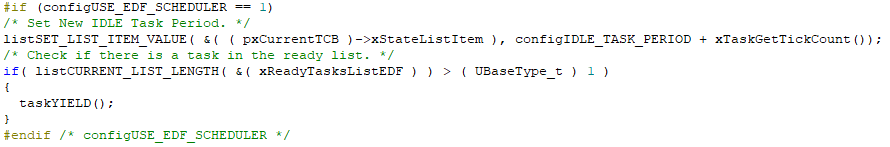
Implementation of EDF scheduler in FreeRTOS

# Changes in Code

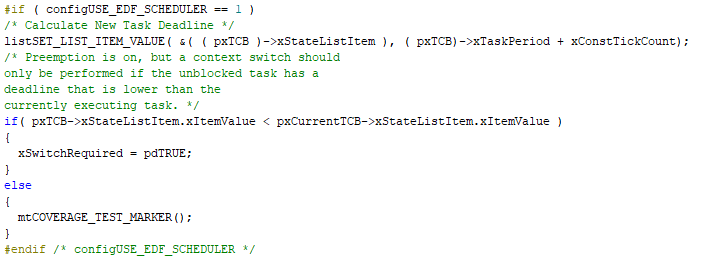
## prvIdleTask

****

In the prvIdleTask Api:

* Set the new idle task deadline.
* Yield the idle task in case there is a task in the ready list.

## xTaskIncrementTick



In the xTaskIncrementTick Api:

* Calculate the new task deadline.
* Set a context switch is required if the unblocked task has a lower deadline than the current executing task.

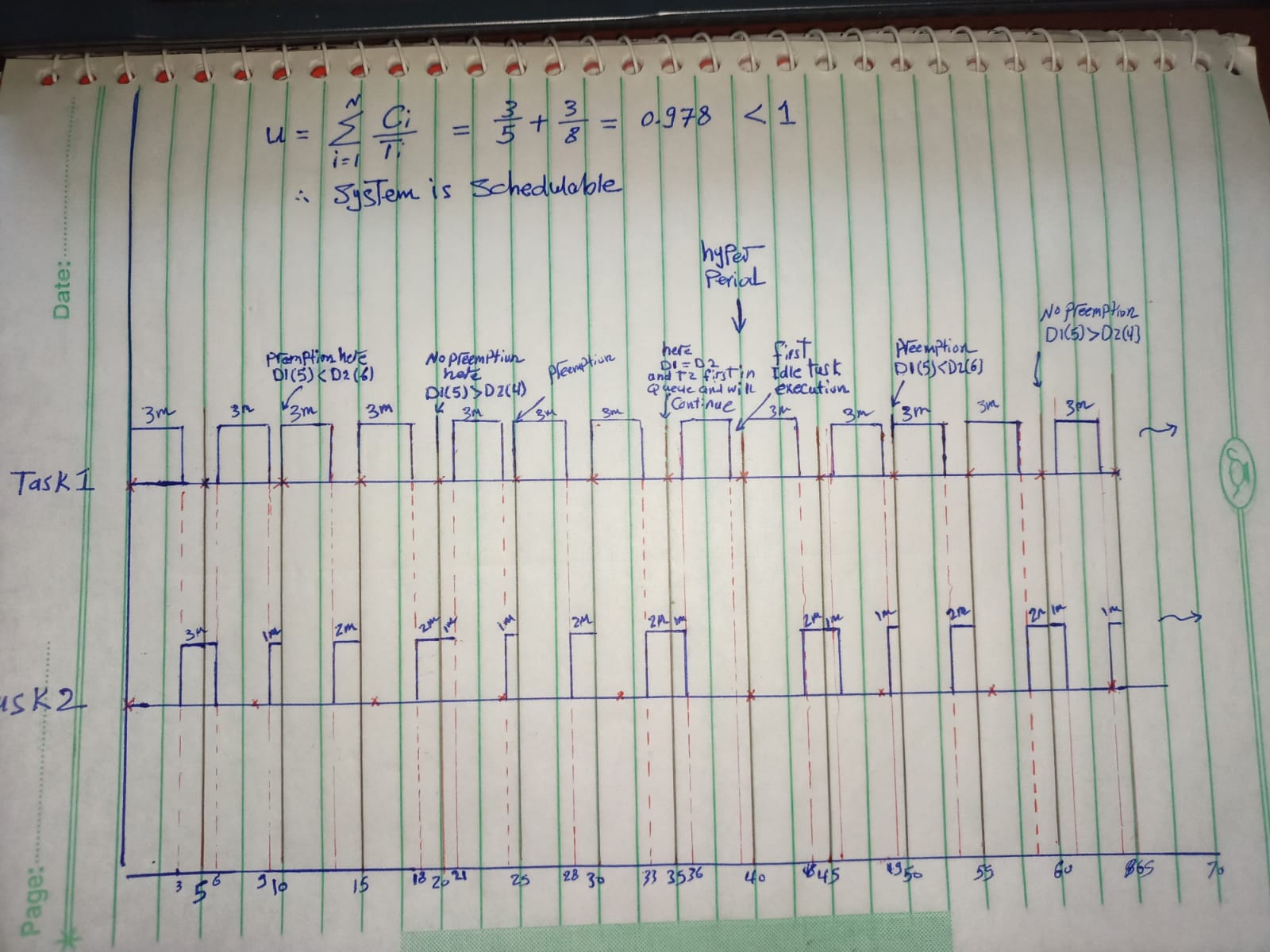
# Task Set

We suggested a task set of two tasks,

* Task 1 with the following parameters {Periodicity: 5, Priority: 2, ET: 3, Deadline: 5}
* Task 2 with the following parameters{Periodicity: 8, Priority: 1, ET: 3, Deadline: 8}

# Verifying the system

## Analytical Methods

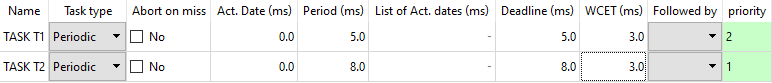


Using the analytical method:

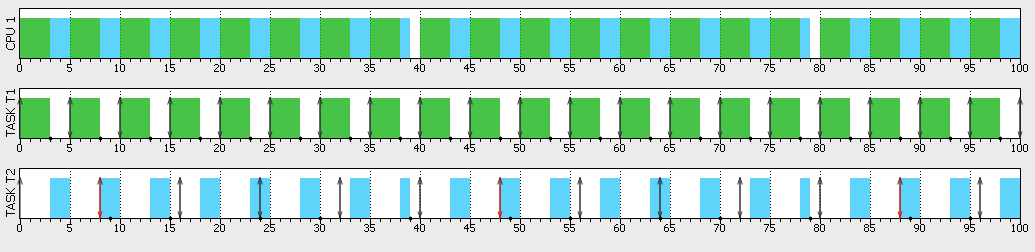
* Verified that the system is schedulable
* Determine the hyperperiod
* Have a good estimate how the system will behave

## Simulator

### Task set

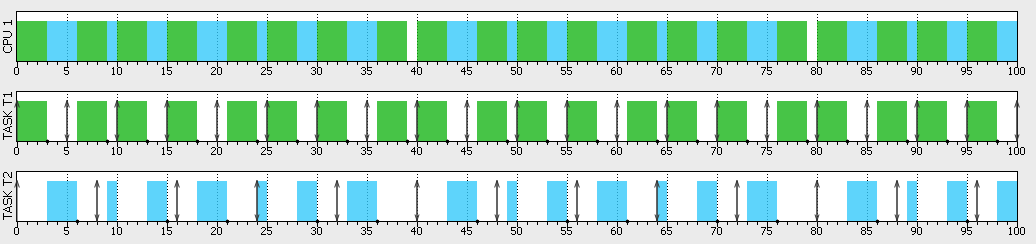


### FP Scheduler



Task 2 Misses its deadline on using a FP scheduler.

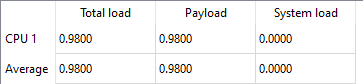
### EDF Scheduler



We verified from the simulation that:

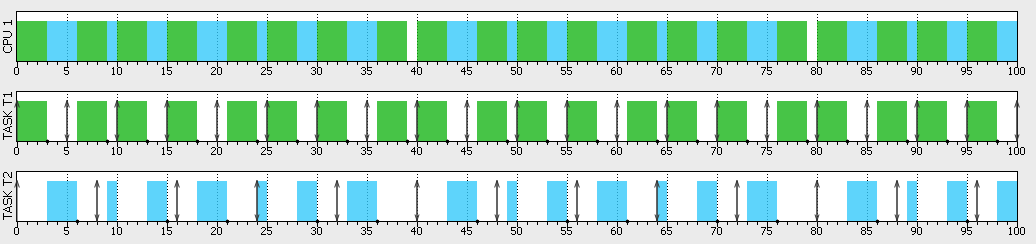
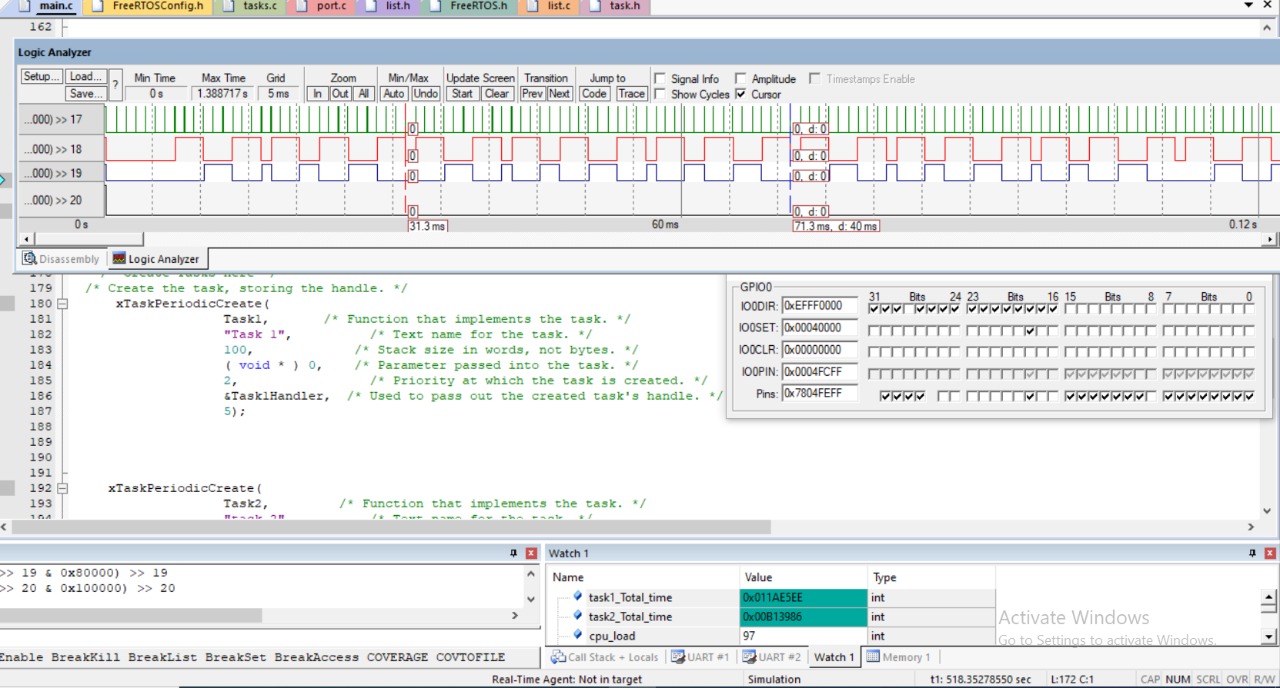
* Task 2 doesn’t miss its deadline on using an EDF scheduler.
* The graph is similar to the graph drawn by hand.

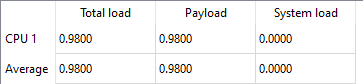
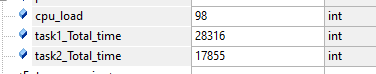
### CPU Load



* CPU Load was the same for the two scheduler
* The system is overloaded, but this is only to demonstrate the difference in missing deadline between the two schedulers

## Runtime Analysis



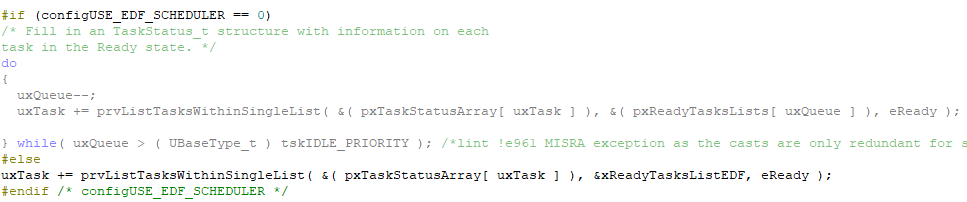


By using system hooks to draw tasks working time and calculate CPU load we can verify that:

* The task graph is similar to the analytical method and simulator.
* The hyperperiod is the same as the analytical method and simulator (40ms).
* The CPU Load is the same as in the simulation.

# Bonus

* This is the suggested change in uxTaskGetSystemState Api to enable runtime stats of the FreeRTOS with the EDF scheduler.



We also verified that the cpu out is the same as the previous method.

