4. FreeRTOS application

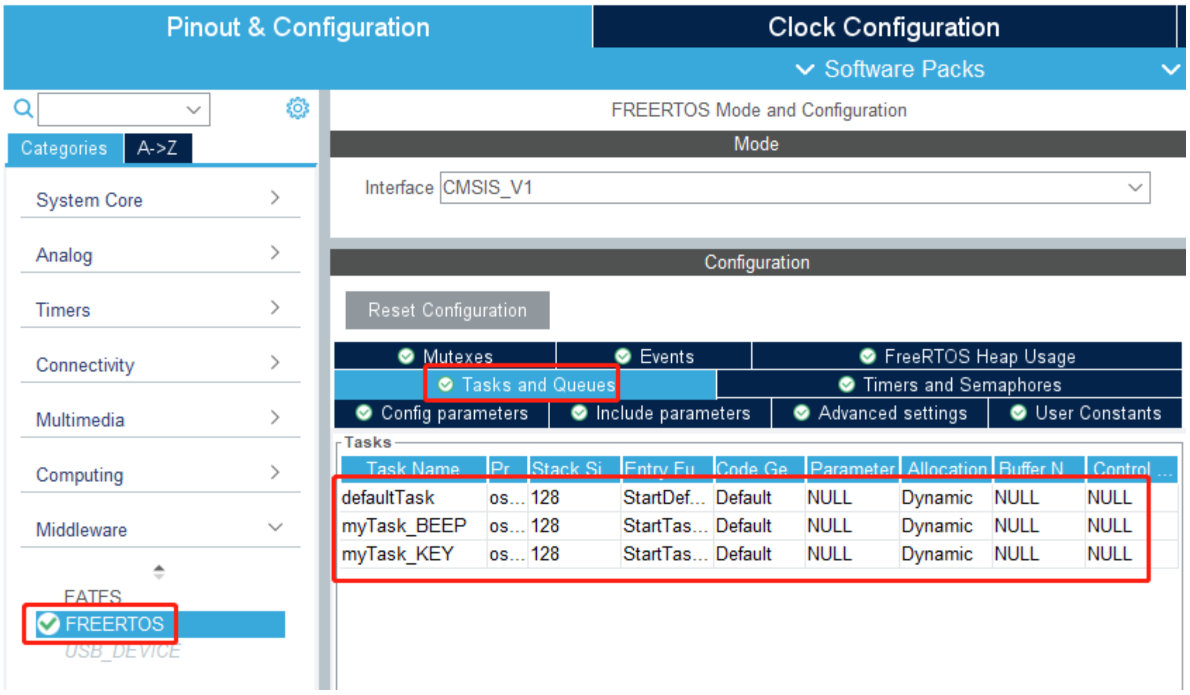
4.1. Purpose of the experiment

Based on the program of "Key Control Buzzer Whistle", the function is imported into the FreeRTOS system to run, to detect the state of KEY1 on the expansion board, and to control the buzzer to whistle. Press the button once, the buzzer will beep (every 200 milliseconds), press the button again, the buzzer will be turned off.

4.2. Configure FreeRTOS information

Since each new project needs configuration information, it is more troublesome. Fortunately, STM32CubeIDE provides the function of importing .ioc files, which can help us save time.

1. Import the ioc file from the BEEP project and name it FreeRTOS.
2. Click Middleware->FreertOS, select CMSIS_V1, click Tasks and Queues, there will be one task by default, and then create two new tasks, one of which manages the buzzer and the other manages the buttons.

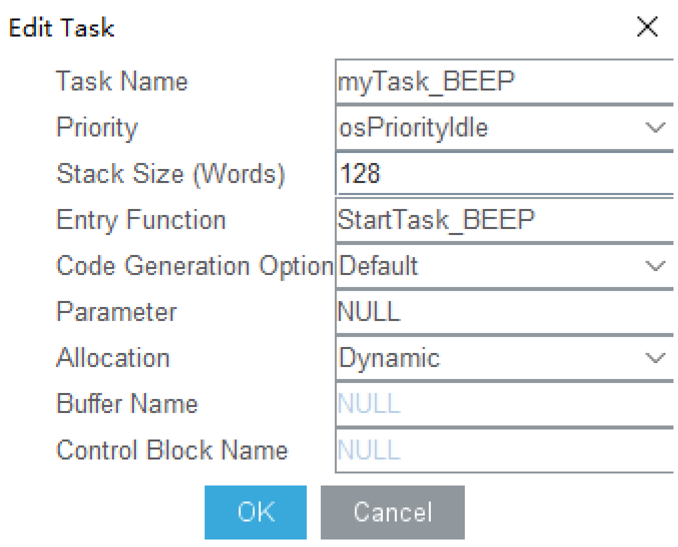


Interface CMSIS_V1

Tasks

Task Name	Pr	Stack Si	Entry Fu	Code Ge	Parameter	Allocation	Buffer N	Control ...
defaultTask	os...	128	StartDef...	Default	NULL	Dynamic	NULL	NULL
myTask_BEEP	os...	128	StartTas...	Default	NULL	Dynamic	NULL	NULL
myTask_KEY	os...	128	StartTas...	Default	NULL	Dynamic	NULL	NULL

3. The buzzer task information is shown in the following figure:


Edit Task

Task Name	myTask_BEEP
Priority	osPriorityIdle
Stack Size (Words)	128
Entry Function	StartTask_BEEP
Code Generation Option	Default
Parameter	NULL
Allocation	Dynamic
Buffer Name	NULL
Control Block Name	NULL

Task Name: The task name.

Priority: Set the priority.

Stack Size: heap space, the size can be modified according to the actual situation.

Entry Function: The task function entity.

Code Generation Option: Code generation configuration, the default is weak to generate task entities, you can choose external not to generate task entities.

Parameter: The task parameter.

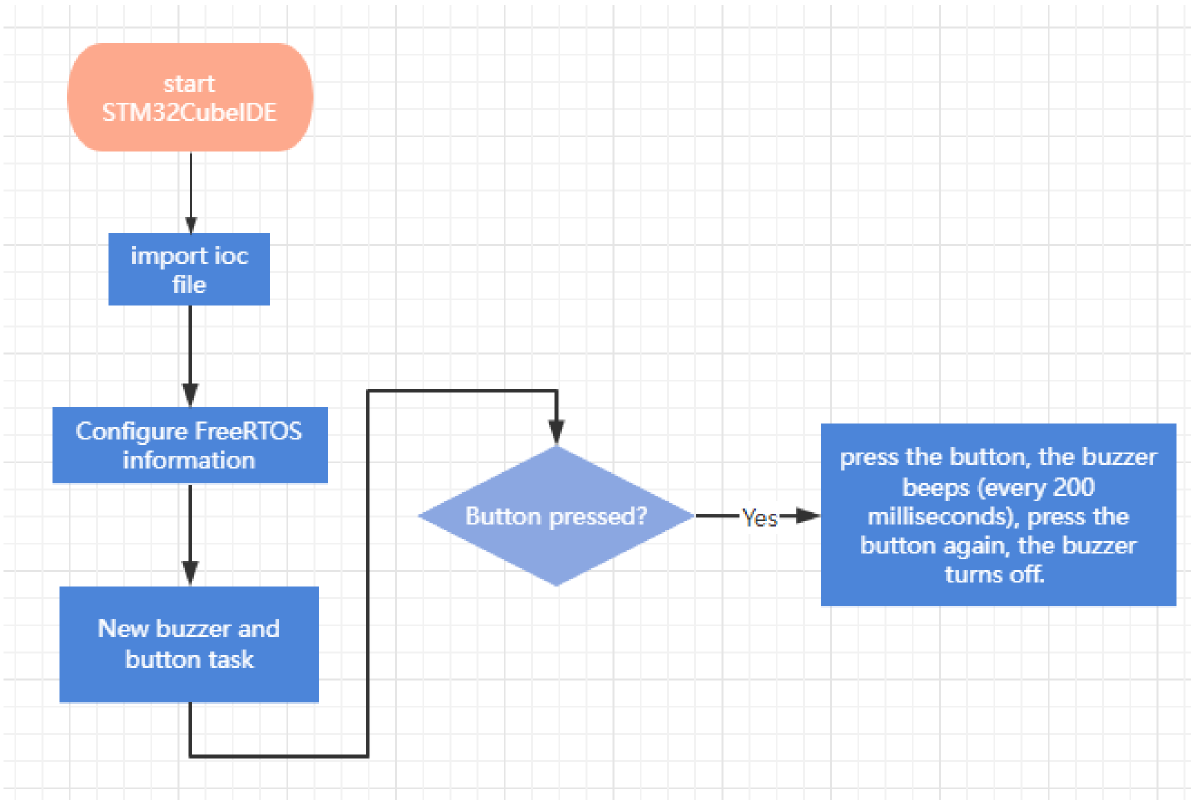
Allocation: You can choose Dynamic dynamic allocation or Static static allocation.

Buffer Name: The statically allocated buff name.

Control Block Name: Statically allocated block name.

The key task is the same, just with a different name.

4.3. Analysis of the experimental flow chart


start
STM32CubeIDE
import ioc file
Configure FreeRTOS information
New buzzer and button task
Button pressed?
Yes
press the button, the buzzer beeps (every 200 milliseconds), press the button again, the buzzer turns off.

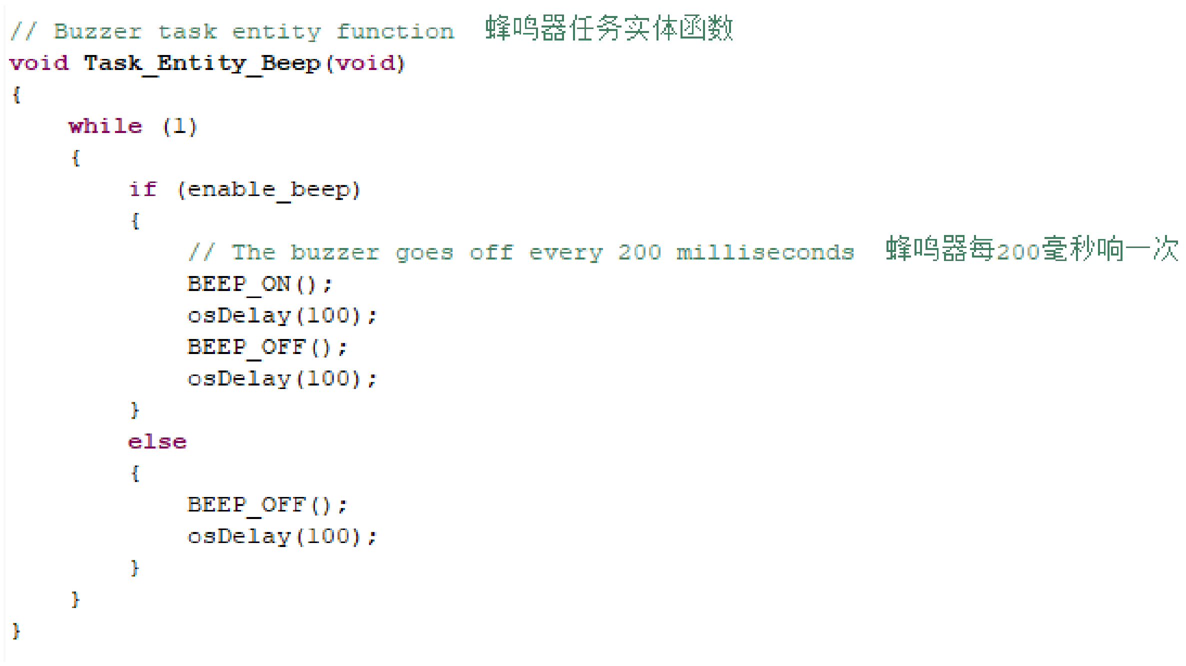
4.4. core code explanation

1. Create new buzzer driver library bsp_task.h and bsp_task.c files in BSP. Add the following to bsp_task.h:

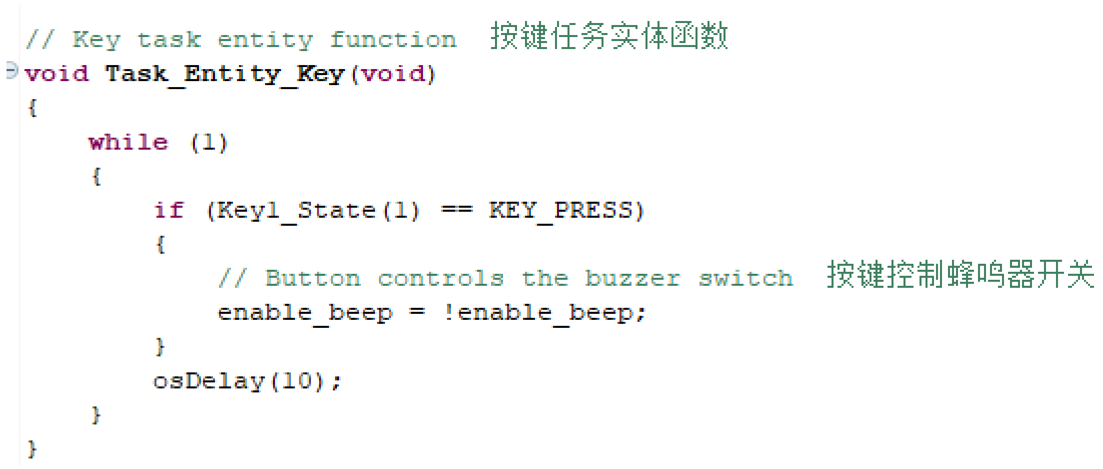
```
void Task_Entity_LED(void);
void Task_Entity_Beep(void);
void Task_Entity_Key(void);
```

Among them, the Task_Entity_LED() function manages the LED lights, the Task_Entity_Beep() manages the buzzer, and the Task_Entity_Key() manages the buttons.

```
// LED light task entity function  LED灯任务实体函数
void Task_Entity_LED(void)
{
    while (1)
    {
        // The indicator lights up every 100 milliseconds  指示灯每隔100毫秒亮一次
        LED_TOGGLE();
        osDelay(100);
    }
}
```

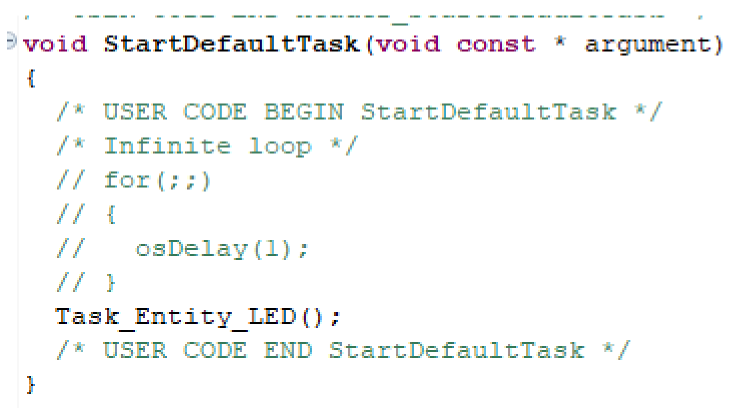


```
// Buzzer task entity function  蜂鸣器任务实体函数
void Task_Entity_Beep(void)
{
    while (1)
    {
        if (enable_beep)
        {
            // The buzzer goes off every 200 milliseconds  蜂鸣器每200毫秒响一次
            BEEP_ON();
            osDelay(100);
            BEEP_OFF();
            osDelay(100);
        }
        else
        {
            BEEP_OFF();
            osDelay(100);
        }
    }
}
```

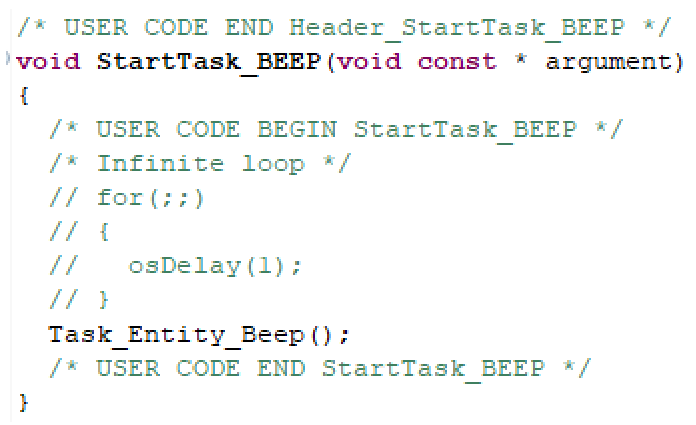


```
// Key task entity function  按键任务实体函数
void Task_Entity_Key(void)
{
    while (1)
    {
        if (Key1_State(1) == KEY_PRESS)
        {
            // Button controls the buzzer switch  按键控制蜂鸣器开关
            enable_beep = !enable_beep;
        }
        osDelay(10);
    }
}
```

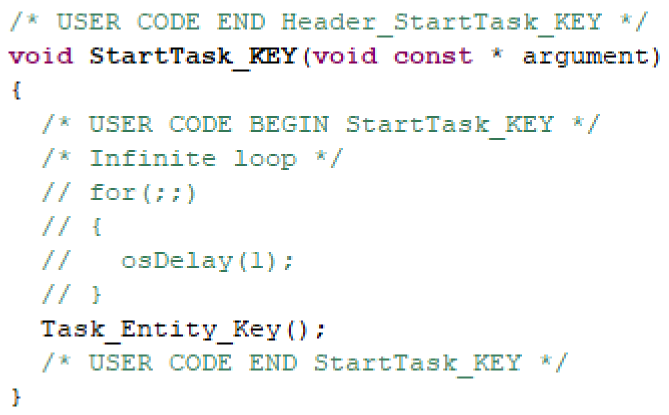
2. Introduce bsp.h into the freertos.c file, find the corresponding entity functions of the three tasks, and call the task functions we manually created.



```
void StartDefaultTask(void const * argument)
{
  /* USER CODE BEGIN StartDefaultTask */
  /* Infinite loop */
  // for(;;)
  // {
  //   osDelay(1);
  // }
  Task_Entity_LED();
  /* USER CODE END StartDefaultTask */
}
```



```
/* USER CODE END Header_StartTask_BEEP */
void StartTask_BEEP(void const * argument)
{
  /* USER CODE BEGIN StartTask_BEEP */
  /* Infinite loop */
  // for(;;)
  // {
  //   osDelay(1);
  // }
  Task_Entity_Beep();
  /* USER CODE END StartTask_BEEP */
}
```



```
/* USER CODE END Header_StartTask_KEY */
void StartTask_KEY(void const * argument)
{
  /* USER CODE BEGIN StartTask_KEY */
  /* Infinite loop */
  // for(;;)
  // {
  //   osDelay(1);
  // }
  Task_Entity_Key();
  /* USER CODE END StartTask_KEY */
}
```

4.5. Hardware connection

The LED light, key KEY1 and buzzer in FreeRTOS application are all onboard components and do not need to be connected manually.

4.6. Experimental effect

After the program is programmed, the LED light flashes every 200 milliseconds, press the button, the buzzer beeps (every 200 milliseconds), press the button again, the buzzer turns off.