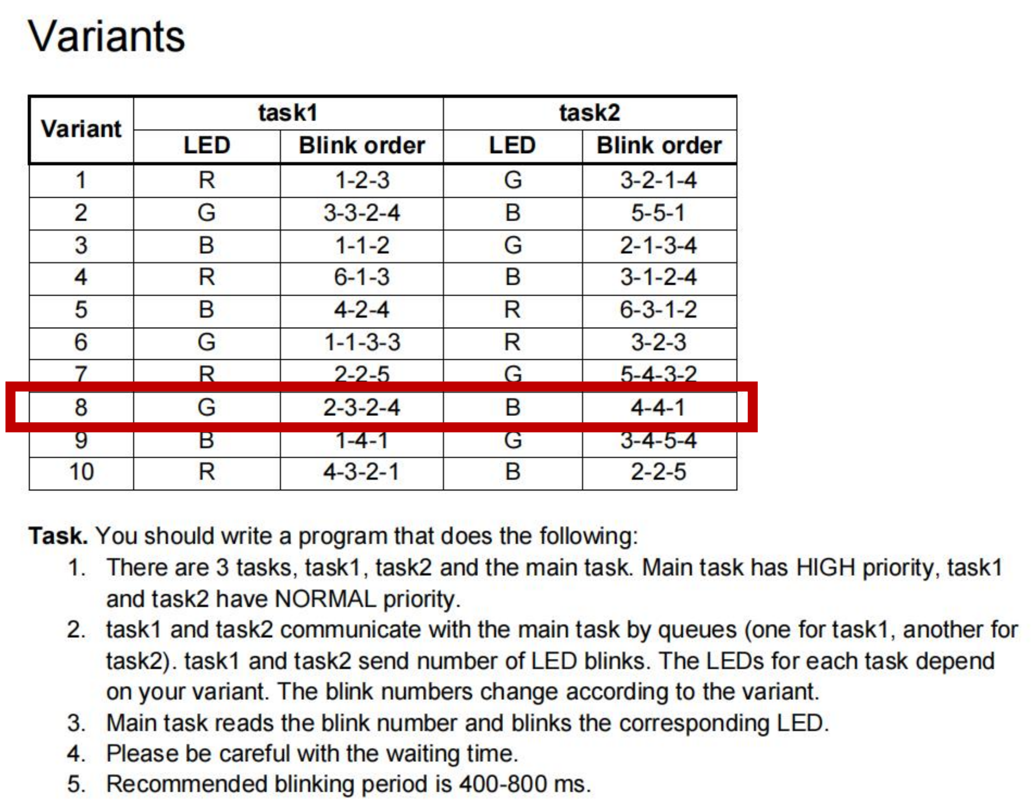
Embedded Systems Design (2022)

Report on LAB 3

- Student: JUNYAN, YANG（杨钧彦）
- Student Number: 212320028
- Date: 30/09/2022
- Yandex Link: https://disk.yandex.ru/i/ZE3CIOCLZRCF6Q

1. Task Statement



Variants

Variant	task1		task2	
	LED	Blink order	LED	Blink order
1	R	1-2-3	G	3-2-1-4
2	G	3-3-2-4	B	5-5-1
3	B	1-1-2	G	2-1-3-4
4	R	6-1-3	B	3-1-2-4
5	B	4-2-4	R	6-3-1-2
6	G	1-1-3-3	R	3-2-3
7	R	2-2-5	G	5-4-3-2
8	G	2-3-2-4	B	4-4-1
9	B	1-4-1	G	3-4-5-4
10	R	4-3-2-1	B	2-2-5

Task. You should write a program that does the following:

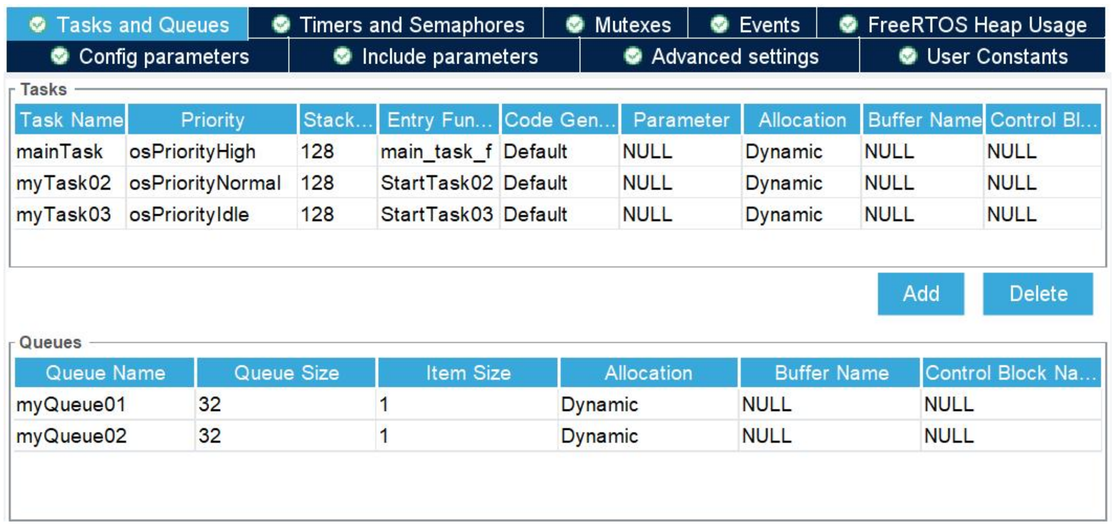
1. There are 3 tasks, task1, task2 and the main task. Main task has HIGH priority, task1 and task2 have NORMAL priority.
2. task1 and task2 communicate with the main task by queues (one for task1, another for task2). task1 and task2 send number of LED blinks. The LEDs for each task depend on your variant. The blink numbers change according to the variant.
3. Main task reads the blink number and blinks the corresponding LED.
4. Please be careful with the waiting time.
5. Recommended blinking period is 400-800 ms.

2. Environment：

Win10, STM32CubeIDE

3. Screenshot for lab3：

FreeRTOS



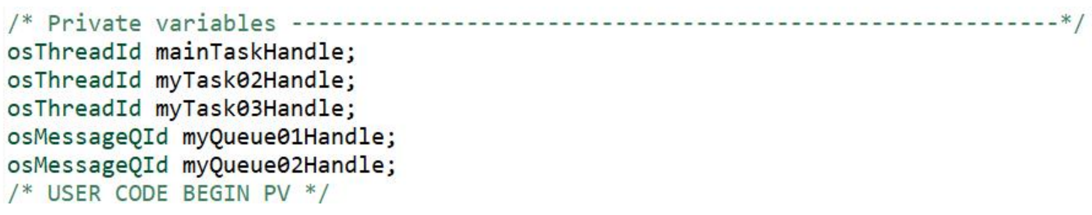
Tasks

Task Name	Priority	Stack...	Entry Fun...	Code Gen...	Parameter	Allocation	Buffer Name	Control Bl...
mainTask	osPriorityHigh	128	main_task_f	Default	NULL	Dynamic	NULL	NULL
myTask02	osPriorityNormal	128	StartTask02	Default	NULL	Dynamic	NULL	NULL
myTask03	osPriorityIdle	128	StartTask03	Default	NULL	Dynamic	NULL	NULL

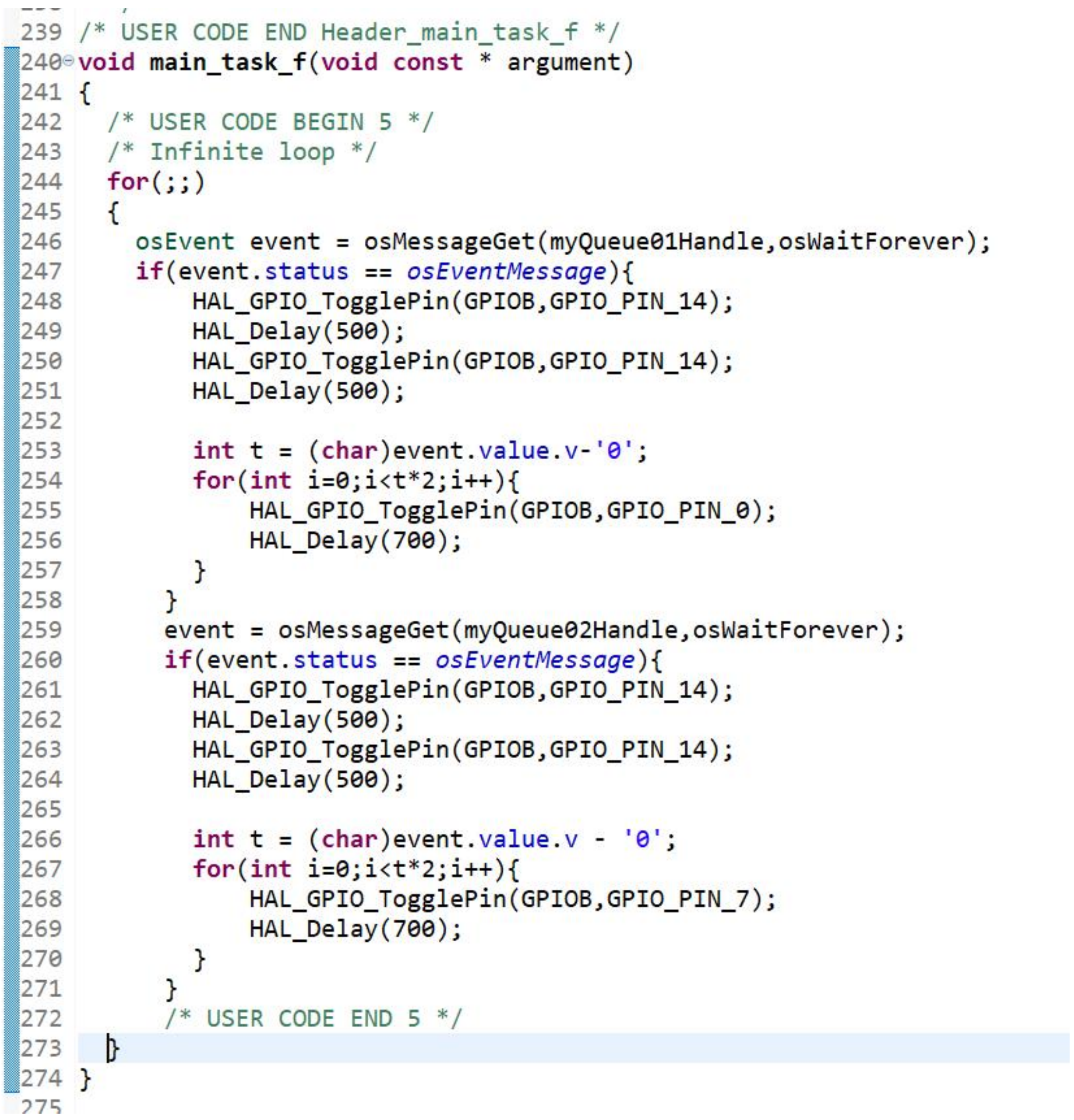
Queues

Queue Name	Queue Size	Item Size	Allocation	Buffer Name	Control Block Na...
myQueue01	32	1	Dynamic	NULL	NULL
myQueue02	32	1	Dynamic	NULL	NULL

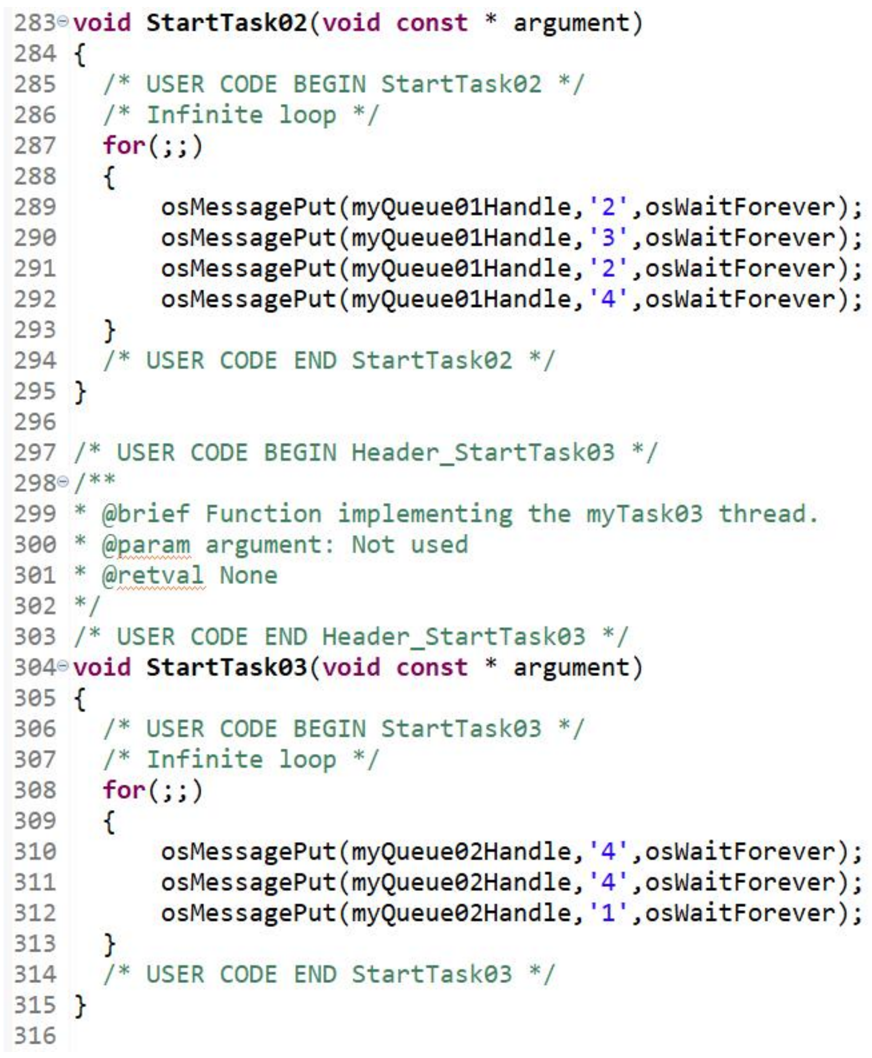
main.c



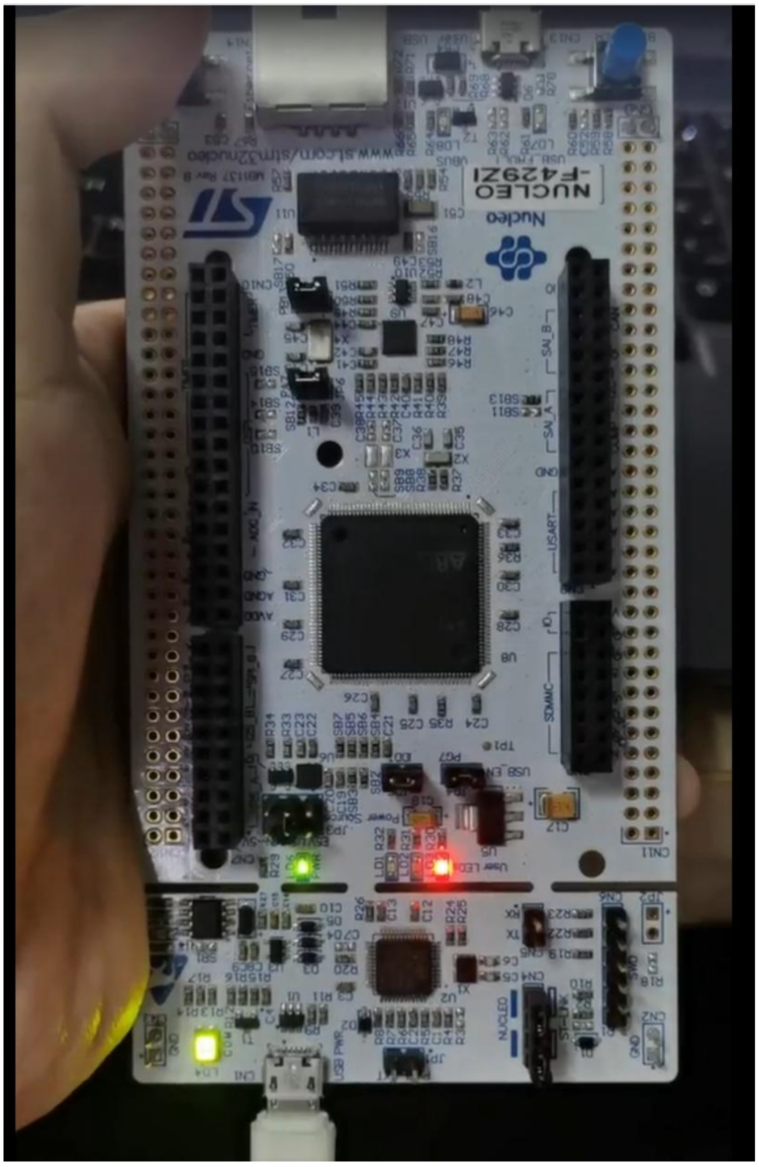
```c
/* Private variables ---------------------------------------------------------*/
osThreadId mainTaskHandle;
osThreadId myTask02Handle;
osThreadId myTask03Handle;
osMessageQId myQueue01Handle;
osMessageQId myQueue02Handle;
/* USER CODE BEGIN PV */
```



```c
/* USER CODE END Header_main_task_f */
void main_task_f(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
    osEvent event = osMessageGet(myQueue01Handle,osWaitForever);
    if(event.status == osEventMessage){
        HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_14);
        HAL_Delay(500);
        HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_14);
        HAL_Delay(500);

        int t = (char)event.value.v-'0';
        for(int i=0;i<t*2;i++){
            HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
            HAL_Delay(700);
        }
      }
      event = osMessageGet(myQueue02Handle,osWaitForever);
      if(event.status == osEventMessage){
        HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_14);
        HAL_Delay(500);
        HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_14);
        HAL_Delay(500);

        int t = (char)event.value.v - '0';
        for(int i=0;i<t*2;i++){
            HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_7);
            HAL_Delay(700);
        }
      }
      /* USER CODE END 5 */
  }
}
```



```c
void StartTask02(void const * argument)
{
  /* USER CODE BEGIN StartTask02 */
  /* Infinite loop */
  for(;;)
  {
      osMessagePut(myQueue01Handle,'2',osWaitForever);
      osMessagePut(myQueue01Handle,'3',osWaitForever);
      osMessagePut(myQueue01Handle,'2',osWaitForever);
      osMessagePut(myQueue01Handle,'4',osWaitForever);
  }
  /* USER CODE END StartTask02 */
}

/* USER CODE BEGIN Header_StartTask03 */
/**
* @brief Function implementing the myTask03 thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_StartTask03 */
void StartTask03(void const * argument)
{
  /* USER CODE BEGIN StartTask03 */
  /* Infinite loop */
  for(;;)
  {
      osMessagePut(myQueue02Handle,'4',osWaitForever);
      osMessagePut(myQueue02Handle,'4',osWaitForever);
      osMessagePut(myQueue02Handle,'1',osWaitForever);
  }
  /* USER CODE END StartTask03 */
}
```


NUCLEO
-F429ZI
Nucleo
www.st.com/stm32nucleo
User LED