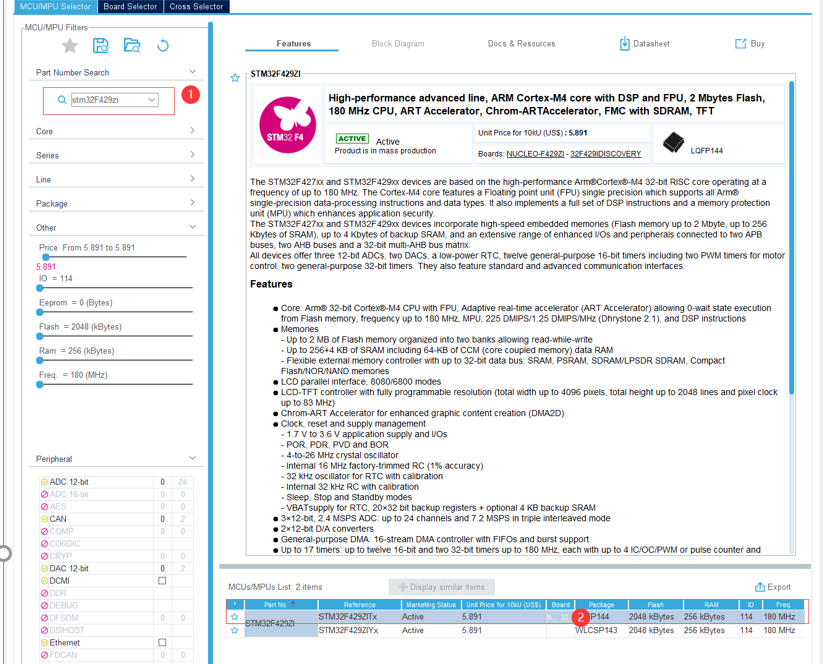
ESD-lab2

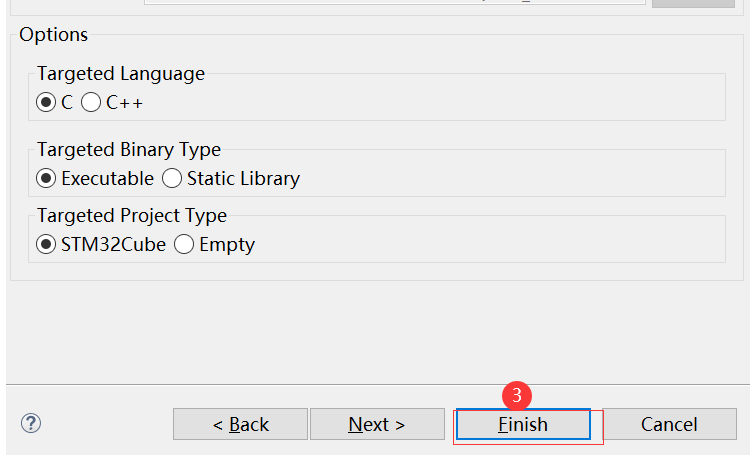
Name:WangYinglun ID: 192050220

# 1 Specific implementation

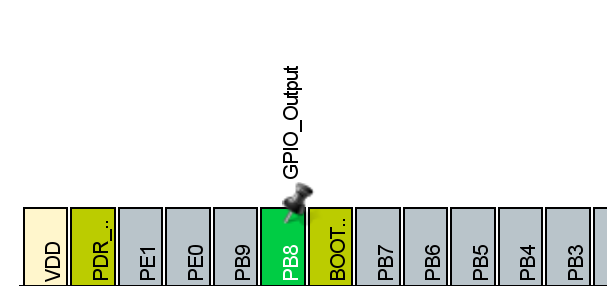
1. Find stm32F429zi to choose our board type

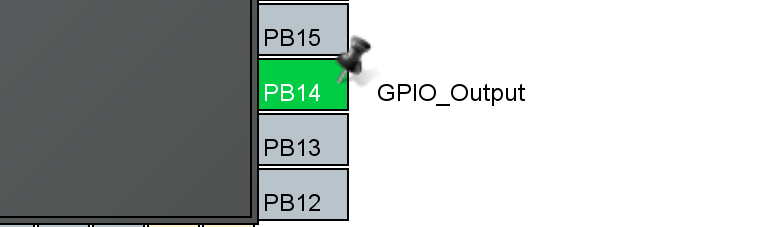


3、Select the name of your project

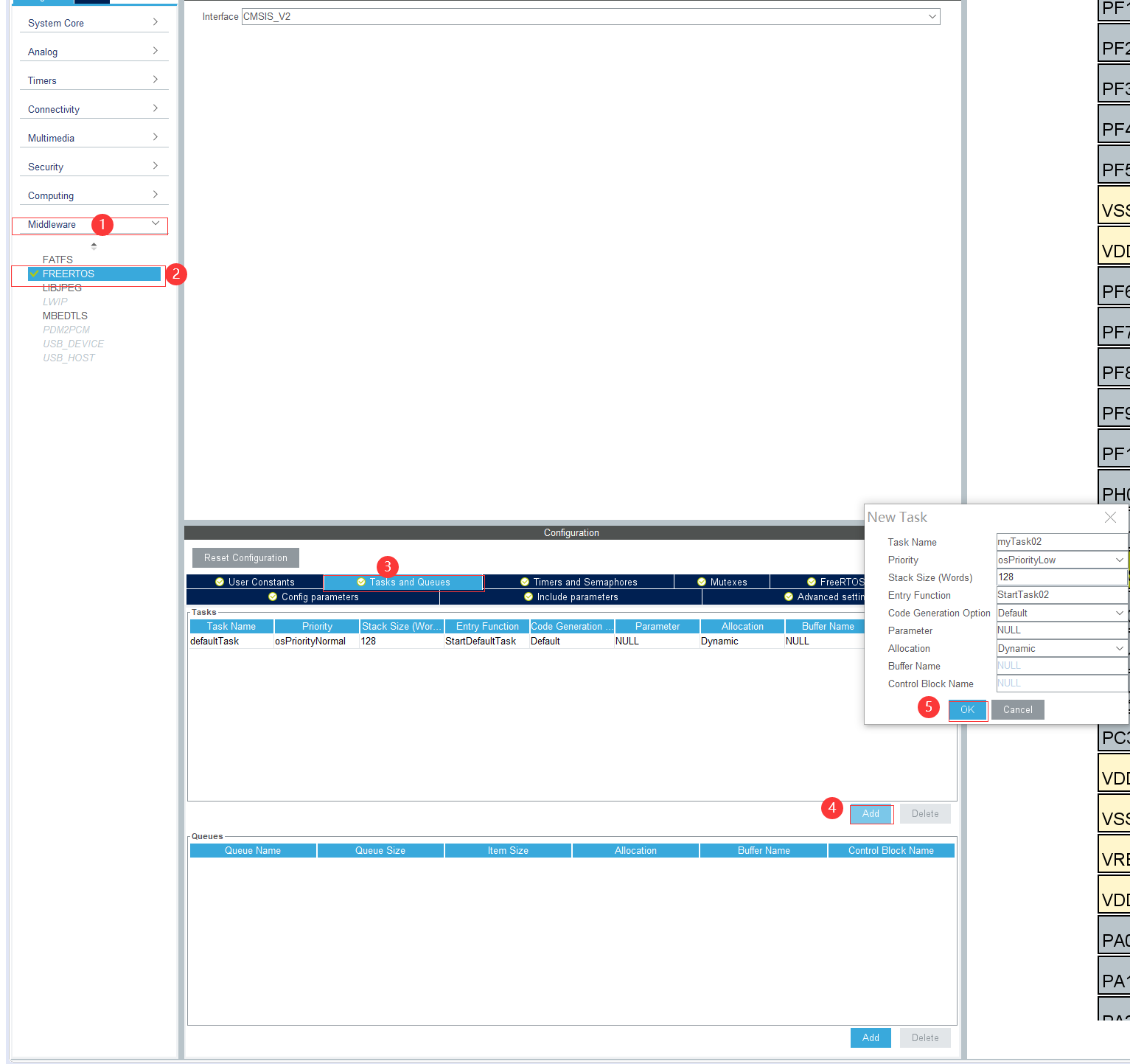


4.Switch IO port to the output.

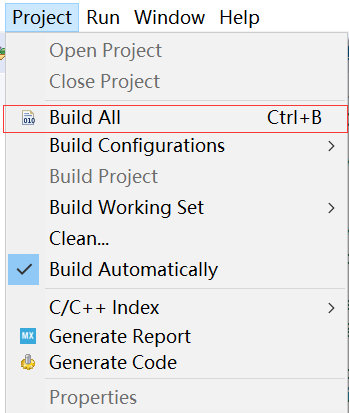


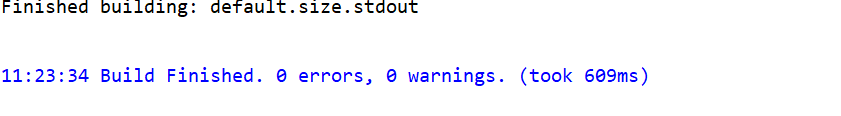


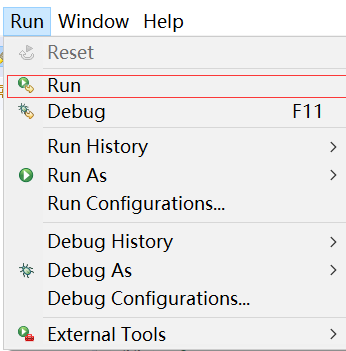
5、Set up Freertos



## 2Compile and run







Demo Results:

https://drive.google.com/file/d/19wePAsXoEdyC8bckK6ICwbdSSI2VYMK\_/view?usp=sharing

# 3Q&A

1. **What features of FreeRTOS do you remember?**

First of all, FreeRTOS is an open source, portable, small, embedded real-time operating system kernel. It supports both pre-built multitasking and collaborative multitasking.

1, users can configure the kernel according to their needs

2, small object code, easy to use

3, Hard real-time operating system kernel, can be modified to non-real-time kernel based on knowledge.

4、Multi-tasking can be supported.

In addition, the features found through the documentation are:

1、FreeRTOS has no limit on the number of tasks, and there can be multiple tasks at the same priority

2、Very flexible task priority assignment, either preemptive or collaborative. The scheduler will decide the task scheduling every time the timed interrupt comes, and external asynchronous events will also cause the scheduler task scheduling

3、FreeRTOS supports queues and several basic task synchronization mechanisms

4、Multi-platform support, open source and free

1. **What are the differences between FreeRTOS, OpenRTOS and SafeRTOS?**

OpenRTOS and FreeRTOS share the same source code. OpenRTOS is a commercial licensed version of FreeRTOS, provided by a third-party Real Time Engineers Ltd. license.

There are two main reasons for users to update from FreeRTOS to OpenRTOS:

1. In order to overcome the GPL license limitation of the modified version of FreeRTOS.

2. In order to obtain additional services, such as professional technical support, high-quality middleware, training, consulting and corresponding tools

SafeRTOS is also based on FreeRTOS, but unlike FreeRTOS, it has been redesigned by security experts.

|  |  |  |
| --- | --- | --- |
|  | FreeRTOS Open Source License |  |
| Is it free | Yes | No |
| Can I use it in a commercial application | Yes | Yes |
| Is it royalty free | Yes | Yes |
| Is a warranty provided | No | Yes |
| Can I receive professional technical support on a commercial basis? | No,FreeRTOS is supported by an online community | Yes,IP infringement profection is provided |
| Is legal protection provided | No | No |
| Do I have to open source my application code that makes use of the FreeRTOS services | No | No |
| Do I have to open source my changes to the RTOS kernel | Yes | No |
| Do I have to document that my product uses FreeRTOS | Yes.if you distribute source code | No |
| Do I have to offer to provide the FreeRTOS code to users of my application? | Yes,if you distribute source code | No |

1. **Why do we need the vTaskStartScheduler() function?**

The function vTaskStartScheduler() is used to start the task scheduler and start scheduling tasks. FreeRTOS is started by the vTaskStartScheduler() function, which is usually encapsulated in osKernelStart(). Its work is as follows:

(1) xTaskCreate() creates the idle task, priority minimum: 0.

(2) Turn off the interrupt function and enable the task scheduler function.

(3) macro definition portCONFIGURE\_TIMER\_FOR\_RUN\_TIME\_. STATS: Initialize the system runtime statistics.

(4)Set the system tick timer to start the first task.

(5)Return the idle task handle.

4) **Why do we need the xTaskCreate() function?**

In the case of using FreeRTOS-MPU, using the xTaskCreate() function allows you to create a Tasks that run in privileged or user mode. So we need to use the xTaskCreate() function to create a new task and add it to the task queue in, ready to run. If the task is successfully created and added to the ready list, the function returns pdPASS, otherwise the function returns an error code.