CSC714: Real Time Systems Project – Report2

Group Members

BBHAT BALASUBRAMANYA BHAT
SBUDANU SANDEEP BUDANUR RAMANNA

Title

Pre-emptive EDF scheduler implementation on an embedded platform.

Task break-up and schedule

Task	Description	Assigned	Status	Completion Date
1	Setting up the environment to run the program on the actual target and the simulator	Bala/Sandeep	Completed	03/23/09
2	Studying the current design of the RMA scheduler on MicroC OS II	Sandeep	Completed	03/29/09
3	High level architecture for the EDF scheduler. Evaluation of different design options.	Bala/Sandeep	Completed	03/29/09
4	Finalizing the APIs to be provided by the scheduler	Bala/Sandeep	Completed	03/29/09
5	Design and implementation of key data structures required for the scheduler	Bala	Completed	03/31/09
6	Implementation of Task Management APIs	Bala/Sandeep	In Progress	04/05/09
7	Startup routines and Interrupt handling	Bala/Sandeep	Not Started	04/05/09
8	Implementation of primary scheduling algorithm	Bala/Sandeep	Not Started	04/12/09
9	Implementation of supporting services (semaphores, profiling APIs etc)	Bala/Sandeep	Not Started	04/12/09
10	Functional Testing	Bala/Sandeep	Not Started	04/19/09
11	Performance evaluation	Bala/Sandeep	Not Started	04/21/09
12	Project Report Generation	Bala/Sandeep	Not Started	04/24/09

Problems

Since the preemptive EDF is a dynamic priority system, we need to change the priority of the tasks dynamically at runtime. Our original design of implementing the EDF scheduler on top of MicroC OS II will result in significant overhead. A code analysis of MicroC OS indicated that the function call to change the priority of tasks results in good amount of code being executed.

Planned Solution

Multiple design choices were evaluated. Since the call to change the priority of a task at runtime is significant, we tried another approach where all tasks except for the one with the earliest deadline will be deactivated. Hence the MicroC will only execute the task which is active. This way the priority scheme is not used as we choose which single task should be active. While this should work fine, it still introduces delays for calling activate and deactivate for all tasks which is still significant. Also in this approach the use of MicroC is almost only limited to maintaining the task control blocks and nothing else. We then decided to eliminate the MicroC OS layer altogether and have a very thin kernel implemented by ourselves. Basically this kernel would provide TCB and Queuing services to our EDF scheduler. This solution should be very efficient compared to using MicroC OS layer.