ECSE426 - Microprocessor Systems

Winter 2017

Lab 4: Multithreaded, interrupt-driven sensor reading and peripheral control.

This lab is aimed at designing a multithreaded system that uses CMSIS-RTOS (RTX based) for initiating and maintaining multiple threads of computation and handling the exception types concurrently. You will be using the temperature and accelerometer sensors to perform a seamless superset of sensing and displaying functions from previous labs. The core of the previous two labs remains the same though the output display will be slightly changed. A 7-Segment clock display will interchangeably display the processed outputs of the thermometer and the entered number from the keypad, where you will switch between the two *modes* whenever a keypad button is pressed. You have to design a system where all the functions are seamlessly integrated in multiple threads of computation and Interrupt Service Routines (ISRs).

(1) Modifications to Lab 2 and Lab 3

As a first step, we suggest you modify the codes of Lab two and Lab three per the new requirements below before combining both labs in a multi-threaded RTOS based design. This will help you make sure the new modifications are actually working on an individual level (module/unit testing).

The system has two main modes, and the user can switch in between them by pressing a keypad button of your choice. The colored lines denote modifications from the original requirements you did in previous labs.

1. **Temperature mode (Lab 2 Component):** In this mode, the 7-Segment module will display the current processor temperature as before. The overheating alarm feature will also be retained. *However, instead of an LED-based alarm, you are required to make the 7-segment flash the displayed value on and off repeatedly to denote danger levels*. The measurement of the temperature will be done concurrently in the background while computing the orientation of the board in the other mode. This means that the computation will be running all the time, but will only be displayed when you are in the temperature mode. *SysTick cannot be used as well as it is reserved for the RTOS.* Note that even if we are in the other mode (Accelerometer), if overheating occurs, the alarm effect will still be triggered in the same way on the 7-Segment display regardless of what is being displayed on the module.

2. **Accelerometer mode (Lab 3 Component):** In this mode, the default function conveys information about the board's tilt angle. Similar to lab three, you will read the MEMS sensor data, calibrate and filter the readings. However, you are to compute the two angles of the board (both roll and pitch). In this mode, whenever the user press any button, the

display should switch to the mode of getting the destination angles. By pressing last # the display should back to the temperature mode. Note that the tilt angle should flash on and off when the temperature value computed in the background reaches dangerous levels.

(2) Lab 4 – RTOS Threads

Finally, the threads of computation should be designed using CMSIS-RTOS primitives such as threads, timers, ISRs and any required OS services (mutexes, semaphores and queues ... etc). In filtering the sensor data, you will use the FIR filter as before. The sensor data should also be calibrated.

Requirements checklist

To summarize, the tasks you need to do are:

1. Calibration and filtering for the sensors (retain the same sampling rates from previous labs),
2. Upon a keypad button press, switch between the two modes:
   - Temperature display / Overheating alarm (both modes)
   - Accelerometer and Timer (PWM mode)
3. All above functions are to be implemented using CMSIS-RTOS.
4. You have to present a graph in demo time which shows how your application is divided into threads, functions, ISRs … etc.

(3) Debugging in real time

Since you will be using CMSIS-RTOS and the underlying OS (RTX), you have to be capable of debugging your code and understanding how to simplify it in face of a sizeable body of OS code. Furthermore, most processing power will be directed towards periodic tasks - interrupt- and exception-driven.

(5) Demonstration and early demo

You will need to demonstrate that your program is correct, and explain in detail how you guarantee the specified operation, and how you tested your solution. You will be expected to demonstrate a functional program and its operational performance in all situations. The final demonstration will be on Friday, March 17th. **Note this is a ONE WEEK lab.** Note that there are **no early demo bonuses** for this lab as it is a one week lab.

(6) Lab Report

The report of this lab must include structured explanation of the work done in the lab. The report should contain the explanation and validation of all major design decisions (e.g., task division into threads) throughout the lab. Focus on RTOS system threads design. And also include all material related to timer(PWM), accelerometer, calibration, filtering within the context of Lab 3. That is, report 2 should cover the material of Lab 3 and Lab 4.
The **report and project files** are to be submitted by **Monday, March. 20th** before midnight.