Thesis Implementation

Messing Changes

1. Make Sure Idle Task Stays At The End Of The EDF List

- In tasks.c in `prvIdleTask()`

```
3446  for (;;) {
3447
3448  #if(configUSE_EDF_SCHEDULER == 1)
3449
3450    /* INCREMENT IDLE TASK DEADLINE MAKE SURE IT HOLDS A VALUE
3451      GREATER THAN ANY TASK THAT WILL AWAKEN AT ANY TIME SO IT
3452      REMAINS AT THE END OF THE LIST */
3453
3454    /* IT WILL ALWAYS BE OFFSETTED BY configINIT_IDLE_PERIOD FROM THE
3455    MAXIMUM TASK DEADLINE */
3456
3457    pxCurrentTCB->xStateListItem.xItemValue = xTickCount
3458    + configINIT_IDLE_PERIOD + xMaxTaskDeadLine;
3459
3460  #endif /* configUSE_EDF_SCHEDULER */
3461
```

- Note: `pxCurrentTCB` inside of the IDLE task code points to the IDLE task itself.
- Note: `xMaxTaskDeadLine` is a variable that holds the maximum deadline value of all created tasks. So The IDLE task will always be late by `configINIT_IDLE_PERIOD` as it always updates with the current tick.
- In tasks.c

```
451  /* Other file private variables. --------------------------------
452  #if(configUSE_EDF_SCHEDULER == 1)
453  PRIVILEGED_DATA static volatile UBaseType_t xMaxTaskDeadLine =
454      (UBaseType_t)0U;
455  #endif /* configUSE_EDF_SCHEDULER */
```

- In tasks.c in `xTaskPeriodicCreate()`

```
1016      /* COMPARE THIS NEW TASK DEADLINE TO THE MAXIMUM DEADLINE */
1017      if(period > xMaxTaskDeadLine) xMaxTaskDeadLine = period;
1018
```

2. Modifiy xTaskIncrementTick To Revaluate Awakened Tasks Deadline. And Insert It At The Right Place In EDF List

- In tasks.c in `xTaskIncrementTick()`

```
2856            * list. */
2857
2858           #if(configUSE_EDF_SCHEDULER == 1)
2859
2860           /* Re-evaluate the awakened task deadline */
2861           /* STORE THE DEADLINE INSIDE OF TASK NODE BEFORE ADDING IT TO THE READY LIST */
2862
2863             listSET_LIST_ITEM_VALUE( &( ( pxTCB )->xStateListItem ),
2864               ( pxTCB->xTaskPeriod + xTickCount));
2865
2866           #endif
2867           prvAddTaskToReadyList(pxTCB);
2868
2869       /* A task being unblocked cannot cause an immediate
```

---

3. Change The Logic Of When A Context Switch Should Happen.

- In tasks.c in `xTaskIncrementTick()`

```
2876            * currently executing task. */
2877           #if(configUSE_EDF_SCHEDULER == 1)
2878
2879           /* COMPARE DEADLINES INSTEAD OF PRIORITIES */
2880           if (pxTCB->xStateListItem.xItemValue < pxCurrentTCB->xStateListItem.xItemValue) {
2881             xSwitchRequired = pdTRUE;
2882           }
2883
2884           #else
2885
2886           if (pxTCB->uxPriority >= pxCurrentTCB->uxPriority) {
2887             xSwitchRequired = pdTRUE;
2888           }
2889
2890           #endif /* configUSE_EDF_SCHEDULER */
2891
2892           else {
2893             mtCOVERAGE_TEST_MARKER();
2894           }
```

- Note: Instead of comparing priorities, I changed it to compare if the just awakened task has lower deadline than the current running task. If that happen then a context switch should take place. No need to worry about what node will the contextswitch method will choose since we have already modified it to choose the head node at the EDF list as said in the Thesis. All we had to do is to signal that a context switch needs

to happen when a task of lower deadline value than the current running task awakens.