# CSC714: Real Time Systems Project Proposal

**Group Info:**
BBHAT BALASUBRAMANYA BHAT
SBUDANU SANDEEP BUDANUR RAMANNA

## Title

Pre-emptive EDF scheduler implementation on an embedded platform.

## Goals / Motivation

This project aims to create a highly efficient periodic scheduler based on pre-emptive Earliest Deadline First scheduling algorithm on an actual embedded platform. The motivation behind this project is to understand the complexities behind implementing a real-time periodic scheduler on an actual embedded platform as well as the Earliest Deadline First algorithm for real-time scheduling. Besides these, if this project is successful in its goal of producing a highly efficient scheduling, we intend to port this scheduler for task scheduling on the SST Controller board in the FREEDM project.

## Introduction

Real Time Operating Systems use specialized scheduling algorithms to provide predictable behavior in hard real-time systems. The scheduler has to guarantee that all periodic tasks meet their respective deadlines. Therefore the scheduler forms the core of any real-time kernel. As part of this project, we intend to implement an Earliest Deadline First scheduler for periodic tasks and a simple static priority based preemptive scheduling for aperiodic tasks.

## Description

Periodic tasks release their jobs at regular intervals each of which needs to be completed before their respective deadline. A hard real-time system has to ensure that all deadlines are met for all the jobs. In a soft real-time system, the deadlines can be missed occasionally. The deadlines can be earlier/same as/greater than the task period. The EDF scheduler always executes the job with the next earliest deadline at every scheduling instant. Aperiodic tasks can be scheduled when no periodic tasks are running (slack time).

Following are the requirements we intend to support in our scheduler:

- The scheduler shall support periodic tasks with deadlines equal to or less than the period.
- It shall support EDF based scheduling for periodic tasks.

- If two or more jobs have same deadline, then they are scheduled in the FIFO manner.
- The schedulability test shall be performed for each periodic task when that task is created. The task shall be successfully created only if the schedulability test passes.
- The scheduler shall be capable of creating tasks based on phase, period, execution time and relative deadlines.
- All time values shall have one micro second resolution.
- The execution time for each job shall be strictly monitored and shall not be allowed to exceed.
- The scheduler shall also support aperiodic tasks.
- The aperiodic tasks shall have lesser priority than all periodic tasks.
- The aperiodic tasks shall be scheduled using static priority based preemptive scheduling.
- The scheduler shall be easily portable to different platforms
- The scheduler shall be as efficient as possible (least scheduling overhead).
- The scheduler shall have smallest possible memory footprint.
- The scheduler shall provide method for periodic and aperiodic tasks to sleep for given amount of time.
- The scheduler shall provide method for tasks to yield to other tasks.
- The scheduler shall provide APIs related to task synchronization like mutex / semaphores etc.
- The scheduler shall provide an idle task which is executed when there are no tasks in ready state.
- The scheduler shall also keep track of the current CPU utilization.
- The following set of APIs shall be supported by the scheduler.
    - task creation / deletion
    - yield / sleep
    - Synchronization (semaphore / mutex) APIs.

# Implementation

We intend to implement the EDF scheduler on top of some other basic real-time kernel such as uCOS II **[2]**. The uCOS II provides static priority based preemptive scheduling. There is no concept of periodic tasks. Our scheduler provides a wrapper around uCOS II APIs to support periodic tasks.

We intend to deploy our scheduler on the C6713 DSK kit from Texas Instruments. This board has a TMS320C6713 DSP processor running at 150MHz.

All the scheduling decisions are made by our EDF scheduler. Only the task determined by EDF scheduler will be in active state, all other tasks which are not running will be in suspended state. Our EDF scheduler determines which task should be in active / suspended state. This makes sure that the underlying RTOS doesn't waste time in determining the highest priority tasks again among active tasks. Since the underlying

RTOS sees only one active task, it simply executes this task rather than trying to determine the highest priority task.

The EDF scheduler will make use of two hardware timers, one for tracking the deadline of tasks and another for tracking the execution time.

## Possible Extensions / Future Work

Based on the availability of time, following improvements can be made.

1. Implement the EDF scheduler on bare hardware without support of another RTOS for better performance.
2. Implement EDF on multi core systems.

## References

**[1]** Real-Time Systems, Jane W.S. Liu, Pearson Education
**[2]** uc/os2 web portal, http://www.micrium.com/products/rtos/kernel/rtos.html
**[3]** Lecture notes