4. FreeRTOS application

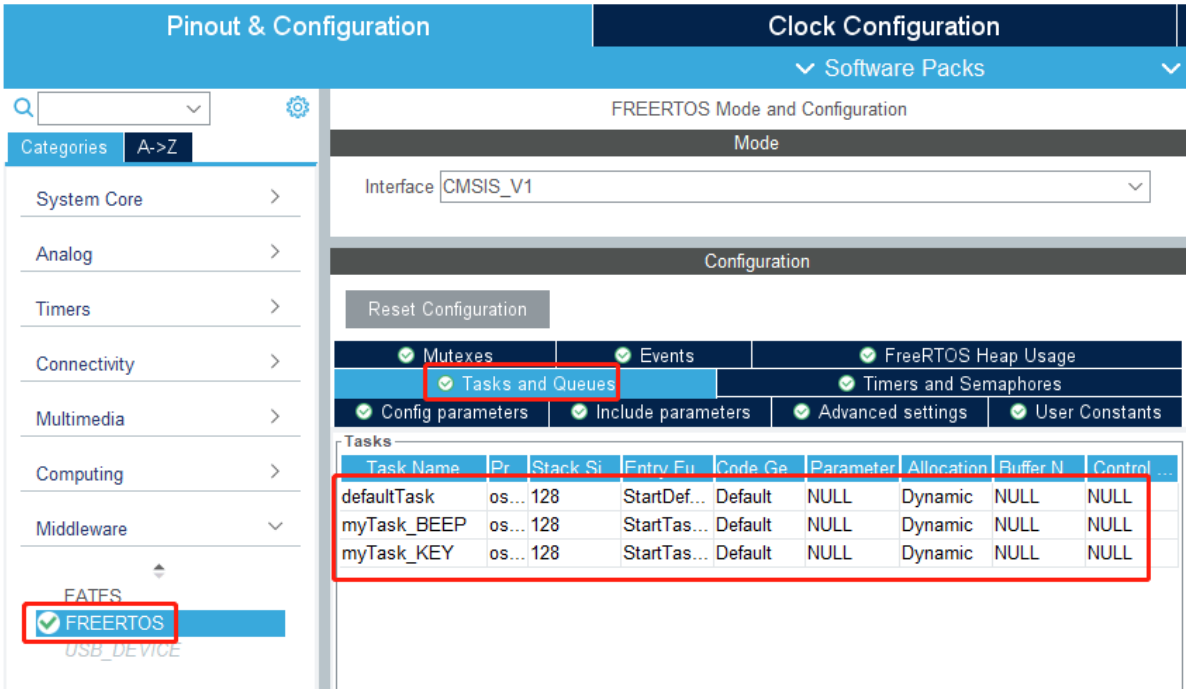
4.1. Experimental purpose

On the basis of the "key control buzzer buzzer" program, the function is imported to the FreeRTOS system to detect the KEY1 status on the expansion board and control the buzzer buzzer buzzer. Press the button, the buzzer drops (every 200 milliseconds), press the button again, the buzzer off.

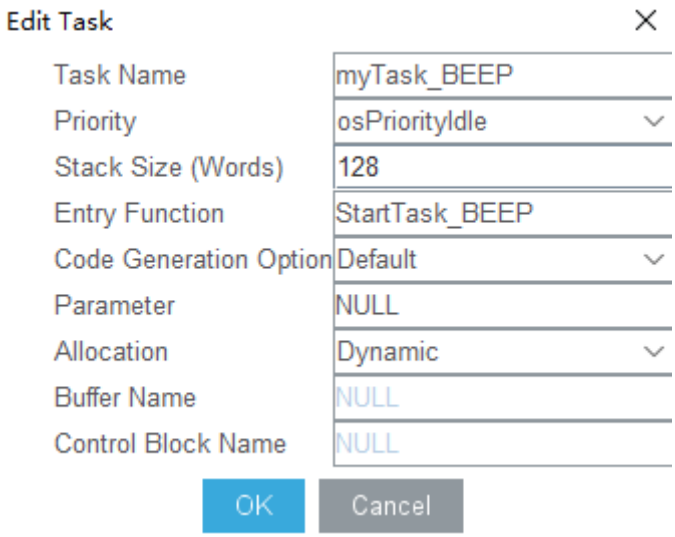
4.2. Configure FreeRTOS information

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

1. Import ioc file from BEEP project and name it FreeRTOS.
2. Click Middleware->FREERTOS, select CMSIS_V1, click Tasks and Queues, there will be a task here by default, and create two new tasks, one to manage the buzzer and the other to manage the keys.



3. The buzzer task information is shown as follows:



Task Name: The name of the task.
Priority: Set the priority.

Stack Size: Heap space, can be modified according to the actual size.
Entry Function: task function entity.

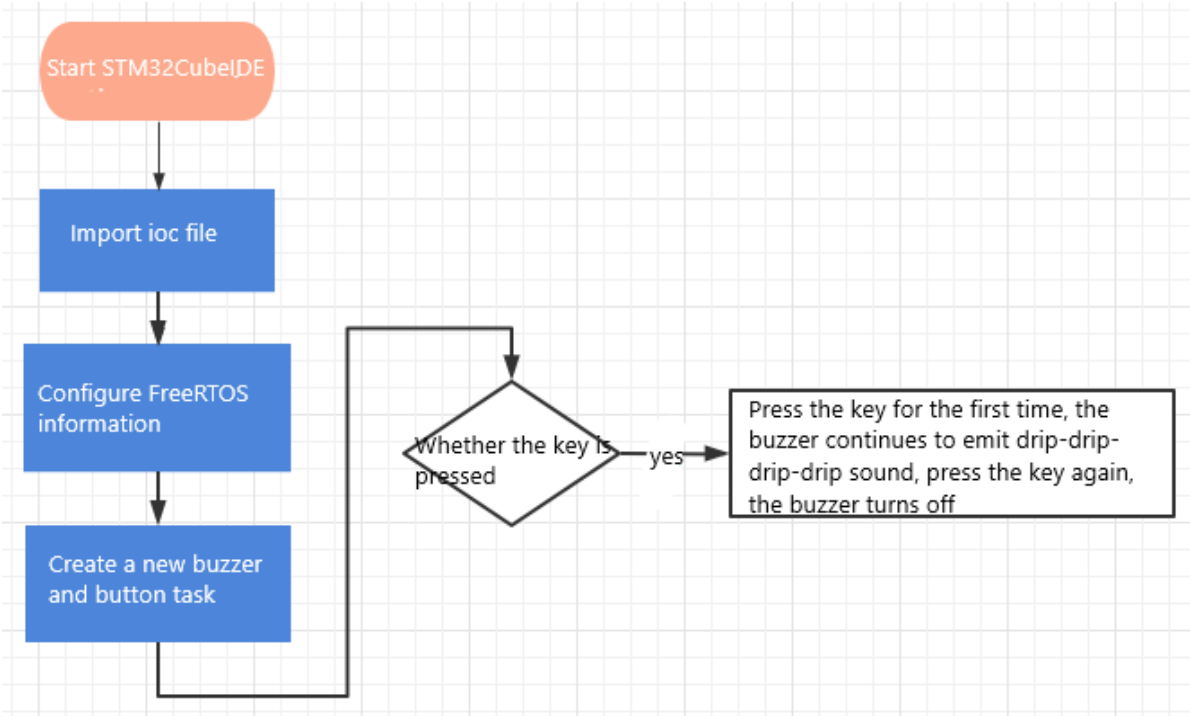
Code Generation Option: indicates the code generation configuration. The default value is weak to generate task entities. You can select external to not generate task entities.

Parameter: Task parameters.

Allocation: Dynamic or Static allocation can be selected.
Buffer Name: The name of the statically allocated buff.
Control Block Name: statically allocated block name.

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

4.3. Analysis of experimental flow chart



4.4. Core code interpretation

1. Create the bsp_task.h and bsp_task.c driver libraries of the buzzer in the BSP. Add the following to bsp_task.h:

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

Task_Entity_LED() manages the LED light, Task_Entity_Beep() manages the buzzer, and Task_Entity_Key() manages the key.

```
// 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 (Keyl_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. Import bsp.h into freertos.c, find the entity functions of the corresponding three tasks, and call the manually established task functions respectively.

```
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

LED lights, keys KEY1 and buzzer in FreeRTOS application are onboard components and do not need to be manually connected.

4.6. Experimental effect

After burning the program, the LED light flashes every 200 milliseconds, press the key, the buzzer drops and drops (every 200 milliseconds), press the key again, the buzzer is off.