Embedded Systems Advanced Nanodegree Program

Real-Time Operating Systems

Project: Implementing EDF Scheduler

Name: Elsayed Ayman Elsayed Ali Habib

Email: sayedaymanhabib@gmail.com

Contents

Introduction

EDF adopts a dynamic priority-based preemptive scheduling policy, meaning that the priority of a task can change during its execution, and the processing of any task is interrupted by a request for any higher priority task. The algorithm assigns priorities to tasks in a simple way: the priority of a task is inversely
proportional to its absolute deadline; In other words, the highest priority is the one with the earliest deadline. In case of two or more tasks with the same absolute deadline, the highest priority task among them is chosen random.

Annalytical Methods

Task	Periodicity	Deadline	occurrence during hyperperiod	Execution Time once	Total Execution Time during hyperperiod
ButtonMonitor_1	50	50	2	5us	10us
ButtonMonitor_2	50	50	2	5us	10us
PeriodicTransmitter	100	100	1	8.8us	8.8us
UartReceiver	20	20	5	6.4us	32us
LoadSimulation_1	10	10	10	5ms	50ms
LoadSimulation_2	100	100	1	12ms	12ms

Hyperperiod: The time period after which a pattern starts to repeat itself is called hyperperiod H, and is, in fact, the smallest multiple of all the periods.

$$\text{Hyperperiod} = LCM(50,50,100,20,10,100)$$

$$\text{Hyperperiod} = 100$$

Utilization (U):

Total Execution Time During Hyperperiod / Hyperperiod

$$U = \left(\frac{(10u) + (10u) + (8.8u) + (32u) + (50m) + (12m)}{100ms}\right) * 100\% = 62.06088\%$$

System Schedulability: What is General Schedulability test?

The schedulability test decides if a given task set can be scheduled such that no tasks in the set miss their deadlines. Exact schedulability tests usually have high time complexities and may not be adequate for online admission control where the system has a large number of tasks or a dynamic workload.

Rate Monotonic:

$$U_{rm} \leq n\left(2^{\frac{1}{n}} - 1\right)$$

$$U_{rm} = 6\left(2^{\frac{1}{6}} - 1\right) = 0.7348$$

$$n = 6$$

$$U = 62.06088\%$$

$$therefor\ U < U_{rm}$$

Therefore, the system is feasible (Schedulable).

Time Demand Analysis:

$$W_i = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k$$

w: worst response time
e: execution time
t: time instance
P: periodicity
i: task number

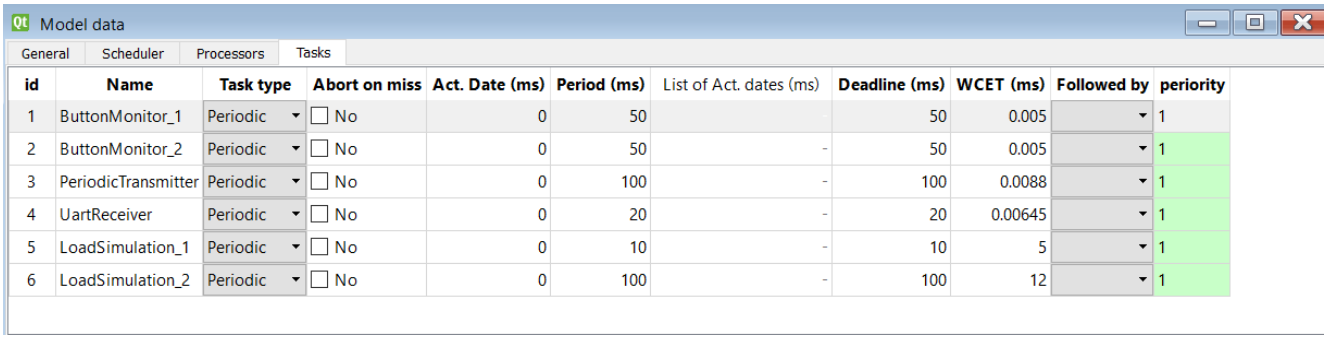
critical instant = 100ms		
LoadSimulation_1 $E: 5ms\ , P: 10ms\ , D: 10ms$	$w_1(10) = 5m + 0 = 5\ , w(10) = 5 < 10$	schedulable
UartReceiver $E: 32.25us\ , P: 20ms\ , D: 20ms$	$w_2(20) = 32.25u + \left(\frac{20}{10}\right) * 5m\ = 10.032, w(20) = 10.0.32 < 20$	schedulable
ButtonMonitor_1 $E: 10us\ , P: 50ms\ , D: 50ms$	$w_3(50) = 10u + \left(\frac{50}{10}\right) * 5m + \left(\frac{50}{20}\right) * 32.25u = 25.09\ , w(50) = 25.09 < 50$	schedulable
ButtonMonitor_2 $E: 10us\ , P: 50ms\ , D: 50ms$	$w_3(50) = 10u + \left(\frac{50}{10}\right) * 5m + \left(\frac{50}{20}\right) * 32.25u + \left(\frac{50}{50}\right) * 10u = 25.1\ , w(50) = 25.1 < 50$	schedulable
PeriodicTransmitter $E: 8.8us\ , P: 100ms\ , D: 100ms$	$w_3(100) = 8.8u + \left(\frac{100}{10}\right) * 5m + \left(\frac{100}{20}\right) * 32.25u + \left(\frac{100}{50}\right) * 10u + \left(\frac{100}{50}\right) * 10u = 51.6613\ , w(100) = 51.6613 < 50$	schedulable
LoadSimulation_2 $E: 12ms\ , P: 100ms\ , D: 100ms$	$w_3(100) = 12m + \left(\frac{100}{10}\right) * 5m + \left(\frac{100}{20}\right) * 32.25u + \left(\frac{100}{50}\right) * 10u + \left(\frac{100}{50}\right) * 10u + \left(\frac{100}{100}\right) * 8.8u = 63.6613\ , w(100) = 63.6613 < 50$	schedulable

Therefore the system is Schedulable

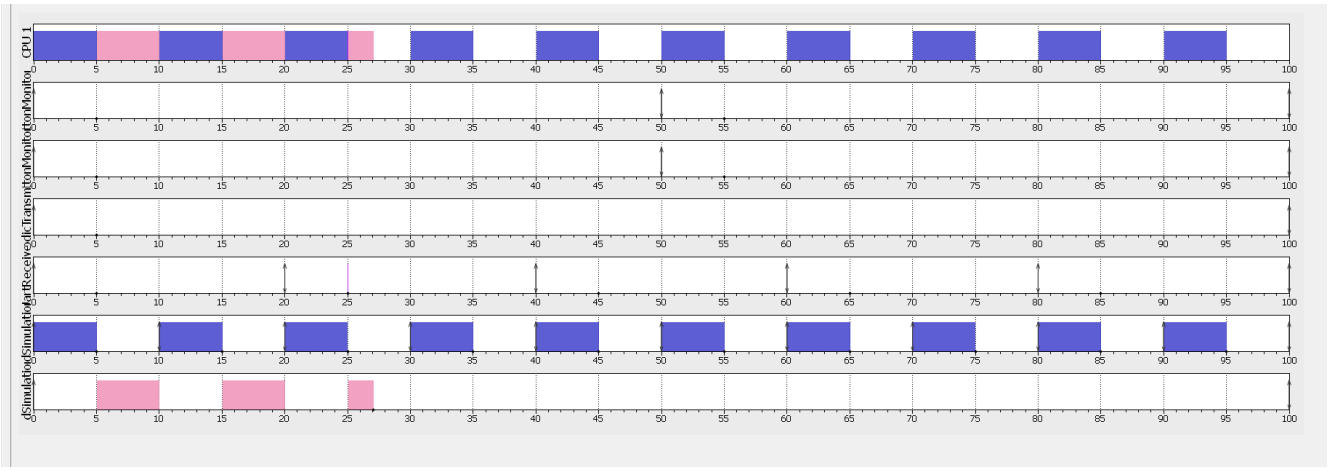
Simso Offline Simulator

Simso Offline Simulator: The simulator used is Simso, It is a discrete event simulator used to evaluate real-time scheduling algorithms (mono or multiprocessor).

➔ We choose the scheduler (EDF rate-monotonic scheduler) and set each task with the required period and execution time.

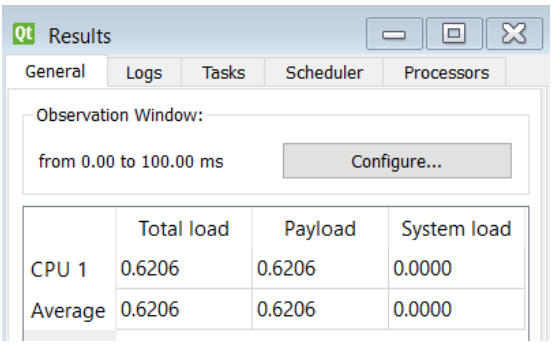


id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	periority
1	ButtonMonitor_1	Periodic	No	0	50	-	50	0.005		1
2	ButtonMonitor_2	Periodic	No	0	50	-	50	0.005		1
3	PeriodicTransmitter	Periodic	No	0	100	-	100	0.0088		1
4	UartReceiver	Periodic	No	0	20	-	20	0.00645		1
5	LoadSimulation_1	Periodic	No	0	10	-	10	5		1
6	LoadSimulation_2	Periodic	No	0	100	-	100	12		1



from the Gantt chart:

1- (ButtonMonitor_1) is Executed twice in the Hyperperiod.
2- (ButtonMonitor_2) is Executed twice in the Hyperperiod.
3- (PeriodicTransmitter) is Executed once in the Hyperperiod.
4- (UartReceiver) is executed five times in the hyperperiod.
5- (LoadSimulation_1) is Executed ten times in the hyperperiod.
6- Execute (LoadSimulation_2) is Executed once in the Hyperperiod.
(From running EDF rate-monotonic scheduler)
CPU_Load = Total load = 62.06%

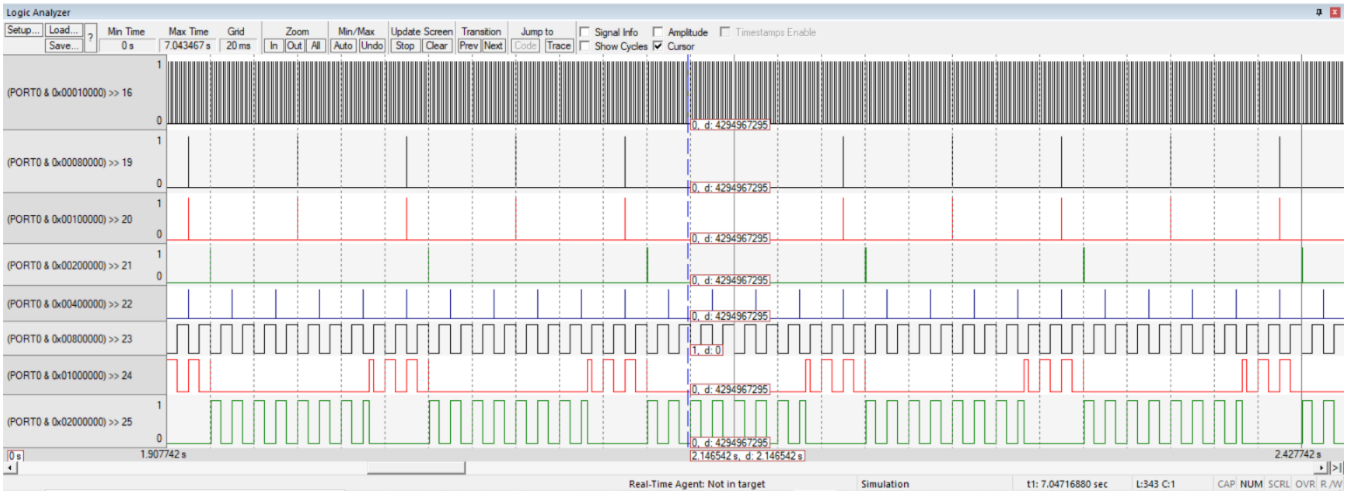


	Total load	Payload	System load
CPU 1	0.6206	0.6206	0.0000
Average	0.6206	0.6206	0.0000

Keil Simulator

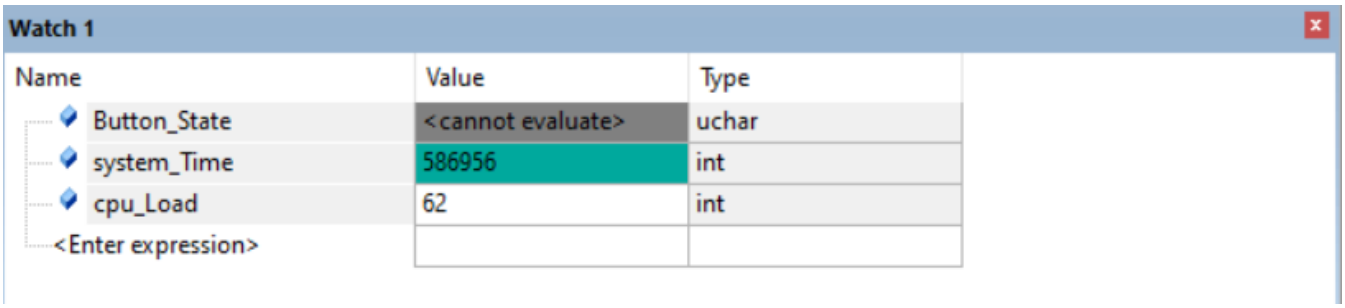
Keil Simulator: The µVision Simulator allows you to debug programs using only your PC and device simulation drivers provided by Keil and various third-party developers we use --> (Logic Analyzer).

➔ Using trace macros and GPIOs to plot the execution of all tasks, tick and the idle task on the logic analyzer.



➔ Calculate cpu load using trace macros and timer1.

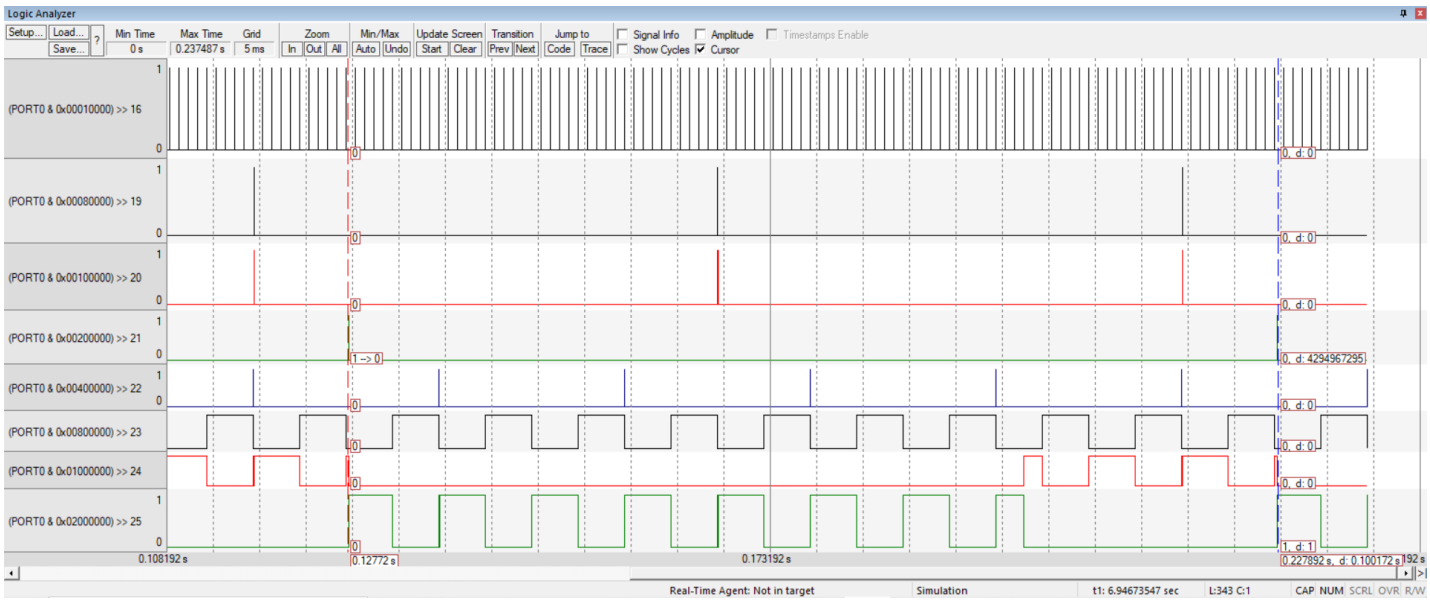
➔ CPU_load for all tasks on execution = 62% of the total CPU.



Logic Analyzer

➔ using **Logic Analyzer** to get:

1. Execution Time of the task.
2. Will the task carry out its task before the deadline or not?
3. Make sure that all tasks are executed a number of times in the hyperperiod depending on the periodicity.



❖ Depending on the previous logic analyzer.

In the first hyperperiod:
The task (LoadSimulation_1) is executed first,
then the task (UartReceiver),
then the task (ButtonMonitor_1),
then the task (ButtonMonitor_2),
then the task (PeriodicTransmitter),
then the task (LoadSimulation_2) and all tasks do not lose the deadline.

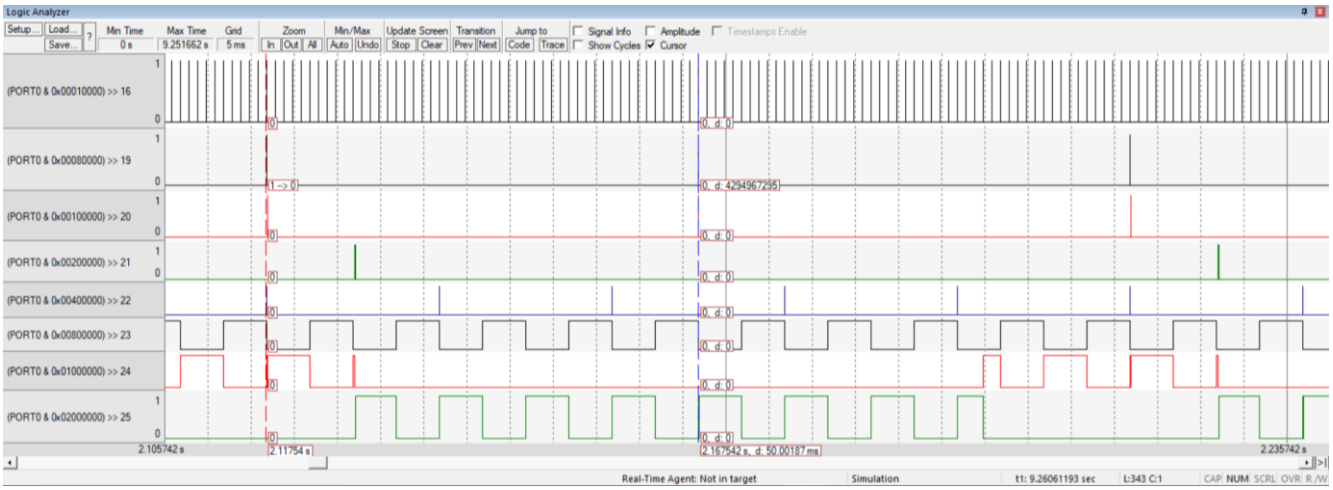
this depends on the periodicity

Because the EDF Scheduler works to execute less periodicity first and then the next in the (xReadyTasksListEDF).

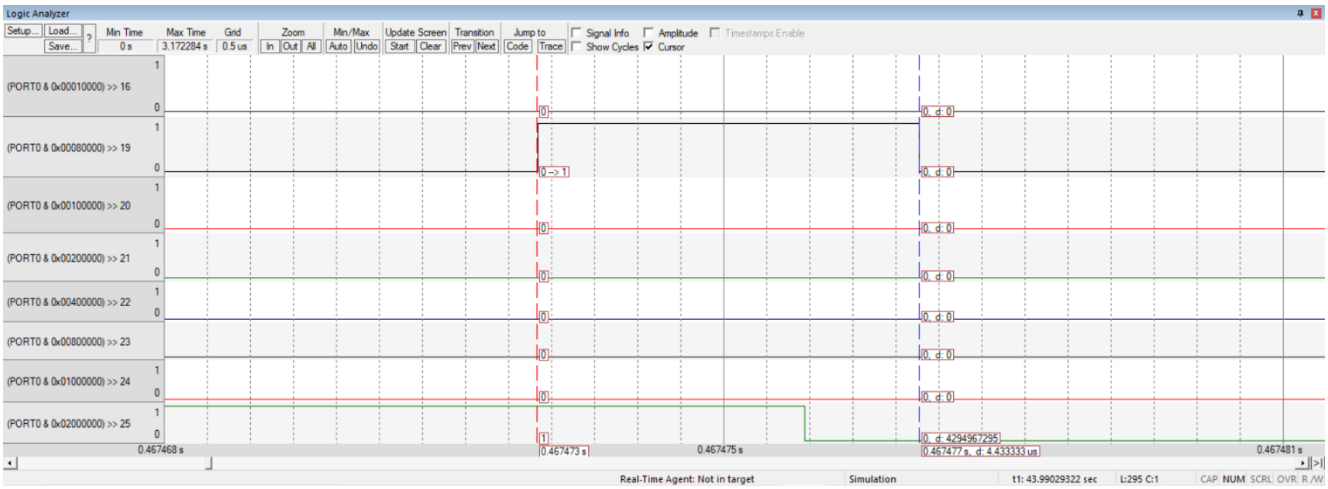
(ButtonMonitor_1) and (ButtonMonitor_2) tasks: execute first depending on which task of them was added in the first of (xReadyTasksListEDF).

Task_1 --> Button_1_Monitor

- Deadline = 50ms



- Execution Time of Task (Button_1_Monitor) = 5 microsecond

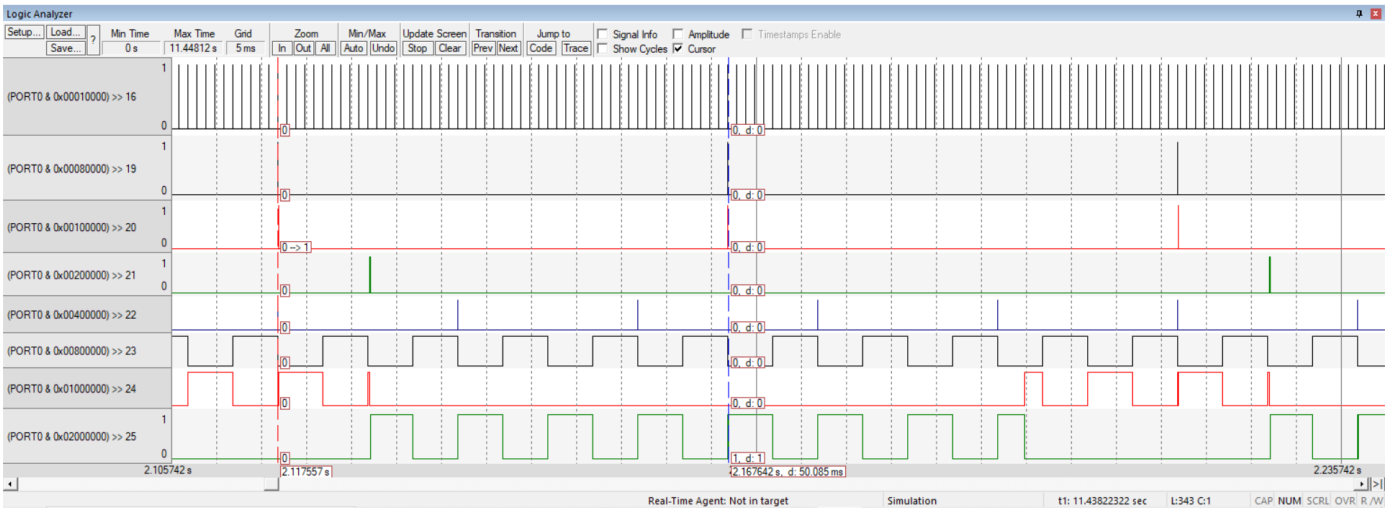


- occurrence during hyperperiod = 2

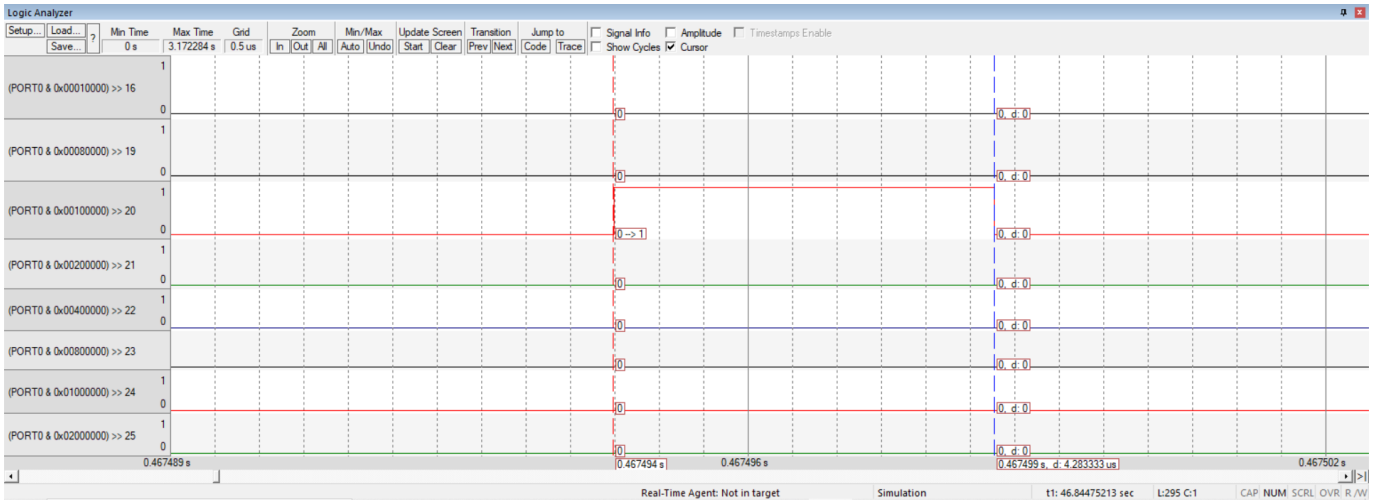
Because hyperperiod equals 100 milliseconds and a task (Button 1 Monitor), It is carried out every 50 milliseconds, so it is carried out twice in one hyperperiod.

Task_2 --> Button_2_Monitor

- Deadline = 50ms



- Execution Time of Task (Button_2_Monitor) = 5 microsecond

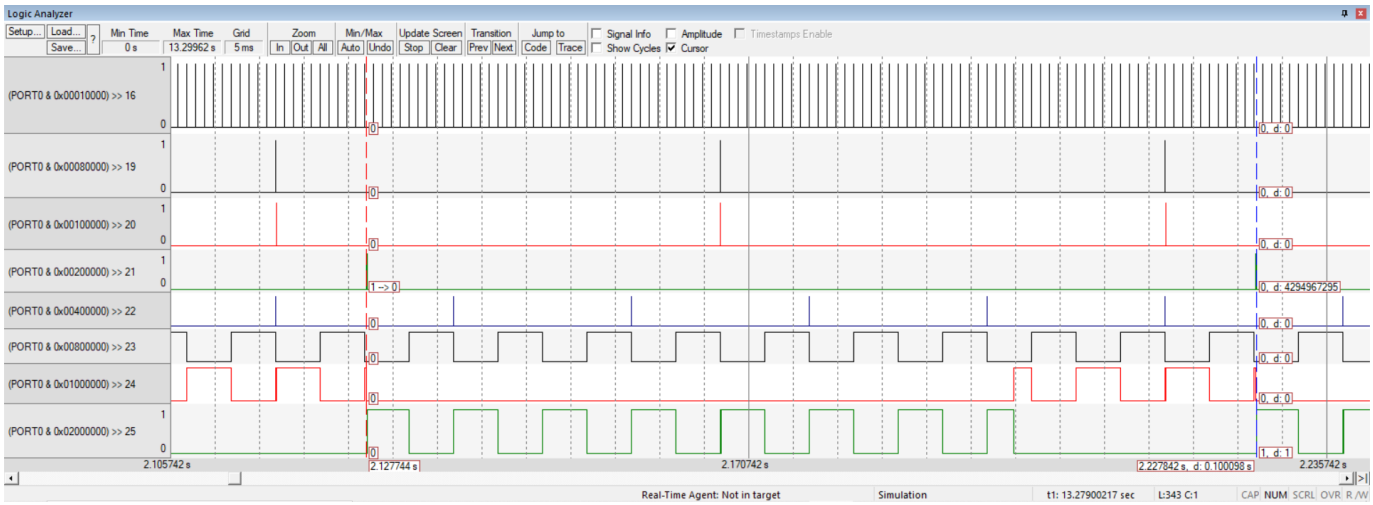


- occurrence during hyperperiod = 2

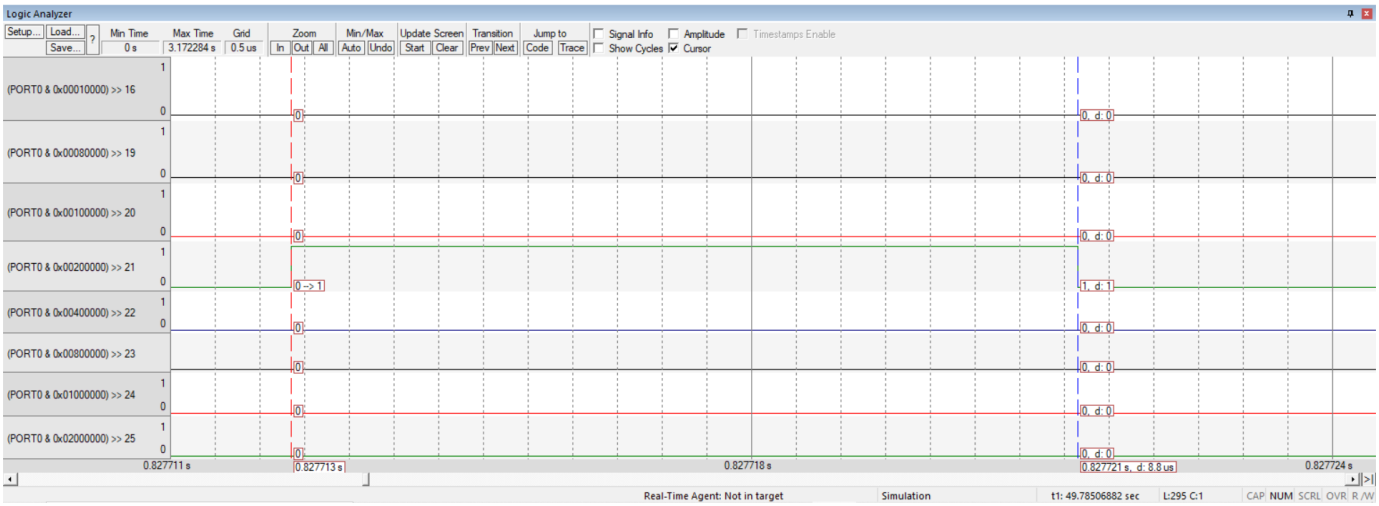
Because hyperperiod equals 100 milliseconds and a task (Button 2 Monitor), It is carried out every 50 milliseconds, so it is carried out twice in one hyperperiod.

Task_3 --> PeriodicTransmitter

- Deadline = 100ms



- Execution Time of Task (PeriodicTransmitter) = 8.8 microsecond

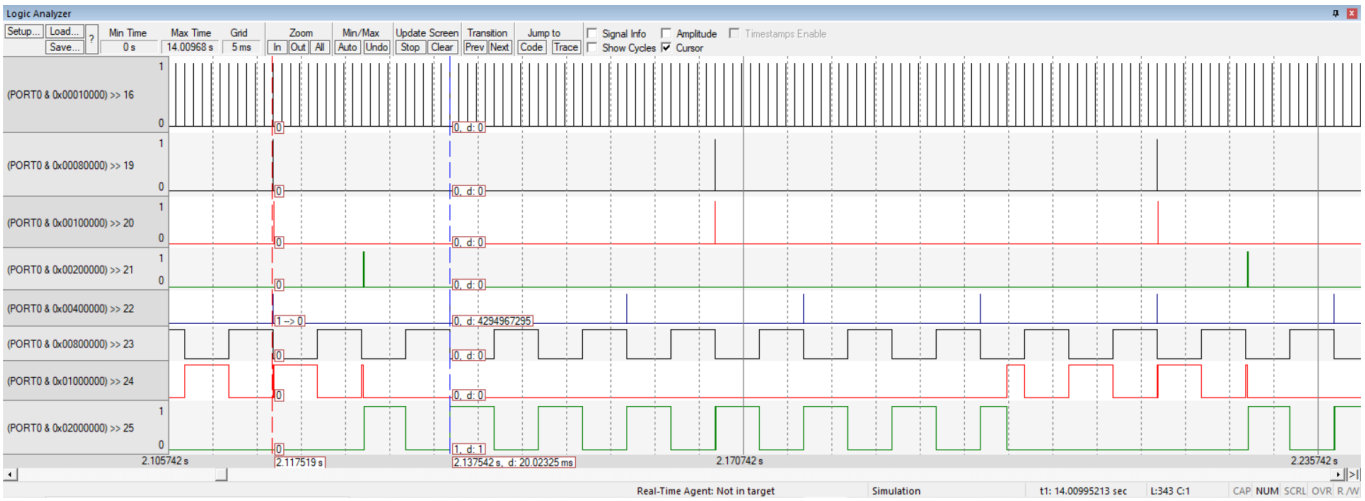


- occurrence during hyperperiod = 1

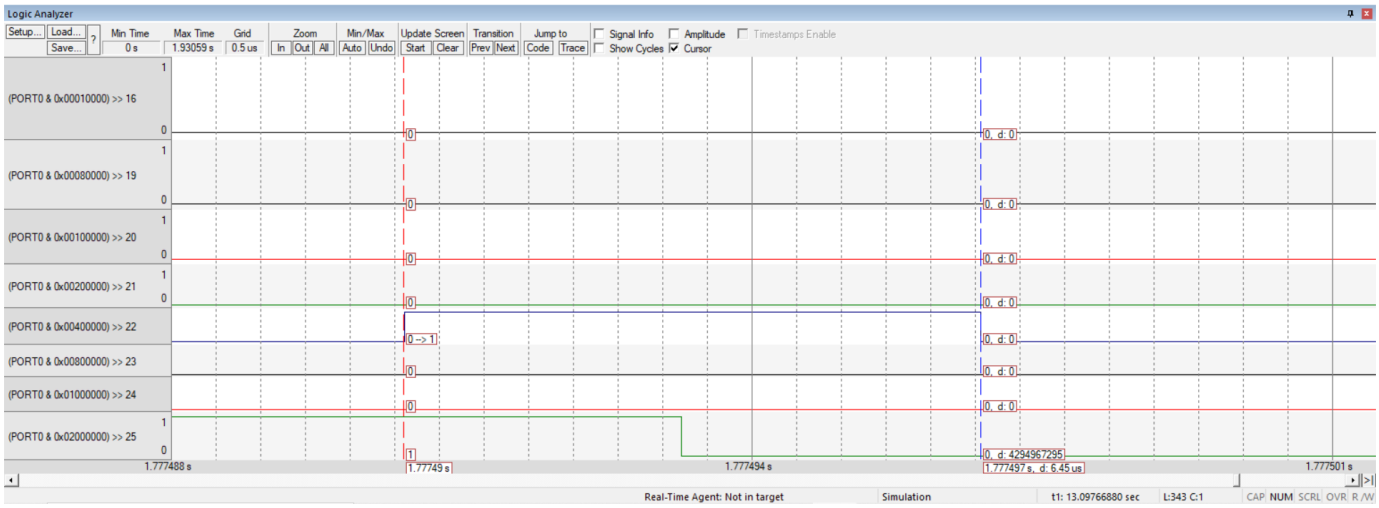
Because hyperperiod equals 100 milliseconds and a task (PeriodicTransmitter), It is carried out every 100 milliseconds, so it is carried out once in one hyperperiod.

Task_4 --> UartReceiver

- Deadline = 20ms



- Execution Time of Task (UartReceiver) = 6.45 microsecond

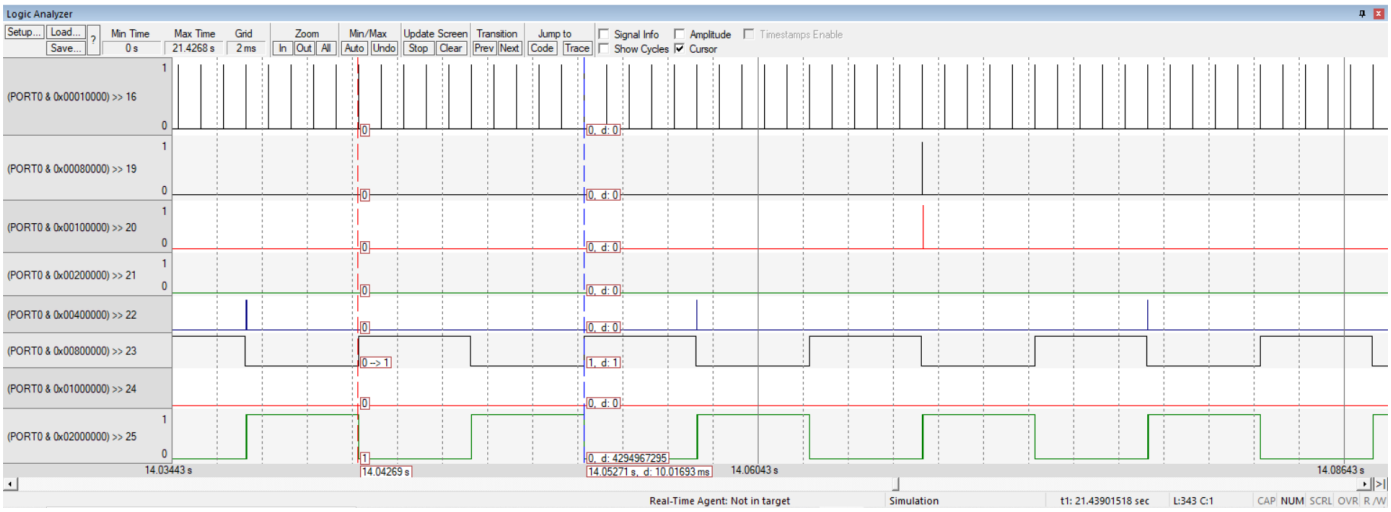


- occurrence during hyperperiod = 5

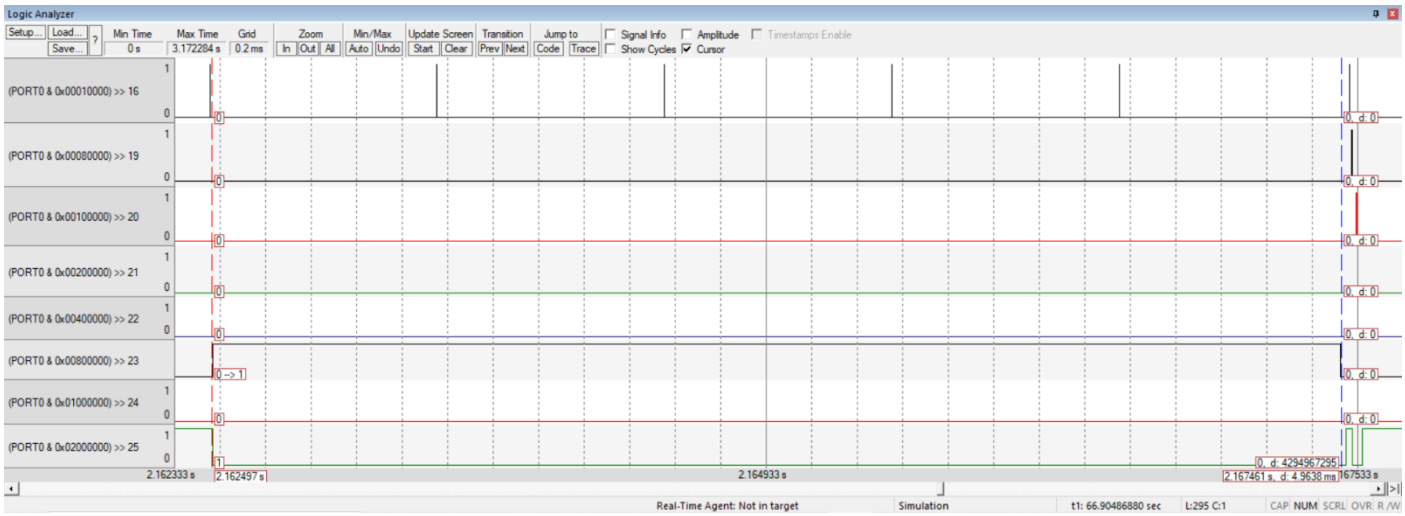
Because hyperperiod equals 100 milliseconds and a task (UartReceiver), It is carried out every 20 milliseconds, so it is carried out Five Times in one hyperperiod.

Task_5 --> LoadSimulation_1

- Deadline = 10ms



- Execution Time of Task (LoadSimulation_1) = 5 milliseconds

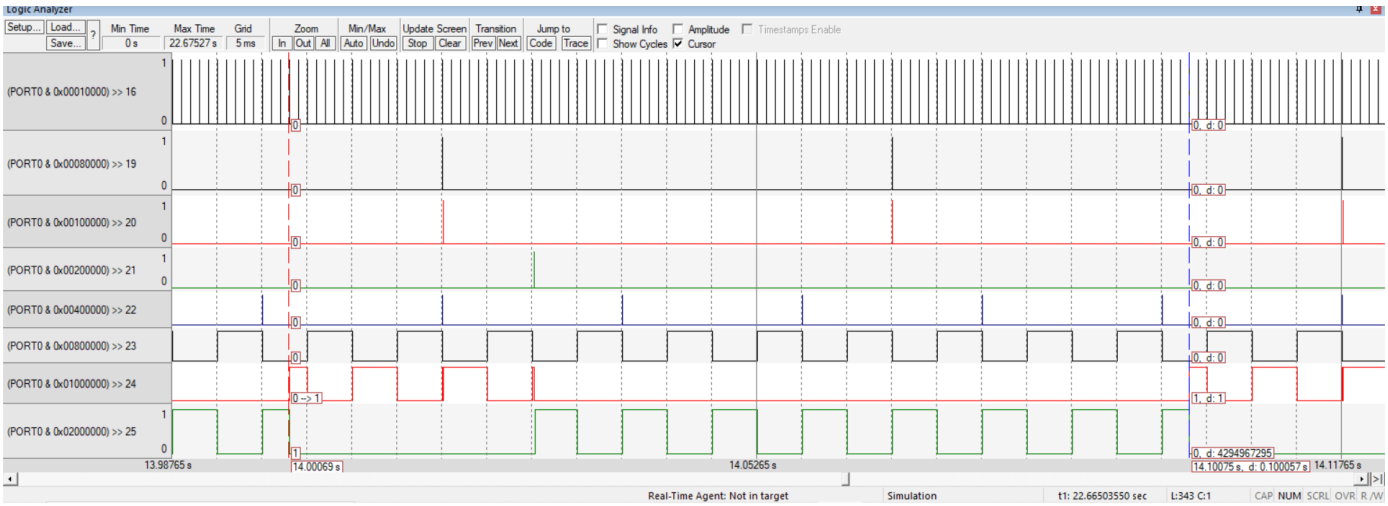


- occurrence during hyperperiod = 10

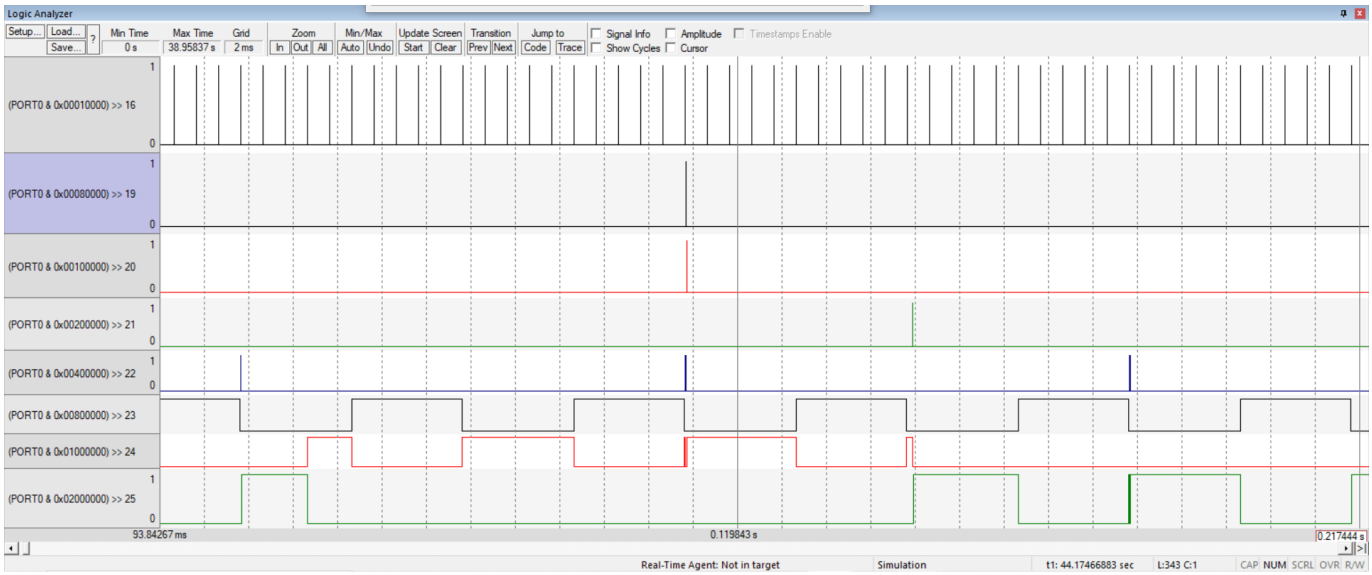
Because hyperperiod equals 100 milliseconds and a task (LoadSimulation_1), It is carried out every 10 milliseconds, so it is carried out Ten Times in one hyperperiod.

Task_6 --> LoadSimulation_2

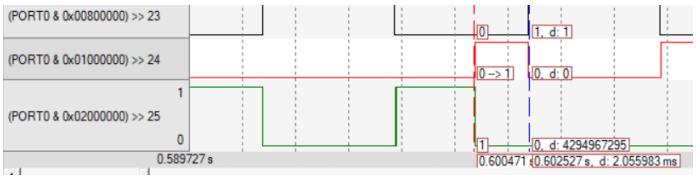
- Deadline = 100ms



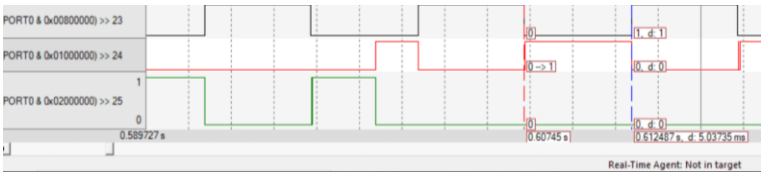
- Execution Time of Task (LoadSimulation_2) = 5 milliseconds



First Execution Time = 2.05ms

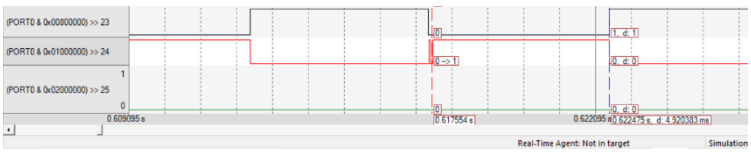


Second Execution Time = 5.03ms



Third Execution Time = 29.75us

Fourth Execution Time = 4.9ms



So, the Total Execution Time of task (LoadSimulation_2) =

First Execution Time +
Second Execution Time +
Third Execution Time +
Fourth Execution Time = 2.05 + 5.03 + 0.02975 + 4.9 = 12.009ms

- This process occurs when set these macros (configUSE_PREEMPTION) and (configUSE_TIME_SLICING) It works on the ability of other tasks to stop them to complete their tasks at the beginning of each tick.

• occurrence during hyperperiod = 1

Because hyperperiod equals 100 milliseconds and a task (LoadSimulation_2), It is carried out every 100 milliseconds, so it is carried out once in one hyperperiod.