Real-Time Operating Systems

ESD 813

Assignment 2

(Chandramouleeswaran Sankaran)

Mouli.sankaran@iiitb.ac.in

Date: 9 Mar 2016

Date of Submission: 10th Apr 16

Assignment 2

< Course: RTOS, ESD 813>

Write C programs using **FreeRTOS APIs** for each of the problems below, following the **GNU coding standards**.

Notes:

1. **Check** for **error** conditions and print appropriate **messages** for all the programs.
2. The **Logic Analyzer output** should include the details of **all the tasks** running in the system.

Attach the following for each of the problems.

a. The **code** implementation for each of the problems.
b. **Logic Analyzer output** generated in the KEIL simulator environment.
c. **Printf output** samples for the run.
d. **Explanation** for the state changes in the Logic Analyzer outputs.

1. Create **four** tasks in the system, namely **T1, T2, T3** and **TP.**
    a. **T1, T2** and **T3** are having the **same priority**.
    b. **Create four Queues Q1, Q2, Q3 and QP.**
    c. **T1, T2 and T3 would wait on Q1, Q2 and Q3 respectively.**
    d. **TP** has the priority **lower** than these **three tasks**.
    e. **TP** is a **Periodic task** waking up **every 1 second**.
        i. **TP** maintains a **running count (*count*)** initialized to **1**.
        ii. On waking up, **TP** sends the ***count*** to **one of the Queues**, in the order **Q1, Q2** and **Q3**.
        iii. After sending the ***count***, ***count*++, TP waits** on **QP**.
        iv. When **TP receives** a message on **QP**, it prints the **Task ID** and **Value** received, **TP** prints "**The Task %d has sent the value %d.\n**", ID, value)
        v. Then **TP** sleeps for another **1 second** and **repeats** the **above steps.**
    f. **T1, T2** and **T3** wake up on receiving message on **Q1**, **Q2** and **Q3** respectively.
        i. The ***count*** received from **TP** is **multiplied** by its own **Task ID**.
        ii. Then, its own **Task ID** and the **multiplied value** are sent to **QP**.
        iii. After sending to **QP**, tasks wait on its own Queues (**Q1** or **Q2** or **Q3**), for the next message from **TP**.

2. Create **five** tasks **T1, T2, T3, T4 and T5.**
    a. The **priorities** of the **tasks** are **P1, P2, P3, P4** and **P5**.
    b. The **priorities** are related as **P1 > P2 > P3 > P4** > **P5,** with **P1** being the **highest** and **P5** is the **lowest**.
    c. All of them are **Periodic tasks**, with periods **1 to 5 seconds**.
        i. **Period** of **T1** being **1 second** and **T5** being **5 seconds**.
    d. Whenever a task **wakes** up, it **prints** it **ID** (**1 or 2 or 3 or 4 or 5**) before sleeping again for the **Period**.

3. Implement a function which accepts an existing **Queue handle** and prints out the following information about the Queue passed to it**.**
    a. Message length.
    b. No of messages.
    c. No. of tasks waiting
    d. Print the task names in the order in which they are waiting on the queue.
    e. Print the contents of the messages also (in the hex format)
    f. Take care of critical section issues since this function (API) can be called by any tasks.
    g. Create multiple queues and make messages and tasks to wait on them and call this function to verify that it works fine.
4. Implement **wrapper** functions to the **QueueSend** and **QueueReceive** functions which store the **task ID** of the sender also along with the original message. So that the receiving tasks can find out the sender of the message using your implementation of QueueReceive.
    a. Using this function print out the name of the sending task while receiving the message from other tasks.
    b. Make sure that the original message contents are preserved.

5. Give a write up on the Mutex/Semaphore implemented by FreeRTOS by looking at the code and data structures used. It should cover the what is supported and what changes to be done to implement **Priority Inheritance and Priority Ceiling protocols**.
    a. Give a sample implementation along with the design document with necessary details.

****