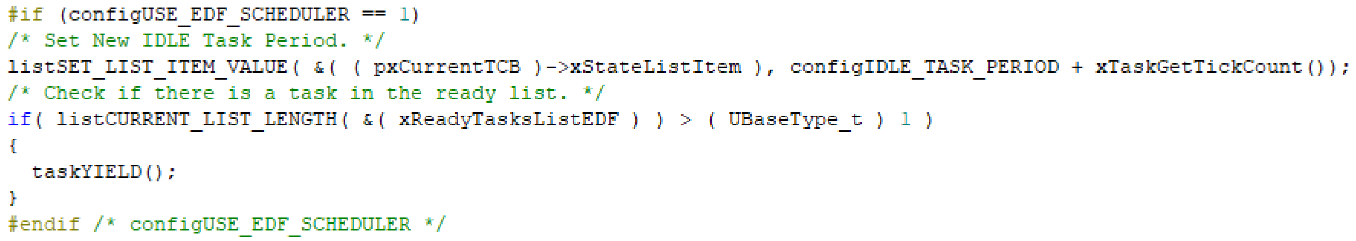
Implementation of EDF scheduler in FreeRTOS

Changes in Code

prvIdleTask

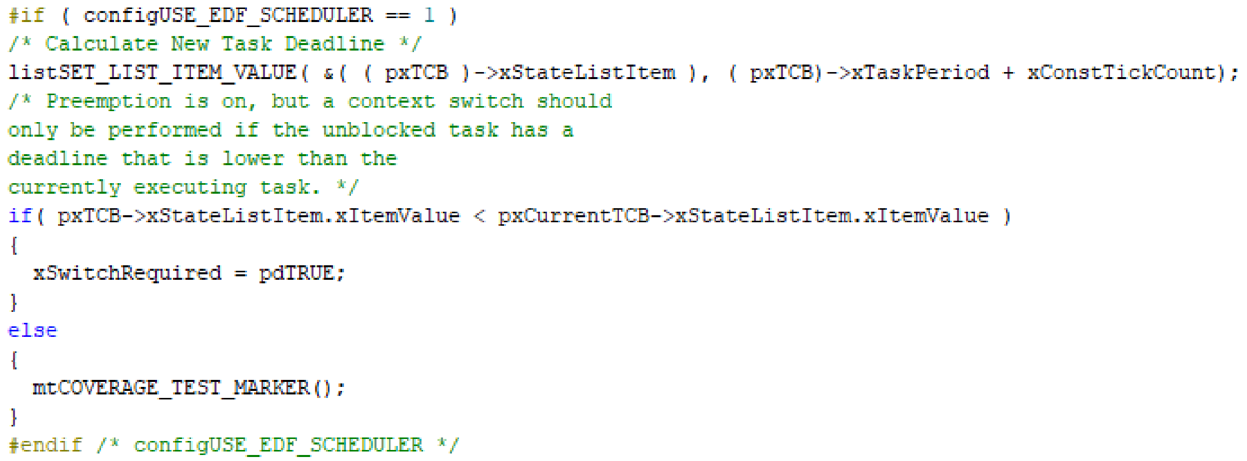


```
#if (configUSE_EDF_SCHEDULER == 1)
/* Set New IDLE Task Period. */
listSET_LIST_ITEM_VALUE( &( ( pxCurrentTCB )->xStateListItem ), configIDLE_TASK_PERIOD + xTaskGetTickCount());
/* Check if there is a task in the ready list. */
if( listCURRENT_LIST_LENGTH( &( xReadyTasksListEDF ) ) > ( UBaseType_t ) 1 )
{
  taskYIELD();
}
#endif /* configUSE_EDF_SCHEDULER */
```

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



```
#if ( configUSE_EDF_SCHEDULER == 1 )
/* Calculate New Task Deadline */
listSET_LIST_ITEM_VALUE( &( ( pxTCB )->xStateListItem ), ( pxTCB)->xTaskPeriod + xConstTickCount);
/* Preemption is on, but a context switch should
only be performed if the unblocked task has a
deadline that is lower than the
currently executing task. */
if( pxTCB->xStateListItem.xItemValue < pxCurrentTCB->xStateListItem.xItemValue )
{
  xSwitchRequired = pdTRUE;
}
else
{
  mtCOVERAGE_TEST_MARKER();
}
#endif /* configUSE_EDF_SCHEDULER */
```

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.

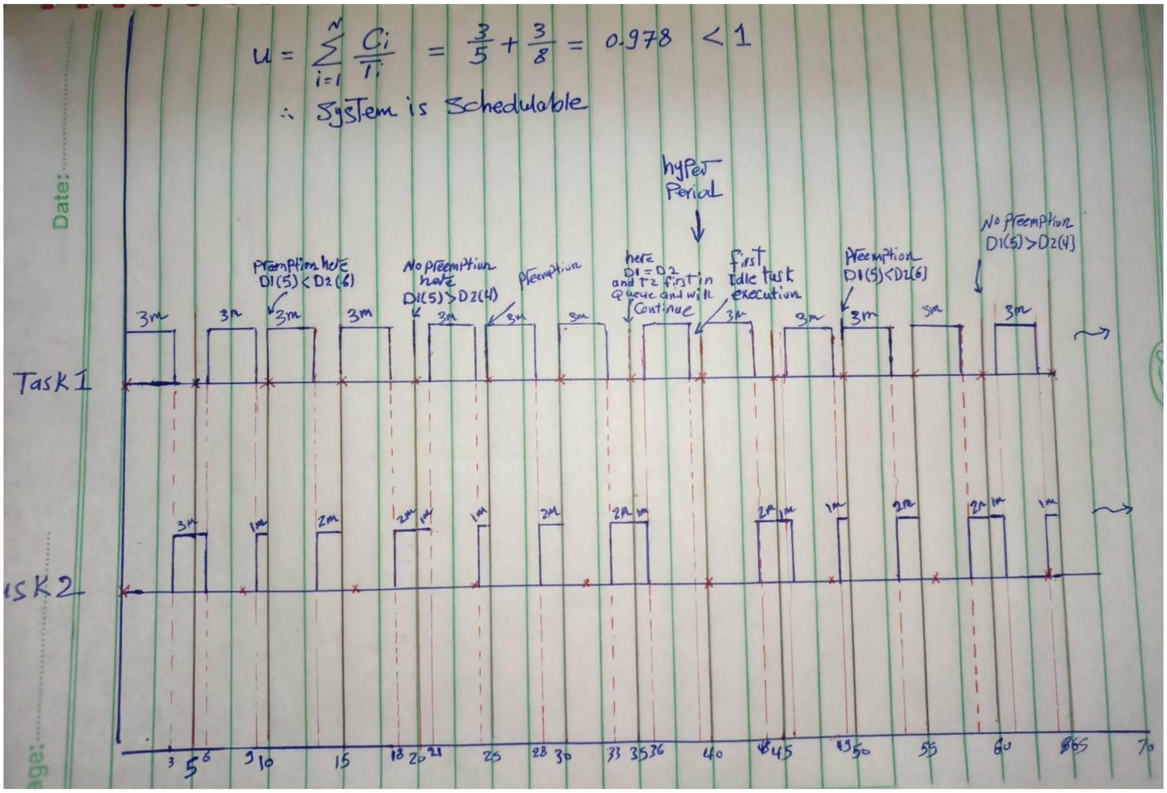
Task Set

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

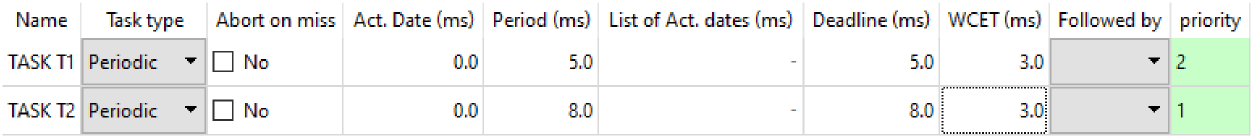

$u = \sum_{i=1}^{N} \frac{C_i}{T_i} = \frac{3}{5} + \frac{3}{8} = 0.978 < 1$
∴ System is Schedulable
hyper Period
Preamption here D1(5) < D2(6)
No preemption here D1(5) > D2(4)
Preemption
here D1 = D2 and T2 first in Queue and will Continue
first Idle task execution
Preemption D1(5)<D2(6)
No preemption D1(5)>D2(4)
Task 1
Task 2
3m
2m
1m
3 5 6 9 10 15 18 20 21 25 28 30 33 35 36 40 45 50 55 60 65 70

Using the analytical method:

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

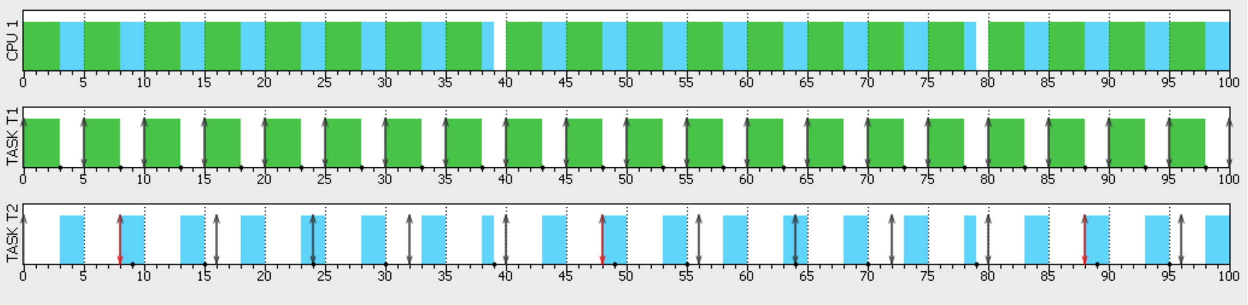
Simulator

Task set



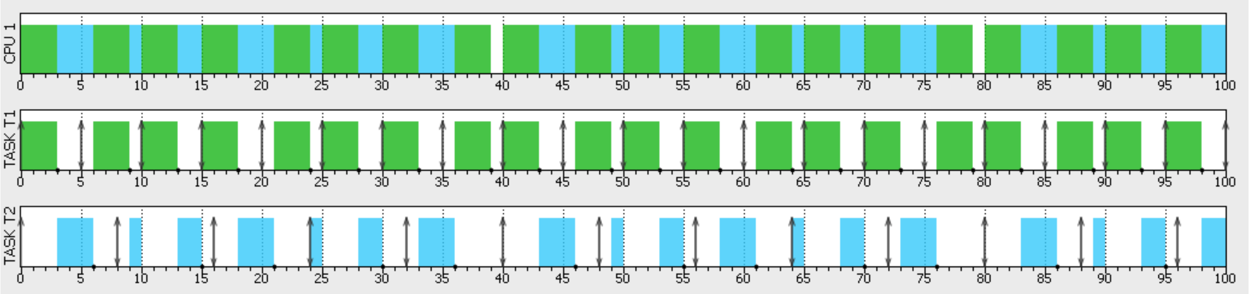
Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	priority
TASK T1	Periodic	□ No	0.0	5.0	-	5.0	3.0		2
TASK T2	Periodic	□ No	0.0	8.0	-	8.0	3.0		1

FP Scheduler



Task 2 Misses its deadline on using a FP scheduler.

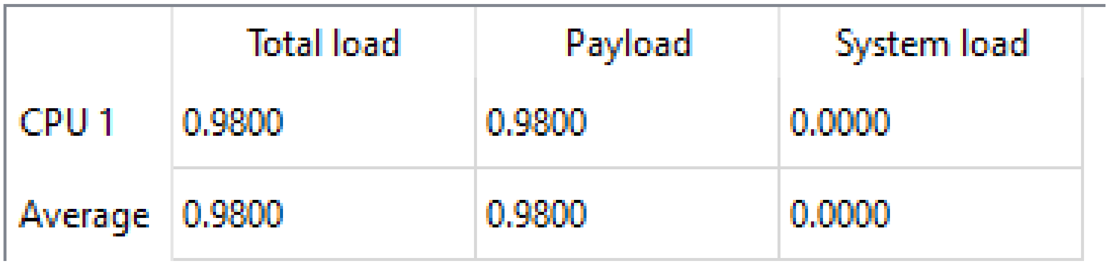
EDF Scheduler



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.

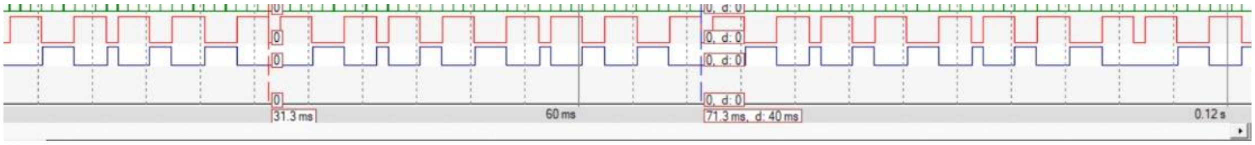
CPU Load

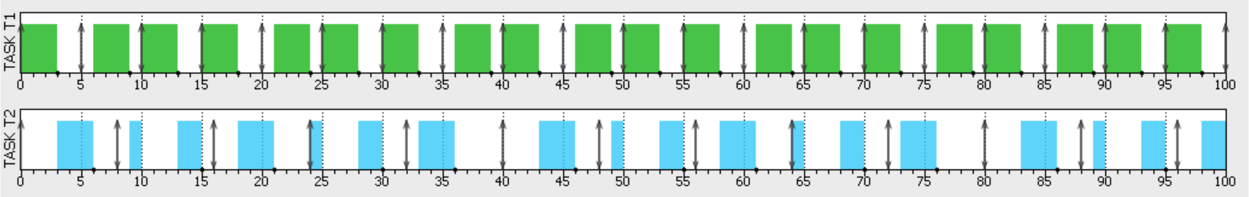


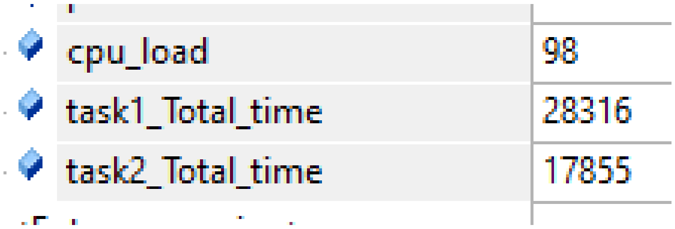
	Total load	Payload	System load
CPU 1	0.9800	0.9800	0.0000
Average	0.9800	0.9800	0.0000

- 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

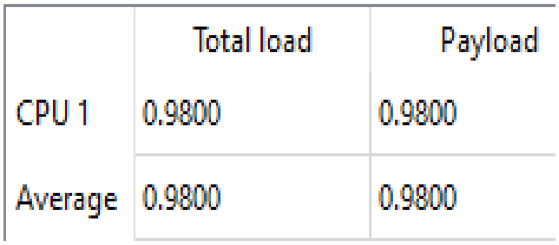
Runtime Analysis







cpu_load	98
task1_Total_time	28316
task2_Total_time	17855



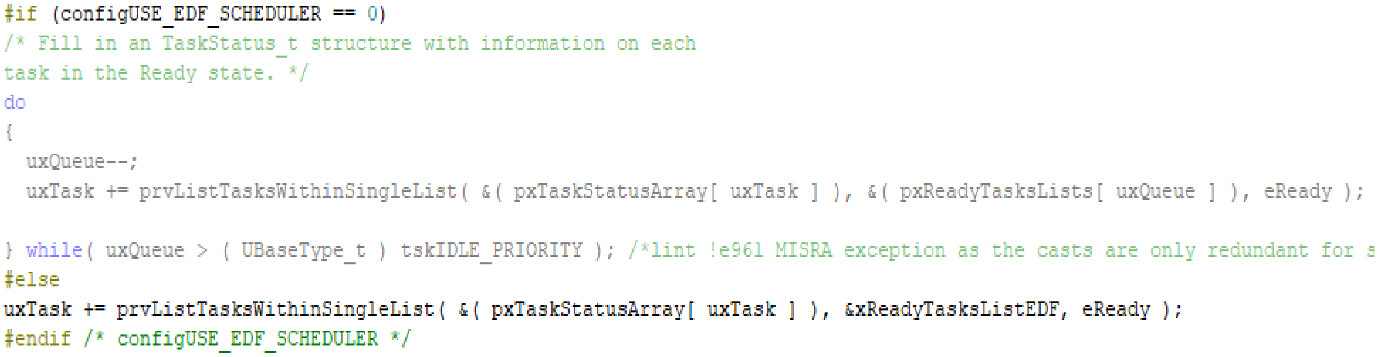
	Total load	Payload
CPU 1	0.9800	0.9800
Average	0.9800	0.9800

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

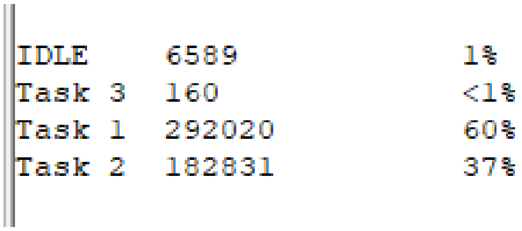
- This is the suggested change in uxTaskGetSystemState Api to enable runtime stats of the FreeRTOS with the EDF scheduler.



```
#if (configUSE_EDF_SCHEDULER == 0)
/* Fill in an TaskStatus_t structure with information on each
task in the Ready state. */
do
{
  uxQueue--;
  uxTask += prvListTasksWithinSingleList( &( pxTaskStatusArray[ uxTask ] ), &( pxReadyTasksLists[ uxQueue ] ), eReady );

} while( uxQueue > ( UBaseType_t ) tskIDLE_PRIORITY ); /*lint !e961 MISRA exception as the casts are only redundant for s
#else
uxTask += prvListTasksWithinSingleList( &( pxTaskStatusArray[ uxTask ] ), &xReadyTasksListEDF, eReady );
#endif /* configUSE_EDF_SCHEDULER */
```

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



```
IDLE     6589          1%
Task 3   160           <1%
Task 1   292020        60%
Task 2   182831        37%
```