实验12：FreeRTOS

一.实验器材

硬件：ARM-STM32开发板，J-Link/St-Link。

软件：Win10, STM32CubeIDE，HAL library Using mail queues to solve the producer-consumer problem

二.实验要求

1、Suppose the buffer size of the Producer–consumer problem is 4. Try to use counting semaphore to solve it. Remember to set the priority of the Consumer task as osPriorityBelowNormal to make sure the Producer task executed first. Try to figure out what the output should be and why before running the program, and compare it with the actual output.

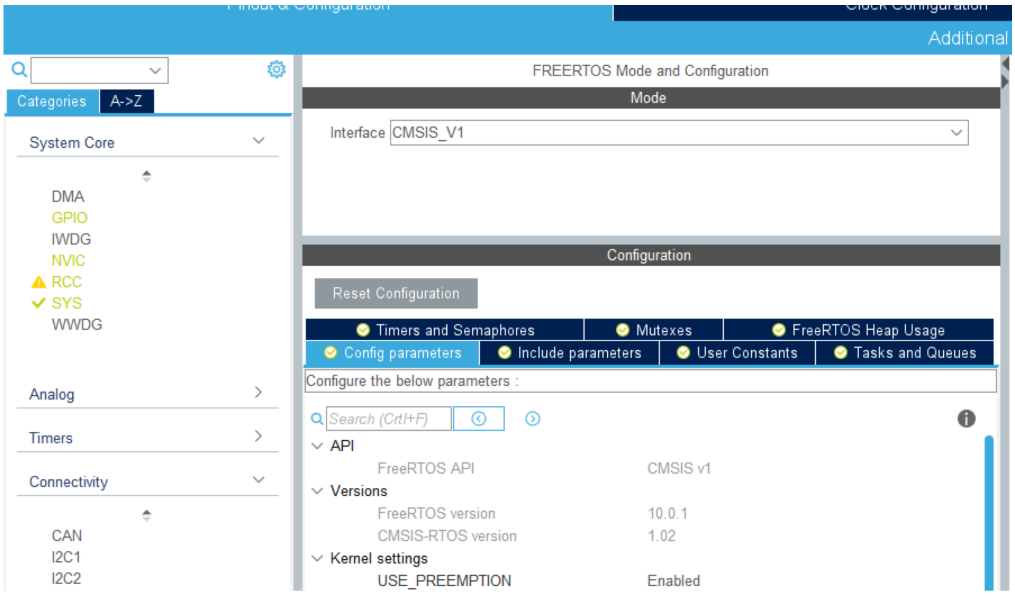
2、Using mail queues to solve the producer-consumer problem.

3、The buffer size of the producer-consumer problem is 4.

三.实验过程

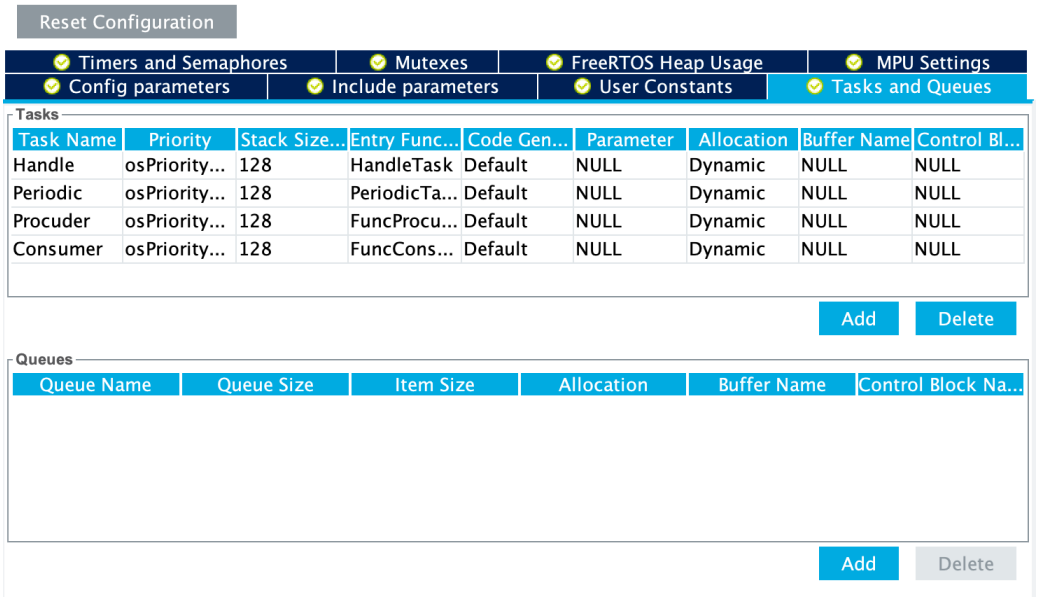
1、Lab11

- set the FreeRTOS API as CMSIS v1.



- Go to the Tasks and Queues, add two tasks Consumer & Producer and a counting semaphore

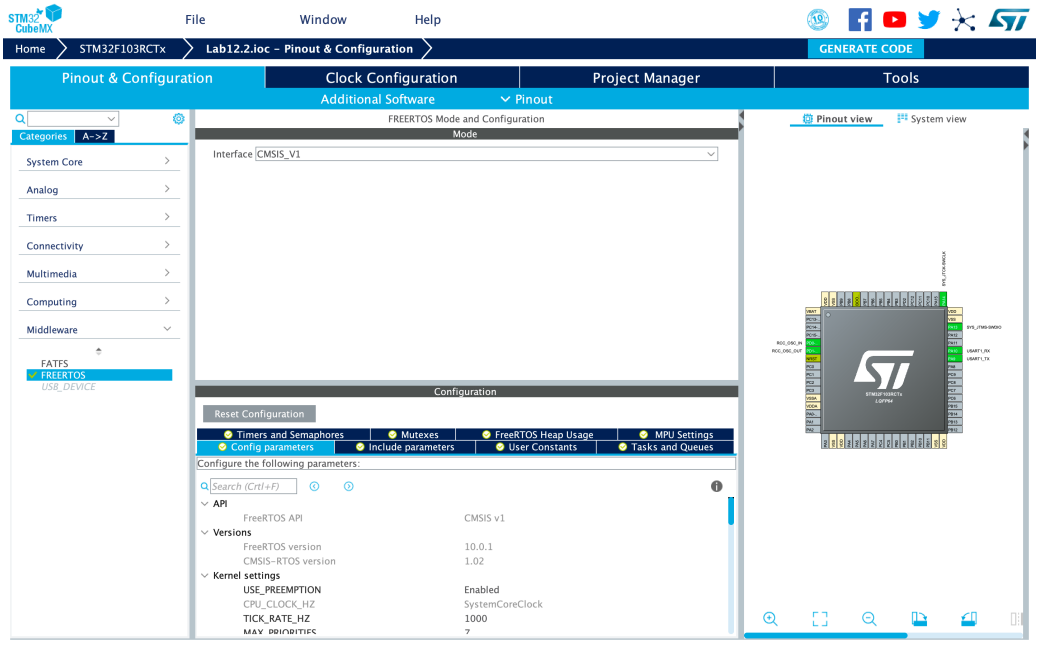
姓名：胡玉斌　　学号：11712121　　同组人:武羿　　日期：2019/12/04



- Generate code and add the code.

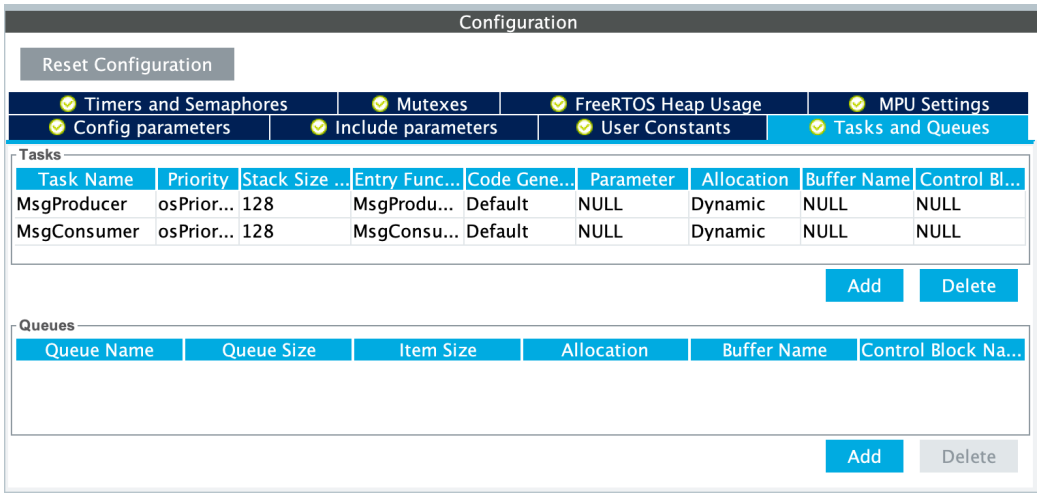
2、Lab12

- set the FreeRTOS API as CMSIS v1.



姓名：胡玉斌　　学号：11712121　　同组人:武羿　　日期：2019/12/04

- Go to the Tasks and Queues, add two tasks MsgConsumer & MsgProducer and a counting semaphore



- Generate code and add the code.

四.实验结果

软件代码（只需用户实现功能的主要代码部分）：

1. Lab11

FuncProductor:

```
void FuncProcuder(void const * argument)
{
 /* USER CODE BEGIN FuncProcuder */
 /* Infinite loop */
 for(;;)
 {
  osSemaphoreWait(bSemEmptyHandle, osWaitForever);  //等待bSemEmptyHandle 信息
  sprintf(msg, "Producer produce data\r\n");
  HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
  HAL_Delay(500);
  osSemaphoreRelease(bSemFilledHandle);  // 释放bSemFilledHandle信息
  osDelay(100);
 }
 /* USER CODE END FuncProcuder */
}
```


姓名：胡玉斌　　学号：11712121　　同组人:武羿　　日期：2019/12/04

FuncConsumr:

```
void FuncConsumer(void const * argument)
{
 /* USER CODE BEGIN FuncConsumer */
 /* Infinite loop */
 for(;;)
 {
  osSemaphoreWait(bSemFilledHandle, osWaitForever);//等待bSemFilledHandle 信息
  sprintf(msg, "Consumer consume data\r\n");
  HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
  HAL_Delay(500);
  osSemaphoreRelease(bSemEmptyHandle);// 释放bSemEmptyHandle 信息
  osDelay(200);
 }
 /* USER CODE END FuncConsumer */
}
```

2. Lab12

MailProducerTask:

```
void MailProducerTask(void const * argument) {

 /* USER CODE BEGIN MailProducerTask */
 mailStruct * mail;
 /* Infinite loop */
 int i=1;
 for (;;) {
       mail = (mailStruct *)osMailAlloc(mail01Handle, osWaitForever);
                  mail->var = i;
                  if(osMailPut(mail01Handle, mail) == osErrorOS) {    //放入一个mail
                        char msg[25];
                        sprintf(msg, "Producer produce fail  %d\r\n",i );
                        HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
                  }else {
                        char msg[25];
                        sprintf(msg, "Producer produce success  %d\r\n",i );
                        HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
                        i++;
                  }
```

```
            osDelay(1000);
    }
    /* USER CODE END MailProducerTask */
}
```

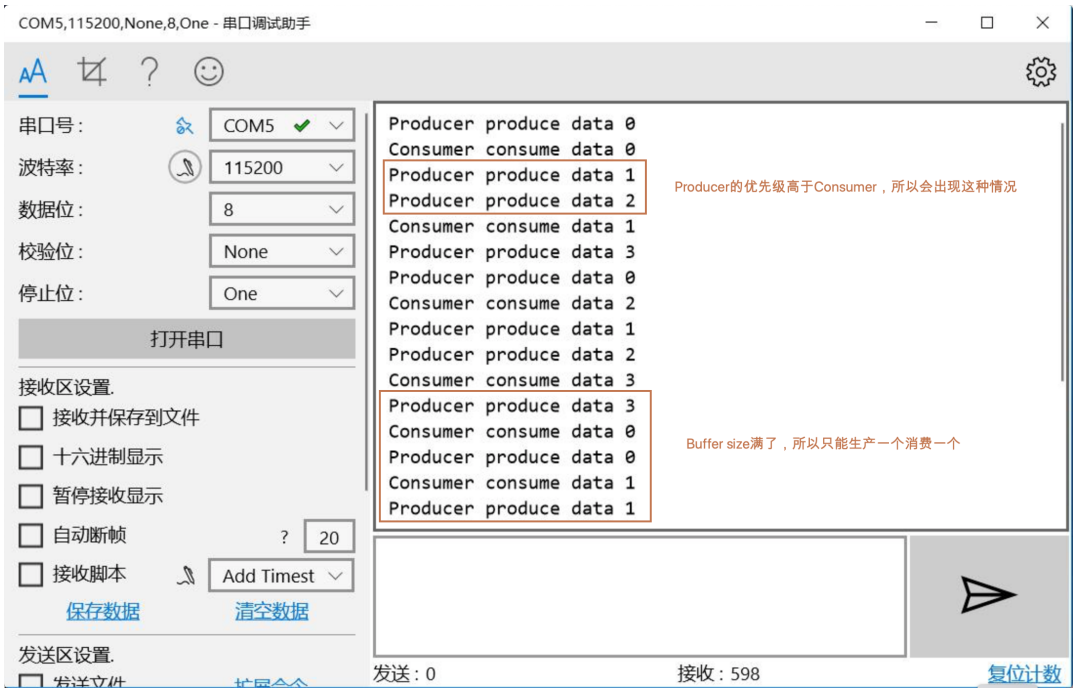
MailConsumerTask:

```
void MailConsumerTask(void const * argument) {
/* USER CODE BEGIN MailConsumerTask */

 osEvent event;
 mailStruct * pMail;
 char msg[25];
for(;;) {osDelay(3000);
        event = osMailGet(mail01Handle, osWaitForever);  //get 一个 event
        if (event.status == osEventMail)
        {
            pMail = event.value.p;  //获取value 并输出
            sprintf(msg, "Consumer consume: %d\r\n", pMail->var);
            HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
            osMailFree(mail01Handle, pMail);  //释放该Mail
        }
}

/* USER CODE END MailConsumerTask */
}
```

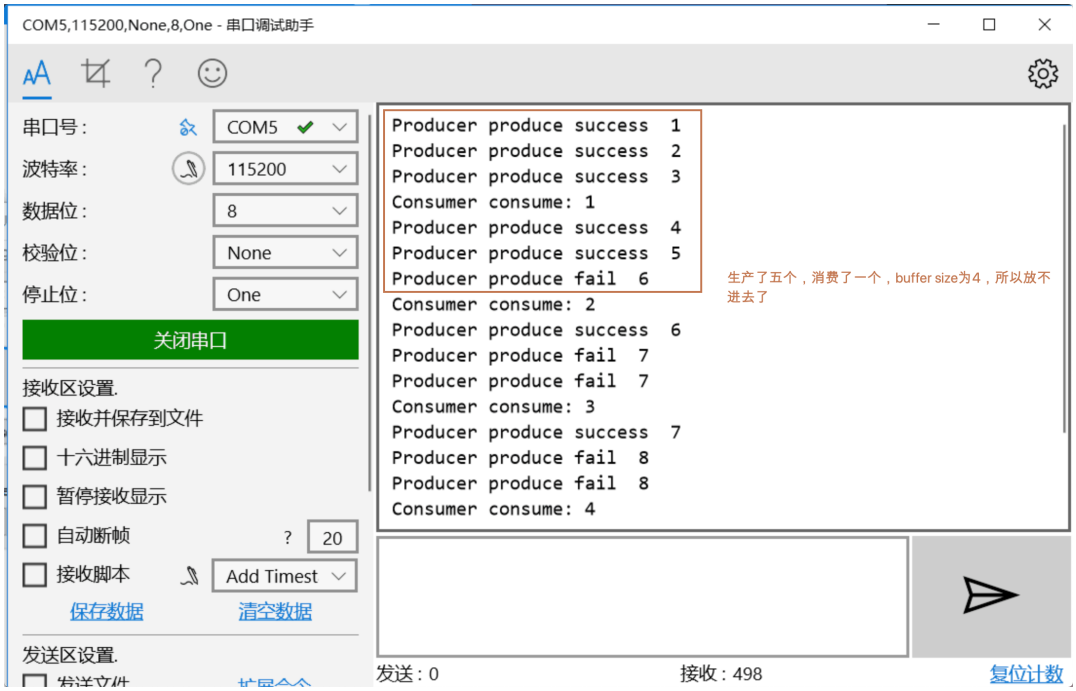

姓名：胡玉斌　　学号：11712121　　同组人:武羿　　日期：2019/12/04

实际验证附图（含开发板状态拍照、仿真截图等）：

1. Lab11



2. Lab12

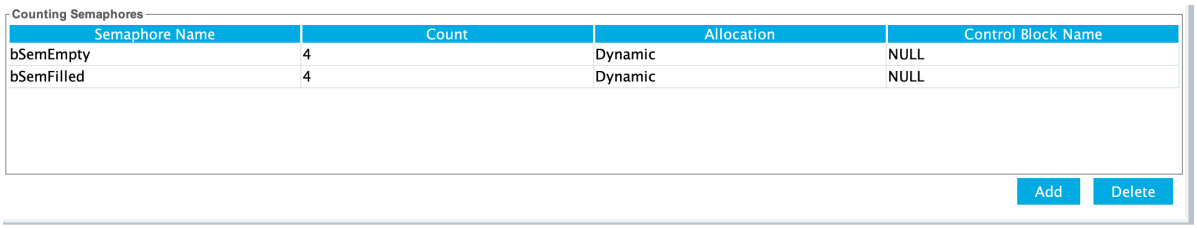


姓名：胡玉斌　　学号：11712121　　同组人:武羿　　日期：2019/12/04

五、实验总结

简述在实验过程中出现的问题，解决的过程和结果，及其他需要说明的情况。

1. 如何设置buffer size为4



2.了解了用时分复用的方式在板子上实现多线程的操作。

3. FreeRTOS为操作队列提供了非常丰富的API函数，包括队列的创建、删除，灵活的入队和出队方式、带中断保护的入队和出队等等。

4. 如何检测队列满了？继续往里面加东西，直到报错为止。