ECSE426 - Microprocessor Systems

Fall 2015

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 accelerometers, where you will switch between the two *modes* by pressing a keypad button. 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 required modifications are simple and should be done in one day.

The system has two main modes, and the user can switch in between them by pressing a keypad button of your choice. The coloured 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 in degrees Celsius. The overheating alarm feature from the previous lab 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 the temperature mode is selected. *The servo motor will not be used in this experiment. 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, the angle to be displayed will be chosen through keypad input keys (Key 1 or 2). 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 moving average filter as before. The sensor data should also be calibrated. An important property is that the system should perform regardless of combination of keypad input, the displays, the current orientation of the board and the temperature change, without any assumptions on current orientation (should work if placed upside-down), or any of the external changes. As in the previous labs, you should do the minimal amount of work in ISRs for best results.

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 pre-selected keypad button press, switch between the two modes:
   - Temperature display / Overheating alarm (both modes)
   - Accelerometer display
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. When validating and debugging your solution, it is especially important to use non-obtrusive methods provided by Cortex M4 and Keil feature set. Your real-time tracing should not adversely affect the program execution.

Refer to the tutorial slides to learn about Keil support for visualizing and debugging thread operation. Keil offers a visual display which will show how threads are temporally scheduled. It also has tools to show the states of each thread, its total stack usage among other important parameters.

(4) Proving and demonstrating your solution

You are expected to demonstrate the resulting program in real time, as per the above specification, as well as the proper interleaving of independent processing threads and

asynchronous events. In addition to using the SWD debug interface, you should be ready to provide the evidence that your solution meets the specification. In particular, y**our solution should be safe when dealing with multiple threads of operation**. For example, you should satisfy the mutual exclusion access property on shared variables if necessary. It should also meet the same timing constraints as before (sampling frequencies of Lab 2 and Lab 3). You are required to keep the continuity of operation and to determine angles from the data available.

(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, Nov, 6th. **Note this is a ONE WEEK lab.** As after that, you will be given four weeks or so to carry out the project.

Note that there are **no early demo bonuses** for this lab as it is a one week lab.

(6) Lab Report

The report must include structured explanation of the **new** 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. The **report should not repeat any parts already covered in labs two and three**. Focus on RTOS system threads design. Only include parts not covered in previous reports.

The report is to be submitted by Monday, Nov. 9th before midnight.