**EE345L – Lab 2: Performance Debugging**

Ty Winkler and Jeremiah Bartlett

02/12/16

1. **OBJECTIVE**

This lab was designed to familiarize students with useful debugging techniques involving oscilloscopes, logic analyzers, and data dumps. Each method has a different degree of intrusiveness which is critical when making an embedded system.

In this lab we used interrupts to take DAC and time-stamp samples. This information was then used to determine jitter and a PMF of the DAC.

1. **ANALYSIS AND DISCUSSION**
   1. **Intrusiveness**

In the ISR PF2 is toggled three times. This type of debugging is minimally intrusive to the program because while it does interfere with the runtime of the program. It does not add a significant amount of time in which the interrupt is run.

* 1. **Data Dumps**

Data dumps when compared to printf statements are similar in what they accomplish. Both data dumps and printfs are a way in which a user can read the results of the program both while it is occurring and after it has completed. Data dumps are useful when dealing with large amounts of data or data that is occurring at a rate that is too fast to be easily read. They are also useful as a log in case of system failure a data dump can be used to understand what was occurring before the system failed. Printfs are more commonly used for debugging small amounts of information or for helping to understand at which point the program is breaking.

* 1. **Critical Sections**

Critical Sections occur when a function attempts to read or write to a global variable while interrupts are active. Critical Sections can cause incorrect information to be stored or erased in the variable when an interrupt changes the values of different registers that were being used in the original function.

* 1. **Intrusiveness**

Minimally intrusive is a term used to describe a debugging practice in a system that has a small but negligible effect on the system it is debugging. Actions such as dumps and led toggling are commonly classified as minimally intrusive in most systems.

* 1. **Hardware Averaging**

Based on the results of the PMF we can see that when using hardware averaging the results of the values read are a lot less noisy, however, using hardware averaging greatly increases the time spent sampling data and in a system where voltage needs to be read instantaneously reading the voltage 4 to 64 times before returning the answer can greatly slow the performance of a system.

1. **Software Design Solutions**
   1. **Jitter Data Dump Excerpt**

Based off the data dump from the jitter array we can see that time between interrupts was roughly off by a degree of 16.

0x31

0x21

0x11

0x01

0xF1

0xE0

0xD0

0xC0

0xB0

0xA0

0x90

0x80

0x70

0x60

0x50

0x40

0x30

0x20

0x10

0x00

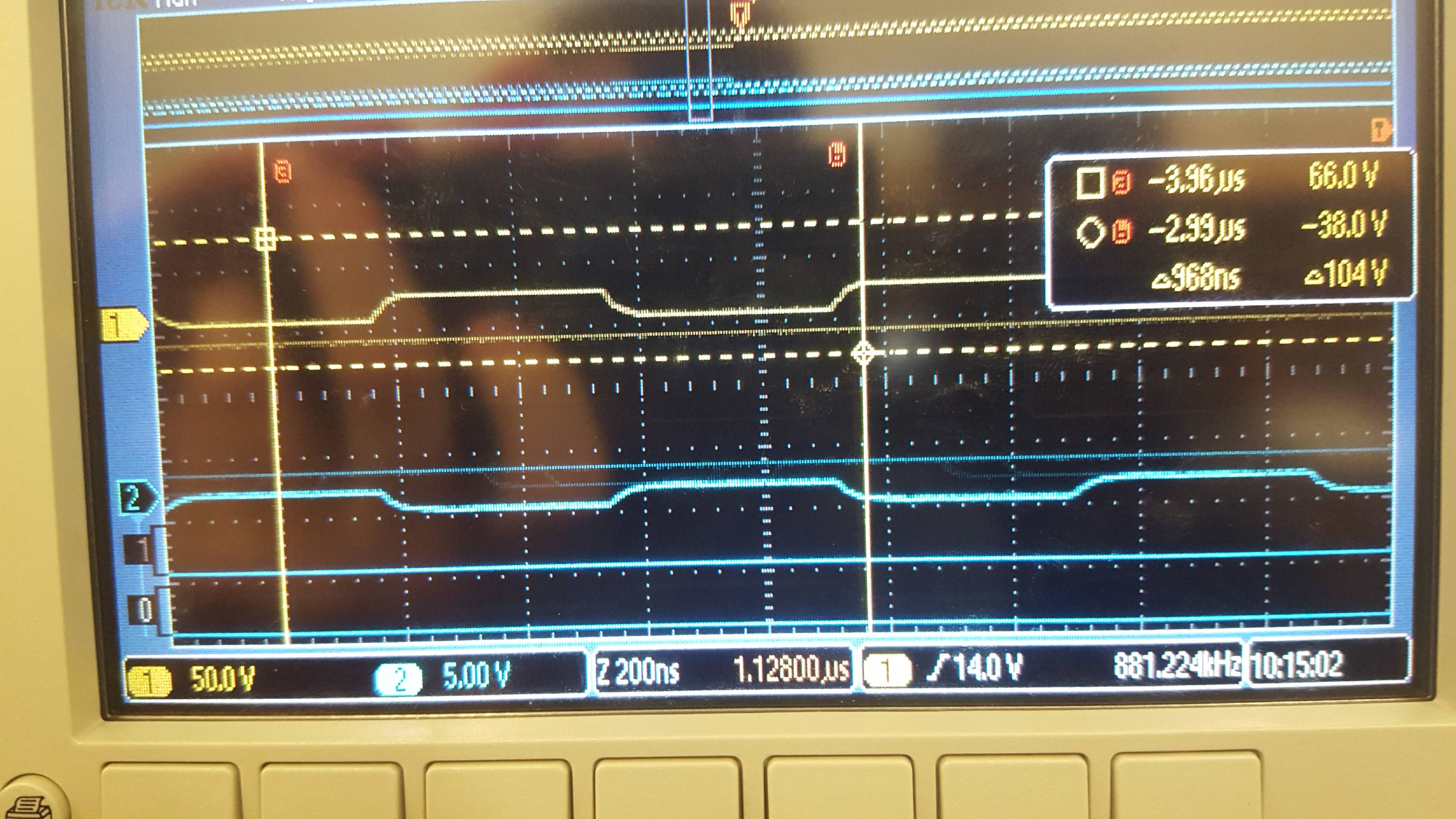
* 1. **Scope Debugging**

Scope debugging on pin PF2



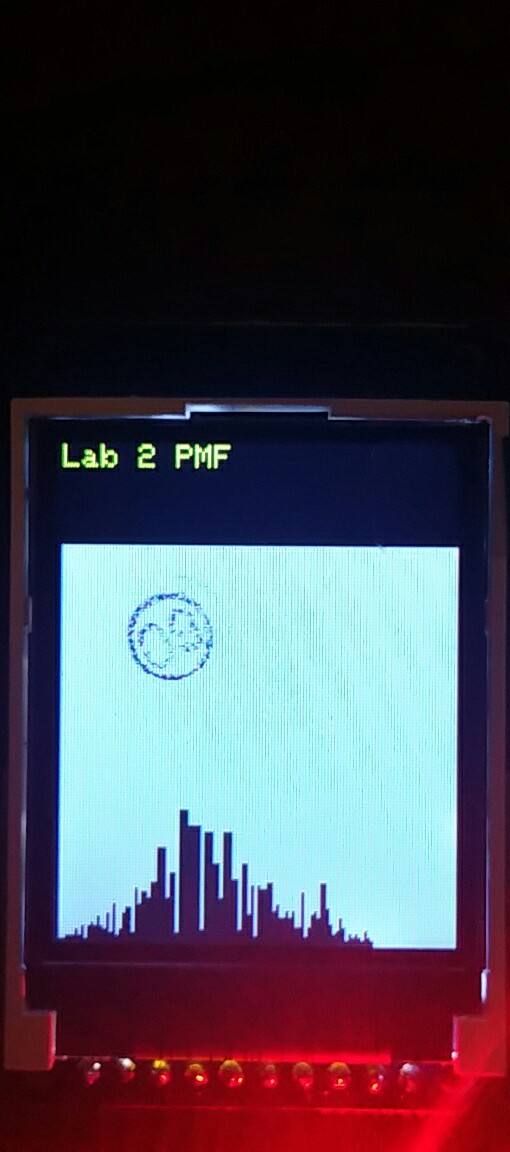
* 1. **Logic Analyzer Debugging**

Debugging PF1 vs PF2 using the scope as a logic analyzer



* 1. **PMF Results**

Results showing the resulting PMF from a 1.6V DAC input



* 1. **Data Dump w/ Averaging Enabled**

Interrupt time data excerpt. As seen from the data the time between the occurrence of interrupts has greatly increased compared to the pre averaging enabled.

|  |  |
| --- | --- |
| After Averaging:  0xFF06  0xFF09  0xFF0C  0xFF0F  0xFE13  0xFE1A  0xFE1E  0xFE21  0xFE23  0xFE26  0xFD2D  0xFD31  0xFD34  0xFD38  0xFD3B  0xFC41  0xFC45  0xFC49  0xFC4C  0xFC4F | Pre Averaging:  0x2D  0x1D  0x0D  0xFD  0xED  0xDC  0xCC  0xBC  0xAC  0x9C  0x8C  0x7C  0x6C  0x5C  0x4C  0x3C  0x2C  0x1C  0x0C  0xFC |