Watchdogs

Please follow the steps precisely in order to complete the objectives of the assignment.

1. Create a **producer** task that takes 1 temperature sensor value every 1ms.
   - After collecting 100 samples (after 100ms), compute the average.
   - Write average value every 100ms (avg. of 100 samples) to a **sensor queue**.
   - Use medium priority for this task (see `util/rtos.hpp`)
   - See `examples/Temperature` for an example on how to use the temperature sensor driver.

2. Create a **consumer** task that pulls the data off the **sensor queue**.
   - Use infinite timeout value during queue receive API
   - Open a file, **sensor.txt**, and append the data to an output file on the SD card.
   - Save the data in this format: `printf("%i, %i\n", time, temperature);`
   - Note that if you write and close a file every 100ms, it may be very inefficient, so try to come up with a better method such that the file is only written once a second or so ...
   - Use medium priority for this task.

3. At the end of the loop of each task, set a bit using FreeRTOS event group API.
   - At the end of each loop of the tasks, set a bit using the `xEventGroupSetBits()`
   - Task 1 should set bit 1, Task 2 should set bit 2 etc ...

4. Create a **watchdog** task that monitors the operation of the two tasks.
   - Use high priority for this task.
   - Wait 1 second for all of the task bits to be set. If there are two tasks, wait for bit1, and bit2 etc...
   - If you fail to detect the bits are set, that means that the other tasks did not reach the end of the loop.
   - In the event that a task failed to set its event group bit, append to a file, **stuck.txt**, with the information about which task is "stuck"
   - Open the file, append the data, and close the **stuck.txt** file to flush out the data immediately.

5. Create a terminal command to "suspend" and "resume" a task by name.
   - "task suspend task1" should suspend a task named "task1"
   - "task resume task2" should suspend a task named "task2"
   - Use the `examples/Commandline` project as an

6. Run the system, and under normal operation, you will see a file being saved with sensor data values.
   - Plot the file data in Excel to demonstrate the variation in the data.
   - Make sure the data has some variation. This can be done by touching the top of the temperature sensor.

7. Suspend the producer task. The watchdog task should display a message and save relevant info to the SD card.

8. Let the system run for a while, and note down the CPU usage in your text file.

What you created is a "software watchdog". This means that in an event when a loop is stuck, or a task is frozen, you can save relevant information such that you can debug at a later time.

> You may use any built in libraries for this lab assignment.

> For File I/O refer examples/FileIO project
> And the API documentation here: http://elm-chan.org/fsw/ff/00index_e.html

> **Extra Credit**
> Every sixty seconds, save the CPU usage info to a file named cpu.txt. See command "info" as a reference.

Revision #2
Created 5 years ago by Khalil Estell
Updated 5 years ago by Khalil Estell

?

?

?