EEE8068 – Coursework Specification

Project supervisors: Alex Bystrov and Albert Koelmans

Report submission: Friday, last teaching day of Semester 2, midnight, electronic format

Write a program for the supplied ARM-XILINX development board and implement a *Time Triggered Cooperative Scheduler* (TTCS) running several *short periodic tasks* and having protection from *overruns.*

Aim: to enforce understanding of the concept of real-time scheduling as an approach to implementation of *concurrent* computational processes in a simple embedded system.

Platform: ARM-XILINX development board with a single ATMEL AT91SAM9261 microcontroller, custom monitor-debugger KMD and the tool chain based on GCC compiler.

Programming language: C.

Scheduler specification: timer tick period 1ms, eight short periodic tasks, four of them are executed in every tick interval, two tasks are executed in every odd tick interval and two tasks are executed in every even tick interval. In the case of an overrun, the running task is interrupted by the same timer which is used to form the ticks, which is followed by the normal sequence of task execution in the following tick period. The routine of overrun recovery must have the minimum impact on the timing of the tasks. The tasks positioned in the queue after the overrunning task may not be started under the overrun conditions, but their reference data (the period and phase of LED flashing) must be kept correct. Test the overrun protection mechanism and demonstrate that it works.

Recommendation – as the first step implement the TTCS scheduler without overrun protection. Use the provided FSM model as the top-level specification of the algorithm of the scheduler. Add overrun protection at a later stage. Separate calculation of timing reference data from each task, thus effectively splitting a task into (a) reference data processing (counters, which are unlikely to cause an overrun) and (b) all other functions. Place the reference data processing tasks earlier in the task queue in order to minimise the risk of their corruption due to overruns. Implement the task queue as a global array of task parameters (structures in C) and a dedicated global variable pointing at the next task to run (index of the next array element).

Task specification: implement eight very similar tasks, each of them controlling a single LED. The LEDs must flash with the periods of 1.0s, 1.1s, 1.2s, 1.3s, 1.4s, 1.5s, 1.6s and 1.7s correspondingly. In each period of flashing an LED must gain and reduce its brightness gradually, which is implemented by using *Pulse Width Modulation* (PWM). Use the *PWM period* of 20ms in order to exploit the inertia of human vision. Implement the maximum possible number of brightness levels for each task.

General comments: The first step is to find how to represent a continuous process as a sequence of short periodic tasks. It is a good idea to start debugging the system with only two tasks. Thus, you will avoid possible problems with overruns. Switch to a very slow tick when debugging the PWM. Try to move the time consuming computations away from the critical cycle.