EDF Implementation and its application

Earliest deadline first (EDF) scheduler has a list ordered in a descending fashion from the shortest deadline to the longest deadline using a dynamic priority scheduling method to set the highest priority for the task with the shortest deadline.

Application: implementation for 6 tasks using EDF scheduler and verifying the method using 3 different methods: Analytical method, offline simulation method and Run-Time analysis. The project is using the demo application (ARM7_LPC2129_Keil_RVDS) and applying all the changes to it in Keil and all the simulation is done using Keil logic analyzer.

Tasks:

1- Task 1: ""Button_1_Monitor"", {Periodicity: 50, Deadline: 50}
This task will monitor rising and falling edge on button 1 and send this event to the consumer task. (Note: The rising and falling edges are treated as separate events, hence they have separate strings)

2- Task 2: ""Button_2_Monitor"", {Periodicity: 50, Deadline: 50}
This task will monitor rising and falling edge on button 2 and send this event to the consumer task. (Note: The rising and falling edges are treated as separate events, hence they have separate strings)

3- Task 3: ""Periodic_Transmitter"", {Periodicity: 100, Deadline: 100}
This task will send periodic string every 100ms to the consumer task

4- Task 4: ""Uart_Receiver"", {Periodicity: 20, Deadline: 20}
This is the consumer task which will write on UART any received string from other tasks

5- Task 5: ""Load_1_Simulation"", {Periodicity: 10, Deadline: 10}, Execution time: 5ms

6- Task 6: ""Load_2_Simulation"", {Periodicity: 100, Deadline: 100}, Execution time: 12ms

Conclusion:

1- All tasks were implemented successfully using the changes in the Thesis and adding some changes to the normal rate-monotonic fixed priority in FreeRTOS to use the scheduling method EDF scheduler.

2- CPU usage is 62% using the three mentioned methods which verifies the schedulability of the system offline, online and using mathematical analysis.

3- The results are as expected using the EDF method

4- The Idle task is dynamically changing its deadline as it gets updated every tick

Very Important Notes:

1- The first four tasks consume almost no time in milliseconds that's why their execution time is almost zero in the mathematical method both in URM and Time Demand analysis

2- Only Load_1_Simulation and Load_2_Simulation consume the processor usage although all the other tasks are scheduled in their time using the EDF schedular (refer to Videos folder to get clear explanation)

3- The highest task deadline should be mentioned if anyone is using the EDF scheduling method because the Idle task gets scheduled before any other task in vTaskStartSchedualr() function. Thus, there is no method to detect the highest deadline in system run time before scheduling Idle task for the first time. That's why there is a configuration macro called configHighest_Deadline to put the highest deadline of the system tasks before the system start and after scheduling the Idle task and all other tasks, all the system tasks gets updated automatically and the Idle task deadline also gets updated on every tick using EDF method to set their priority in the ready list based on their deadline.

4- Without the "configHighest_Deadline" the system will not work properly for tasks with 100 ms deadline or higher as the thesis assume that 100 ms is the idle task periodicity and deadline and farthest one. This cannot work with tasks in application or any users' tasks with periodicity equals or higher than 100 ms (for clearer explanation see Videos folder)

5- In run-time analysis, The CPU load without tasks 5 and 6 is almost under 1% because tasks are executed in just very few microseconds.

6- All run-time analysis is using Trace Hooks and GPIO pins methods (as shown in Videos Folder).

7- Execution times of tasks 5 and 6 are calculated as follows: if n times in for loop = 10000 it represents 1.52 ms using the tick counter and tick hook. Thus, 5 ms = 32895 in for loop and 12 ms = 78948 in for loop