Event groups

Exercise 1

Write a program with four tasks and an event group. Task 1 waits for user to press a button. When the button is pressed task 1 sets bit 0 of the event group. Tasks 2 – 4 wait on the event bit with infinite timeout. When the bit is set the tasks start running. Each task prints task number and number of elapsed ticks at random interval that is between 1 – 2 seconds. Remember to protect access to the serial port with a semaphore.

Exercise 2

Write a program with three tasks and an event group. Each task monitors the state of one button. Task 1 monitors SW1, Task 2 monitors SW2 and Task 3 monitor SW3. Each task requires the button to be pressed as many times as the task number is before continuing. When task continues it synchronizes execution with event bits: task1 uses bit1, etc.

When all tasks have reached the synchronization point each of the tasks prints the task number and number of elapsed ticks and then goes to sleep. Remember to protect access to the serial port with a semaphore.

Exercise 3

Write a program with four tasks and an event group. Task 4 is a watchdog task that monitors that tasks 1 – 3 run at least once every 30 seconds. Tasks 1 – 3 implement a loop that runs when a button is **pressed and released** and sets a bit the event group on each loop round. The task must not run if button is pressed constantly without releasing it. Each task monitors one button.

Task 4 prints “OK” and number of elapsed ticks from last “OK” when all (other) tasks have notified that they have run the loop. If some of the tasks does not run within 30 seconds Task 4 prints “Fail” and the number of task plus the number of elapsed ticks for each task that did not meet the deadline and then Task 4 suspends itself.