Michael Daniel Werezak - 20303777

Lab Assignment 4 – EDF Scheduler

EDF Task Data Structure

In order to implement the EDF Scheduler, some additional data must be stored for each FreeRTOS task. These larger entities built around the FreeRTOS tasks will be referred to as EDF tasks. Each EDF task's data is stored in a TaskInfo data structure, which are kept in a statically allocated array.

In theory, an EDF task can be specified by only the execution time and the period. To implement the EDF, some additional data was needed. This included the time of the last deadline (i.e. the start of the current period) and the amount of work already completed in the current period, a handle for an associated FreeRTOS task and FreeRTOS timer, as well as some diagnostic information.

Priority Calculation

Implementation of the earliest-deadline-first property is done primarily through manipulating FreeRTOS task priorities. Thus, the EDF scheduler needs to ensure that the tasks maintained priority in order of their earliest deadline. Since FreeRTOS only supports a small number of priorities, this could not be done by setting priority using the inverse of each task's next deadline. Instead, task priorities are updated by sorting each task by their next deadline time using a temporary array. A bubble sort algorithm was chosen for ease of implementation, and because the number of tasks to be sorted is very small.

The priority of each task is then set according to their position in the array. Since the priorities of every task are updated at once, the update is done inside a critical section to avoid leaving the task priorities in an inconsistent state.

Initial Priorities

Since the priority of each task is only known once all tasks have been created, the initial priority of each task could not be set using xTaskCreate(). As well, it was found that calling vTaskPrioritySet() before the scheduler has started causes FreeRTOS to crash. Therefore, an initialization task was used to set the initial task priorities. This task is created with maximum priority and it initializes the priority of each EDF task before deleting itself.

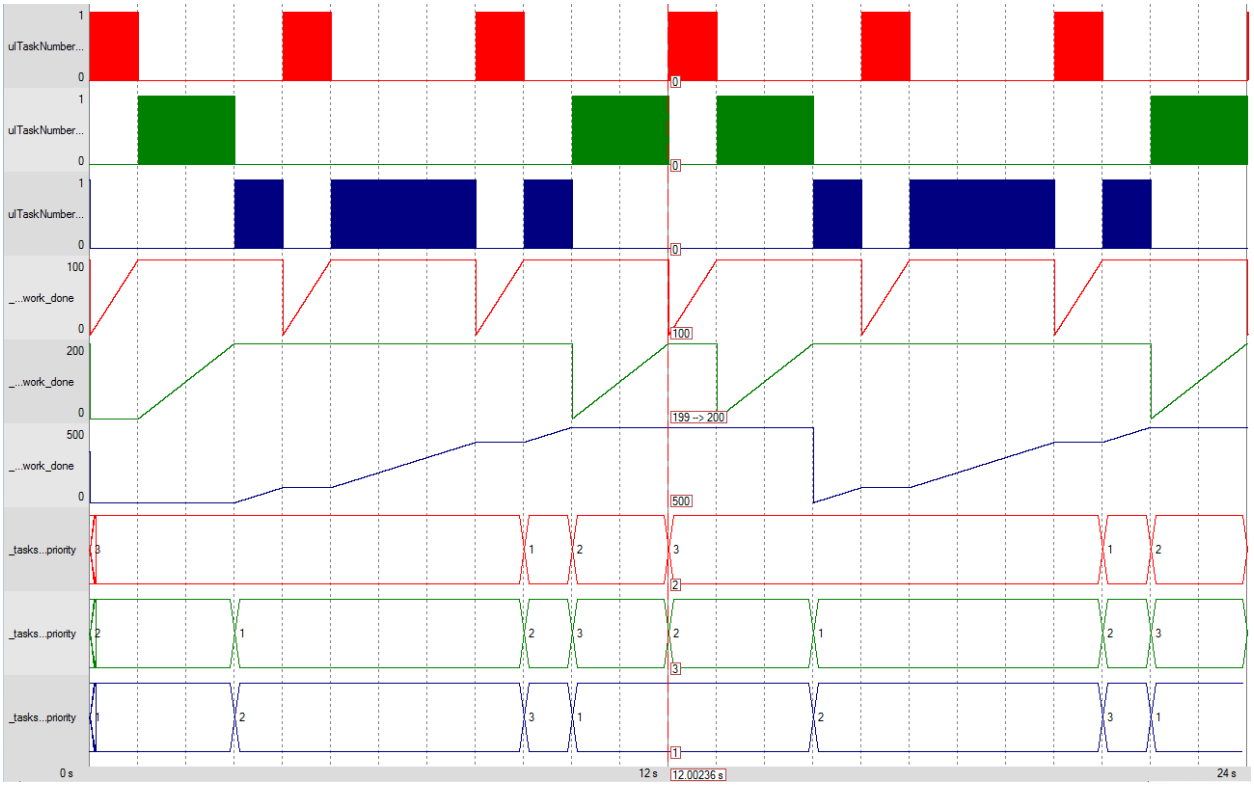
Task Execution

Each EDF task uses a FreeRTOS timer used as a one-shot timer to simulate execution time. When executing, each task continuously starts the timer and waits for the timer to fire, at which point counter is incremented and the timer is started again (one "unit of work"). This is done so that execution progress can be tracked in the case of pre-emption. The desired execution time is achieved by determining the required number of units of work.

While executing, each task checks to see if a deadline is missed by comparing the current time to its next deadline. If a deadline was missed, any current execution progress is lost, and priorities are reassessed using the next deadline for that task. Because the scheduler ensures that the currently executing task is the one with the earliest deadline, only the deadline of the currently executing task needs to be checked.

Since the task periods are constant and do not change once the scheduler has started, the task priorities only need to be reassessed once a task completes execution for its current period or misses a deadline.

Scheduling Diagram



The diagram above was created using the following set of tasks:

Red – T1(1,4)
Green – T2(2,6)
Blue – T3(5,12)

Two hyper-periods are shown, and the cursor has been placed on the boundary between the two. In addition to the scheduling, the amount of work completed in the current period and the assigned priority for each task is also shown.