MIC5 Week 1 – lab exercise

Introduction

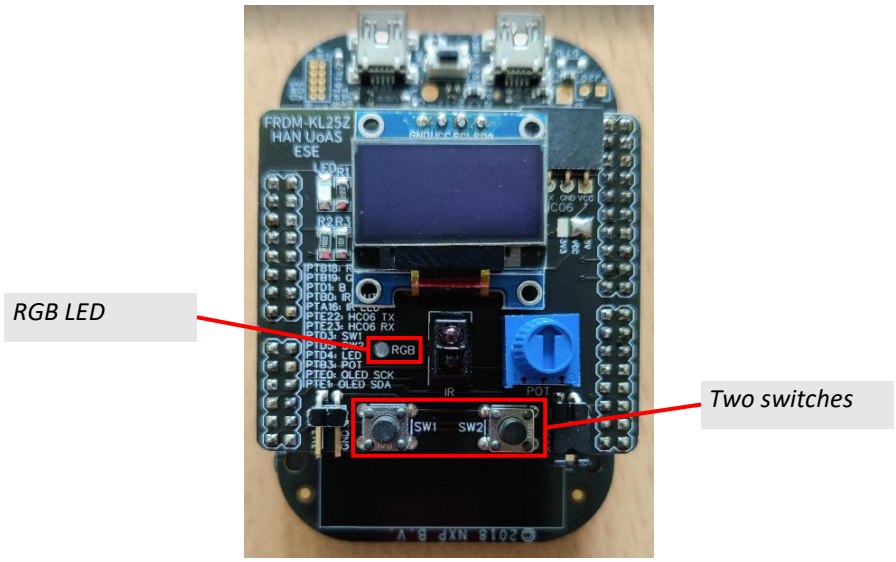
This lab exercise has two goals. The first goal is to verify and setup the installation of the MCUXpresso Integrated Development Environment (IDE). The second goal is to add several additional tasks to a template FreeRTOS project for the FRDM-KL25Z board.

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)	-
Two switches	PTD3 (SW1) PTD5 (SW2)	Remember to enable the internal pullup resistors.

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



Software

A template project is provided:

Week1 – Lab.zip

The next section uses this project to verify the installation of the MCUXpresso IDE.

Install MCUXpresso IDE

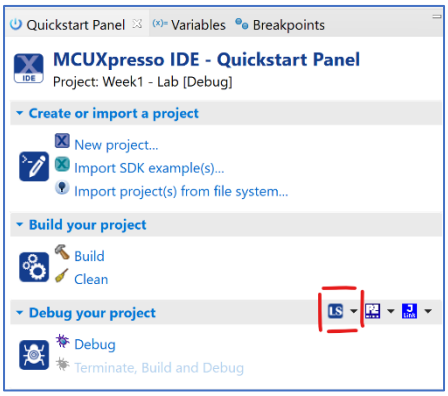
Download and install MCUXpresso IDE from the NXP website:

> https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE

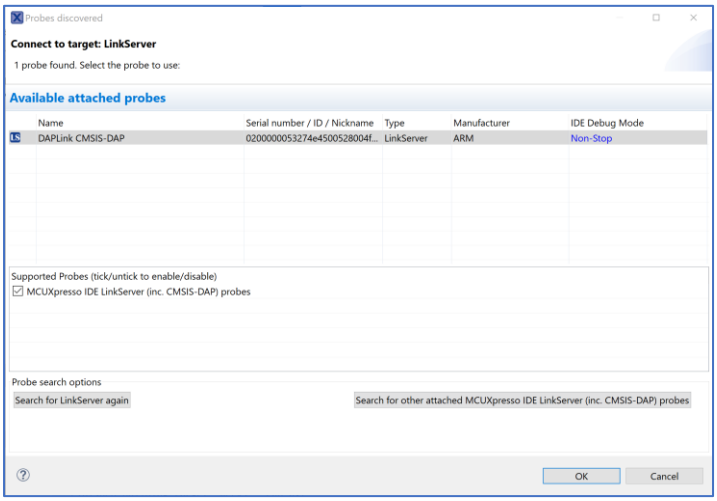
On this website you will find the installer and documentation, such as the User Guide and Installation Guide.

After installation, open the *Week1 – lab* project. Make sure a FRDM-KL25Z board is connected.

- Build the project (CTRL+B).
  The output shows *0 errors* and *3 warnings*. Ignore these warnings for now, because the functions vTask1, vTask2 and vTask3 are not yet used in the main application. This will be your assignment.
- Select the Quickstart panel.

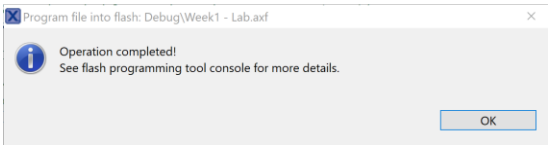


- Select the drop down arrow from the Link Server option and click **Program flash action using LinkServer**. A *probes discovered* window should popup, click OK.



TIP: the next time you need to program the microcontroller, you can simply click the LS button!

- Programming the microcontroller starts and the progress is shown in the Console. Click OK as soon as the operation completed message shows up.



That's it, the IDE is successfully setup and you are ready to start the first lab assignment.

Adding tasks

The goal of this exercise is to add three tasks:

1. Task 1 periodically blinks a green LED.
2. Task 2 detects pressing and releasing of both switches.
3. Task 3 is created at the falling edge of SW1. It goes into blocking state for 5 seconds and will delete itself.

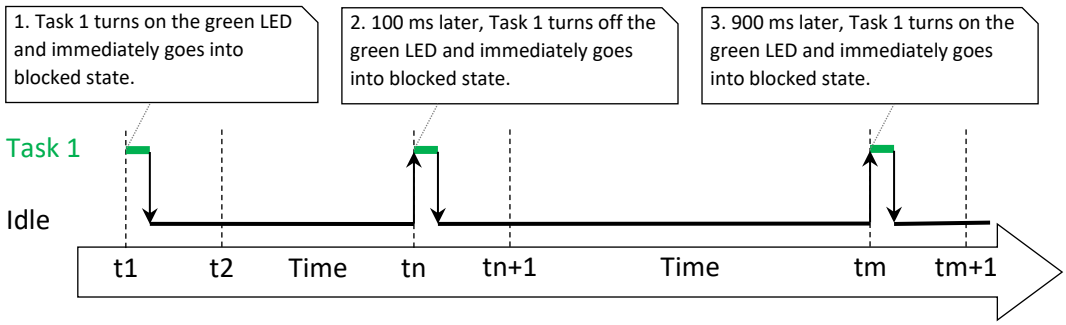
The remainder of this section describes how to implement each task step-by-step.

Task 1: Blink the green LED

Task 1 is responsible for blinking the green LED on and off. This is an indication that the system is up-an-running and that at least this task functions properly.

- The green LED must be on for 100 ms.
- The green LED must be off for 900 ms.

An example of the Task 1 timing is depicted in the following timing diagram.



Missing a deadline for Task 1 is considered the least problematic for this example application. It will therefore get the lowest priority 1 (besides the Idle task).

Solution

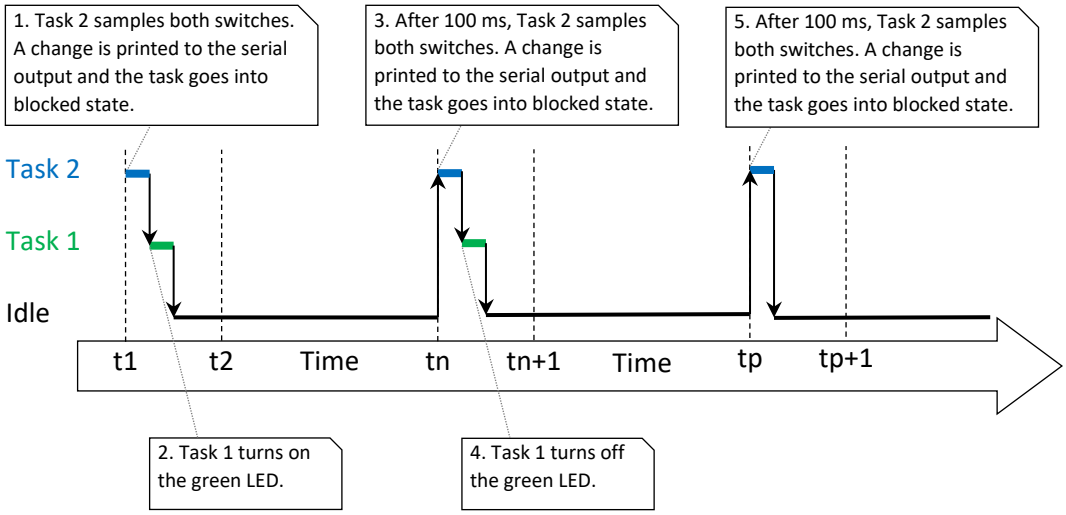
Task 2: Detecting switch changes

Task 2 is responsible for detecting switch changes: pressing and releasing one of the two switches. This is considered the most important task of this application and a deadline must not be missed. It must therefore get the highest priority 3. The task must run exactly 10 times per second and in each run it samples both microcontroller pins.

A changing switch is printed to the serial output. An example of what the output might look like is as follows:

```
FRDM-KL25Z FreeRTOS Week 1 - Lab exercise

[Task 2] SW1 pressed
[Task 2] SW1 released
[Task 2] SW2 pressed
[Task 2] SW2 released
[Task 2] SW1 pressed
[Task 2] SW2 pressed
[Task 2] SW2 released
[Task 2] SW2 pressed
[Task 2] SW2 released
[Task 2] SW1 released
```

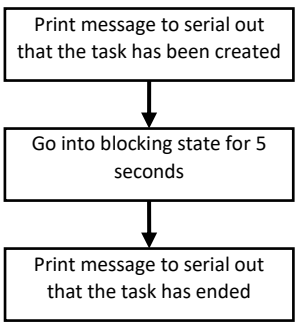
An example of the Task 2 and Task 1 timing is depicted in the following timing diagram.



Solution

Task 3: Creating a task on a falling edge of SW1

Task 3 is created when a falling edge is detect on SW1. The task behaves as depicted in the following flowchart:



The priority of Task 3 is 2.

The creation of Task 3 will be printed to the serial output. An example of what the output looks like when pressing the SW1 is as follows:

```
FRDM-KL25Z FreeRTOS Week 1 - Lab exercise

[Task 2] SW1 pressed
[Task 2] Task 3 created
[Task 3] Started
[Task 2] SW1 released
[Task 3] Ended
```

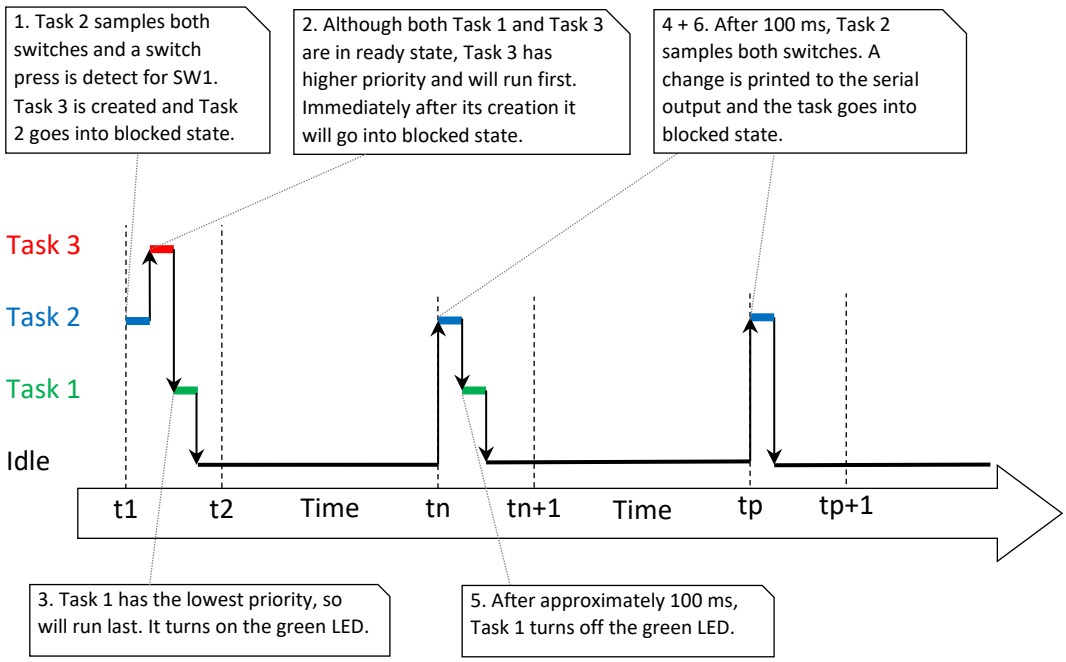
An example of what the output looks like when pressing SW1 twice in quick succession is as follows:

```
FRDM-KL25Z FreeRTOS Week 1 - Lab exercise

[Task 2] SW1 pressed
[Task 2] Task 3 created
[Task 3] started
[Task 2] SW1 released
[Task 2] SW1 pressed
[Task 2] Task 3 created
[Task 3] started
[Task 2] SW1 released
[Task 3] ended
[Task 3] ended
```

The xTaskCreate() function returns pdPASS if the creation of the task was successful. If the creation of the task was not successful, a message is printed by Task 2 to serial output. An example of what the output looks like when Task 3 could not be created:

```
FRDM-KL25Z FreeRTOS Week 1 - Lab exercise

[Task 2] Task 3 not created
```

An example of all tasks timing is depicted in the following timing diagram.



Solution

Questions

After having finished the implementation of all three tasks, answer the following questions.

1. Press switch SW1 many times within 5 seconds. How many Task 3 tasks can successfully be created with the given FreeRTOS configuration settings?
2. Why should you use the vTaskDelayUntil() function in Task 2?
3. The following task states are used in this example. Explain these task states.
    a. Running
    b. Blocked
    c. Ready
4. What scheduling algorithm has been selected?

Answers