ELEC 6260 Final Project – Real-Time Operating Systems (RTOS)

Due: Friday, May 1, 2015 by 4:00 pm  (No late assignments will be accepted.)

Exercise the ARM/CMSIS-RTOS Real-Time Executive in a design that utilizes the LIS302DL 3-axis MEMS accelerometer on the STM32F4-Discovery board, demonstrating it with the following set of tasks.

<table>
<thead>
<tr>
<th>LED3</th>
<th>LED3 (orange - PD13)</th>
<th>X-axis tilt</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED4</td>
<td>LED4 (green - PD12)</td>
<td>Y-axis tilt</td>
</tr>
<tr>
<td>LED5</td>
<td>LED5 (red- PD14)</td>
<td>Z-axis tilt</td>
</tr>
<tr>
<td>LED6</td>
<td>LED6 (blue- PD15)</td>
<td>variable rate blinking</td>
</tr>
</tbody>
</table>

**Basic operation:**
1. One or more of LED3-LED4-LED5 should turn on if there is “significant” tilt of the board in the X, Y, and Z directions, respectively.
2. LED6 is to blink at a rate corresponding to which axis has the largest tilt.
   - X-axis: blink on and off every ½ second
   - Y-axis: blink on and off every 1 second
   - Z-axis: blink on and off every 2 seconds

**Implementation**
- RTX threads are to be used for schedulable processes
- RTX message queue functions are to be used to send and receive messages.
- RTX signal event functions are to be used to signal and detect events.

1. One thread must be responsible for reading the accelerometer data and sending messages to other threads, with these messages containing the X, Y, and Z axis values. The accelerometer should be read 100 times per second, corresponding to its lower output data rate (100Hz). The accelerometer thread should be scheduled to execute at this rate, with the thread suspended between executions.
2. There must be separate threads to control each of LEDs 3, 4, and 5, corresponding to accelerometer axes X, Y and Z, respectively.
   - Each of these threads is to be suspended until it receives a message from the accelerometer thread. The message contains the new accelerometer value.
   - If the accelerometer value exceeds some threshold (you should determine a suitable threshold experimentally), the corresponding LED should be turned on. Otherwise, the LED should be turned off.
   - After turning its LED on/off, the thread should signal an “event” to the other two LED-control threads, and then suspend itself until the other two LED threads have likewise signaled to this thread that they are finished.
   - When all three LED tasks have finished, each should send its respective accelerometer value in a message to the LED6 thread, and then suspend itself until a new message is received. (See alternative approach in #4 below)
3. One thread is to control LED6. This thread should execute with a period equal to half of the blinking rate (i.e. it should execute every time LED6 is to be toggled.) If messages have been
sent from the other three LED tasks, this task should determine which axis has the largest accelerometer reading, and set the blinking rate as indicated above.

4. **ALTERNATIVE FOR LED6-CONTROL THREAD** – Instead of each of the other three LED-control threads sending messages with their respective accelerometer readings to this thread, you may have the accelerometer thread determine the maximum of the three axis acceleration values, and then send a message to the LED6-control thread indicating which axis has the largest value. (This would eliminate a couple of message queues.)

**Key resources for the accelerometer:**

LIS302DL/LIS3DSH data sheets and application note (on the course web page).

Drivers for the LIS302DL accelerometer device and STM32F4-Discovery board, in the Board Support Package (BSP) director of the Keil “pack” for our microcontroller.

The HAL_SPI driver for the STM32F4xx microcontrollers.