# TI DSP, MCU, Xilinx Zynq FPGA
# 프로그래밍 전문가 과정

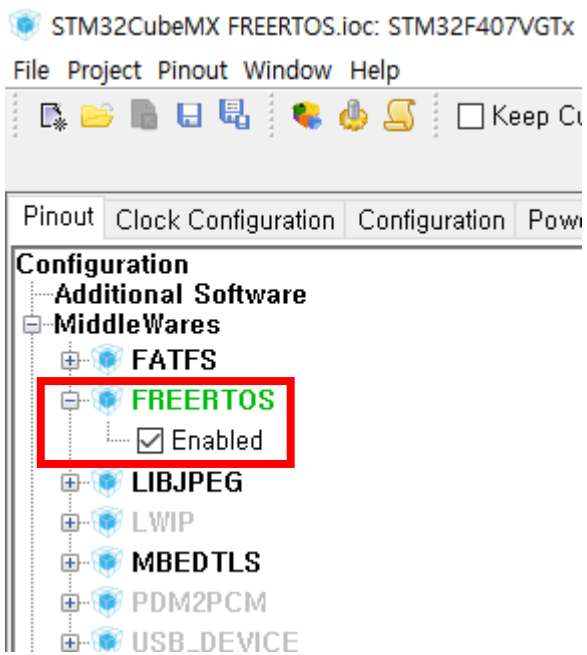
## FreeRTOS based on STM32F407

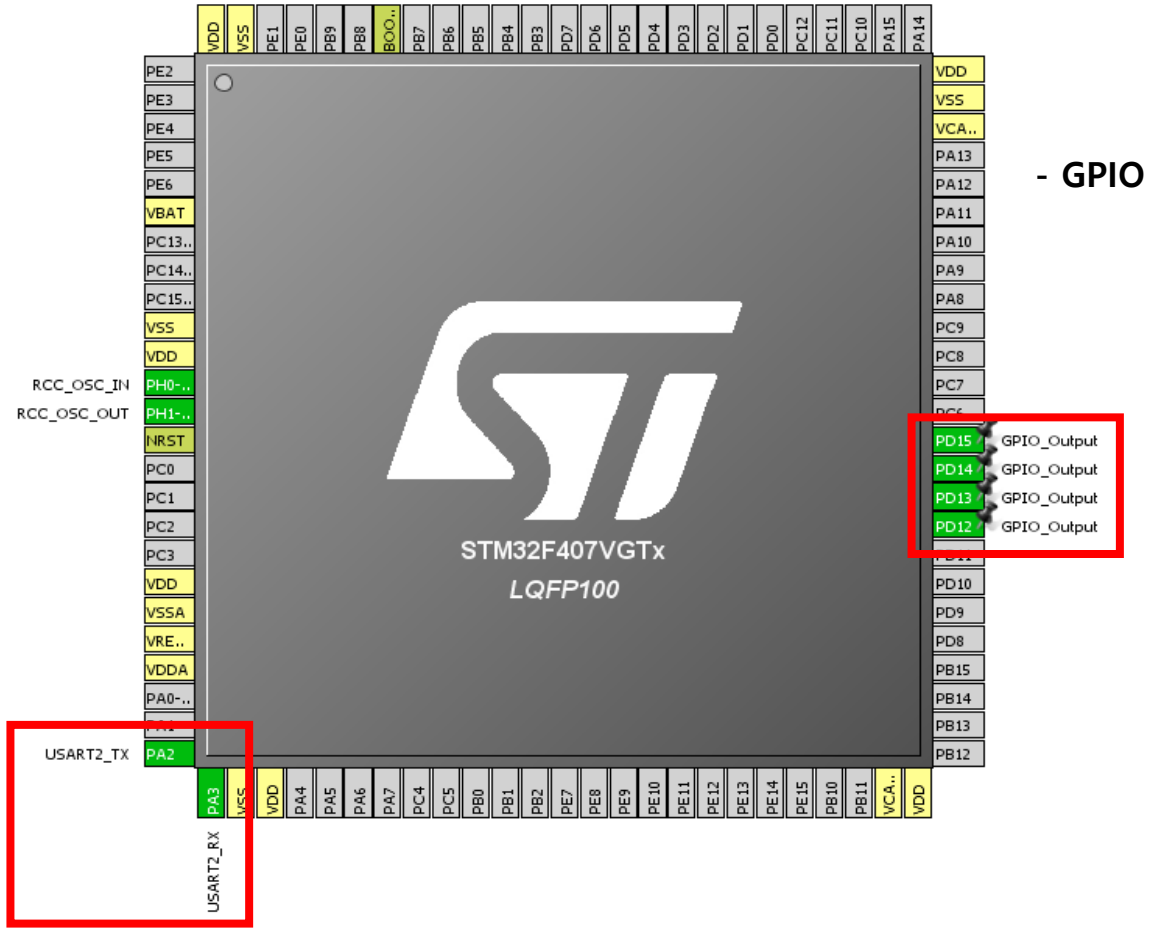
**강사 – Innova Lee(이상훈)**
gcccompil3r@gmail.com

**학생 – 안상재**
sangjae2015@naver.com

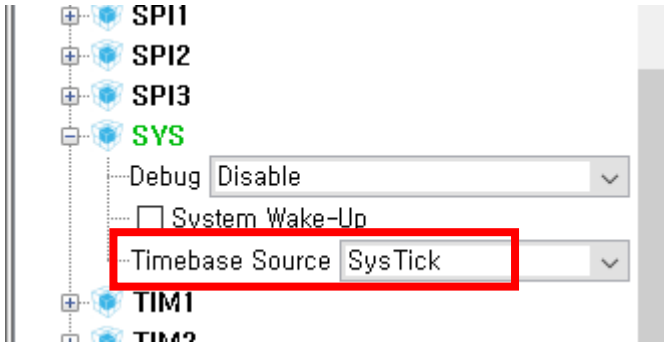
# *1. TASK 스케줄링을 통한 LED on/off*

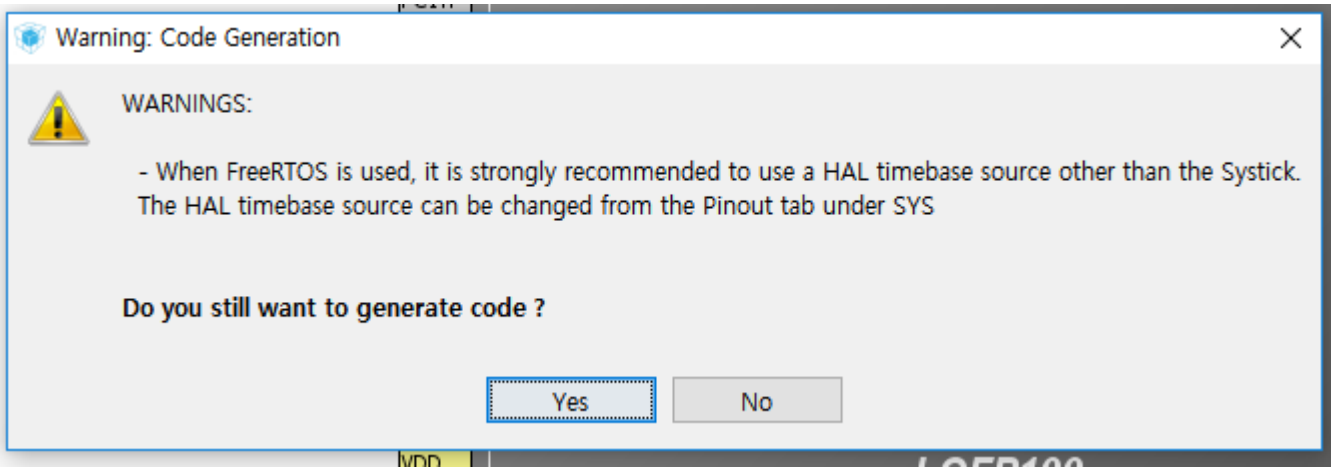
- FreeRTOS CubeMX 설정 (클럭 트리 설정 생략)



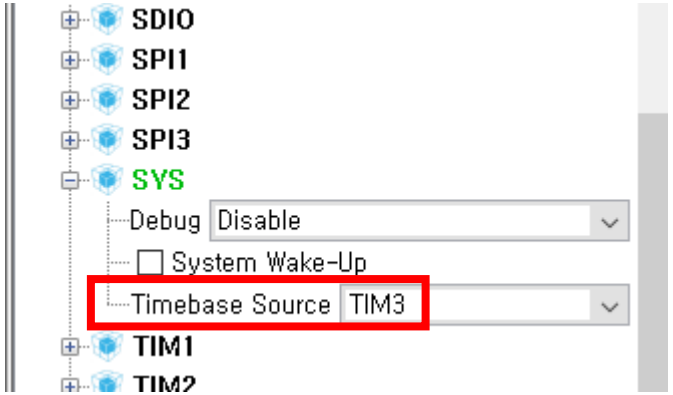




**- GPIO 와 USART2를 TEST 함!**



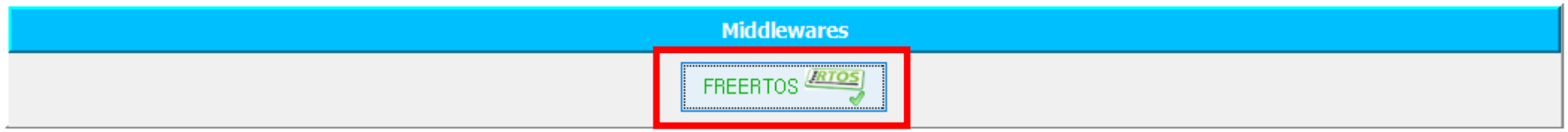




SYS의 Timebase Source를 SysTick 으로 설정하면 우측과 같은 메시지가 나옴

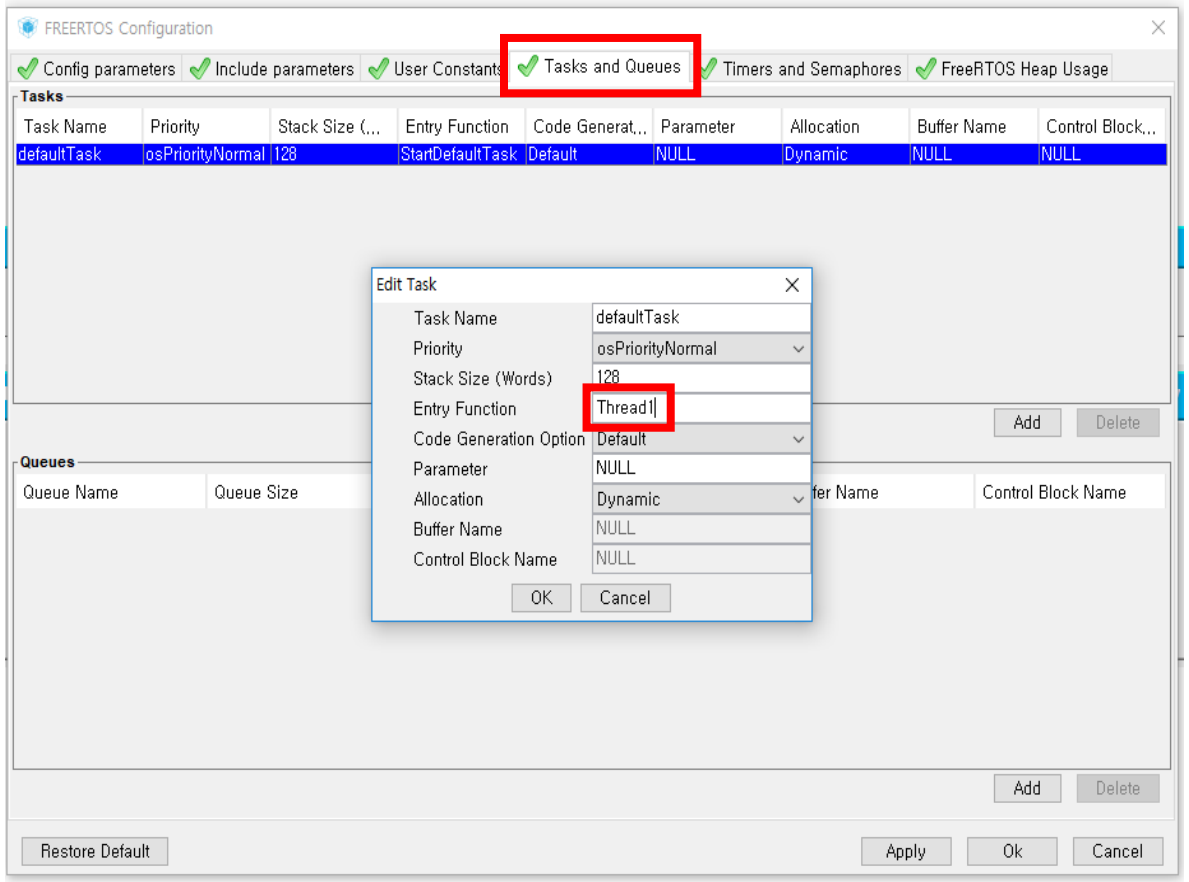


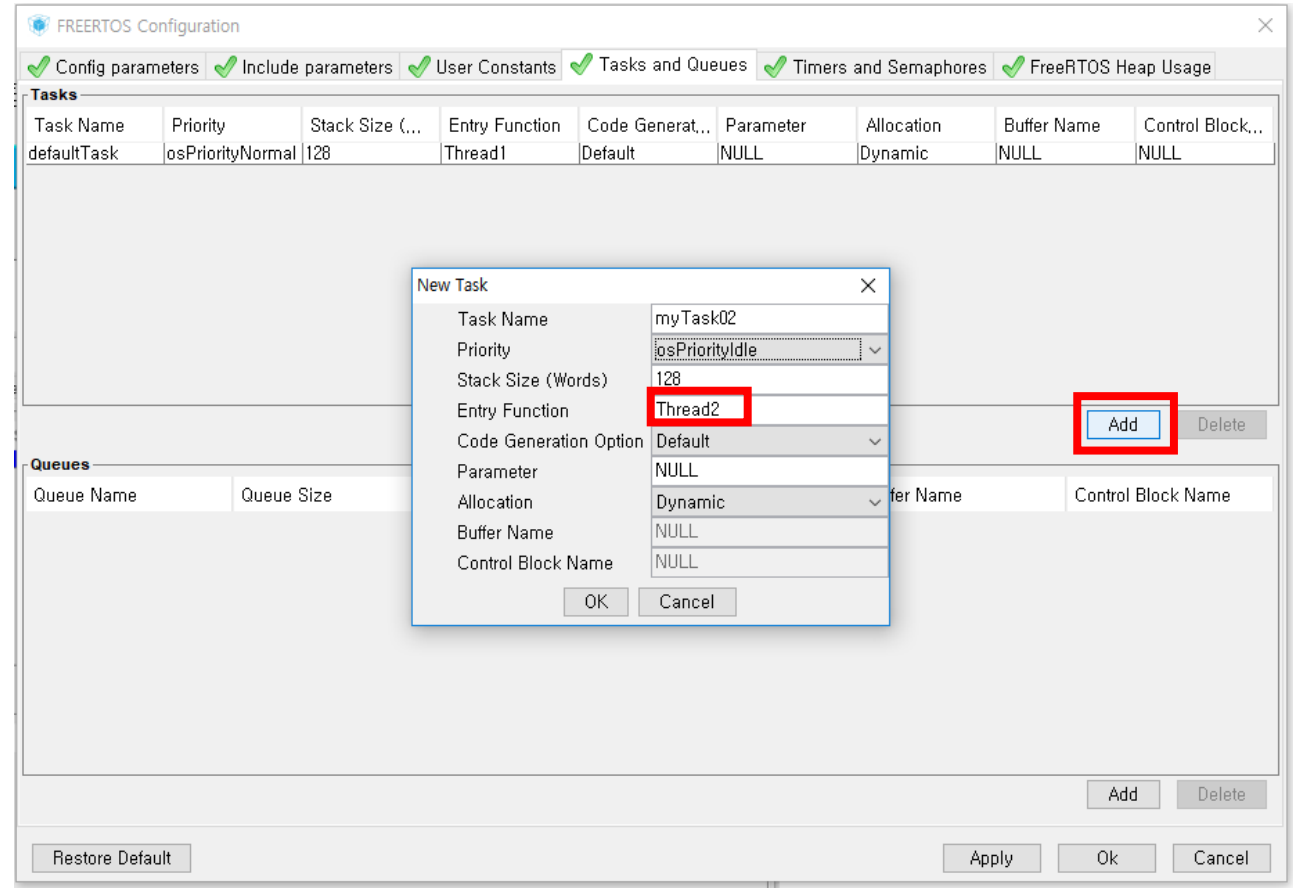

TIM3 으로 설정해줌

configuration 탭

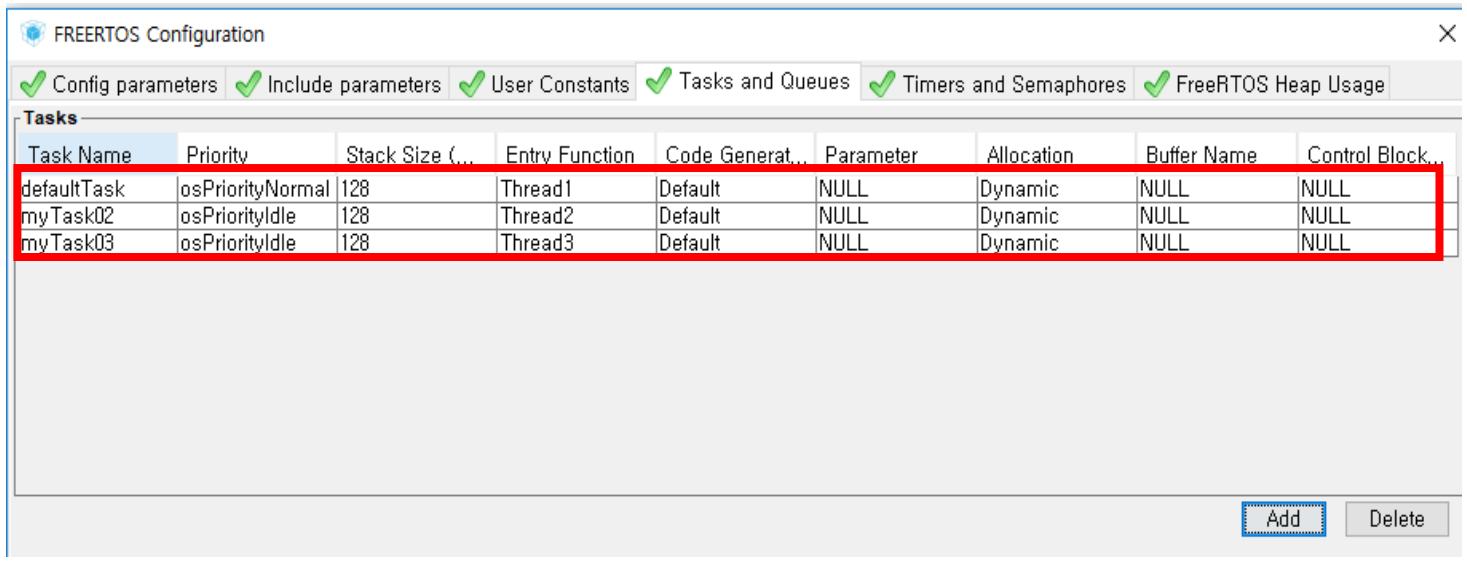



## - Thread 생성

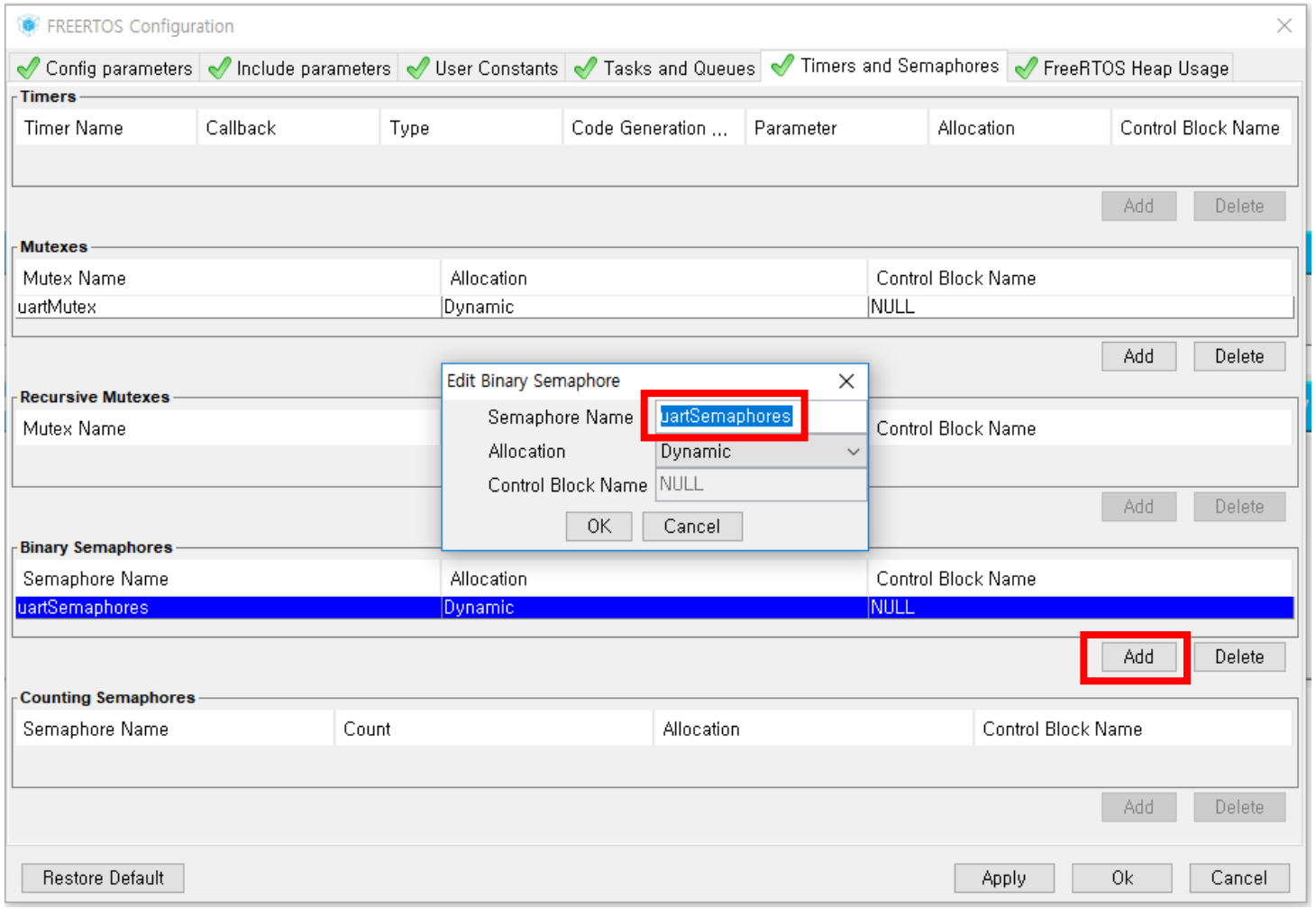


## Thread 총 3개 생성



- Semaphores 생성



- 소스 코드

main.c | freertos.c | semphr.h | portm

```c
/* USER CODE END PD */

/* Private macro -------------------------
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------
/* USER CODE BEGIN Variables */

/* USER CODE END Variables */
osThreadId defaultTaskHandle;
osThreadId myTask02Handle;
osThreadId myTask03Handle;
osSemaphoreId uartSemaphoresHandle;

/* Private function prototypes -----------
/* USER CODE BEGIN FunctionPrototypes */

/* USER CODE END FunctionPrototypes */

void Thread1(void const * argument);
void Thread2(void const * argument);
void Thread3(void const * argument);
```

Thread ID 선언

Semaphore ID 선언

Thread 함수 선언

**Semaphore 정의 및 생성**

```c
/* Create the semaphores(s) */
/* definition and creation of uartSemaphores */
osSemaphoreDef(uartSemaphores);
uartSemaphoresHandle = osSemaphoreCreate(osSemaphore(uartSemaphores), 1);
```

**Thread 정의 및 생성**

```c
/* Create the thread(s) */
/* definition and creation of defaultTask */
osThreadDef(defaultTask, Thread1, osPriorityNormal, 0, 128);
defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);

/* definition and creation of myTask02 */
osThreadDef(myTask02, Thread2, osPriorityIdle, 0, 128);
myTask02Handle = osThreadCreate(osThread(myTask02), NULL);

/* definition and creation of myTask03 */
osThreadDef(myTask03, Thread3, osPriorityIdle, 0, 128);
myTask03Handle = osThreadCreate(osThread(myTask03), NULL);
```

Thread 함수 정의

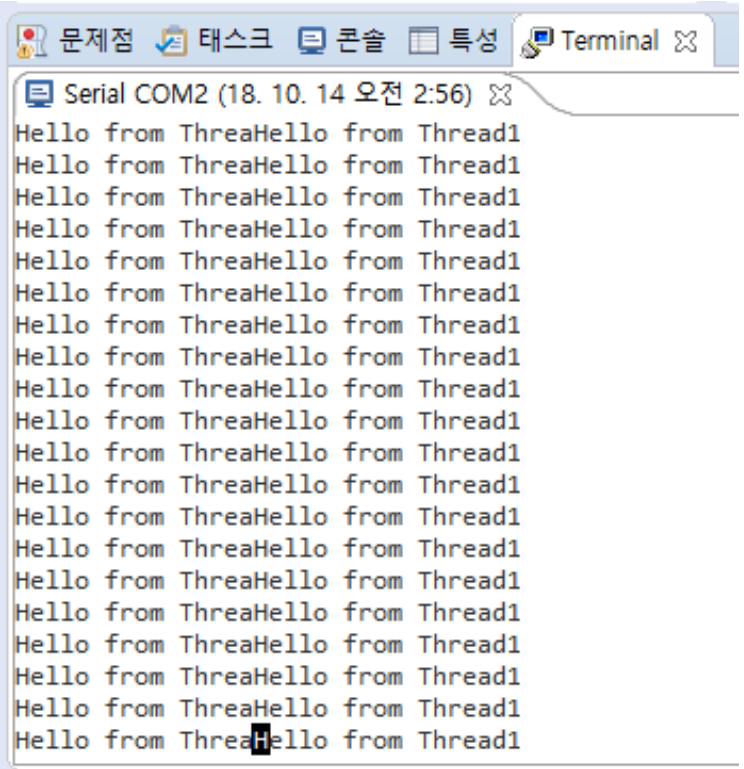
```c
void Thread1(void const * argument)
{

  /* USER CODE BEGIN Thread1 */
    uint8_t tx_data[20] = "Hello from Thread1\r\n";
  /* Infinite loop */
  for(;;)
  {
      HAL_UART_Transmit(&huart2, tx_data, 20, 1);
    osDelay(1000);
  }
  /* USER CODE END Thread1 */
}


void Thread2(void const * argument)
{
  /* USER CODE BEGIN Thread2 */
    uint8_t tx_data[20] = "Hello from Thread2\r\n";
  /* Infinite loop */
  for(;;)
  {
      HAL_UART_Transmit(&huart2, tx_data, 20, 1);
    osDelay(1000);
  }
  /* USER CODE END Thread2 */
}


void Thread3(void const * argument)
{
  /* USER CODE BEGIN Thread3 */
    uint8_t tx_data[20] = "Hello from Thread3\r\n";
  /* Infinite loop */
  for(;;)
  {
      HAL_UART_Transmit(&huart2, tx_data, 20, 1);
    osDelay(1000);
  }
  /* USER CODE END Thread3 */
}
```

- 결과 화면




=> 멀티 스레딩의 영향으로 출력되는 문자가 겹쳐지게 됨

Thread 함수 정의

```c
/* USER CODE END Header_Thread1 */
void Thread1(void const * argument)
{

  /* USER CODE BEGIN Thread1 */
    uint8_t tx_data[20] = "Hello from Thread1\r\n";
  /* Infinite loop */
  for(;;)
  {
      xSemaphoreTake(uartSemaphoresHandle, portMAX_DELAY);
      HAL_UART_Transmit(&huart2, tx_data, 20, 1);
      xSemaphoreGive(uartSemaphoresHandle);
    osDelay(1000);
  }
  /* USER CODE END Thread1 */
}

void Thread2(void const * argument)
{
  /* USER CODE BEGIN Thread2 */
    uint8_t tx_data[20] = "Hello from Thread2\r\n";
  /* Infinite loop */
  for(;;)
  {
      xSemaphoreTake(uartSemaphoresHandle, portMAX_DELAY);
      HAL_UART_Transmit(&huart2, tx_data, 20, 1);
      xSemaphoreGive(uartSemaphoresHandle);
    osDelay(1000);
  }
  /* USER CODE END Thread2 */
}

void Thread3(void const * argument)
{
  /* USER CODE BEGIN Thread3 */
    uint8_t tx_data[20] = "Hello from Thread3\r\n";
  /* Infinite loop */
  for(;;)
  {
      xSemaphoreTake(uartSemaphoresHandle, portMAX_DELAY);
      HAL_UART_Transmit(&huart2, tx_data, 20, 1);
      xSemaphoreGive(uartSemaphoresHandle);
    osDelay(1000);
  }
  /* USER CODE END Thread3 */
}
```

- 결과 화면

문제점 태스크 콘솔 특성 Terminal

Serial COM2 (18. 10. 14 오전 3:50)

```
Hello from Thread1
Hello from Thread2
Hello from Thread3
Hello from Thread1
Hello from Thread2
Hello from Thread3
Hello from Thread1
Hello from Thread2
Hello from Thread3
Hello from Thread1
Hello from Thread2
Hello from Thread3
Hello from Thread1
Hello from Thread2
Hello from Thread3
Hello from Thread1
Hello from Thread2
Hello from Thread3
Hello from Thread1
Hello from Thread2
Hello from Thread3
Hello from Thread1
```

=> 세마포어로 인해 문자를 완전히 출력할 때까지는 멀티 스레딩이 일어나지 않게함