Assignment 3: RTOS

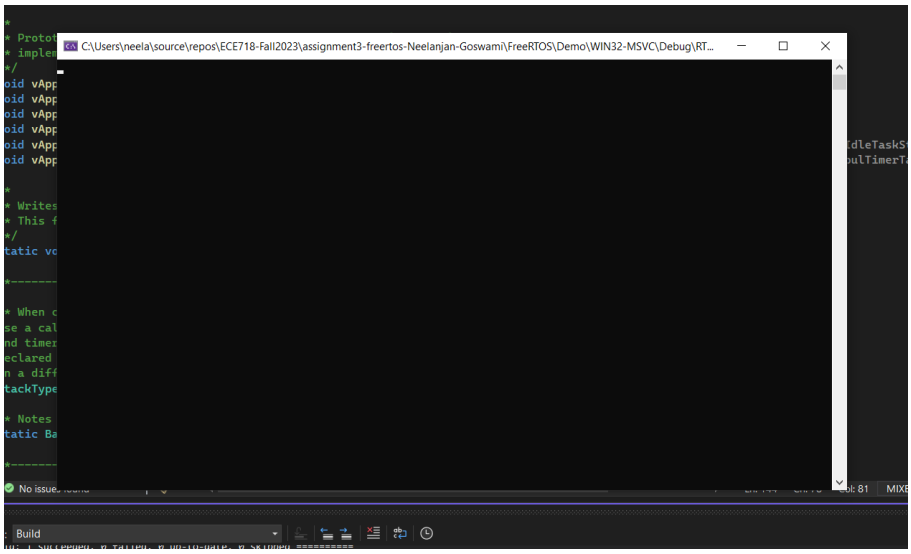
Neelanjan Goswami
400414867

[Entire code file: main.c]

Part 1: A Hello World Example

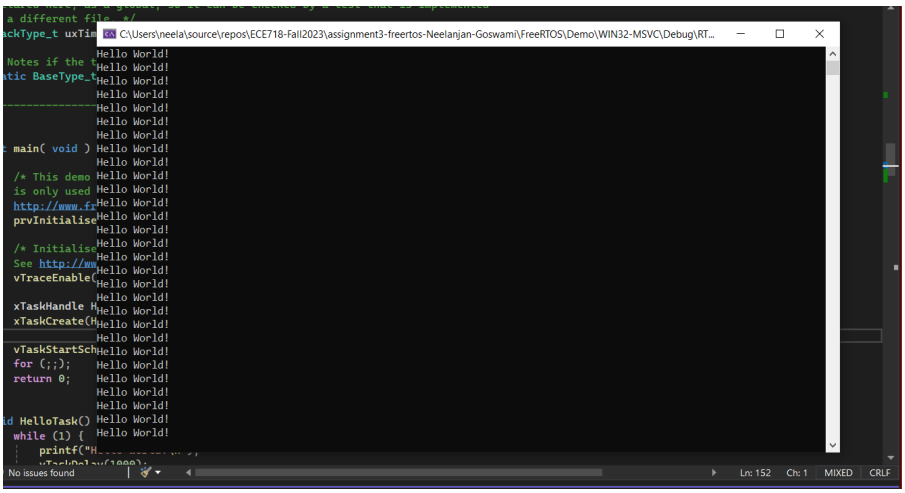
[Code file: FreeRTOS-Part1.c]

Q1) When you execute the project, What do you get? What is the reason for that?



Answer:

Upon selecting the play button on the panel, the project begins to assemble and run. The lack of a task running results in a blank Terminal opening.



Answer:

Following the addition of the "Hello world" job to the "main.c" file. "Hello World!" is shown on the terminal once per second. indicating that there is an endless loop in the task.

Q2) Figure out what is the meaning of the parameters passed to the xTaskCreate function

xTaskCreate(HelloTask, "HelloTask", configMINIMAL_STACK_SIZE, NULL, 1, &HT);

Answer:

1) **HelloTask**: This is the name of the task function that will be executed when the task runs. In your example, it refers to the HelloTask function.

2) **"HelloTask"**: This is a human-readable name given to the task. It is mainly used for debugging and identification purposes.

3) **configMINIMAL_STACK_SIZE**: This parameter specifies the size of the task's stack, which is the memory allocated for the task to use during its execution. configMINIMAL_STACK_SIZE is a constant defined in your configuration, likely representing the minimum stack size required for a task.

4) **NULL**: This is the parameter that can be used to pass any data to the task when it starts. In your example, no additional data is passed, so it's set to NULL.

5) **1**: This parameter is the priority of the task. Tasks with higher priority values will run before tasks with lower priority values. In your example, the priority is set to 1.

6) **&HT**: This is the handle to the task. After calling xTaskCreate, the handle will point to the created task. In your example, &HT is the address of the xTaskHandle variable HT, which will be filled with the task handle after the task is created.

Part 2: Multiple Tasks

[Code file: FreeRTOS-Part2.c]

1) You are required to create two tasks with the following parameters (hint look at xTaskCreate() in the API):

Task1 name="Task1"Task2 name="Task2"
Task1 stack size = 1000
Task2 stack size = 100
Task1 priority = 3
Task2 priority = 1

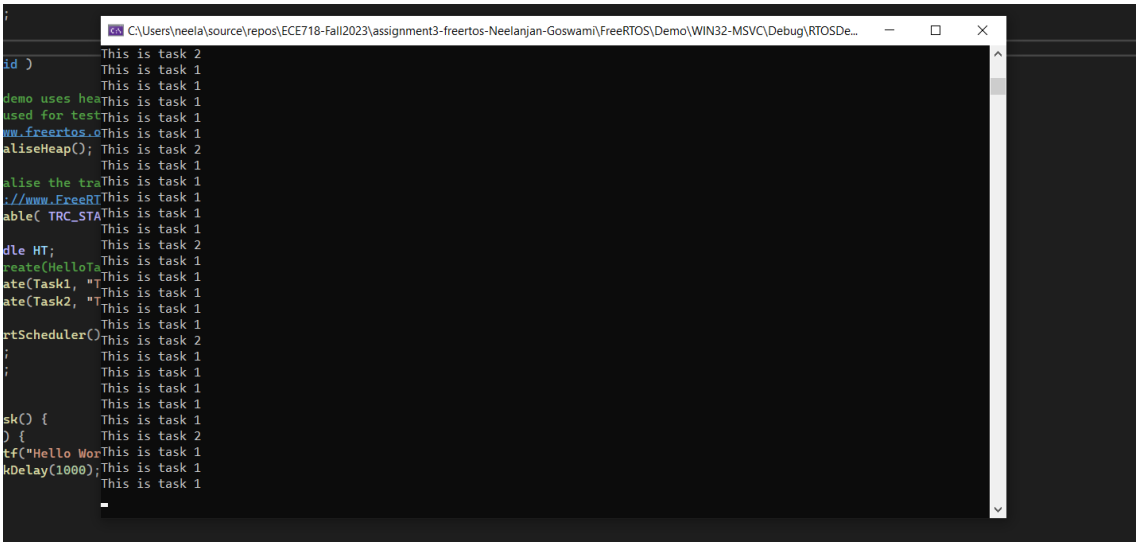
2) Create task functions containing the following functionality

- Task1 should print out "This is task 1" every 100 milliseconds (hint use fflush(stdout) after printf())
- Task2 should print out "This is task 2" every 500 milliseconds

Q3) Provide

1) a screenshot of the execution in a report and

2) the file containing the source code of your solution called FreeRTOS-Part2.
Answer:



Part 3: Task Priorities

[Code file: FreeRTOS-Part3.c]

Q4) "communicationtask" must send a simulated data packet every 200ms but is often blocked by matrixtask, fix this problem without changing the functionality in the tasks.

Answer:
Set the priority of "communicationtask" higher than "matrixtask" such that it is not blocked.
Priorities set for simulation:

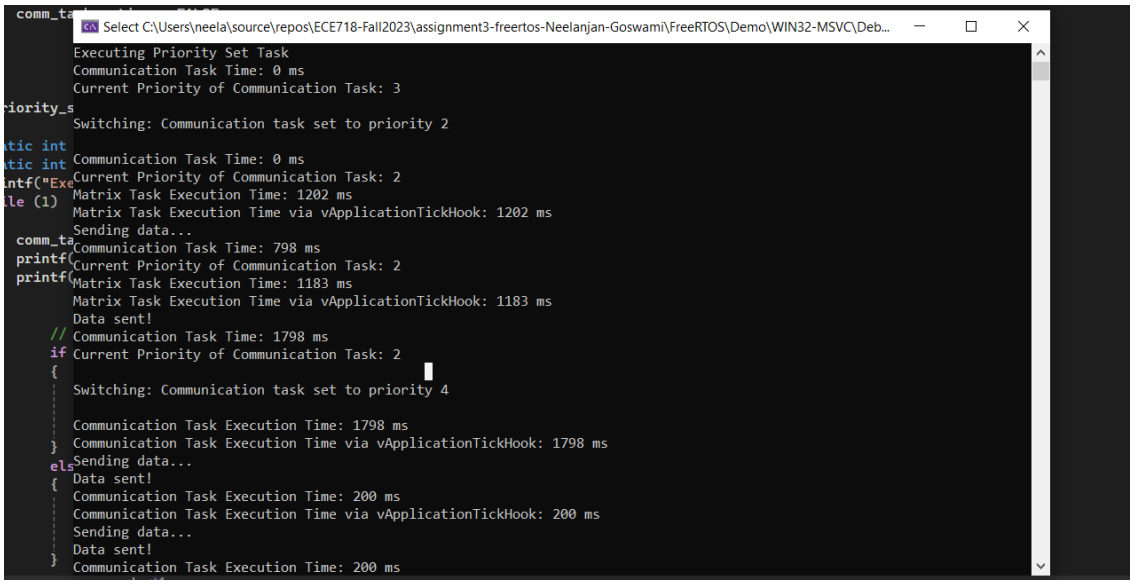
- matrixtask – 3
- communicationtask – 1

Create a new task "prioritysettask" which:

- Sets the priority of "communicationtask" to 4 in case its execution time is more than 1000 milliseconds (Hint: look at vApplicationTickHook() to measure it)
- Sets the priority of "communicationtask" to 2 in case its execution time is less than 200 milliseconds (Hint: look at vApplicationTickHook() to measure it)

Q5) Provide a screenshot of the execution and answer the following questions in a report:

Answer:



- **Why is "matrixtask" using most of the CPU utilization?**

Answer:
Since it requires more computing power than the other tasks, the matrix task takes longer to complete. There is an O(n4) complexity since each entry in the matrix stores the sum of (row*col).

- **Why must the priority of "communicationtask" increase in order for it to work properly**

Answer:
Due to its greater priority, the matrix task is preventing the communication task from executing. It also takes time to complete since it is a computationally demanding process. The communication task has to have a higher priority in order to preempt the matrix task and function correctly.

- **What happens to the completion time of "matrixtask" when the priority of "communicationtask" is increased?**

Answer:
As predicted, there doesn't seem to be much of a difference, considering how little computing power a communication job requires. Whether matrixtask is prioritized higher or lower than communicationstask has very little effect on how long it takes to execute.

- **How many seconds is the period of "matrixtask"? (Hint: look at vApplicationTickHook() to measure it)**

Answer:
Time period of matrix task approximately: 1.1 seconds (1143 microseconds)

Part 4: A periodic Task

[Code file: FreeRTOS-Part4.c]

1) Implement the task "matrixtask" from Part 3.
2) Add a software timer in main() to trigger a software interrupt every 5 seconds. (Documentation found Here.)
3) Define a Timer callback function outside main() with the following functionality:
4) Create an aperiodic task using the following functionality:

Q6. The following questions should be solved with programming and the questions should be answered in a report:

- **Is the system fast enough to handle all aperiodic tasks? Why?**

Answer:
No, the system cannot handle every aperiodic job due to the matrix task's greater priority and constant pre-emption of the aperiodic task.

- **If not, solve this problem without alter the functionality of any task**

Answer:
Decrease the matrix task's priority or increase the aperiodic task's priority. To put it briefly, aperiodic task needs to be given greater priority to avoid being pre-empted by another task.

- **What is the response time of the aperiodic task?**

Answer:
Time period of aperiodic task: approximately 1.2 seconds (1211 microseconds)
Response time: 0 seconds

- **Provide a screenshot of the running system**

Answer: