Real Time Task Admittance and Scheduling

Program:

Course Code: CSE 345

Course Name: Real Time and Embedded Systems Design

Examination Committee

Dr. Omar Alkelany

Ain Shams University
Faculty of Engineering
International Credit Hours Engineering Programs (I-CHEP)

Spring Semester – 2020

Student Personal Information for Group Work

Student Names:	Student Codes:
Engy Samy Salah	16P3004
Gina Emil Attia	16P3022
Mayar Wessam Hassan	16P3008
Rowan Hazem Wagieh	16P3023
Yara Hossam Mohamed	16P3002

Plagiarism Statement

I certify that this assignment / report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they are books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment / report has not been previously been submitted for assessment for another course. I certify that I have not copied in part or whole or otherwise plagiarized the work of other students and / or persons.

Signature/Student Name: **Engy Samy, Gina Emil, Mayar Wessam, Rowan Hazem, Yara Hossam**

Date: 31/05/2020

Submission Contents

01: System Design Part

02: Technical Details Part

01

First Topic

System Design Part

1.1) Project specification and description:

In this project we are designing a task scheduler in real time. This scheduler has many properties in creating the tasks and running them in a preemption way.

- First a random integer number is generated between the range of 2 and the maximum number of tasks that can be generated.
- Then, the program generates tasks that are equal to that random generated number.
- The scheduler operates in two modes: a safe mode and a no guarantee mode.
- Each task created has a few properties:
  - A time at which the task arrives to the scheduler (Ta)
  - A time of how long is the period of the task (Tp)
  - A time of how much time will the task need to finish its computation (Tc)

After the tasks have been created, the program calculates each task's CPU utilization and makes a schedulability check for each task according to that CPU utilization and also calculates each task's rate.

The program then sorts the tasks giving each one of them a priority that is set according to their periods and rates. After the tasks are sorted, the program starts acting like a scheduler and swaps between the tasks so every task runs according to its period, priority and arrival times.

While the tasks are running, a random integer number is generated from an interval of the number of tasks created. This number defines the number of the task that will be deleted randomly during the runtime. After this task is deleted the scheduler continues swapping between the remaining tasks after re-sorting them according to their periods and priorities once again.

Conclusion:

The program acts like a scheduler that keeps swapping between the tasks created. Tasks are being swapped according to their priorities, so tasks with the highest priority number (equals to 5

for example if the total number of tasks is 5) should be running first. And while the program is running, a random task can be deleted, then the scheduler continues its work normally.

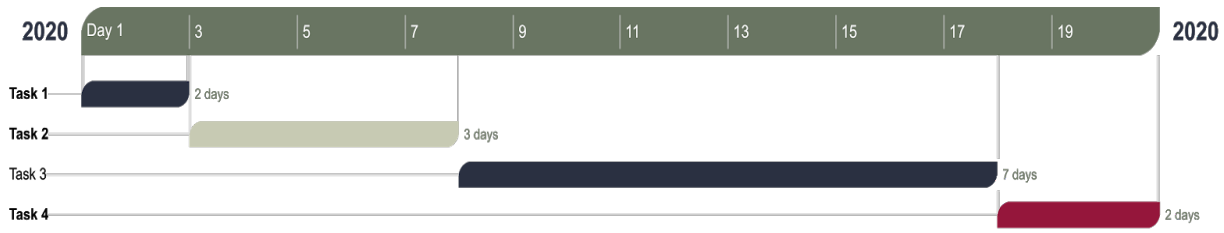
1.2) Team members responsibilities:

We all have worked together in everything. Everytime we decide to work on the project we make sure that we are all free and ready to work, then we open a meeting on "zoom" application and the one who is responsible for writing the code (Rowan or Engy) shares the screen so that all the team members can see hence, we can brainstorm the ideas and share it together.

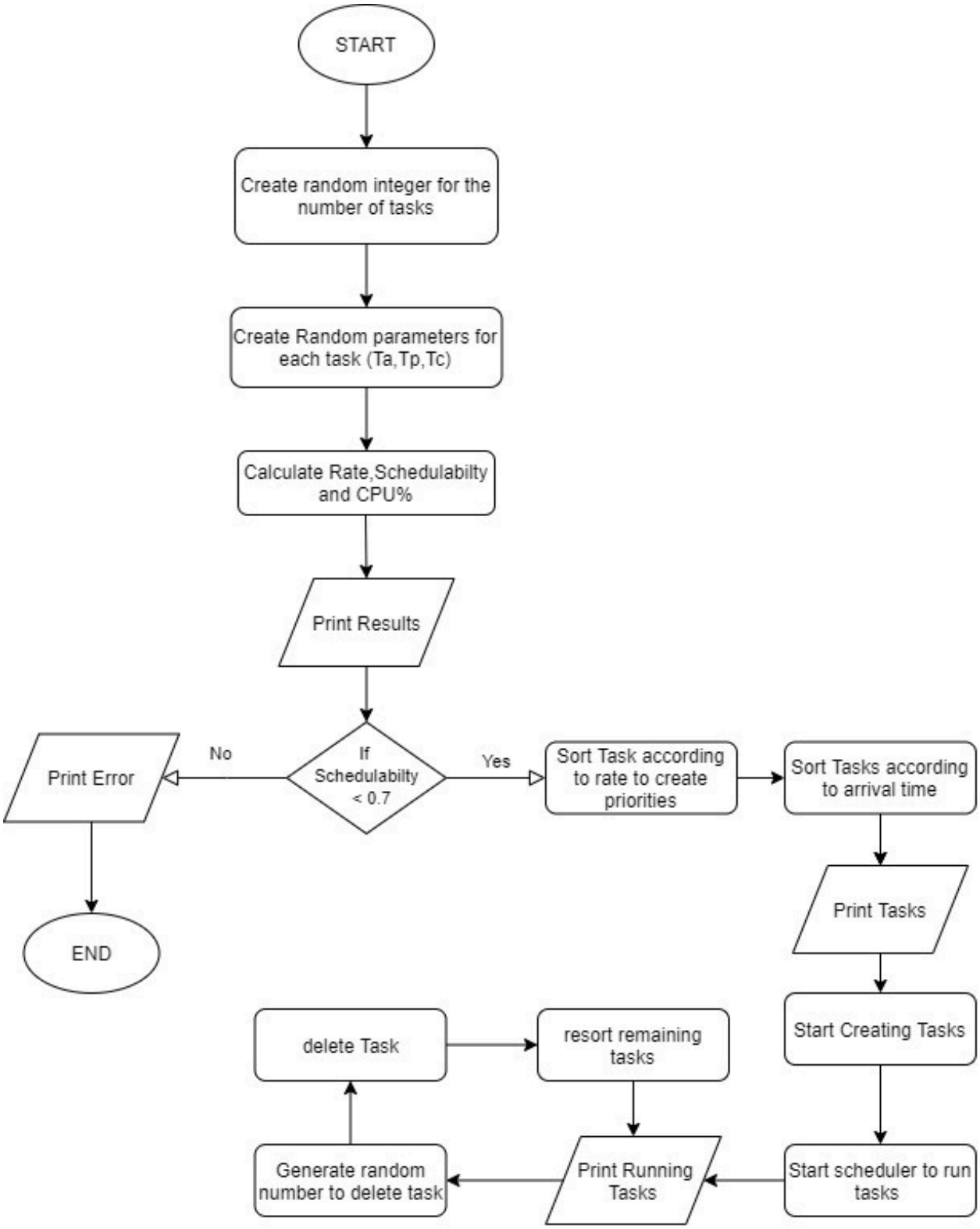
We found that this was the easiest way to communicate in the current situation we are all in. We also have worked together a lot, so we always try to make sure that every and each person in the team understands what the project wants and how we implemented it while writing the code.

1.3) Plan of timeline

- **Task (1):** Design a task admittance function with limit of *n* periodic randomly set and concurrent tasks. Each one must have three parameters which are: Ta , Tp , Tc for arrival , period (safe mode or no grantee mode)and computation time respectively.
- **Task (2):** Calculate total CPU utilization and perform a schedulability test
- **Task (3):** Set priorities for the tasks based on their rates and perform a sorting algorithm to order it.
- **Task (4):** Allow more dynamics by using dynamic data structure instead of the static ones so that we could delete randomly some of the tasks and reorder all the priorities of the tasks and resume scheduling.



1.4) Flow Chart



02

Second Topic

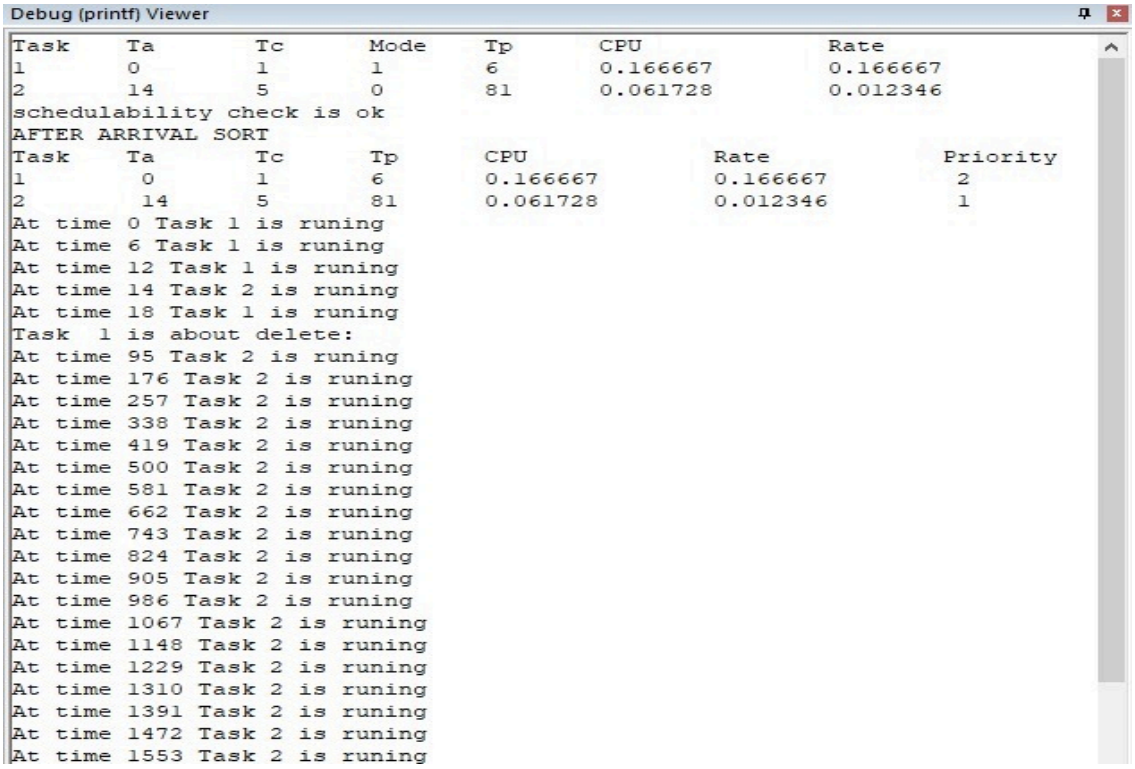
Technical Details Part

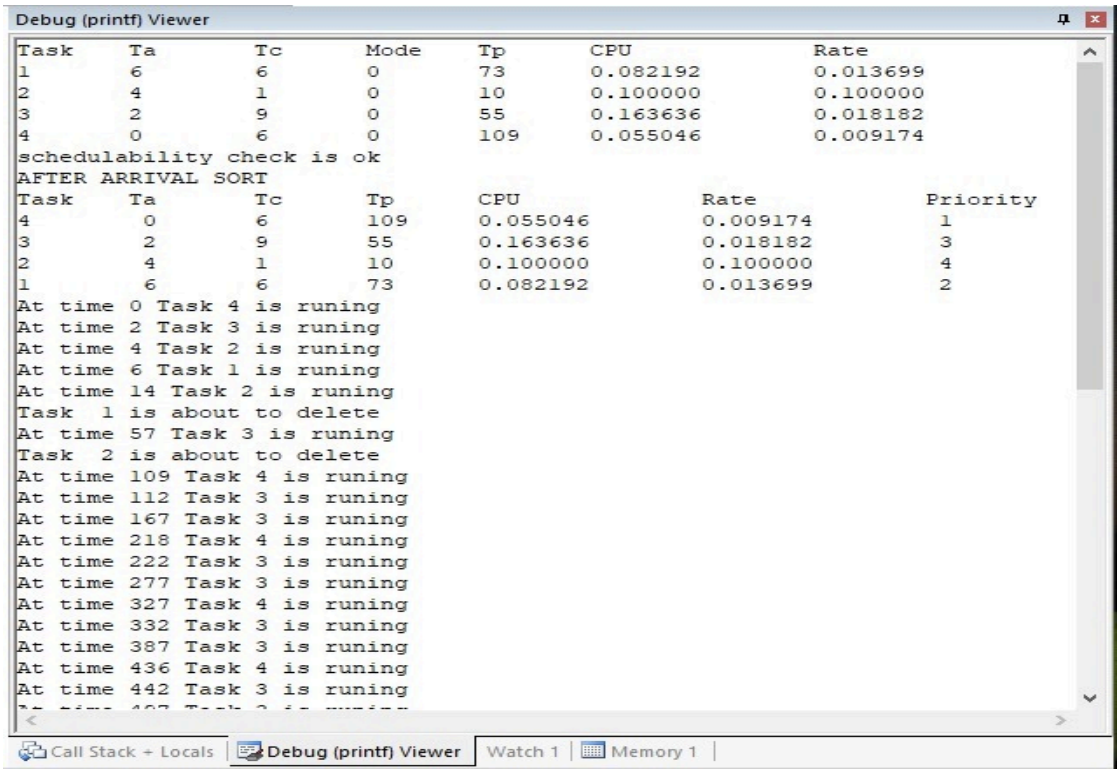
2.1) Design details

How to test: to be able to test this program we can assume four test cases which are:

- **Test Case (1):**

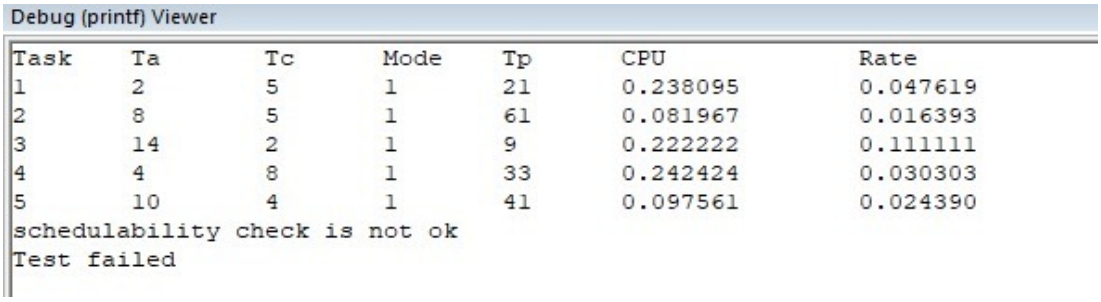
  By adjusting the value passed to the "Srand" function to have a different random number of tasks and making sure that calculated CPU utilization and schedulability test is right.

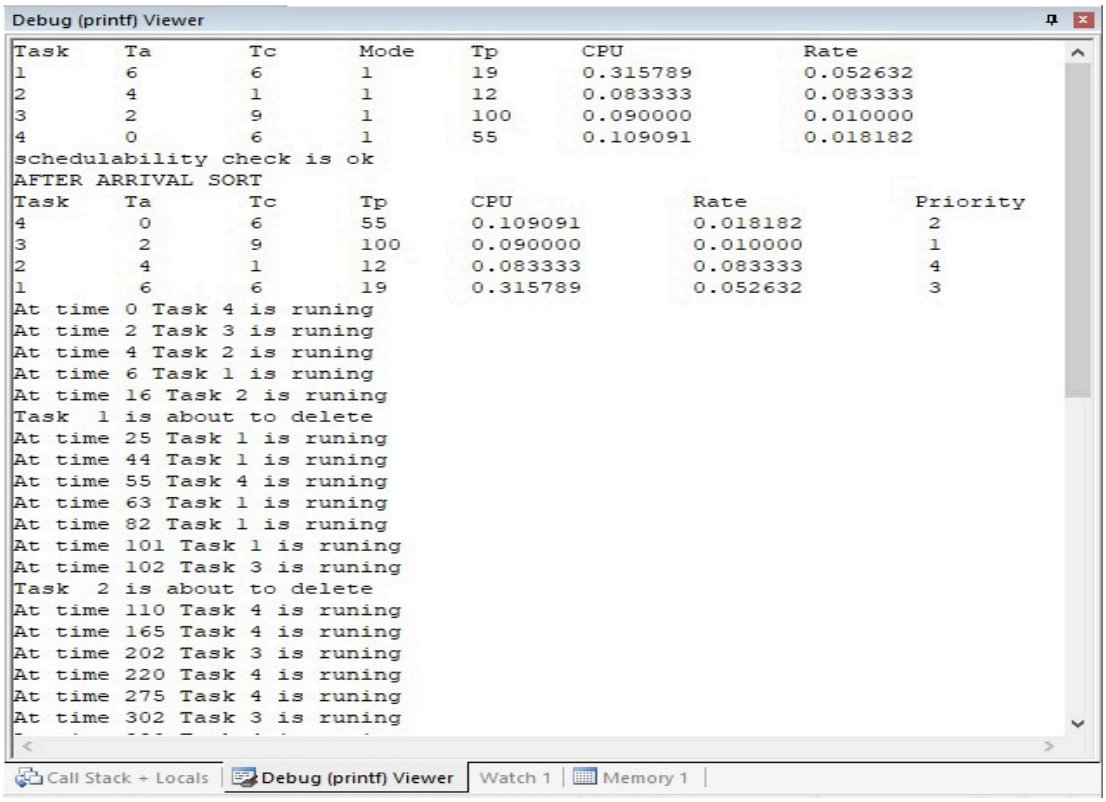




- **Test Case (2):**

Making sure to have the CPU utilization < 0.7 to get the schedulability check "OK" and another time CPU utilization > 0.7 to get the schedulability check "not OK"





- **Test Case (3):**

  Having task with "Safe Mode" with Tp(i) set from 3xTc(i) to maximum_period_multiplier x Tc(i) (assuming it has a value of zero)

- **Test Case (4):**

  Having task with "No Guarantee Mode" with Tp(i) set from 3xTc(i) to 10x Tc(i) (assuming it has a value of 1)

2.2) Code listing

Insertion Sort: This function uses insertion sort algorithm to sort the tasks in the array based on their periodic time. The tasks are sorted from ascending to descending order. The first task after sorting has highest priority and last task has lowest priority.

Arrival Sort: This function uses insertion sort algorithm to sort the tasks in the array based on their arrival time. The tasks are sorted from ascending to descending order. We sort tasks according to their arrival time to create them in the right time after starting the schedule in time zero.

Task: Firstly, this function starts by adding random number from 2 to N-1 where N is the maximum number of tasks that can be created and defined globally using #define and the random value is added to n (which is considered as number of tasks created).Then starting a for loop from zero to number of tasks created inside this loop we initialize the arrival time,

computation time, periodic time, CPU, and rate for each task. We add a random number from zero to 1 if it is equal zero then the periodic time works in a safe mood, if it is one then there will be no guarantee mode. When the for loop ends, we make the schedulability check by adding the CPU time of all tasks.

vTaskMaster: This task is created in main right before calling vtaskschedular, the responsibility of this task is to create all the tasks according to their arrival times. Initially, we use vtaskdelayuntil with a time difference between the arrival time of tasks created then adding a number to task which has to be created then create task which has to turn and pass all the values of task to it and after all tasks created we delete master task.

vPeriodicTask: In the beginning, we print the time in which task starts running and task number. We initialize random numbers from zero to 20 to delete tasks randomly, if this random number is less than 5 then a task will be deleted. Inside the if condition that delete tasks we make a loop to check if the task has been deleted before or not, as we have an array containing the number of tasks deleted and check if the random number of the task to be deleted is in the array or not. If the rand didn't come before then we will suspend all and print the number of task that is going to be deleted and delete it and call function of insertion sort to re arrange according to periodic time to re-calculate the priority after deleting task and then add number of deleted task to the array. After all, we re-calculate the priority and call arrival sort again and resume all, then call vtaskdelay with the periodic time of the running task.

Main: Initially, in the main we call function srand with any constant number, then call function task then insertion sort. After calculating all the variables needed for tasks and arranging them according to periodic time, we start calculating priority and if two tasks have the same periodic, they will have the same priority. We call arrival sort after assigning priority to arrange based on their arrival time. We have an array which is called arr which contains the difference between the arrival time of tasks by order. Then create a handler for each task and add it to the array of handlers. Finally, we create a master task with higher priority and call start scheduler.

2.3) Lessons learned

- We learned that if there was an error in a function while debugging, we have to open the config.h file and see its value and change it according to what we need.
- When all of the tasks are created but only 2 are running regardless of the priorities of the other tasks, we learned that we have to change the heap size that is given to the xTaskCreate for each task so that they are able to exchange and to make sure that no task is idle.
- We learned how to use the simulator, see the debug (printf) viewer screen, and also how to watch specific variables and their values while debugging, as it was very helpful to know how the system is working and to catch the errors easily.
- We learned that we can't define or initialize anything after using the "srand" or "rand" functions as it will always give an error.

- Finally, we learned how to use and deal with keil more than before, we also learned how to do our own scheduler and deal with the tasks by suspending & resuming them, by setting their priorities and moreover.

2.4) Problems faced

We have faced a lot of problems while working on that project;

- We had an issue to run the simulator instead of working on the tivac board as the previous project.
- We tried to use the function printf() but somehow it wasn't working, so we had to take one of the doctor's examples to work on it.
- When we tried to use the time() in srand() function so that every time we run the code, the value of the random number could change, it didn't work as it's not implemented in keil.
- vTaskDelete, vTaskDelayUntil and vTaskPrioritySet at first when we tried to run the code they gave us an error we have never seen before, so we had to search to find the problem, and we found that we have to set their values by 1 in the config.h file.
- After creating the tasks, we found that only 2 tasks were running and the rest were idle regardless of the priorities of the other tasks.