2022

EDF Scheduler Implementation

Made By Hussam Ali Ahmed

in fulfillment of Egfwd Advanced Embedded Systems Track - Real Time Operating Systems Part – Final Project

Verifying System Implementation

System Data

Task	Periodicity(ms)	Deadline(ms)	Execution Time
Button_1_Monitor	50	50	0.0013
Button_2_Monitor	50	50	0.0013
Periodic_Transmitter	100	100	0.0021
Uart_Receiver	20	20	0.0016
Load_1_Simulation	10	10	5
Load_2_Simulation	100	100	12

Table 1 System Data

1) Analytical Methods

• System Hyper Period

As we know that the hyper period can be calculated easily:

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

So, in our case Hyper Period is 100 ms.

• CPU Load

CPU Load = ((0.0013 * 2) + (0.0013 * 2) + (0.0021 * 1) + (0.0016 * 5) + (5 * 10) +(12 * 1)) / 100 = 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 = $n\left(2^{\frac{1}{n}}-1\right)=6\left(2^{\frac{1}{6}}-1\right)=0.73477.$

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:

$$w_i(t) = e_i + \sum_{k=1}^{i-1} \left(\frac{t}{p_k}\right) e_k \, for \; 0 < p_i$$

W = Worst response time

E = Execution time

P = Periodicity

T = Time instance

1) Calculate time demand for **Load_1_Task** as it will be the first task to be scheduled as it has the earliest deadline:

   W(10) = 5 + 0 = 5, W(10) is less than its deadline, so **Load_1_Task** is schedulable.

2) Calculate time demand for **Uart_Receiver** as it will be the second task to be scheduled as it has the second earliest deadline:

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

3) Calculate time demand for **Button_1_Monitor** as it will be the third task to be scheduled as it has the third earliest deadline:

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

4) Calculate time demand for **Button_2_Monitor** as it will be the fourth task to be scheduled as it has the same deadline as **Button_1_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) Calculate time demand for **Periodic_Transmitter** as it will be the fifth task to be scheduled as it has the fifth earliest deadline:

   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) Calculate time demand for **Load_2_Simulation** as it will be the sixth task to be scheduled as it has the sixth earliest deadline:

   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 **Load_2_Simulation** is schedulable.

From the above analysis, we find that our system is totally schedulable.

2) Simso Offline Simulator

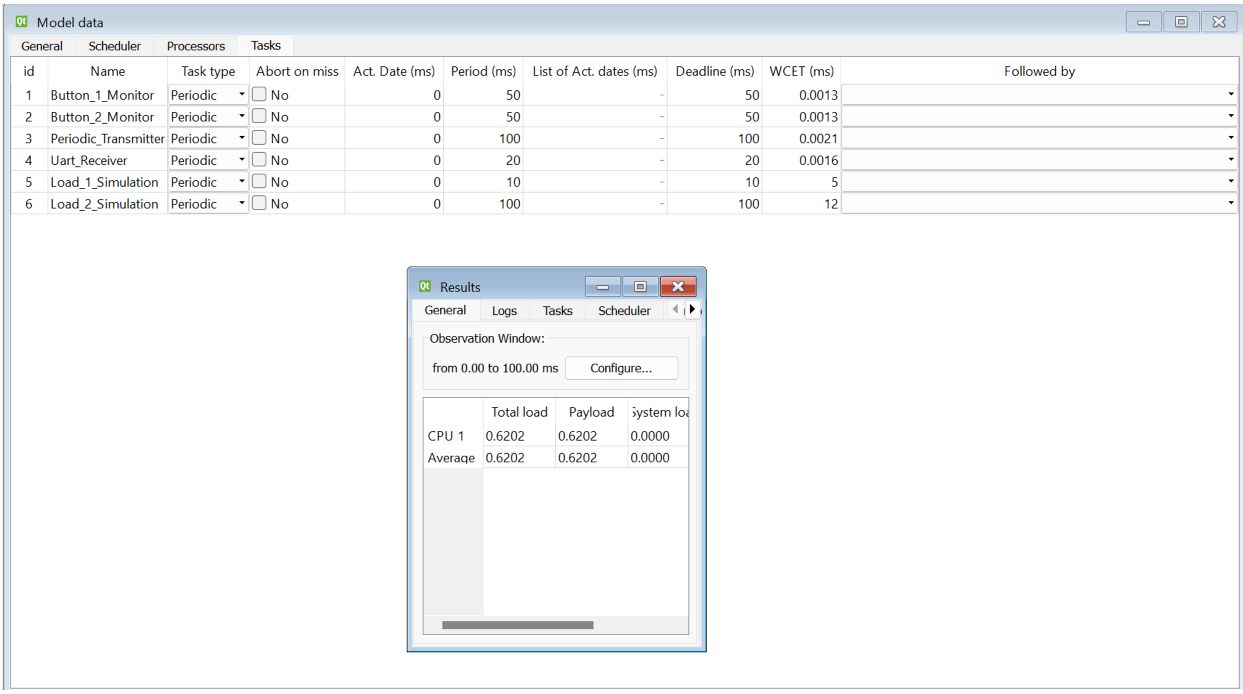

Figure 1 Tasks Set

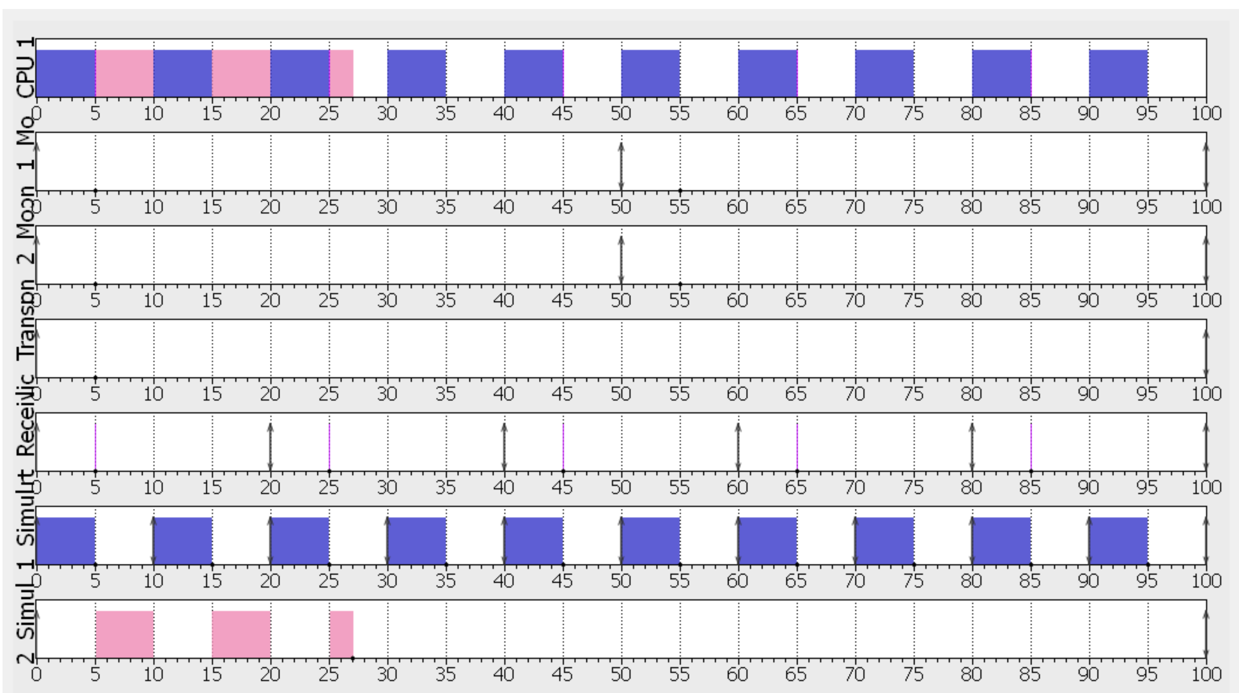

Figure 2 Gantt Chart

From the above figures, we can see that Load_2_Task is being pre-empted by Load_1_Task which is expected as Load_1_Task has periodicity less than that of Load_2_Task. Therefore, based on Rate-Monotonic scheduler lower periodicity task will have higher priority.

Also, the CPU Load is the same as calculated before analytically and there is no deadlines being missed which makes our system schedulable.

3) Keil Simulator

- **Pins**

Pins (PORT0)	Usage
PIN0(16)	Button 1 Input
PIN1(17)	Button 2 Input
PIN2(18)	Button 1 Task Tracing
PIN3(19)	Button 2 Task Tracing
PIN4(20)	Periodic Transmitter Task Tracing
PIN5(21)	UART Receiver Task Tracing
PIN6(22)	Load 1 Simulation Task Tracing
PIN7(23)	Load 2 Simulation Task Tracing
PIN8(24)	Ticks Tracing
PIN9(25)	Idle Task Tracing

Table 2 Pins

- **CPU Load**

Figure 3 CPU Load

The CPU Load was calculated by using timer1 and trace macros which is implemented in FreeRTOSConfig.h file.

This CPU Load agrees with the analytical methods and with simso which verifies our work.

- **Tasks Execution Plotting**

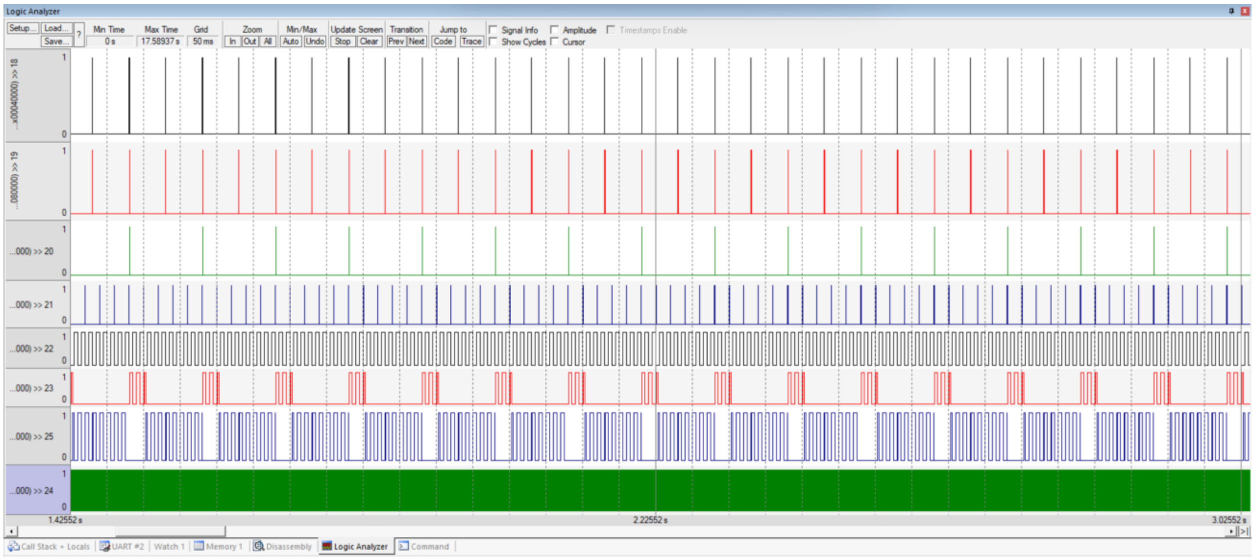

Figure 4 Tasks Execution Plotting

As seen in PIN25 which is the Idle Task, it is executed when no other task is running and stops when other tasks run.

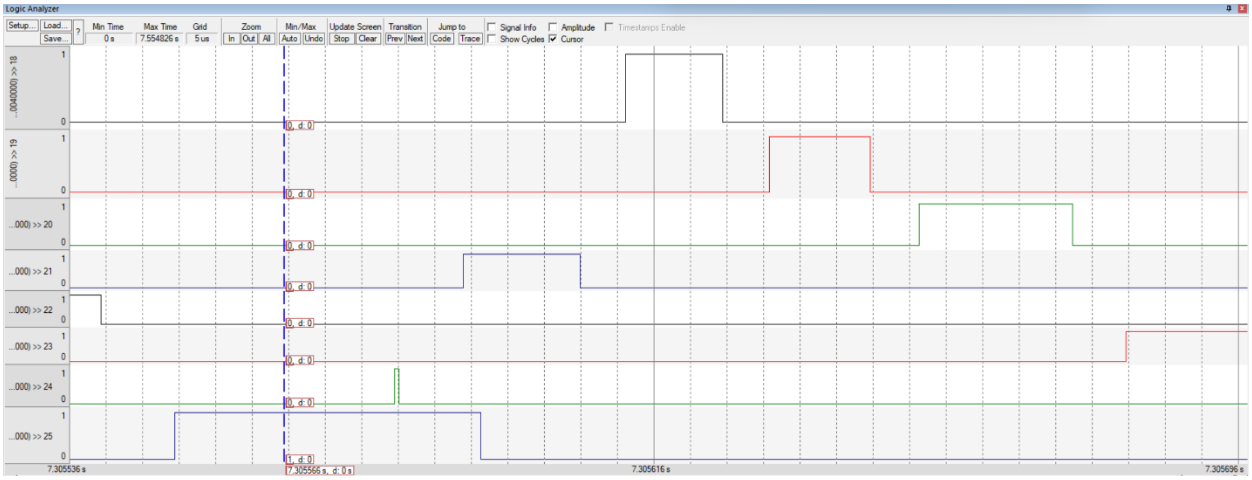

Figure 5 Zoomed In Plotting

As seen in the above graphs, Load_1_Task has the earliest deadline (10 ms) so it comes first which can be seen in PIN22, then UART_Receiver_Task (20 ms) PIN21, then Button_1_Task (50 ms) PIN18, then Button_1_Task (50 ms) PIN19, then Periodic_Transmitter_Task (100 ms) PIN20.

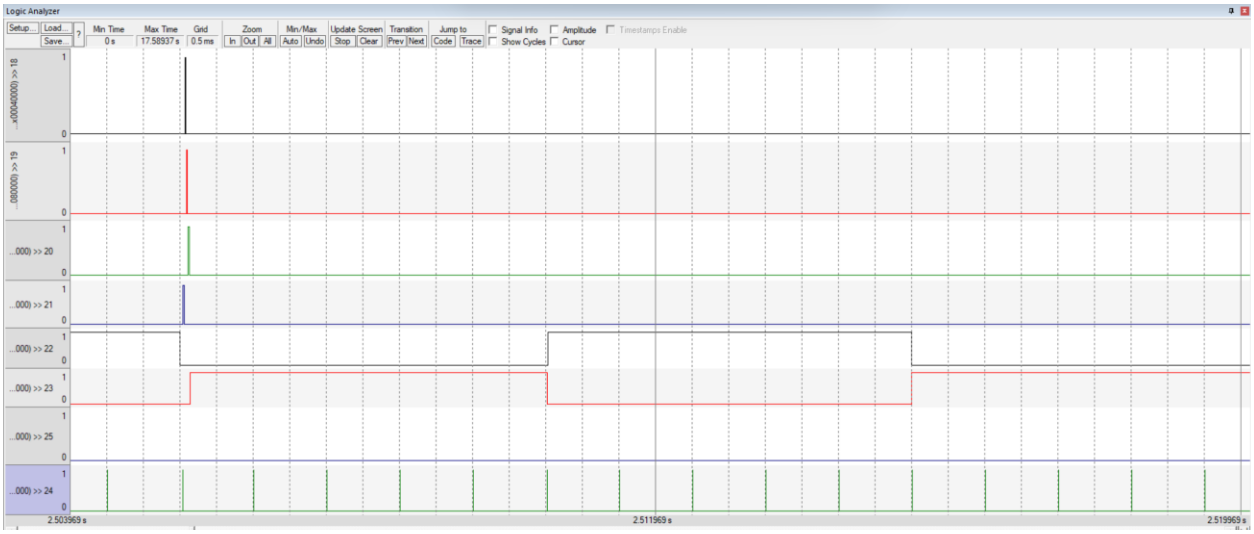

Figure 6

This figure shows that Load_1_Task preempts Load_2_Task as Load_1_Task has earliest deadline, so it comes first and preempts any running task.