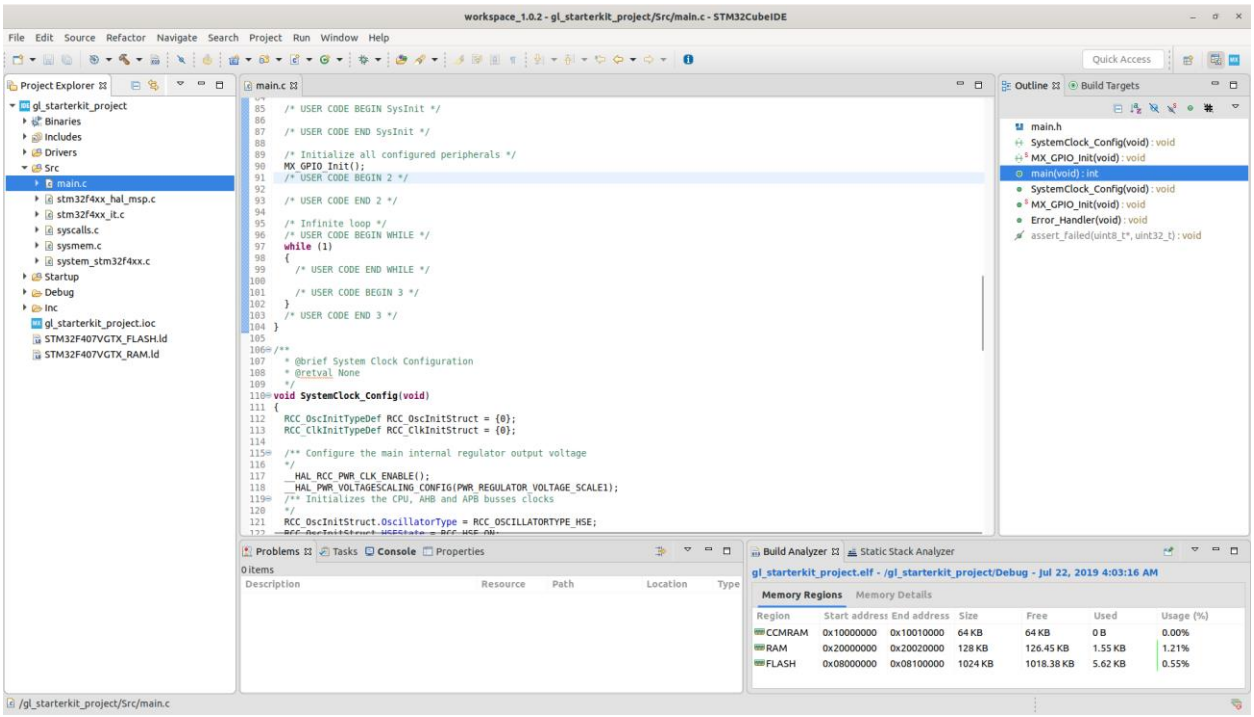

Остальные настройки - по своему усмотрению.
Для генерации кода перейти в *Project -> Generate Code (Alt+K)*




Либо в контекстном меню файла с расширением *.ioc* (*gl_starterkit_project.ioc*) выбрать *Generate Code*




В *Project Explorer* откроем файл *Src/main.c*




*Инициализация PD15* выполняется в функции *static void MX_GPIO_Init(void)*

main.c | main.h | stm32f4xx_hal_conf.h | stm32f4xx_hal_m

```
146 }
147
148 /**
149   * @brief GPIO Initialization Function
150   * @param None
151   * @retval None
152   */
153 static void MX_GPIO_Init(void)
154 {
155   GPIO_InitTypeDef GPIO_InitStruct = {0};
156
157   /* GPIO Ports Clock Enable */
158   __HAL_RCC_GPIOH_CLK_ENABLE();
159   __HAL_RCC_GPIOD_CLK_ENABLE();
160   __HAL_RCC_GPIOA_CLK_ENABLE();
161
162   /*Configure GPIO pin Output Level */
163   HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);
164
165   /*Configure GPIO pin : PD15 */
166   GPIO_InitStruct.Pin = GPIO_PIN_15;
167   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
168   GPIO_InitStruct.Pull = GPIO_NOPULL;
169   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
170   HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
171
172 }
```

Теперь в цикле *while* добавим код для мигания светодиодом

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* Toggle PD15 output state */
    HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
    HAL_Delay(500);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

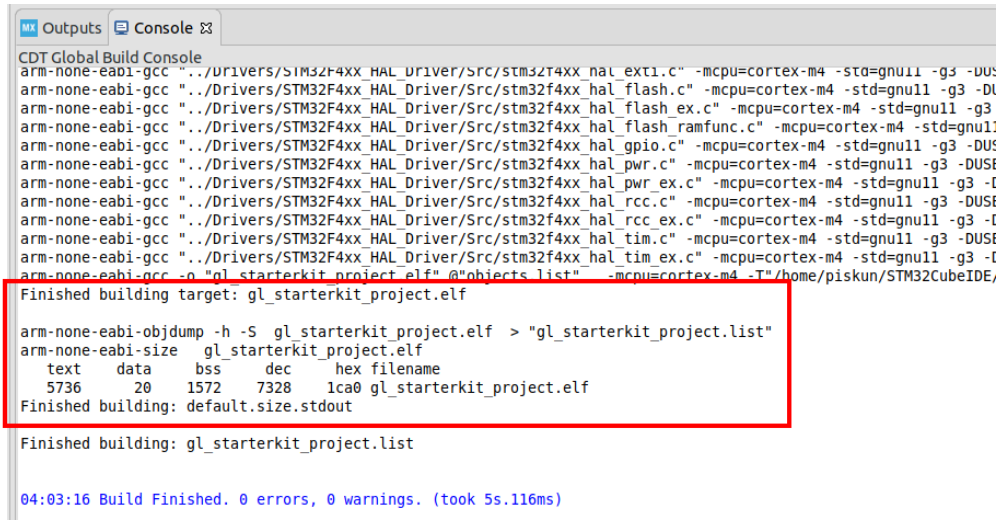
*HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);*
*HAL_Delay(500);*

**ВАЖНО: для того, чтобы написанный код сохранялся после перегенераций кода, он должен находится между подобными коментариями**

**/* USER CODE BEGIN 2 */**

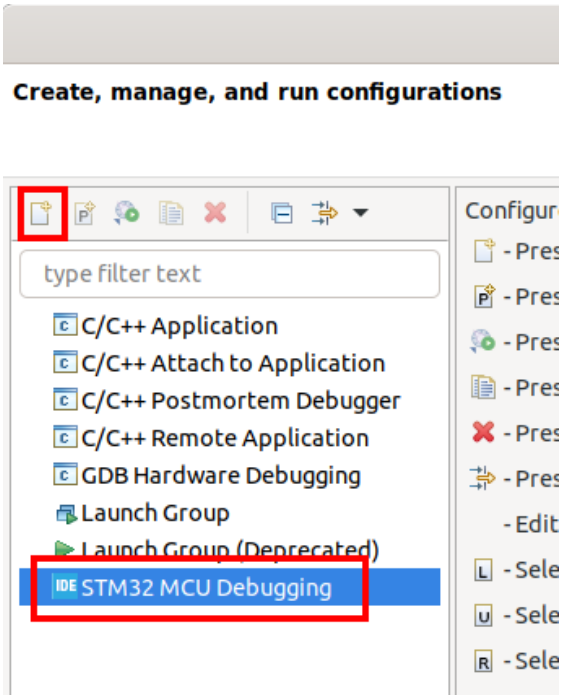
**/* USER CODE END 2 */**

Собираем проект (*Project -> Build Project*). Если проект собрался успешно, то в *Console* должен быть следующий вывод



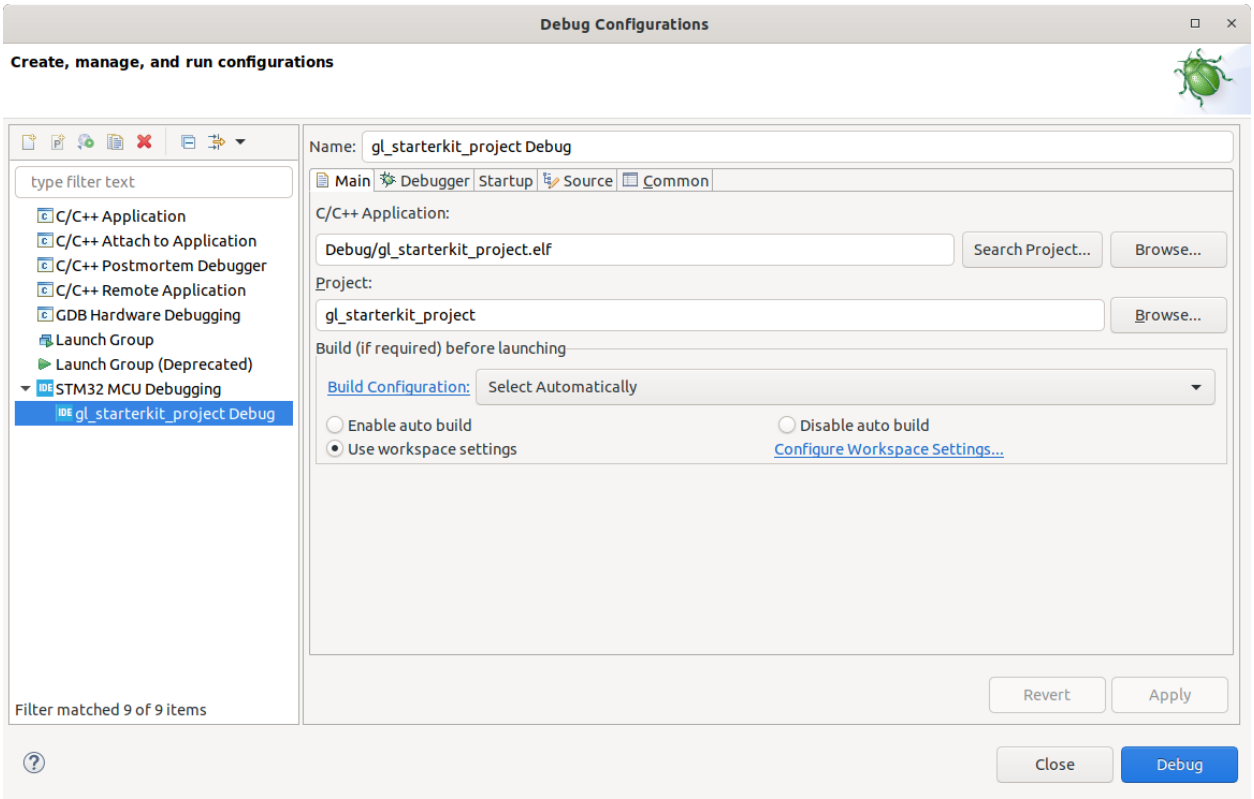

Создадим отладочную конфигурацию. Для этого перейдем в *Run -> Debug Configurations…*

В открывшемся окне выбрать *STM32 MCU Debugging* и нажать *New launch configuration*

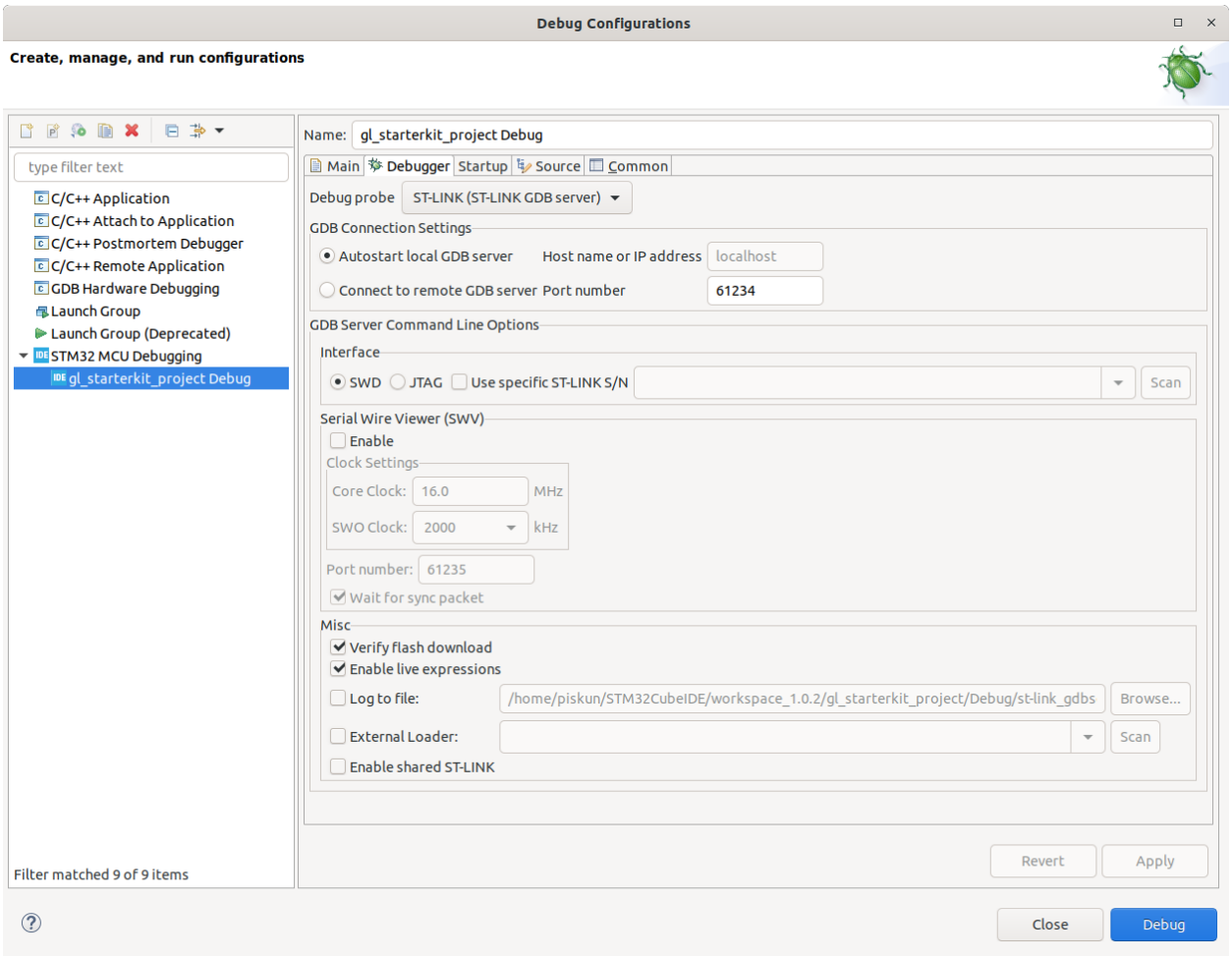



Отладочная конфигурация должна быть настроена автоматически

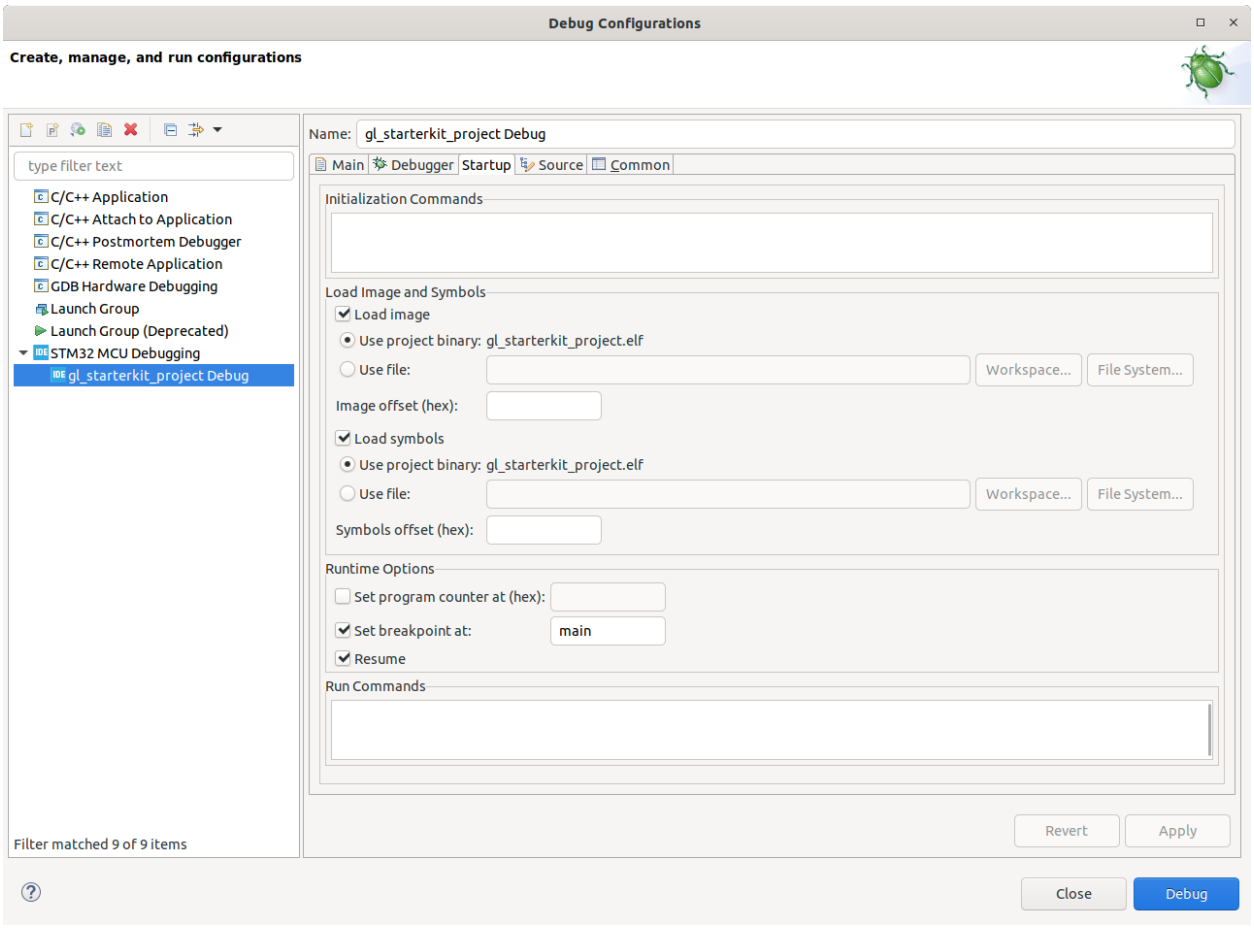
Вкладка *Main*



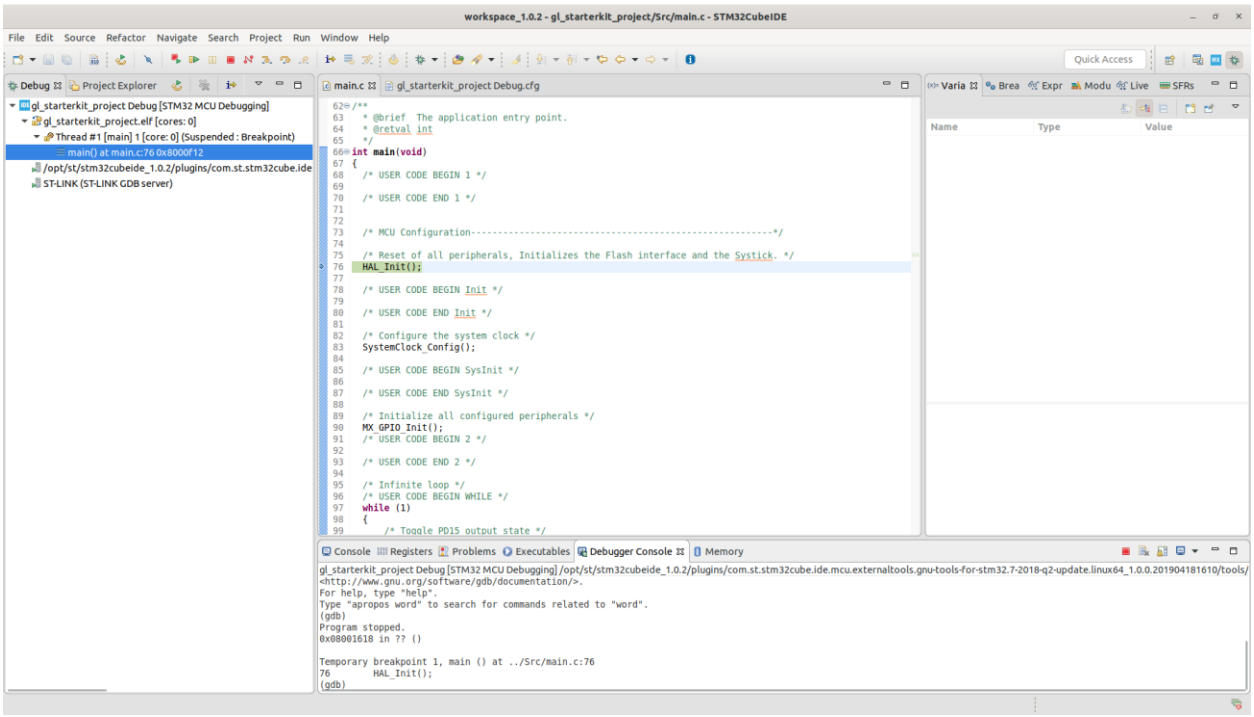

Вкладка *Debugger*




Вкладка *Startup*




Нажать кнопку *Debug*.
После старта отладочной сессии выполнение программы будет остановлено в начале функции *main()*.




F6/F5 - пошаговое выполнение программы;
F8 - продолжить выполнение. Выполнение будет остановлено на точке останова либо при нажатии кнопки “пауза” (*Suspend*).