

# Earliest Deadline Scheduler (EDF)

Implementation of EDF in FreeRTOS

Mustafa Mahmoud Ali

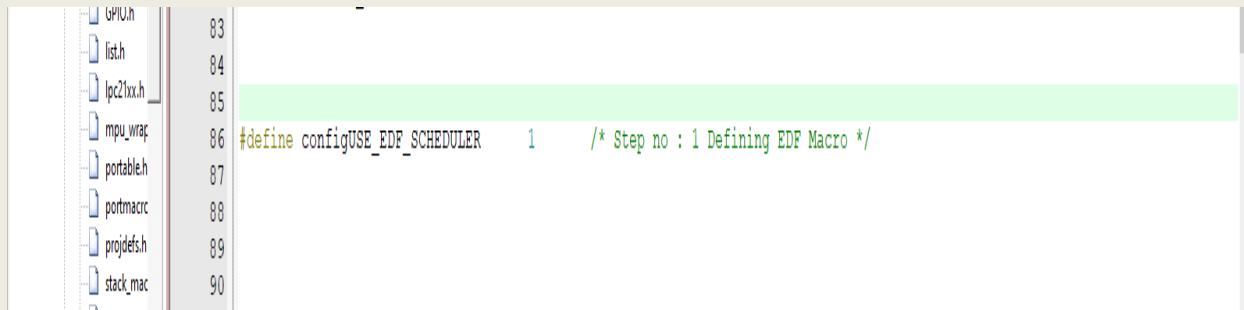
## Earliest Deadline First:

The first scheduler we will implement is based on the Earliest Deadline First algorithm (EDF). EDF adopts a dynamic priority-based preemptive scheduling policy, meaning that the priority of a task can change during its execution, and the processing of any task is interrupted by a request for any higher priority task.

## Implementation:

### Step one:

Defining the EDF Scheduler Macro to the “FreeRTOSConfig.h” file

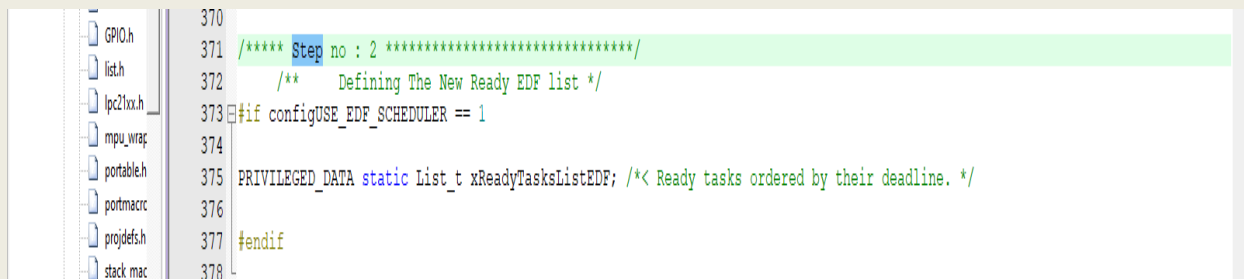


```
83  
84  
85  
86 #define configUSE_EDF_SCHEDULER 1 /* Step no : 1 Defining EDF Macro */  
87  
88  
89  
90
```

And the rest of the steps in “Task.c” File.

### Step two:

Ready List is declared: *xReadyTasksListEDF*



```
370  
371 /***** Step no : 2 *****/  
372 /** Defining The New Ready EDF list */  
373 #if configUSE_EDF_SCHEDULER == 1  
374  
375 PRIVILEGED_DATA static List_t xReadyTasksListEDF; /*< Ready tasks ordered by their deadline. */  
376  
377 #endif  
378
```

Step three:

*prvInitialiseTaskLists()* method, that initialize all the task lists at the creation of the first task, is modified adding the initialization of *xReadyTasksListEDF* :

```
lpc21xx.h 3847  /**** EDF *****/
mpu_wrap 3848  #if configUSE_EDF_SCHEDULER == 1
portable.h 3849  vListInitialise( &xReadyTasksListEDF ); /**** Step no : 3 initialising the Ready list *****/
portmacro 3850  #endif
projdefs.h 3851
```

Step four:

*prvAddTaskToReadyList()* method that adds a task to the Ready List is then modified as follows:

```
GPIO.c 231
GPIO_cfg.c 232  /*** EDF Scheduler ***/
main.c 233  /**** Step no : 4 ****/
serial.c 234  #if configUSE_EDF_SCHEDULER == 1
FreeRTOS 235  #define prvAddTaskToReadyList( pxTCB )\
tasks.c 236  vListInsert( &(xReadyTasksListEDF), &( ( pxTCB )->xStateListItem ) )
deprcate 237
FreeRTOS.h 238  #endif
FreeRTOS.h 239  /*-----*/
GPIO.h 240
```

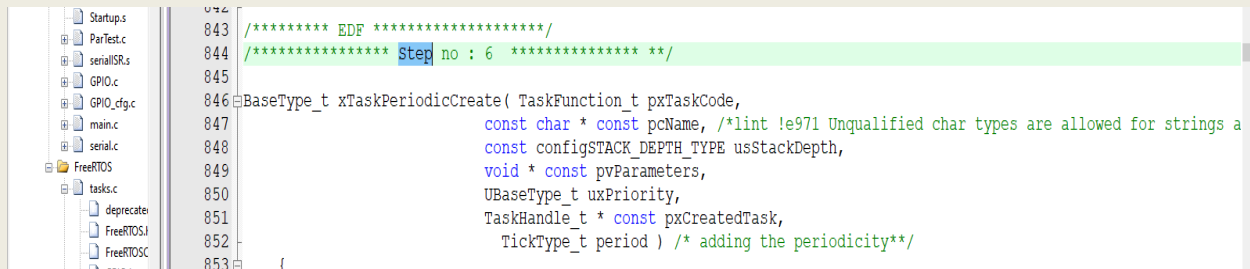
Step five:

A new variable is added in the *tskTaskControlBlock* structure (TCB):

```
FreeRTOS 345  /***** EDF *****/
tasks.c 346  /****Step no : 5 the period of a task *****/
deprcate 347  #if ( configUSE_EDF_SCHEDULER == 1 )
FreeRTOS.h 348  TickType_t xTaskPeriod; /*< Stores the period in tick of the task. > */
FreeRTOS.h 349  #endif
GPIO.h 350  } tskTCB;
list.h 351
```

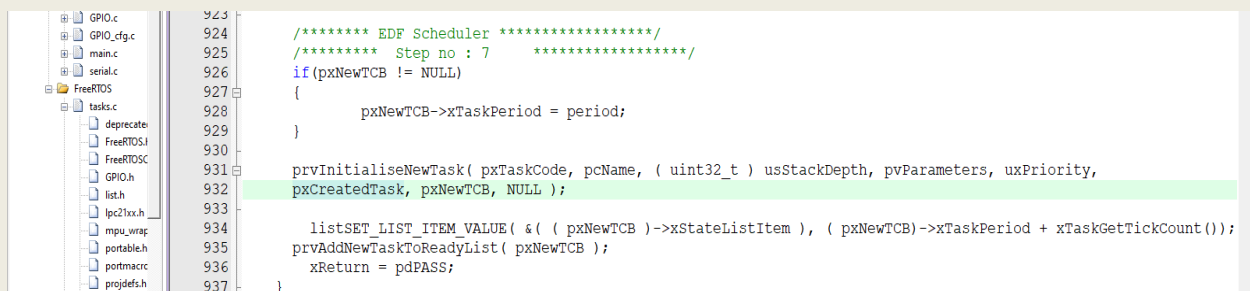
Step six:

Modifying the xCreatTask to be xperiodicCreate() with new features which are :  
adding the new period and update it before adding it to the xStatelist



```
843 /***** EDF *****/
844 /***** Step no : 6 *****/
845
846 BaseType_t xTaskPeriodicCreate( TaskFunction_t pxTaskCode,
847                                const char * const pcName, /*lint !e971 Unqualified char types are allowed for strings a
848                                const configSTACK_DEPTH_TYPE usStackDepth,
849                                void * const pvParameters,
850                                UBaseType_t uxPriority,
851                                TaskHandle_t * const pxCreatedTask,
852                                TickType_t period ) /* adding the periodicity*/
853 {
```

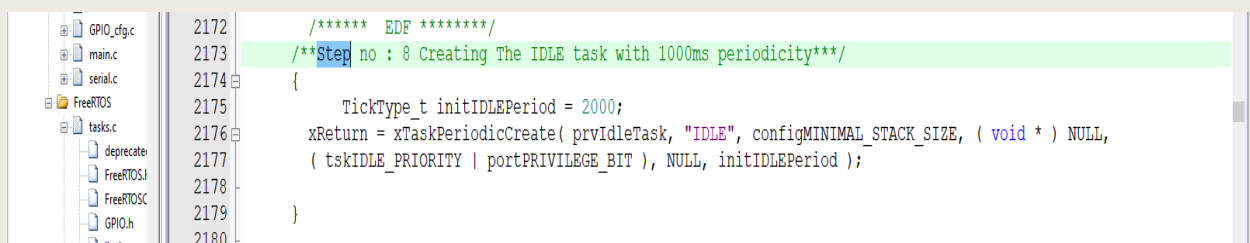
Step seven:



```
924 /***** EDF Scheduler *****/
925 /***** Step no : 7 *****/
926 if(pxNewTCB != NULL)
927 {
928     pxNewTCB->xTaskPeriod = period;
929 }
930
931 prvInitialiseNewTask( pxTaskCode, pcName, ( uint32_t ) usStackDepth, pvParameters, uxPriority,
932                      pxCreatedTask, pxNewTCB, NULL );
933
934 listSET_LIST_ITEM_VALUE( &( ( pxNewTCB )->xStateListItem ), ( pxNewTCB )->xTaskPeriod + xTaskGetTickCount());
935 prvAddNewTaskToReadyList( pxNewTCB );
936 xReturn = pdPASS;
937 }
```

Step eight:

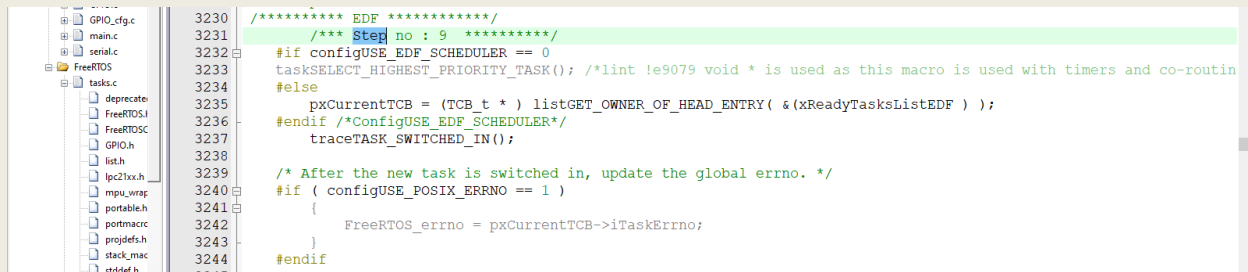
Modifying the IDLE task to be longest task to avoid being in running state while  
Other task must be in instead



```
2172 /***** EDF *****/
2173 /***** Step no : 8 Creating The IDLE task with 1000ms periodicity****/
2174 {
2175     TickType_t initIDLEPeriod = 2000;
2176     xReturn = xTaskPeriodicCreate( prvIdleTask, "IDLE", configMINIMAL_STACK_SIZE, ( void * ) NULL,
2177                                   ( tskIDLE_PRIORITY | portPRIVILEGE_BIT ), NULL, initIDLEPeriod );
2178 }
2179
2180
```

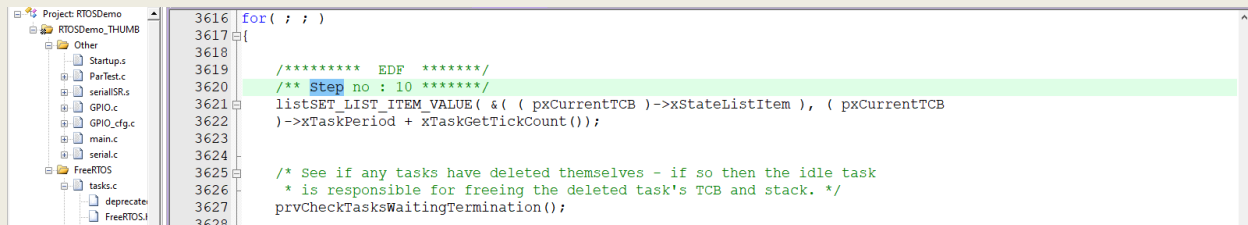
Step nine:

Make the TCB always points to the top of the ready list which is ready to run



```
3230 /***** EDF *****/
3231 /** Step no : 9 *****/
3232 #if configUSE_EDF_SCHEDULER == 0
3233 taskSELECT_HIGHEST_PRIORITY_TASK(); /*lint !e9079 void * is used as this macro is used with timers and co-routine */
3234 #else
3235 pxCurrentTCB = (TCB_t *) listGET_OWNER_OF_HEAD_ENTRY( &(xReadyTasksListEDF) );
3236 #endif /*ConfigUSE_EDF_SCHEDULER*/
3237 traceTASK_SWITCHED_IN();
3238
3239 /* After the new task is switched in, update the global errno. */
3240 #if ( configUSE_POSIX_ERRNO == 1 )
3241 {
3242     FreeRTOS_errno = pxCurrentTCB->TaskErrno;
3243 }
3244 #endif
```

Step ten:



```
3616 for( ; ; )
3617 {
3618     /***** EDF *****/
3619     /** Step no : 10 *****/
3620     listSET_LIST_ITEM_VALUE( &( pxCurrentTCB )->xStateListItem, ( pxCurrentTCB
3621     )->xTaskPeriod + xTaskGetTickCount() );
3622
3623     /* See if any tasks have deleted themselves - if so then the idle task
3624     * is responsible for freeing the deleted task's TCB and stack. */
3625     prvCheckTasksWaitingTermination();
3626 }
3627
```

-The Tasks in the main function are:

T1 {P: 5, E: 2.5, D: 5}, T2 {P: 15, E: 4.5, D: 15}, T3 {P: 20, E: 3.5, D: 20}

- First we need to calculate the U for the System (CPU Load )

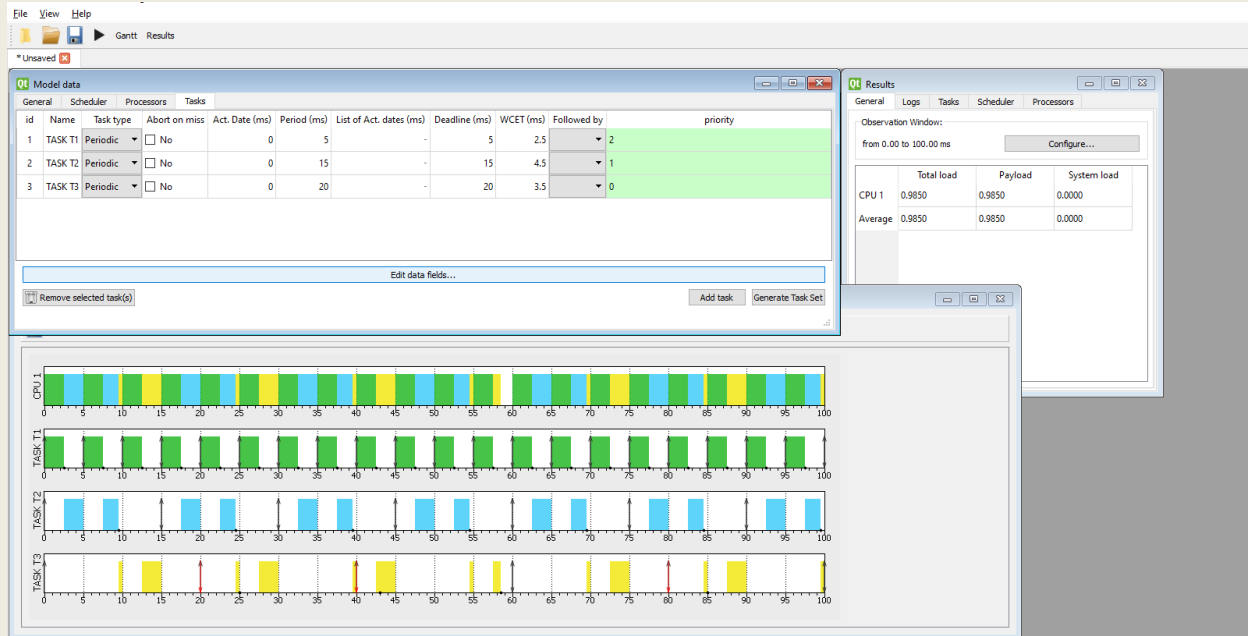
$$U = (2.5/5) + (4.5/15) + (3.5/20) = 0.975$$

- URM using the equation of RM =  $n(2^{1/n} - 1)$

$$3(2^{1/3} - 1) = 0.779$$

$U > URM$ , Then the system is guaranteed not schedulable

Task 3 always misses its deadline time



### (Rate-Monotonic Scheduler)

Calculating the Time Analysis For the system

For Task 1 :

$$W(1 \text{ to } 5) = 2.5 + 0 = 2.5 < 5 \text{ T1 is schedulable}$$

For Task 2 : The execution would happen after Task 1 finishes its time

$W() =$  The execution of task 1 + task 2

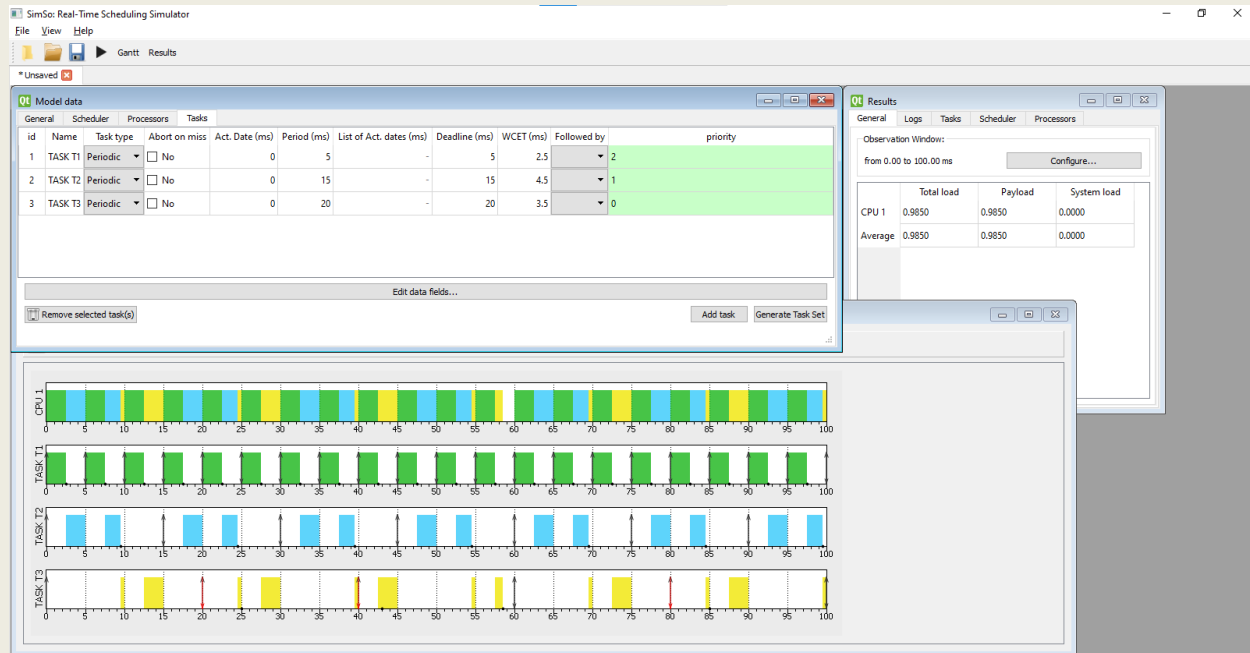
$$W(1 \text{ to } 5) = 4.5 + (1/5)2.5 = 7$$

$$W(10) = 4.5 + (10/5)2.5 = 9.5$$

$$W(15) = 4.5 + (15/5)2.5 = 12 < 15 \text{ T2 is schedulable}$$

For Task 3 :  $W() =$  Total execution time

$$W(16) = 3.5 + (16/5)4.5 + (16/5)2.5 = 25.9 > 20 \text{ Task 3 always misses its deadline}$$



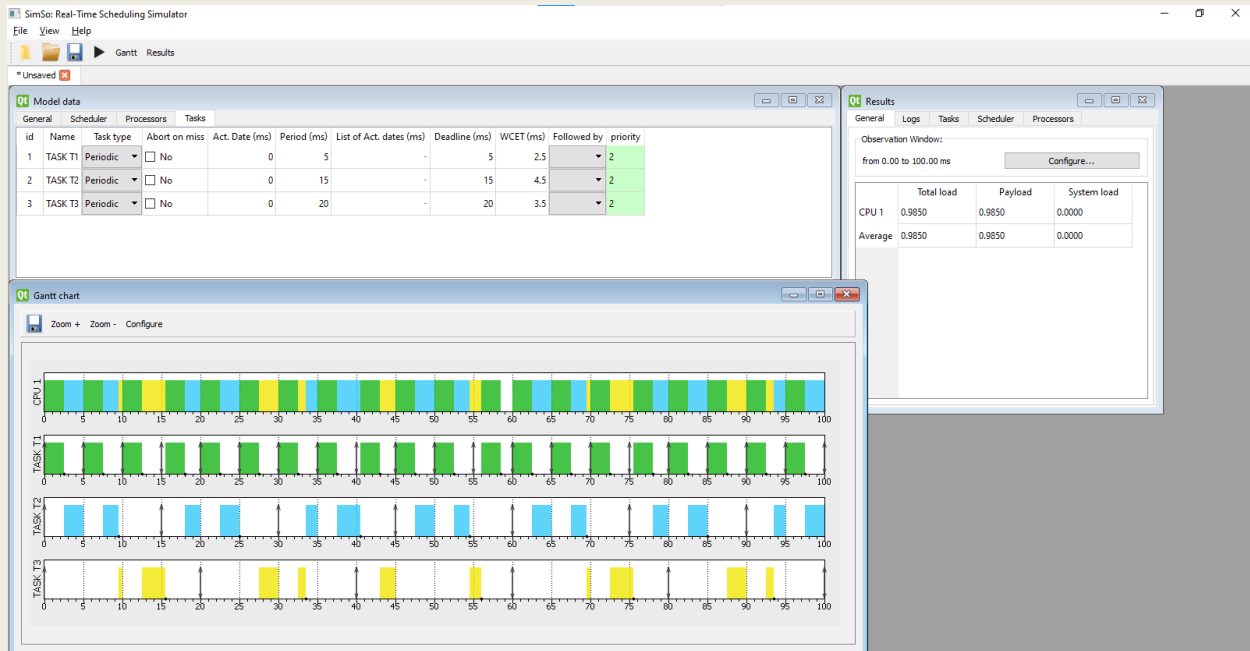
### (Run-Time FP Scheduler)

In the EDF scheduler the system is said to be schedulable when

$$U = \sum_{i=1}^N \frac{C_i}{T_i} \leq 1$$

$$U = (2.5/5) + (4.5/15) + (3.5/20) = 0.975 < 1$$

Then the system is schedulable



Applying the System on the logic analyzer in keil:

