Electrical, Electronic & Computer Engineering
School of Engineering & Physical Sciences

B31DG: Embedded Software 2023
Assignment 3, Released on Week 8, Demo due on Week 12, Report Week 13

1. Problem

Your program needs to satisfy the same functional and real-time requirements of assignment #2, i.e. it needs to perform the same (hard RT) tasks at the same rates.

In addition, your program should satisfy the following additional requirements:

1. Monitor one digital input - to be connected to a pushbutton. Make sure the input is debounced. The frequency at which your program monitor this digital input is left to your choice.
2. Control a LED. The state of the LED should be toggled by the pushbutton.

Your program needs to use FreeRTOS (so you can use any number of FreeRTOS tasks, timers, queues, semaphores, etc.). It is your choice whether you implement the required tasks using actual FreeRTOS tasks or callback functions called by FreeRTOS software timers, or a combination of the two approaches.

In particular, your program needs to:

- Monitor the digital input and control the LED from two independent tasks.
- Use an event queue to enable the communication between those two tasks, i.e. when the pushbutton is pressed the monitor digital input task will queue an event, which needs to be received and acted upon by the control LED task.
- Use a global structure (struct), to store the frequencies measured by Task #2 and Task #3 (from Assignment #2) and printed to the serial port by Task #5 (also from assignment 2). Access to this structure must be adequately protected, using FreeRTOS' semaphore(s).
- Use any other FreeRTOS constructs you consider useful for your program.

3. Demo (Week 12)

At the demo lab session (during Week 12), you will be asked to:

- Run your code and show it meets all its specification. You will demonstrate all the features and show the signal generated (requirement 1) and the execution time of task 2 using the oscilloscope.
- Modify the periods of any of the tasks you have implemented, and to run again your system.
- Answer questions on your system, e.g. design, testing, performance.

Note that it is not mandatory to use the monitoring library you used for Assignment #2.

5. Submission (Week 13)

Submit the following paperwork:

- Fully documented source code
- Short (maximum 5 pages) report describing the design of your program (using one or more diagrams, e.g. data-flow, UML, as you consider necessary to document your design), and summarising and analysing results (tasks successfully implemented, performances ...).

  Your report should include answers to the following points:

- How did you decide to set the priorities of your FreeRTOS tasks? Why?
- How did you protect the access to the global structure? Why?
- What tests did you perform to check whether all the RT requirements of assignment #2 are respected in this new implementation using FreeRTOS? Make sure to describe these tests in detail, providing evidence of the results of any tests you have performed, if any.
- What is the worst case delay (response time) between the time the push button is pressed and the time the LED is toggled? Justify your answer.
- How does your FreeRTOS implementation compares with the custom cyclic executive system you implemented for assignment #2? Is the performance different? Which is easier to implement, maintain? Which is easier to adjust given a change in requirements to the system? What other differences do you notice?

- Signed authorship statement

6. Marking criteria

- Work as an individual. Students **MUST** work on this project on their own. Please do not use code from another student or offer code to another student. Code may be run through a similarity checker. You will be asked questions about your code during the demo session. The assignment will not be marked if plagiarism is suspected. You need to acknowledge any third-party source material (e.g. snippets of code you find online, including on the course folders on Canvas) in your code (in comments) and in your report.
- The submission deadline is at the end of Week 13.
- The assignment contributes 20% towards the 50% continuous assessment mark.
  - Quality of the design and implementation is worth 13%, with full marks for:
    - Functional correct program, i.e. all 5 tasks from Assignment #2 should work correctly (adhering to functional and RT requirements) and the new digital input monitor and LED control tasks should work as specified. 2% points max for each of the 5 tasks in Assignment #2, 3% points max for the new digital input monitor control tasks (max 13% points).
    - Efficient, modular, easy to maintain and re-usable code; up to 2% points will be removed if the code shows issues for any of these aspects.
      In particular, your program should:
      - Share resources and decouple/ synchronise tasks, by keeping the use of global variables as limited as possible, and with functions/FreeRTOS tasks as independent as possible from each other.
      - Avoid wasting CPU (using other delay functions in addition to the ones you have already used to implement the tasks in Assignment #2) and also memory (including by properly sizing stack sizes).
    - Good programming style, including appropriate names for variables, good layout /indentation, and commenting style; up to 2% points will be removed if the code shows issues for any of these aspects.
    - Documentation is worth 7%, with full marks for:
      - Clear description of design and tests performed
      - Answers to all the questions highlighted in point #5
      - Well written, complete and well organised report, effective use of language, no typos, good use of diagrams, references to third-party sources.