RGB searchlight (GPIO)

This tutorial demonstrates: controlling the onboard RGB searchlight on the development board through the output function of **GPIO**.

The first tutorial in this chapter will be more detailed than the following tutorials. The purpose is to demonstrate from the new project to the complete effect, and guide users how to use STM32CubeIDE to develop

1. Software-Hardware

- **STM32F103CubeIDE**
- **STM32 Robot Development Board**

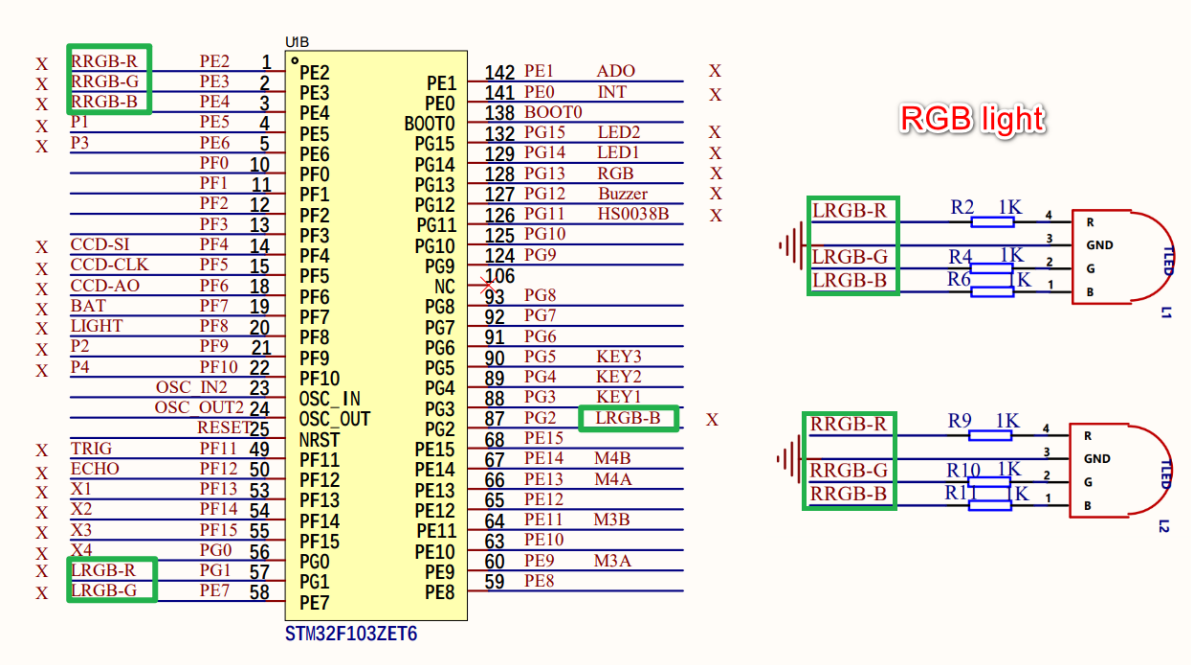
  GPIO: chip internal peripherals

  RGB searchlight: onboard
- **Type-C data cable or ST-Link**

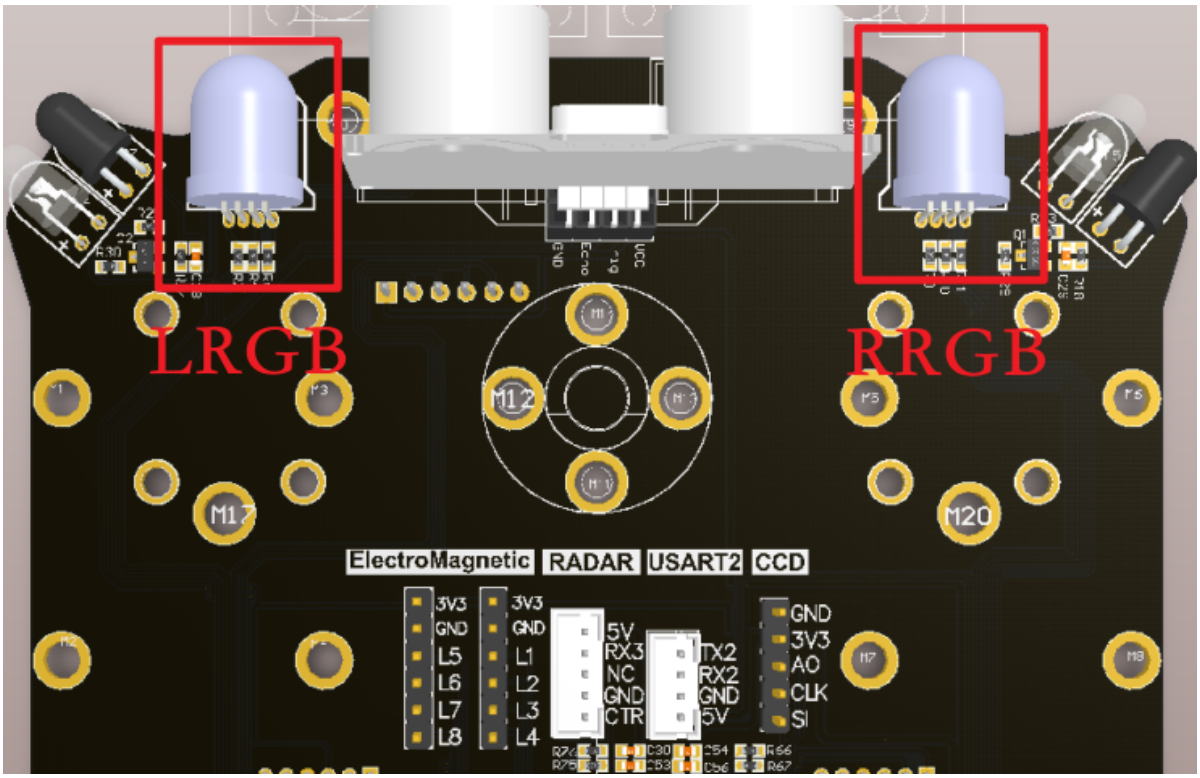
  Download programs or simulate the development board

2. Brief principle

1. Hardware schematic diagram



2. Physical connection diagram



3. Control principle

By controlling the high and low levels of the RGB light pins, the color displayed by the RGB light is controlled.

RGB: High level lights up, low level turns off

RGB（原理图名称）	控制引脚	功能
LRGB-R	PG1	Control the red light display of the left RGB light
LRGB-G	PE7	Control the green light display of the left RGB light
LRGB-B	PG2	Control the blue light display of the left RGB light
RRGB-R	PE2	Control the red light display of the RGB light on the right
RRGB-G	PE3	Control the green light display of the right RGB light
RRGB-B	PE4	Control the blue light display of the right RGB light

```
RGB lights can display other colors by combining different brightnesses of the three basic colors of red, green, and blue.
```

3. Engineering experience

You can use the project files we provide to directly experience the corresponding functions of the development board.

Later tutorials do not provide this content to avoid duplication of content. You can go to [2. Development environment construction and use: engineering experience and transplantation] to view the operation

Open project

- **Project file location**

  Project file path: Under the [Project Source Code] folder of the Chapter 3 tutorial

- **Open project file**

  Copy the project file to the directory of **English path**, use STM32CubeIDE to open the project file, open the project file and select the **.project** file


```
Tip: You can create a new STM32 folder on the desktop and select the STM32 folder
as the workspace when opening the project file.
```

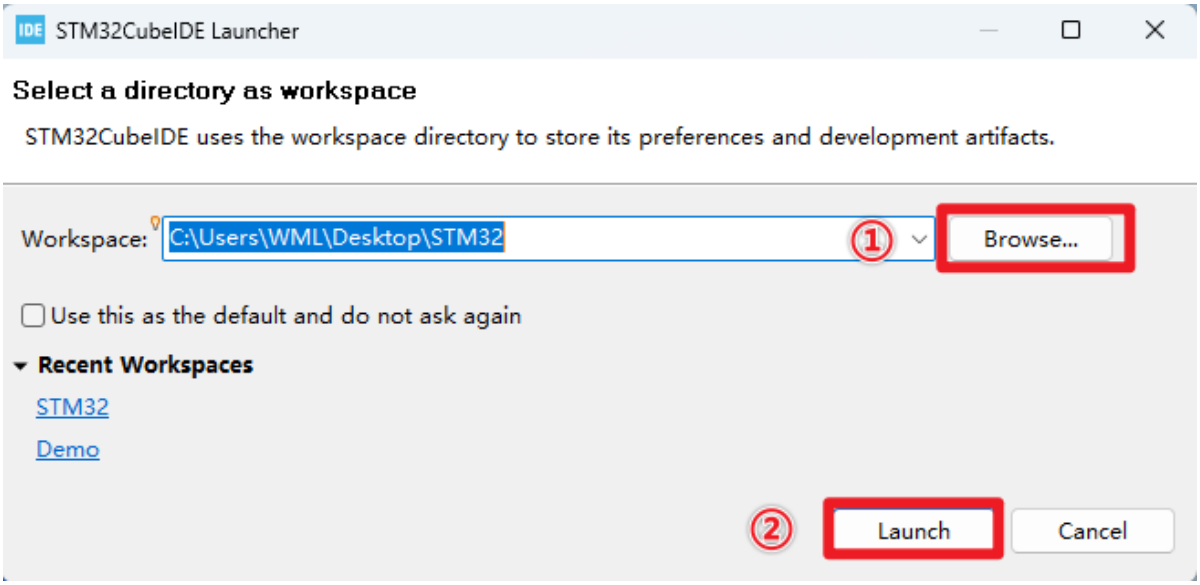
4. Project configuration

This tutorial will completely demonstrate the configuration process. Later, the content of **new project, chip selection, project settings, pin settings of SYS, RCC configuration, clock configuration and project configuration** will be omitted. Any changes will be stated in the tutorial. .

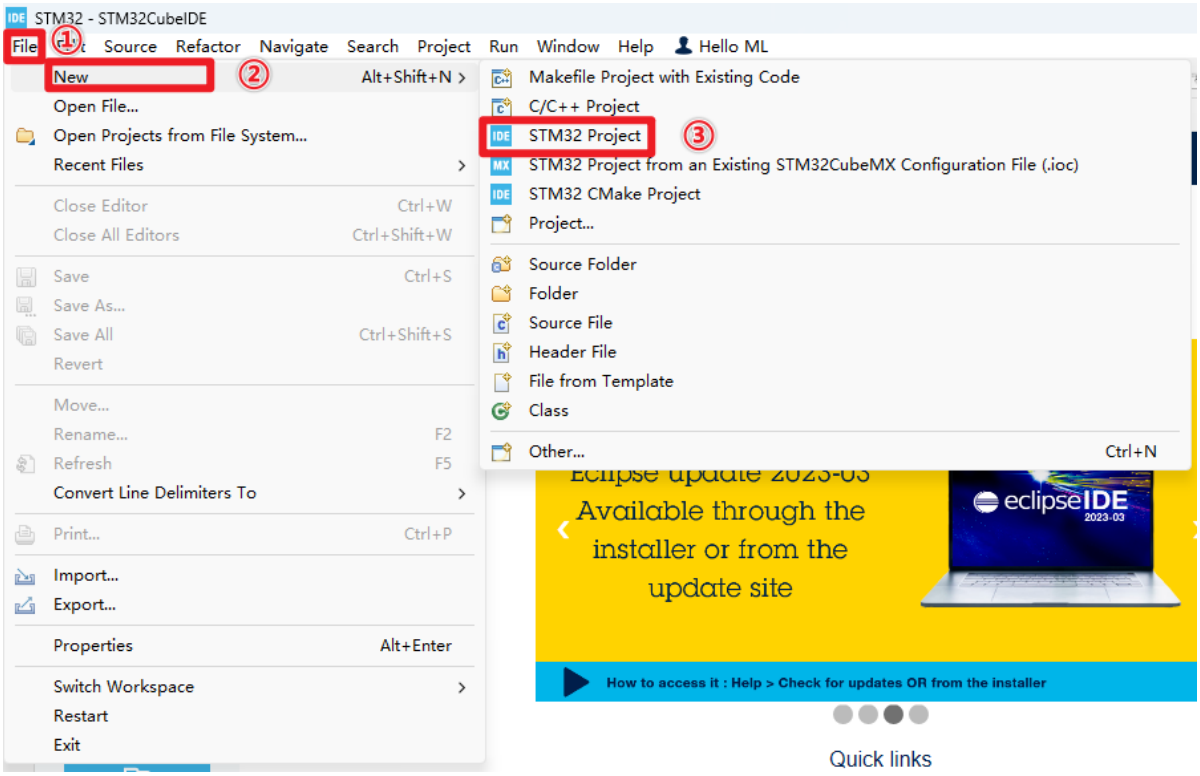
1. New project

- **Select workspace**

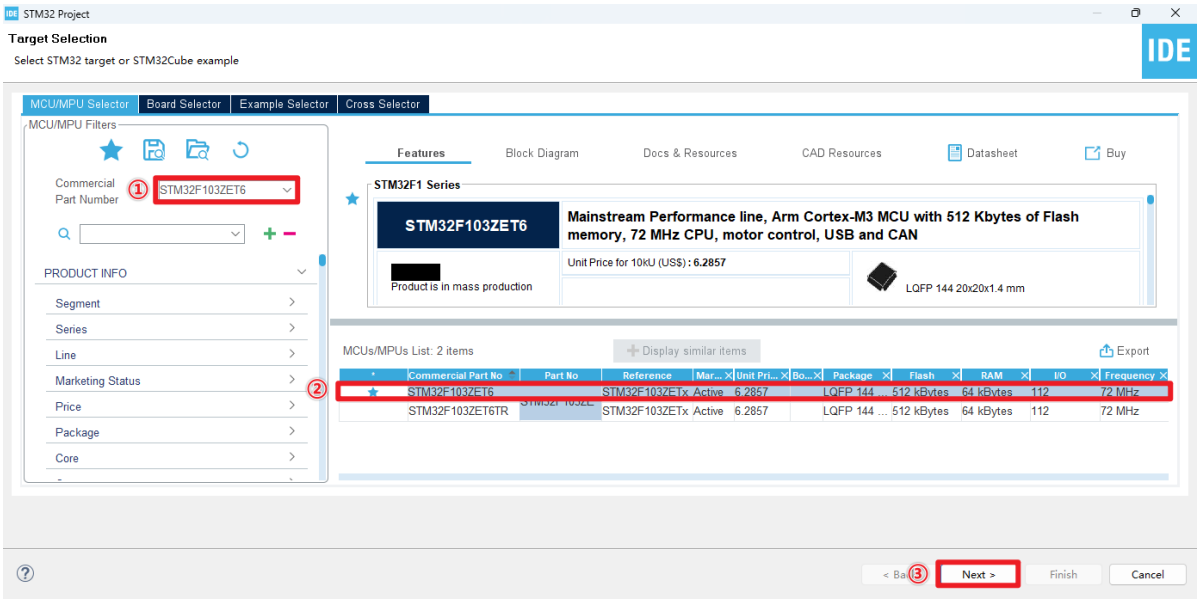
  The path of a new project or the path of an existing project file: the path cannot contain Chinese characters



- **New Project**

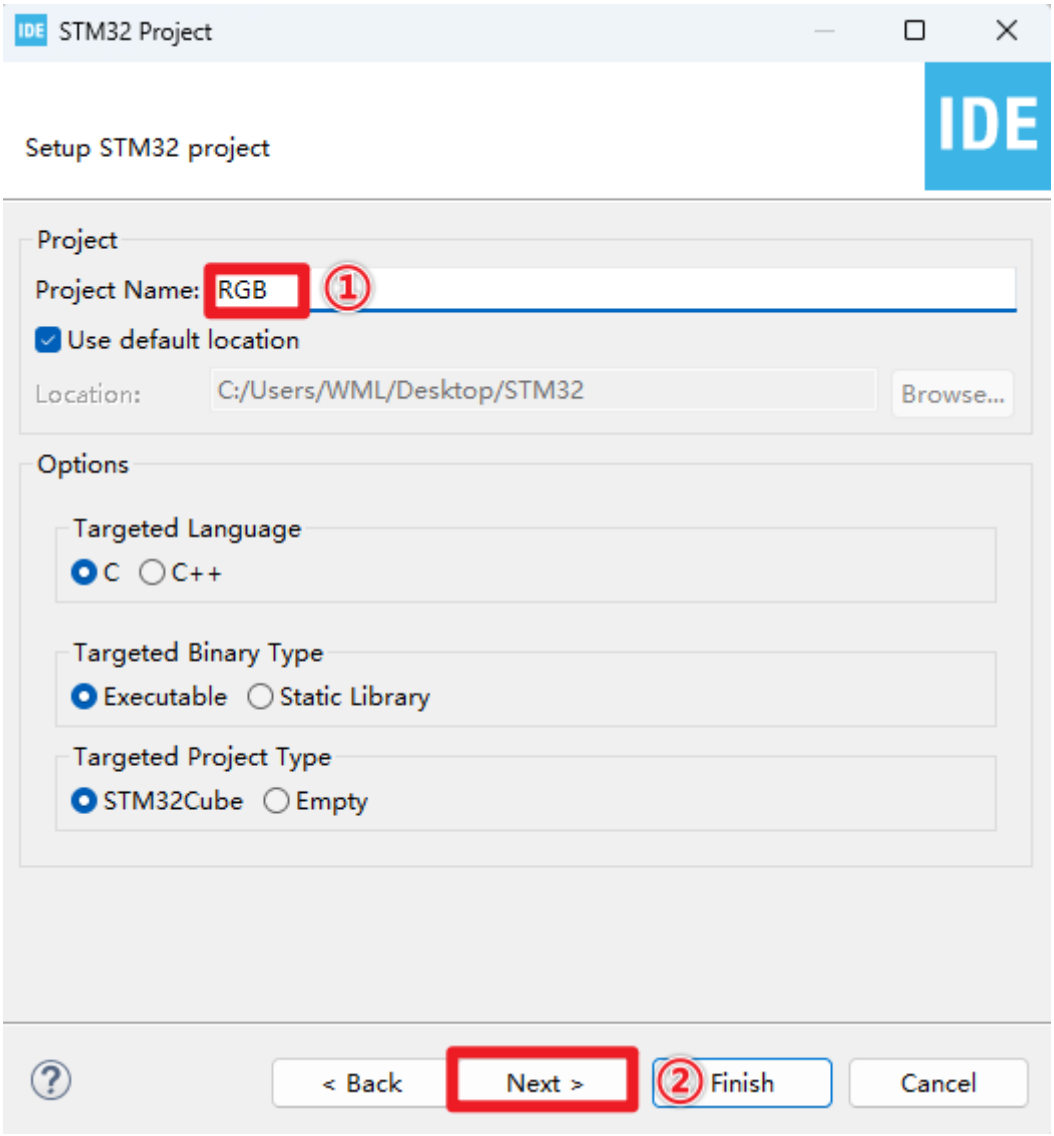


2. chip selection

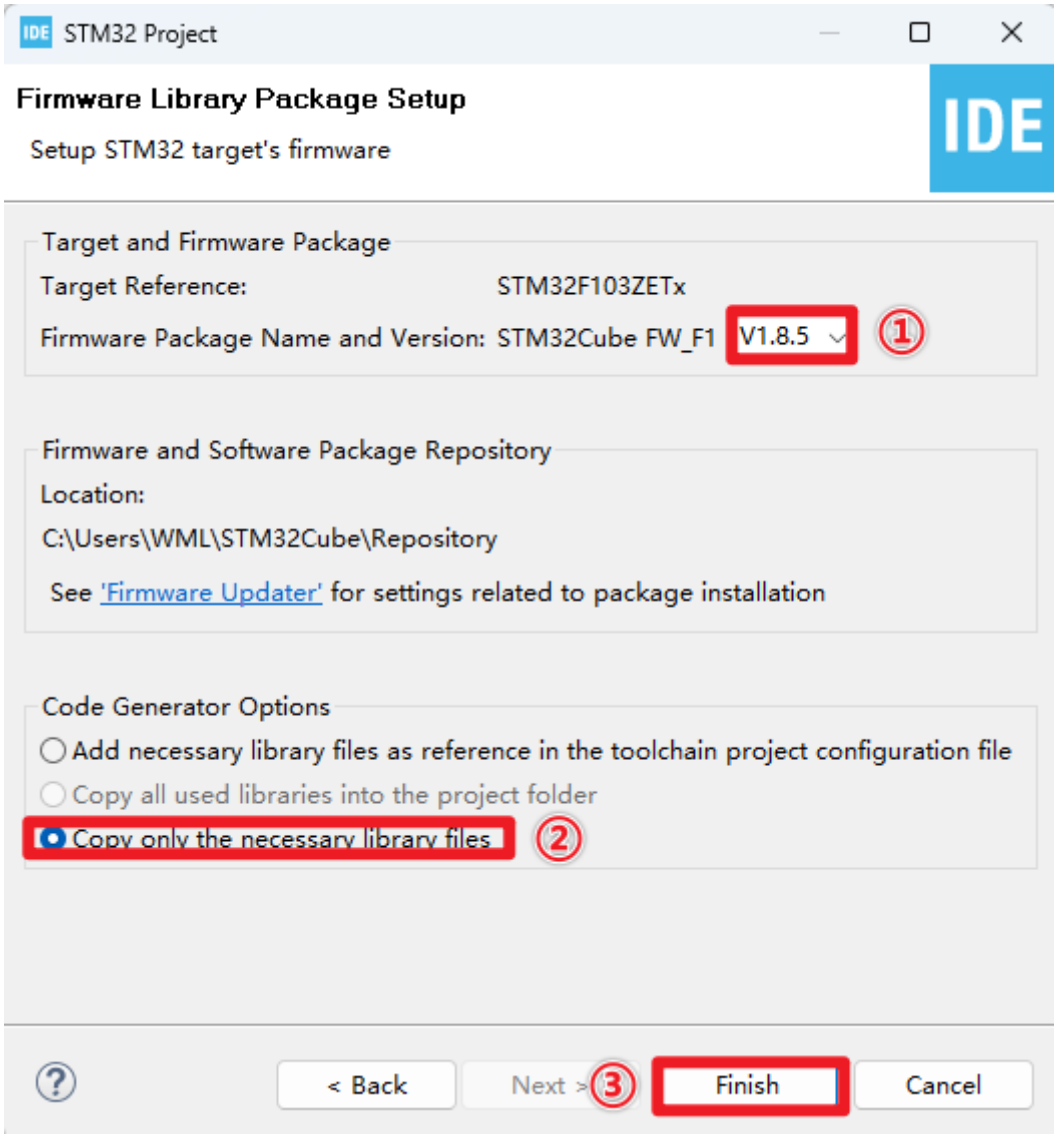


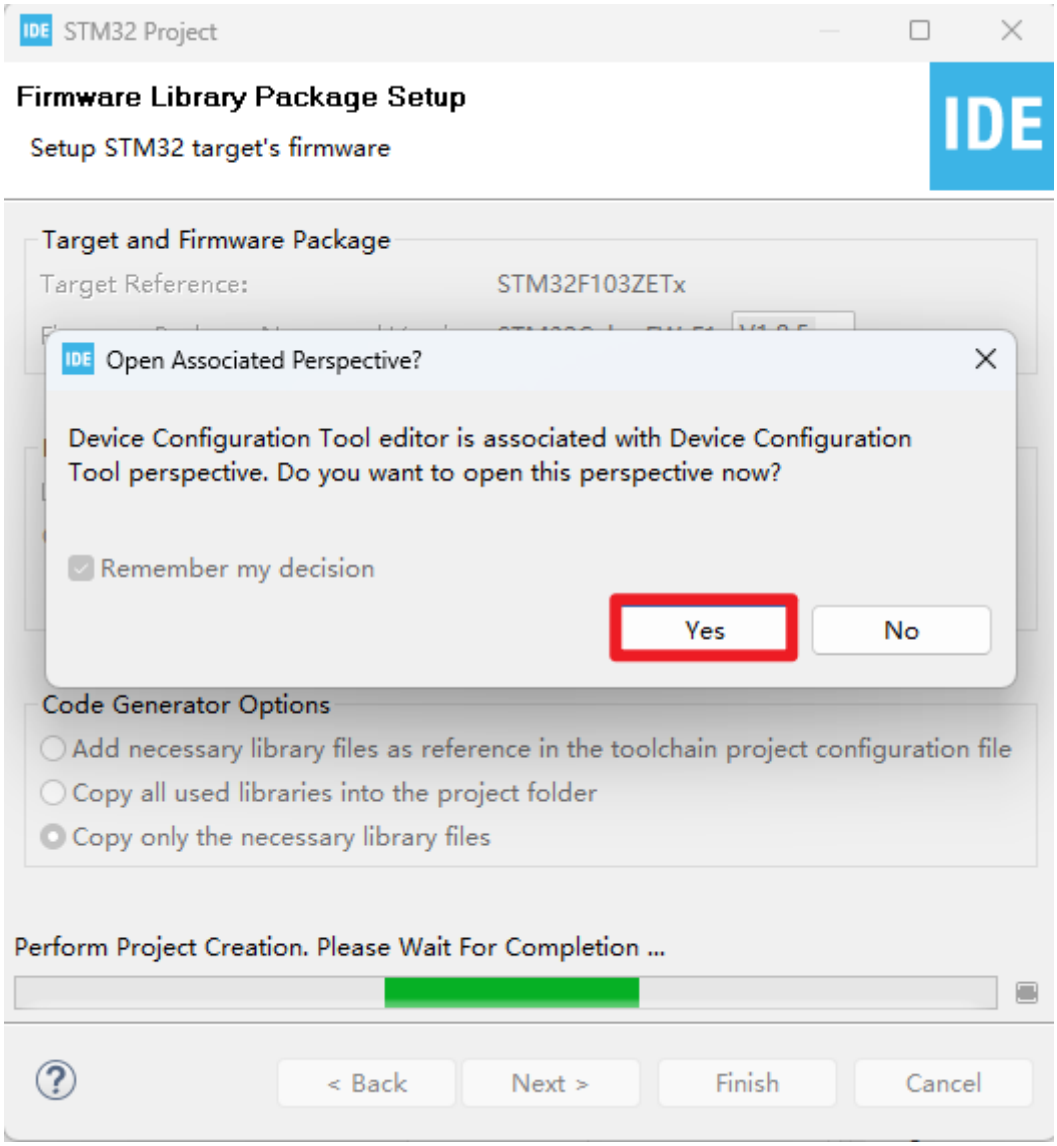
3. Project settings

- **project name**



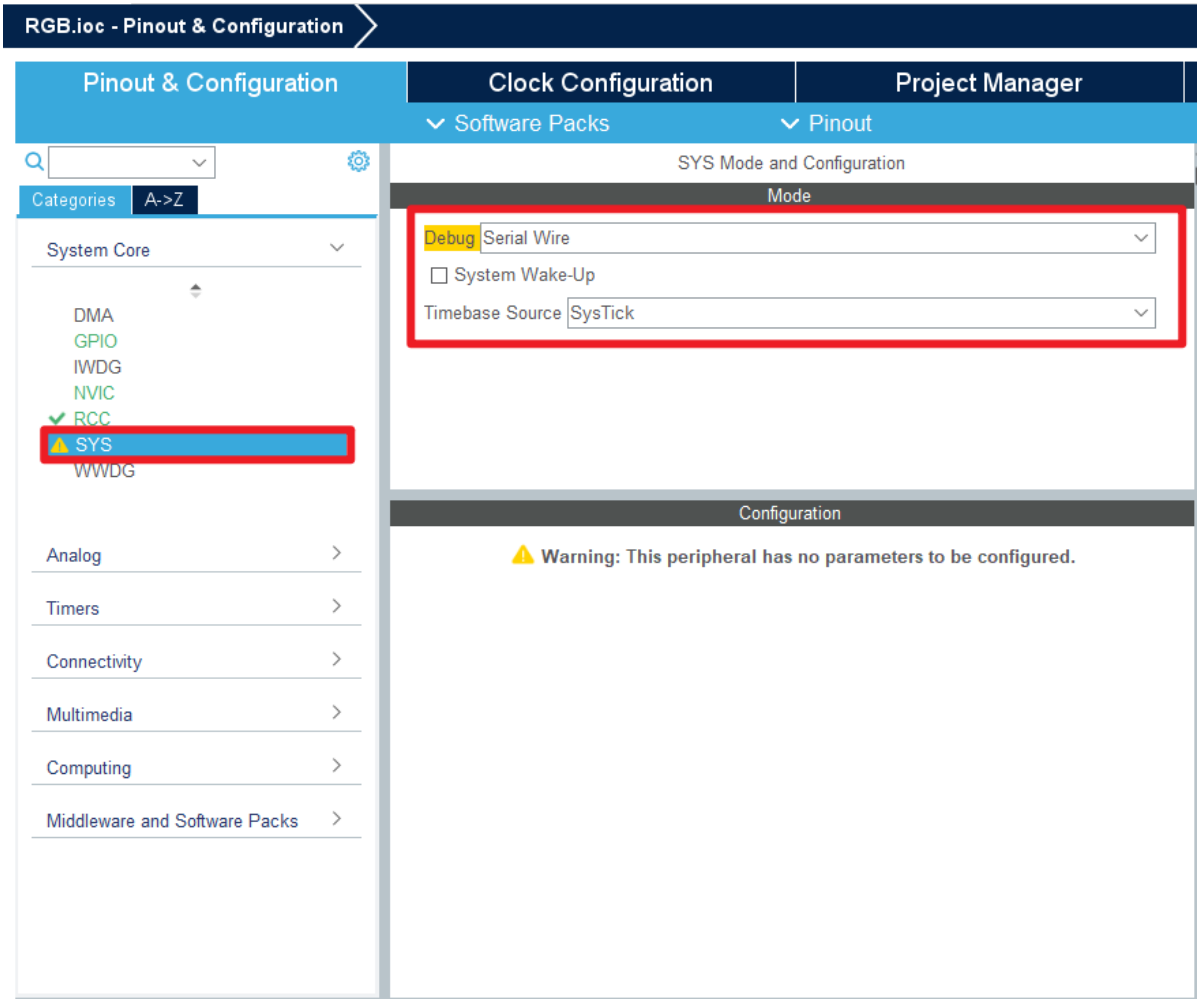
- **Firmware version**


STM32 Project
Firmware Library Package Setup
Setup STM32 target's firmware
IDE
Target and Firmware Package
Target Reference:
STM32F103ZETx
Firmware Package Name and Version:
STM32Cube FW_F1
V1.8.5
1
Firmware and Software Package Repository
Location:
C:\Users\WML\STM32Cube\Repository
See 'Firmware Updater' for settings related to package installation
Code Generator Options
Add necessary library files as reference in the toolchain project configuration file
Copy all used libraries into the project folder
Copy only the necessary library files
2
< Back
Next >
3
Finish
Cancel

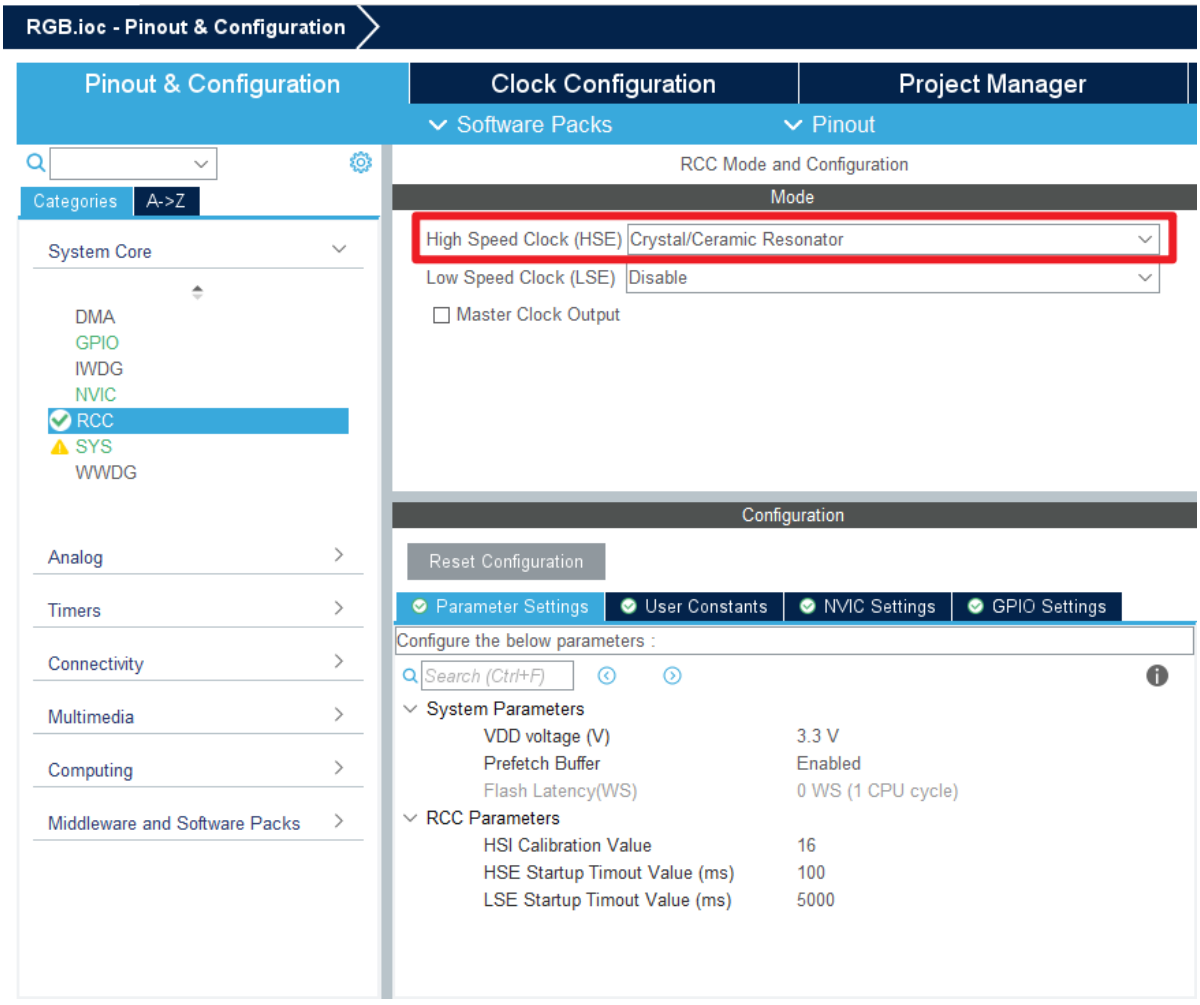


4. Pin configuration

- **SYS**

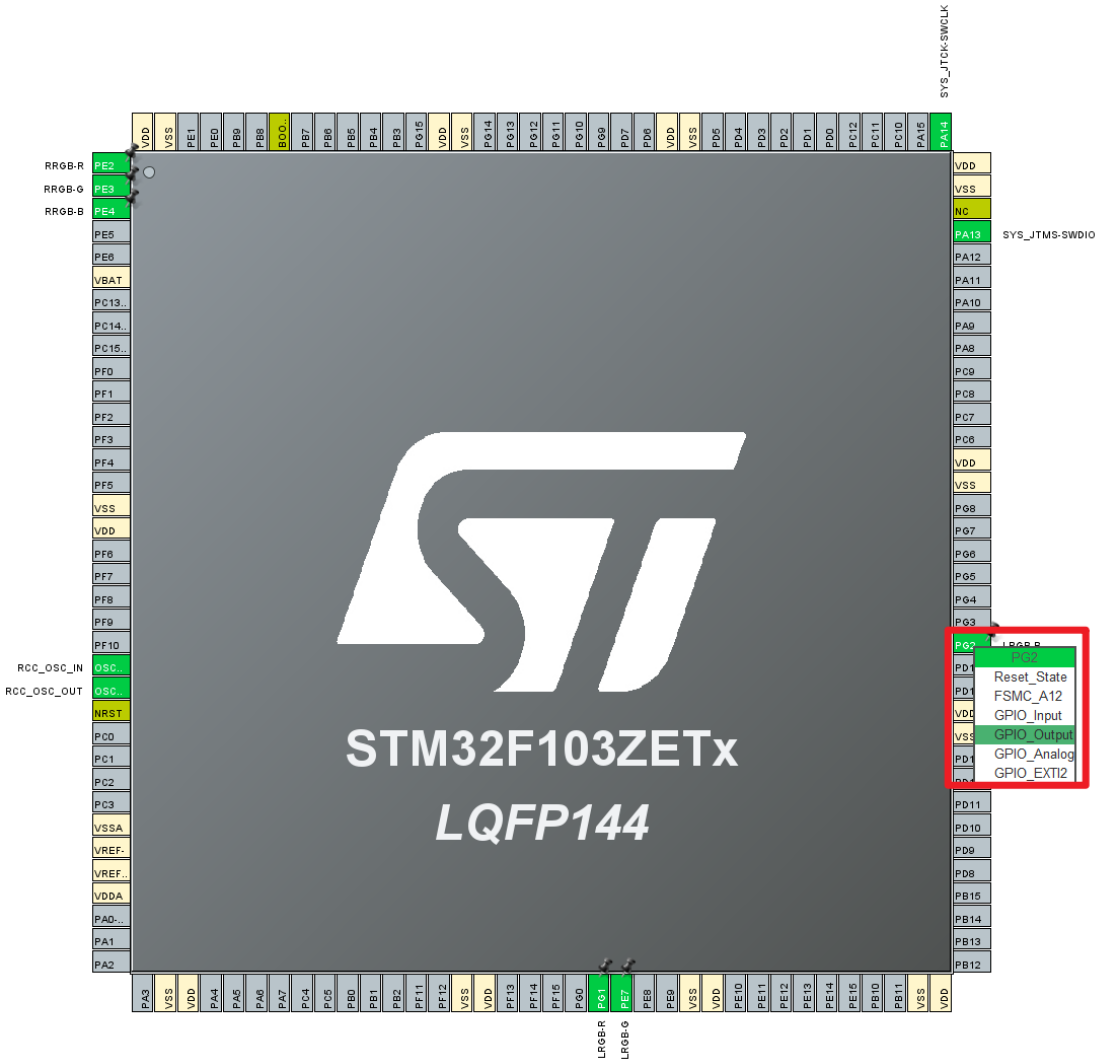

RGB.ioc - Pinout & Configuration
Pinout & Configuration
Clock Configuration
Project Manager
Software Packs
Pinout
Categories
A->Z
System Core
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
Timers
Connectivity
Multimedia
Computing
Middleware and Software Packs
SYS Mode and Configuration
Mode
Debug
Serial Wire
System Wake-Up
Timebase Source
SysTick
Configuration
Warning: This peripheral has no parameters to be configured.

- **RCC**

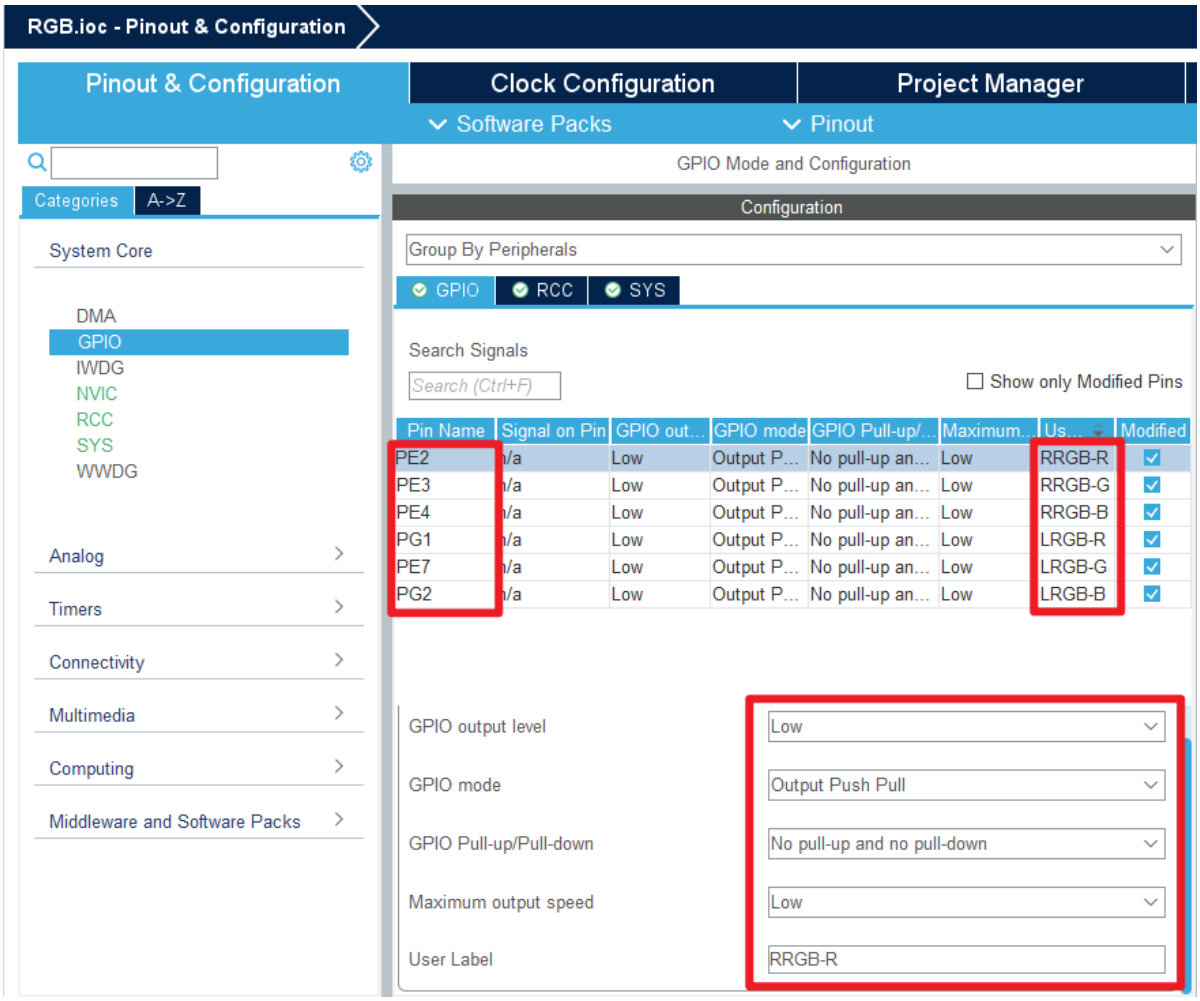

RGB.ioc - Pinout & Configuration
Pinout & Configuration
Clock Configuration
Project Manager
Software Packs
Pinout
Categories
A->Z
System Core
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
Timers
Connectivity
Multimedia
Computing
Middleware and Software Packs
RCC Mode and Configuration
Mode
High Speed Clock (HSE)
Crystal/Ceramic Resonator
Low Speed Clock (LSE)
Disable
Master Clock Output
Configuration
Reset Configuration
Parameter Settings
User Constants
NVIC Settings
GPIO Settings
Configure the below parameters :
Search (Ctrl+F)
System Parameters
VDD voltage (V)
3.3 V
Prefetch Buffer
Enabled
Flash Latency(WS)
0 WS (1 CPU cycle)
RCC Parameters
HSI Calibration Value
16
HSE Startup Timout Value (ms)
100
LSE Startup Timout Value (ms)
5000

- **Configure specified pin function**

You can directly select the corresponding pin number in the pin view, and the corresponding options will appear when you left-click the mouse.

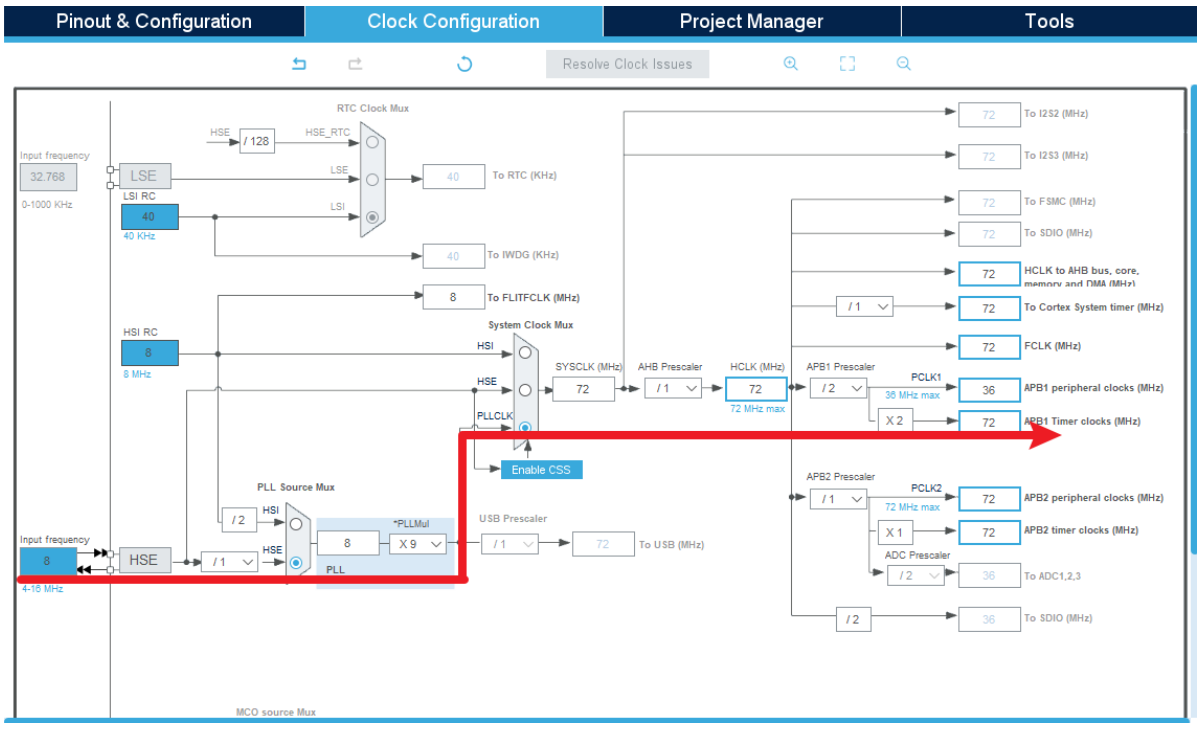


- **GPIO**



Please refer to the figure above for specific configuration options. It is recommended to add tags. STM32CubeIDE will generate corresponding macro definitions.

5. Clock configuration

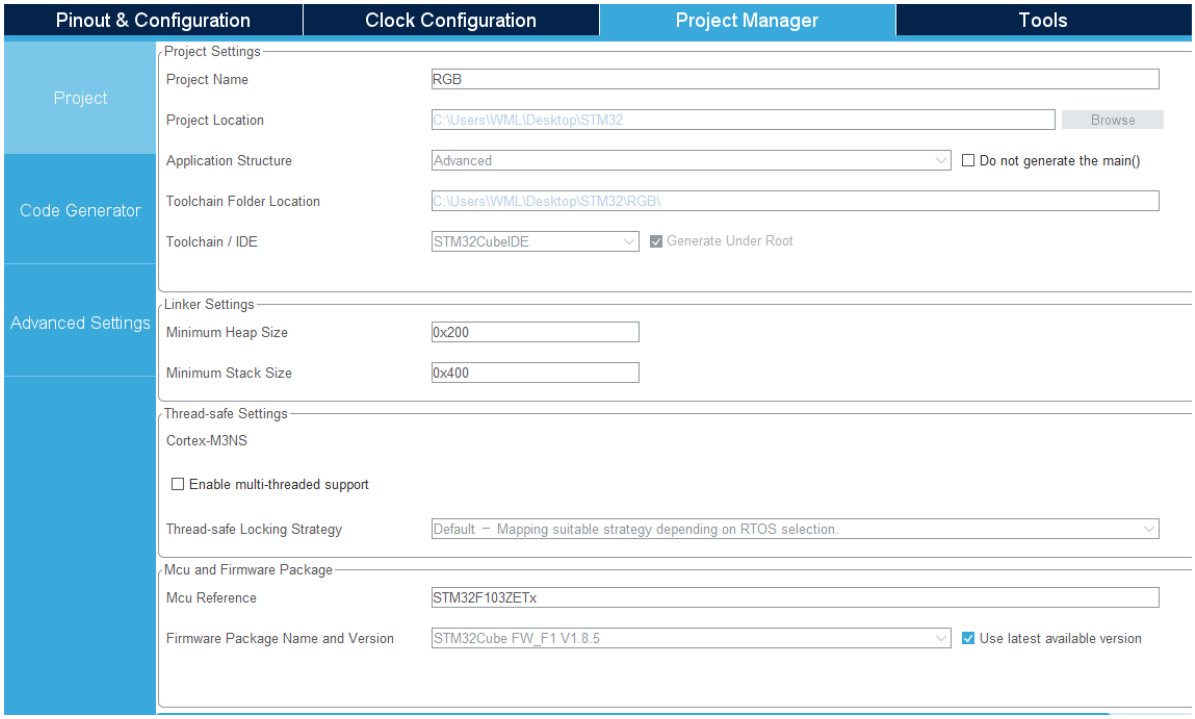


Refer to the options covered by the red arrows

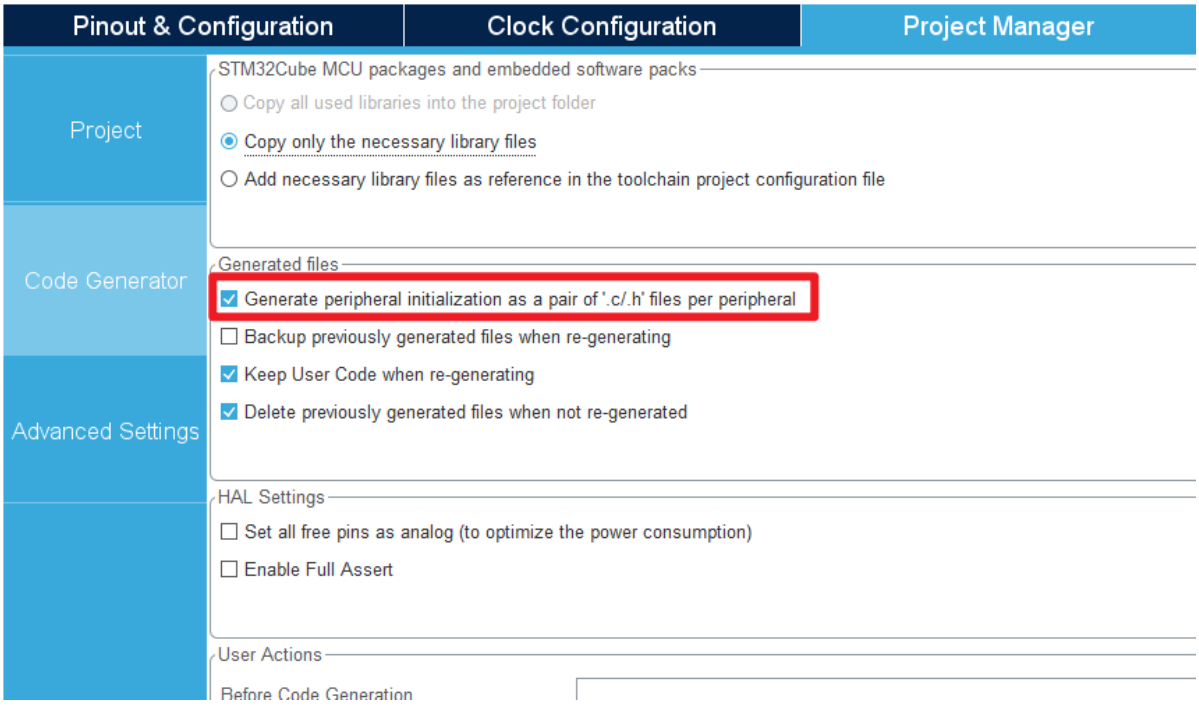
6. Project configuration

- **Project**

  do not need to change

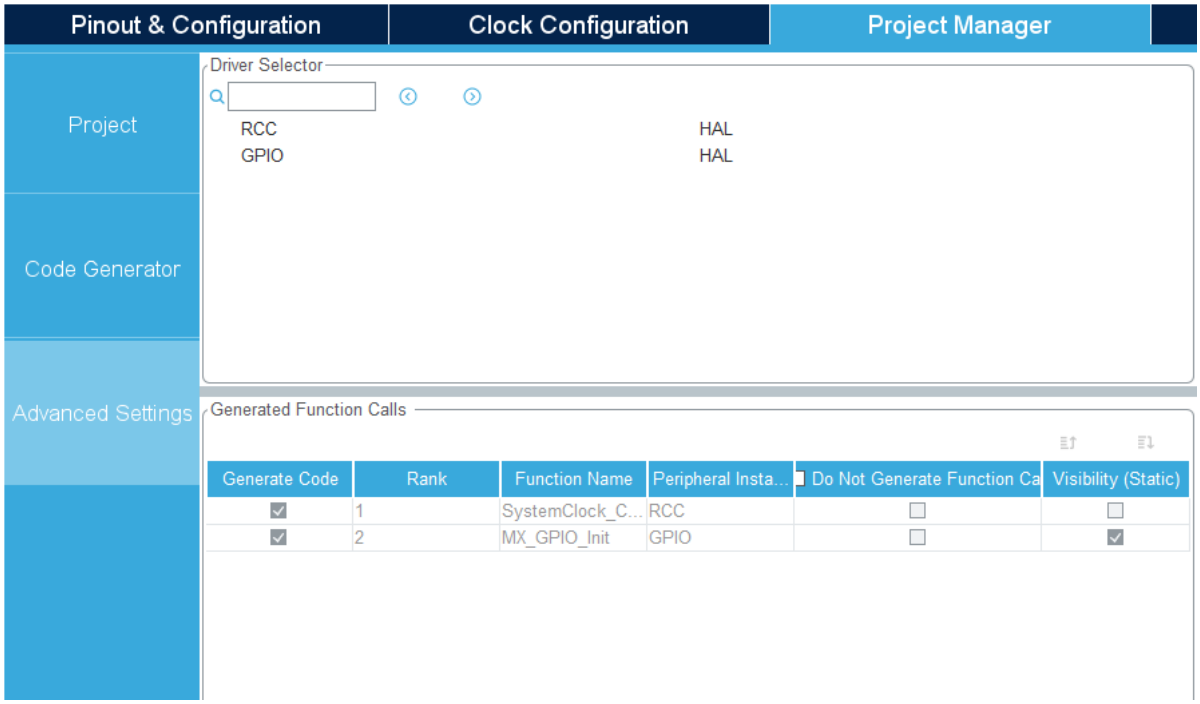


- **Code Generator**



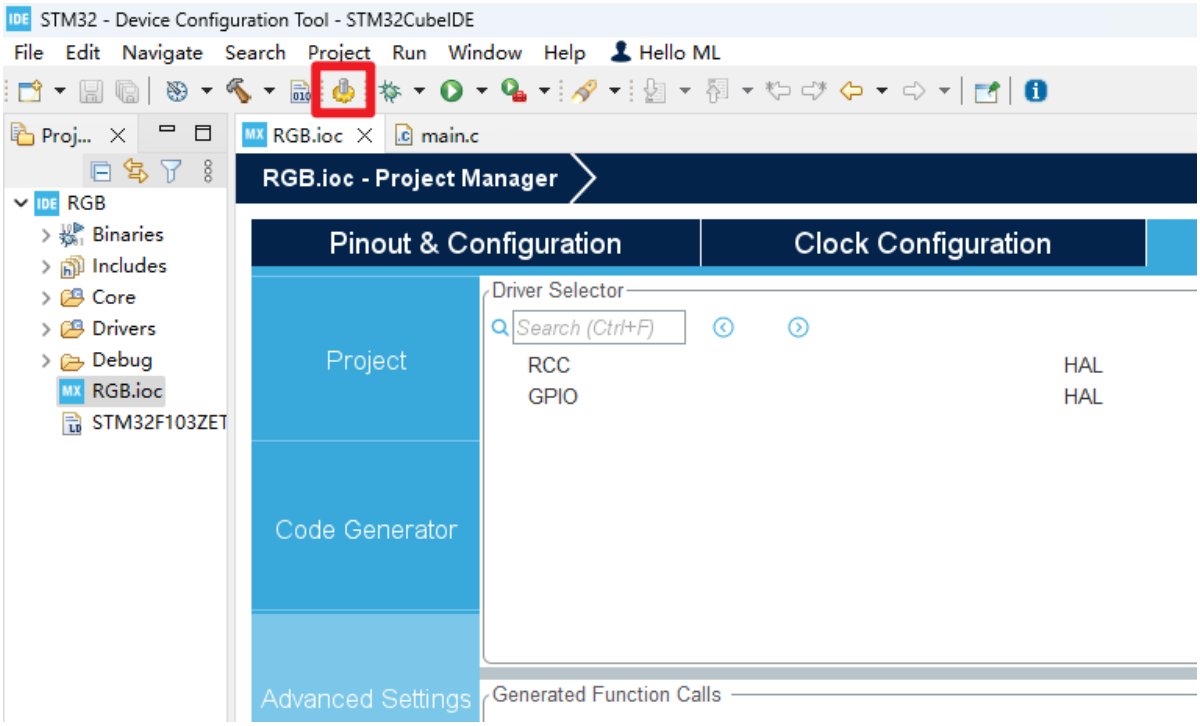
- **Advanced Settings**

  do not need to change



7. Generate code

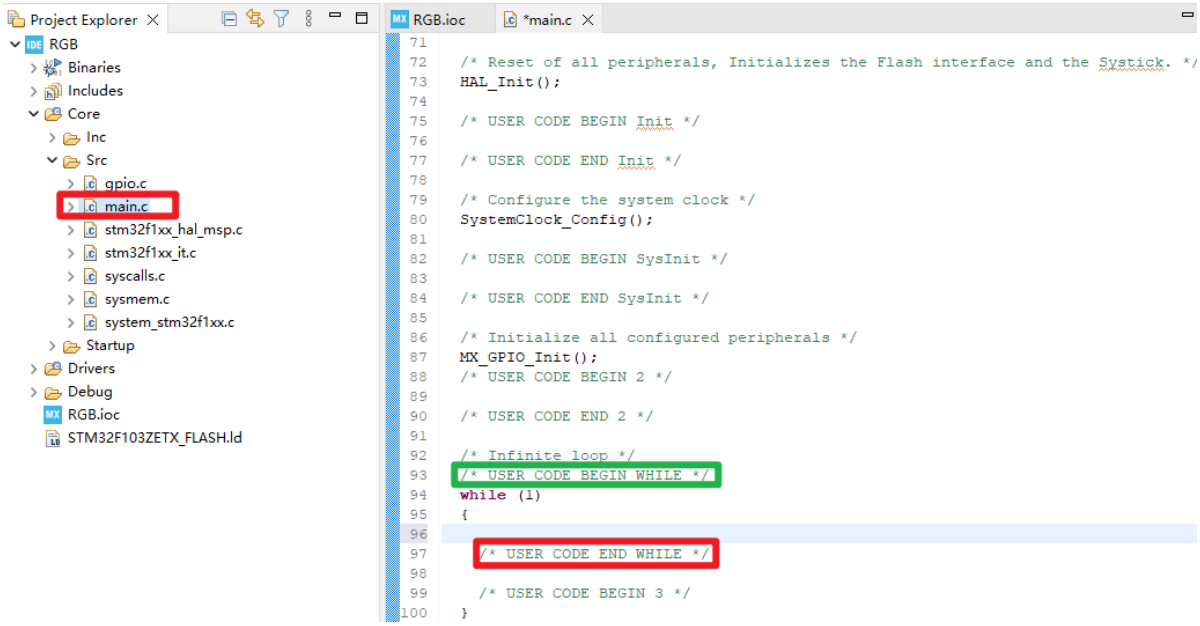
- **Click on the "pinion" icon**




```
Click here to save or the Ctrl+C shortcut key to generate code.
```

- **Edit code**

  The user code must be located between USER CODE BEGIN and USER CODE END.



The above is the peripheral configuration and initialization code generation.

5. Main functions

It mainly introduces the functional code written by the user. For detailed code, you can open the project file provided by us yourself and enter the Bsp folder to view the source code. **

1. User function

> **function：BSP_Init**

function prototype	**void BSP_Init(void)**
Function description	Initialize the underlying/peripheral driver
Input parameters	none
return value	none

> **function：BSP_Loop**

function prototype	**void BSP_Loop(void)**
Function description	Low-level/peripheral loop functions
Input parameters	none
return value	none

> **function：Set_color_R**

function prototype	**void Set_color_R(RGB_Color color)**
Function description	Demonstration of setting up the RGB light display on the right
Input parameters	**color**: Set RGB light color
return value	none

> **function：Set_color_L**

function prototype	void Set_color_L(RGB_Color color)
Function description	Demonstration of setting up the RGB light display on the left
Input parameters	**color**: Set RGB light color
return value	none

> **function：Set_RGB**

function prototype	void Set_RGB(car_RGB light,RGB_Color color)
Function description	Set the color displayed by the left and right RGB lights
Input parameters 1	**light**: Set the displayed RGB light
Input parameters 2	**color**: Set RGB light color
return value	none

2. HAL library function parsing

Since using STM32CubeIDE will automatically generate initialization code, only the HAL library functions involved in this tutorial are introduced here.

```
If you want to find the HAL library and LL library function analysis involved in the entire tutorial, you can view the documents in the folder [8. STM32 Manual: STM32F1_HAL Library and LL Library_User Manual]
```

> **function：HAL_GPIO_Init**

function prototype	void HAL_GPIO_Init(GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_Init)
Function description	**Initialize GPIO pin parameters**
Input parameters 1	**GPIOx**: Set the GPIO port, x takes the value A, B, C, D, E, F, G
Input parameters 2	**GPIO_Init**: GPIO initialization structure
return value	none

> **function：HAL_GPIO_WritePin**

function prototype	void HAL_GPIO_WritePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
Function description	**Set/Clear the specified data port bit**

function prototype	void HAL_GPIO_WritePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
Input parameters 1	**GPIOx**: Set the GPIO port, x takes the value A, B, C, D, E, F, G
Input parameters 2	**GPIO_Pin**: Set GPIO pin, x value is 0-15
Input parameters 3	**PinState**: Bit_RESET: clear the data port bit (low level); Bit_SET: set the data port bit (high level)
return value	none

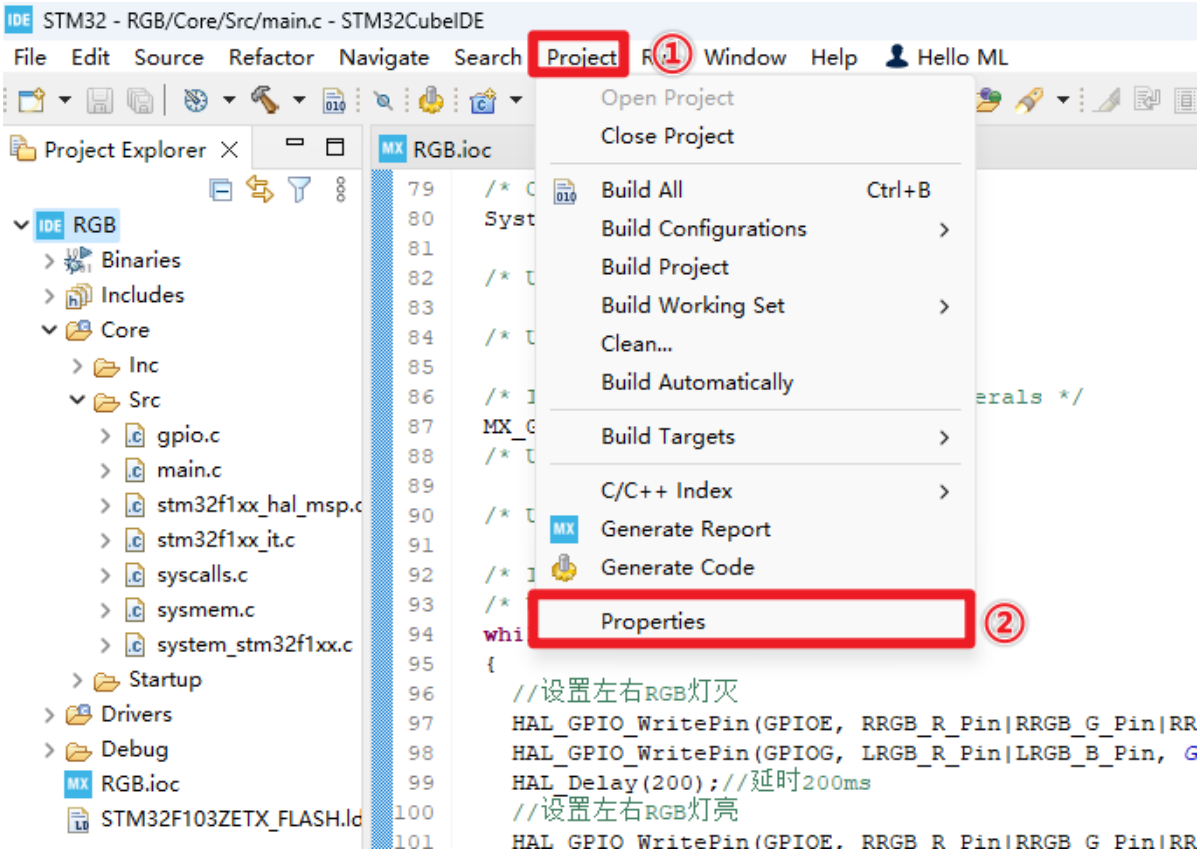
> **function: HAL_Delay**

function prototype	void HAL_Delay(uint32_t Delay)
Function description	Delay for a certain period of time
Input parameters	**Delay**: Set the delay in milliseconds
return value	none

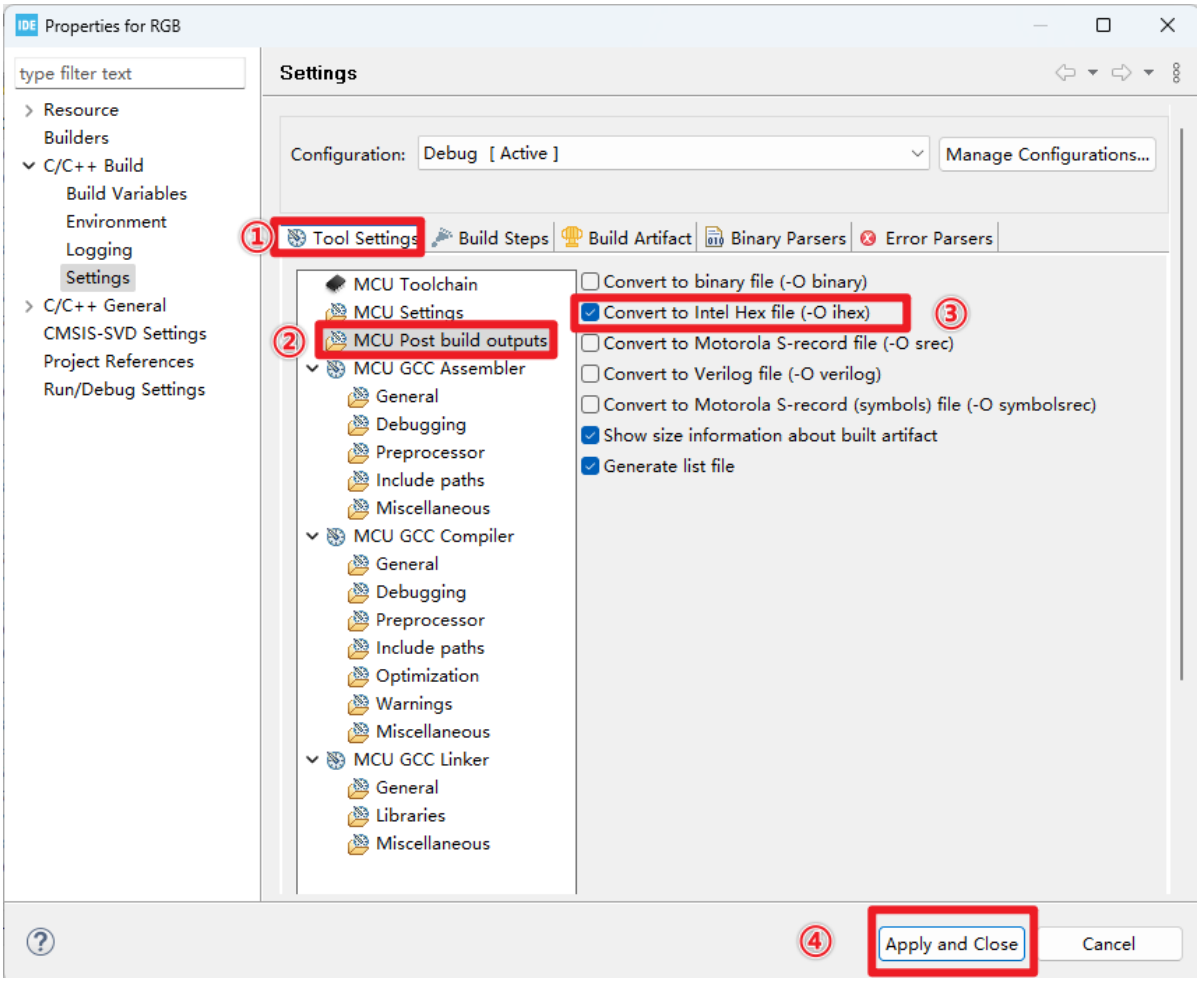
6. Program download

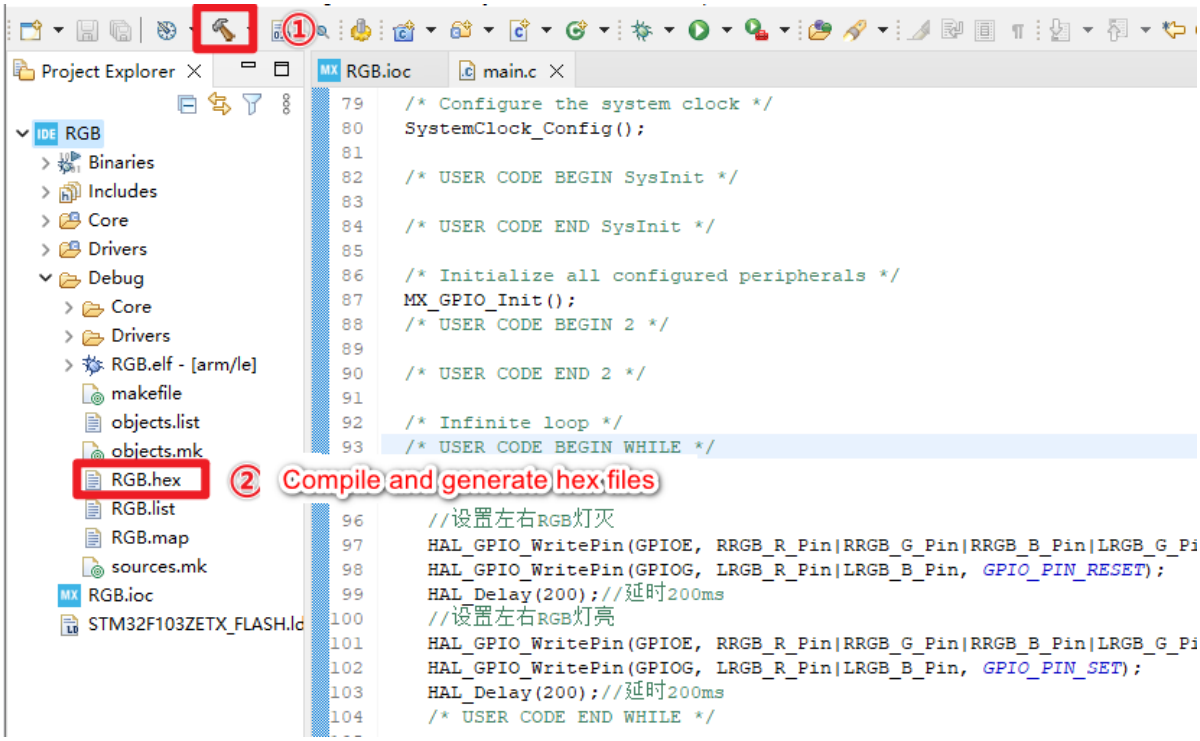
1. Serial port download

- **Generate .hex file**: Left-click the project → select "Properties"



- **Check the corresponding option**

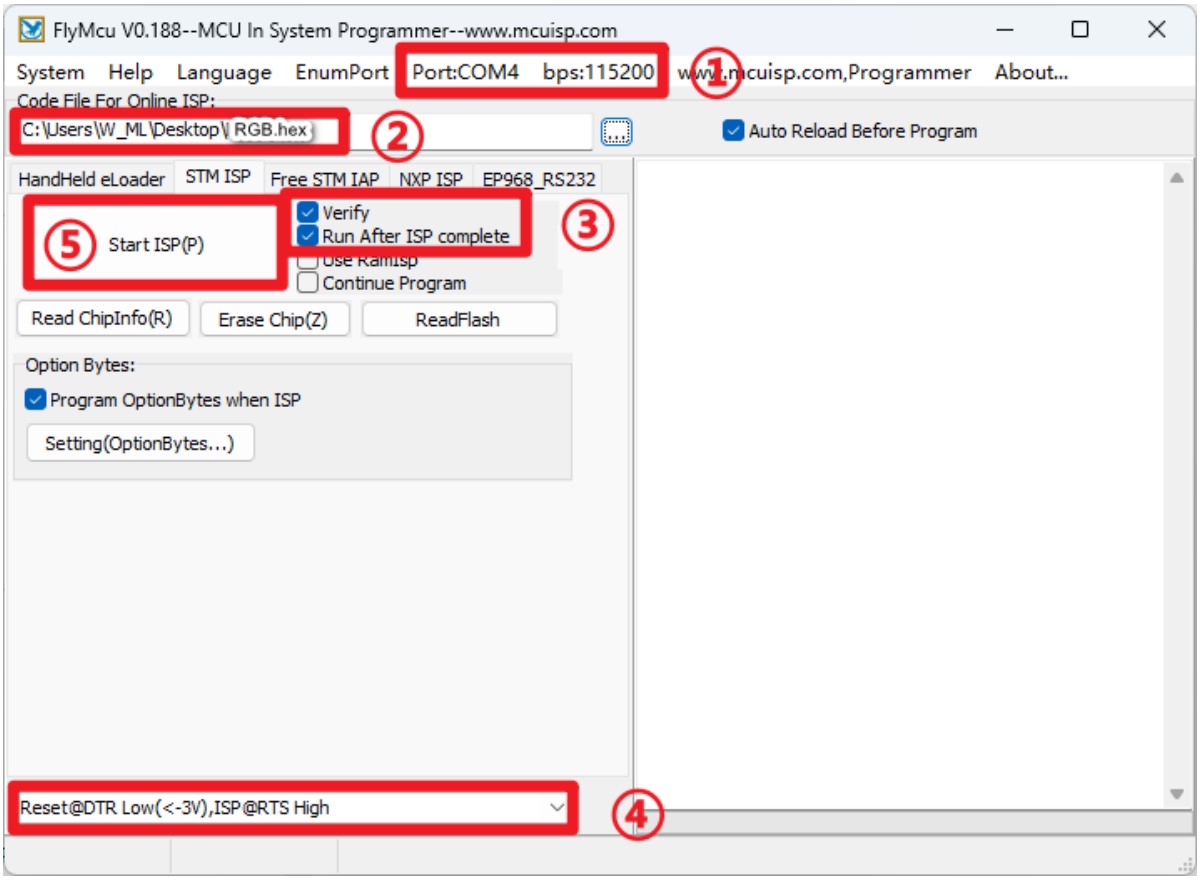





```
hex file location: under the Debug folder of the project file
```

- **Program Download**

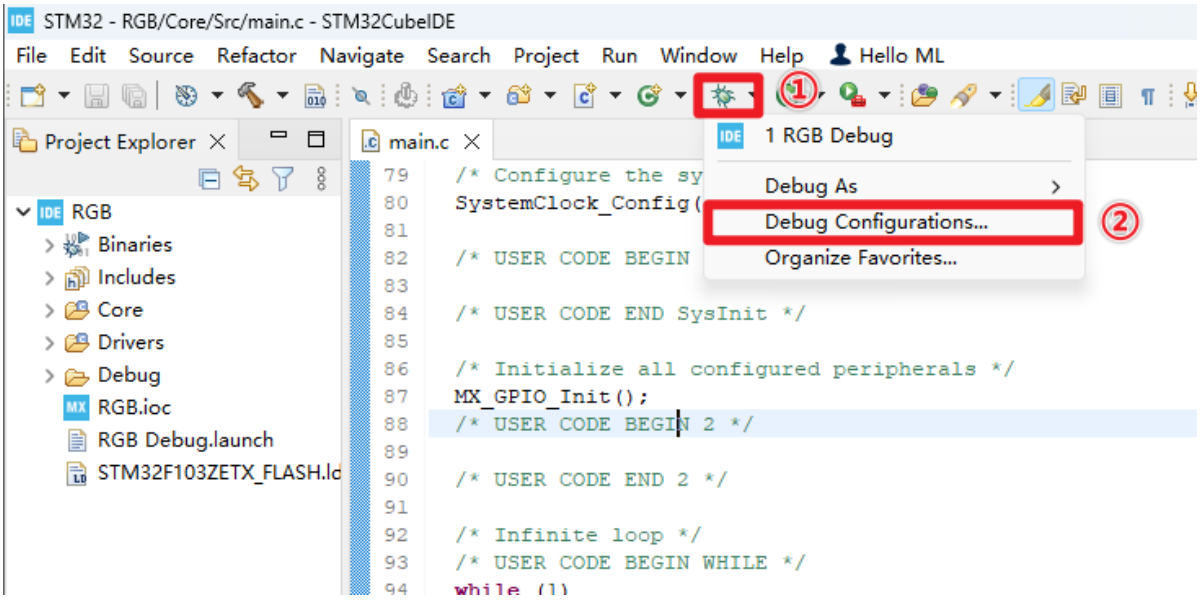
  Use the Type-C data cable to connect the development board and computer. For more detailed programming procedures, please refer to [**2. Development environment construction and use: program download and simulation**]

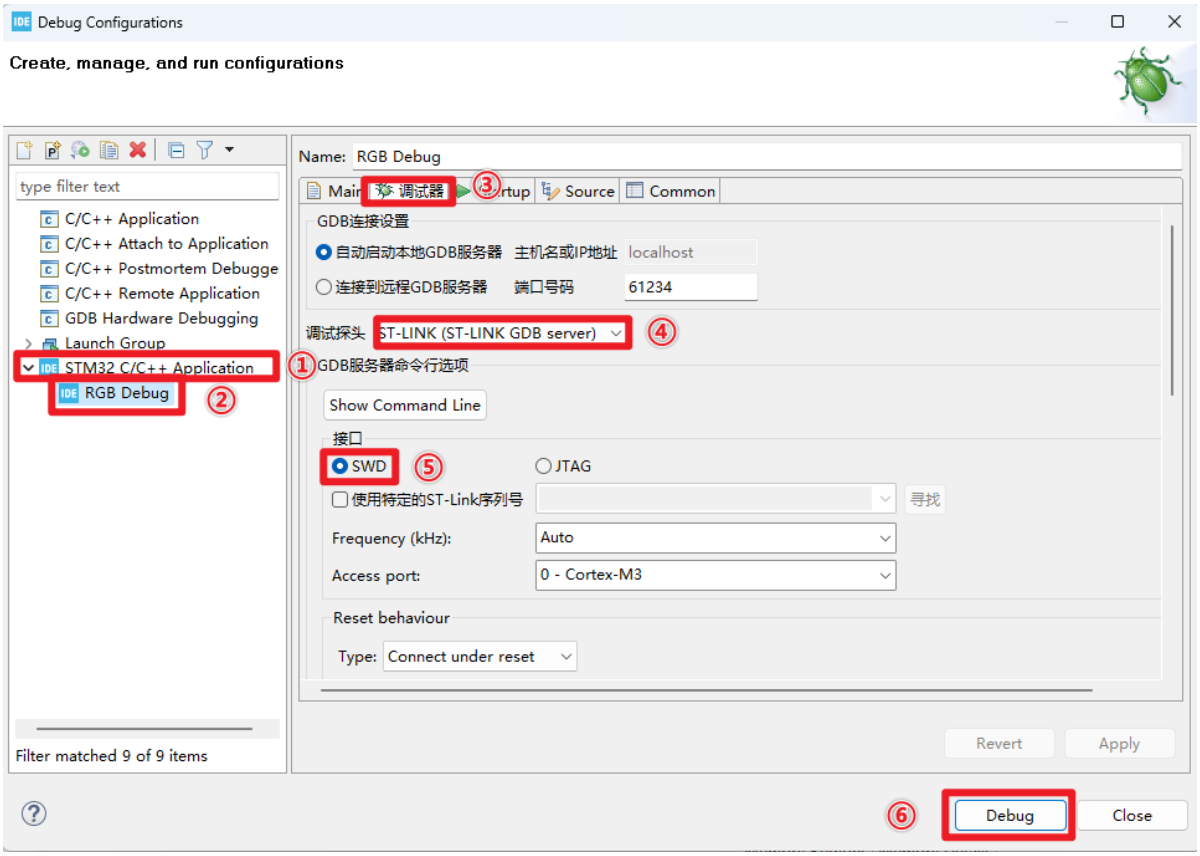


Pay attention to the contents selected in the red box, which must be consistent

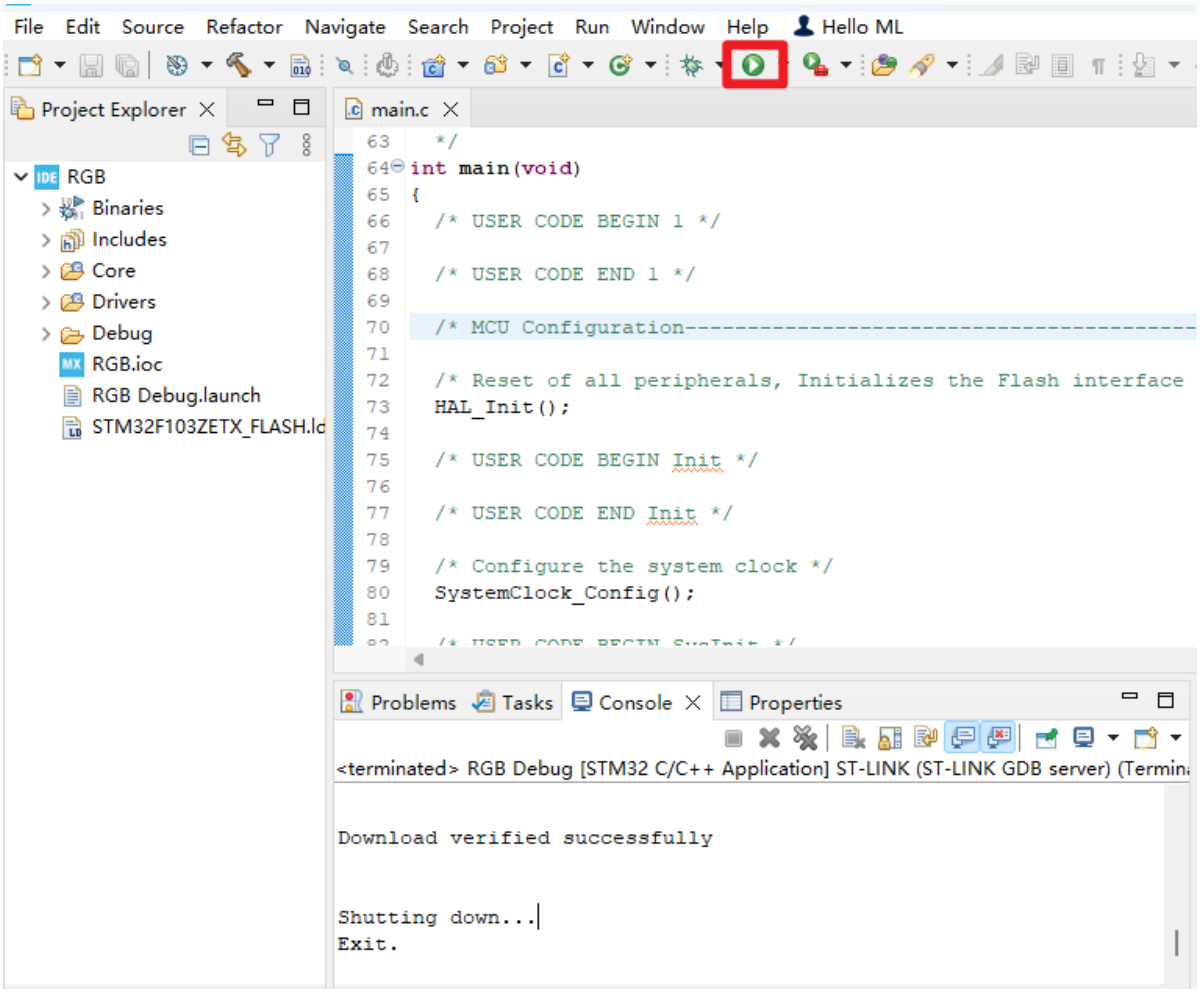
2. ST-Link Download

- **Debug Settings**: ST-Link → SWD





- **Program Download**: Click the Debug icon



7. Experimental Phenomenon

After downloading the program successfully, press the RESET button of the development board and observe the development board phenomenon!

```
For program download, please refer to [2. Development environment construction
and use: program download and simulation]
```

Phenomenon:

The RGB light on the right turns red for 0.2 seconds and turns off for 0.2 seconds;

The RGB light on the left turns green for 0.2 seconds and turns off for 0.2 seconds;

The RGB lights on the left and right sides turn purple for 0.2 seconds and turn off for 0.2 seconds;

```
You can see the experimental phenomenon [RGB Searchlight_Experimental
Phenomenon.mp4]
```