COE4DS4 – Lab #6 Report
Group 18
Khalid Asad (1147299) and Vishal Kharker (1140837)
asadk@mcmaster.ca, kharkevp@mcmaster.ca
March 17, 2016

EXERCISE 1:

The major changes in this exercise were to change the execution time of task 0 to 250*6 = 1500 and 250*4 = 1000 which are the amount of letters in the name 'Khalid Asad'. The os delay values for task 1 were 850*6 = 5100 and 850*7 = 5950 for 'Vishal Kharker'. By using OSSchedLock(), we can lock the processes for the non-pre-emptive case. These tasks are running one after another respective to their priority because they are periodic. In this case, the tasks will not be interrupted. However, in the pre-emptive case, tasks can be interrupted. By using OSSchedUnlock(), this case can be used. If task 0 executes after task 1, the former will be pre-emptive and task 1 will have to wait in the queue until the other task is completed.

EXERCISE 2:

In this exercise, we replaced the original semaphores with a queue using an array and an index to keep track of the push buttons that were pressed. We kept the scheduler, removed the task delete function and created a new function to handle priority assignments. The schedule function goes through the queue every 3 seconds so it can keep track of when multiple push buttons are pressed. We created an array to keep track of priorities. An initial sequence of 0, 1, 2, 3 would cause 3 to have the highest priority and 0 to have the lowest priority. If 3 was pressed again, it would be assigned the lowest priority. Any subsequent push button pressed next would be assigned the lowest priority. If multiple push buttons are pressed at once, the program pushes these requests into a queue/stack to process them sequentially.

EXERCISE 3:

i) Push buttons are pressed in increasing order (switches 16, 14, 10, 2, 1, 0 ON):
Red LEDs 0, 1, 2 ON, then Green LEDs 0, 1, 2 and Red LEDs 1, 5, 7 both ON but Red LEDs dim before Green LEDs. After these dim, Green LEDs 1, 5, 7 ON for 2.5s. Task 0 shares Switch A Mutex with Task 1 and Task 0 has a higher priority, therefore Task 1 should wait until Task 0 has executed. Task 2 does not share a Mutex so it should not have to wait for Task 1. Task 3 shares the Switch B Mutex with Task 2, so it has to wait until it finishes. This exercise demonstrates priority inversion in action.

ii) Push buttons are pressed in decreasing order (switches 16, 14, 10, 2, 1, 0 ON):
Green LEDs 1, 5, 7 ON, then Green LEDs ON 0, 1, 2 ON after Green LEDs turn off (2.5 seconds), then Red LEDs 0, 1, 2 ON after green turn off, Red LEDs turn off, then RED LEDs 1, 5, 7 ON for 3.5 seconds.
Task 3 starts first, then Task 2 wants to go, however it is sharing a switch mutex with Task 3, so it has to wait until that finishes, but in that time PB1 is pressed and shares a green LED mutex with Task 3 and it has a higher priority than Task 2, so Task 1 will be the next to execute. While that is running, PB0 is pressed, and it has the highest priority and also shares a switch mutex with Task 1, so this will be the next to run. Finally Task 2 can execute and finish the set of actions.