**Objective: -**

The aim of this lab was to familiarize ourselves with different software debugging techniques. We learnt to debug issues in our code in real time and through visualization of the program’s activities. Our focus was also shifted towards the concept of dumping time and data values into arrays. The lab provided us with the opportunity to play around with the oscilloscope and the logic analyzer to debug our software without any intrusions. We experienced concepts of real time probability mass function and central limit theorem, as well as observe critical sections.

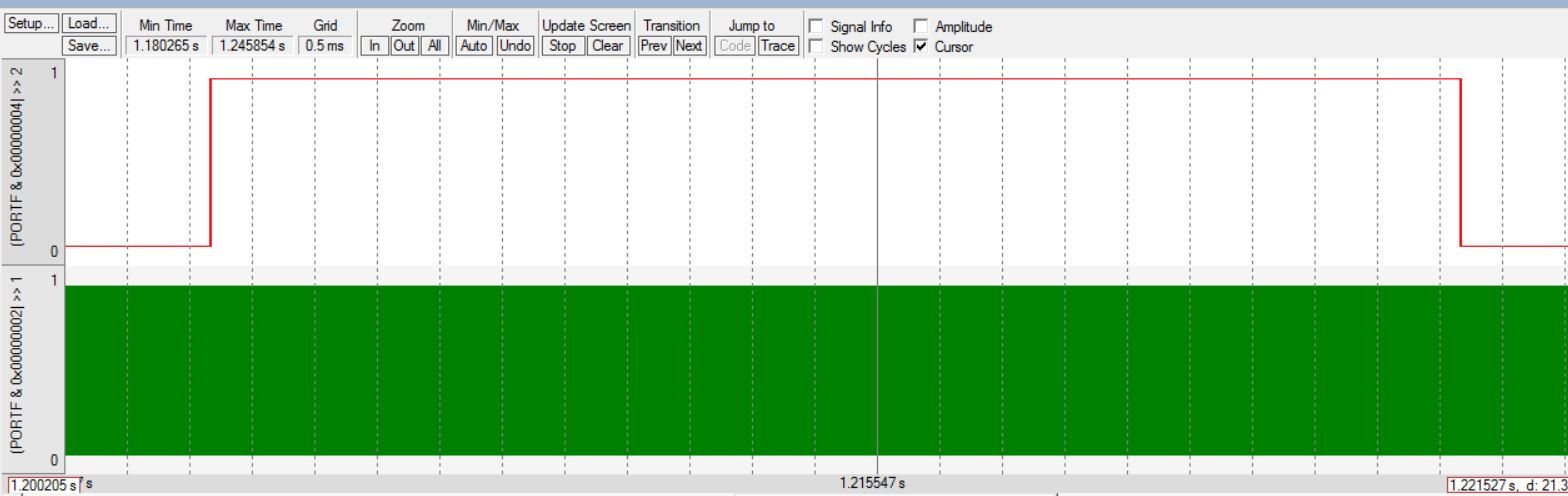
**Deliverables: -**

1. **Oscilloscope**
2. **Logic Analyzer: -**

The logic analyzer allows us to view the voltage at which values are being outputted (i.e. high, low). We calculated that within the ISR it ran for about 8.625 µm and occurred four times during the time span of 30.01 ms. Therefore, the percentage of running in ISR vs main was equal to 0.115%

1. **Critical Section**

PF2 was the incorrect pin where we observed the critical section. The critical section was observed when the Interrupt Service Routine was triggered when main was changing the PF2 bit. When both main and ISR attempt to write to the PF2 bit, ISR having higher priority would change it, and when it completes running, main would change it back to the original value. One way to fix this problem would be to use another port, and not allowing there to be a common global variable.



1. **Time Jitter**
2. PMF data and discussion of results. Does your data support CLT? If not why?
3. Debugging profile of execution time in ISR with hardware averaging. Why is it different

**Analysis and Discussion**

1. The ISR flipping PF2 is minimally intrusive because of what we have seen from the deliverables, that there is barely any delay to the overall function.
2. Dumping data is much faster than using a printf statement because calling a function and then printing the data on the screen is slower than simply adding the data into an array. Printf statement is better than data dumping in terms of memory and it does not take any of the board’s memory to print to the screen as opposed to allocating locations for each data being sampled.
3. Critical sections occur when two threads try to access the same global variable. For example, Interrupts can cause critical sections if other threads are trying to modify the same global variable as we have seen in LAB2.
4. Minimally intrusive is a debugging technique that changes the program but barely has any effect of the speed/performance of the program. For example, a heartbeat.
5. Hardware averaging provides us with a higher accuracy of data at the cost of time required to collect the data, and so it is not always used.