Embedded Systems Design

Developing tasks in FreeRTOS

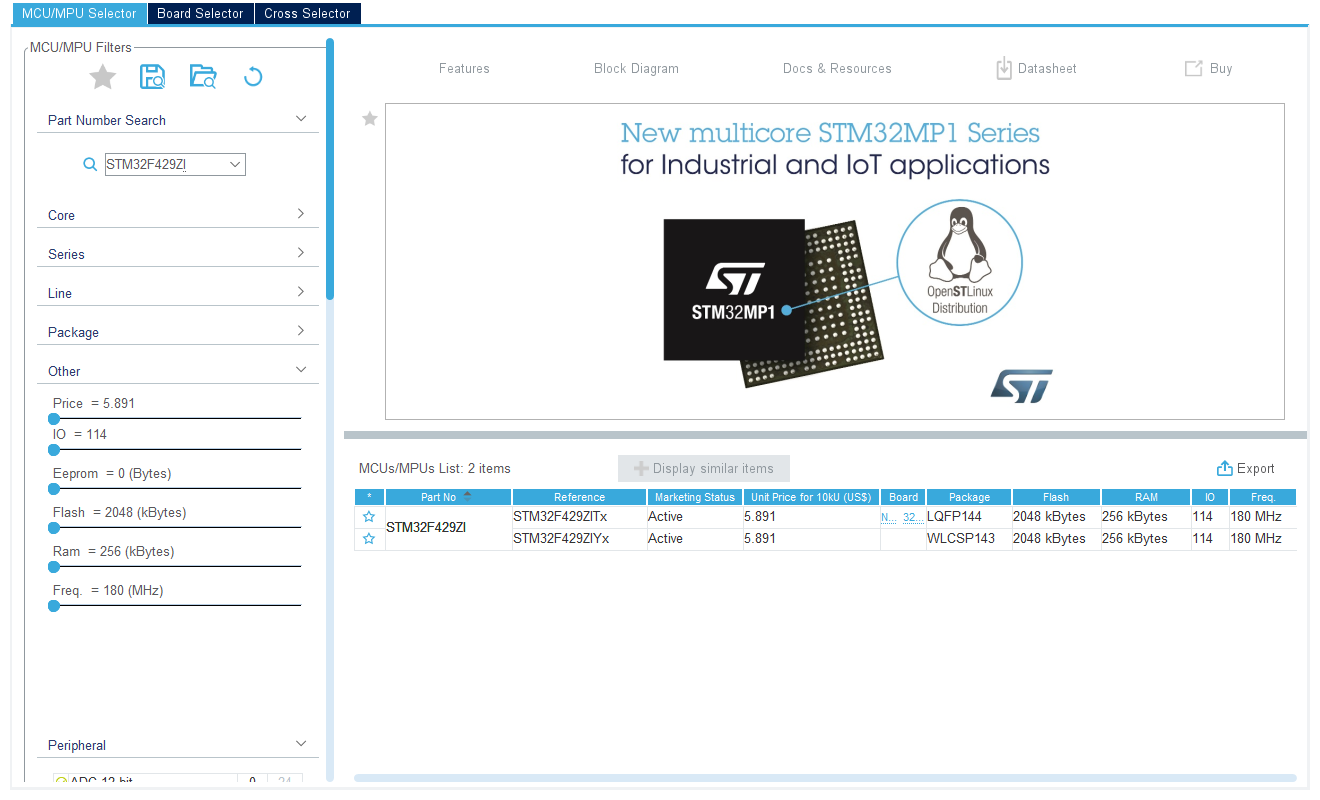
Name：孙煨 Sun\_Wei

No.192050207

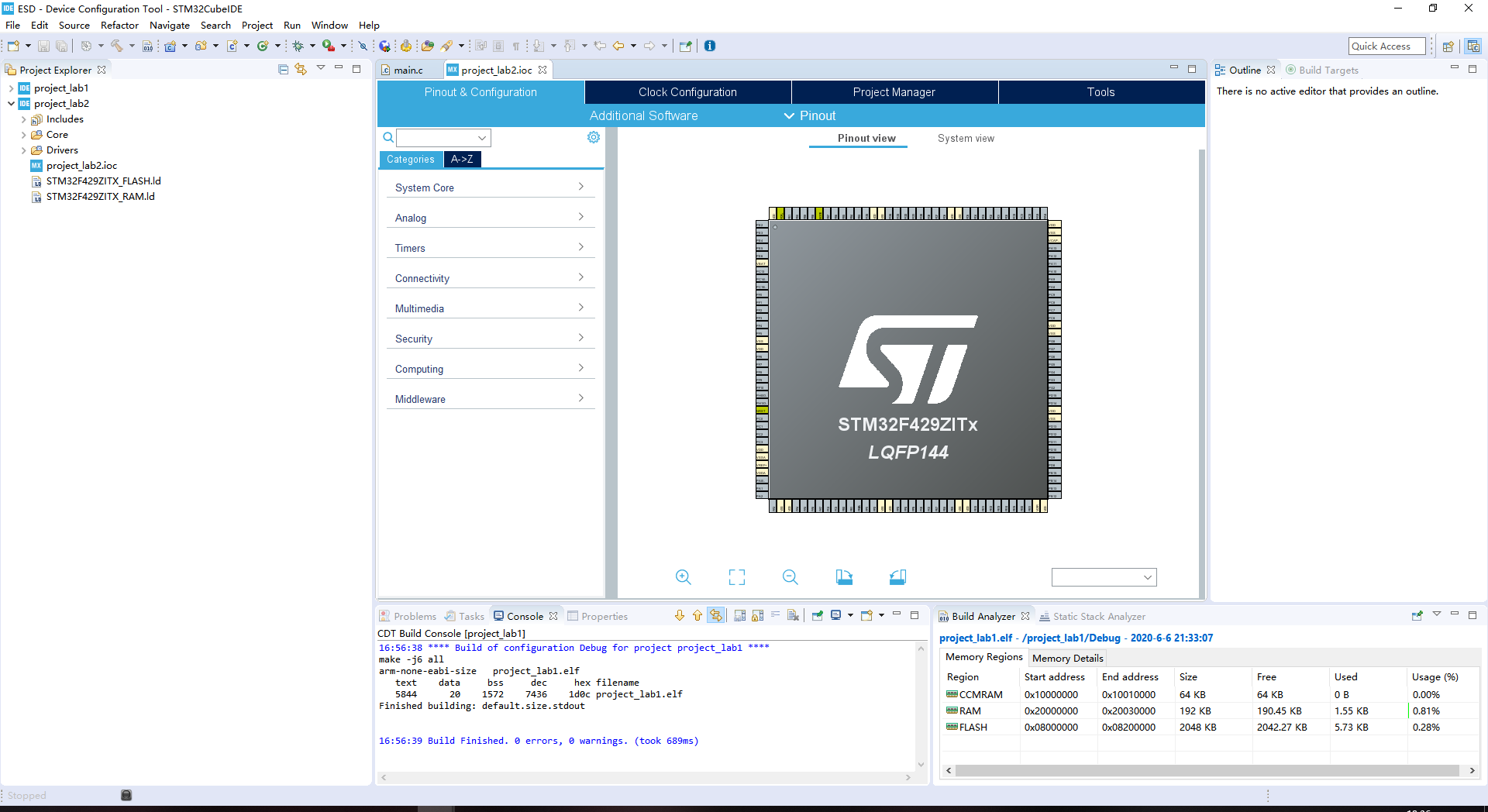
Date: 2020-6-17

1. The creation of project

I search the STM32F429 and choose the first whose package is LQFP144.

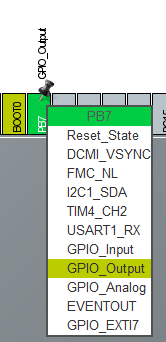
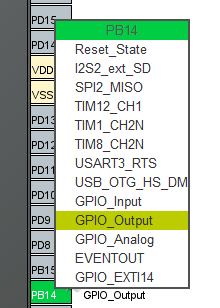


After the set up, I can see the following picture.



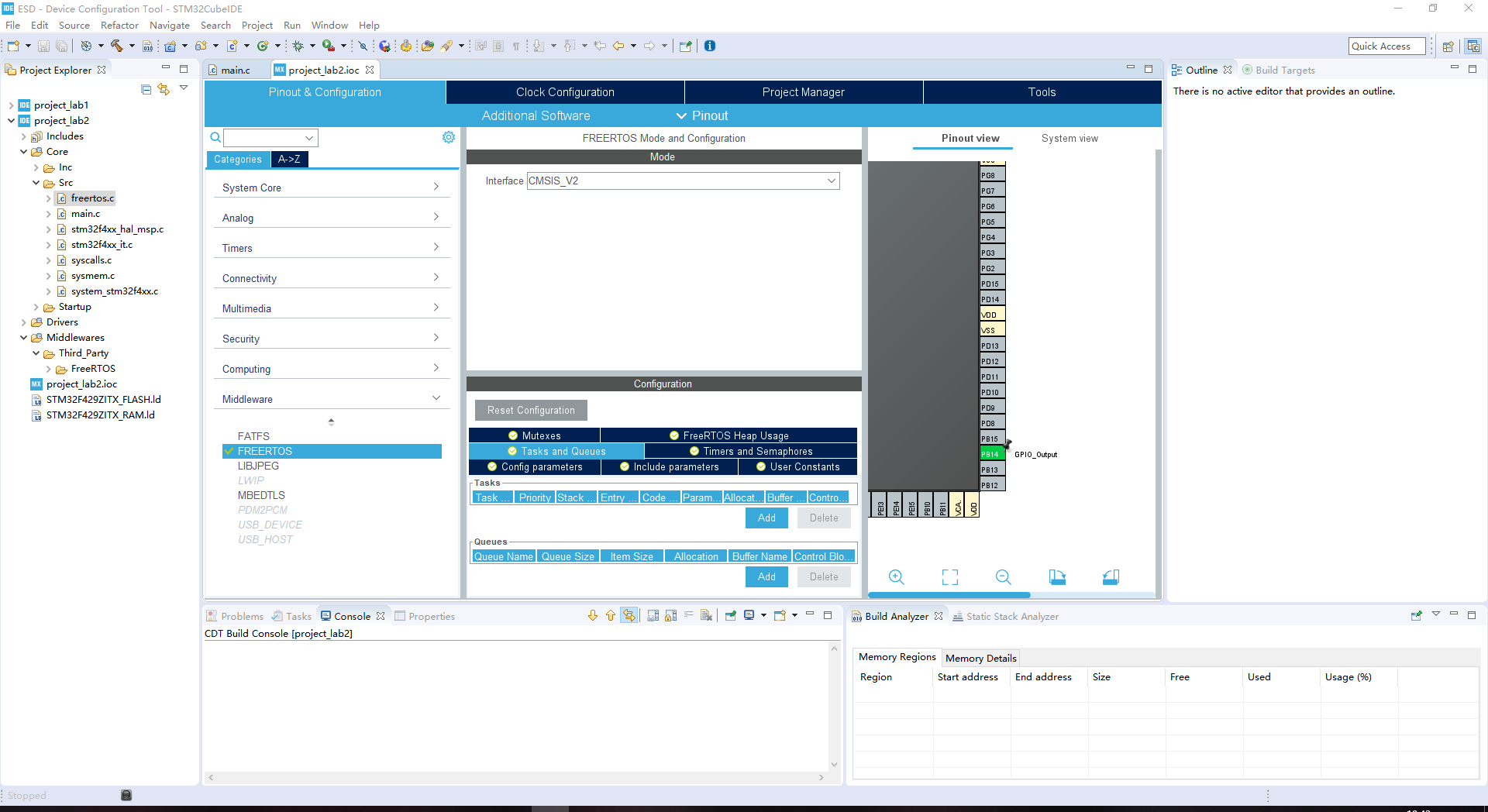
1. Set

I set the PB7 and PB14 to GPIO\_OUTPUT and save it.

1. Set FreeRTOS

Then I clink Middleware->FREERTOS, and select the Interface to CMSIS\_V2.After that, I press ALT+K to generate code.

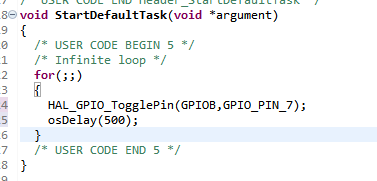


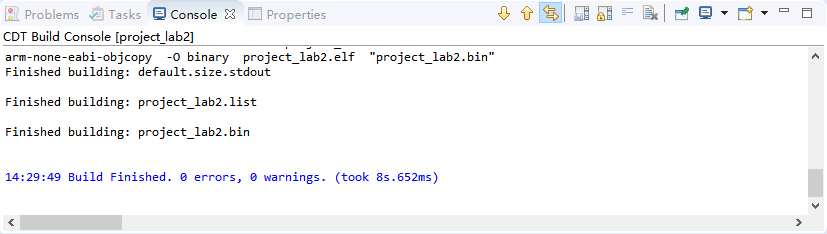
1. Task1

I write some codes in main.c and click build all.I click run and I can see the blue LED twinkle.

HAl\_GPIO\_TogglePin(GPIOB,GPIO\_PIN\_7) means flip state.

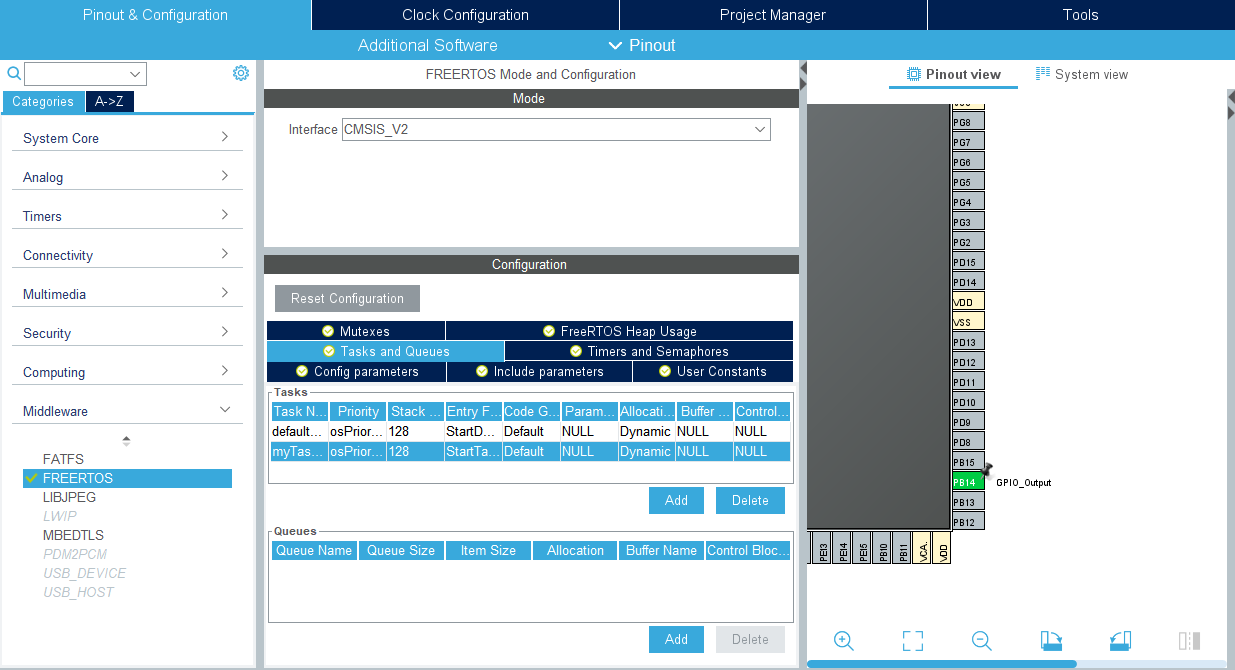
HAL\_Delay(500) means delay 500 ms.





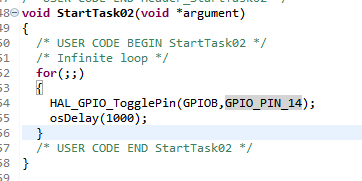
1. Task2

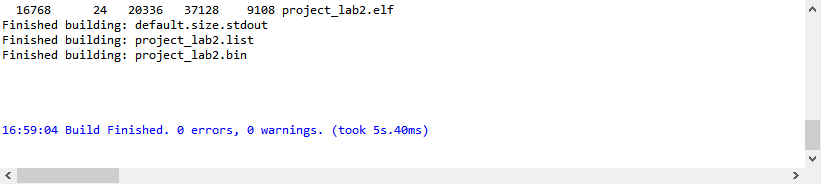
I click the FreeRTOS in project\_lab2.ioc and add a task.



After that, I click ‘generate code’ and go to main.c.I write the code into main.c and build all again.Then I run it again.I can see the blue LED twinkle per 500ms,the red LED twinkle per 1000ms.

HAl\_GPIO\_TogglePin(GPIOB,GPIO\_PIN\_14) means flip level.





1. Questions

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

Answer:Pre-emptive or co-operative operation

 Very flexible task priority assignment

 Flexible, fast and light weight task notification mechanism

 Queues

 Binary semaphores

 Counting semaphores

 Mutexes

 Recursive Mutexes

 Software timers

 Event groups

 Tick hook functions

 Idle hook functions

 Stack overflow checking

 Trace recording

 Task run-time statistics gathering

Optional commercial licensing and support

 Full interrupt nesting model (for some architectures)

 A tick-less capability for extreme low power applications

 Software managed interrupt stack when appropriate (this can help save RAM)

**6.2What are the differences between FreeRTOS, OpenRTOS and SafeRTOS?**

Answer:FreeRTOS: It can be used in commercial applications.It remains freely available to everybody. Its users retain ownership of their intellectual property.

OpenRTOS is a commercially licensed version of FreeRTOS provided under license from Real Time Engineers Ltd. by a third party.

SafeRTOS shares the same usage model as FreeRTOS, but has been developed in

accordance with the practices, procedures, and processes necessary to claim compliance with various internationally recognized safety related standards.

**6.3Why do we need the vTaskStartScheduler() function?**

The function is used to starts the FreeRTOS scheduler running.before the scheduler has been started, main() will be executing. After the scheduler has been started, only tasks and interrupts will ever execute.Starting the scheduler causes the highest priority task that was created while the scheduler was in the Initialization state to enter the Running state.

Its features are:1.Create an idle task with xTaskCreate();

2.Close the interrupt function and enable the task scheduling function;

3.Set the system tick timer and start the first task;

4.Return the idle task handle;

5.Initialization of system running time statistics;

So if we want to run the FreeRTOS program, it is indispensable.

**6.4Why do we need the xTaskCreate() function?**

The function is used to creates a new instance of a task.Each task requires RAM that is used to hold the task state (the task control block, or TCB),and used by the task as its stack. If a task is created using xTaskCreate() then the required RAM is automatically allocated from the FreeRTOS heap. Newly created tasks are initially placed in the Ready state, but will immediately become the running state task if there are no higher priority tasks that are able to run.

Our aim is creating two tasks.So it is indispensable.