Assignment 1 GPIO and Interrupt Latency Measurement in Zephyr RTOS (100 points)

Two important performance metrics of RTOS are interrupt latency and context switching overhead. Interrupt latency is the total delay between the interrupt signal being asserted and the start of the interrupt service routine execution. This delay may be extended if an interrupt arrives when the RTOS is in a non-preemptive critical region. As for context switching overhead, it is the delay of context switching process of saving the context of the executing thread, restoring the context of the new thread, and starting the execution of the new thread.

In this assignment, you are requested to develop an application to measure interrupt latency and context switching overhead of Zephyr RTOS (version 1.14.0) on Galileo Gen 2 board. To generate interrupts, you can loop back either gpio or pwm output signals to a gpio input pin which has interrupt enabled at rising or falling edge. To trigger a context switch, you can let a thread unlock a mutex for which a thread of higher priority is waiting. To record the instants of event occurrences, you can read x86's Time Stamp Counter (TSC) which provides a much better granularity than typical OS timer.

Your application should perform 3 measurements: interrupt latency without background computing, interrupt latency with background computing, and context switching overhead. The background computing you need to test is a message passing operation between two threads via a message queue. For each measurement, 100 samples at the rate of 1 sample per 0.02 seconds should be collected and saved in separate buffers. A shell module should be included in your application which can start one of the three measurements and print out the collected samples from any one of buffer after the measurements. In addition, your program should turn on a RGB led to indicate the status of measurement, such as red for busy and green for ready.

All your application files should reside in the directory "zephyr/samples/measure_*nn*" of Zephyr source tree where *nn* is the last 2 digits of your ASU id. There should be no changes to the original Zephyr source (except the patches applied to gpio_dw.c, pwm_pca9685.c, and pinmux.c). This implies that all measurements should be done in main and/or application threads.

References

- Zephyr 1.14.0 source code: https://github.com/zephyrproject-rtos/zephyr/tree/v1.14-branch
- Zephyr documentation: https://docs.zephyrproject.org/1.14.0/
- Intel Quark processor and Galileo board documentation: https://www.dropbox.com/sh/xkftt2l3bv7csny/AADBKBHlEwL2LiHTn4jDadqqa?dl=0

Due Date

TBD.

What to Turn in for Grading

- Compress the directory "zephyr/samples/measure_*nn*" into a zip archive file named cse522-assignment_01.zip. Note that any object code or temporary build files should not be included in the submission. Submit the zip archive to the course Canvas by the due date and time.
- 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 and TA to drop a submission.
- The assignment must be done individually. No collaboration is allowed, except the open discussion in the forum on Canvas.

- Here are few general rule for deductions:
    - 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.