**Measurement in Zephyr RTOS**

**Assignment 3 : Report**

*Submitted for the Subject:*

**Real Time Embedded Systems CSE522**

*By:*

**Group 9  
Sarvesh Patil (1213353386) Vishwakumar Doshi (1213322381)**

*Under the guidance of:* **Prof. Yann-Hang Lee**



**ArizonaStateUniversity Tempe Campus,AZ**

**March,2018**

**ACKNOWLEDGEMENT**

We owe a great thanks to many people who helped and supported us during the completion of this project.

Our deepest thanks go to **Prof. Yann-Hang Lee**, Professor for the subject Real Time Embedded System who acted as our mentor and guided us throughout this project. We thank him for giving ideas and briefs for the work that had to be done.

A special thanks to **Shiksha Patel**, Teaching Assistant, Real Time Embedded Systems, for constant help and guidance in the labs as well as discussion group.

We also thank all our friends for helping us out during the downs. We respect everyone for giving their time when we were in need.

Thank You

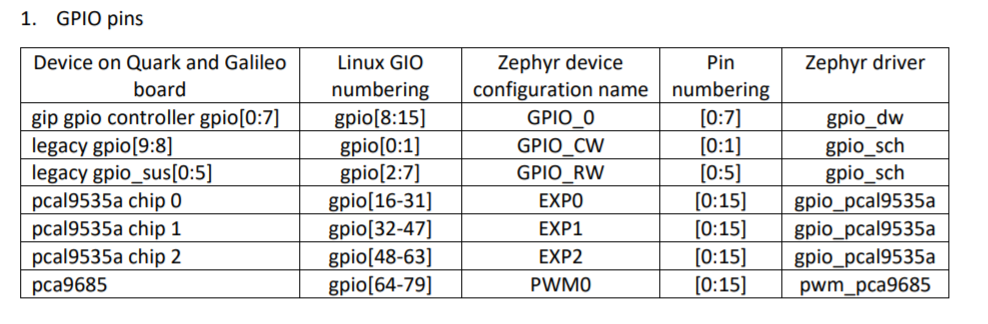
Following project is used to develop an application to measure interrupt latency and

context switching overhead of Zephyr RTOS (version 1.10.0) on Galileo Gen 2 board.

Application performs 3 measurements in sequence : interrupt latency without background computing, interrupt latency with background computing, and context switching overhead. For background computing we are using message passing between two threads via a message queue.

We are referring to below table for GPIO pin configurations & write. Pinmux.c is used to configure multiplexing.





Latency computation 1 :

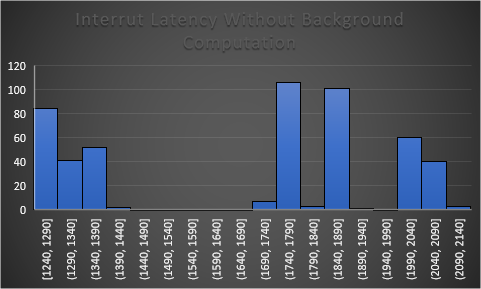
Context switching :

Two threads have been created of different priorities.   
At beginning low priority task has acquired a mutex due to which high priority task gets blocked as it cannot acquire same.

We find the timestamp at just before low priority task releases mutex & again when high priority task acquires mutex.

The difference between these two readings gives the required overhead latency value.

As we can observe, values of latency are spread between 1240 to 2140 ns

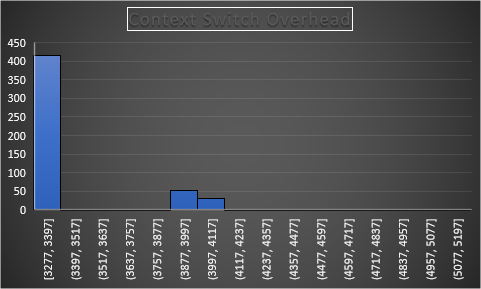


Latency computation 2 :

Interrupt Latency with PWM generating interrupts without background tasks:

In this case we are triggering interrupt periodically using inbuilt PWM functionality.

We find the timestamp first at after each instruction as PWM is in background & arrival of interrupt is undetermined & second reading is taken when ISR handler is called where we read timestamp of instruction at which preemption happened.



Latency computation 3 :

Interrupt Latency with background task computations :

Here we create 2 threads and a message queue. One thread sends message to the queue and other one receives messages. Apart from these 2 background threads, rest is same as measurement 2. Due to message sending and receiving being atomic tasks, interrupt is disabled during those tasks and ISR gets serviced on completion of these tasks.

Observation : Here we see latency values following approximately gaussian curve. Here we also observe that the latency overhead is greater than what we observed in case of context switching without background tasks because ISR waits for background tasks.

