EgFWD
Advanced Embedded Systems
Project (2) | RTOS
EDF Scheduler Report

1. Using analytical methods calculate the following for the given set of tasks:

- System Hyperperiod = LCM(10, 20, 50, 100) = 100 milliseconds

- Calculate the CPU load (Total execution time of all tasks during hyperperiod / hyperperiod)
  - For only four tasks = (0.28*2 + 0.28*2 + 0.285*1 + 0.003*5) /100 = 0.0142 (1.42%)
  - After adding heavier loads = (0.28*2 + 0.28*2 + 0.285*1 + 0.003*5 + 5*10 + 12*1) /100 = 0.6342 (63.42%)

Comment: After adding 2 heavier loads, the system became overloaded and unschedulable as tasks will definitely start missing their deadline.

- Check system schedualability using URM (Assuming the given set of tasks are scheduled using a fixed priority rate -monotonic scheduler)
  - For only four tasks:
    - U = (0.28/50 + 0.28/50 + 0.285/100 + 0.003/20) = 0.0142
    - Urm = n*(2^(1/n) -1) = 4*(2^(1/4) – 1) = 0.7568
    - U<Urm
    - Comment: It is guaranteed the system is schedualable for a fixed priority rate-monotonic scheduler.
  - After adding heavier loads:
    - U = (0.28/50 + 0.28/50 + 0.285/100 + 0.003/20) + 5/10 + 12/100) = 0.6342
    - Urm = n*(2^(1/n) -1) = 6*(2^(1/6) – 1) = 0.7348
    - U<Urm
    - Comment: It is guaranteed the system is schedualable for a fixed priority rate-monotonic scheduler.

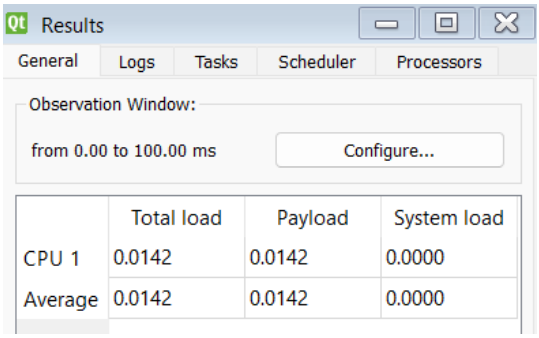
- Check system schedualability using Time Demand Analysis technique (Assuming the given set of tasks are scheduled using a fixed priority rate -monotonic scheduler)
  - For only four tasks:
    - Tasks' priority are in this order: Uart_Receiver, Button1_Monitor, Button2_Monitor, Periodic_Transmitter
    - Tasks' Calculations are as follows:
      - Uart_Receiver: W(20) = 0.003 + 0 = 0.003<20; therefore, schedulable.
      - Button1_Monitor: W(50) = 0.28 + ((50/20)*0.003) = 17.00252<50; therefore, schedulable.
      - Button2_Monitor: W(50) = 0.28 + ((50/50)*0.28) + ((50/20)*0.003) = 0.0859<50; therefore, schedulable.

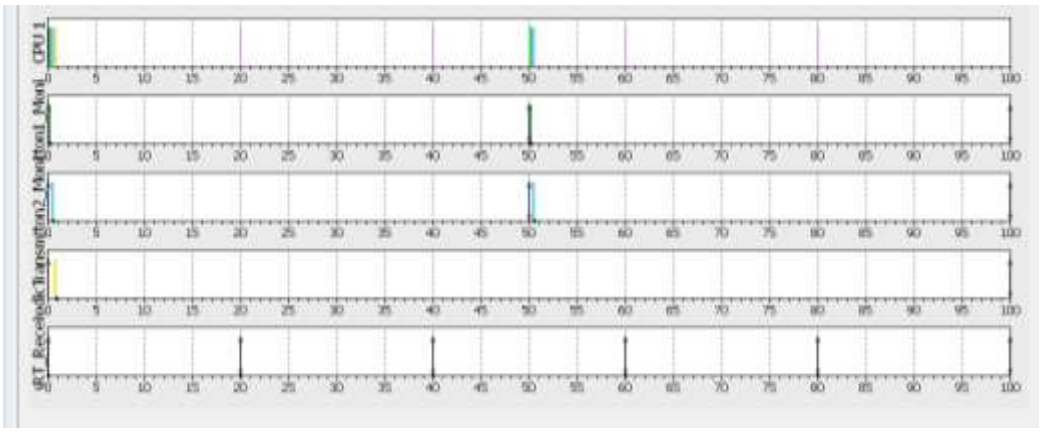
        - Periodic_Transmitter: W(100) = 0.285 + ((100/50)*0.28) + ((100/50)*0.28) + ((100/20)*0.003) = 1.42<100; therefore, schedulable.
    - Comment: The system is schedulable with the previous four tasks as no task misses its deadline. All calculations are taken at the instance of task deadline as it's the critical point at which if time needed is less than time provided then the task can be scheduled.

- After adding heavier loads:
    - Tasks' priority are in this order: Load1_Simulation, Uart_Receiver, Button1_Monitor, Button2_Monitor, Periodic_Transmitter, Load2_Simulation
    - Tasks' Calculations are as follows:
        - Load1_Simulation: W(10) = 5 + 0 = 5<10; therefore, schedulable.
        - Uart_Receiver: W(20) = 0.003 + ((20/10)*5) = 10.003<20; therefore, schedulable.
        - Button1_Monitor: W(50) = 0.28 + ((50/20)*0.003) = 0.2875<50; therefore, schedulable.
        - Button2_Monitor: W(50) = 0.28 + ((50/50)*0.28) + ((50/20)*0.003) + ((50/10)*5) = 25.5675<50; therefore, schedulable.
        - Periodic_Transmitter: W(100) = 0.285 + ((100/50)*0.28) + ((100/50)*0.28) + ((100/20)*0.003) + ((100/10)*5) = 51.42<100; therefore, schedulable.
        - Load2_Simulation: W(100) = 12 + ((100/100)*0.285) + ((100/50)*0.28) + ((100/50)*0.28) + ((100/20)*0.003) + ((100/10)*5) =63.42 <100; therefore, schedulable.
    - Comment: The system is schedulable with the previous six tasks as no task misses its deadline. All calculations are taken at the instance of task deadline as it's the critical point at which if time needed is less than time provided then the task can be scheduled.

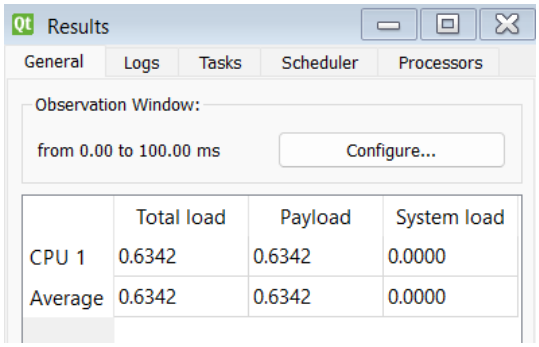
2. Using Simso offline simulator, simulate the given set of tasks assuming fixed priority rate monotonic scheduler

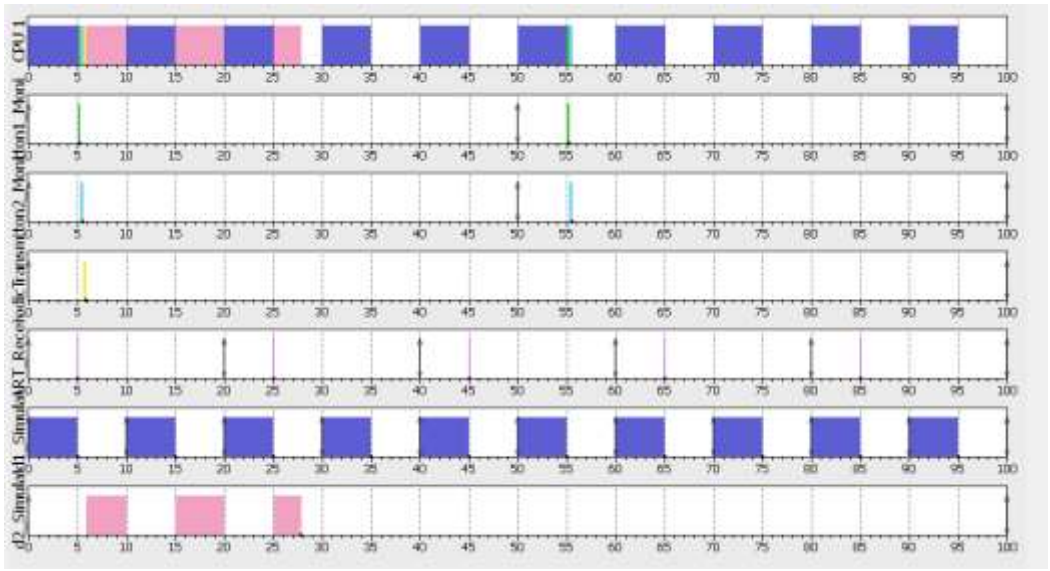
- For only four tasks:





- After adding heavier loads:

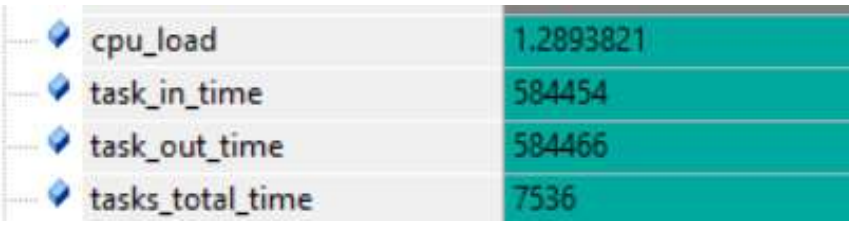




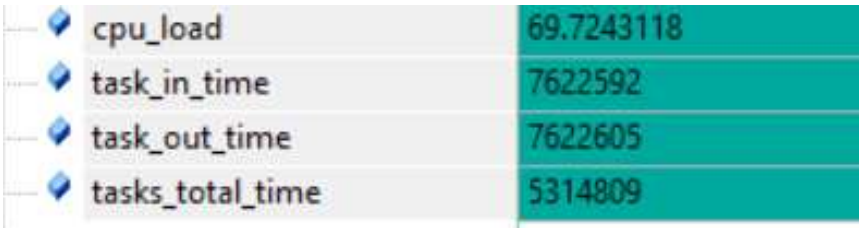
Comment: Analytical method and Simso offline simulator both got the same results, 1.42% when only four tasks are in the system and 63.42 after adding 2 heavier loads simulated as empty for loops for 5 and 12 milliseconds.

3. Using Keil simulator in run-time and the given set of tasks:

- Calculate the CPU usage time using timer 1 and trace macros
    - For only four tasks: cpu_load = 86%

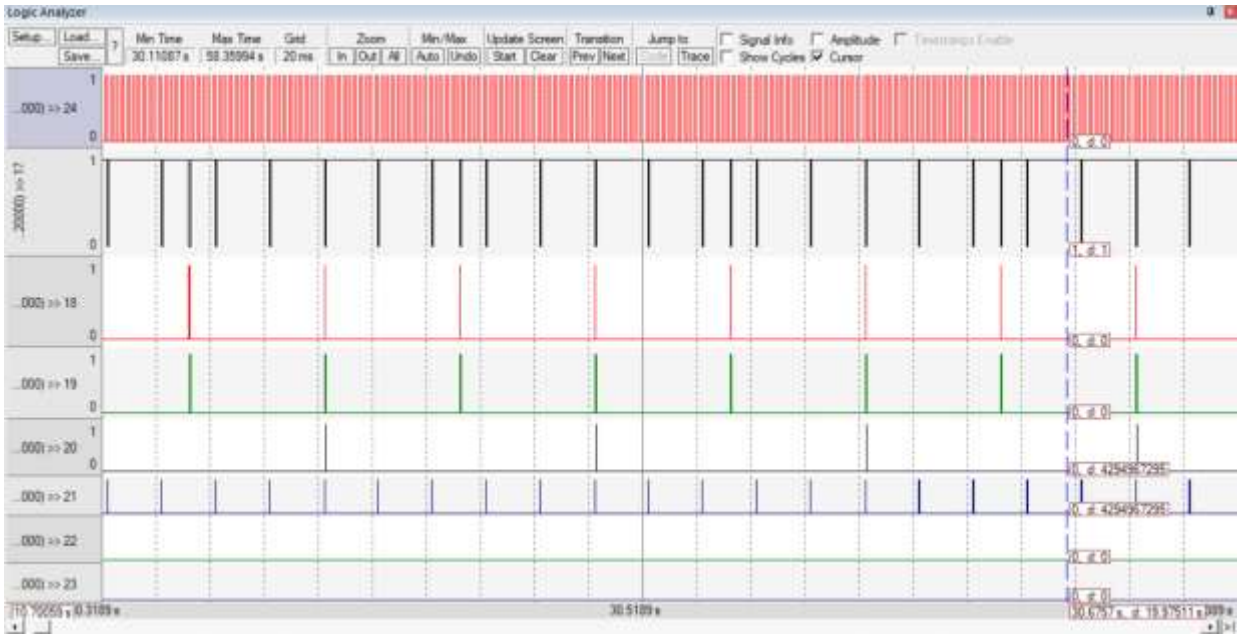


    - After adding heavier loads: cpu_load = 69.72% (overloaded)



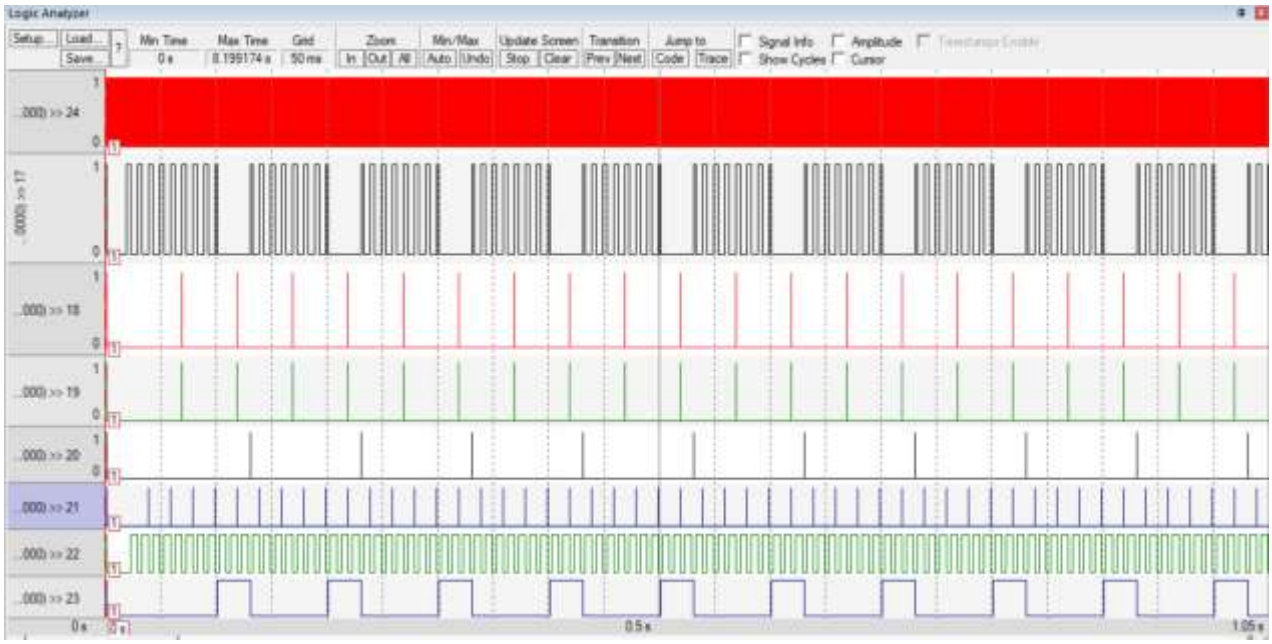
Comment: Almost same CPU Load numbers from analytical method and Simso Offline Simulator.

- Using trace macros and GPIOs, plot the execution of all tasks, tick, and the idle task on the logic analyzer:
    - For only four tasks: (Tick Hook, Idle Task, Button1-Monitor, Button2-Monitor, Periodic-Transmitter, Uart-Receiver, Load1-Simulaion and Load2-Simulation respectively)



Comment: System starts with scheduling Uart_Receiver task since it has the nearest deadline at 20 milliseconds then Button1_Monitor and Button2_Monitor at the same deadline of 50 milliseconds but since Button1_Monitor was pushed first to the EDF_ReadyList so it got scheduled first. Then finally ending by Periodic_Transmitter Task at the farthest deadline which is 100 milliseconds and then repeating all over again throughout the timeline. It's also observed that the Tick Hook is so cloudy as it comes every 1 millisecond.

- After adding heavier loads: (Tick Hook, Idle Task, Button1-Monitor, Button2-Monitor, Periodic-Transmitter, Uart-Receiver, Load1-Simulaion and Load2-Simulation respectively)



Comment: Now the system starts with Load1_Simulation as it has the nearest deadline at 10 milliseconds then Uart_Receiver at a deadline of 20 milliseconds and so on according to nearest deadline. Tick hook is still so cloudy as it comes every 1ms.