Project 2

Implementing EDF Scheduler

Developed By :

Eslam Mohamed Hashem Fayad

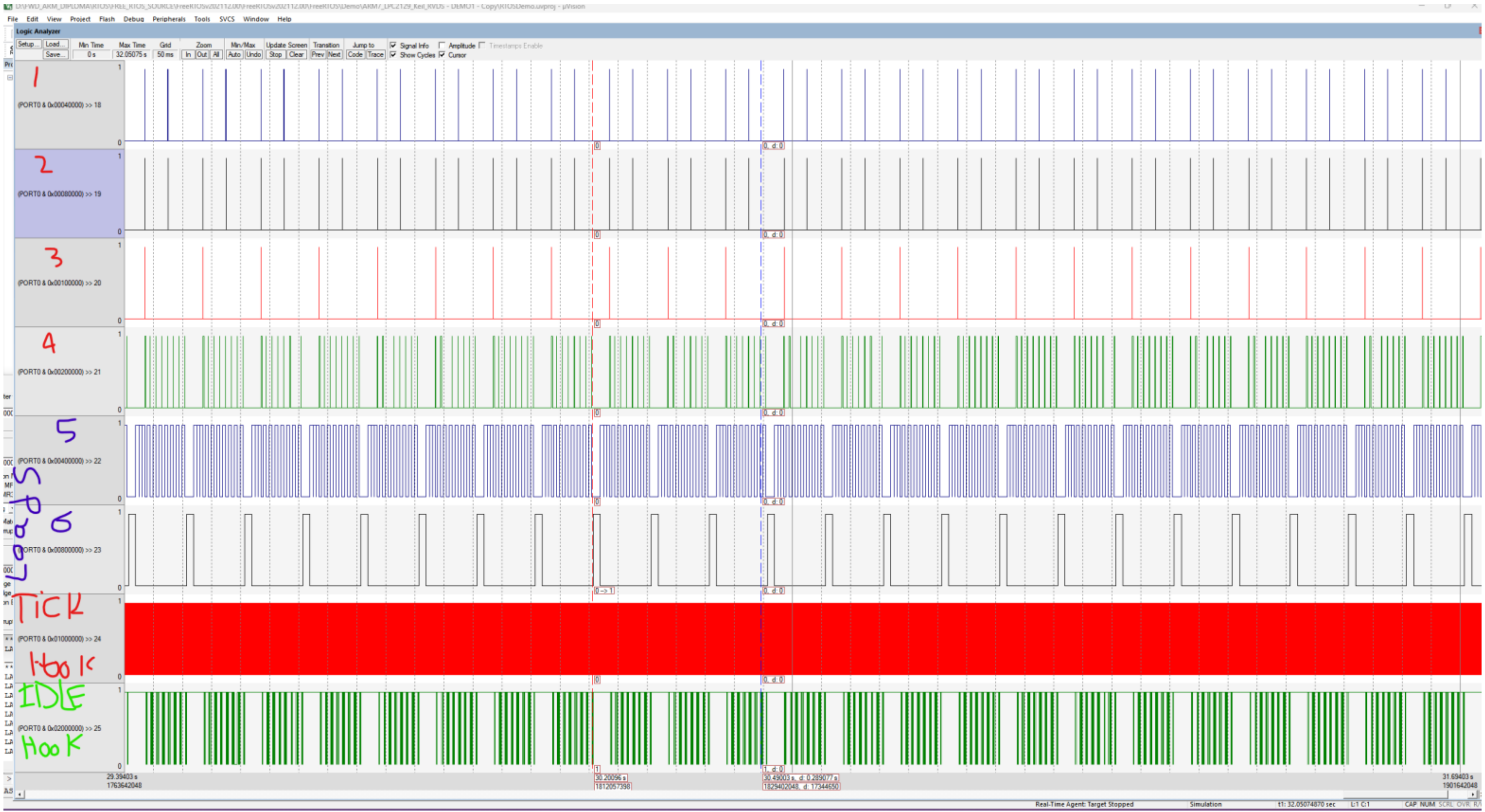
PROJECT SPECIFICATION

CRITERIA	MEETS SPECIFICATIONS
Read a thesis and implement the required changes	The following thesis discuss how to implement an EDF scheduler using FreeRTOS. 1- Download the following thesis: "Implementation and Test of EDF and LLREFSchedulers in FreeRTOS". 2- Read chapter 2 : "FreeRTOS Task Scheduling". This is an important chapter to build a profound base before starting the project. 3- Read chapter 3 : "EDF Scheduler". This chapter is the main chapter you will use to implement the EDF scheduler using FreeRTOS. 4- Watch the final project explanation video to further understand the thesis and the FreeRTOS dependencies. 5- Implement the changes mentioned in chapter 3.2.2 : "Implementation in FreeRTOS". The changes will be implemented in tasks.c source file only. "For this criteria please deliver the following: Tasks.c source file with changes implemented from chapter 3.2.2 from the thesis"
Implement the missing changes from the thesis	Inorder for the EDF scheduler to work correctly, you still need to implement some changes that are not mentioned in the thesis: "1. In the ""prvIdleTask"" function: Modify the idle task to keep it always the farest deadline" "2. In the ""xTaskIncrementTick"" function: In every tick increment, calculate the new task deadline and insert it in the correct position in the EDF ready list" "3. In the ""xTaskIncrementTick"" function: 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" "For this criteria please deliver the following: Tasks.c source file only with the changes mentioned above implemented"
Implement 4 tasks using EDF scheduler	Inorder to verify the EDF scheduler, you need to implement an application: "1. Create 4 tasks with the following criteria: 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) 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) Task 3: ""Periodic_Transmitter"", {Periodicity: 100, Deadline: 100} This task will send preiodic string every 100ms to the consumer task Task 4: ""Uart_Receiver"", {Periodicity: 20, Deadline: 20} This is the consumer task which will write on UART any received string from other tasks " "2. Add a 5th and 6th task to simulate a heavier load: Task 5: ""Load_1_Simulation"", {Periodicity: 10, Deadline: 10}, Execution time: 5ms Task 6: ""Load_2_Simulation"", {Periodicity: 100, Deadline: 100}, Execution time: 12ms 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)" Implement all the tasks mentioned above in the same main.c source file. "For this criteria please deliver the following: 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. Deliver main.c, task.c and freertosconfig.h"
Verifying the system implementation	Now you should verify your system implementation with the EDF scheduler using the following methods: "1. Using analytical methods calculate the following for the given set of tasks:

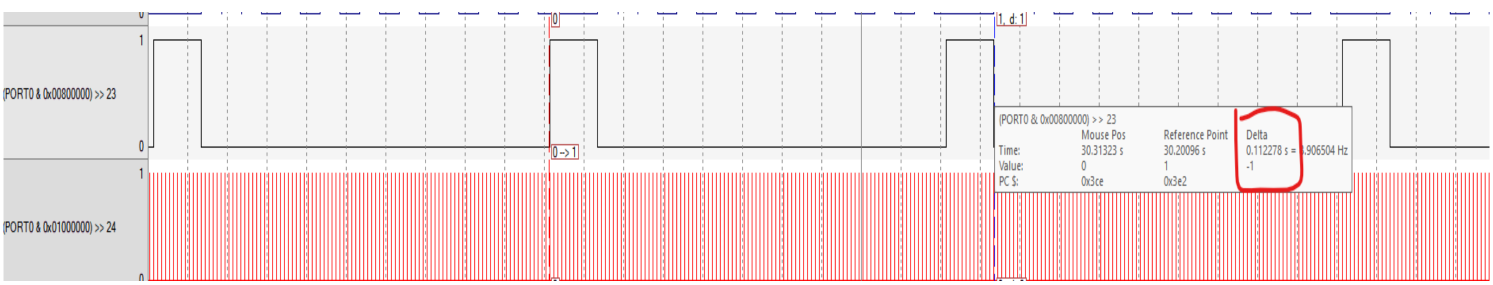
	Calculate the system hyperperiod Calculate the CPU load Check system schedulability using URM and time demand analysis techniques (Assuming the given set of tasks are scheduled using a fixed priority rate -monotonic scheduler) Note: For all the tasks you should calculate the execution time from the actual implemented tasks using GPIOs and the logic analyzer" "2. Using Simso offline simulator, simulate the given set of tasks assuming: Fixed priority rate monotonic scheduler " "3. Using Keil simulator in run-time and the given set of tasks: Calculate the CPU usage time using timer 1 and trace macros Using trace macros and GPIOs, plot the execution of all tasks, tick, and the idle task on the logic analyzer" "For this criteria please deliver the following: 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 ...). Deliver main.c, task.c and freertosconfig.h"

Verifying the system implementation

1- ALL TASKS EXCUTION PLOT USING TRACE MACROS AND GPIOS



2- Calculate the system hyperperiod

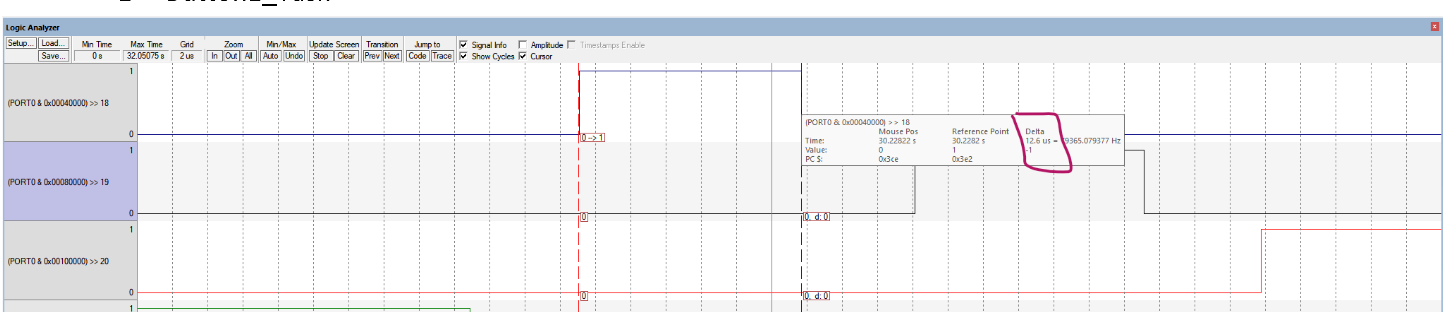


Hyper period Measured from Keil simulator = 100 ms

3- Calculate CPU Load

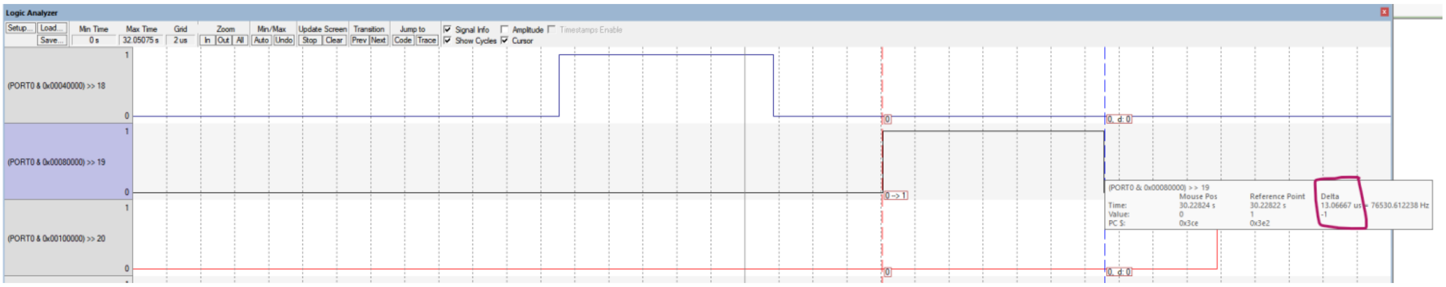
First we will Get Every Task Execution Time

1- Button1_Task



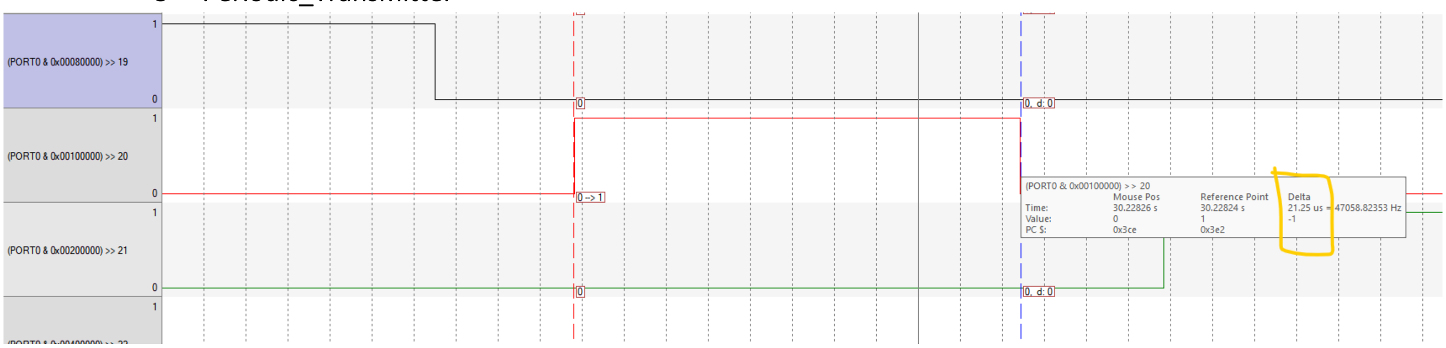
Button1_Task Excution time = 13 us

2- Button2_Task



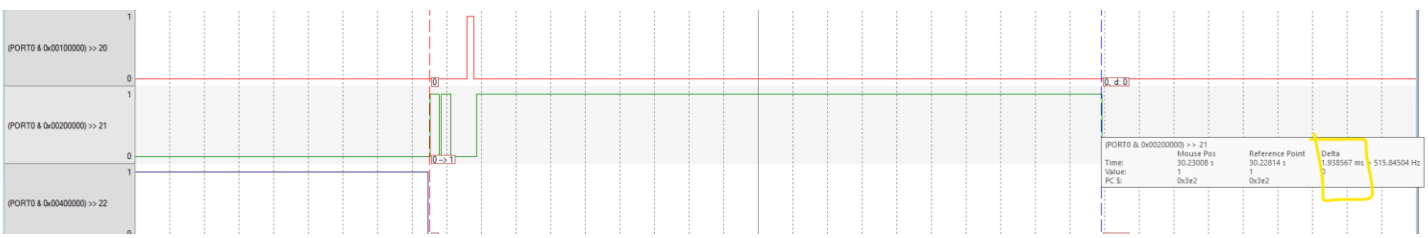
Button2_Task Excution time = 13 us

3- Periodic_Transmitter



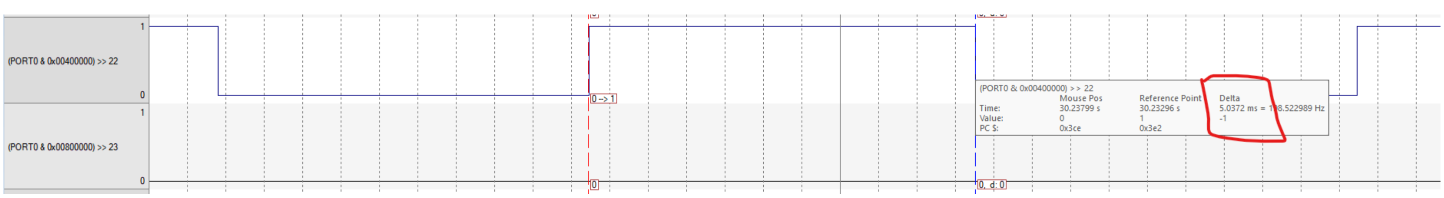
Periodic_Transmitter Excution time = 11.15 us

4- Uart_Receiver



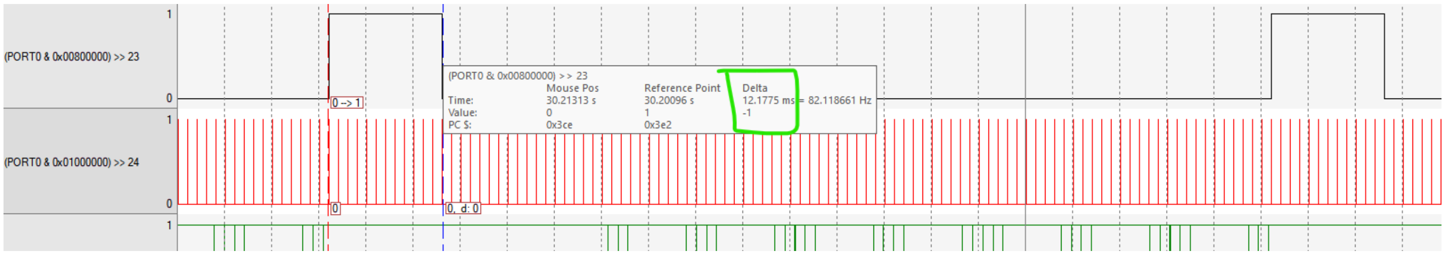
Uart_Receiver Excution time = 2 ms

5- Load_1_Simulation



Load_1_simulation Task Excution time = 5 ms

6- Load_2_Simulation



Load_1_simulation Task Excution time = 12 ms

Summary of Tasks Excution Time

Task Name	Excution Time	Periodicity
Button1_Task	13 us	50
Button2_Task	13 us	50
Periodic_Transmitter	11.15 us	100
Uart_Receiver	2000 us	20
Load_1_Simulation	5000 us	10
Load_2_Simulation	12000us	100

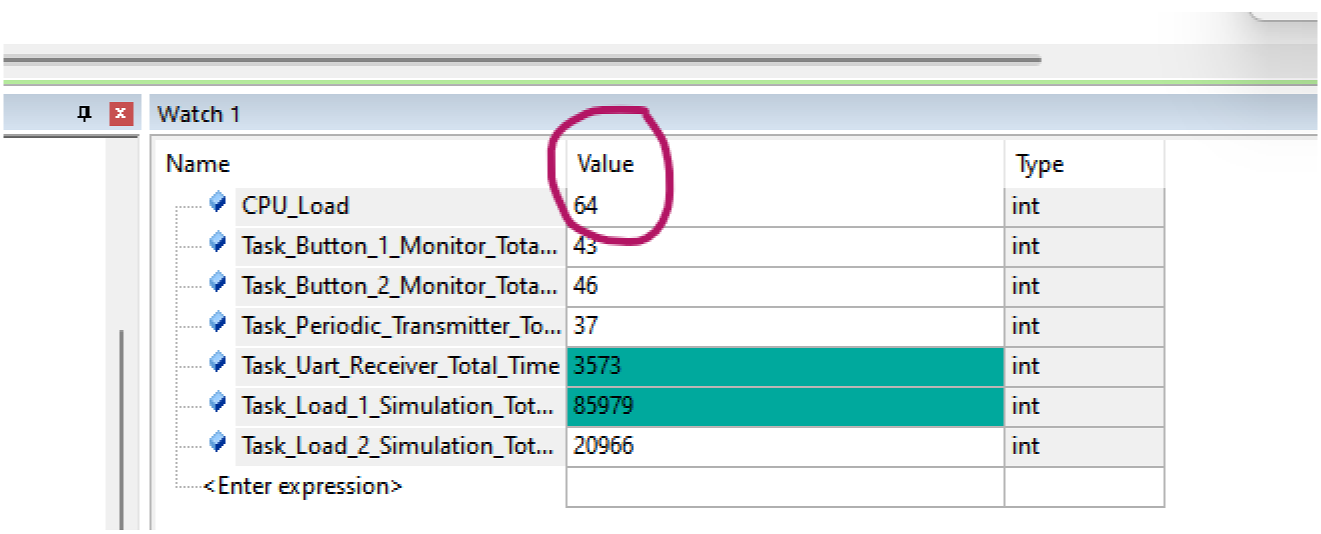
Total Execution Time in Hyper Period

13(100/50)+13(100/50)+11.15(100/100)+2000(100/20)+5000(100/10)+12000(100/100) = 72000us

Total Execution Time = 72ms

- CPU Load = 72/100 = 72 %

Calculate CPU Load using TRACE and GPIO from Keil



- CPU Load = 64 %

- **Include a comment on the results of these analysis (Ex: Are the results as expected ?, Does the results indicate a successful implementation ?, etc ...).**

Yes, the results are as expected.
• In case of manual calculations, the results are calculated in worst case scenario, for example:
the UART task execution time during receiving signals (2ms) is repeated for every 20ms.
• On the other hand, the trace calculations, the results are calculated based on the actual operating system.
Ex.
In worst case:
UART task = 2 * 5 =10 ms.
In actual:
UART task = 2 * 1 =2 ms.
In worst case - In actual = 8 ms.
The difference represents 8 % of CPU load, if added to the actual calculations (64%) it will give (72%).
the same manual calculations (72%).

4- **Schedulability using URM and time demand analysis techniques (Assuming the given set of tasks are scheduled using a fixed priority rate -monotonic scheduler)**

1- **USING URM**

$$U=\sum_{i=1}^{n}\frac{Ci}{Pi} \qquad \textit{should be } \leq n(2^{\frac{1}{n}}-1)$$

With CPU Load is 64% n is number of tasks =6

URM = 6*(2^(1/6) -1) = 73.477 %

So C/p(64%) is less Than URM (73.477 %)

SO we Guaranteed that system is schedulable

Also if we calculate at U= (72%) < URM(73.477 %) system also Guaranteed to be schedulable

2- **USING TIME DEMAND Analysis Technique**

1- time demand analysis for Load_1_Simulation
w (10) = 5000 + 0 = 5000 < 10000
Load_1_Simulation is schedulable.

2- time demand analysis for Task Uart Receiver
w(20) = 5000*2 + 1900 = 11900 < 20000
Task Uart Receiver is schedulable.

3- time demand analysis for Task_Button_1_Monitor
w(10) = 5000*5 + 2000*2+12.6 =29014 < 50000
Task_Button_1_Monitor is schedulable.

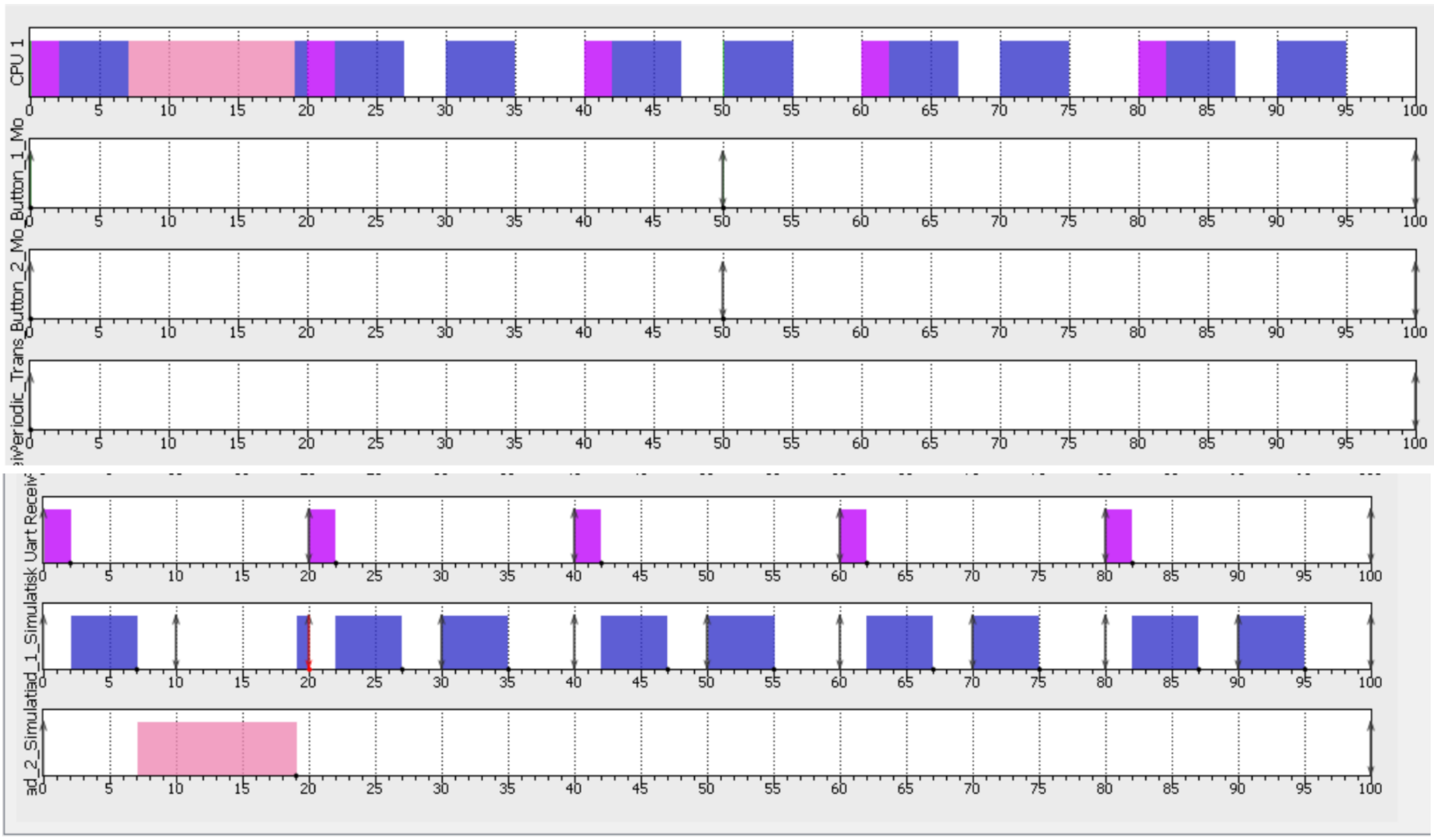
4- time demand analysis Task_Button_2_Monitor
w(10) = 5000*5 +2000*2+13*1+13 = 29026 < 50000
Task_Button_2_Monitor is schedulable.

5- time demand analysis for Task_Periodic_Transmitter
w(10) = 5000*10 + 2000*5+13*2+13*2+21 = 60072 < 100000
Task_Periodic_Transmitter is schedulable.

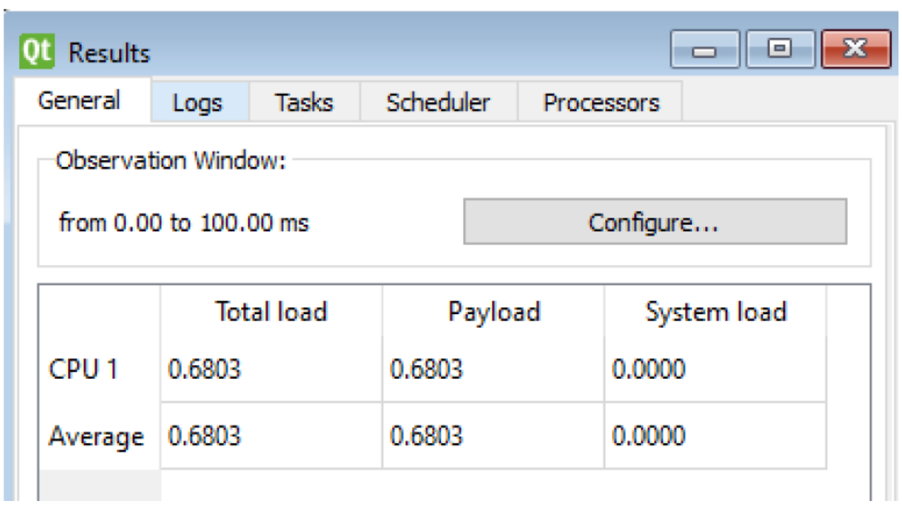
6- time demand analysis for Load_2_Simulation
w(10) = 5000*10 + 2000*5+12.6*2+13*2+21 +12000 = 72072 < 100000
Load_2_Simulation is schedulable

Using Simso simulator:

- Fixed Priority Scheduler:



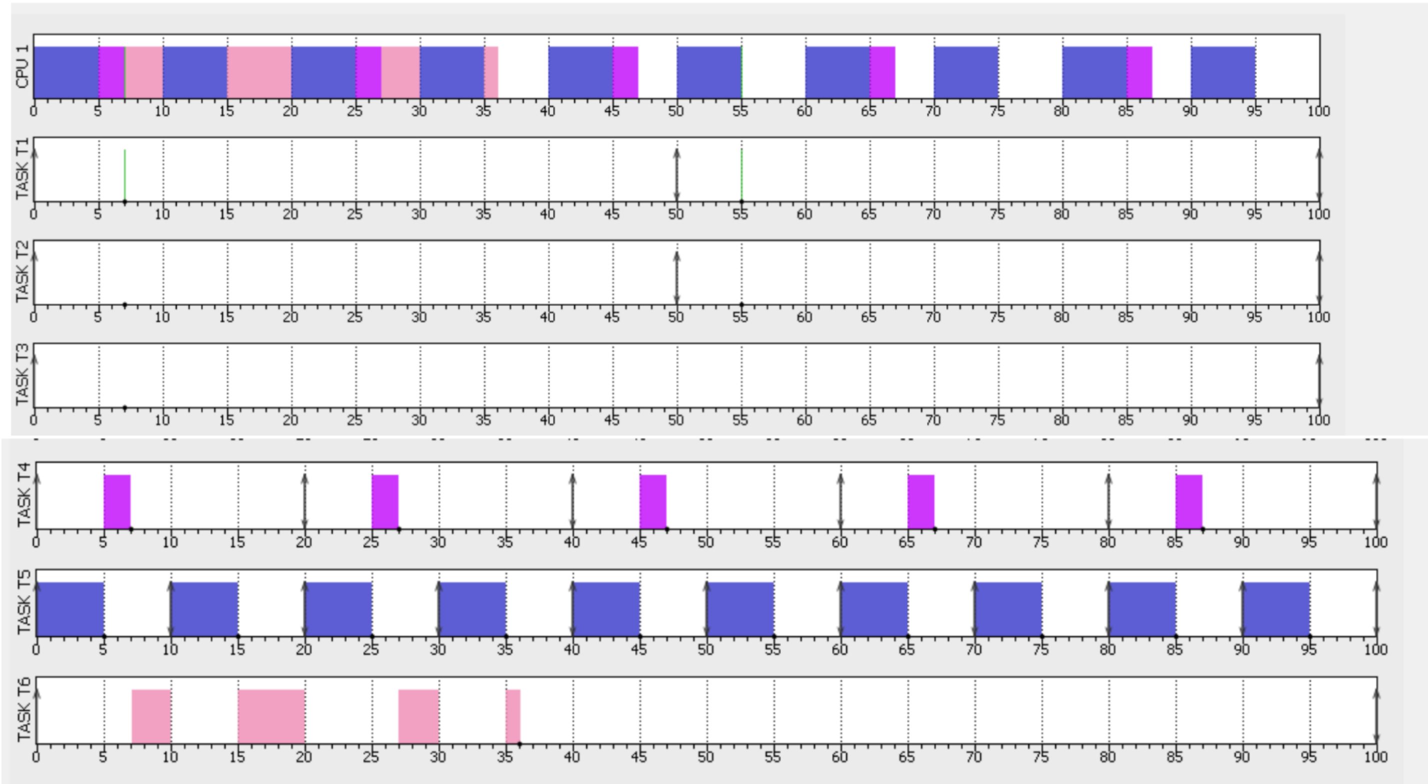
Tasks



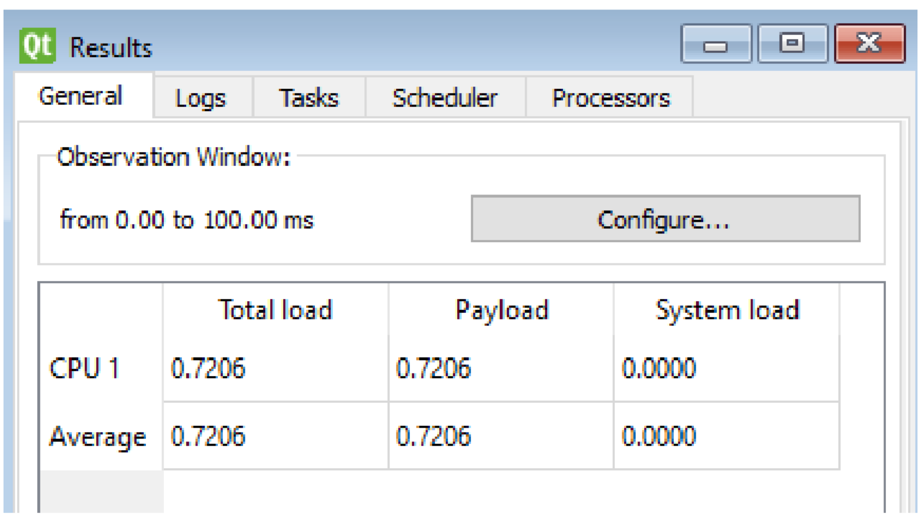
	Total load	Payload	System load
CPU 1	0.6803	0.6803	0.0000
Average	0.6803	0.6803	0.0000

CPU LOAD

EDF and Fixed priority rate monotonic scheduler:



Tasks



CPU LOAD

The CPU load result is the same as manual calculations since Simso calculates the results based on the worst-case scenario