Implementation and Test of EDF Scheduler in FreeRTOS

Objective:

-Implementation of a dynamic priority-based preemptive scheduler that changes priorities of tasks according to their deadlines. The priority of a task is inversely proportional to its absolute deadline.

Implementation considerations:

-Tasks are periodic only. i.e. no sporadic or aperiodic tasks.
-Tasks deadlines are their periods. i.e. each task must be completed before the next task-request.
-Tasks are independent.
-Tasks are of constant Run-time.

Due to the previous assumptions tasks can be characterized by only two parameters:

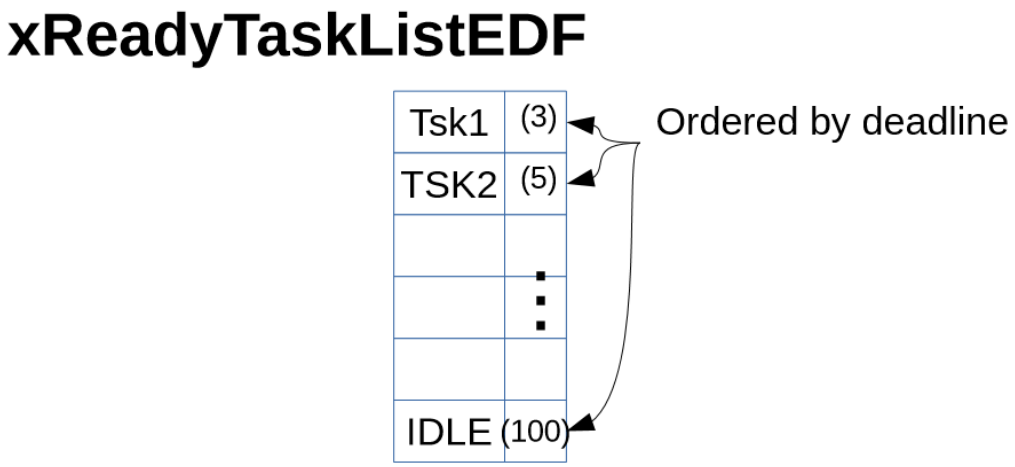
-Task period.
-Task Run-Time.

Theorem used to prove the schedulability of tasks using EDF schedule:

$$\mu = \sum_{i=1}^{N} \frac{Ci}{Ti} \leq 1$$

Implementation in FreeRTOS:

-The general Idea is to create a new Ready List able to manage a dynamic task behavior: it will contain tasks ordered by increasing deadline time, where positions in the list represents the tasks priorities



Step(1) Define “configUSE_EDF_SCHEDULER” as a configuration parameter inside “FreeRTOSConfig.h” file

```
#define configUSE_EDF_SCHEDULER                 1
```

Step(2) Declare a list “ xReadyTasksListEDF” for ready tasks that are scheduled using EDF scheduler in the “tasks.c” file.

```
#if ( configUSE_EDF_SCHEDULER == 1 )
PRIVILEGED_DATA static List_t xReadyTasksListEDF;          /*< Ready
tasks ordered by their deadline >*/
#endif
```

Step(3) Initialize the new list of tasks that are to be scheduled using the EDF scheduler. This initialization is done inside “`prvInitialiseTaskLists()`” method in the “tasks.c” file.

```
#if ( configUSE_EDF_SCHEDULER == 1 )
vListInitialise( &xReadyTasksListEDF );
#endif
```

Step(4) Modify the method "`prvAddTaskToReadyList( pxTCB )`" that adds the task to the ready list in the "tasks.c" file.

```
#if ( configUSE_EDF_SCHEDULER == 0 )
#define prvAddTaskToReadyList( pxTCB )                          \
    traceMOVED_TASK_TO_READY_STATE( pxTCB );                    \
    taskRECORD_READY_PRIORITY( ( pxTCB )->uxPriority );         \
    listINSERT_END( &( pxReadyTasksLists[ ( pxTCB )->uxPriority ] ), &(
( pxTCB )->xStateListItem ) ); \
    tracePOST_MOVED_TASK_TO_READY_STATE( pxTCB )
#else
#define prvAddTaskToReadyList( pxTCB )                           \
      vListInsert( &(xReadyTasksListEDF), &( (pxTCB)->xStateListItem )
);
#endif
```

Step(5) Adding the task period Parameter to the TCB structure in the "tasks.c" file.

```
#if ( configUSE_EDF_SCHEDULER == 1 )
            TickType_t xTaskPeriod; /*Stores period of task in tick*/
            #endif
} tskTCB;
```

Step(6) Modify "`xTaskCreate()`" protoType to add the period parameter in the "tasks.c"&"tasks.h" files.

```
#if ( configUSE_EDF_SCHEDULER == 0 )
    BaseType_t xTaskCreate( TaskFunction_t pxTaskCode,
                            const char * const pcName,
const configSTACK_DEPTH_TYPE usStackDepth,
                            void * const pvParameters,
                            UBaseType_t uxPriority,
                            TaskHandle_t * const pxCreatedTask )
PRIVILEGED_FUNCTION;
#else
            BaseType_t xTaskCreate( TaskFunction_t pxTaskCode,
                            const char * const pcName,     /*lint !e971
Unqualified char types are allowed for strings and single characters
only. */
                            const configSTACK_DEPTH_TYPE usStackDepth,
                            void * const pvParameters,
                            UBaseType_t uxPriority,
                            TaskHandle_t * const pxCreatedTask,

            TickType_t period ) PRIVILEGED_FUNCTION;
#endif
```

Step(7) Initialize the period parameter of TCB inside the "`xTaskCreate()`" method in the

“tasks.c” file.

```
pxNewTCB->xTaskPeriod = period;
```

Step(8) Insert the period value in the xStateListItem before adding the task to the new ready list in the “tasks.c” file.

```
listSET_LIST_ITEM_VALUE( &((pxNewTCB)->xStateListItem),
(pxNewTCB)->xTaskPeriod - xTaskGetTickCount()%(pxNewTCB)->xTaskPeriod);
```

Step(9) Modify the function call of “`xTaskCreate()`” inside the “`vTaskStartScheduler()`” the period value, this function call is used to create the IDLE Task.

```
xReturn = xTaskCreate( prvIdleTask,
                       configIDLE_TASK_NAME,
                       configMINIMAL_STACK_SIZE,
                       ( void * ) NULL,
                       portPRIVILEGE_BIT,
                       &xIdleTaskHandle
                       #if (configUSE_EDF_SCHEDULER == 1)   /**/
                       , 100  /*step(9)Insert the period value*/
                       #endif
                   );
```

Step(10) update the Idle's task deadline periodically inside it’s context.

```
 for( ; ; )
    {
      #if (configUSE_EDF_SCHEDULER == 1)
      listSET_LIST_ITEM_VALUE( &( (pxCurrentTCB)->xStateListItem),(
pxCurrentTCB )->xTaskPeriod - xTaskGetTickCount()%( pxCurrentTCB
)->xTaskPeriod);
                         prvAddTaskToReadyList(pxCurrentTCB);
      #endif
```

Step(11) Modify the “`vTaskSwitchContext()`” method in the “tasks.c” file.

```
#if (configUSE_EDF_SCHEDULER == 1)
pxCurrentTCB = (TCB_t *)
listGET_OWNER_OF_HEAD_ENTRY(&(xReadyTasksListEDF));
#else
taskSELECT_HIGHEST_PRIORITY_TASK();
#endif
```

Step(12) The last step is to update the task’s deadline inside the “`xTaskIncrementTick()`” method in the “tasks.c” file.

```
#if (configUSE_EDF_SCHEDULER == 1)

listSET_LIST_ITEM_VALUE(&((pxTCB)->xStateListItem), (pxTCB)->xTaskPeriod
- xTaskGetTickCount()%(pxTCB)->xTaskPeriod);
#endif
```

```
prvAddTaskToReadyList( pxTCB );
```

The calculation of deadline at the Current Tick:

TicksToTaskDeadline = TaskPeriod - (CurrentTick % TaskPeriod)

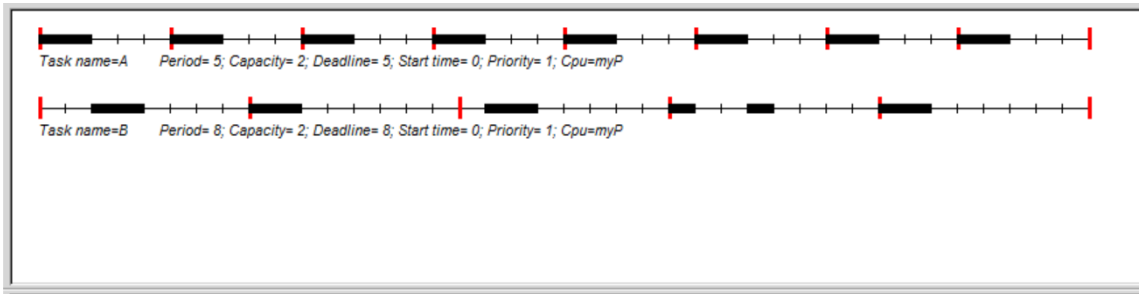
The task set that we used to test our EDF scheduler:

Task(A) of period = 5 Ticks and execution time = 2 Ticks.

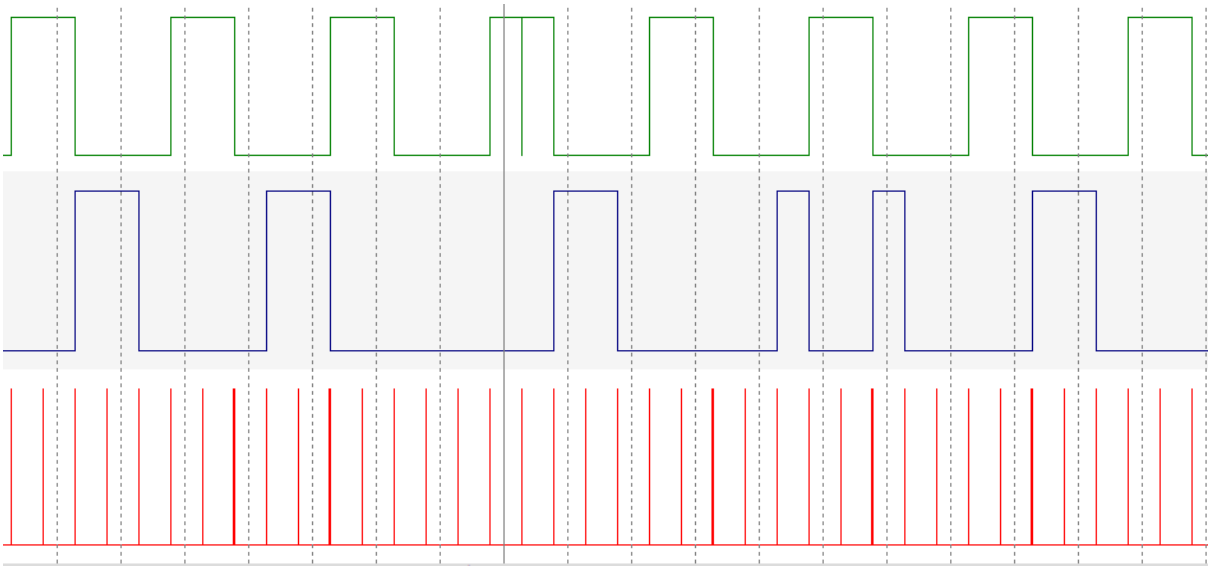
Task(B) of period = 8 Ticks and execution time = 2 Ticks.

$$U = \frac{2}{5} + \frac{2}{8} = 0.65 < 1$$

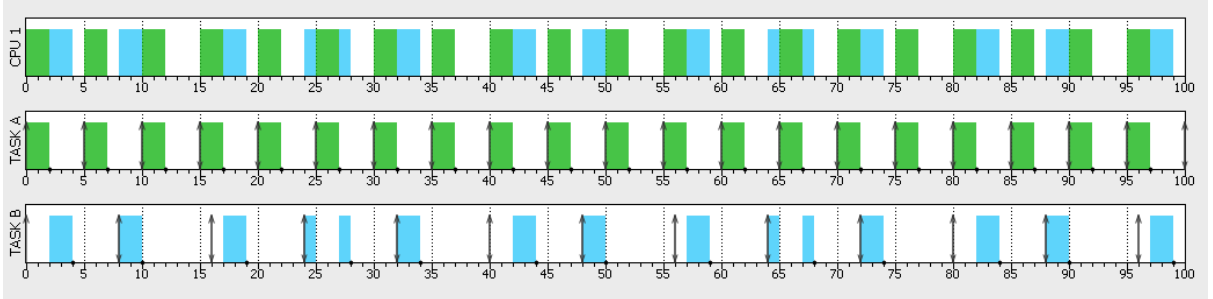
So according to the Theorem our Task set is schedulable.



The following figure is the expected result for the EDF scheduling of our Task set.



The following figure is the actual result for the EDF scheduling of our Task set.



The following figure is the simulation result for the EDF scheduling of our Task set. Simulation is done by Simso.