Servo control

This tutorial demonstrates: using the **basic timer (TIM7) interrupt** to simulate PWM control of the external S1 interface servo on the development board.

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

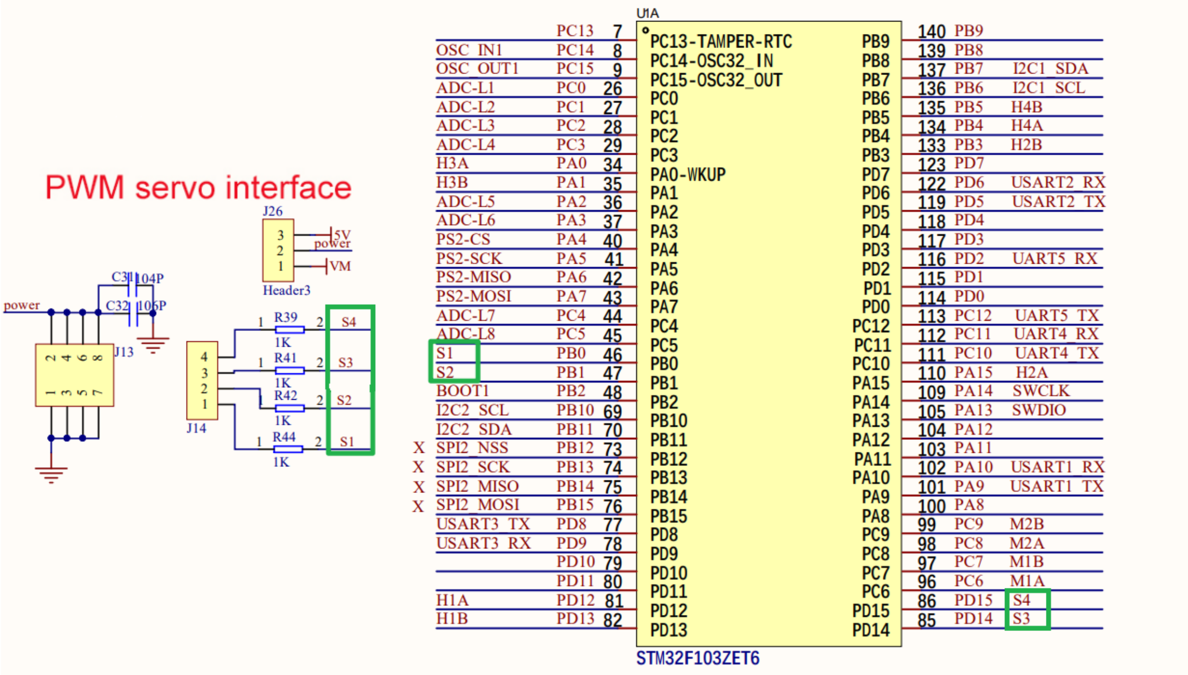
1. Software-Hardware

- **STM32F103CubeIDE**
- **STM32 Robot Development Board**
- **180° servo**
- **Type-C data cable or ST-Link**

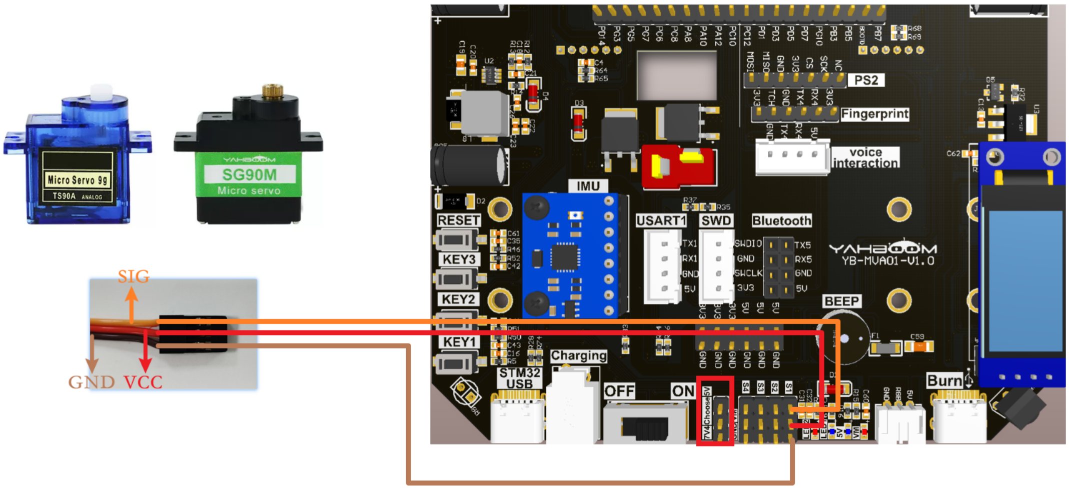
  Download programs or simulate the development board

2. Brief principle

1. Hardware schematic diagram



2. Physical connection diagram



Note: Refer to the color of the servo wire for servo wiring. **This experiment requires connecting 5V and GND jumper caps**

Steering gear	Development board
VCC(red)	5V
SIG(yellow)	S1/S2/S3/S4
GND(brown)	GND

3. Control principle

By changing the duty cycle of the PWM signal, the angle of rotation of the servo is controlled

- **PWM（Pulse Width Modulation）**

PWM is the abbreviation of pulse width modulation, which is a technology that controls the level by adjusting the pulse width of the signal.

Period: The duration of a complete PWM waveform;

Duty cycle: the ratio of high level duration to cycle time;

Frequency: The reciprocal of the period is called frequency, which is the number of PWM cycles generated per second;

- ### PWM servo

Set the period of the PWM control signal to 20ms, which is a frequency of 50Hz; the high level time of the pulse determines the angle of rotation of the servo.

High level pulse width corresponding to common angles

Steering gear (180°)	High level pulse width (us)
0°	500
45°	1000
90°	1500
135°	2000
180°	2500

- ### Basic timer

Using the TIM7 timer interrupt function on the STM32F103ZET6 development board

PWM servo (schematic name)	Control pin	Function
S1	PB0	Analog PWM output control S1 servo
S2	PB1	Analog PWM output control S2 servo
S3	PD14	Analog PWM output control S3 servo
S4	PD15	Analog PWM output control S4 servo

```
For basic knowledge of basic timers, please refer to [3. Basic Development Board
Tutorial: Basic Timers]
```

> **Timing formula**

$$T(s) = \frac{(ARR+1)*(PSC+1)}{TIM\_CLK(Hz)}$$

Timing time of this project: 10us

$$T(s) = \frac{(ARR+1)*(PSC+1)}{TIM\_CLK(Hz)} = \frac{(9+1)*(71+1)}{72000000(Hz)} = 0.00001s = 10us$$

3. Engineering experience

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

1. Open the project

- **Project file location**

  Project file path: Under the [Project Source Code] folder of the Chapter 4 tutorial

名称	类型	大小
1.timer_servo	文件夹	

- **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.
```

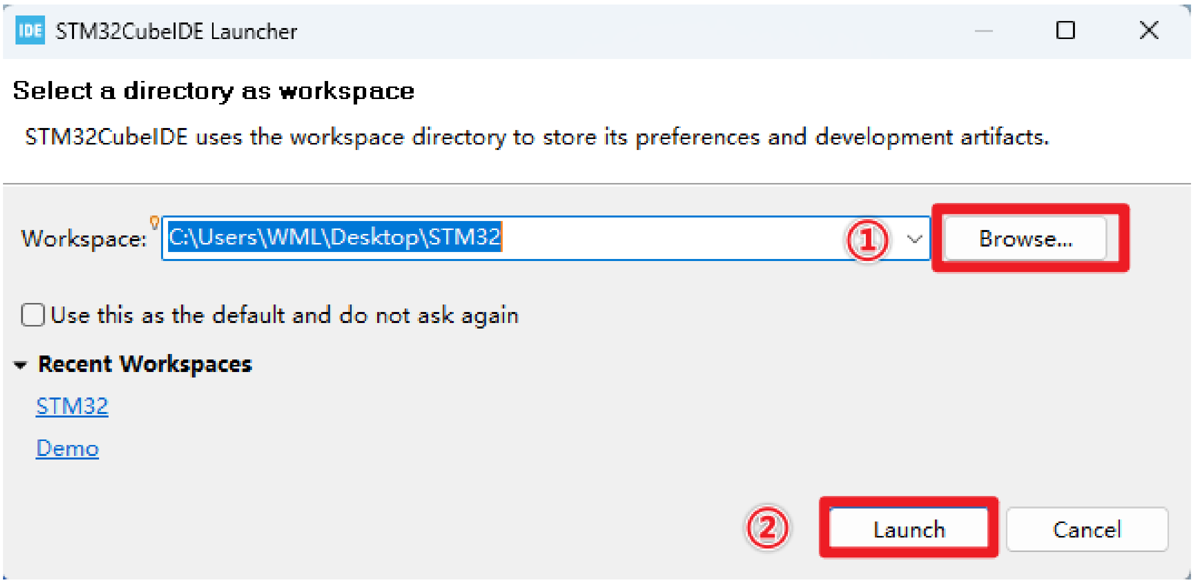
4. Project configuration

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



STM32CubeIDE Launcher

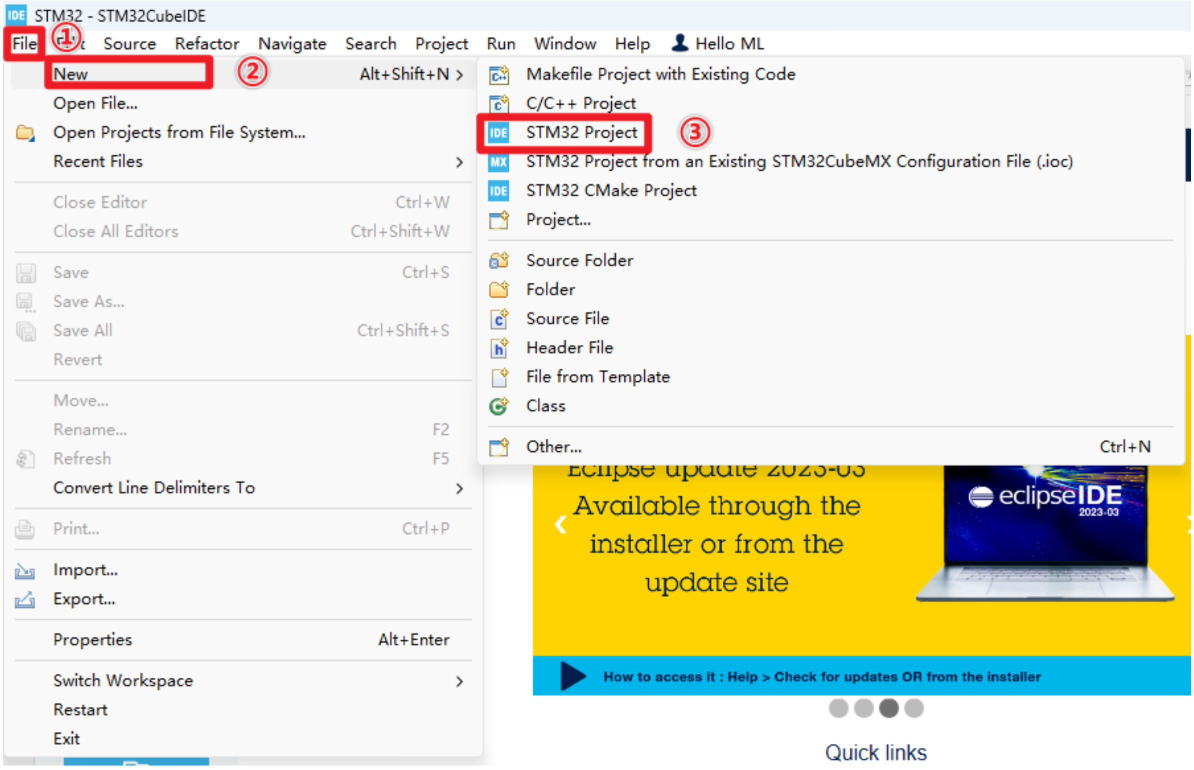
Select a directory as workspace

STM32CubeIDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\WML\Desktop\STM32

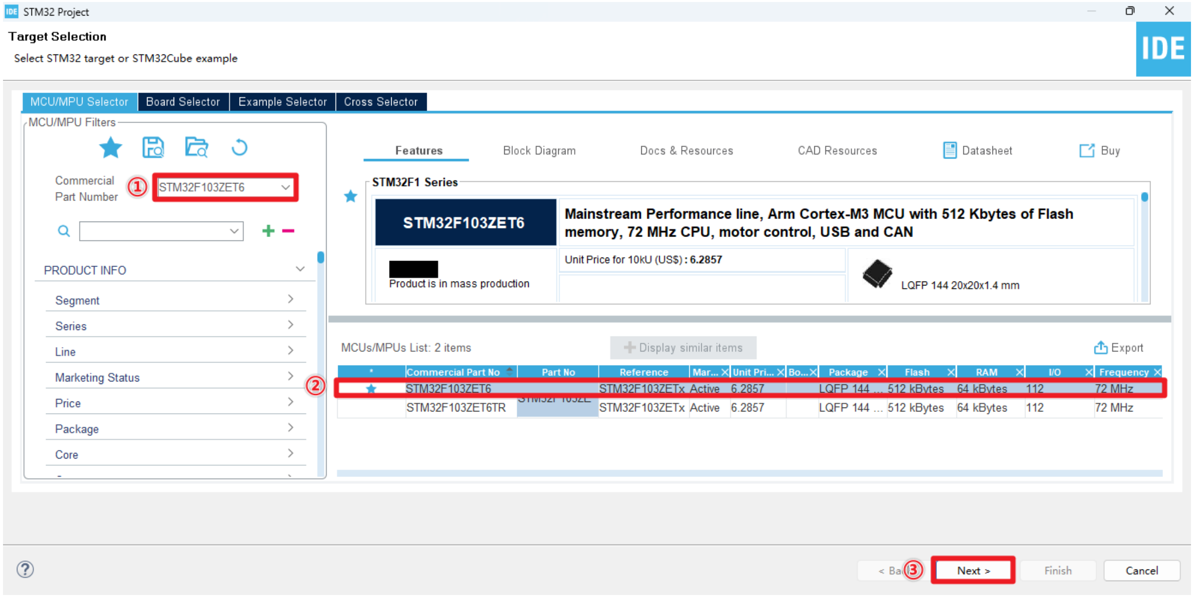
☐ Use this as the default and do not ask again

- **New Project**



STM32 - STM32CubeIDE

2、chip selection



STM32 Project

Target Selection

Select STM32 target or STM32Cube example

① Commercial Part Number: STM32F103ZET6

STM32F1 Series

STM32F103ZET6

Mainstream Performance line, Arm Cortex-M3 MCU with 512 Kbytes of Flash memory, 72 MHz CPU, motor control, USB and CAN

Unit Price for 10kU (US$) : 6.2857

Product is in mass production

LQFP 144 20x20x1.4 mm

MCUs/MPUs List: 2 items

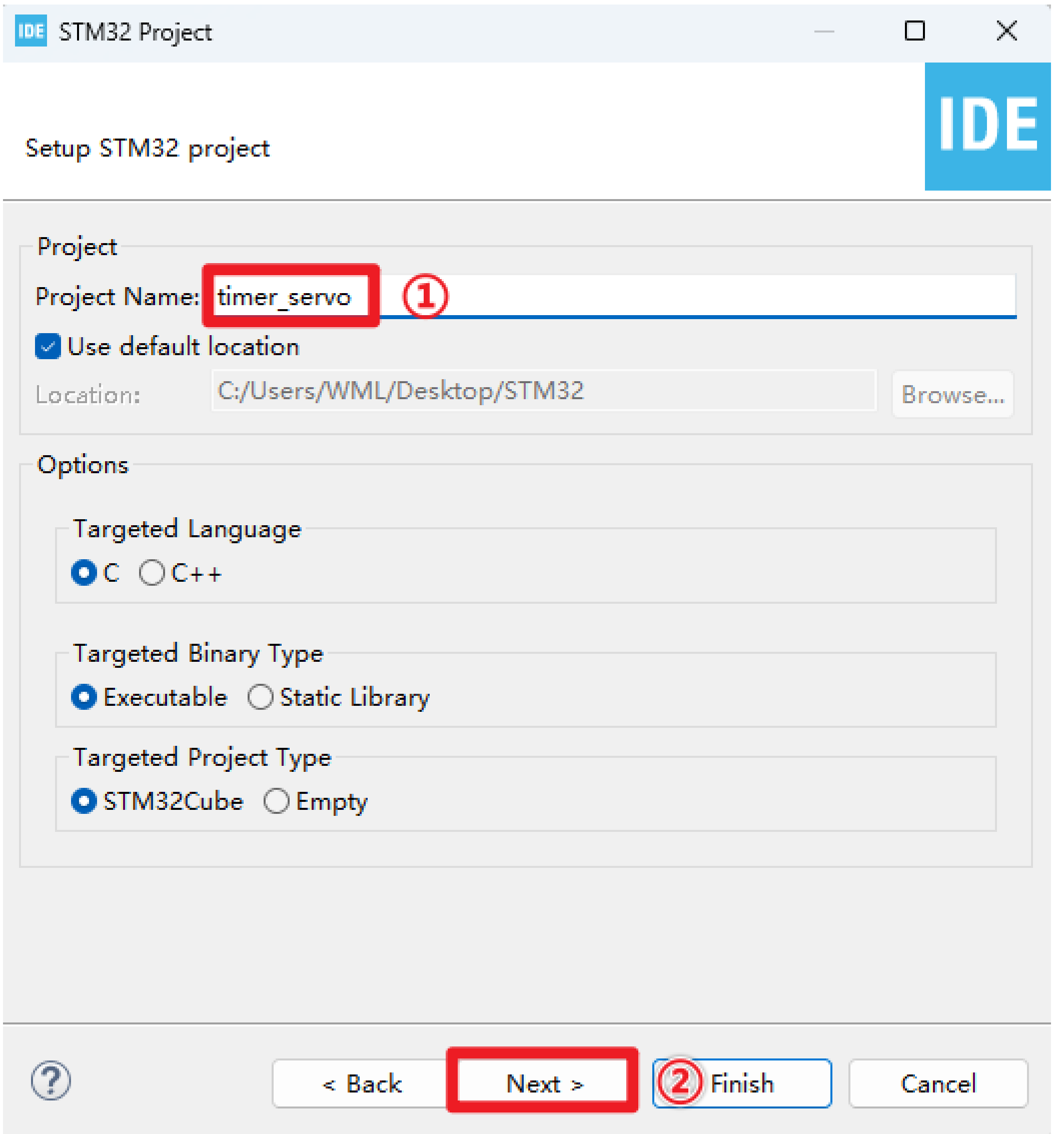
*	Commercial Part No	Part No	Reference	Mar...	Unit Pri...	Bo...	Package	Flash	RAM	I/O	Frequency
★	STM32F103ZET6	STM32F103ZE	STM32F103ZETx	Active	6.2857		LQFP 144 ...	512 kBytes	64 kBytes	112	72 MHz
	STM32F103ZET6TR		STM32F103ZETx	Active	6.2857		LQFP 144 ...	512 kBytes	64 kBytes	112	72 MHz

② (row STM32F103ZET6 selected)

③ Next >

3、Project settings

- **project name**



STM32 Project

Setup STM32 project

Project

Project Name: timer_servo ①

☑ Use default location

Location: C:/Users/WML/Desktop/STM32

Options

Targeted Language

● C ○ C++

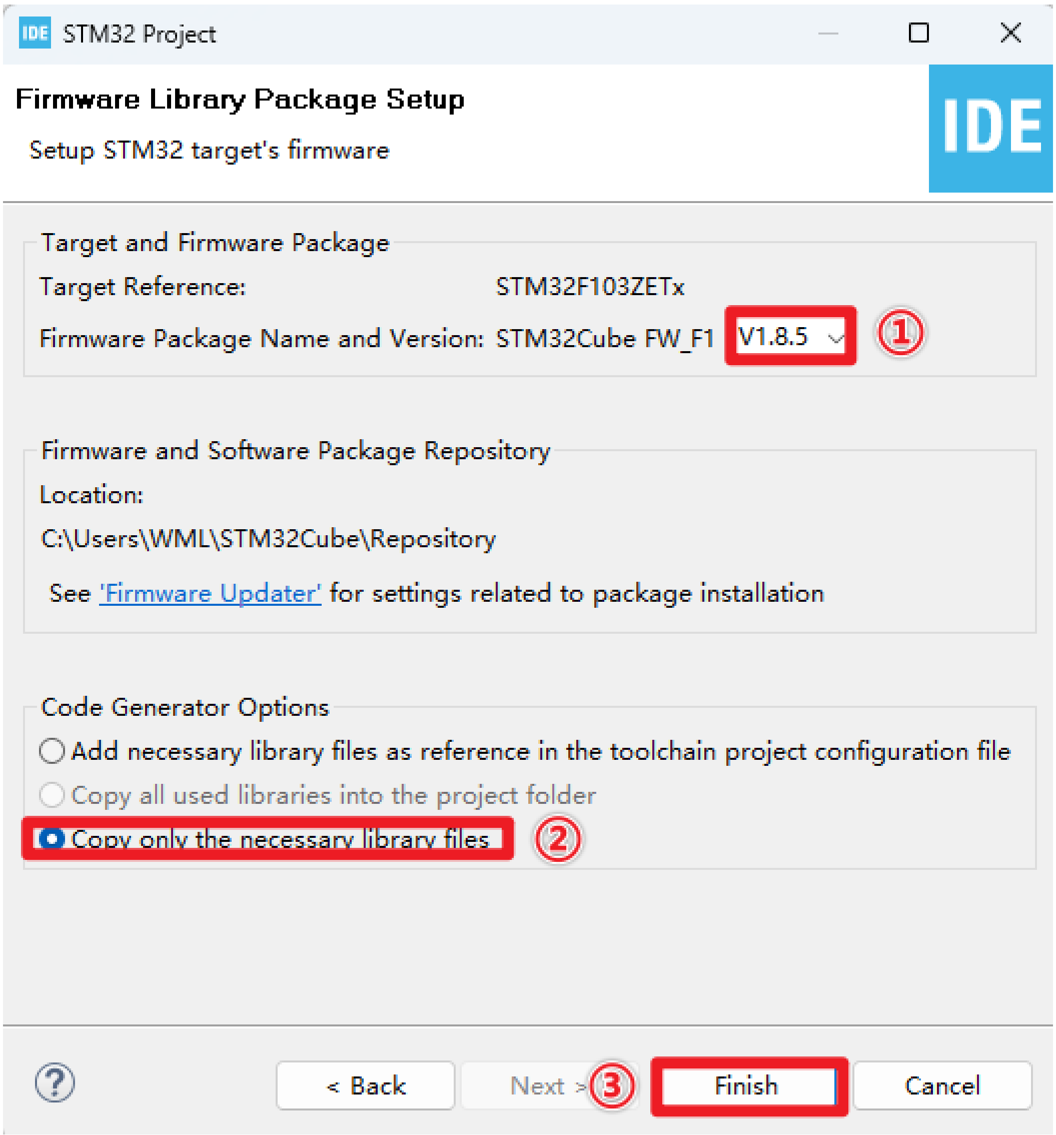
Targeted Binary Type

● Executable ○ Static Library

Targeted Project Type

● STM32Cube ○ Empty

- **Firmware version**



STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

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

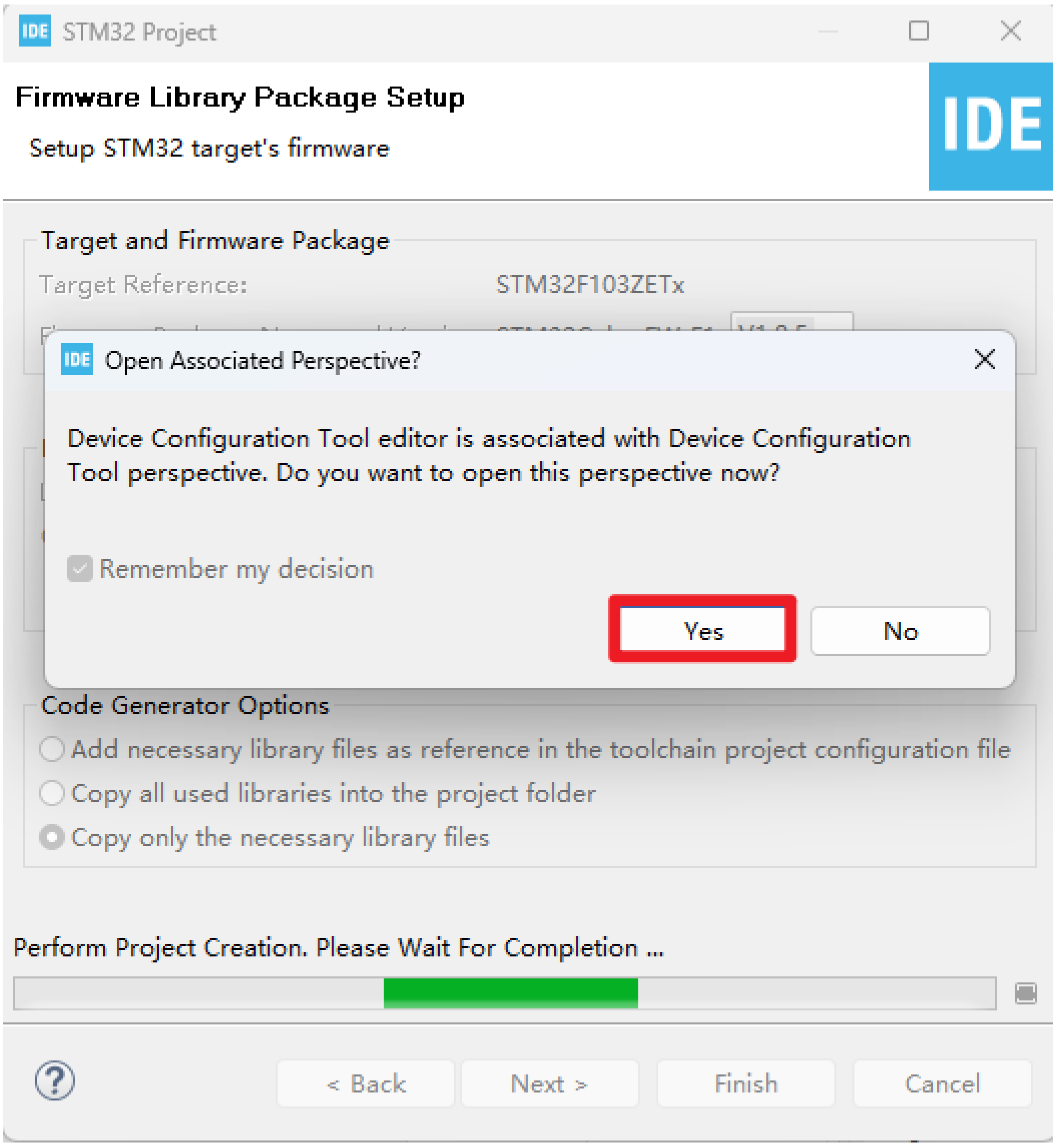
See 'Firmware Updater' for settings related to package installation

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



STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference: STM32F103ZETx

Open Associated Perspective?

Device Configuration Tool editor is associated with Device Configuration Tool perspective. Do you want to open this perspective now?

☑ Remember my decision

Code Generator Options

○ Add necessary library files as reference in the toolchain project configuration file

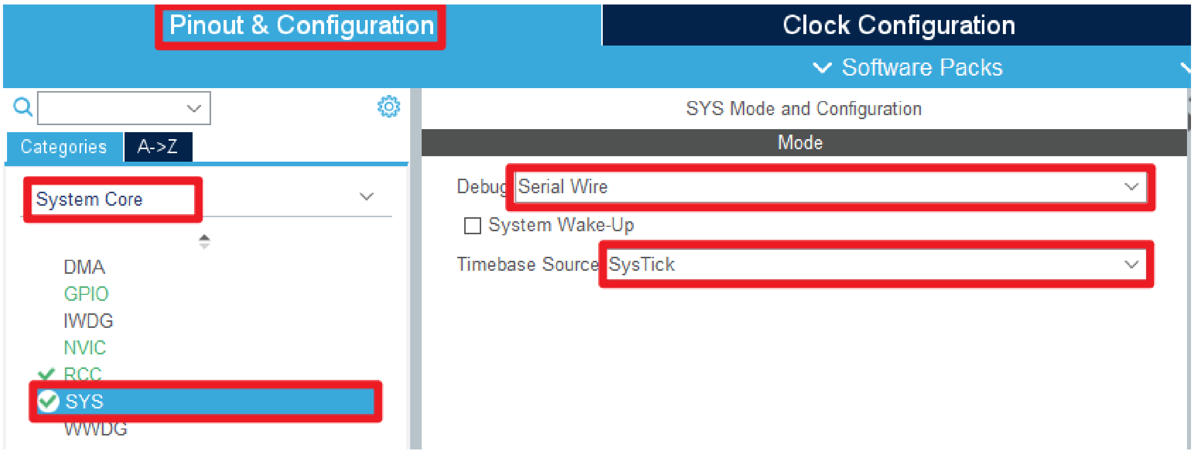
○ Copy all used libraries into the project folder

● Copy only the necessary library files

Perform Project Creation. Please Wait For Completion ...

4、Pin configuration

- **SYS**



SYS Mode and Configuration

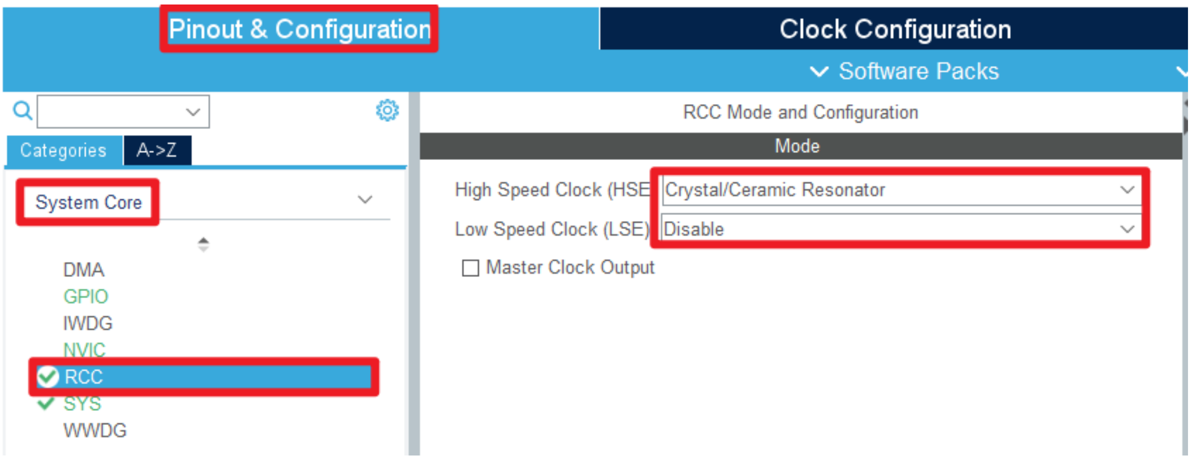
Mode

Debug Serial Wire

□ System Wake-Up

Timebase Source SysTick

- **RCC**

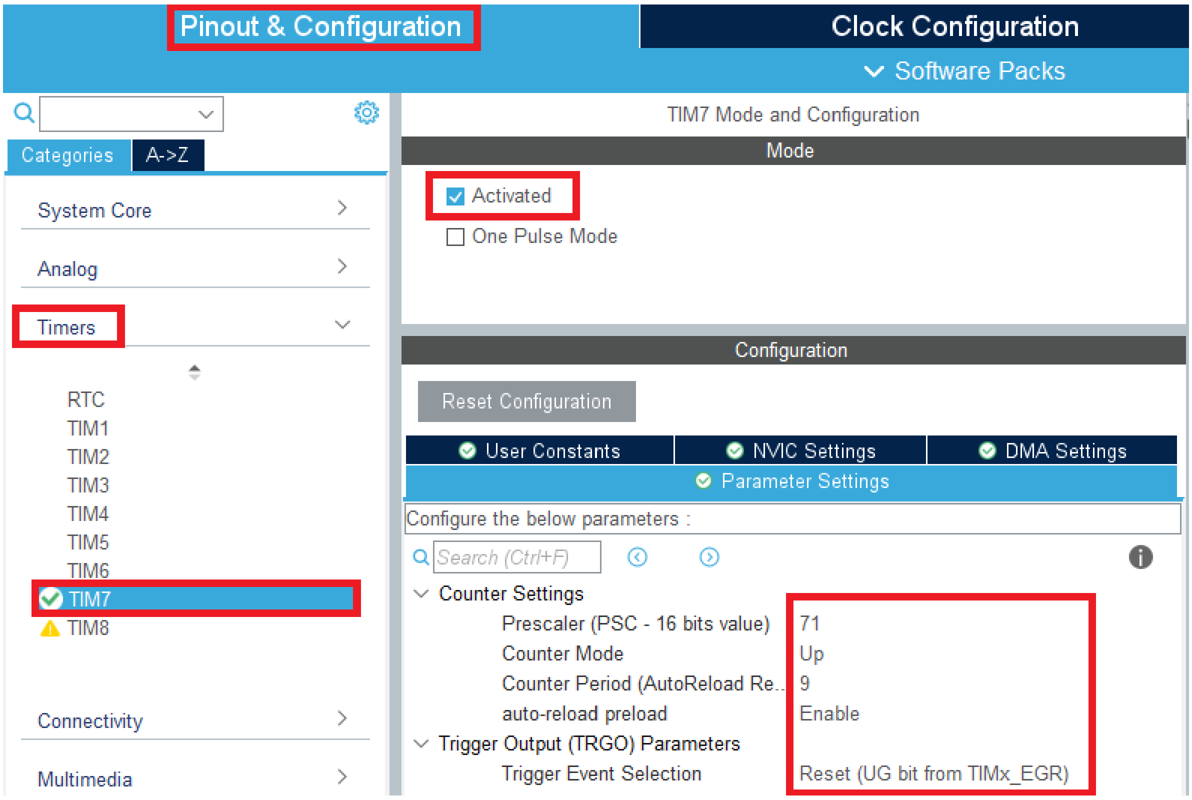


High Speed Clock (HSE: Crystal/Ceramic Resonator

Low Speed Clock (LSE): Disable

□ Master Clock Output

- **TIM**



☑ Activated

□ One Pulse Mode

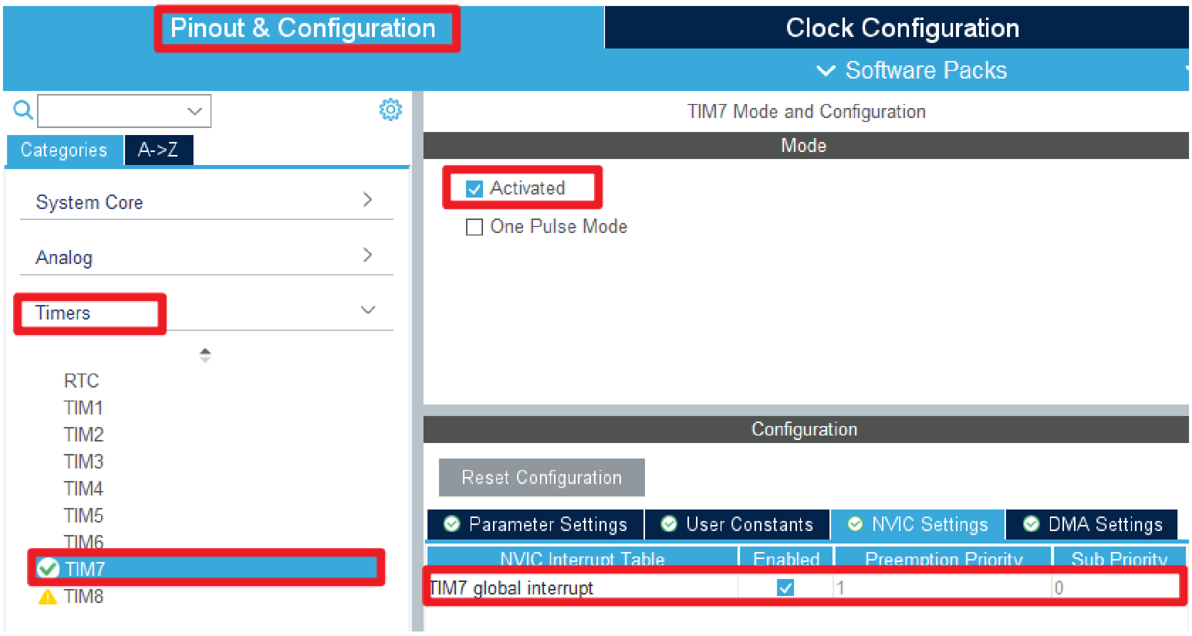
Configure the below parameters :

Counter Settings

Prescaler (PSC - 16 bits value)	71
Counter Mode	Up
Counter Period (AutoReload Re..	9
auto-reload preload	Enable

Trigger Output (TRGO) Parameters

Trigger Event Selection	Reset (UG bit from TIMx_EGR)

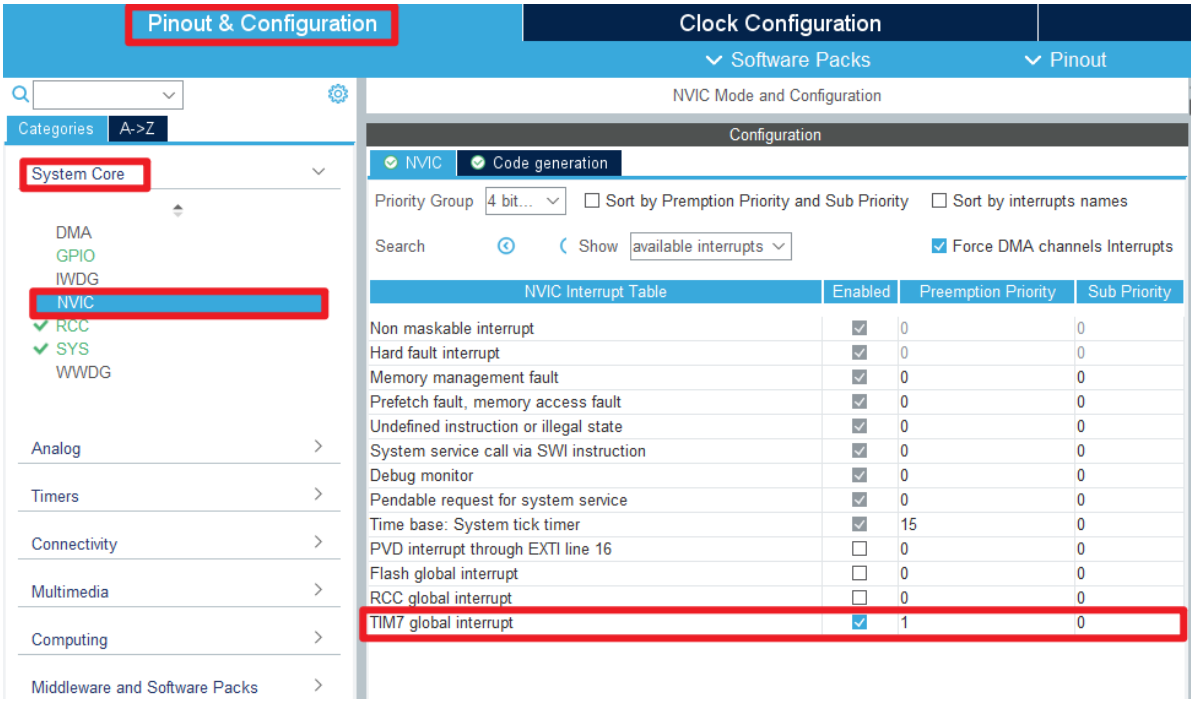


☑ Activated

□ One Pulse Mode

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
TIM7 global interrupt	☑	1	0

- **NVIC**

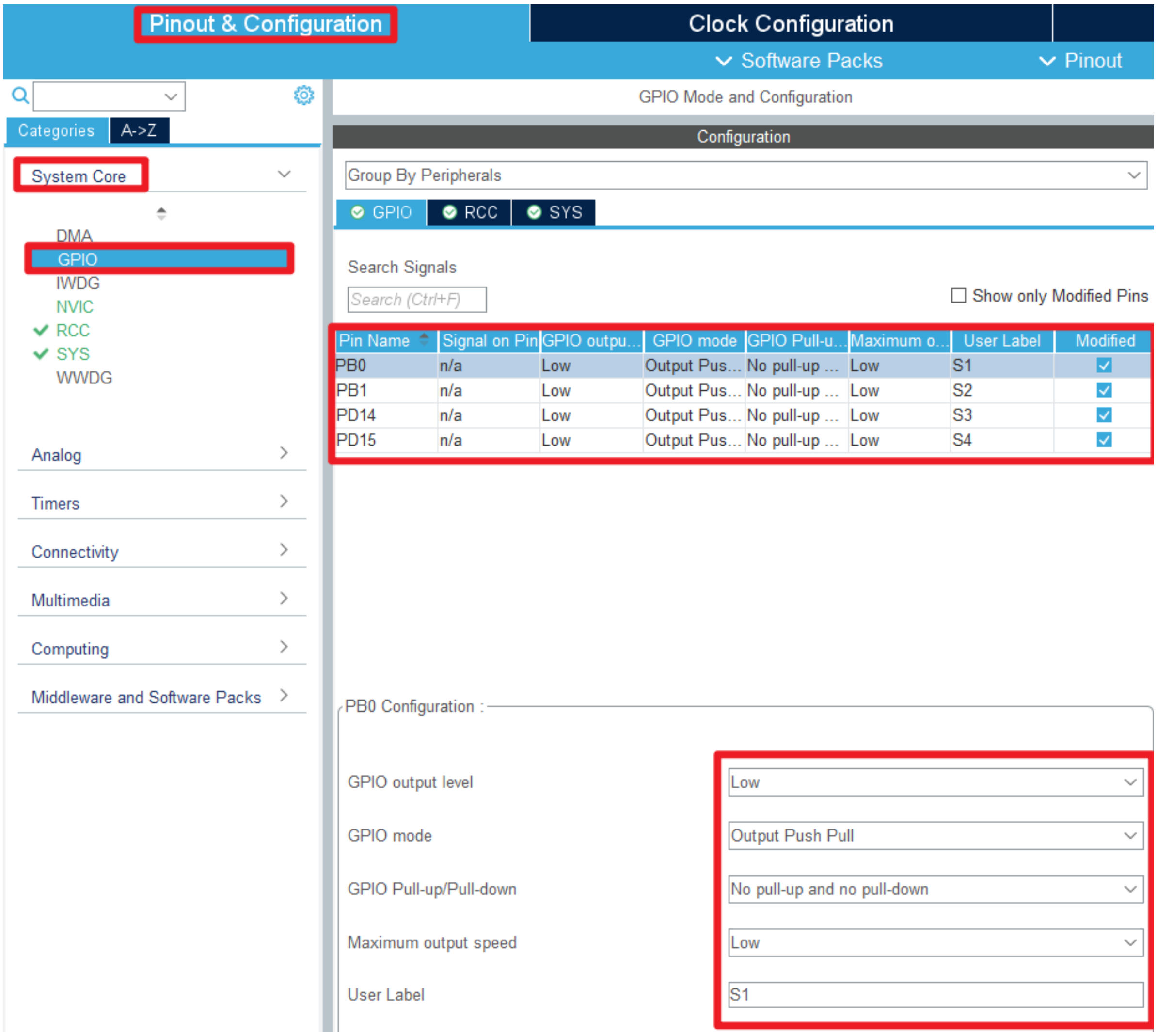


Priority Group: 4 bit... □ Sort by Premption Priority and Sub Priority □ Sort by interrupts names

Search — Show: available interrupts ☑ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Prefetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	15	0
PVD interrupt through EXTI line 16	□	0	0
Flash global interrupt	□	0	0
RCC global interrupt	□	0	0
TIM7 global interrupt	☑	1	0

- **GPIO**



□ Show only Modified Pins

Pin Name	Signal on Pin	GPIO outpu...	GPIO mode	GPIO Pull-u...	Maximum o...	User Label	Modified
PB0	n/a	Low	Output Pus...	No pull-up ...	Low	S1	☑
PB1	n/a	Low	Output Pus...	No pull-up ...	Low	S2	☑
PD14	n/a	Low	Output Pus...	No pull-up ...	Low	S3	☑
PD15	n/a	Low	Output Pus...	No pull-up ...	Low	S4	☑

PB0 Configuration :

GPIO output level: Low

GPIO mode: Output Push Pull

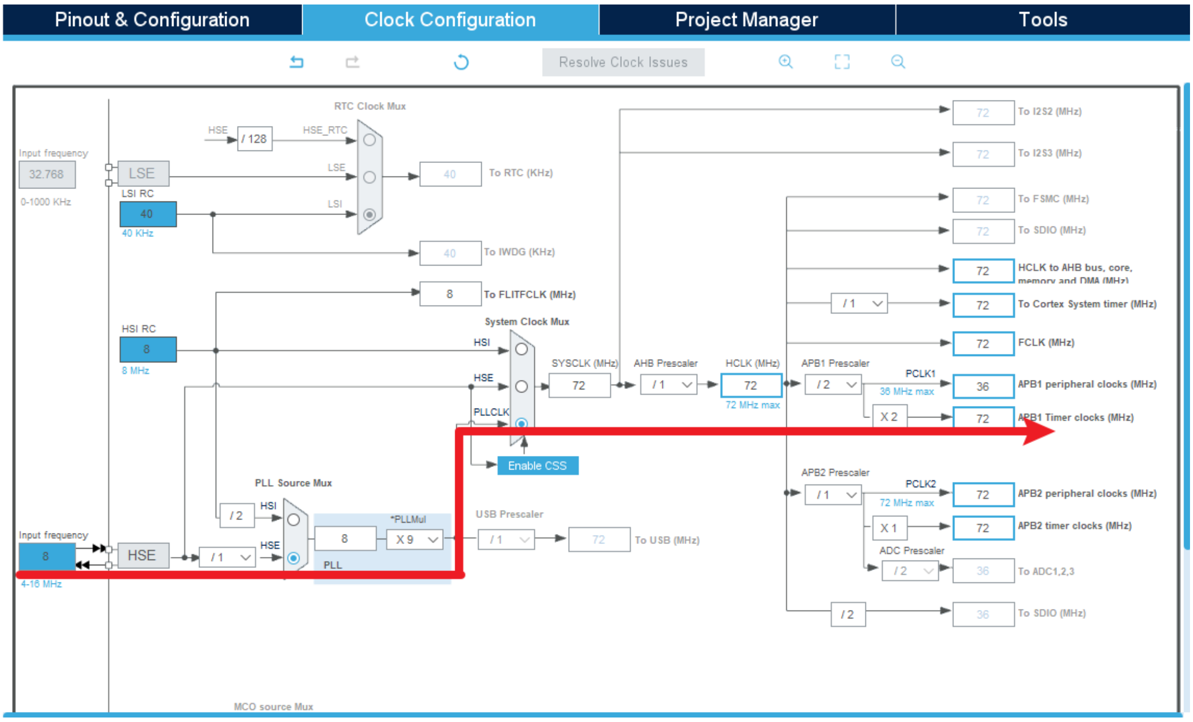
GPIO Pull-up/Pull-down: No pull-up and no pull-down

Maximum output speed: Low

User Label: S1

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

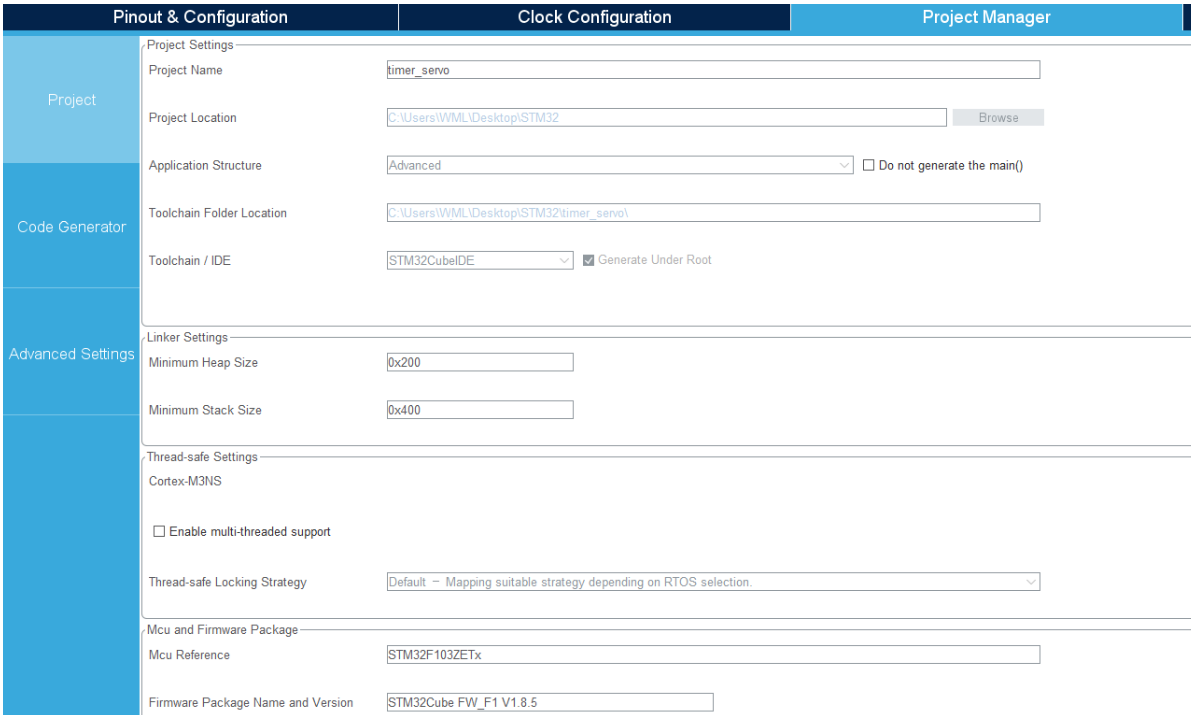


```
Refer to the options covered by the red arrows
```

6、Project configuration

- **Project**

  do not need to change



Project Settings

Project Name: timer_servo

Project Location: C:\Users\WML\Desktop\STM32

Application Structure: Advanced　□ Do not generate the main()

Toolchain Folder Location: C:\Users\WML\Desktop\STM32\timer_servo\

Toolchain / IDE: STM32CubeIDE　☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Thread-safe Settings

Cortex-M3NS

□ Enable multi-threaded support

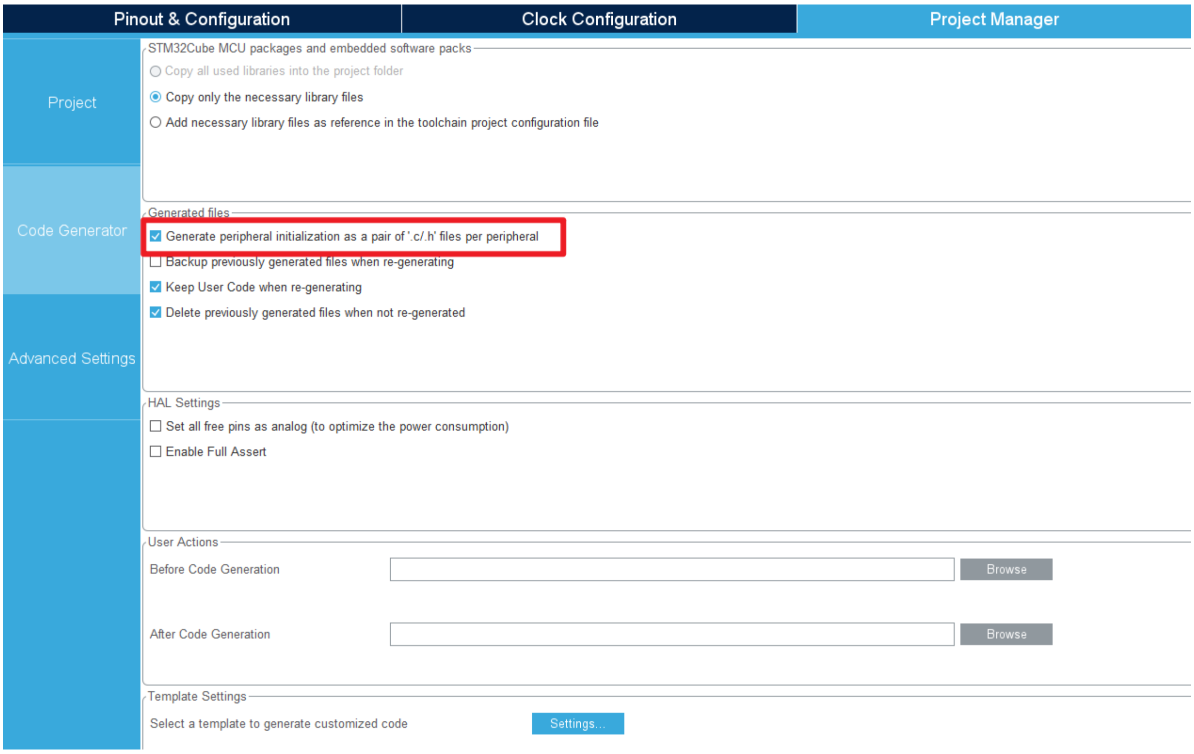
Thread-safe Locking Strategy: Default – Mapping suitable strategy depending on RTOS selection.

Mcu and Firmware Package

Mcu Reference: STM32F103ZETx

Firmware Package Name and Version: STM32Cube FW_F1 V1.8.5

- **Code Generator**



STM32Cube MCU packages and embedded software packs

○ Copy all used libraries into the project folder

● Copy only the necessary library files

○ Add necessary library files as reference in the toolchain project configuration file

Generated files

☑ Generate peripheral initialization as a pair of '.c/.h' files per peripheral

☐ Backup previously generated files when re-generating

☑ Keep User Code when re-generating

☑ Delete previously generated files when not re-generated

HAL Settings

☐ Set all free pins as analog (to optimize the power consumption)

☐ Enable Full Assert

User Actions

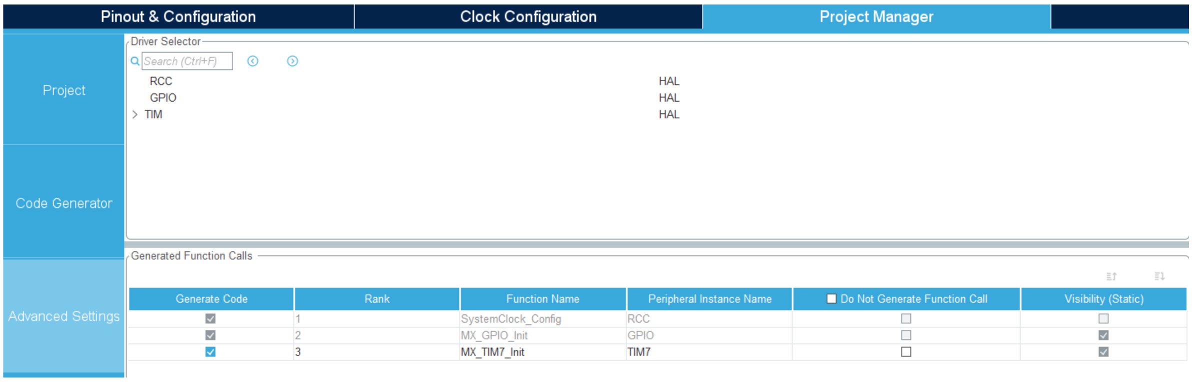
Before Code Generation [Browse]

After Code Generation [Browse]

Template Settings

Select a template to generate customized code [Settings...]

- **Advanced Settings**



Driver Selector

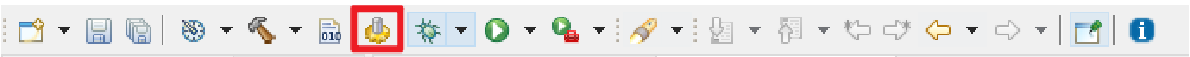
RCC	HAL
GPIO	HAL
> TIM	HAL

Generated Function Calls

Generate Code	Rank	Function Name	Peripheral Instance Name	☐ Do Not Generate Function Call	Visibility (Static)
☑	1	SystemClock_Config	RCC	☐	☐
☑	2	MX_GPIO_Init	GPIO	☐	☑
☑	3	MX_TIM7_Init	TIM7	☐	☑

7、Generate code

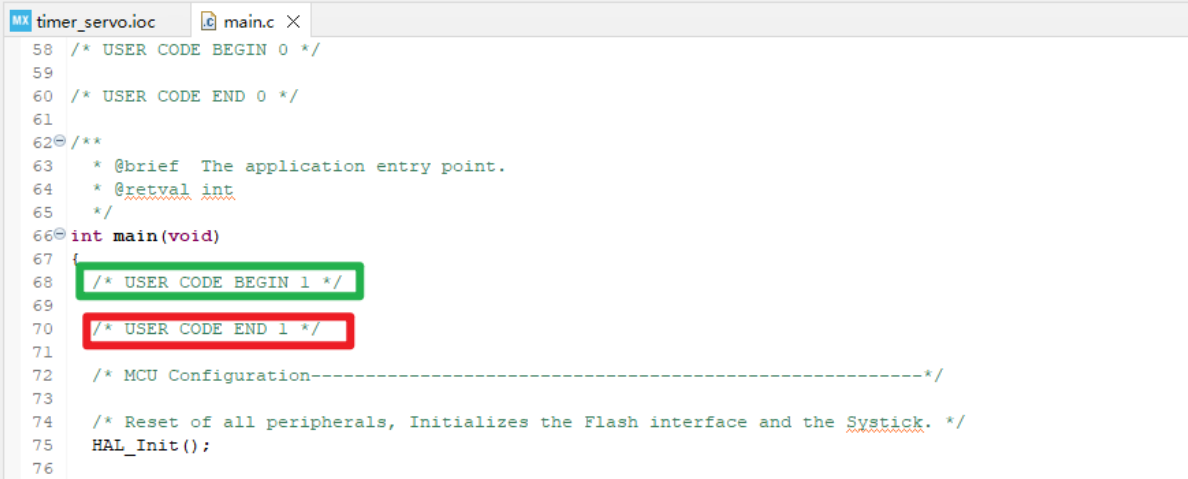
- **Click on the "pinion" icon**



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

- **Edit code**

  User code must be located between USER CODE BEGIN and USER CODE END.



```
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

```

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: PwmServo_Handle**

Function prototype	**void PwmServo_Handle(void)**
Function description	Called in timer interrupt, simulate output PWM signal
Input parameters	None
Output parameters	None
Notes	This function is called by the timer update interrupt callback function HAL_TIM_PeriodElapsedCallback

> **Function: PwmServo_Angle_To_Pulse**

Function prototype	**uint16_t PwmServo_Angle_To_Pulse(uint8_t angle)**
Function description	Convert the specified angle into pulse width value
Input parameters	**angle**: rotation angle
Output parameters	Pulse width value

> **Function: PwmServo_Init**

Function prototype	**void PwmServo_Init(void)**
Function description	Set the initial angle of each servo to 90°

Function prototype	**void PwmServo_Init(void)**
Input parameters	None
Output parameters	None

> **Function: PwmServo_Set_Angle**

Function prototype	**void PwmServo_Set_Angle(uint8_t index, uint8_t angle)**
Function description	Set the initial angle of a single servo
Input parameter 1	**index**: servo serial number
Input parameter 2	**angle**: rotation angle
Output parameters	None

> **Function: PwmServo_Set_Angle_All**

Function prototype	**void PwmServo_Set_Angle_All (uint8_t angle_s1, uint8_t angle_s2, uint8_t angle_s3, uint8_t angle_s4)**
Function description	Set all servo rotation angles
Input parameter 1	S1 servo rotation angle
Input parameter 2	S2 servo rotation angle
Input parameter 3	S3 servo rotation angle
Input parameter 4	S4 servo rotation angle
Output parameters	None

2. HAL library function analysis

> **Function: HAL_GPIO_Init**

Function prototype	**void HAL_GPIO_Init(GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_Init)**
Function description	**Initialize GPIO pin parameters**
Input parameter 1	**GPIOx**: Set the GPIO port, x takes the value A, B, C, D, E, F, G
Input parameter 2	**GPIO_Init**: GPIO initialization structure
Output parameters	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 pin**
Input parameter 1	**GPIOx**: Set the GPIO port, x takes the value A, B, C, D, E, F, G
Input parameter 2	**GPIO_Pin**: Set the GPIO pin, x value is 0-15
Input parameter 3	**PinState**: Bit_RESET: clear the data port bit (low level); Bit_SET: set the data port bit (high level)
Output parameters	None

> **Function: HAL_TIM_Base_Init**

Function prototype	**HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleTypeDef *htim)**
Function description	**Initialize timer time base unit**
Input parameters	**htim**: timer handle address
Output parameters	**HAL status value**: HAL_OK, HAL_ERROR, HAL_BUSY, HAL_TIMEOUT
Notes	This function will call the MCU underlying initialization function HAL_TIM_Base_MspInit to complete the settings of pins, clocks and interrupts

> **Function: HAL_TIM_Base_MspInit**

Function prototype	**void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim);**
Function description	**Initialize the peripheral clock, GPIO and NVIC of the timer**
Input parameters	**htim**: timer handle address
Output parameters	None

> **Function: HAL_TIM_Base_MspDeInit**

Function prototype	**void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef *htim)**
Function description	**Cancel the initialization of timer peripheral clock, GPIO and NVIC**
Input parameters	**htim**: timer handle address

Function prototype	**void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef *htim)**
Output parameters	None

> **HAL_TIM_IRQHandler: timer interrupt service function**

Function prototype	**void HAL_TIM_IRQHandler(TIM_HandleTypeDef *htim)**
Function description	**Timer interrupt service function**
Input parameters	**htim**: timer handle address
Output parameters	None
Notes	Internally, this function needs to first determine the interrupt type and clear the corresponding interrupt flag bit, and finally call the callback function

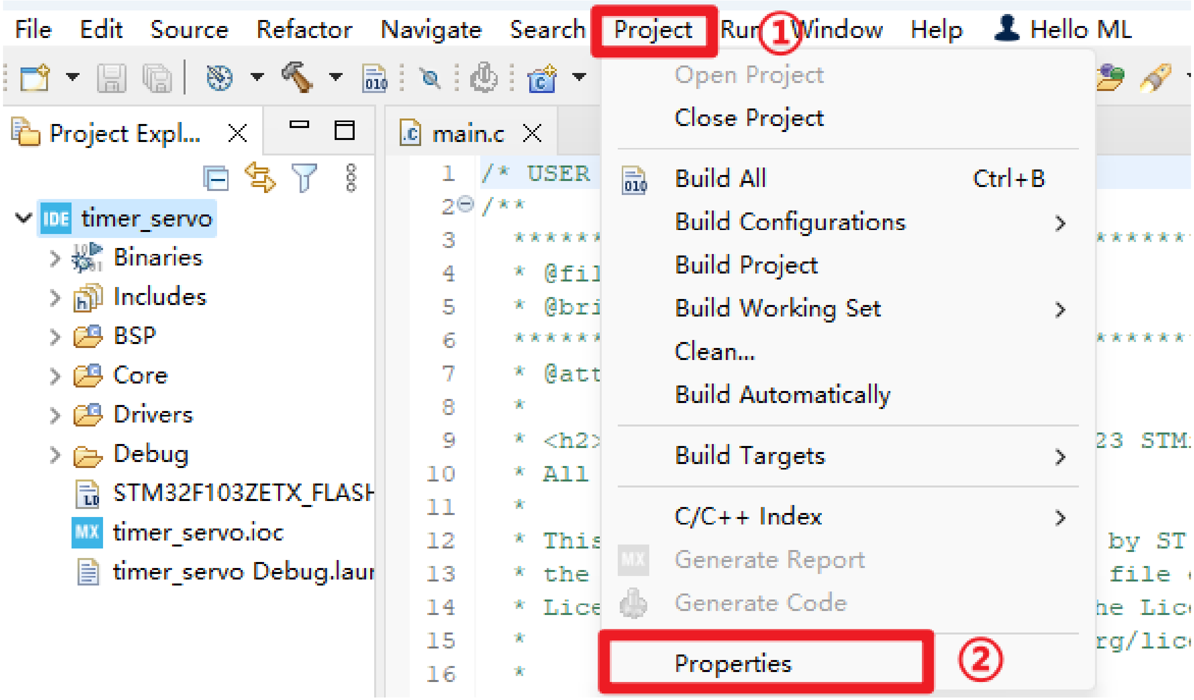
> **HAL_TIM_PeriodElapsedCallback: Timer update interrupt callback function**

Function prototype	**void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)**
Function description	**Timer update interrupt callback function**
Input parameters	**htim**: timer handle address
Output parameters	None
Notes	This function is called by HAL_TIM_IRQHandler, and specific processing tasks can be written internally

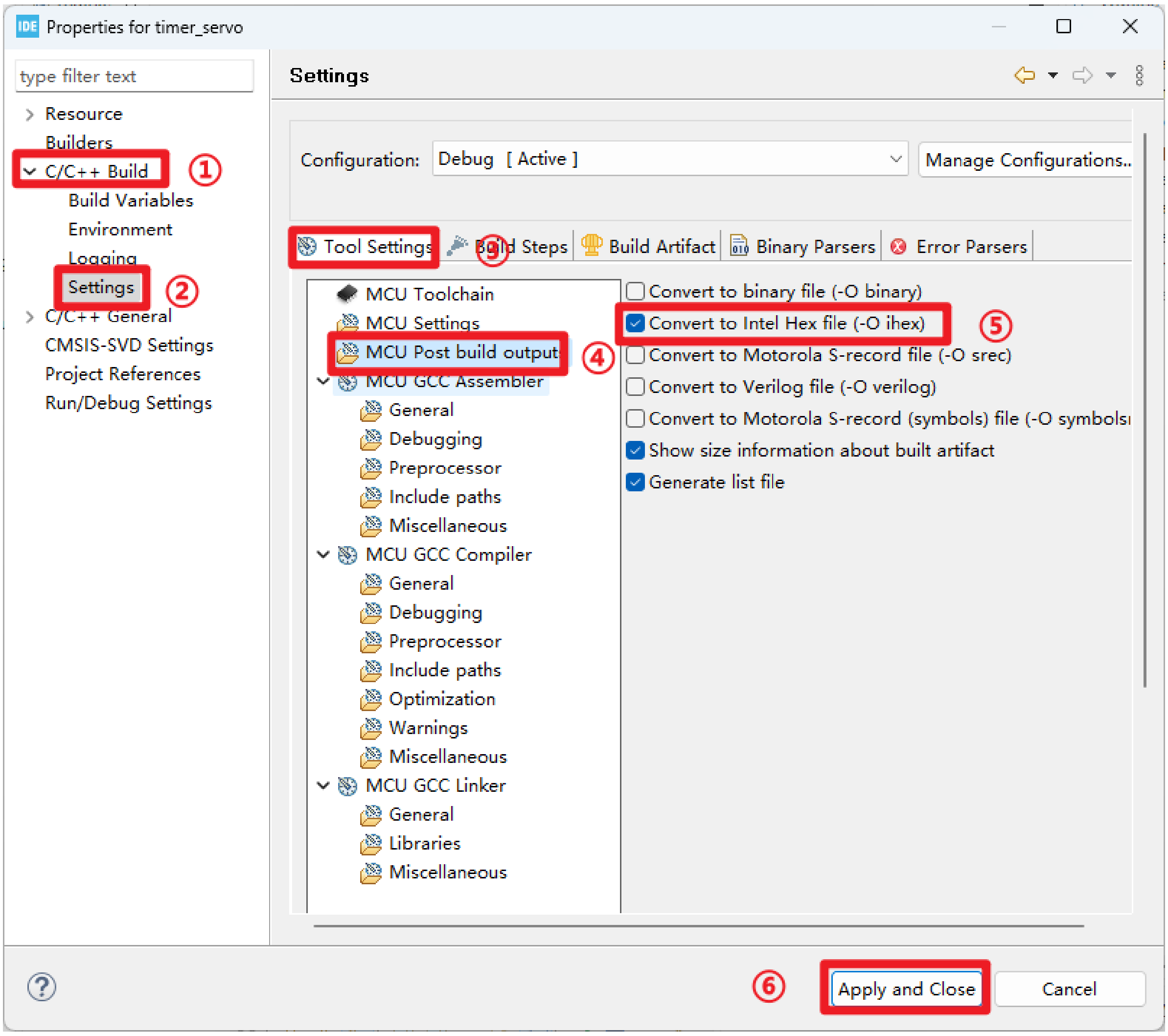
6. Program download

1. Serial port download

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

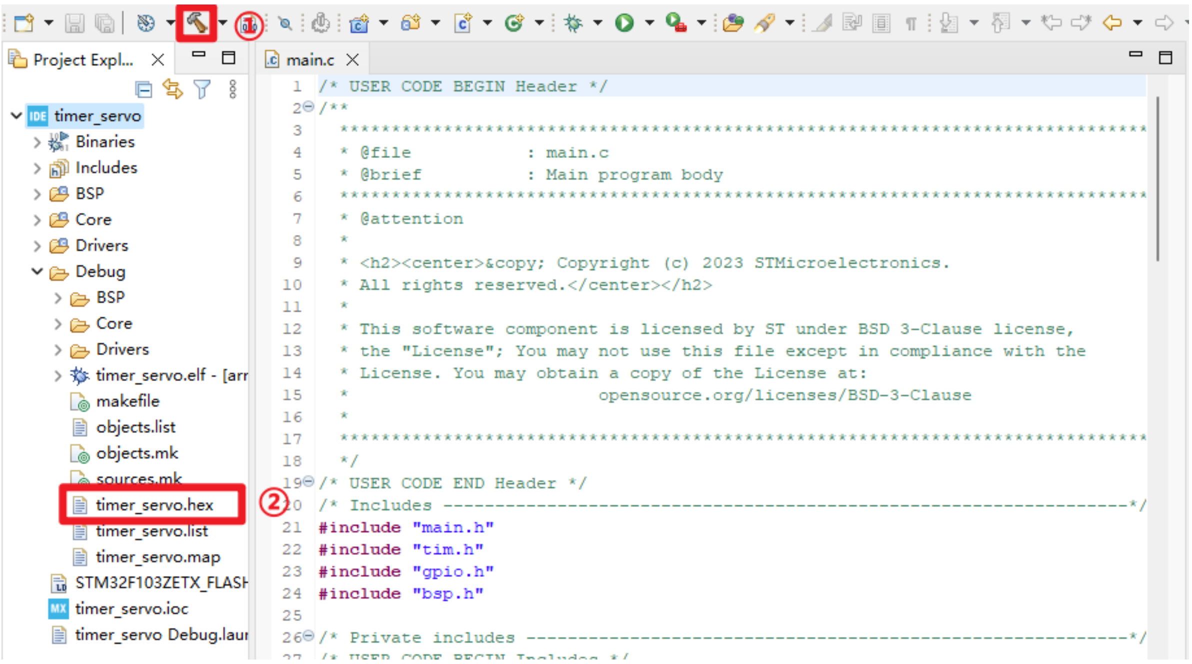


- **Check the corresponding option**



Configuration: Debug [Active]

☐ Convert to binary file (-O binary)
☑ Convert to Intel Hex file (-O ihex)
☐ Convert to Motorola S-record file (-O srec)
☐ Convert to Verilog file (-O verilog)
☐ Convert to Motorola S-record (symbols) file (-O symbols
☑ Show size information about built artifact
☑ Generate list file



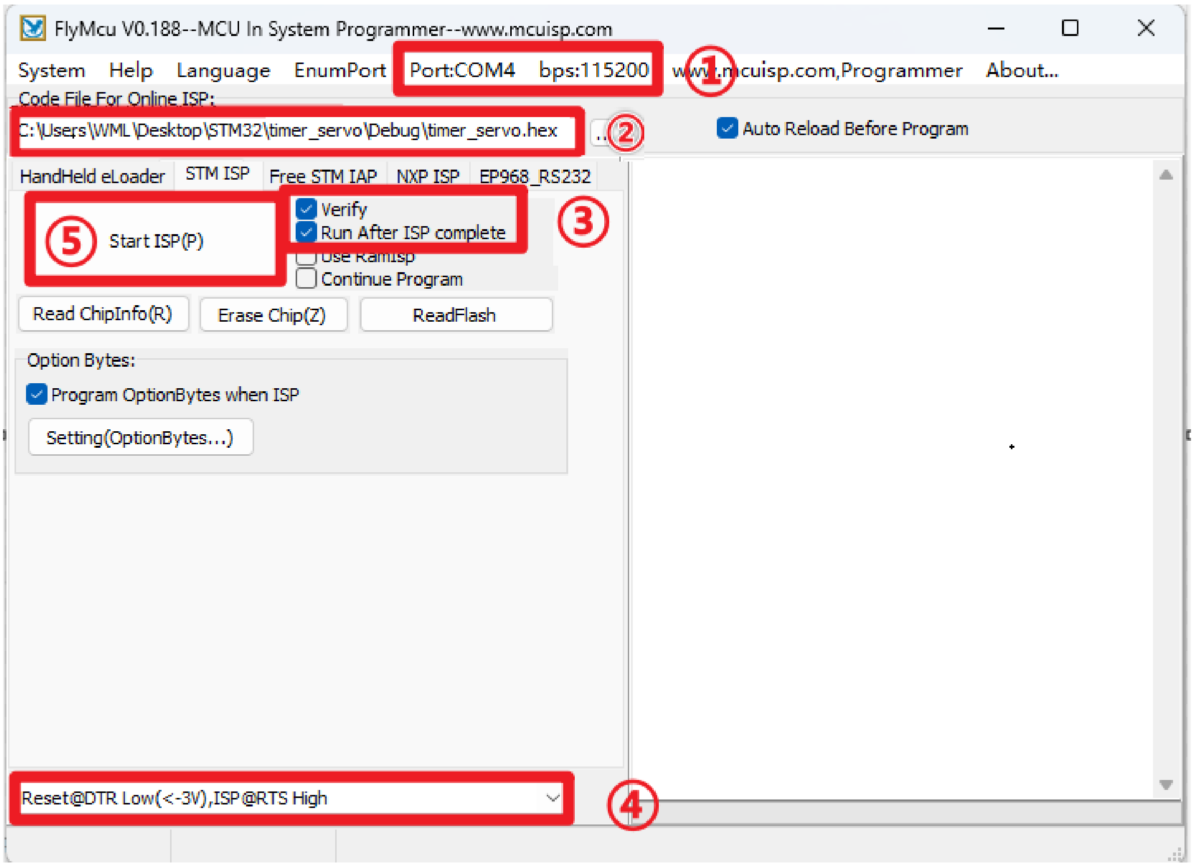
```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "tim.h"
#include "gpio.h"
#include "bsp.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
```

```
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 the computer. For more detailed burning process, please refer to **[2. Development environment construction and use: program download and simulation]**



Port:COM4 bps:115200

Code File For Online ISP:
C:\Users\WML\Desktop\STM32\timer_servo\Debug\timer_servo.hex

☑ Auto Reload Before Program

☑ Verify
☑ Run After ISP complete
☐ Use RamIsp
☐ Continue Program

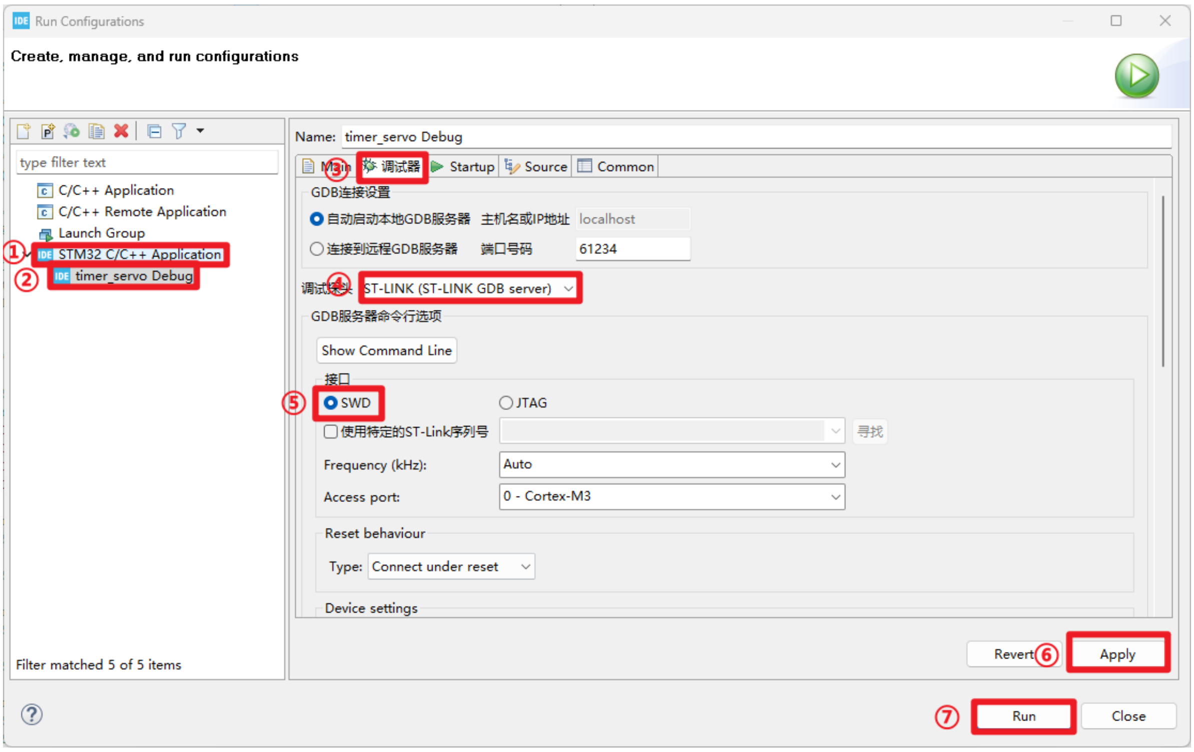
Option Bytes:
☑ Program OptionBytes when ISP

Reset@DTR Low(<-3V),ISP@RTS High

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

2、ST-Link Download

- **Debug Settings**: ST-Link → SWD



Run Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Remote Application
- Launch Group
- ① STM32 C/C++ Application
  - ② timer_servo Debug

Filter matched 5 of 5 items

Name: timer_servo Debug

Main | ③ 调试器 | Startup | Source | Common

GDB连接设置

● 自动启动本地GDB服务器 主机名或IP地址 localhost

○ 连接到远程GDB服务器 端口号码 61234

调试探头 ④ ST-LINK (ST-LINK GDB server)

GDB服务器命令行选项

Show Command Line

接口

⑤ ● SWD ○ JTAG

□ 使用特定的ST-Link序列号 寻找

Frequency (kHz): Auto

Access port: 0 - Cortex-M3

Reset behaviour

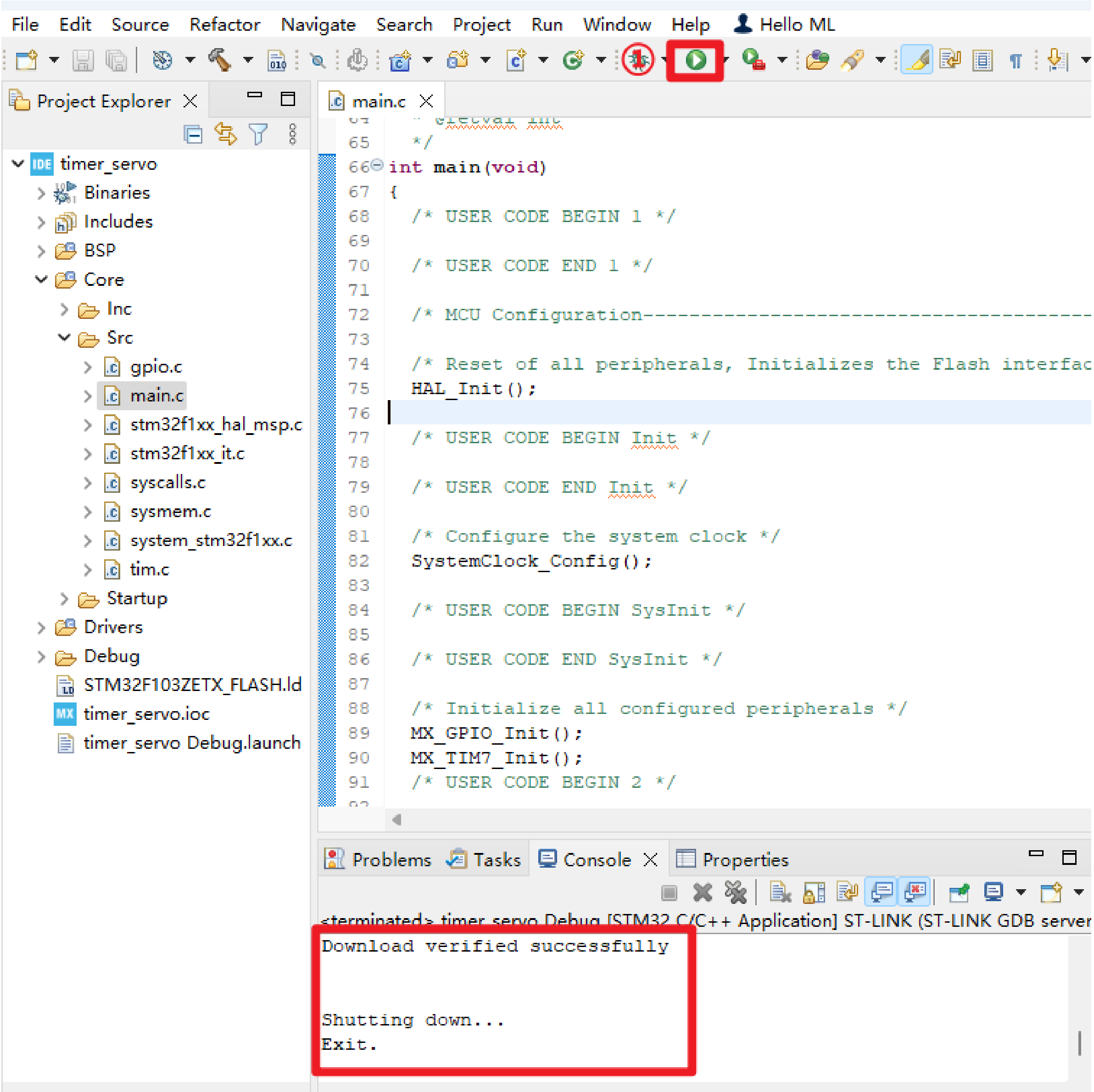
Type: Connect under reset

Device settings

Revert ⑥ Apply

⑦ Run Close

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



```
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------

  /* Reset of all peripherals, Initializes the Flash interfac
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM7_Init();
  /* USER CODE BEGIN 2 */
```

<terminated> timer_servo Debug [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server

```
Download verified successfully


Shutting down...
Exit.
```

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:

S1 interface servo: cyclic rotation from 0° → 45° → 90° → 135° → 180° → 135° → 90° → 45° → 0°.

```
The provided project only controls the S1 interface servo. You can modify the
interface function parameters by yourself to control other servos.
For experimental phenomena, please see [Servo Control_Experimental
Phenomenon.mp4]
```