More task management in uC/OS-II
Mutual Exclusion

1 Introduction

This lab is the second lab concerning the real-time kernel uC/OS-II and its use on the `LPC_2378_STK` development board. It looks at the problems that can arise when tasks share resources. It also introduces the use of two new devices: the LCD display and the potentiometer. At the end of the lab, you should know:

- how to write `main.c` to create tasks and execute them
- how to suspend tasks by using delays
- how to identify interference problems when tasks share resources
- how to protect a critical section using Peterson's algorithm
- how to obtain a reading from the potentiometer
- how to display information textually and graphically on the LCD

2 In the lab

1. Download the file worspace.zip into a suitable directory either on a pen drive or in your University workspace. I suggest you call the directory `EN572/labs/lab06`. Unzip `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. Make sure that you understand the solution to `lab05` – it's in the project `lab05S`. Download and debug this project. Test the program. Make sure that you understand how the `appTaskButtons` task has been added to the project. In particular, pay attention to the declaration of :
    - the priority `APP_TASK_BUTTONS_PRIO`
    - the stack size `APP_TASK_BUTTONS_STK_SIZE` and the stack `appTaskButtonsStk`
    - the function prototype `appTaskButtons`

- the function `appTaskButtons`

In addition, you should notice:

- the use of `OSTaskCreate` with `appTaskButtons` – where does this occur?
- the declaration of local functions `incDelay` and `decDelay` to increase and decrease the delay between LED flashes
- the declaration of global, shared variables `flashing`, `linkLedDelay` and `connectLedDelay` for communication between tasks. Which tasks read and write these variables?

5. When you understand the code for `lab05S`, you should move on to `lab06`.

6. Download and debug `lab06`. Run the program and observe its behaviour. What do you notice? What do you think is causing the behaviour that you see?

7. Now switch projects to `lab06a`. This is an attempt to fix the problem exhibited by `lab06`. Download and debug `lab06a`. Run the program and observe its behaviour. What do you notice this time?

8. Carefully study the code of both `lab06` and `lab06a`. Write down all the differences that you notice. Explain why `lab06a` works correctly.

9. Comment out the `OSTimeDlyHMSM(0,0,0,1)` calls in both `appTaskCount1` and `appTaskCount2` (leave the other calls to `OSTimeDlyHMSM()` alone). Download and debug the project. Run the program. What do you notice about the behaviour? Give a detailed explanation of the reasons for the behaviour that you see. *N.B.* This is quite tricky. You'll need to think hard about it. Peterson's solution is usually reported without delays like these. What assumptions must we make about the scheduling policy in order for such a solution to work correctly?

10. Clean the project `lab05S`. Copy the `lab05S` directory to a new directory: `lab06b`. Delete the `Flash`, and `settings` directories, and the file `lab05S.dep`, from `lab06b`. Rename `lab05S.ewp` to `lab06b.ewp` and `lab05S.ewd` to `lab06b.ewd`. Add the new `lab06b` project to your existing workspace.

11. Now download and debug the `lab06b` project. Run the program. Make sure that it behaves just like the original `lab05S` project. The remaining exercises use `lab06b`.

12. Add a fourth task to read and display the value of the potentiometer on the LCD. Use only the declarations in `potentiometer.h` and `lcd.h` for the new functionality. You do NOT need to look at the implementations `potentiometer.c` and `lcd.c`. Notice that `lcdWrite` is implemented as a macro replacement for `printf`. This means that you can use the usual `printf` format strings with `lcdWrite`.

13. When your potentiometer task is working to your satisfaction, modify it so that it also provides a graphical view of the potentiometer reading. This should take the form of a horizontal bar chart that spans the display and lengthens as the potentiometer is turned clockwise, i.e. the length of the bar should be 0 when the knob is turned fully anti-clockwise and should be at its maximum length when the knob is turned fully clockwise. Test your solution.