18-847 Lab 4:
Enabling Interrupts with SPI

Jawahar Chigurupati
Carnegie Mellon University, Silicon Valley
November 19th, 2019

***Abstract*— This report details how interrupts were added to our Arduino code from the last lab. The PowerDue board was then used along with a magnetic detector to capture signals that met a certain threshold and figure out how many times magnetic material was brought within range of the magnetometer. This lab required that we modify our SPI code from the last lab to implement interrupt and calibration functions.**

I. INTRODUCTION

Interrupts are a key component of embedded systems programming. By sending an interrupt signal to the processor, you can signify that an event has occurred which requires handling in some way. An interrupt service routine (ISR) would examine the sent signal and determine what course of action should be taken by the processor, and returns an interrupt value which can be used as confirmation. The code from the previous lab was updated so that if a certain threshold was met – an interrupt would be fired. The baseline magnetic signal was calibrated and then we utilized FreeRTOS tasks of varying values to be triggered in sequence when the threshold was passed.

II. APPROACH

In order to complete this lab, the previous code for the SPI section from lab 3 was re-purposed. New implementations for interrupt functions and a calibration function for the magnetometer needed to be created. Once that was complete, we could attempt to flash the code onto the PowerDue and then add tasks for the interrupt service routine to read and process the calibrated data.

The magnetometer board has a chip select pin at 51, aka GPIO. This was changed to 4 as instructed in the lab writeup and an interrupt pin was placed at 51.

Two interrupt functions were written, beginInterrupt and endInterrupt. The INT1 pin on the breakout board was used to generate interrupts on pin 51. For the ISR, we needed to ensure that no interrupts occurred while the data and tasks were being added to the FreeRTOS queues.

Calibrating the magnetometer was another important step. An interrupt should be fired if the magnetic signal passes a set threshold value. The equation used to determine the threshold limit was:

$$T = \mu + C\sigma$$

The Greek letter mu is the averaged set of values, C is a coefficient for sensitivity, and sigma is the standard deviation of the mean. This threshold value is then input to the M_THS_X_MSB and M_THS_X_LSB registers on the FXOS8700CQ board.

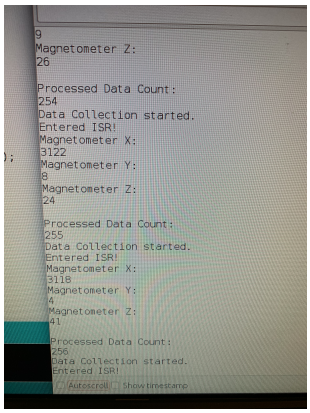

Fig. 1: Serial monitor output with the magnetometer values looping.

The chip select and interrupt pin have been set at pins 4 and 51. Within the Interrupt Service Routine function, all data was written to a queue and the number of data collections is output to the serial monitor. We then attach the interrupt to the ISR function and look for a 'FALLING' call to trigger the interrupt.

III. RESULTS

Figure 1 shows a screenshot of the Serial Monitor output as our Arduino Sketch was running. By moving a metal pin to and from the magnetometer the values outputted would vary. This fluctuation is due to the nominal magnetic field that is present on Earth. To calculate the threshold level, we collected a running sum of 20 magnetometer readings. The average and standard deviation was calculated for x, y and z readings. The calculated standard deviation was multiplied by 4 to establish the threshold level and ensure that any minor variable signals would not trigger the interrupt prematurely.

IV. CONCLUSIONS

This lab had us implement an interrupt service routine on the PowerDue so that we could use the magnetometer to measure magnetic signals. The calibration of the magnetometer and the interrupt functions were implemented without any verification errors. The biggest error during this lab was ensuring the semaphores were being taken and

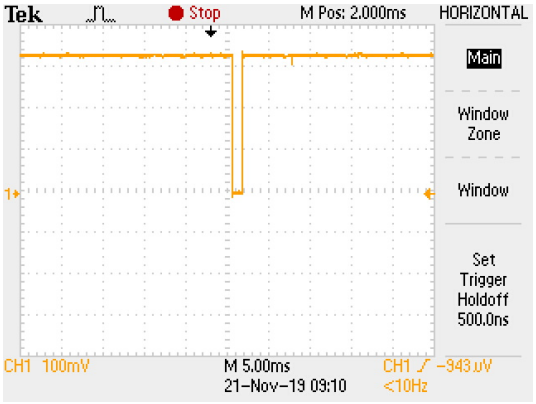

Fig. 2: Screenshot of the oscilloscope reading when the magnetometer was triggered.

given in the correct manner. I was running into a crash error which was interrupting the program's loop. This was the real debugging problem that needed to be solved or I was not able to get any oscilloscope readings. Past that, its easy to notice the rising and falling actions which is what we were measuring via interrupts.

The link to the github repository is: https://github.com/cjawahar/wssaLab4

V. References

1. S.V Kalluru Srinivas "Lab 4 Manual" [Online]
3. B. Iannuci, "The CMU Powerdue". [Online]