ECSE426 - Microprocessor Systems

Winter 2013

Lab 4: Multithreaded, Interrupt-Driven Multi-sensor Detection

(1) The Lab

This lab is aimed at designing a multithreaded system that uses CMSIS-RTOS for initiating and maintaining multiple threads of computation and the exception types executed concurrently. You will be using the temperature and accelerometer sensors to perform a seamless superset of sensing and displaying functions from previous labs. The LED lights will interchangeably display the processed outputs of the thermometer and the accelerometers, where you will switch between the two *modes* by taping the board. In each mode, the two displays of the functions from the previous labs as well as this one will be triggered by pressing the user button. You have to design a system where all the functions are seamlessly integrated in multiple threads of computation and Interrupt Service Routines (ISRs).

The system will have two main modes, which switch in between when a **tap** is detected:

1. **Temperature mode:** In this mode, LEDs will either display the change trend in temperature over a scale of 8 degrees with wrap around, or flash all LEDs (as in Lab 2) whether the button was pressed to change from one sub-mode to another. The measurement of the temperature will be done concurrently in the background with determining the orientation of the board in the other mode. This means that the computation will be running all the time in the background, but will only be displayed when the accelerometer mode is selected.
2. **Accelerometer mode:** In this mode, the default function conveys information about the board's dominant angle. As a result, the LED nearest to the edge making the largest positive angle with the horizontal will be on. As you can see, you might have two LEDs lit on at a time if your board is tilted in two directions. However, you need to add a suitable threshold, say 10º before lighting any LED. When the board is horizontal or the tilt is below the threshold for all angles, then all lights are off.

   When the button is pressed, you will switch to movement mode, which is similar to the first mode but deals with the board under motion rather than rotating in 3D place. You will track the direction the board moves in 3D space, and light the LEDs to convey the direction of the board movement. The difference is that the acceleration value you read is the sum of two components, gravity and actual linear acceleration. Since you need to only consider the dominant movement direction, you need to cancel out the gravity effects as it might affect your results. You are expected to filter of gravity effects by any means you find possible.

In the previous lab, results red from the accelerometer were observed in the debugger

window, however; in this lab you will implement your own Board User interface (using the LEDs) to convey accelerometer information (movement direction). The design of User Interface (UI) is partly left to your ingenuity – you are free to implement the actions that are not explicitly specified in your way, but with the objectives of providing the *consistent interface* and having an *intuitive operation* for the users. You should provide the best overall UI experience and provide the most easily accessible information to the users.

Rather than having the ARM microcontroller perform read operations to obtain results from the accelerometer, you will use the Direct Memory Access (DMA) features of the microcontroller to accomplish this task. The microcontroller will initiate a read operation, the accelerometer will then write the desired data to a specified portion of memory. When this data transfer is complete, an interrupt will be set off from the DMA controller, informing the microcontroller that the read operation is finished and the data available. The purpose of implementing DMA is to allow the microcontroller to perform other tasks while a data transfer is happening.

The threads of computation should be designed using CMSIS-RTOS primitives using threads, timers, ISRs and any required OS services. 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 taps, clicks, the current orientation of the board and the temperature change, without any assumptions on current orientation (should work if placed upside-down), temperature change, or any of the external changes.

You should ensure that a button click does not adversely impact the tap detection, i.e., that a click on the user button does not get interpreted as a tap. As in the last lab, you should do the minimal amount of work in ISRs for best results.

(2) Requirements checklist

To summarize, the tasks you need to do are:

1. Calibration and filtering for the sensors (retain the sampling rates from previous labs),
2. Detect **single taps** and use them to switch between the two main modes (Temp and Accl),
3. Temperature mode, upon button press switch between sub-modes:
    - Temperature trend display (8 degrees with wrap around),
    - Flashing mode,
4. Accelerometer mode, upon button press switch between sub-modes
    - Dominant angle display,
    - Board movement direction display (design your own UI, filter gravity),
5. Reading accelerometer data should now be through the DMA controller,
6. All above functions are to be implemented using CMSIS-RTOS.

(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.

(4) Proving and demonstrating your solution

You are expected to demonstrate the resulting program in real time, as per 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. Your solution should also be safe that is, 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, even when the tap is detected.

(5) Demonstration

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 operation performs in all situations. The final demonstration should take place on Monday, Feb. 25. However, due to midterms in your other courses, you will be allowed to demo the lab during the whole week of Feb. 25 (until Feb. 28). Note, that Lab 5 will start on Feb. 26, and can be demoes on Monday after the Winter Break.

(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., filtering, sampling, detecting, displaying and handling tapping) throughout the lab. You should pay attention to describing and evaluating the final UI design, including a short description of what UI components you have changed.

The report is to be submitted by Thursday, Feb. 27th. Note, that there is a possibility of the extension of the report into first days of the reading week, if requested.