FreeRTOS

This tutorial demonstrates: Real-time control of RGB lights, leds, and active buzzers with FreeRTOS.

1、software-hardware

- **STM32F103CubeIDE**
- **STM32 robot expansion board**

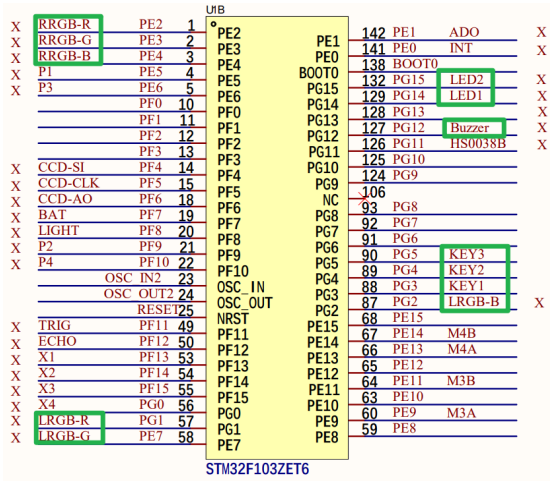
  GPIO: Internal peripheral of the chip

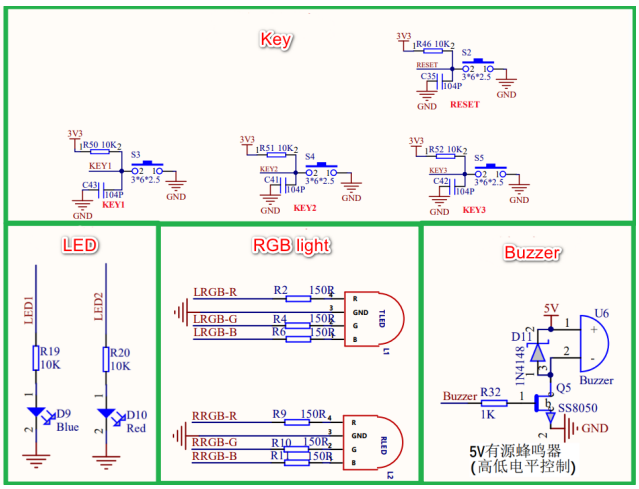
  RGB, LED, key, active buzzer: onboard
- **Type-C cable or ST-Link**

  Download or simulate the program of the development board

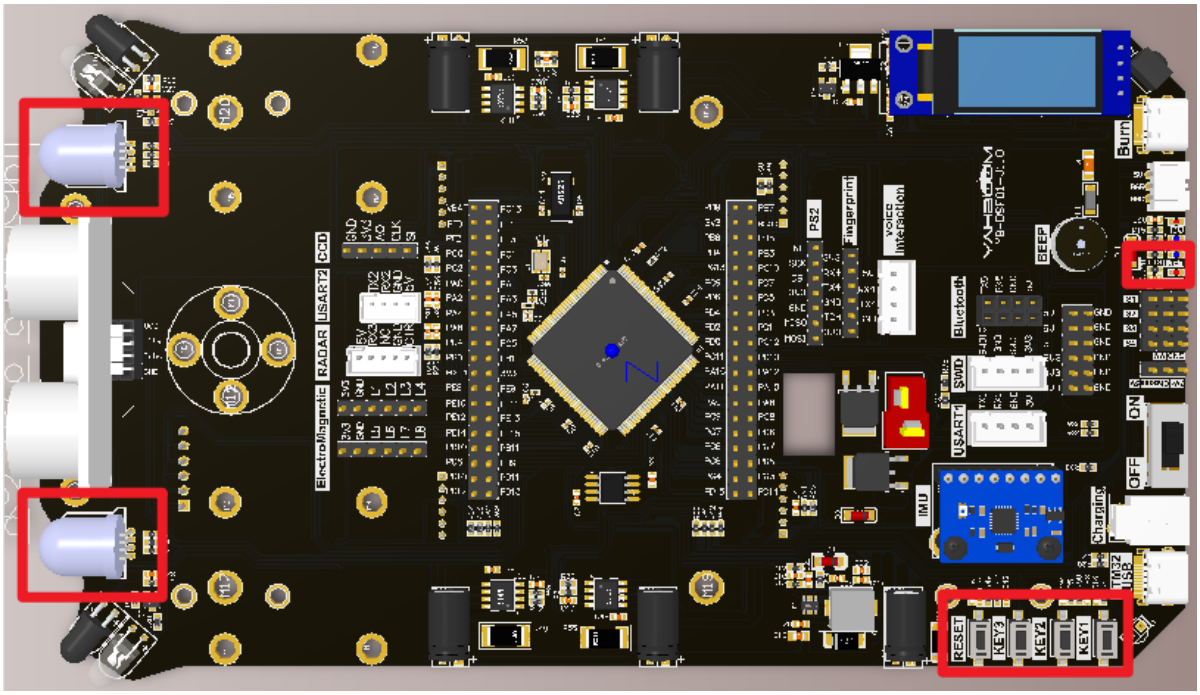
2、Brief principle

2.1、Hardware schematic diagram





2.2、Physical connection diagram



2.3、Principle of control

Bare metal systems：Most of our cases run on bare metal systems, where programs are usually inside the main loop or interrupts;
Multi-task system：The real-time operating system divides the program into independent applets, each applet is a task, each task is independent, non-interfering with each other, and has its own priority, which is configured by the operating system.

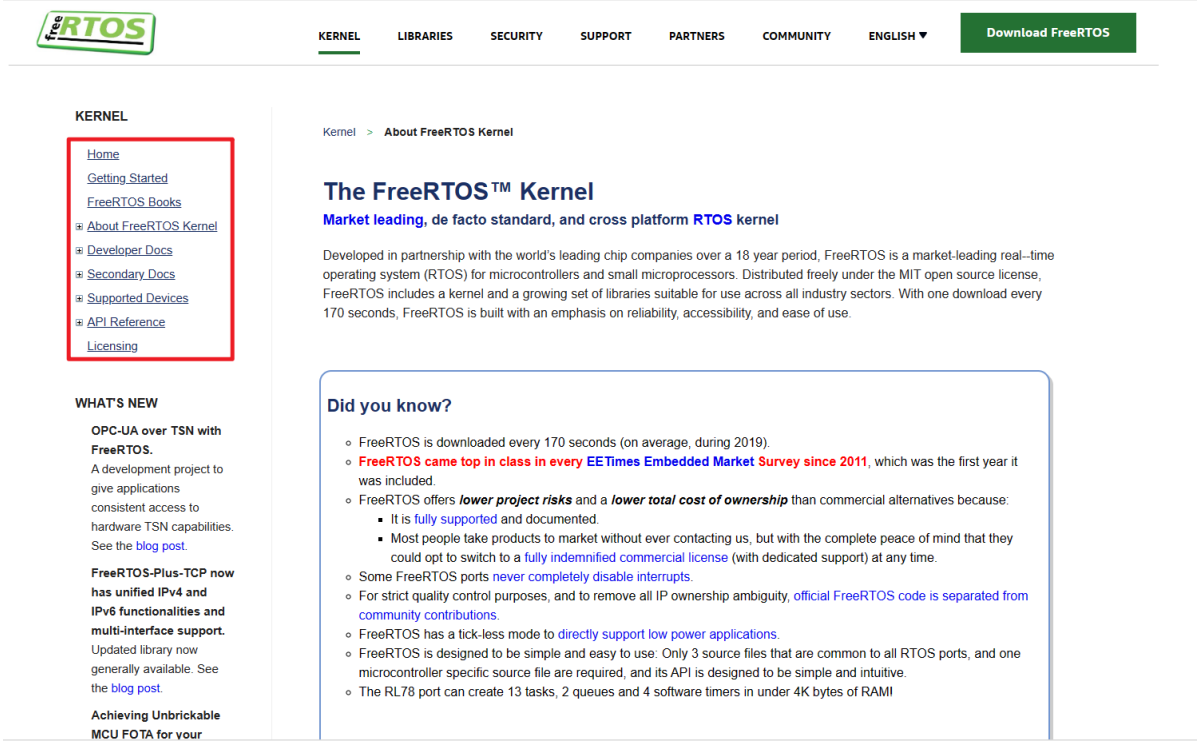
- **FreeRTOS**

FreeRTOS（Real Time Operating System）

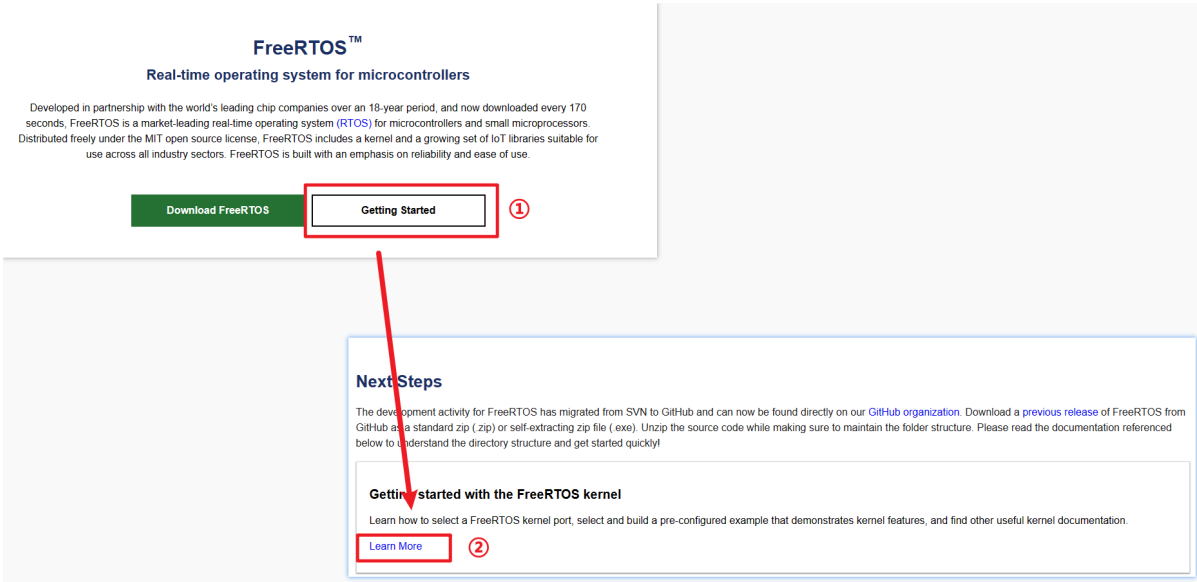
Because real-time operating system involves a lot of content, the tutorial will briefly introduce the website learning materials location and real-time operating system involved in the concept.

Official website Learning website：https://www.freertos.org/RTOS.html

The official online resources provide FreeRTOS books, documentation about the FreeRTOS kernel, the FreeRTOS API reference manual, and more.



If you enter from the official website by yourself, you should click "Getting Strated" → "Getting started with the FreeRTOS kernel: Learn More" in turn.



Basics of real-time operating systems

FreeRTOS	feature
Task	The basic unit of FreeRTOS
Scheduler	Manage the execution order and switching of tasks
Semaphore	It is used for synchronizing and mutually exclusive access to shared resources between tasks
Event Flags	Used for event notification and synchronization between tasks
Queue	It is used for data transfer and communication between tasks.
Mutex	It is used to implement mutually exclusive access to resources, ensuring that only one task can access a shared resource at a time.

FreeRTOS	feature
Timer	It can be used to periodically trigger events or perform tasks

- **Practical Applications**

We can use the CubeMX plugin to initialize and configure FreeRTOS, and configure FreeRTOS components such as tasks, timers, semaphores, and message queues according to our requirements. The following table is the name and function of the newly created tasks during the project configuration:

Task	feature
myTask_RGB	Control RGB
myTask_Beep	Control the buzzer
myTask_KEY	Check the KEY1 status
myTask_LED	Control LED

```
The task entity functions are defined in the bsp_task.c file and actually called in freertos.c
```

- **RGB：High level light, low level off**

RGB（Schematic name)	Control pin	feature
LRGB-R	PG1	Control the red light display of the RGB light on the left
LRGB-G	PE7	Control the green display of the RGB light on the left
LRGB-B	PG2	Control the blue light display of the RGB light on the left
RRGB-R	PE2	Control the red light display of the RGB light on the right
RRGB-G	PE3	Control the green display of the RGB light on the right
RRGB-B	PE4	Control the blue light display of the RGB light on the right

- **LED：High level on, low level off**

LED（Schematic name)	Control pin	feature
LED1	PG14	控制LED1亮灭
LED2	PG15	控制LED2亮灭

- **KEY：Default high level, press the key low level**

KEY（Schematic name）	Control pin	feature
KEY1	PG3	Change the KEY1 pin level state
KEY2	PG4	Change the KEY2 pin level state
KEY3	PG5	Change the KEY3 pin level state

- **Buzzer: high level sound, low level do not sound**

buzzer（Schematic name）	Control pin	feature
Buzzer	PG12	Control the buzzer to sound

3、Engineering configuration

Project Configuration: Prompts for configuration options in the STM32CubeIDE project configuration process

3.1、Notes

Omitted project configuration: **New project, chip selection, project configuration, SYS for pin configuration, RCC configuration, clock configuration, and project configuration** content

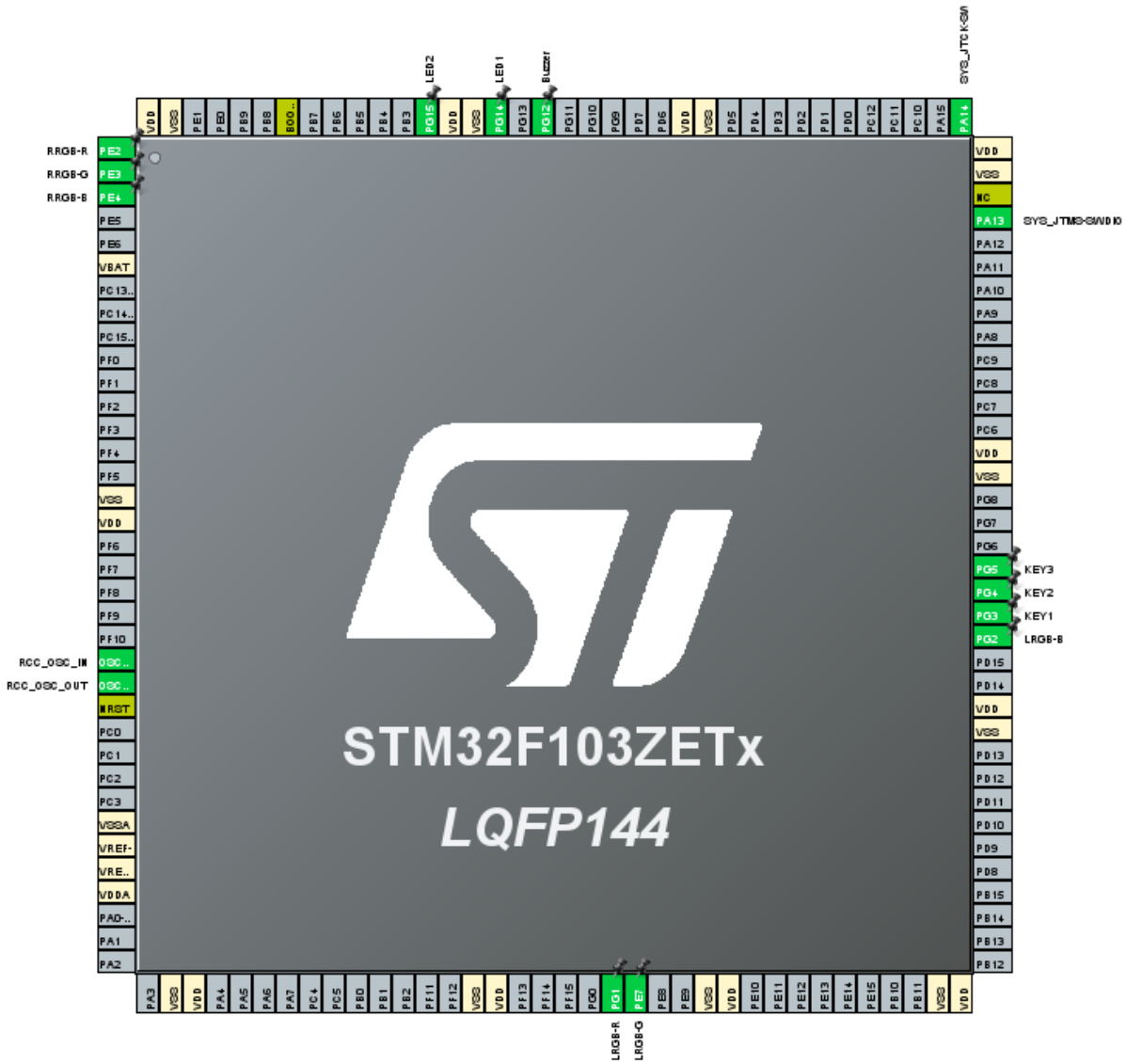
The project configuration part, which is not omitted, is the key point to configure in this tutorial.

```
Please refer to [2. Development environment construction and use: STM32CubeIDE installation and use] to understand how to configure the omitted parts of the project.
```

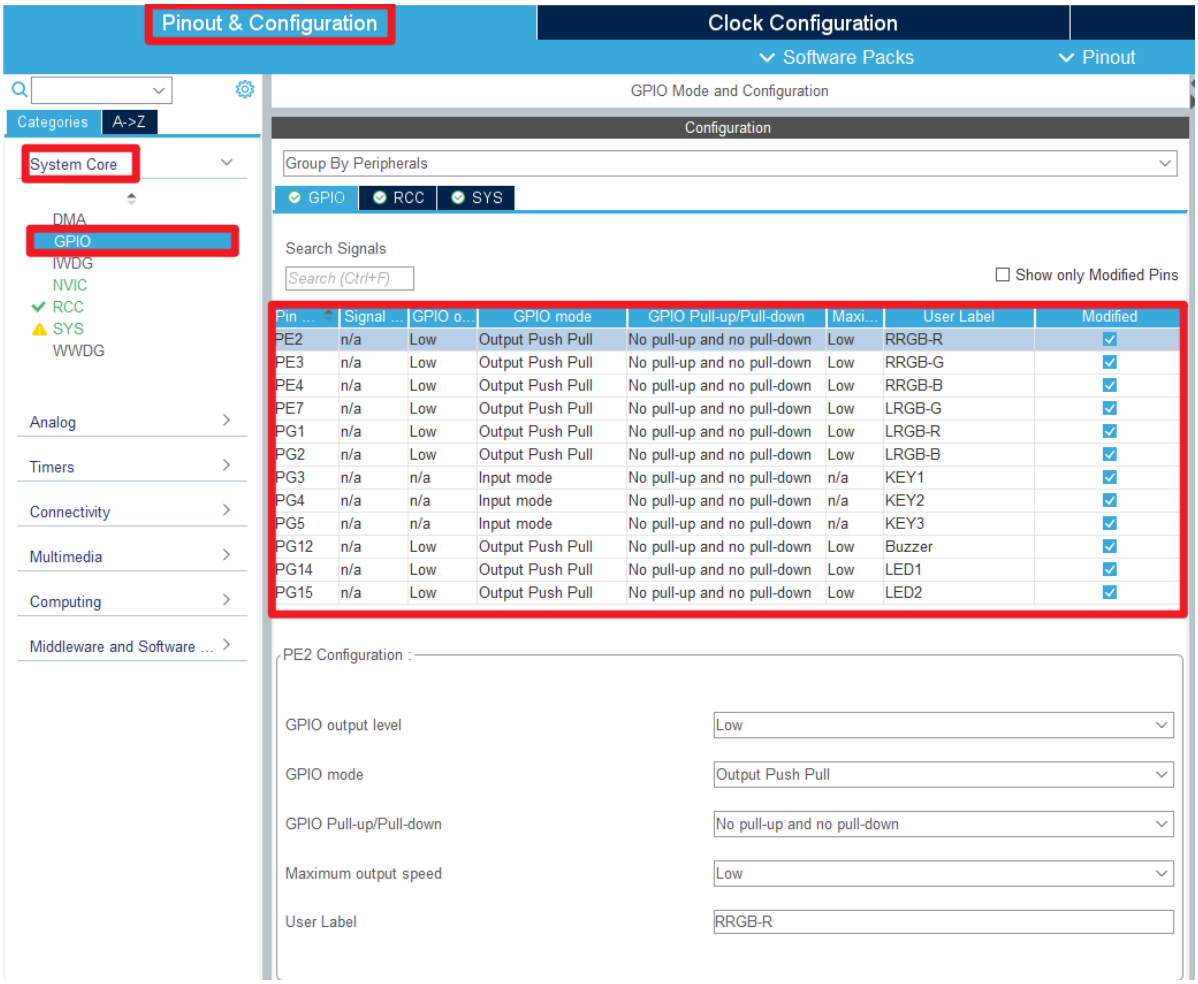
3.2、Pin configuration

- **Configure the specified pin function**

You can directly select the corresponding pin number in the pin view, and the corresponding option will appear when the mouse is left clicked

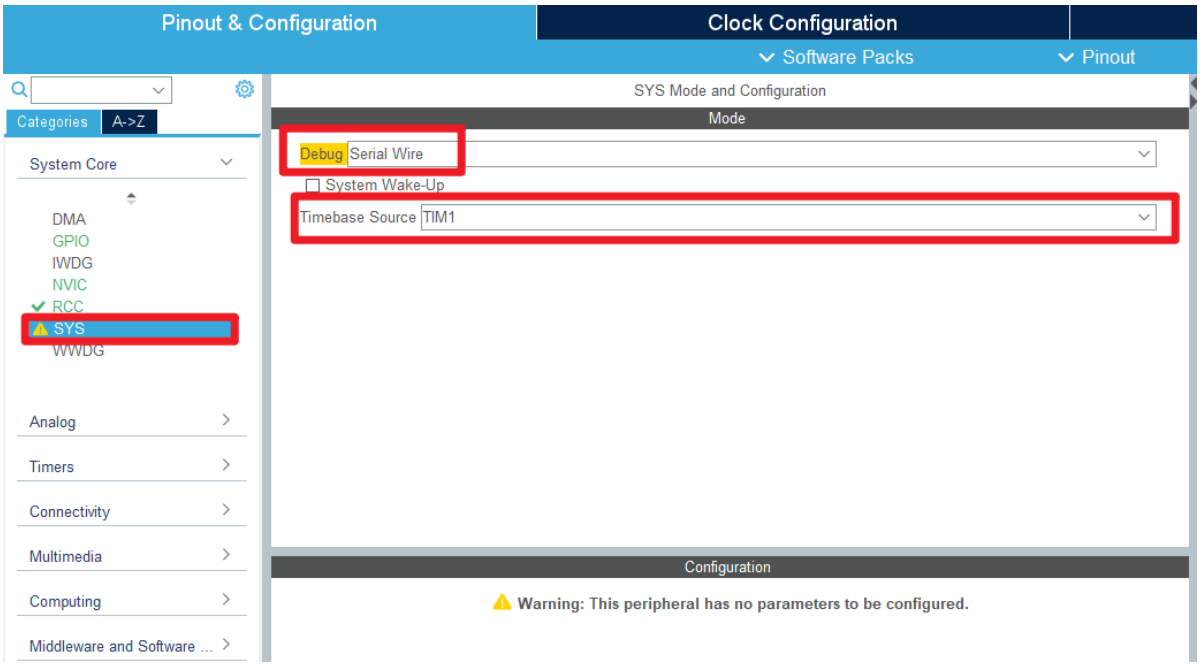

LED2
LED1
Buzzer
SYS_JTCK-SWCLK
R.RGB-R
R.RGB-G
R.RGB-B
SYS_JTMS-SWDIO
KEY3
KEY2
KEY1
LRGB-B
RCC_OSC_IN
RCC_OSC_OUT
LRGB-R
LRGB-G
STM32F103ZETx
LQFP144

- **GPIO**



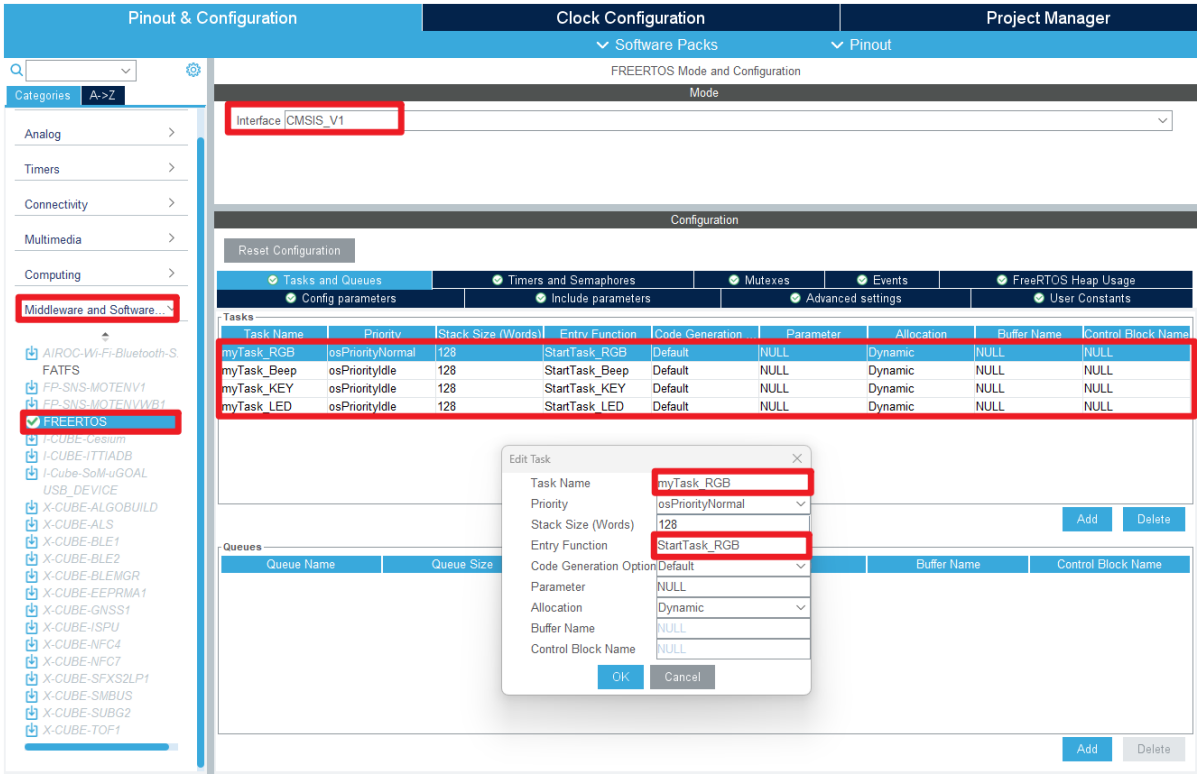
- **SYS**

The reference is changed to TIM1



- **FREERTOS**

The four tasks differ only in task name and task function entity



Task options	feature
Task Name	Task name
Priority	Set priorities
Stack Size (Words)	Heap space
Entry Function	Task function entity
Code Generation Option	Code generation configuration, default is weak generation task entity
Parameter	Task parameters
Allocation	You can choose between Dynamic allocation or Static allocation
Buffer Name	Statically allocated buff name
Control Block Name	The statically allocated block name

4、Main Function

This section mainly introduces the functional code written by users. **Detailed code can be viewed by opening the project file provided by us.** .

4.1、User function

> **function：Task_Entity_RGB**

Function prototypes	void Task_Entity_RGB(void)
Functional Description	RGB task entity function: controls the left and right RGB light color switching
Input parameters	None
Return value	None

> function：Task_Entity_BEEP

Function prototypes	void Task_Entity_BEEP(void)
Functional Description	Buzzer task entity function: Control the buzzer to sound
Input parameters	None
Return value	None

> function：Task_Entity_LED

Function prototypes	void Task_Entity_LED(void)
Functional Description	LED task entity function: Control the left and right RGB light color switching
Input parameters	None
Return value	None

> function：Task_Entity_KEY

Function prototypes	void Task_Entity_KEY(void)
Functional Description	Key task entity function: switch the state of buzzer and LED light
Input parameters	None
Return value	None

4.2、HAL library function parsing

The HAL library functions that were covered in the previous tutorial will not be covered

```
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：MX_FREERTOS_Init

Function prototypes	void MX_FREERTOS_Init(void)
Functional Description	Tasks to initialize FreeRTOS: Create and initialize components such as tasks, queues, mutexes, etc
Input parameters	None
Return value	None

> **function：osKernelStart**

Function prototypes	osStatus osKernelStart (void)
Functional Description	Start the scheduler of the RTOS kernel and start executing tasks
Input parameters	None
Return value	**osStatus**：The result of the function execution

> **function：osDelay**

Function prototypes	osStatus osDelay (uint32_t millisec)
Functional Description	The operating system's latency function
Input parameters	The delay (in ms)
Return value	**osStatus**：The result of the function execution
Tips	Pause the thread, execute another thread, and wait until the delay is complete

5、Experimental phenomenon

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

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

Phenomenon:

Left and right RGB lights：toggle different colors all the time;

KEY1 button：The buzzer sound switch, sound and no sound interval of 1s (press once to open, then press once to close);

KEY2 button：The switch for LED1 (press once to be on and again to be off);

KEY3 button：The LED2 switch (press once to be on and again to be off).

```
The experimental phenomenon can be seen [freertos_experimental Phenomenon.mp4]
```