Kevin Tong kyt259
Som Wakdikar
17710

Lab 2 Report

OBJECTIVES

Utilize the debugging techniques such as dumping and logic analyzers along with oscilloscopes and spectrum analyzers to understand real-time systems, jitter, critical sections, bit banding and shared resources, central limit theorem, and noise.

HARDWARE DESIGN

On github: https://github.com/EE445L-FALL-2022/lab-2-som-wakdikar-kevin-tong

SOFTWARE DESIGN

On github: https://github.com/EE445L-FALL-2022/lab-2-som-wakdikar-kevin-tong

ANALYSIS AND DISCUSSION

1. The ISR toggles PF2 three times. Is this debugging intrusive, nonintrusive or minimally intrusive? Justify your answer.

This debugging style is minimally intrusive because you are changing the code but it has little to no effect on its efficiency.

2. In this lab we dumped strategic information into arrays and processed the arrays later. Notice this approach gives us similar information we could have generated with a printf statement. In what ways are printf statements better than dumps? In what ways are dumps better than printf statements?

Printf statements are better than dumps because you can see what the output is in real-time. Dumps are better than print statements because they are minimally intrusive and dont effect the systems performance.

3. What are the necessary conditions for a critical section to occur? In other words, what type of software activities might result in a critical section?

A critical section occurs when a multiple processes accesses and changes the same code segment. For example if multiple ISR's access and change the same global variable then errors might occur. To fix this one can disable interrupts when accessing the code segment.

4. Define "minimally intrusive".

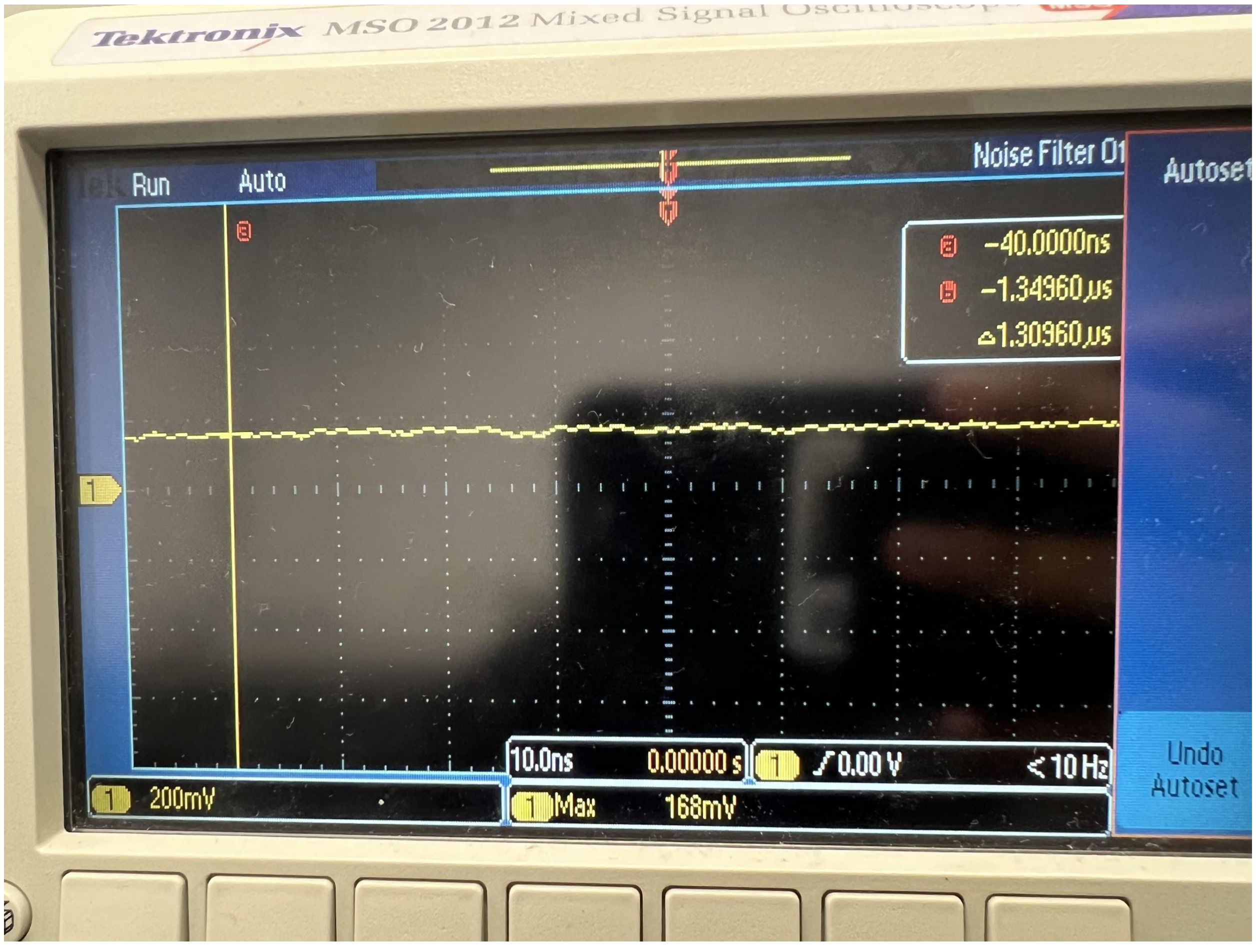
A debugging technique is classified as minimally intrusive if it has a negligible effect on on the system's performance and efficiency.

5. The PMF results should show hardware averaging is less noisy than not averaging. If it is so good, why don't we always use it?

Although hardware averaging is much more accurate, it takes a lot more time. In some cases, hardware averaging will not meet the time requirements of a project.

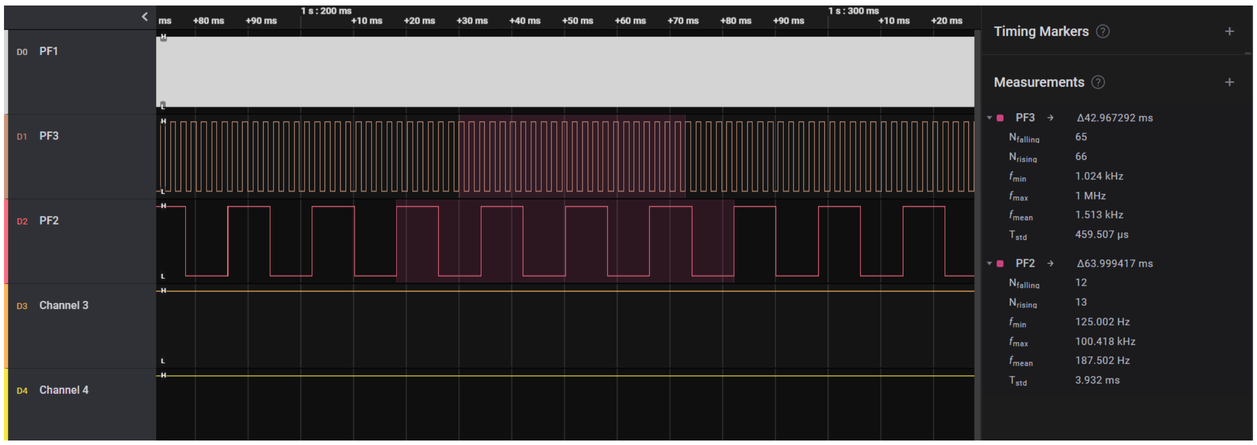
MEASUREMENT DATA

2.



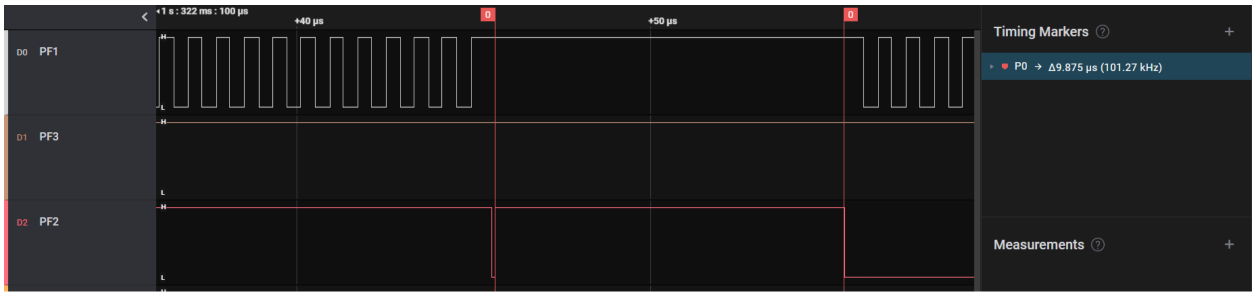
Analog voltage versus time measured with a real oscilloscope.

4.



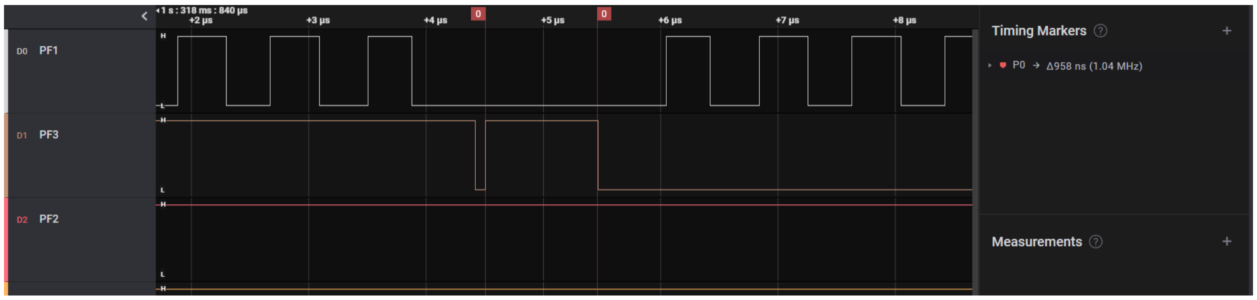
```
PF3 →      Δ42.967292 ms
N_falling  65
N_rising   66
f_min      1.024 kHz
f_max      1 MHz
f_mean     1.513 kHz
T_std      459.507 µs

PF2 →      Δ63.999417 ms
N_falling  12
N_rising   13
f_min      125.002 Hz
f_max      100.418 kHz
f_mean     187.502 Hz
T_std      3.932 ms
```

Zoomed out view of the PF1 PF2 PF3 recording to see a) the Timer0A runs at 125 Hz, b) Timer2A runs at 1024 Hz, and c) most of the processor time is allocated to running the main program.



```
P0 → Δ9.875 µs (101.27 kHz)
```

Zoomed in view of the PF1 PF2 PF3 recording to see a) the main program does not run while the Timer0A ISR is running and b) the time to execute the Timer0A ISR is about 10us (most of this 10us occurs converting the ADC) This recording was taken with ADC0_SAC_R=0.



```
P0 → Δ958 ns (1.04 MHz)
```

Zoomed in view of the PF1 PF2 PF3 recording to see a) the main program does not run while the Timer2A ISR is running and b) the time to execute the Timer2A ISR is about 1us. This recording was taken with ADC0_SAC_R=0.

CALCULATIONS

Measure P0, the interrupt period for the Timer0A (should be 1/125Hz).
8ms

Calculate the Timer0A ISR utilization percentage. This is T0/P0.
0.0012

Measure P2, the interrupt period for the Timer2A (should be 1/1024Hz).
976 us

Calculate the Timer2A ISr utilization percentage. This is T2/P2.
.00098

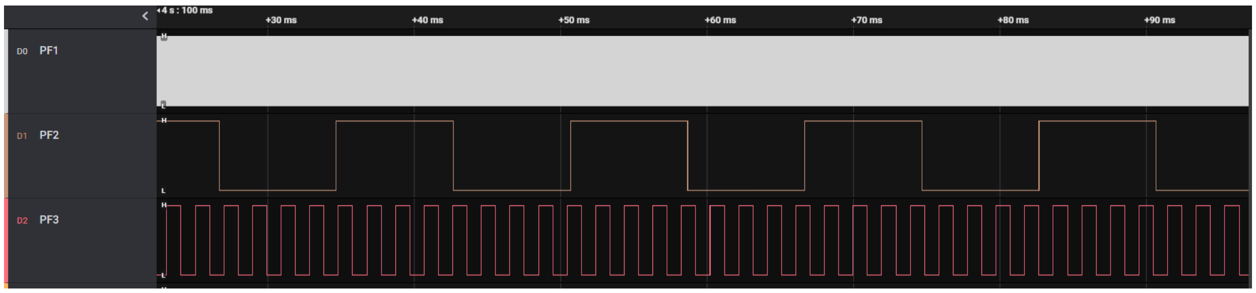
Calculate the total utilization percentage of the program. This is about 1-T0/P0-T2/P2.
.9977

Measure T0, the time to complete the Timer0A ISR (should be about 10us with ADC0_SAC_R=0).
9.875 us

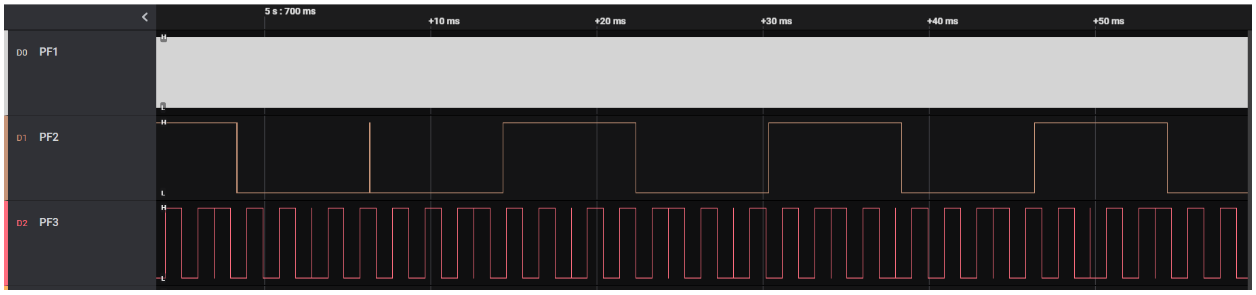
Measure T2, the time to complete the Timer2A ISR (should be about 1us, depending on your Jitter_Measure).
958 ns

5. When the adc is sampled, there will always be a lot of noise. Some places the noise comes from are mechanical heat, corrosion of the slide pot, vdd, faulty wiring, and various other sources. Due to the noise, the data must be sampled multiple times, which will make it unlikely for the interrupts to meet the time requirements which is jitter. Jitter is how off the interrupt is from the required time. Timer2A by itself is classified as real time because there are no other interrupts to disrupt its ISR. Therefore, by itself, it will have near-zero jitter.
When there are 2 ISRs, since Timer0A has a higher priority, Timer2A ISR will always be interrupted by the timer0A interrupt. The time jitter is proportional to 2^SAC because 2 ^ SAC is the amount of samples that is averaged. For example, SAC = 3 is 2^3=8 samples averaged. In this case, Timer0A is considered real time due to its higher priority as its ISR will finish serving first no matter what. Timer2A is not real time because if it gets interrupted by the Timer0A ISR, it will have to wait for the Timer0A ISR to finish running.

6.



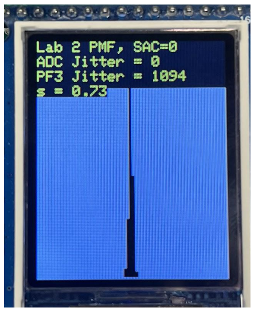
Zoomed in view of the PF1 PF2 PF3 recording before we introduced a Critical Section



Zoomed in view of the PF1 PF2 PF3 recording after we introduced a Critical Section

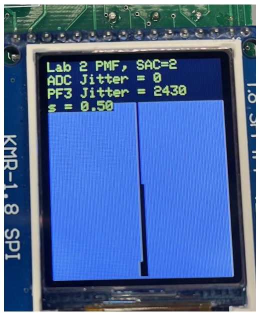
With bit specific addressing, both PF2 and PF3 heartbeats show a controlled behavior with a specific period. However, by addressing using GPIO_PORTF_DATA_R, critical sections are introduced. After introducing this change to all PF2, PF1, and PF3 variables, an uncontrolled behavior can be seen in the logic analyzer traces.

7.



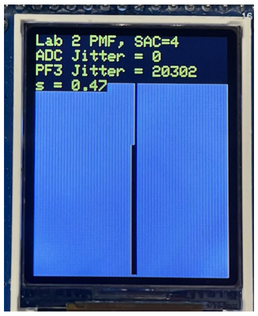
```
Lab 2 PMF, SAC=0
ADC Jitter = 0
PF3 Jitter = 1094
s = 0.73
```

Photo of main3 output with a constant voltage applied to the analog input (SAC=0).



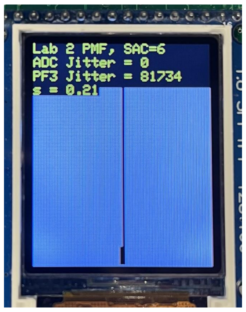
```
Lab 2 PMF, SAC=2
ADC Jitter = 0
PF3 Jitter = 2430
s = 0.50
```

Photo of main3 output with a constant voltage applied to the analog input (SAC=2).



```
Lab 2 PMF, SAC=4
ADC Jitter = 0
PF3 Jitter = 20302
s = 0.47
```

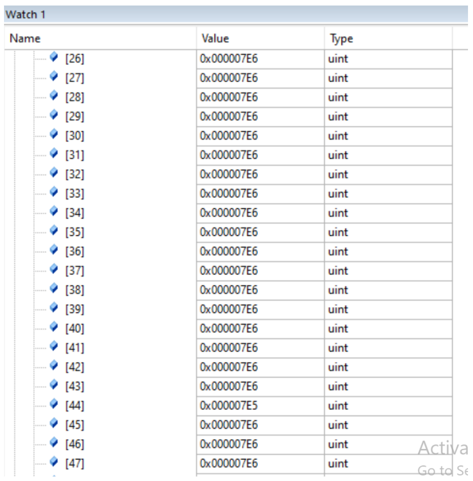
Photo of main3 output with a constant voltage applied to the analog input (SAC=4).



```
Lab 2 PMF, SAC=6
ADC Jitter = 0
PF3 Jitter = 81734
s = 0.21
```

Photo of main3 output with a constant voltage applied to the analog input (SAC=6).

8. Estimate your ADC resolution with SAC=4 (16-point averaging).



Watch 1

Name	Value	Type
[26]	0x000007E6	uint
[27]	0x000007E6	uint
[28]	0x000007E6	uint
[29]	0x000007E6	uint
[30]	0x000007E6	uint
[31]	0x000007E6	uint
[32]	0x000007E6	uint
[33]	0x000007E6	uint
[34]	0x000007E6	uint
[35]	0x000007E6	uint
[36]	0x000007E6	uint
[37]	0x000007E6	uint
[38]	0x000007E6	uint
[39]	0x000007E6	uint
[40]	0x000007E6	uint
[41]	0x000007E6	uint
[42]	0x000007E6	uint
[43]	0x000007E6	uint
[44]	0x000007E5	uint
[45]	0x000007E6	uint
[46]	0x000007E6	uint
[47]	0x000007E6	uint

A photo of DumpDataBuf, the dump data buffer, after it is filled with SAC = 4. There were only 2 different values.

ADC Resolution = number of different values / 4096 = 2/4096 = 1/2048
ADC Resolution = 1/2048