Assignment 2   Thread programming and device driver in Zephyr RTOS (100 points)

In this assignment, you are required to develop a HC-SR04 sensor driver in Zephyr RTOS 1.14.2 running on Galileo Gen 2. The driver module should be placed in /Zephyr/drivers/sensor/hc-sr04 directory. When the driver is initialized, it should create two instances of HC-SR04 sensors. Here are few requirements:

1. Configuration options and CMake file should be added for kernel build and should be properly defined for the Zephyr kernel on Galileo board. Example configuration options include the names of the sensors (HCSR0 and HCSR1), the gpio pins to be used as echo and trigger pins. For Galileo board, the default trigger and echo pins are IO1 and IO3 for HCSR0, and IO10 and IO12 for HCSR1, respectively.
2. The device works only in one-shot mode. The sensor driver api should consists of the following functions:
   - *sensor_sample_fetch:* the sample fetch call triggers a new measurement if there is no on-going measurement operation. The measured distance (centimeter in a 32-bit integer) and a timestamp (a 64 integer collected from rdtsc) should be saved in a per-device buffer for the following read. The buffer should be over-written by the next *sample fetch* call.
   - *sensor_channel_get*: a distance measure saved in the per-device buffer is returned. If the buffer is empty, the channel get call is blocked until a new measure arrives. Note that, if there is no on-going sample fetch operation, the device should be triggered to collect a new measure. After each channel get operation, the buffer is cleared.
   - *sensor_attr_set*: A timeout duration parameter (in microsecond) should be provided in each HCSR sensor. A blocked channel get call should return -1 when the timeout duration is expired.

Once you have the driver for HC-SR04, you will develop an application that takes distance measure and records the measure plus timestamp in an internal buffer. Your implementation should be added to /Zephyr/samples/HCSR_app. The application should use a shell module to accept commands from console. The commands include:

1. HCSR select *n* (n=0, 1, 2, or 3 to enable none, HCSR0, HCSR1, or both)
2. HCSR start *p* (to start collecting *p* distance measures from the selected HCSR sensor(s), where $p \leq 256$. The interval between consecutive measures is set to 0.5 seconds. If more than 1 sensors are selected, the samples are collected simultaneously from all the sensors)
3. HCSR dump *p1 p2* ($p1 \leq p2$, to dump (print out) the *p1-th* to *p2-th* distance measures saved in the buffer on console. The information dumped out should include sensor device name, distance in centimeter, and timestamps.
4. HCSR clear to clear all measures in the internal buffer.

Your application and driver should avoid any busy waiting. Hence, you should use interrupt-driven approach for the echo signal of HCSR devices, and thread sleeping function. Also, the timestamp for each distance measure represents the elapse time since the beginning of the measurement. You can use x86 TSC to compute the elapse time in microsecond.

In the assignment, you have a chance to work on Zephyr gpio device driver and to develop the driver for HC-SR04. The gpio driver is based on several Zephyr components, such as gpio_dw, gpio_pcal9535a, i2c_dw, pci, and interrupt. It will be a good exercise to examine and study how these components are implemented and initialized in Zephyr source code. For the assignment, you are asked to compile a report to explain all details of setting up device objects, acquiring device parameters, and initializing devices, starting from the macro DEVICE_INIT() and DEVICE_AND_API_INIT(). Also, list all devices created in *test_led* program (that was provided in assignment 1) and the order these devices are initialized. The report should be formatted to single-space, 11-point font, and be limited to 8 pages. Code pieces from Zephyr and its build directory can be included in the report as long as the source is cited.

Due Date

The due date is 11:59pm, March 5.

What to Turn in for Grading

- Your submission is a zip archive, named RTES-LastName-FirstInitial_02.zip, that includes
    - A pdf report to include a description of your implementation. Don't forget to add your name and ASU id in the report.
    - A patch file to include all your changes in the Zephyr 1.14.0 commit for the implementation of HCSR-04 sensor driver and a sample application. The patch file should be named as *hcsr04.patch* and should be created from the root directory of Zephyr source tree.
    - A readme text file with all the commands you use. Alternatively, you could have a script file with comments inside on how to use it.
- Note that any object code or temporary files should not be included in the submission. Submit the zip archive to the course Canvas 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 your 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 Canvas. If you have multiple submissions, only the newest one will be graded. If needed, you can send an email to the instructor to drop a submission.
- The assignment must be done individually. No collaboration is allowed, except the open discussion in the forum on Canvas. The instructor reserves the right to ask any student to explain the work and adjust the grade accordingly.
- Failure to follow these instructions may cause deduction of points.
- Here are few general rule for deductions:
    - Cannot apply patch or compilation error -- 0 point for 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.

How your program will be graded:

1. Unzip your submission and apply your patch to Zephyr source code
2. cd */Zephyr/samples/HCSR_app*, mkdir build, cd build, cmake, and make
3. Run your application image on Galileo Gen 2 board
4. Examine your source code and report.