Assignment 4 Thread Programming and Device driver in Zephyr RTOS (100 points)

In this assignment, you are required to develop a distance sensor channel for HC-SR04 and an I2C-based EEPROM device driver (a Microchip 24FC256 CMOS EEPROM) in Zephyr RTOS running on Galileo Gen 2. The sensor channel module, named "hc-sr04", should be placed in /Zephyr/drivers/sensor and the EEPROM driver should be included in /Zephyr/drivers/flash directories. When the Galileo board is booted up, two instances of HC-SR04 sensor instances (HCSR0 and HCSR1) and one instance of EEPROM device should be created. Here are few requirements:

1. Configuration options and Makefile should be added for kernel build and should be properly defined for the Zephyr kernel on Galileo board. For instance, you can add configuration parameters in Kconfig file for Galileo board (/Zephyr/boards/galileo/Kconfig). Example configuration options include the names of the sensor and the EEPROM (HCSR0, and EEPROM_0) and any IO pin configurations.
2. The HC-SR04 sensor driver must follow the api of Zephyr sensor subsystem, defined in /Zephyr/include/sensor.h, such that applications can invoke *sensor_sample_fetch()* and *sensor_channel_get()* to read in distance measures from HC-SR04 device.
3. The device driver for 24FC256 should follow the api of flash devices given in /Zephyr/include/flash.h.

Once you have the sensor module and the EEPROM drivers, you will develop an application that takes the distance measure and records the measure plus timestamp in an EEPROM device (24FC256 EEPROM). Your implementation should be added to /Zephyr/samples/HCSR_app and should meet the following requirements:

1. Use a shell module to accept commands from console:
   - Enable n (n=0, 1, or 2, to enable none, HCSR0, or HCSR1)
   - Start *p* (to erase all pages of the EEPROM and to start collecting and recording distance measures from page 0. The operation stops until *p* pages of EEPROM are filled, where $p \leq 512$.)
   - Dump p1 p2 ($p1 \leq p2$, to dump (print out) the distance measures recorded in pages *p1* to *p2* on console)
2. The EEPROM are connected to i2c bus of Galileo board via digital pins 18 and 19 of Arduino IO connector. Hence, your application can use the existing i2c driver in Zephyr to read from and write to the EEPROM.
3. To enable concurrent data collection and recording, dual buffers (each is with EEPROM page size of 64 bytes) should be allocated. While current measures are saved in one buffer, the other buffer is used for writing to EEPROM.
4. The distance measurement should be done as quick as the hardware allows and the buffer is available.
5. The distance measure is a 32-bit integer in centimeter. The timestamp is another 32-bit integer which represents the elapse time since the beginning of the measurement. You can use x86 TSC to compute the elapse time in microsecond.

Your application should maximize the bandwidth of recording, i.e., to recording *p* pages of distance measures in a minimal amount of duration. The secondary optimization goal is to have an equal interval between two consecutive distance measurements while the bandwidth is maximized. You are free to choose multiple threads to perform the operations. There should not be any changes to the existing Zephyr source program files, except configuration options.

Due Date

The due date for both parts 1 is 11:59pm, April 27.

What to Turn in for Grading

- Create a working directory, named "cse522-teamX-assgn04", for the assignment to include your source files (.c and .h), config files, readme, and a short report (in pdf format) to describe your approach of optimizing the recording operations. Compress the directory into a zip archive file named cse522-teamX-assgn04.zip. Note that any object code or temporary build files should not be included in the submission. Submit the zip archive to Blackboard by the due date and time.
- Please make sure that you comment the source files properly and the readme file includes a description about how to make and use your software. Don't forget to add each team member's name and ASU id in the readme file.
- There will be 20 points penalty per day if the submission is late. Note that submissions are time stamped by Blackboard. If you have multiple submissions, only the newest one will be graded. If needed, you can send an email to the instructor and TA to drop a submission.
- Your team must work on the assignment without any help from other teams and is responsible to the submission in Blackboard. No collaboration between teams is allowed, except the open discussion in the forum on Blackboard.
- Failure to follow these instructions may cause deduction of points.
- Here are few general rule for deductions:
    - No make file or compilation error -- 0 point for the part of the assignment.
    - Must have "–Wall" flag for compilation -- 5-point deduction for each warning.
    - 10-point deduction if no compilation or execution instruction in README file.
    - Source programs are not commented properly -- 10-20-point deduction.
- ASU Academic Integrity Policy (http://provost.asu.edu/academicintegrity), and FSE Honor Code (http://engineering.asu.edu/integrity) are strictly enforced and followed.