MIC5 Week 4 – lab exercise

Introduction

In this lab exercise we will keep track of date and time and schedule alarms by using the Real Time Clock (RTC) peripheral. The RTC keeps track of time by incrementing a 32-bit counter every second. We use this counter and library functions to convert the date and time so it can be printed to the serial monitor. For example:

```
FRDM-KL25Z FreeRTOS Week 4 - Lab
[vShowDatetimeTask] 2022-02-22 12:00:02
```

This visualisation is updated in a task every second by using an interrupt that is generated by the RTC.

As soon as the user presses the spacebar, an alarm will be scheduled in 10 seconds. Scheduling of the alarm and handling the alarm, are visualized in the serial monitor:

```
FRDM-KL25Z FreeRTOS Week 4 - Lab
[vShowDatetimeTask] 2022-02-22 12:00:02          -> LED initially blinks green
[vSetAlarmTask      ] Alarm set in 10 seconds    -> User presses spacebar
[vShowDatetimeTask] 2022-02-22 12:00:11          -> From now on the LED blinks red
[vAlarmHandlerTask ] ALARM HANDLING
[vShowDatetimeTask] 2022-02-22 12:00:12          -> From now on the LED blinks green
```

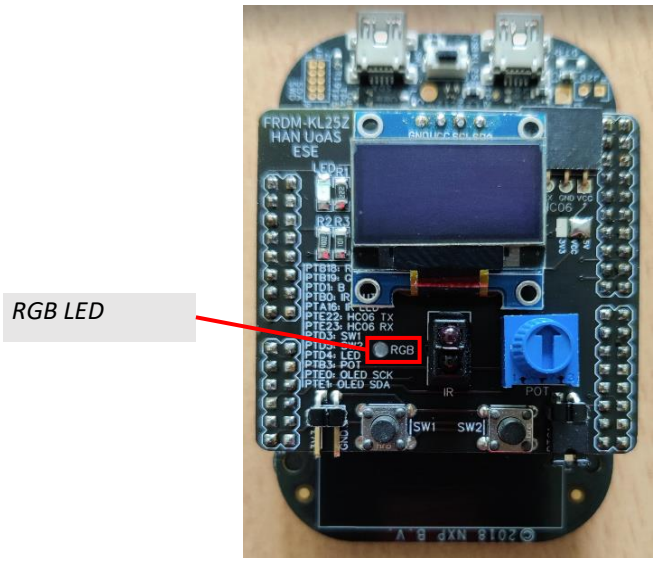
While an alarm is scheduled, a red LED blinks. Otherwise, a green LED blinks.

Hardware

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

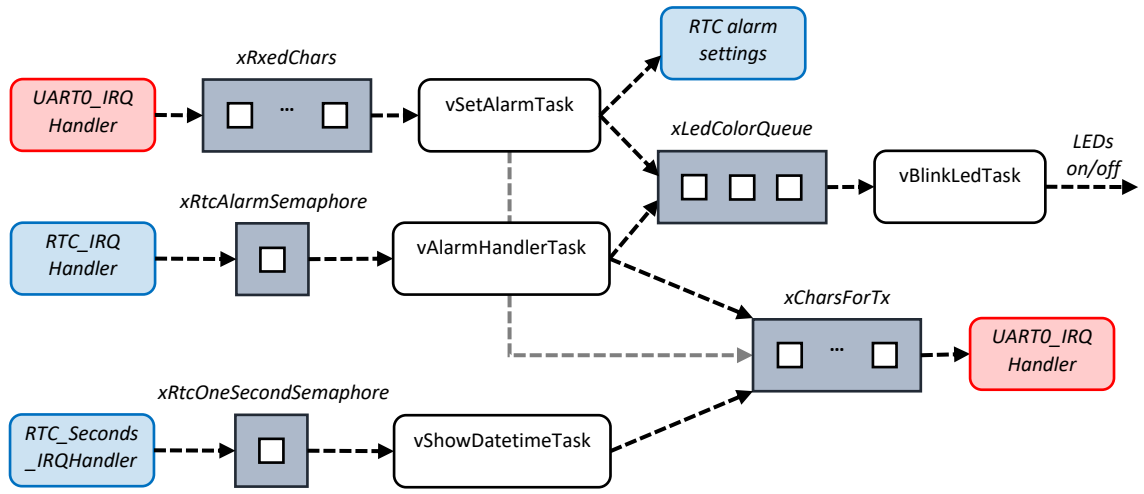
Description	MKL25Z128VLK4 pins	Notes
RGB LED	PTB18 (red) PTB19 (green) PTD1 (blue)	-

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



Software

The software design is depicted in the following image. A description of the design elements follows next.



- The UART0_IRQHandler() is already implemented in the file *serial.c*.
    - The ISR receives data that is send to the xRxedChars queue. Use the function xSerialGetChar() to check if data is available.
    - The ISR transmits data as longs as the xCharsForTx queue is not empty. Use the function vSerialPutString() to write an entire string of characters thread safe to the queue.
- The configuration of the RTC is already partially implemented in the file *rtc.c*.
    - The RTC_Seconds_IRQHandler() ISR is executed every second. It should *give (+1)* the xRtcAlarmSemaphore semaphore.
    - The RTC_IRQHandler() ISR is executed when an alarm occurs. It should *give (+1)* the xRtcOneSecondSemaphore semaphore.
- The **vSetAlarmTask** task implements a blocking wait for any incoming data. Use the function xSerialGetChar() for this purpose. If the incoming data is equal to a space (‘ ‘), an alarm will be scheduled by updating the RTC alarm settings. A new alarm can be scheduled by updating the Time Alarm Register (TAR) as follows:

```
// Next alarm in 10 seconds
RTC->TAR = RTC->TSR + 9;
```

  After the alarm has been set:
    - Send a string to the xCharsForTx queue, such as: "\r\n[vSetAlarmTask] Alarm set in 10 seconds\r\n".
    - The value RED is send to the xLedColorQueue queue.
- The xLedColorQueue queue holds 3 items of the color_t type, which is defined as follows:

```
typedef enum
{
    RED,
    GREEN,
    BLUE,
}color_t;
```

- The **vAlarmHandlerTask** task blocking waits for the xRtcAlarmSemaphore. If the semaphore can be *taken (-1)*, it does two things:
    - Send a string to the xCharsForTx queue, such as "\r\n[vAlarmHandlerTask] ALARM HANDLING\r\n".
    - Send the value GREEN to the xLedColorQueue queue.
- The **vShowDatetimeTask** task blocking waits for the xRtcOneSecondSemaphore. If the semaphore can be *taken (-1)*, it converts the RTC datetime to a readable representation and sends a string to the xCharsForTx queue, for example:

```
rtc_get(&datetime);
sprintf(str, "[vShowDatetimeTask] %04hd-%02hd-%02hd " \
             "%02hd:%02hd:%02d\r",
             datetime.year, datetime.month, datetime.day,
             datetime.hour, datetime.minute, datetime.second);
vSerialPutString(str);
```

- The **vBlinkLedTask** task blinks the RGB LED.
    - An LED is on for exactly 100 ms.
    - An LED is off for exactly 900 ms.

  The color of the LED that blinks is updated when an item can be received from the xLedColorQueue queue. Initially, the green LED blinks.

A template project is provided:

Week4 – Lab.zip

Updating RTC interrupt handlers

Update the RTC interrupt handlers so they use the semaphores as described above. The file *rtc.c* already declares the two semaphores xRtcOneSecondSemaphore and xRtcAlarmSemaphore. There is also an external declaration in the header file *rtc.h*, so in main you can simply refer to these two semaphores.

Answer

Updating main

Implement the functionality in main by:

- Defining color_t.
- Creating the semaphores.
- Creating the LED color queue.
- Creating the four tasks.
- Implementing the four tasks.

Answer