EDF SCHEDULER IMPLEMENTATION IN FREERTOS

egFWD Embedded Systems Advanced Track

DECEMBER 12, 2022

Hassan Abd El-Kareem Kassem Mostafa

I. Introduction:

Earliest Deadline First (EDF) CPU scheduling algorithm

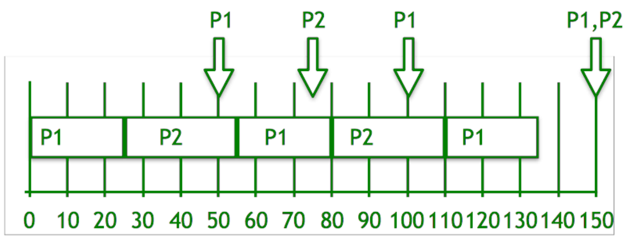
Earliest Deadline First (EDF) is an optimal dynamic priority scheduling algorithm used in real-time systems. It can be used for both static and dynamic real-time scheduling.

EDF uses priorities to the jobs for scheduling. It assigns priorities to the task according to the absolute deadline. The task whose deadline is closest gets the highest priority. The priorities are assigned and changed in a dynamic fashion. EDF is very efficient as compared to other scheduling algorithms in real-time systems. It can make the CPU utilization to about 100% while still guaranteeing the deadlines of all the tasks.

EDF includes the kernel overload. In EDF, if the CPU usage is less than 100%, then it means that all the tasks have met the deadline. EDF finds an optimal feasible schedule. The feasible schedule is one in which all the tasks in the system are executed within the deadline. If EDF is not able to find a feasible schedule for all the tasks in the real-time system, then it means that no other task scheduling algorithms in real-time systems can give a feasible schedule. All the tasks which are ready for execution should announce their deadline to EDF when the task becomes runnable.

EDF scheduling algorithm does not need the tasks or processes to be periodic and also the tasks or processes require a fixed CPU burst time. In EDF, any executing task can be preempted if any other periodic instance with an earlier deadline is ready for execution and becomes active. Preemption is allowed in the Earliest Deadline First scheduling algorithm.

Example:
Consider two processes P1 and P2.
Let the period of P1 be $p_1 = 50$
Let the processing time of P1 be $t_1 = 25$ Let the period of P2 be $\text{period}_2 = 75$
Let the processing time of P2 be $t_2 = 30$



Steps for solution:

1. Deadline pf P1 is earlier, so priority of P1>P2.
2. Initially P1 runs and completes its execution of 25 time.
3. After 25 times, P2 starts to execute until 50 times, when P1 is able to execute.
4. Now, comparing the deadline of (P1, P2) = (100, 75), P2 continues to execute.
5. P2 completes its processing at time 55.
6. P1 starts to execute until time 75, when P2 is able to execute.
7. Now, again comparing the deadline of (P1, P2) = (100, 150), P1 continues to execute.
8. Repeat the above steps…
9. Finally at time 150, both P1 and P2 have the same deadline, so P2 will continue to execute till its processing time after which P1 starts to execute.

II. Our System:

N	Task	Period (P)	Deadline (D)	Execution Time
1	Button_1_Monitor	50 ms		20 us
2	Button_2_Monitor	50 ms		20 us
3	Periodic_Transmitter	100 ms		20 us
4	UART Receiver	20 ms		24 us
5	Load_1_Simulation	10 ms		5 ms
6	Load_2_Simulation	100 ms		12 ms

Our system contains **6 tasks** with system tick of **10 ms** and hyper-period of **100 ms.**

III. CPU Load:

CPU Load = [(20×2)+(20×2)+(20×1)+(24×5)+(5000×10)+(12000×1) / 100000] * 100 = **62.22 %**

Max CPU load occurs when the two buttons are always toggling (so strings continuously being sent to consumer task).

IV. Checking system schedulability using Time Demand analysis:

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

Times in us.

1. **Button_1_Monitor:**

W(50000) = 20 + (50000/20000)*24 + (50000/10000)*5000 = **30092**
W(50000) < 50000, **The task is schedulable.**

2. **Button_2_Monitor:**
W(50000) = 20 + (50000/50000)*20 + (50000/20000)*24 + (50000/10000)*5000 = **30132**
W(50000) < 50000, **The task is schedulable.**

3. **Periodic_Transmitter:**

W(100000) = 20 + (100000/50000)*20 + (100000/50000)*20 + (100000/20000)*24 + (100000/10000)*5000 = **55284**

W(100000) < 100000, **The task is schedulable.**

4. <u>**UART_Receiver:**</u>

W(20000) = 24 + (20000/10000)*5000 = **15024**
W(20000) < 20000, **The task is schedulable.**

5. <u>**Load_1_Simulation:**</u>

W(10000) = **5000**
W(10000) < 10000, **The task is schedulable.**

6. <u>**Load_2_Simulation:**</u>

W(100000) = 12000 + (100000/100000)*20 + (100000/50000)*20 + (100000/50000)*20 + (100000/20000)*24 + (100000/10000)*5000 = **67304**
W(100000) < 100000, **The task is schedulable.**

From previous results: <u>the system is schedulable.</u>

V. Checking system schedulability using URM analysis:

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

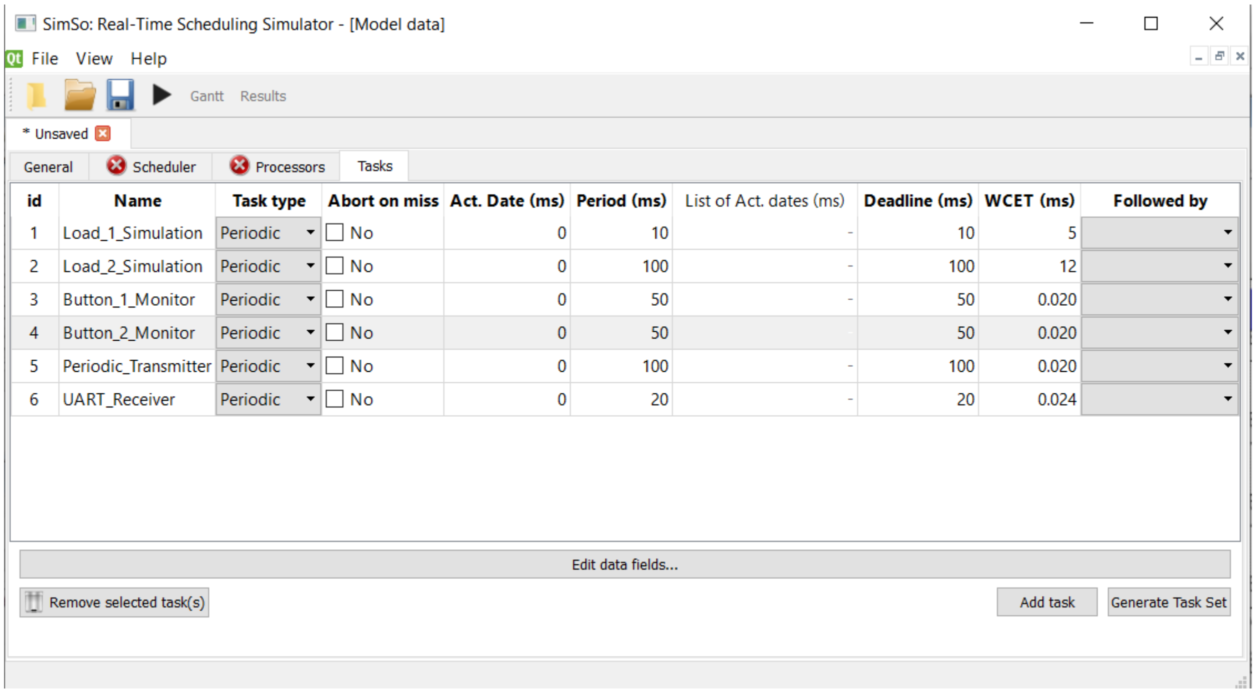
6* (2^(1/6) − 1) = $\mathbf{0.734}$

U = 20/50000 + 20/50000 + 20/100000 + 24/20000 + 5000/10000 + 12000/100000 = $\mathbf{0.622 < 0.734}$

From previous results: <u>the system is schedulable.</u>

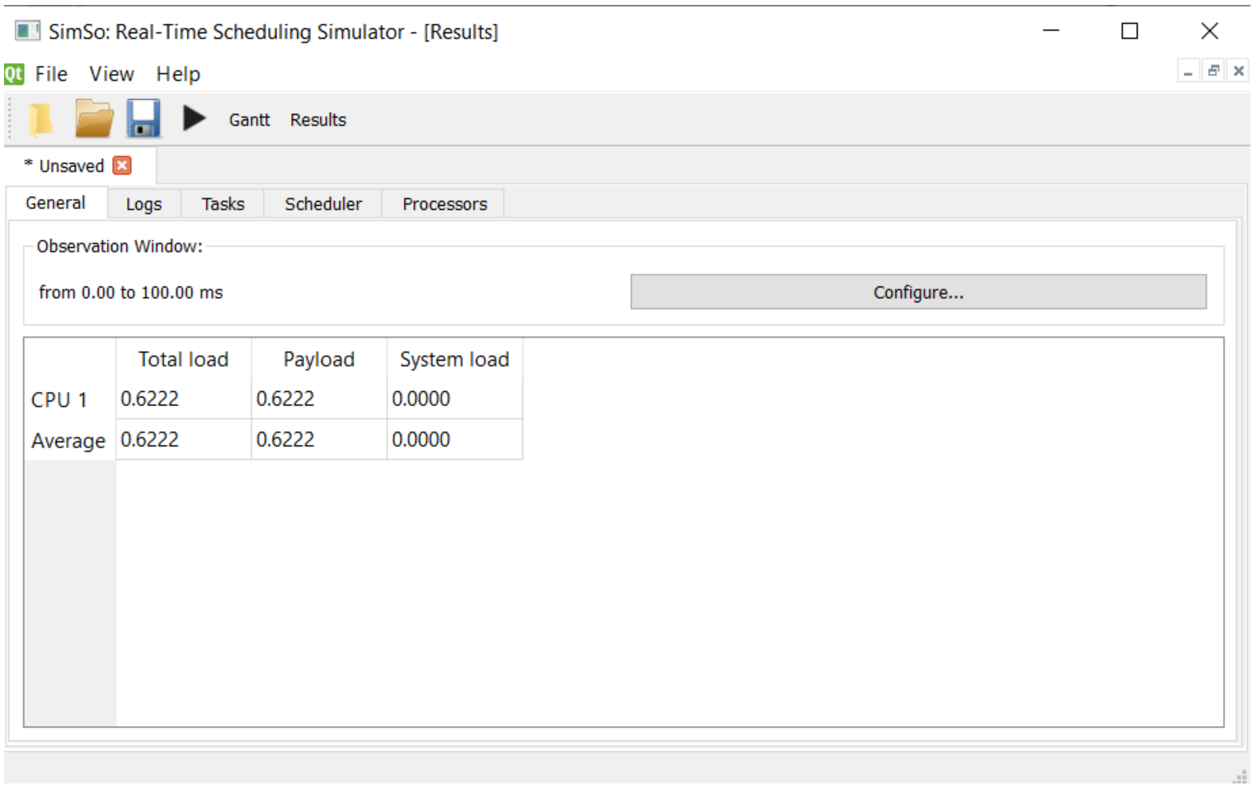
VI. Simulation in SIMSO:

1. Tasks



id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by
1	Load_1_Simulation	Periodic	No	0	10	-	10	5	
2	Load_2_Simulation	Periodic	No	0	100	-	100	12	
3	Button_1_Monitor	Periodic	No	0	50	-	50	0.020	
4	Button_2_Monitor	Periodic	No	0	50	-	50	0.020	
5	Periodic_Transmitter	Periodic	No	0	100	-	100	0.020	
6	UART_Receiver	Periodic	No	0	20	-	20	0.024	

2. CPU Load

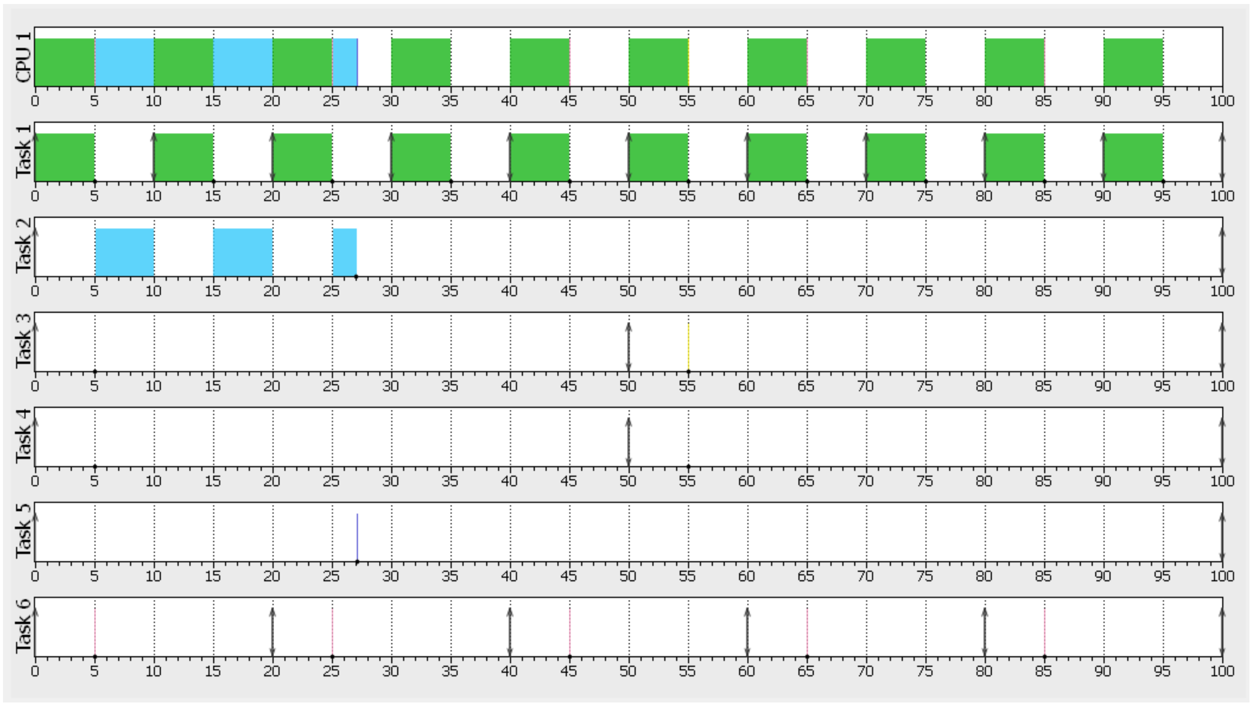

SimSo: Real-Time Scheduling Simulator - [Results]
File View Help
Gantt Results
* Unsaved
General Logs Tasks Scheduler Processors
Configure...

Observation Window:

from 0.00 to 100.00 ms

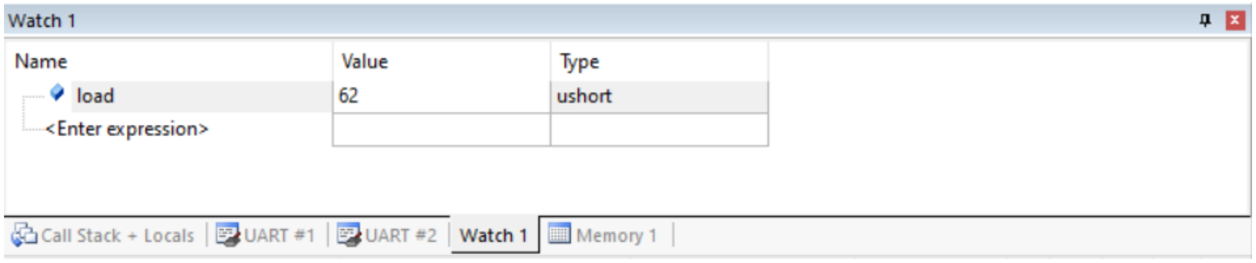
	Total load	Payload	System load
CPU 1	0.6222	0.6222	0.0000
Average	0.6222	0.6222	0.0000

3. Grantt Chart



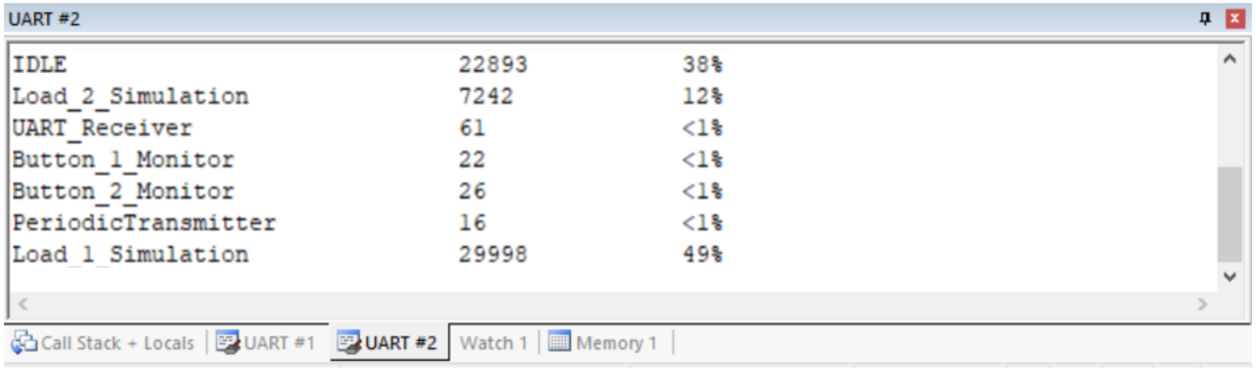
VII. Simulation in KEIL:

1. CPU load



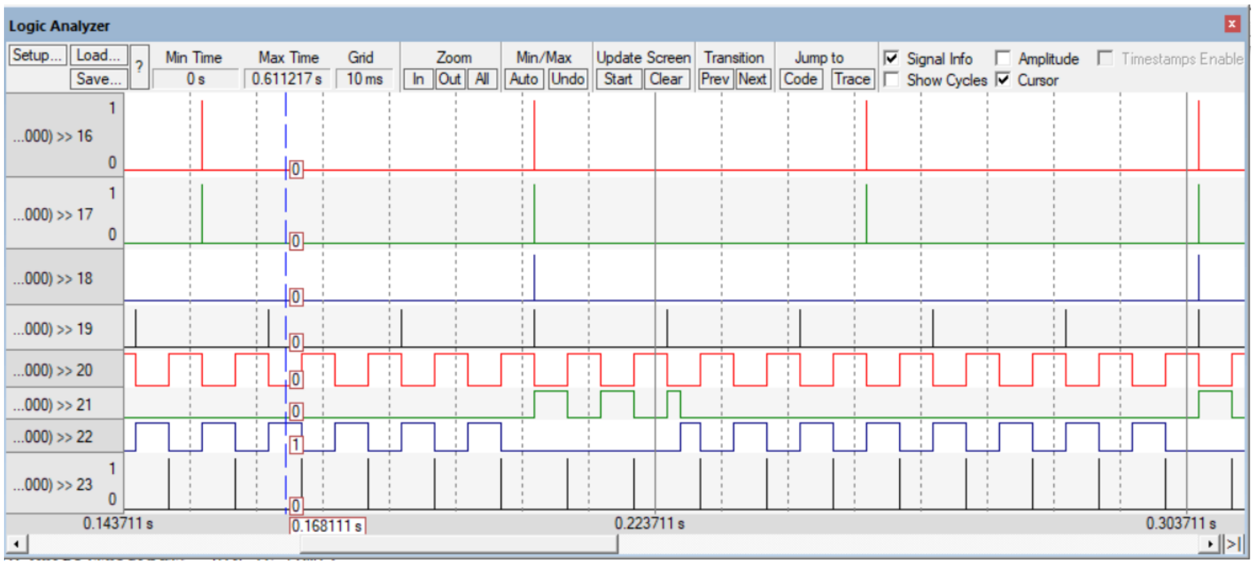
Name	Value	Type
load	62	ushort
<Enter expression>		

2. System runtime stats



```
IDLE                              22893           38%
Load_2_Simulation                 7242            12%
UART_Receiver                     61              <1%
Button_1_Monitor                  22              <1%
Button_2_Monitor                  26              <1%
PeriodicTransmitter               16              <1%
Load_1_Simulation                 29998           49%
```

3. Logic Analyzer



VIII. Comments:

1. System is schedulable (all tasks meet their deadlines) .
2. Earliest deadline tasks are executed first at each hyper-period.
3. Earliest deadline tasks preempt latest deadline tasks.
   [Load_1_Simulation task preempt Load_2_Simulation task every time Load 2 is available (Unblocked)].
4. Idle task gets preempted if there is a task in ready state, other than that, Idle task is being executed.

From previous: This is a valid implementation of EDF scheduler.