MIC5 Week 2 – lab exercise

Introduction

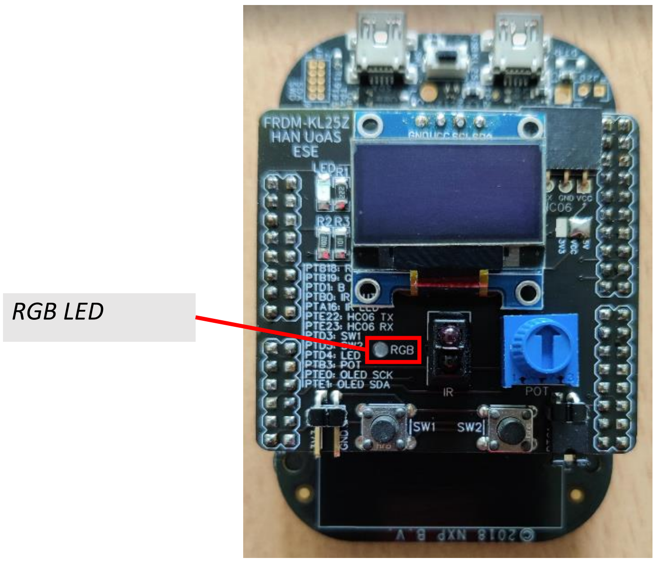
This lab exercise has two goals. The first goal is to create a FreeRTOS project from scratch in the MCUXpresso IDE. The second goal is to try to create and run as many tasks simultaneously on the FRDM-KL25Z board by configuring FreeRTOS in the FreeRTOSConfig.h file.

Hardware

The hardware required for this project is outlined in the following table.

Description	MKL25Z128VLK4 pins	Notes
RGB LED	PTB18 (red) PTB19 (green) PTD1 (blue)	-

This hardware is available on the FRDM-KL25Z board and the oled shield.



Software

There is no template available. You will create the MCUXpresso IDE project from scratch.

Create a FreeRTOS project in MCUXpresso IDE

Refer to the paragraph 1.4 in the book *Mastering the freertos real time kernel*[1]. This paragraph describes how to create a project by adapting a demo project or by creating a project from scratch. We will roughly follow the seven steps as described in **Creating a New Project From Scratch**. The following description is **NOT** a step-by-step guide, but provides hints and tips for creating a FreeRTOS project.

1. Start MCUXpresso IDE
2. Start a new project: File > New > New C/C++ Project
3. Copy whatever files from one of the example projects, such as the rgb.c and rgb.h. Leave out the FreeRTOS files.
4. Copy and paste project settings from one of the example projects, such as:
    - Project > Properties > C/C++ Build > Settings > Tool Settings > MCU C Compiler > Preprocessor
    - Project > Properties > C/C++ Build > Settings > Tool Settings > MCU C Compiler > Includes
5. Implement a simple blinky application in the main.c file and make sure that it can be uploaded to the FRDM-KL25Z board.
6. Add the FreeRTOS source files to the project. The latest version can be downloaded here. Refer to one of the example projects for the actual required files and their location.
7. Copy the FreeRTOSConfig.h file from one of the example projects. Refer to one of the example projects for the file location.
8. Make sure that the following directories are added to the path the project will search to locate header files (Project > Properties > C/C++ Build > Settings > Tool Settings > MCU C Compiler > Includes):
    - FreeRTOS/Source/include
    - FreeRTOS/Source/portable/[compiler]/[architecture]
    - The directory containing the FreeRTOSConfig.h header file
9. Build the project, making sure there are no errors and no warnings.
10. In main.c, implement one task that blinks the green LED for 100 ms on and 400 ms off.

[1] Barry, R. (2016). *Mastering the freertos real time kernel. Pre-release 161204 Edition*. Real Time Engineers Ltd.

Creating as many tasks as possible

The goal of this exercise is to create as many of the following tasks as possible:

```c
void vTask2( void *pvParameters )
{
    char str[24];
    sprintf(str, "[Task 2] Created %04d\r\n", *(int *)pvParameters);
    vSerialPutString(str);

    TickType_t xLastWakeTime = xTaskGetTickCount();

    for( ;; )
    {
        // Wait before sampling the next time
        vTaskDelayUntil( &xLastWakeTime, pdMS_TO_TICKS( 300 ) );
    }
}
```

1. Copy-and-paste this task to main.c
2. In vTask1, try to create as many of these tasks as follows:

```c
void vTask1( void *pvParameters )
{
    const TickType_t xDelay100ms = pdMS_TO_TICKS( 100 );
    const TickType_t xDelay400ms = pdMS_TO_TICKS( 400 );

    BaseType_t i=1;

    /* As per most tasks, this task is implemented in an infinite loop. */
    for( ;; )
    {
        rgb_green_on(true);
        vTaskDelay( xDelay100ms );
        rgb_green_on(false);
        vTaskDelay( xDelay400ms );

        BaseType_t res = xTaskCreate( vTask2, "Task 2",
            configMINIMAL_STACK_SIZE, &i, 2, NULL );

        if(res == pdPASS)
        {
           i++;
        }
    }
}
```

Questions

1. How many times is Task 2 created successfully? How did you find this?
2. Make changes to the FreeRTOS configuration in FreeRTOSConfig.h and/or in the project settings (Project > Properties > C/C++ Build > Settings > Tool Settings > MCU Linker > Managed Linker Script > Heap Size and Stack Size). What changes have you made and what is the maximum number of Task 2 tasks you are able to create?

3. Task 2 tasks are created by calling the xTaskCreate() function. The priority of Task 2 is set to 2. Change this priority to 1 and observe the output in a serial window before and after changing the priority. Explain the difference.

Answers