

Analytical Methods

This document includes analytical methods of verifying EDF Scheduler based on FreeRTOS.

1. System Hyperperiod

- This project contains 6 tasks as follows :

Task	Period	Execution Time
Button_1_Monitor	50ms	13us
Button_2_Monitor	50ms	13.2us
Periodic_Transmitter	100ms	17.5us
Uart_Receiver	20ms	27.35us
Load_1_Simulation	10ms	5ms
Load_2_Simulation	100ms	12ms

- Using the above table, we can easily calculate the Hyperperiod for these tasks (the period after which all tasks repeat execution again)

$$\text{Hyperperiod} = 100\text{ms}$$

2. CPU Load

- To calculate the CPU load, we need to calculate the execution time for each task multiplied by number of times these tasks came through one hyperperiod, then by summing these times and divide by the Hyperperiod we get the CPU load.
- Using the same above table, we get :

$$\text{i. CPU load} = \frac{[(13 * 2) + (13.2 * 2) + 17 * 5 + (27.35 * 5)] * 10^{-3} + (5 * 10) + 12}{100} \times 100\%$$

$$\text{CPU Load} = 62.2\%$$


3. System Schedulability

- This property can be determined using two methods:
 - i. Rate-Monotonic utilization bound (only for RM Schedulers)
 - ii. Time demand analysis
- Let's start with Rate-Monotonic method

Rate Monotonic Utilization Bound

$$U = \sum_{i=1}^n \frac{C_i}{P_i} \leq n(2^{\frac{1}{n}} - 1)$$

U = Total Utilization
C = Execution time
P = Periodicity
N = Number of tasks

- There are two side of the above equation
 - i. The right-hand side calculates the summation of ratio between the execution time of a task and the periodicity of that task
 - ii. The left-hand side is called URM which is considered the metric of the system schedulability
- By comparing these two sides of equation :
 - i. if the right-hand side is less than or equal to the URM term, system is schedulable
 - ii. Otherwise, system is not schedulable
- By Applying this equation on our system :
 - The right-hand side :
$$\frac{13 \cdot 10^{-3}}{50} + \frac{13.2 \cdot 10^{-3}}{50} + \frac{17.5 \cdot 10^{-3}}{100} + \frac{27.35 \cdot 10^{-3}}{20} + \frac{5}{10} + \frac{12}{100} = 0.622$$
 - The left-hand side (URM) :
$$6 \left(2^{\frac{1}{6}} - 1 \right) = 0.73477$$
 - So, $0.622 \leq 0.73477$  **System is Schedulable**

- For now, let's illustrate the second method

Time Demand Analysis

- This method measures the time required against the time provided for each task

$$w_i(t) = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k \quad \text{for } 0 < t \leq p_i$$

W = Worst response time
 E = Execution time
 P = Periodicity
 T = Time instance

- Let's reorder our tasks based on priorities to match Rate-Monotonic scheduler rules
 - tasks with higher periodicity (comes faster) take higher priorities

Priority	Task	Period	Execution Time
0	Load_1_Simulation	10ms	5ms
1	Uart_Receiver	20ms	27.35us
2	Button_1_Monitor	50ms	13us
2	Button_2_Monitor	50ms	13.2us
3	Periodic_Transmitter	100ms	17.5us
3	Load_2_Simulation	100ms	12ms

** Priority : lower value means higher priority

- Using the above equation, let's calculate response time for each task taking into consideration the effect of other tasks if they have higher priority
 - Load_1_Simulation :

$$W(10) = 5 + 0 = 5\text{ms} \leq 5\text{ms} \rightarrow \text{Schedulable}$$
 - Uart_Receiver :

$$W(20) = 27.35 * 10^{-3} + 5 * \left(\frac{20}{10}\right) = 10.027\text{ms} \leq 20\text{ms} \rightarrow \text{Schedulable}$$
 - Button_1_Monitor :

$$W(50) = 13 * 10^{-3} + 27.35 * 10^{-3} * \frac{50}{20} + 5 * \left(\frac{50}{10}\right) = 25.095\text{ms} \leq 50\text{ms} \rightarrow \text{Schedulable}$$
 - Button_2_Monitor :

$$W(50) = 2 * 13 * 10^{-3} + 27.35 * 10^{-3} * \frac{50}{20} + 5 * \left(\frac{50}{10}\right) = 25.108\text{ms} \leq 50\text{ms} \rightarrow \text{Schedulable}$$

➤ Periodic_Transmitter :

$$W(100) = (17.5 * 10^{-3}) + \left(2 * 13 * 10^{-3} * \frac{100}{50}\right) + \left(27.35 * 10^{-3} * \frac{100}{20}\right) + \left(5 * \frac{100}{10}\right) = 50.188\text{ms} \leq 100\text{ms}$$

→ Schedulable

➤ Load_2_Simulation :

$$W(100) = 12 + (17.5 * 10^{-3}) + \left(2 * 13 * 10^{-3} * \frac{100}{50}\right) + \left(27.35 * 10^{-3} * \frac{100}{20}\right) + \left(5 * \frac{100}{10}\right) = 62.206\text{ms} \leq 100\text{ms}$$

→ Schedulable

- Using the above results, all tasks are schedulable



System is Schedulable

SIMSO Offline Results

In this part we will show simso offline simulator results for our project including same tasks discussed above in the previous part. This step mainly purpose is to verify the above results of analytical approach and see if they match or not.

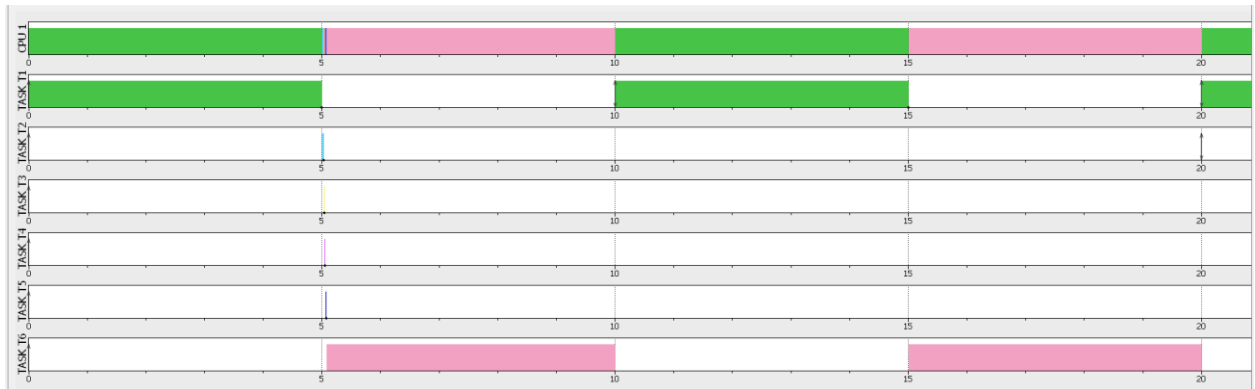
1.

Model data								
General		Scheduler	Processors	Tasks				
id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)
1	TASK T1	Periodic	<input type="checkbox"/> No	0	10	-	10	5
2	TASK T2	Periodic	<input type="checkbox"/> No	0	20	-	20	0.02735
3	TASK T3	Periodic	<input type="checkbox"/> No	0	50	-	50	0.013
4	TASK T4	Periodic	<input type="checkbox"/> No	0	50	-	50	0.0132
5	TASK T5	Periodic	<input type="checkbox"/> No	0	100	-	100	0.0175
6	TASK T6	Periodic	<input type="checkbox"/> No	0	100.0	-	100.0	12.0

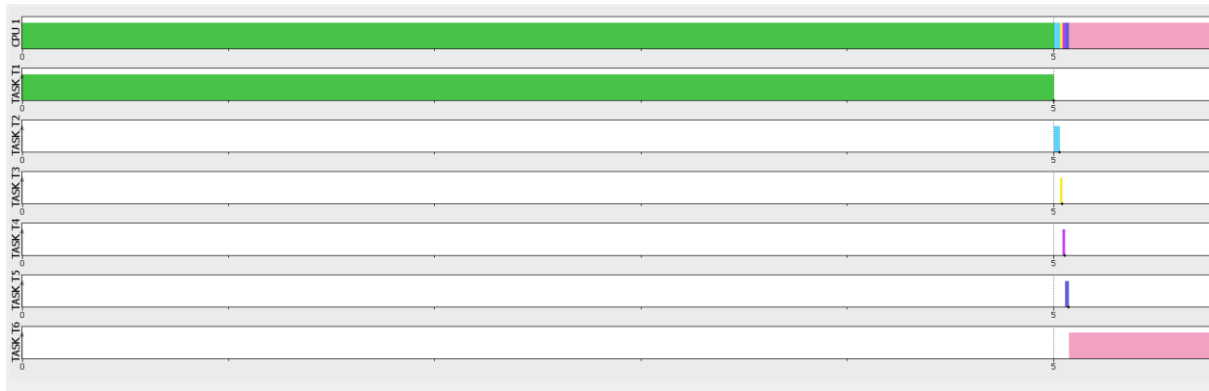
Results			
General		Logs	Tasks
Observation Window:			
from 0.00 to 100.00 ms		Configure...	
	Total load	Payload	System load
CPU 1	0.6221	0.6221	0.0000
Average	0.6221	0.6221	0.0000

- As we see that CPU load = 62.2% which similar to analytical approach

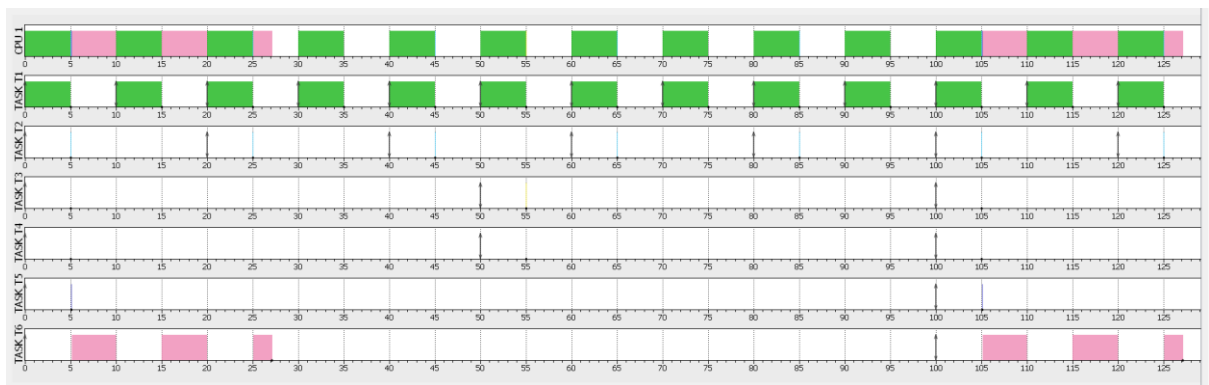
2.



- Here we can see that first task (Load_1_Simulation) executes for 5ms and happen every 10ms with highest priority, which is totally expected
- Similarly with rest tasks we can see each task and when to execute
- Task2, Task3, Task4 and Task5 have very small execution time so it's not clear enough on simulation



- This is a clearer screenshot which clarifies those short execution time tasks.

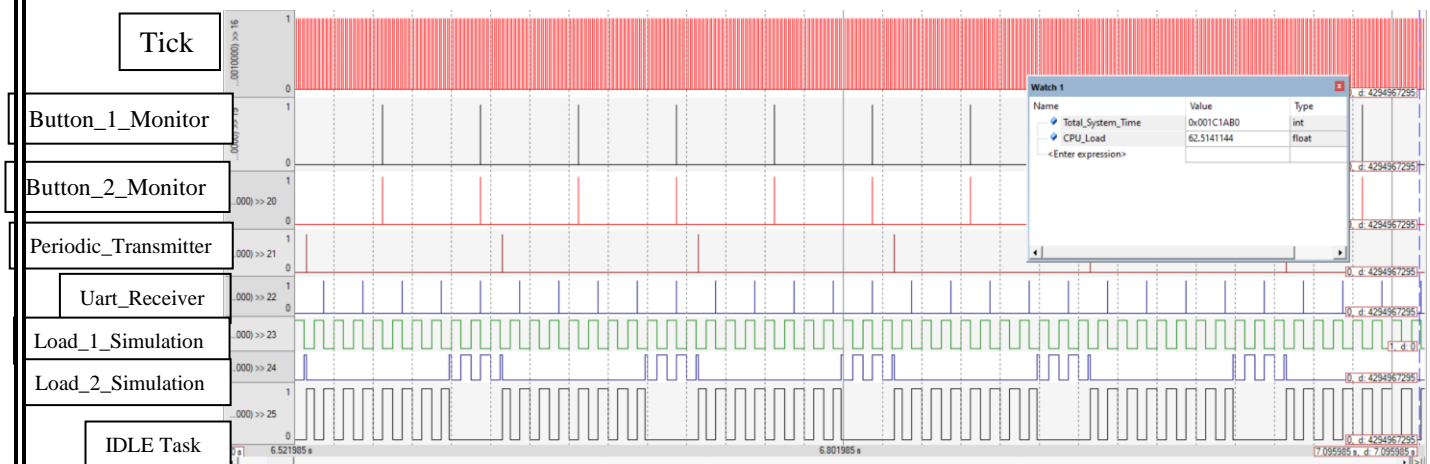


- Here is the whole situation including all tasks (short time tasks will not appear due to zoom out)
- We can see the case of executing task6 while task1 came (higher priority and earlier deadline) , so task1 preempted task6 and this happens three times every execution of task6
- From above screenshots using simso, and the above results of analytical approach, we found that both approaches lead to same results and there is a matching between them.

Keil Simulator Results

In this part we will show Keil simulator results for our project including same all tasks discussed above in the two previous parts. Using timer1 and macro tracing we will calculate the CPU usage (load) and verify that all tasks run well with correct time constraints.

1.

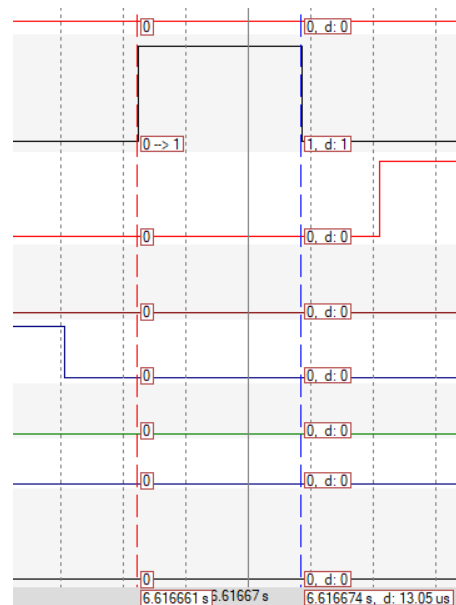


Here we can see that CPU_Load is about 62.5% (saturate between 62% and 63%).

This result matches our previous two approaches (analytical and simso simulator)

2.

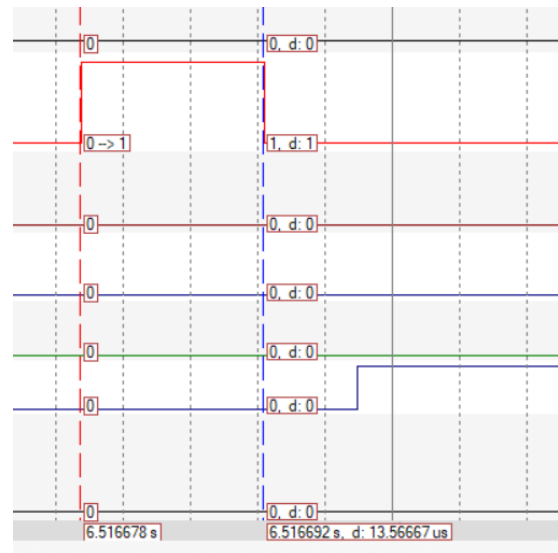
This shows the execution time of Button_1_Monitor task in run time. Which is around 13us.



3.

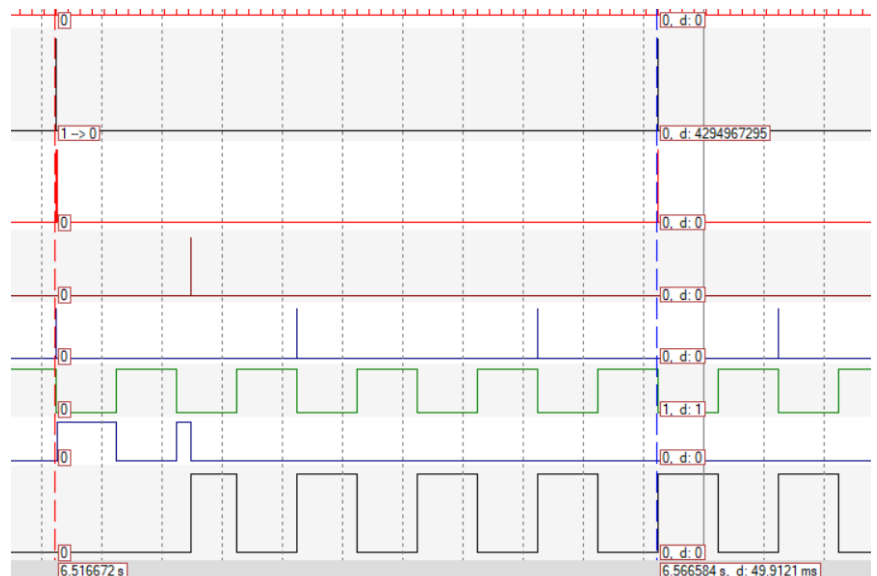
This shows the execution time
of Button_2_Monitor task in run time.

Which is around 13us.



4.

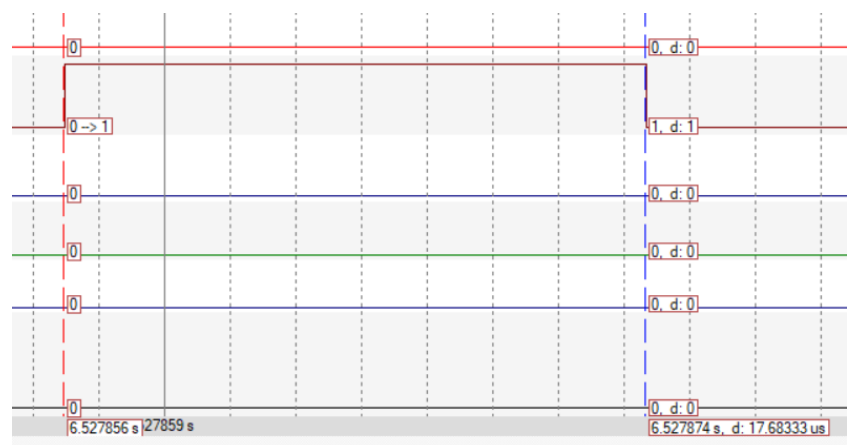
This shows That both
Button_1_Monitor and
Button_2_Monitor tasks
periodicity is about 50ms



5.

This shows the execution
time of Periodic_Transmitter
task in run time.

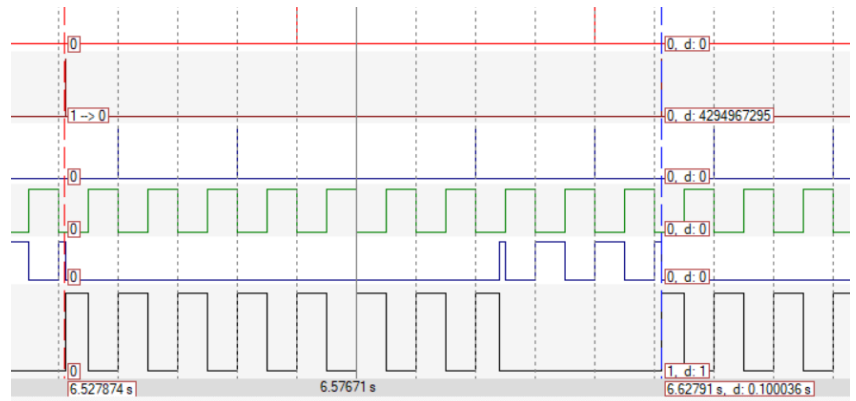
Which is around 17.5us.



6.

This shows the periodicity of
Periodic_Transmitter task

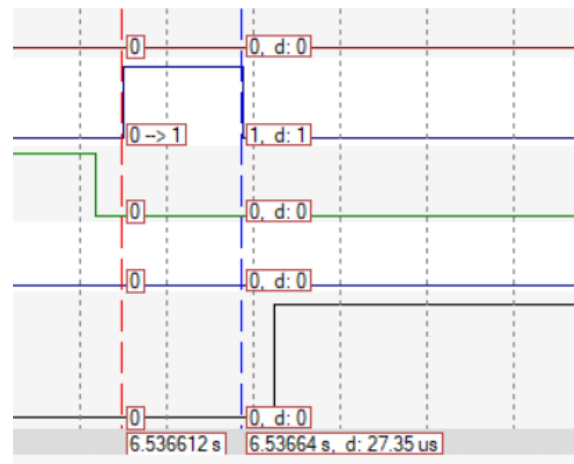
Which is around 100ms



7.

This shows the execution time of
Uart_Receiver task in run time.

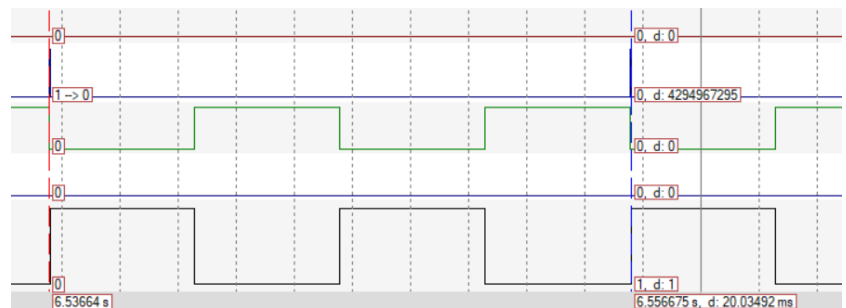
Which is around 27.5us.



8.

This shows the periodicity of
Uart_Receiver task

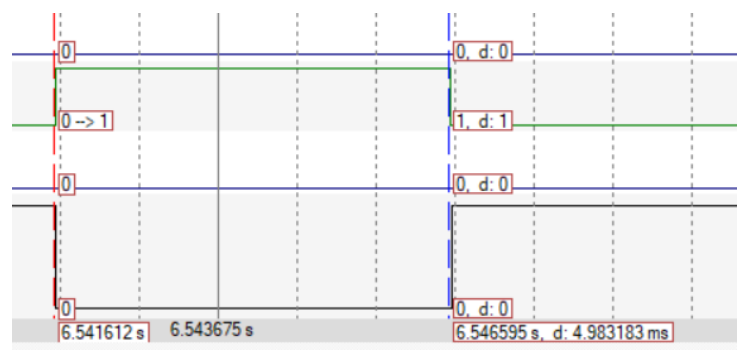
Which is around 20ms



9.

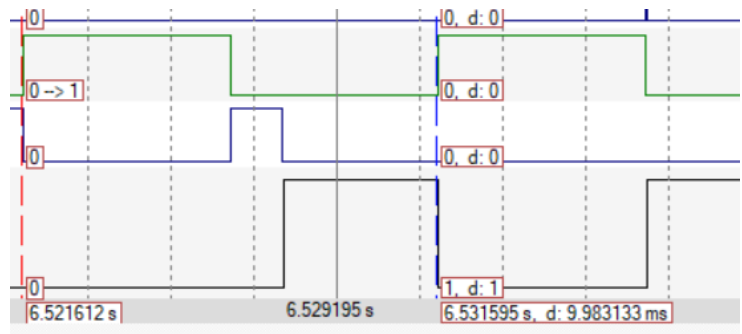
This shows the execution time of
Load_1_Simulation task in run time.

Which is around 5ms as required.

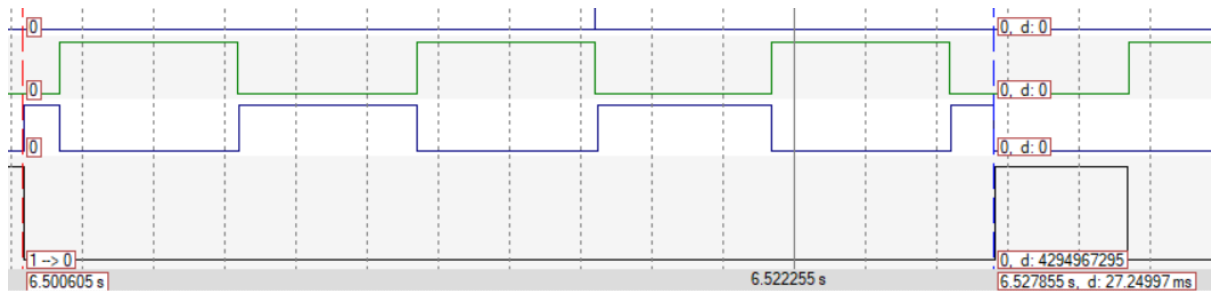


10.

This shows the periodicity of
Load_1_Simulation task
Which is around 10ms

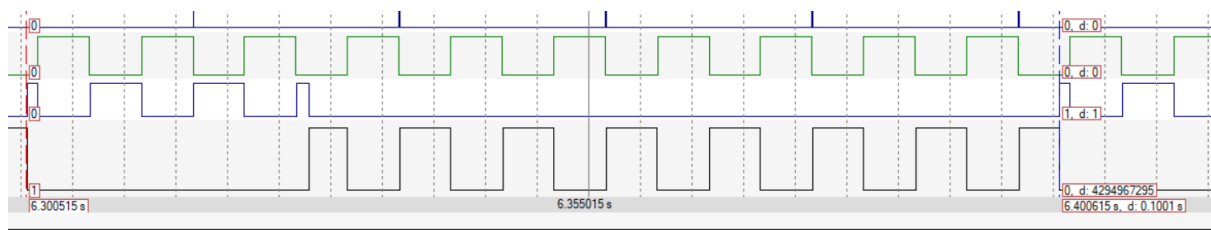


11.



This case shows that Load_2_Simulation task (the blue one) is preempted by the above task (Load_1_Simulation) three times, so total execution time for Load_2_Simulation task = $27 - (3 \times 5) = 12$ ms which equals the time required in specifications.

12.



This one shows the periodicity of Load_2_Simulation task (the blue one) which is 100ms.