MIC5 Week 5 – lab exercise

Introduction

In this lab exercise we will use the ssd1306 oled display and make sure that this resource is managed properly. Three tasks with the same priority will write a counter value to the oled display. The counter values are updated with the following frequencies:

- Task 1 increments the counter every second
- Task 2 increments the counter every two seconds
- Task 3 increments the counter every three seconds

The expected visual output on the oled display is as follows:

After 1 second:

```
Week 5 – Lab
Task 1:    1
```

After 2 seconds:

```
Week 5 – Lab
Task 1:    2
Task 2:    1
```

After 3 seconds:

```
Week 5 – Lab
Task 1:    3
Task 2:    1
Task 3:    1
```

After 4 seconds:

```
Week 5 – Lab
Task 1:    4
Task 2:    2
Task 3:    1
```

After 5 seconds:

```
Week 5 – Lab
Task 1:    5
Task 2:    2
Task 3:    1
```

After 6 seconds:

```
Week 5 – Lab
Task 1:    6
Task 2:    3
Task 3:    2
```

After 10 seconds:

```
Week 5 – Lab
Task 1:   10
Task 2:    5
Task 3:    3
```

After 12 seconds:

```
Week 5 – Lab
Task 1:   12
Task 2:    6
Task 3:    4
```

After 30 seconds:

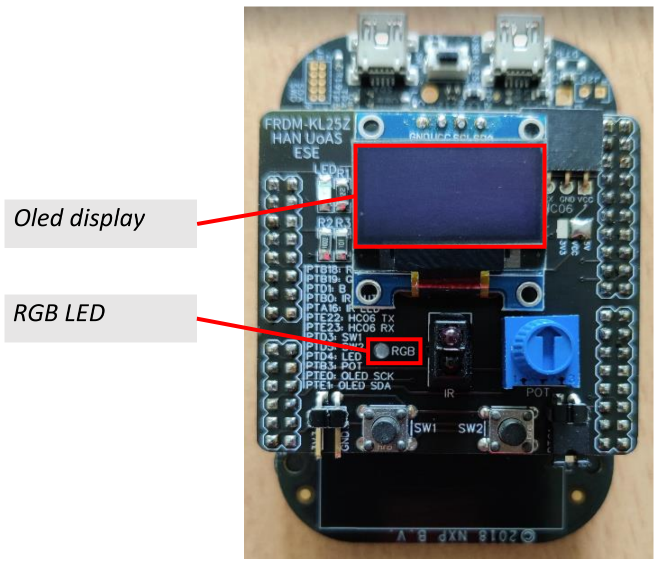
```
Week 5 – Lab
Task 1:   30
Task 2:   15
Task 3:   10
```

Hardware

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

Description	MKL25Z128VLK4 pins	Notes
Oled display	PTE0 (I2C1_SDA) PTE1 (I2C1_SCL)	-
RGB LED	PTB18 (red) PTB19 (green) PTD1 (blue)	*This project uses the blue LED only*

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

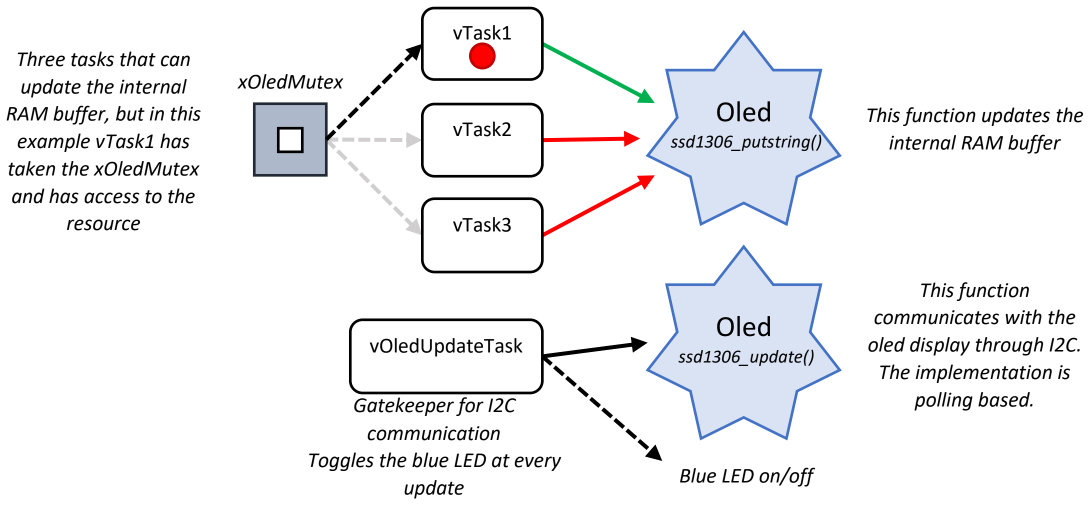


Software

A library is provided for using the oled display, see the folder called *'oled'.* This library does not make use of any FreeRTOS functions and is therefore not thread safe. The library must be used as follows:

- Initialise the oled display by calling the ssd1306_init() function.
- Configure the display, for example its orientation and font.
- The library uses an internal RAM buffer to temporarily store all display data. Use the function ssd1306_putstring() to update this internal RAM buffer. The execution time of this function depends on several parameters, such as the string length and the font size. For this exercise, assume a maximum duration of 10 ms for writing a single string of 50 characters.
- The updated RAM buffer is transmitted to the oled display through the I2C interface by calling the function ssd1306_update(). The I2C implementation is polling based and takes a maximum of 30 ms to execute.

The software design is depicted in the following image. For clarity, the queues and semaphore related to serial communication have been omitted. A description of the design elements follows next.


Three tasks that can update the internal RAM buffer, but in this example vTask1 has taken the xOledMutex and has access to the resource
xOledMutex
vTask1
vTask2
vTask3
Oled
ssd1306_putstring()
This function updates the internal RAM buffer
vOledUpdateTask
Gatekeeper for I2C communication
Toggles the blue LED at every update
Oled
ssd1306_update()
This function communicates with the oled display through I2C. The implementation is polling based.
Blue LED on/off

- The tasks vTask1, vTask2 and vTask3 all are very similar:
  - The task is initialized and a message is printed to the serial monitor.
  - In an endless loop:
    - Go into Blocking state for an exact amount of time:
      - vTask1 for 1000 ms
      - vTask2 for 2000 ms
      - vTask3 for 3000 ms
    - As soon as the task moves into Running state it increments a local counter and prints the counter value to the oled display by calling the ssd1306_putstring() function. Printing is done only as soon as the *xOledMutex* has been *taken*. If the *xOledMutex* could not be *taken* within 20 ms, a warning message is printed to the serial monitor.
- The vOledUpdateTask task acts as a gatekeeper for the communication with the oled display. It calls the ssd1306_update() function exactly every 100 ms. At every call to this function, the blue LED toggles.

A template project is provided:

Week5 – Lab.zip

Create the gatekeeper task

Implement the gatekeeper task called vOledUpdateTask. Create the task in main() and set its priority to 1.

Answer

Create the tasks without the mutex

Implement the three tasks vTask1, vTask2 and vTask3. Do not implement the mutex yet, but simply increment the counter and call the function ssd1306_putstring() as soon as a task is moved into Running state.

Create the three tasks in main() and set the priority of all tasks to 2. Run the application. Explain what is visible on the oled display after one, two and three seconds.

Answer

Create the mutex

Create the mutex xOledMutex.

Answer

Use the mutex in the tasks

Instead of simply calling the function ssd1306_putstring() in the tasks vTask1, vTask2 and vTask3, manage the resource by taking the xOledMutex. As described above, writing a single string to the internal RAM buffer takes no more than 10 ms. Use this information to set the timeout for taking the mutex. If the timeout fails, a message must be printed to the serial monitor.

Answer

Verifying the runtime characteristics

The execution time of the gatekeeper task vOledUpdateTask is approximately 25 ms. As this task runs every 100 ms it should take approximately 25% of the total runtime.

vTask1 should approximately have twice the runtime of vTask2. And vTask3 twice that of vTask2. The total run time of either task will be very short, because it spends most of its time in Blocking state.

Use the FreeRTOS kernel aware debugger to verify the above. Start the debugger and run the application for several seconds at least. Set a breakpoint in any of the tasks and verify the runtime in the *RTOS > FreeRTOS >Task List* output pane.

Answer