BÁO CÁO THỰC HÀNH BÀI 2

Môn học: **CHUYÊN ĐỀ THIẾT KẾ HỆ THỐNG NHÚNG 1**- Mã lớp: **CE437.N11**
Giảng viên hướng dẫn thực hành: Phạm Minh Quân

	Mã số sinh viên	**Họ và tên sinh viên**
Thông tin các sinh viên	19520887	Phạm Trung Quốc
	19521651	Phạm Trọng Huỳnh
	19520928	Viên Minh Tân
	19520036	Phạm Quốc Đăng
Link các tài liệu tham khảo *(nếu có)*		
Đánh giá của giảng viên: *+ Nhận xét* *+ Các lỗi trong chương trình* *+ Gợi ý*		

[Báo cáo chi tiết các thao tác, quy trình sinh viên đã thực hiện trong quá trình làm bài thực hành. Chụp lại hình ảnh màn hình hoặc hình ảnh kết quả chạy trên sản phẩm. Mô tả và giải thích chương trình tương ứng để cho ra kết quả như hình ảnh đã trình bày. Sinh viên xuất ra file .pdf và đặt tên theo cấu trúc: MSSV_HoTen_Labx_Report.pdf (Trong đó: MSSV là mã số sinh viên, HoTen là họ và tên, x trong Labx là chỉ số của bài thực hành tương ứng]

Mục lục

Câu 1. **Viết chương trình để thực hiện các tác vụ liên quan đến lập trình UART và RTOS trên kit OPEN405R-C PACKAGE A, STM32F4 development board:**

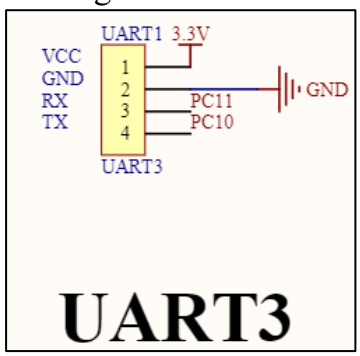
- Tác vụ 1: Đọc các dữ liệu nhận được từ UART
- Tác vụ 2: Xuất ra dữ liệu đã nhận được từ UART, xuất ra theo chu kỳ 500ms bản tin: "Chương trình đang chạy…"
- Tác vụ 3: Xử lý dữ liệu nhận được từ UART và điều khiển các LED theo dữ liệu đã được nhận từ UART

1.1. Thiết lập các cấu hình liên quan

1.1.1. Thiết lập cấu hình sử dụng USART3

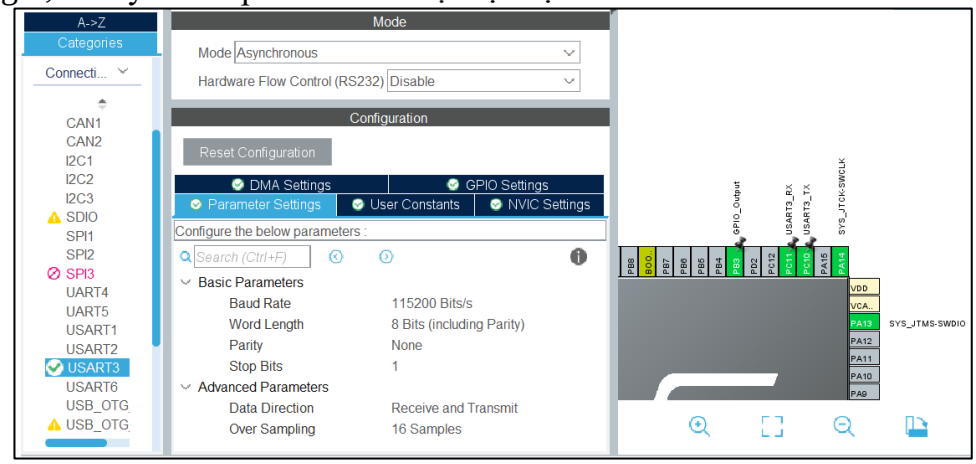
Các cấu hình ban đầu và xung clock làm như bình thường.

Thực hiện cấu hình cho USART3 tại thẻ Connectivity và dựa theo sơ đồ nguyên lý mà nhà sản xuất cung cấp:

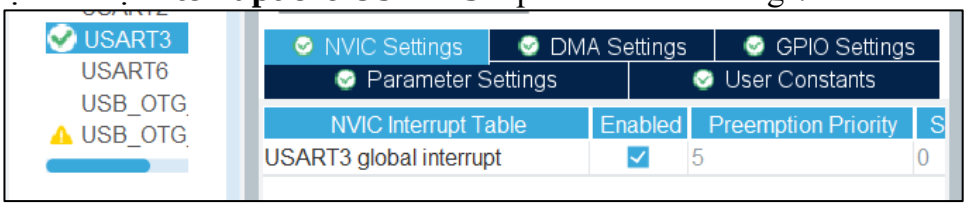


- PC11 - RX
- PC10 - TX

Chọn mode **Asynchronous** và cài đặt các thông số Baud Rate, Word Length, Parity và Stop Bits ở chế độ mặc định.

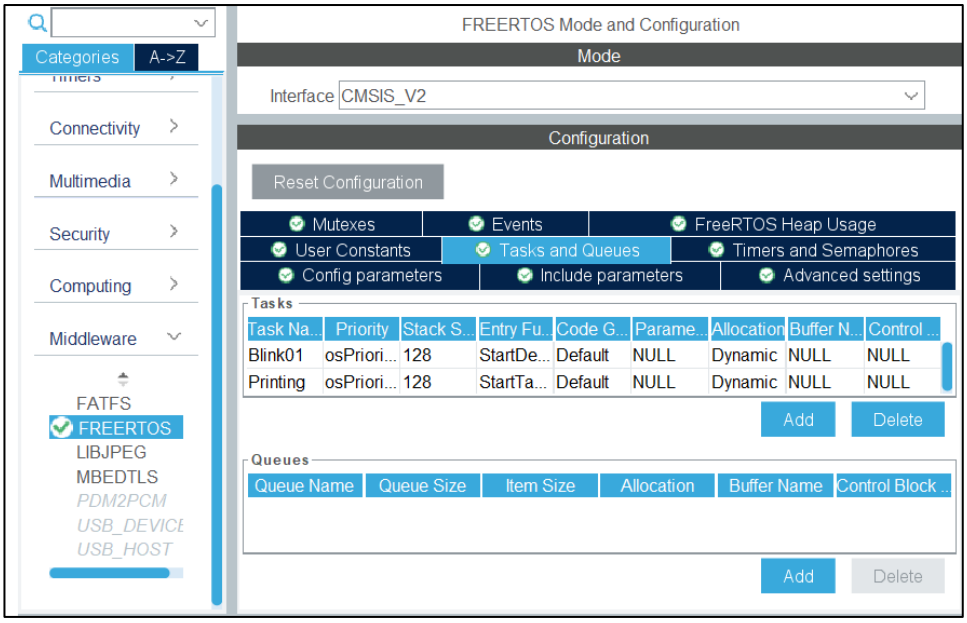


Bật chế độ **Interrupt** cho USART3 ở phần NVIC Settings.



1.2. Thiết lập cấu hình sử dụng FreeRTOS

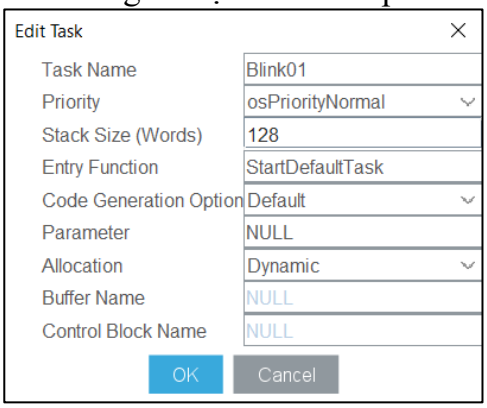
Để cấu hình sử dụng hệ điều hành thời gian thực, chọn thẻ Middleware và chọn FREE RTOS. Chọn Interface CMSIS_V2 là phiên bản mới nhất của FreeRTOS. Sau đó chọn thẻ Tasks and Queues để quản lý các tác vụ.

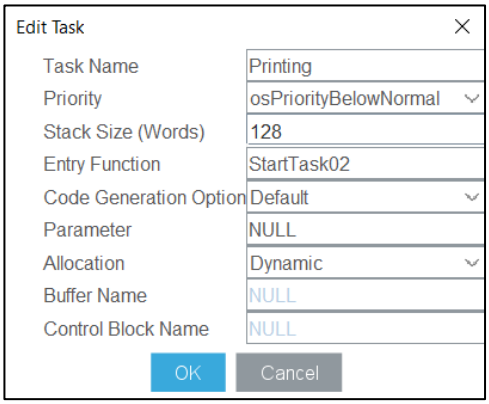


Tạo hai task (Add) chạy song song có độ ưu tiên khác nhau có tên là:

- Blink01: Dùng để xử lý nhận dữ liệu và bật tắt các Led
- Printing: Dùng để in ra dòng "Program running…"

Task Printing có độ ưu tiên thấp hơn Blink1





1.3. Sơ đồ khối

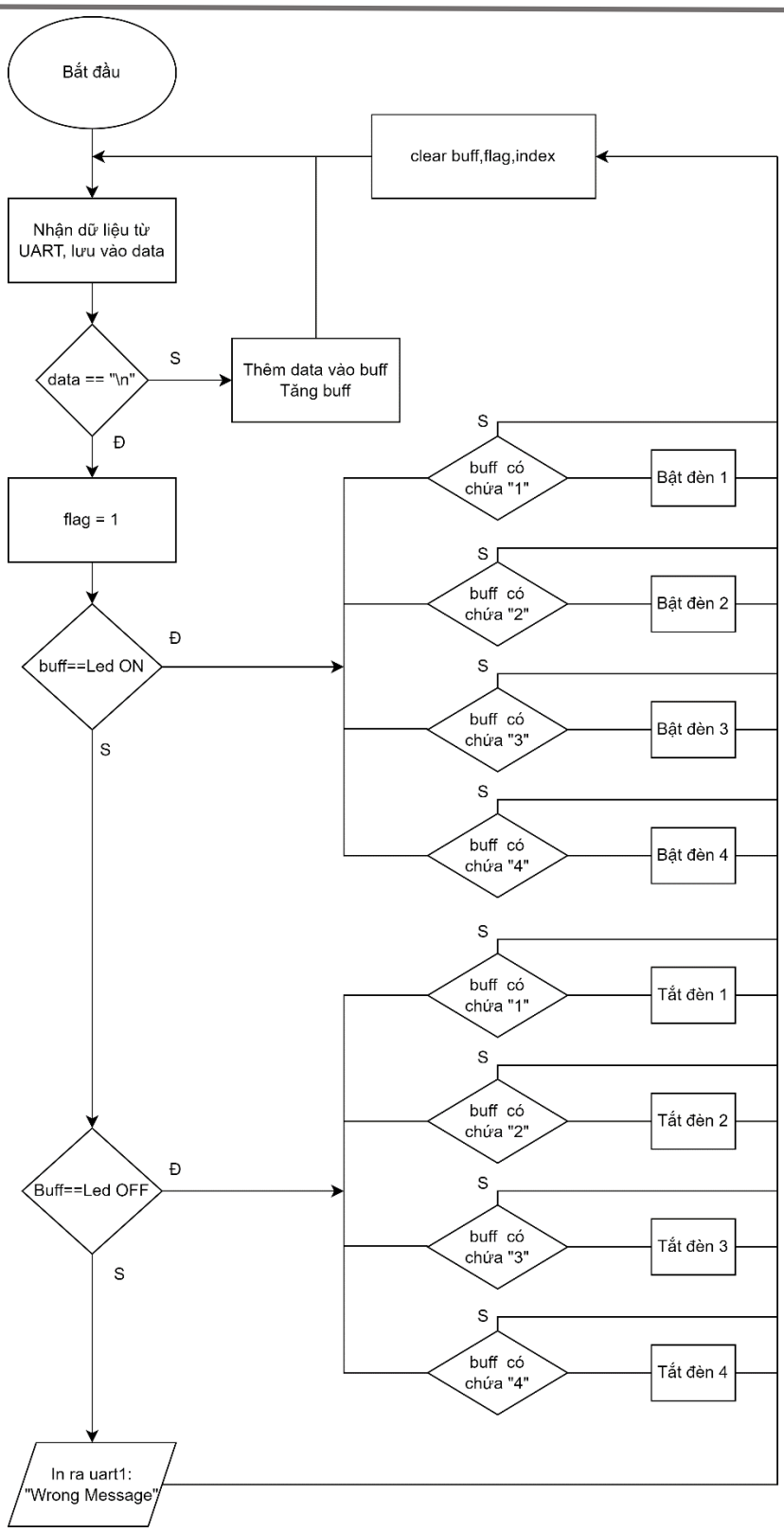
Quy ước bật tắt Led theo cú pháp sau:

Led + *chế độ + *số thứ tự của Led + <LF>

- *chế độ gồm: ON / OFF
- *số thứ tự của Led gồm: 1/2/3/4
- <LF>: kí tự xuống dòng (\n)

Ví dụ: Muốn bật đèn 2 và 4: Led ON 24 <LF>

Sơ đồ khối mô tả cách hoạt động của chương trình và giải thích cơ bản



Khi UART3 nhận được chuỗi ký tự, các ký tự sẽ được lưu theo từng byte vào một buffer tên buff, nếu ký tự kiểm tra được là "\n", bật cờ flag lên 1, kiểm tra chuỗi lưu trong buff, nếu có chứa "Led ON" hoặc "Led OFF", tiếp tục xử lý, xác định ký tự định danh của đèn (gồm 1,2,3,4), chứa định danh nào thì thực hiện bật/tắt đèn đó.

1.4. Mã nguồn và giải thích

Mã nguồn dưới đây chỉ giải thích các dòng được thêm vào cho mục đích xử lý và giải quyết yêu cầu đề bài.

```
19 /* Includes ------------------------------------------------------------------*/
20 #include "main.h"
21 #include "cmsis_os.h"
22
23 /* Private includes ----------------------------------------------------------*/
24 /* USER CODE BEGIN Includes */
25 #include "string.h"
26 #include "stdio.h"
27 /* USER CODE END Includes */
```

Thêm các thư viện cần thiết bao gồm:

- string.h: Dùng để xử lý chuỗi ký tự nhận được thông qua UART từ máy tính
- stdio.h: Dùng để sử dụng hàm printf (retarget để chuyển UART thành output mặc định của hàm printf)

```
43 /* Private variables ---------------------------------------------------------*/
44 UART_HandleTypeDef huart3;
45
46 /* Definitions for Blink01 */
47 osThreadId_t Blink01Handle;
48 const osThreadAttr_t Blink01_attributes = {
49   .name = "Blink01",
50   .stack_size = 128 * 4,
51   .priority = (osPriority_t) osPriorityNormal,
52 };
53 /* Definitions for Printing */
54 osThreadId_t PrintingHandle;
55 const osThreadAttr_t Printing_attributes = {
56   .name = "Printing",
57   .stack_size = 128 * 4,
58   .priority = (osPriority_t) osPriorityBelowNormal,
59 };
```

Khởi tạo đối tượng huart3 và hai luồng osThread đã nêu trên

```
64 /* Private function prototypes -----------------------------------------------*/
65 void SystemClock_Config(void);
66 static void MX_GPIO_Init(void);
67 static void MX_USART3_UART_Init(void);
68 void StartDefaultTask(void *argument);
69 void StartTask02(void *argument);
```

Gọi các hàm khởi tạo cần thiết cho hệ thống

```c
/* USER CODE BEGIN PFP */
#define MAX_BUFFER_SIZE 20
#ifdef __GNUC__
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
/**
  * @brief  Retargets the C library printf function to the USART.
  * @param  None
  * @retval None
  */
PUTCHAR_PROTOTYPE
{
  HAL_UART_Transmit(&huart3, (uint8_t *)&ch, 1, 300);
  return ch;
}
```

Đoạn prototype trên dùng cho việc retarget để chuyển UART thành output mặc định của hàm printf.

```c
typedef struct
{
        uint8_t flag;
        uint8_t buff[MAX_BUFFER_SIZE];
        uint8_t index;
} Uart_Typedef;
static uint8_t        data_rx;
Uart_Typedef huart_data;
/* USER CODE END PFP */
```

Tạo một struct Uart_Typedef chứa các thông tin:

- flag: cờ kích hoạt để xử lý bật tắt đèn, khi gặp ký tự kết thúc “\n”
- buff: chứa dữ liệu nhận được
- index: index của ký tự nhận được hiện tại

Biến data_rx chứa từng byte kí tự nhận được

```c
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
        if(huart->Instance == huart3.Instance)
        {
            if(data_rx == '\n')
            {
                    huart_data.flag = 1;
            }
            else
            {
                huart_data.buff[huart_data.index++] = data_rx;
                if(huart_data.index > MAX_BUFFER_SIZE)
                {
                    huart_data.index = 0;
                }
            }
            HAL_UART_Receive_IT(&huart3, &data_rx, 1);
        }
}
```

Chỉnh sửa hàm Receive Complete Callback, khi một receive callback được gọi, nếu huart3 có dữ liệu, tiến hành kiểm tra biến data_rx có phải ký tự “\n”

hay không, nếu đúng thì flag bật lên 1 và xử lý chuỗi, nếu chưa phải thì tiếp tục nhận vào lưu vào buff. Đồng thời tăng index thêm 1, nếu index lớn hơn MAX_BUFFER_SIZE (20), index đặt lại bằng 0. Gọi lại hàm Receive interrupt để đọc ký tự lần tiếp.

```
150   /* USER CODE BEGIN 2 */
151   HAL_UART_Receive_IT(&huart3, &data_rx, 1);
152   /* USER CODE END 2 */
```

Gọi hàm Receive interrupt lần đầu để các lần tiếp theo sẽ được gọi lại callback.

```
319 void StartDefaultTask(void *argument)
320 {
321   /* USER CODE BEGIN 5 */
322   /* Infinite loop */
323   for(;;)
324   {
325     printf("Program running...\n");
326     osDelay(500);
327   }
328   /* USER CODE END 5 */
329 }
```

Luồng DefaultTask dùng để in ra dòng Program running mỗi 500ms

```
338 void StartTask02(void *argument)
339 {
340   /* USER CODE BEGIN StartTask02 */
341   /* Infinite loop */
342   for(;;)
343   {
344       if(huart_data.flag == 1)
345       {
346           if(strstr((char *)huart_data.buff, "Led ON"))
347           {
348                   if(strstr((char *)huart_data.buff, "1"))
349                   {
350                           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
351                           HAL_UART_Transmit(&huart3, "LED 1 ON ALREADY\n", 17, 300);
352                   }
353                   if(strstr((char *)huart_data.buff, "2"))
354                   {
355                           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_SET);
356                           HAL_UART_Transmit(&huart3, "LED 2 ON ALREADY\n", 17, 300);
357                   }
358                   if(strstr((char *)huart_data.buff, "3"))
359                   {
360                           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, GPIO_PIN_SET);
361                           HAL_UART_Transmit(&huart3, "LED 3 ON ALREADY\n", 17, 300);
362                   }
363                   if(strstr((char *)huart_data.buff, "4"))
364                   {
365                           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_SET);
366                           HAL_UART_Transmit(&huart3, "LED 4 ON ALREADY\n", 17, 300);
367                   }
368           }
```

```
369             else if(strstr((char *)huart_data.buff, "Led OFF"))
370             {
371                     if(strstr((char *)huart_data.buff, "1"))
372                     {
373                             HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
374                             HAL_UART_Transmit(&huart3, "LED 1 OFF ALREADY\n", 18, 300);
375                     }
376                     if(strstr((char *)huart_data.buff, "2"))
377                     {
378                             HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_RESET);
379                             HAL_UART_Transmit(&huart3, "LED 2 OFF ALREADY\n", 18, 300);
380                     }
381                     if(strstr((char *)huart_data.buff, "3"))
382                     {
383                             HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, GPIO_PIN_RESET);
384                             HAL_UART_Transmit(&huart3, "LED 3 OFF ALREADY\n", 18, 300);
385                     }
386                     if(strstr((char *)huart_data.buff, "4"))
387                     {
388                             HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_RESET);
389                             HAL_UART_Transmit(&huart3, "LED 4 OFF ALREADY\n", 18, 300);
390                     }
391             }
392             else
393                     printf("Wrong Message\n");
394             //HAL_UART_Transmit(&huart1, huart_data.buff, huart_data.index, 100);
395             memset(huart_data.buff, 0, MAX_BUFFER_SIZE);
396             huart_data.index = 0;
397             huart_data.flag = 0;
398
399         }
400     }
401     /* USER CODE END StartTask02 */
402 }
```

Khi flag được bật lên 1(đã nhận được ký tự "\n"), luồng task02 tiến hành xử lý chuỗi, khi phát hiện chuỗi có chứa dòng Led ON hoặc Led OFF, kiểm tra tiếp số thứ tự đèn cần được bật/tắt và bật/tắt chúng, đồng thời in ngược lại uart3 dòng LED … ALREADY. Nếu không phát hiện đúng chuỗi Led ON/Led OFF, in lại dòng Wrong Message. Sau đó, dùng memset để xóa dữ liệu trong buff, đặt lại flag và index về 0.

1.5. Video demo

Video demo được đặt ở đường dẫn sau:
https://drive.google.com/file/d/11PIPUiGv0hSYV5ZG_gPN8asJJ4IgkZvH/view?usp=share_link