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.

2- Submission is via the google classroom via any other means is rejected immediately and gets a grade of ZERO.

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 the provided .docx template** and should not exceed 5 pages. Otherformats 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.

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

There are three sender tasks. Two has same priority and one has a higher priority. Each sender task sleeps for a RANDOM 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. These counters should be kept per task. The sender task is then blocked for another random period again. The random period is drawn from a uniform distribution as specified below.

The receiver task sleeps for another FIXED 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 three tasks is performed via three 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 1000 messages, the receiver timer callback function calls the "Reset" function that performs the following:

1- Print the total number of successfully sent messages and the total number of blocked messages
2- Print the statistics per sender task (the high priority and the two lower priority tasks).
3- Reset the total number of successfully sent messages, the total number of blocked messages and received message
4- Clears the queue
5- Configure the values controlling the sender timer period Tsender to the next values in two arrays specifying the lower and upper bound values of the uniformly distributed timer period. The first array holds the values {50, 80, 110, 140, 170, 200} and the second holds the values {150, 200, 250, 300, 350, 400} expressing in msec the timer lower and upper bounds for a uniform distribution. When the system starts initially it starts with the values 50 and 150. If all values in the array are used, destroy the timers and print a message "Game Over"and stop execution.
6- In all iterations Treceiver is fixed at 100 msec,

Note also part of the "Reset" 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 average sender timer period. Explain the gap between the number of sent and received messages in the running period.
- Also plot the number of blocked messages as function of average value of Tsender.
- Show also graphs for the high and low priority sender tasks.

Use a queue of size 3, then repeat for a queue of size 10. What happens when queue size increases?

You need to think of how to generate random numbers efficiently or if you will use C run-time library for the rand function for example.

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)