Mauricio Gutierrez

Assignment 3: Measurements in Zephyr RTOS

For this assignment we develop a program that measures interrupt latency and context switching overhead in Zephyr RTOS on the Galileo Gen 2 Board. Interrupts are generated by configuring an input GPIO pin and another output GPIO pin. The output pin generates an output signal to the input pin which has interrupts enabled at the rising edge. On the board, I have connected the pins IO3 and IO10 together with a cable as shown in Figure 1 below.

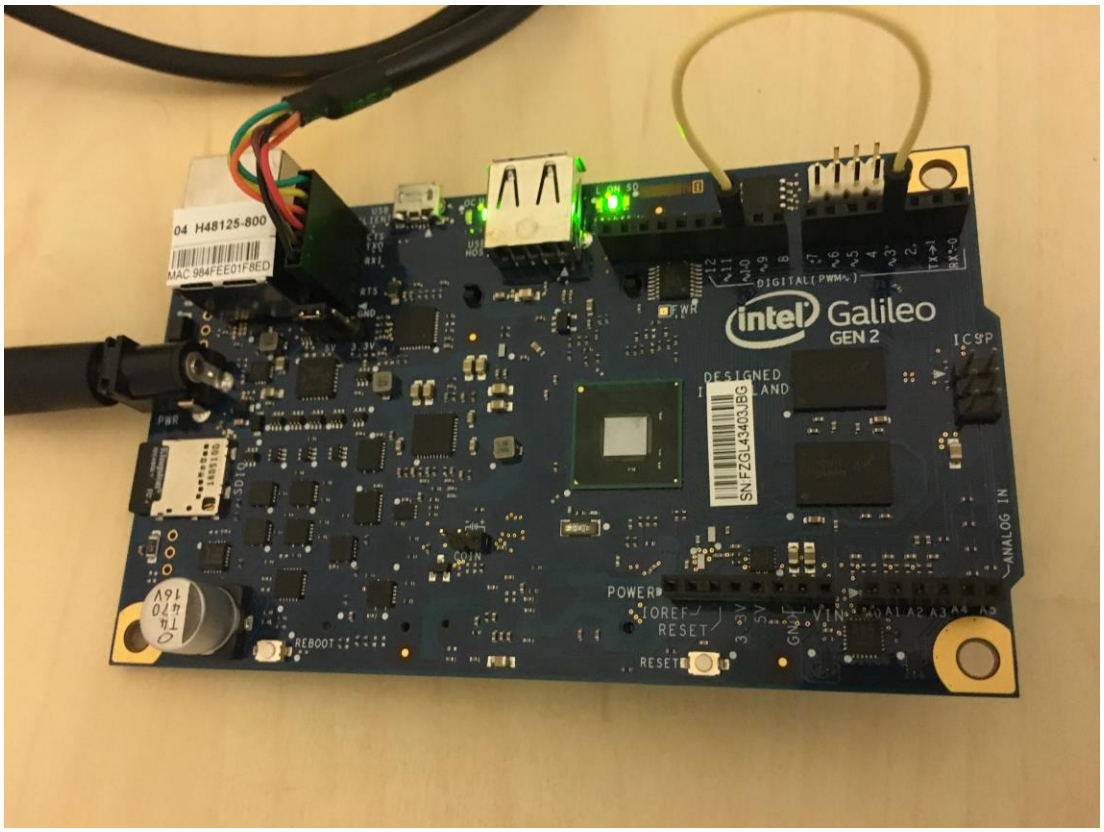

Figure 1: Board Setup

For context switching, priority threads and semaphores are used to measure both the latency of just a system call and the latency of both a call and a context switch. The former occurs when a thread of higher priority releases a lock, but is not pre-empted by the lower priority thread whereas the latter occurs when a lower priority thread releases a resource to a higher priority thread. Background computing is also added in the form of a message passing system between two threads to measure the effects it has on interrupt latency. Finally, a shell module named "measure" is provided so that the user may input commands to print the results stored in buffers.

We gather our results by logging the outputs in Putty from the Galileo Board to a file and plot 500 samples for each case to produce the histogram diagram shown in Figure 2. Table 1 shows the averaged latencies in number of CPU cycles obtained from the x86 TSC.

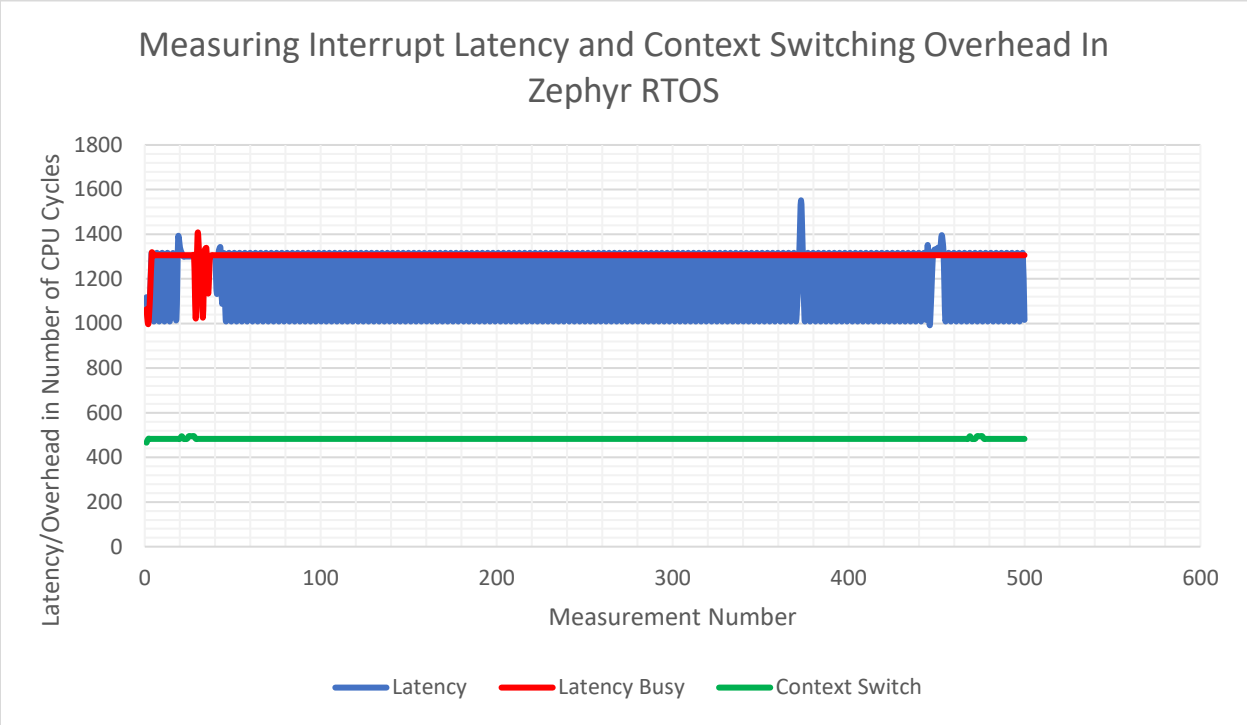

Figure 2: Histogram Diagram of measured latencies and overheads

Table 1: Average latencies and overheads

Average interrupt latency with no background computing	Average interrupt latency with background computing	Average context switching overhead
1161.276	**1303.356**	**483.206**

We see that interrupt latency can be quite erratic but is fairly well bounded as would be expected for a real-time operating system. The average interrupt latency comes out to be about 1161 cycles. However, when we add background computing we find that the latency is constantly at the very limits of the upper bound and the average rises to 1303 clock cycles. This is to be expected because we are passing messages through a message que which has a critical section to ensure no contention on reading or writing the messages. In the background, we are constantly passing messages non-stop so there would always be a reader or writer. We know that the interrupt delay will be extended if an interrupt arrives when the RTOS is in a non-preemptive critical region. In this case, the background activity considerably increases the average latency as we find ourselves constantly in the worst-case scenario.

Finally, we have the context switching overhead which is actually quite constant across all the samples and this is what we want to see from a real-time operating system. In Zephyr, context switching has an overhead of about 483 cycles which is considerably better than the

latency of interrupts. Overall, we do find that Zephyr provides the predictability and bounded timing performance required by any real time operating system.