Semaphores and Bounded Buffer

1 Introduction

In this lab you are required to implement a bounded buffer and to use it to solve some simple synchronisation and communication problems.

2 In the lab

1. In this lab you should be working with the initial workspace provided for the assignment (workspace.zip).
2. Start up EWARM and load the workspace `workspace/workspace.eww`.
3. Connect a `LPC-2378-STK` board to a USB port on your computer.
4. Download and debug the project. Run the program and observe its behaviour. It's designed to get each of the led tasks to report its status using the LCD. The task reports whether or not it is flashing and its current flashing delay, e.g.

   ```
   (LINK) F:ON  D:3300
   (CNCT) F:OFF D:4100
   ```

   This is similar to the program that you were asked to construct for exercise Lab07.9. You may already have completed the next two tasks as part of that exercise (or as part of Lab07.10).
5. Do you observe any interference between the led tasks? If not, increase the flashing rate of one of the leds until you do observe some interference. Why does this occur? Remember the LCD is now a shared resource.
6. Use a semaphore to provide mutually exclusive access to the LCD, so that the interference is prevented. Comment on the effectiveness of your solution.
7. Add new files `buffer.h` and `buffer.c` to the appropriate directories and project groups. Implement the naive circular buffer that was described in lecture 7. Add some simple test code to your tasks to test the behaviour of the buffer (For now, execute the `put` code in a task of higher priority than the task where you execute the `get` code.)

8. The naive buffer has a number of potential problems: interference between multiple producers or consumers; failure if a producer attempts to put to a full buffer; and failure if a consumer attempts to get from an empty buffer. Now implement 3 new functions, `blockingBufferInit()`, `blockingBufferPut()` and `blockingBufferGet()` to provide a solution to these problems, as described in lecture 7 from the slide `Elements of a solution` onwards. You'll need to decide on appropriate parameters for these functions yourself. Use your naive buffer functions to implement the operations to add and remove items from the buffer.

9. Now modify your main program by:

   (a) removing all synchronisation code - use of semaphores etc.

   (b) adding a new LCD task

   The new LCD task is the only task now that will be allowed to write to the LCD. Modify your led tasks so that, instead of writing to the LCD, they send messages to the LCD task reporting their status. The job of the LCD task is to read these messages and update the LCD accordingly. You should use your `blockingBuffer` implementation for passing messages between tasks.

Ask your lab tutor for help in interpreting these instructions and for guidance about your implementation.