ELC 2080
RTOS Communicating Tasks Project

Important Note: This is the initial version. Final version will contain the exact requirements of the project. So, expect that some details may change but not very much.

1 Objectives

Apply knowledge learned in the embedded programming part of the course and get hands on experience on RTOS concepts such as:

1- Tasks
2- Timers
3- Queues
4- Semaphores

2 Rules

1- Maximum two students per group. Students should sign in one of the available Blackboard groups. The group ID is arbitrary.

2- Submission via any other means is rejected immediately.

3- Submit one file called main.c based on project template supplied for you by Eng. Hassan to work on.

4- Submit documentation in MS-WORD .docx format. Any other format including .doc and .PDF will be automatically rejected. Name your file RTOS_Proj_Rprt.docx.

5- Documentation is to be submitted using provided .docx template and should not exceed 5 pages. Other formats or longer reports will be penalized.

6- Copying other groups code or document will result in all students in the two groups getting -10 grade. Thank you for not cheating. Do not try to test this, if you copy we will find that out and you will be penalized.

7- Submission after deadline is penalized at -10% and keep increasing. Only 3 days after deadline is allowed.

3 Project Specifications

The project is implemented using FreeRTOS on the target emulation board provided via Eclipse CDT Embedded.

Two tasks communicate via a queue of fixed size as described below:

The sender task sleeps for a period of time Tsender and when it wakes up it sends a message to the queue containing the string “Time is XYZ” where XYZ is current time in system ticks. If the queue is full, the sending operation fails and a counter counting total number of blocked messages is incremented. Upon successful sending, a counter counting total number of transmitted messages is incremented. The sender task is then blocked again.

The receiver task sleeps for another period of time Treceiver and then wakes up and checks for any received message in the queue. If there is a message in the queue, it reads it, increments total number of received messages and sleeps again. If there is no message it sleeps again immediately. Note that receiver reads one message at a time even if there are more than one message in the queue.

The sleep/wake control of the two tasks is performed via two periodic timers one for each task. The callback function for each timer is specified as follows:

Sender Timer Callback Function: When called it releases a dedicated semaphore on which the sender task is waiting/blocked on. The sender task is then unblocked and can send to the queue.

Receiver Timer Callback Function: When called it releases a dedicated semaphore on which the receiver task is waiting/blocked on. The receiver task is then unblocked and performs a read on the queue as described above. When the receiver receives 500 messages, the receiver timer callback function calls the "Init" function that performs the following:

1- Print total number of successfully sent messages and total number of blocked message
2- Reset total number of successfully sent messages, total number of blocked message and received message
3- Clears the queue
4- Reset the sender timer period Tsender to the next value in an array. The array holds the values {100, 140,180, 220, 260, 300} expressing the timer value in msec. When the system starts initially it starts with the value 100. If all values in the array are used, destroy the timers and print a message "Game Over"and stop execution.
5- In all iterations Treceiver is fixed at 200 msec,

Note also part of the Init function (performing steps 3,4,5 above) is also called when the program first runs. So, it is called from the main function as well. You may need to perform steps 1-5 above in two functions. So, this is part of the design.

When your output for all runs is obtained, plot the number of total sent messages as function of sender timer period. Explain the gap between the number of sent and received messages in the period.
Also plot the number of blocked messages as function of Tsender.

Use a queue of size 2, then repeat for a queue of size 20. What happens when queue size increases?

4 Documentation

The documentation needs to use the attached template and should contain the sections:

1- System Design section: illustrating the overall flow of the program and how the tasks communicate and synchronize. You can explain the logic of your program and can copy parts of the code as needed.
2- Results section: The requested graphs and their interpretation
3- References (if you used any)