

FreeRTOS EDF scheduler verification

Execution time tables :

| Task | Execution Time |
|----------------------|----------------|
| Button1_monitor | 13us |
| Button2_monitor | 13us |
| Periodic Transmitter | 20.8us |
| UART Receiver | 15us |
| Load1 | 5ms |
| Load2 | 12ms |

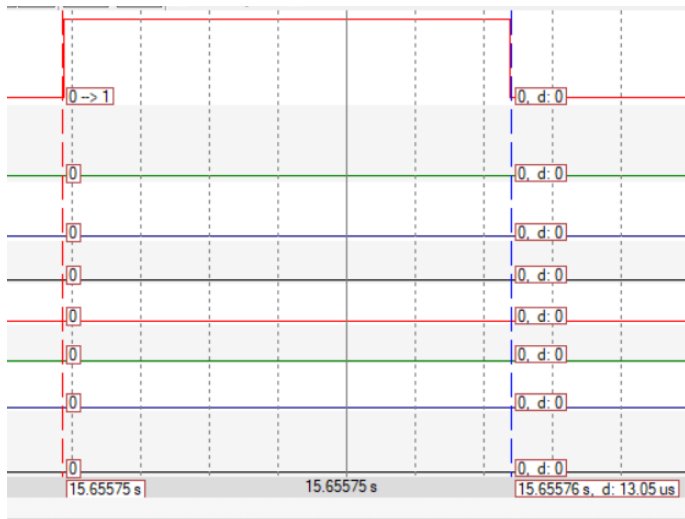


Figure 1 Button1 task execution time

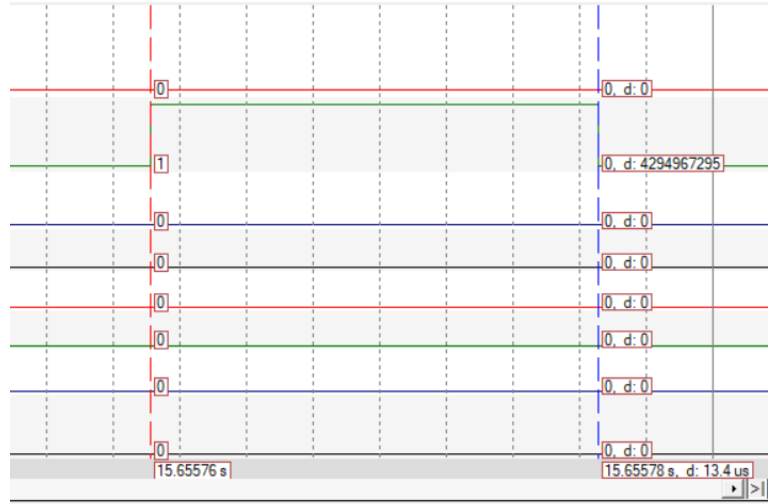


Figure 2 Button2 task execution time

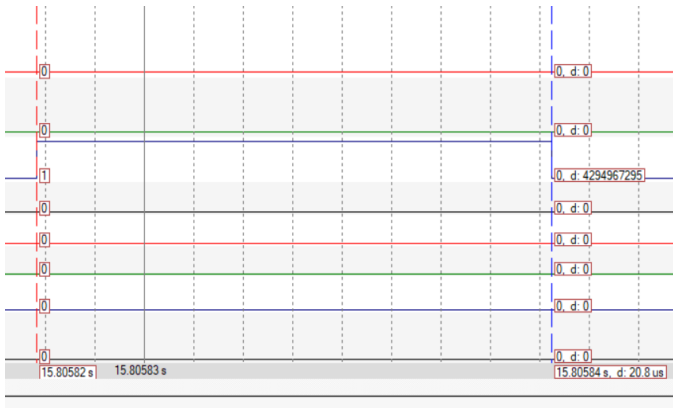


Figure 3 Periodic Task execution time

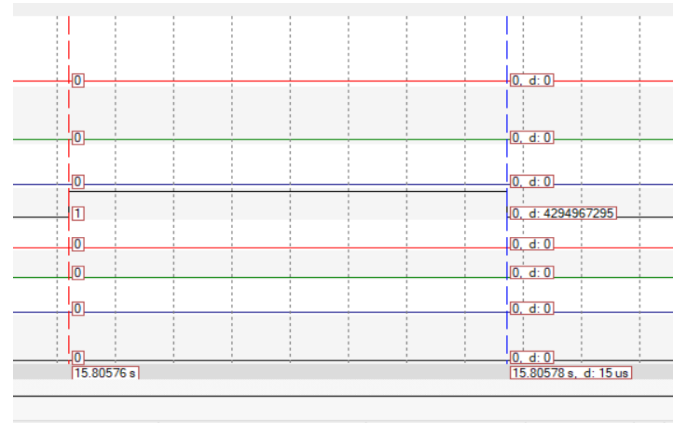


Figure 4 UART Receiver execution time

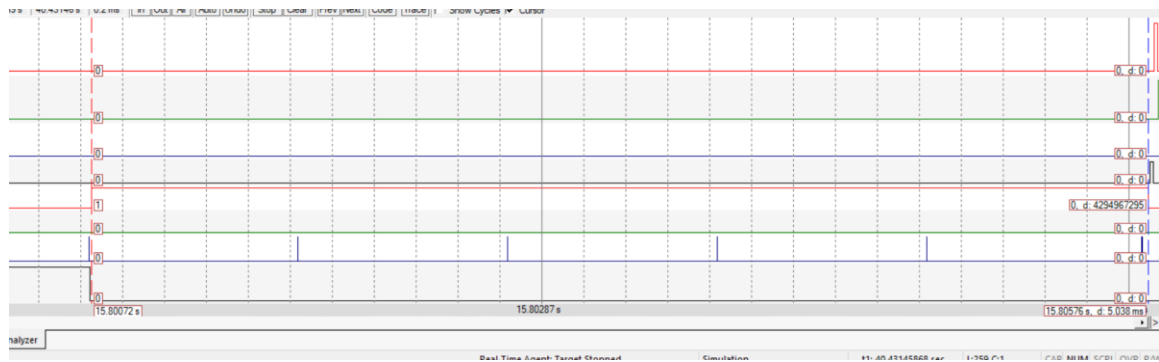


Figure 4 load1 Execution time

Analytical methods:

- System Hyper Period

The hyper period can be calculated easily using the following equation:

Hyper Period (H) = LCM(P_i), where P_i is all periodicities.

So, in our case, Hyper Period is 100 ms.

- CPU Load

CPU Load = $((0.0013 * 2) + (0.0013 * 2) + (0.0021 * 1) + (0.0015 * 5) + (5 * 10) + (12 * 1)) / 100 = \underline{\underline{0.62}}$

- System Schedulability by using URM Method

For the system to be schedulable, Total Utilization (U) must be less than or equal to Rate-Monotonic utilization bound (URM).

We previously calculated the total utilization, and it is found to be **0.62**.

$$URM = n \left(2^{\frac{1}{n}} - 1 \right) = 6 \left(2^{\frac{1}{6}} - 1 \right) = \mathbf{0.7347}$$

Utilization is found to be less than the URM, Therefore the system is guaranteed schedulable.

- System Schedulability by using Time Demand Method:

System schedulability can be analyzed through this equation:

$$W_i(t) = e_i + \sum_{k=1}^{i-1} \left(\frac{t}{P_k} \right) e_k \text{ for } 0 < P_i$$

Where :

W = Worst response time

E = Execution time

P = Periodicity

T = Time instance

Now we will calculate the time demand in earliest deadline order (earliest deadline comes first)

1) **Load1_Task :**

$W(10) = 5 + 0 = 5$, $W(10)$ is less than its deadline, so **Load1_Task** is schedulable.

2) **Uart_Receiver :**

$W(20) = 0.0016 + (20/10) * 5 = 10.0016$, $W(20) < 20$, so **Uart_Receiver** is schedulable.

3) **Button_1_Monitor**

$W(50) = 0.0013 + (50/20) * 0.0016 + (50/10) * 5 = 25.0053$, $W(50) < 50$, so **Button_1_Monitor** is schedulable.

4) **Button_2_Monitor**

$W(50) = 0.0013 + (50/20) * 0.0016 + (50/10) * 5 + (50/50) * 0.0013 = 25.0066$, $W(50) < 50$, so **Button_2_Monitor** is schedulable.

5) **Periodic_Transmitter**

$W(100) = 0.0021 + (100/20) * 0.0016 + (100/10) * 5 + (100/50) * 0.0013 + (100/50) * 0.0013 = 50.0153$, $W(100) < 100$, so **Periodic_Transmitter** is

schedulable.

6) Load2_Task

$W(100) = 12 + (100/20) * 0.0016 + (100/10) * 5 + (100/50) * 0.0013 + (100/50) * 0.0013 + (100/100) * 0.0021 = 62.0153$, $W(100) < 100$, so **Load2_Task** is schedulable.

From the above analysis we can conclude that our system is totally schedulable.

Using Simso offline simulator:

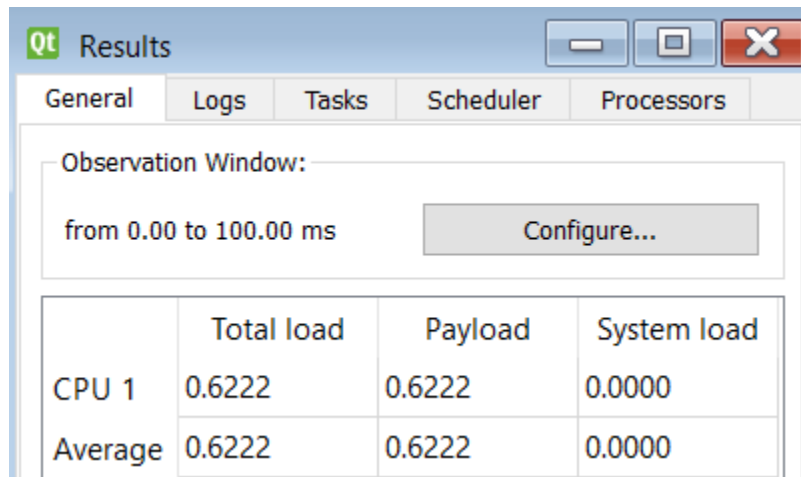


Figure 5 CPU load using Simso

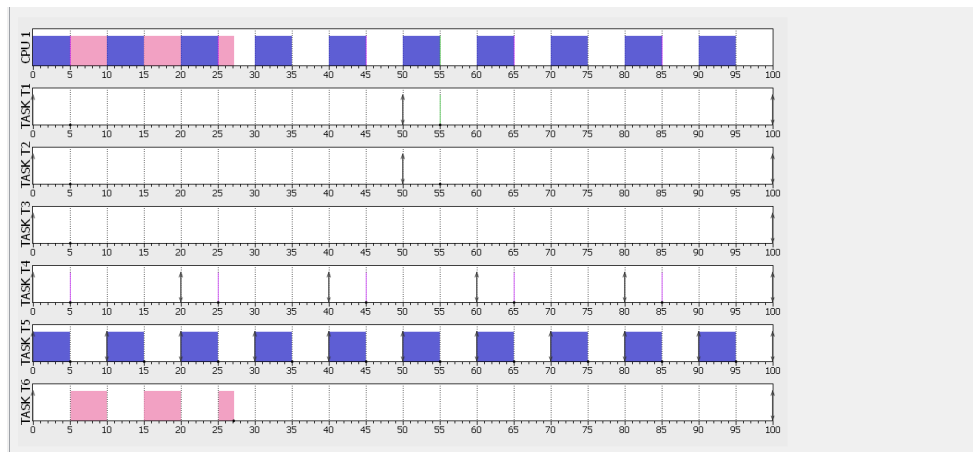


Figure 6 shows the Execution time of the tasks in Simso

As we can see the previous figure that Load2 task is pre-empted by Load1 task as it has lower periodicity.

Using Keil simulator:


| Watch 1 | | |
|--|-------|------|
| Name | Value | Type |
|  cpu_load | 62 | int |
| <Enter expression> | | |

Figure 7 CPU load in run-time in Keil

The CPU load is 62% as shown, the system is not too much loaded and the implementation is successful.

Second, the execution of all tasks, tick and idle tasks using trace macros and gpios.

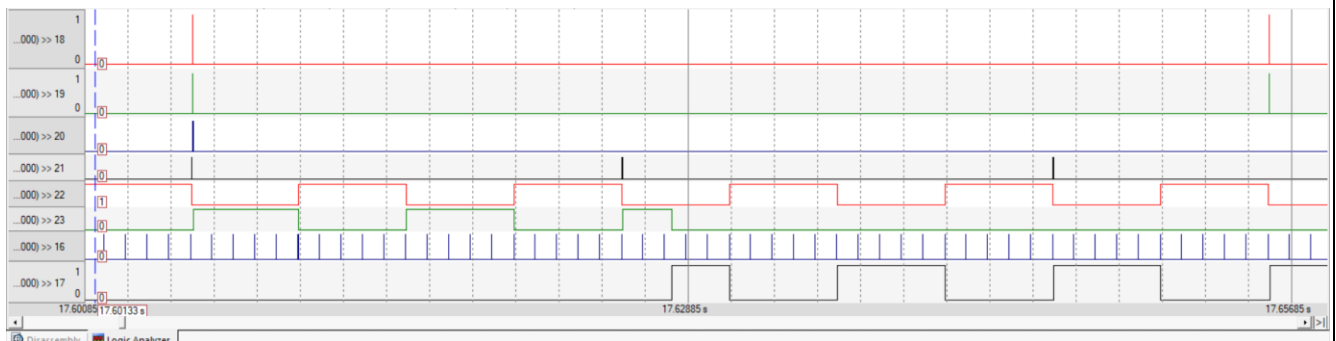


Figure 8 run-time in a logic analyzer

As we can see the previous figure the Load2 task is pre-empted by Load1 task as it has lower periodicity just like Simso so the implementation is successful.

Furthermore, when Button1 Task, Button2 Task, and Periodicity Task come at the same time, Button1 Task executes first then Button2 Task then Periodicity.