

PROJECT SPECIFICATION

Implementing EDF Scheduler

Design EDF scheduler

CRITERIA	MEETS SPECIFICATIONS
Read a thesis and implement the required changes	<p>The following thesis discuss how to implement an EDF scheduler using FreeRTOS.</p> <p>1. Download the following thesis: "Implementation and Test of EDF and LLREFSchedulers in FreeRTOS".</p> <p>2. Read chapter 2 : "FreeRTOS Task Scheduling". This is an important chapter to build a profound base before starting the project.</p> <p>3. Read chapter 3 : "EDF Scheduler". This chapter is the main chapter you will use to implement the EDF scheduler using FreeRTOS.</p> <p>4. Watch the final project explanation video to further understand the thesis and the FreeRTOS dependencies.</p> <p>5. Implement the changes mentioned in chapter 3.2.2 : "Implementation in FreeRTOS". The changes will be implemented in tasks.c source file only.</p> <p>"For this criteria please deliver the following:</p> <p>Tasks.c source file with changes implemented from chapter 3.2.2 from the thesis"</p>
Implement the missing changes from the thesis	<p>Inorder for the EDF scheduler to work correctly, you still need to implement some changes that are not mentioned in the thesis:</p> <p>"1. In the ""prvIdleTask"" function:</p> <p>Modify the idle task to keep it always the fareset deadline"</p> <p>"2. In the ""xTaskIncrementTick"" function:</p> <p>In every tick increment, calculate the new task deadline and insert it in the correct position in the EDF ready list"</p> <p>"3. In the ""xTaskIncrementTick"" function:</p> <p>Make sure that as soon as a new task is available in the EDF ready list, a context switching should take place. Modify preemption way as any task with sooner deadline must preempt task with larger deadline instead of priority"</p> <p>"For this criteria please deliver the following:</p> <p>Tasks.c source file only with the changes mentioned above implemented"</p>

CRITERIA	MEETS SPECIFICATIONS
Implement 4 tasks using EDF scheduler	<p>Inorder to verify the EDF scheduler, you need to implement an application:</p> <p>"1. Create 4 tasks with the following criteria:</p> <p>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 failling edges are treated as separate events, hence they have separate strings)</p> <p>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 failling edges are treated as separate events, hence they have separate strings)</p> <p>Task 3: ""Periodic_Transmitter"", {Periodicity: 100, Deadline: 100} This task will send preiodic string every 100ms to the consumer task</p> <p>Task 4: ""Uart_Receiver"", {Periodicity: 20, Deadline: 20} This is the consumer task which will write on UART any received string from other tasks "</p> <p>"2. Add a 5th and 6th task to simulate a heavier load:</p> <p>Task 5: ""Load_1_Simulation"", {Periodicity: 10, Deadline: 10}, Execution time: 5ms Task 6: ""Load_2_Simulation"", {Periodicity: 100, Deadline: 100}, Execution time: 12ms</p> <p>These two tasks shall be implemented as en empty loop that loops X times. You shall determine the X times to achieve the required execution time mentioned above. (Hint: In run-time use GPIOs and logic analyzer to determine the execution time)"</p> <p>1. Implement all the tasks mentioned above in the same main.c source file. "For this criteria please deliver the following:</p> <p>2. A (maximum 3min) video showing the system working in run-time using Keil simulation. in this video you shall show how the system is working in run-time according to the requirements.</p> <p>3. Deliver main.c, task.c and freertosconfig.h"</p>

CRITERIA	MEETS SPECIFICATIONS
Verifying the system implementation	<p>Now you should verify your system implementation with the EDF scheduler using the following methods:</p> <p>"1. Using analytical methods calculate the following for the given set of tasks:</p> <ul style="list-style-type: none">• Calculate the system hyperperiod• Calculate the CPU load• Check system schedulability using URM and time demand analysis techniques (Assuming the given set of tasks are scheduled using a fixed priority rate -monotonic scheduler) <p>Note: For all the tasks you should calculate the execution time from the actual implemented tasks using GPIOs and the logic analyzer"</p> <p>"2. Using Simso offline simulator, simulate the given set of tasks assuming:</p> <ul style="list-style-type: none">• Fixed priority rate monotonic scheduler " <p>"3. Using Keil simulator in run-time and the given set of tasks:</p> <ul style="list-style-type: none">• Calculate the CPU usage time using timer 1 and trace macros <ul style="list-style-type: none">• Using trace macros and GPIOs, plot the execution of all tasks, tick, and the idle task on the logic analyzer" <p>"For this criteria please deliver the following:</p> <ul style="list-style-type: none">• A PDF report that includes screenshots from the above verification methods and their results. Your report shall also include a comment on the results of these analysis (Ex: Are the results as expected ?, Does the results indicate a successful implementation ?, etc ...). <ul style="list-style-type: none">• Deliver main.c, task.c and freertosconfig.h"

Suggestions to Make Your Project Stand Out!

"When implementing the missing changes from the thesis
<br data-md>
Modify the function to read the tasks status from the EDF ready list in the ""uxTaskGetSystemState"" function."
"When you are Verifying the system implementation
<br data-md>
Use FreeRTOS run-time stats function ""uxTaskGetSystemState"", print the stats summary of all tasks."