LAB N° 4

Direct task notification

Lab Objectives

- Using task notification instead of semaphore.
- Using event group for a notification.

4.1 Direct task notification (Lab4-1)

A *direct to task notification* is an event sent directly to a task, rather than indirectly to a task via an intermediary object such as a queue, event group or semaphore. The *direct to task notification* is faster than using an intermediary objects and has a RAM Footprint benefits.

- Duplicate the « lab3-2_two_sem_clk » folder to « lab4-1_two_notifications_clk ».
- Answer the following questions :
  — Study the parameters of the *xTaskNotifyGive()* function. Web help

  — Study the parameters of the *ulTaskNotifyTake()* function. Web help

- Replace the two separate semaphores (i.e. notification of the *IncTable* and *DecTable* tasks) by a the *direct to task notification* mechanism. For the *ulTaskNotifyTake()* function, we set the first parameter *xClearCountOnExit* to *TRUE*. Print the return value (pending event counter) of *ulTaskNotifyTake()* function for each task.

- Run the program and explain the behavior.
- In the *Timer* task, add a new *give()* notification to *IncTable* task. So, you have 2 notifications in a row to the *IncTable* task.
- Run the program and explain the behavior.
- We set now the first parameter *xClearCountOnExit* to *FALSE* for the *ulTaskNotifyTake()* function. What is the difference of behavior regarding to the previous question ?

4.2 Direct task notification with a event value (Lab4-2)

To illustrate the principle, we are going to modify the algorithm of the *Timer* task. Its functional behavior is as follows :

```
Task Timer is
    Properties: Priority = 5, ACTION = eSetBits
    Out        : Clk is event
    Cycle :
        waitForPeriod(250 ms);
        computeTime(20 ms);
        print("Task Timer : Notify Give (count=%d)", count);
        // IncTable task notifications
        notify(incTableHandler, (0x01 « count), eSetBits);
        notify(incTableHandler, (0x02 « count), ACTION);
        // DecTable task notification
        notify(decTableHandler, (0x01 « count), eSetValueWithoutOverwrite);
        // counter modulo 4
        count = (count + 1)
    end
end
```

Algorithm 4.1: New Timer algorithm.

1. Duplicate the « lab4-1_two_notifications_clk » folder to « lab4-2_two_notifications_clk2 ».
2. Answer the following questions :
   - Study the parameters of the *xTaskNotify()* function. Web help
   - Study the parameters of the *xTaskNotifyWait()* function. Web help
3. Modify the code of *Timer* task.
   - The *ACTION* property can be a constant value and can take the *eSetBits*, *eSetValueWithoutOverwrite* or *eSetValueWithOverwrite* value.
   - We set by default the *ACTION* property to *eSetBits*.
   - What is the best value of the first parameter (*ulBitsToClearOnEntry*) of the *xTaskNotifyWait()* function ?

- We set the second parameter (*ulBitsToClearOnExit*) of the *xTaskNotifyWait()* function to 0.

4. Run the program, copy the console until 107 ticks and explain the behavior, in particular the pending counter value in each 2 tasks.

5. We set the *ACTION* property to *eSetValueWithoutOverwrite*. Run the program, copy the console until 107 ticks and explain the behavior.

6. We set the *ACTION* property to *eSetValueWithOverwrite*. Run the program, copy the console until 107 ticks and explain the behavior.