# LAB 2 REPORT

*Arpad Voros – aavoros@ncsu.edu - ECE 560*

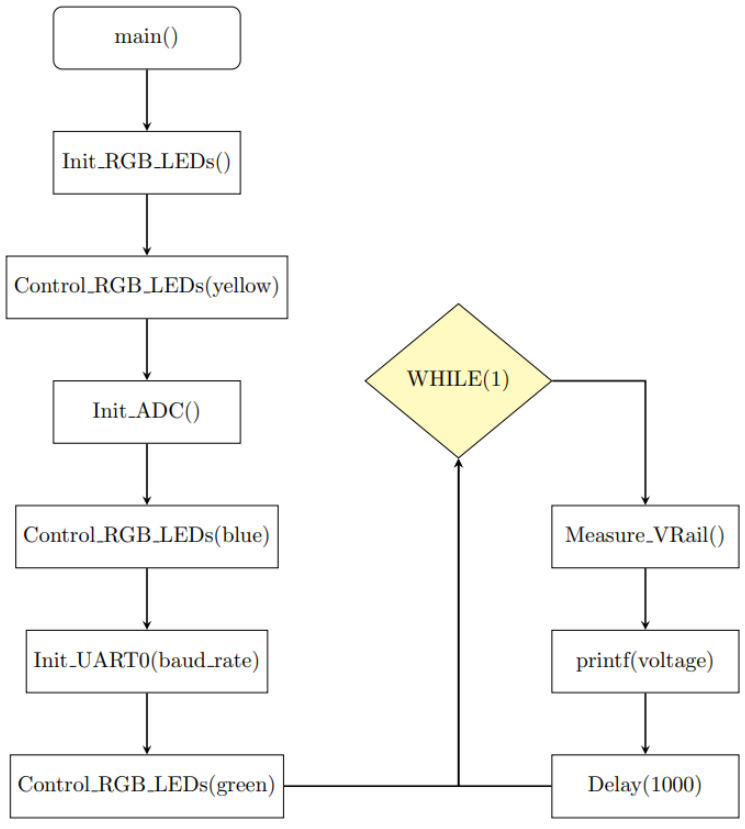
## SYSTEM ARCHITECTURE ANALYSIS

### HARDWARE

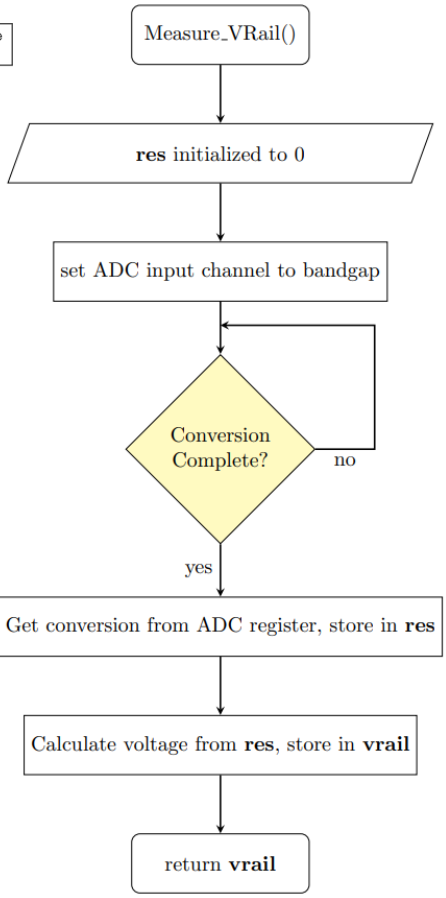
1. Peripherals (other than UART0 and ADC0):
    a. System Integration Module, **SIM** (enabling clocks to UART0, Ports A, B, & D)
    b. Ports **A, B, D**
    c. General Purpose Input/Output, **GPIO**
    d. Power Management Controller, **PMC**
2. It seems that the ADC is being polled from **main**.
3. There doesn't seem to be any **_IRQHandler** functions in the source code.
4. There doesn't seem to be any peripherals that generate control signals for other peripherals in this program.

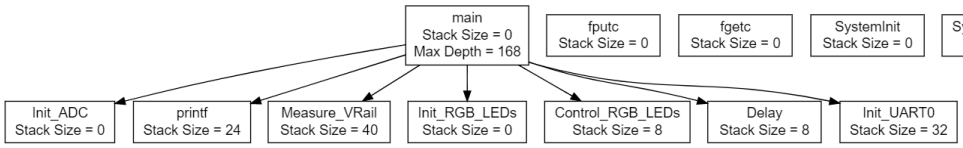
### SOFTWARE

5. There is currently no explicit scheduler or kernel used to schedule the processors time. The system currently just relies on **main**.
6. Note that this program does not service any interrupts, so the handlers are unused. But here's a list of all the threads and handlers:
    a. Threads:
        i. Main
    b. Handlers:
        i. UART0 – Status and error
        ii. ADC0
        iii. Port A, D – Pin detect
        iv. HardFault
7. Flowchart for **main** ⟶



8. Flowchart for **Measure_VRail** ⟶




9. Function call graph for **main** ^

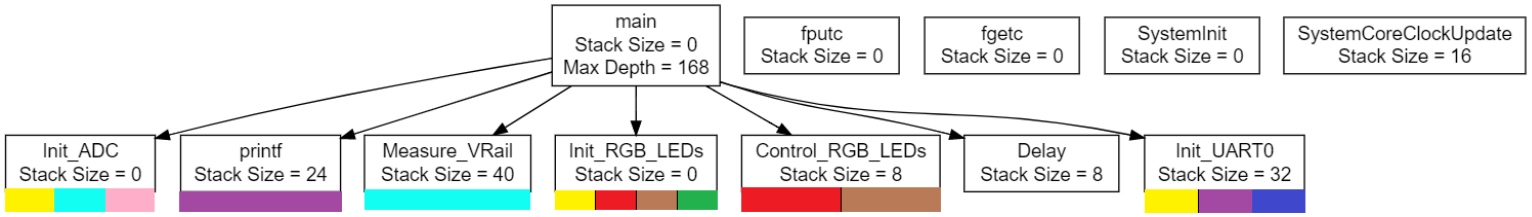
## HARDWARE/SOFTWARE INTERACTIONS

10. Table with all peripherals each function **directly** accesses

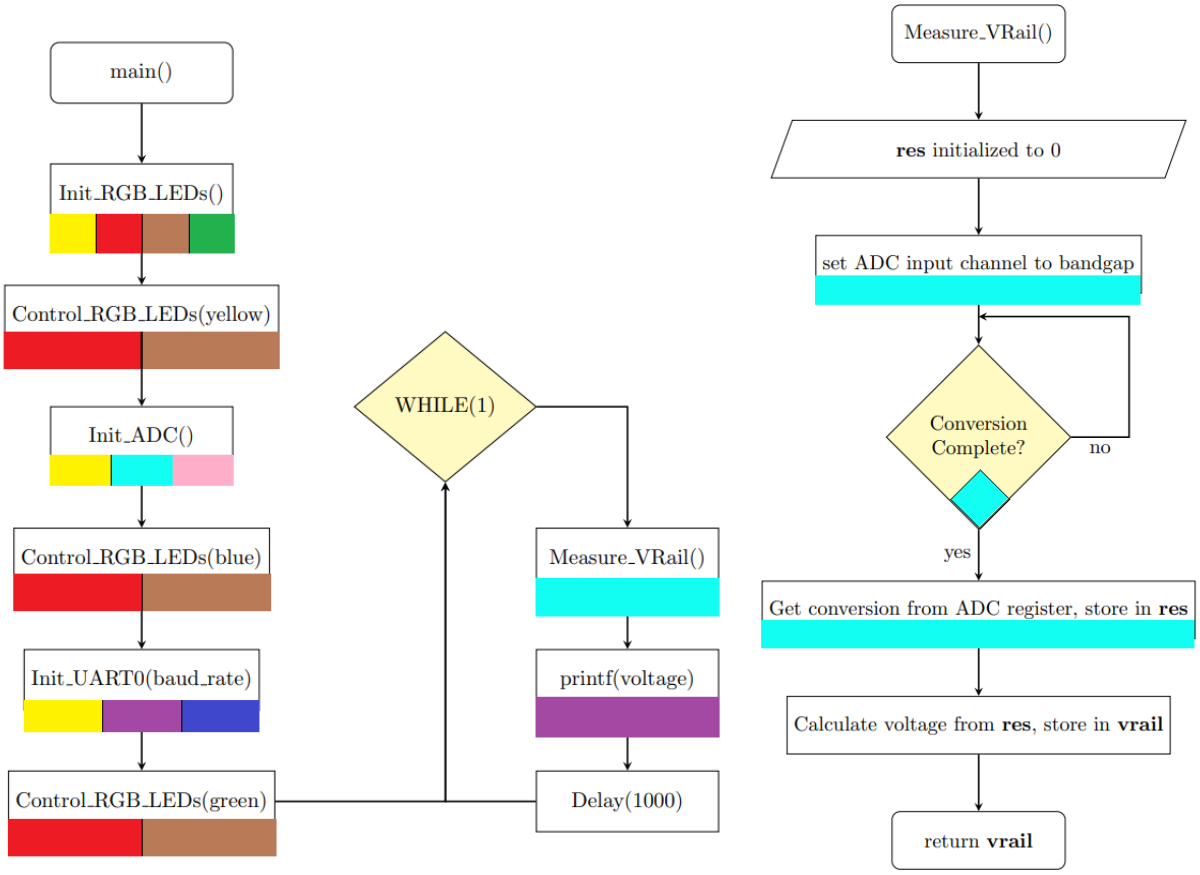
| **Function** | **Peripherals Accessed** |
|---|---|
| main | None |
| Init_RGB_LEDs | SIM, Port B, Port D, GPIO |
| Init_ADC | SIM, ADC0, PMC |
| Init_UART0 | SIM, UART0, Port A, |
| Control_RGB_LEDs | Port B, Port D |
| Delay | None |
| Measure_VRail | ADC0 |
| printf | UART0 |

11. Color code table

| **Peripheral** | **Color** |
|---|---|
| Port A | Blue |
| Port B | Red |
| Port D | Brown |
| SIM | Yellow |
| UART0 | Purple |
| ADC0 | Teal |
| GPIO | Green |
| PMC | Pink |



12. Color coded flowcharts

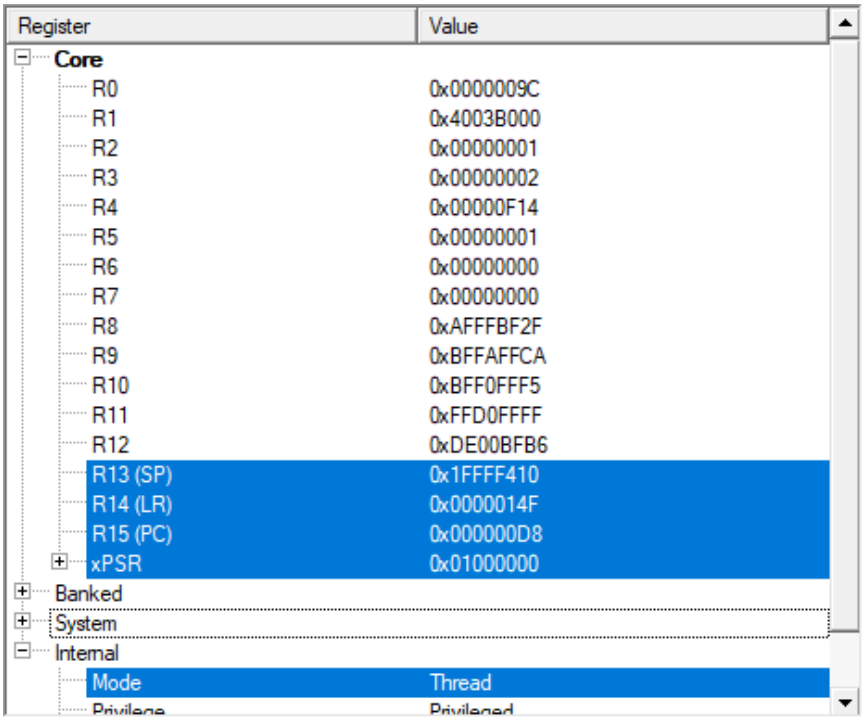



## BUG 1

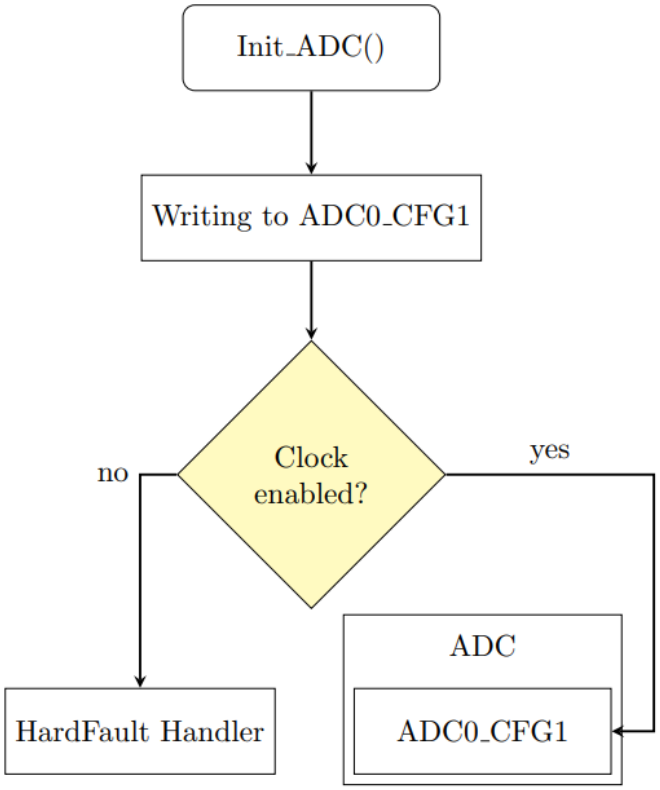
13. HardFault handler entered after entering Init_ADC.

| Name | Location/Value | Type |
|---|---|---|
| HardFault_Handler | 0x000003E6 | void f() |
| Init_ADC | 0x000000D4 | void f() |
| main | 0x00000000 | int f() |
| voltage | 0 | auto - float |

14. R1 is word aligned – 0x4003B000 and 0x4003B008 are divisible by 4.

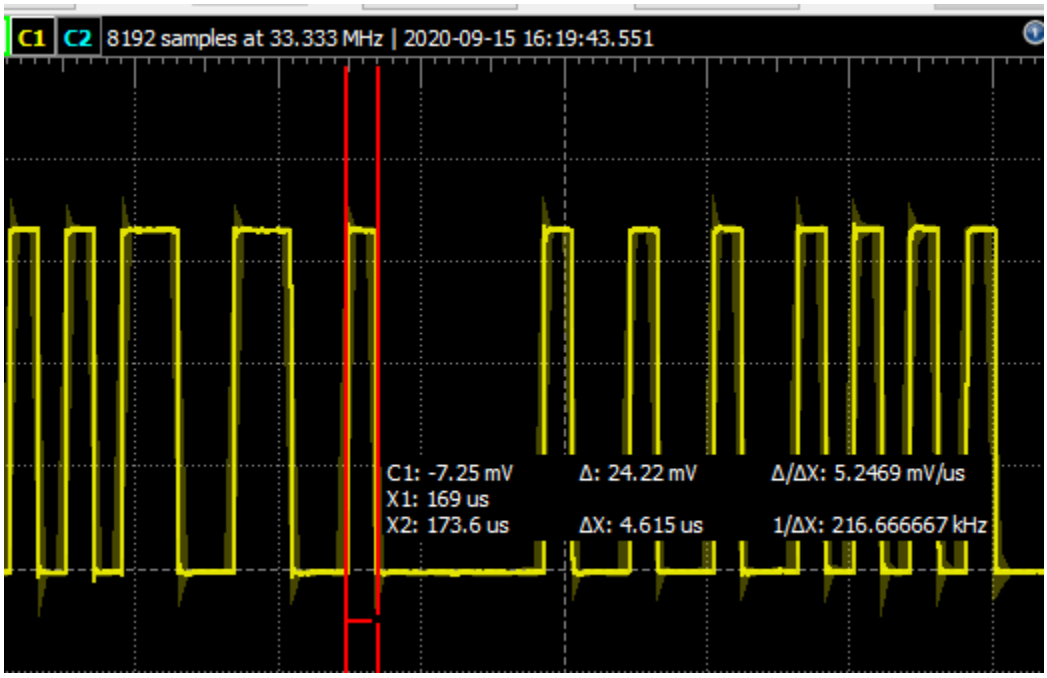



15. As can be seen, the Init_ADC function is unable to write to the ADC peripheral for configuration before the clock is enabled using SIM.

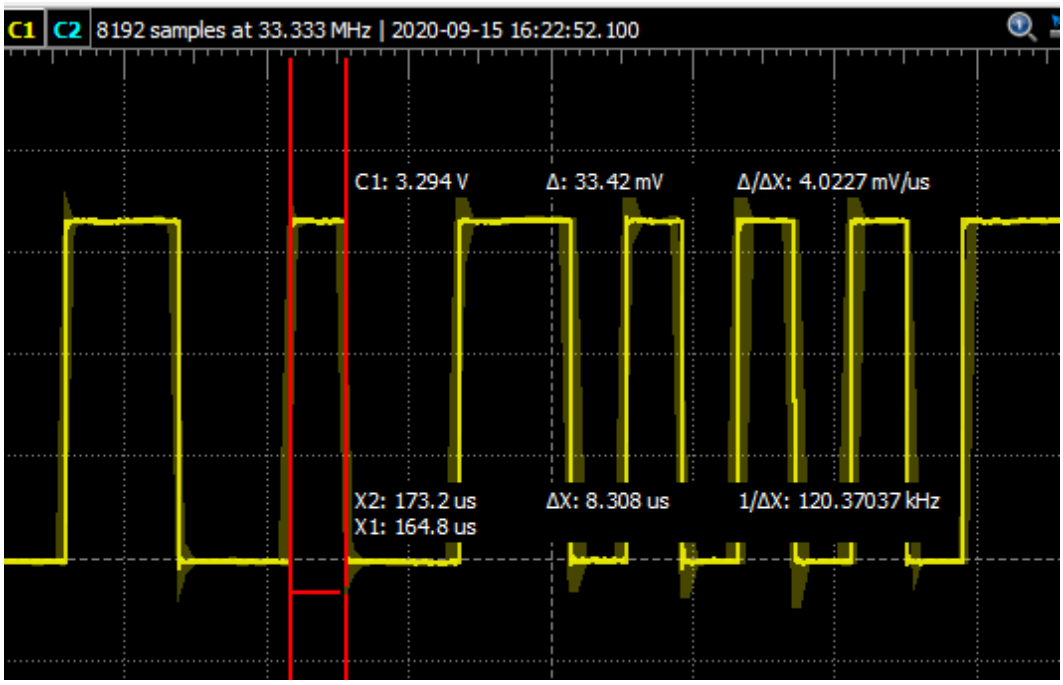


## BUG 2

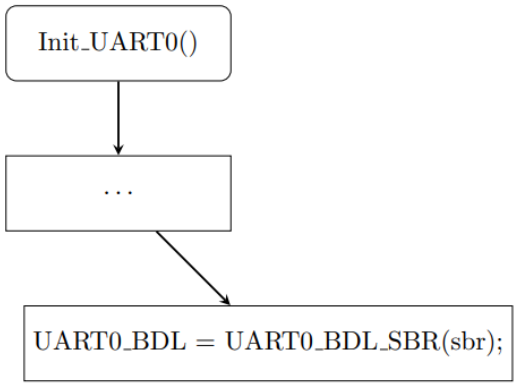
16. Each bit has a temporal width of 4.6 microseconds, transmitting on PTA2 to the laptop. This is not the 8.68 microseconds which is expected.




17. Each bit has a temporal width of 8.3 microseconds, transmitting on PTA2 to the laptop. This **is** the correct baud rate of 115,200 (8.68 microseconds). The slight error must come with my AD2, since later in the lab the laptop is correctly able to read the serial communications of said baud.



18. Since the value for **sbr** was previously bit shifted by 1 to the right, it decreased the divisor by 2, hence making the initial transmission twice as fast.




### 39.2.2 UART Baud Rate Register Low (UARTx_BDL)

This register, along with UART_BDH, control the prescale divisor for UART baud rate generation. The 13-bit baud rate setting [SBR12:SBR0] can only be updated when the transmitter and receiver are both disabled.

UART_BDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled; that is, UART_C2[RE] or UART_C2[TE] bits are written to 1.

Address: Base address + 1h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read / Write | SBR | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**UARTx_BDL field descriptions**

| Field | Description |
|---|---|
| 7–0 SBR | Baud Rate Modulo Divisor<br><br>These 13 bits in SBR[12:0] are referred to collectively as BR. They set the modulo divide rate for the UART baud rate generator. When BR is cleared, the UART baud rate generator is disabled to reduce supply current. When BR is 1 - 8191, the UART baud rate equals baud clock/(OSR×BR). |

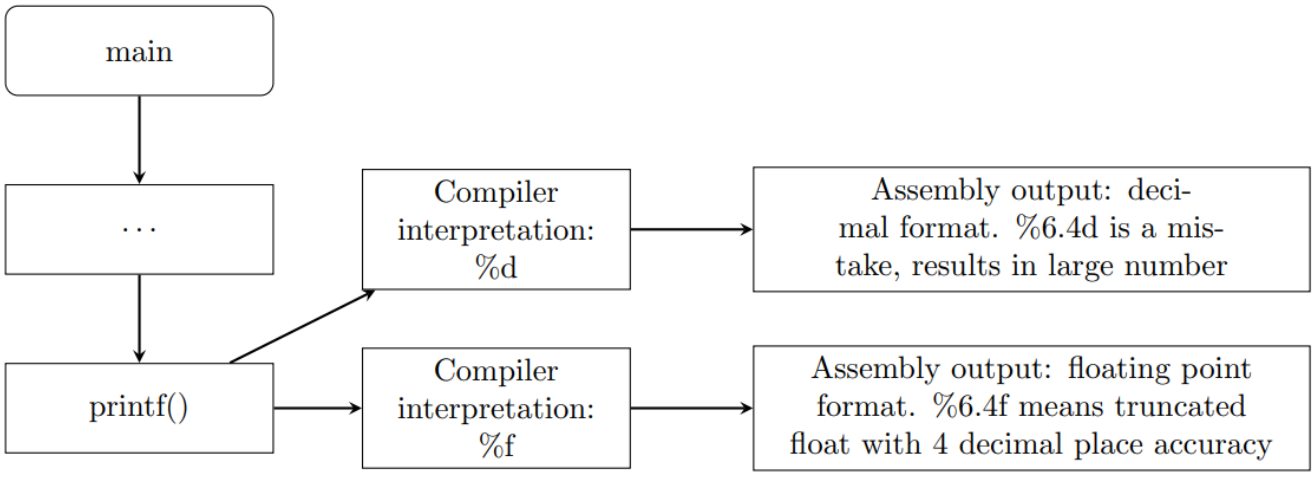
## BUG 3

19. The text displayed shows a very large voltage. Clearly a mistake in how its calculated or displayed. We soon figure out, it's a bug with how its displayed and transmitted. I am using Termite to read serial communications through the USB for bugs 3 and 4.

```
s at 1073910987 V
P3V3 Rail is at 1073910987 V


Debug me!
P3V3 Rail is at 1073910987 V
P3V3 Rail is at 1073920732 V
P3V3 Rail is at 1073754651 V
P3V3 Rail is at 1073829783 V
P3V3 Rail is at 1073863259 V
P3V3 Rail is at 1073439101 V
P3V3 Rail is at 1073628948 V
P3V3 Rail is at 1073829783 V
P3V3 Rail is at 1073861271 V
P3V3 Rail is at 1073460725 V
P3V3 Rail is at 1073346997 V
P3V3 Rail is at 1073558437 V
P3V3 Rail is at 1073427232 V
P3V3 Rail is at 1073671556 V
P3V3 Rail is at 1073496400 V
P3V3 Rail is at 1073889951 V
```

20. The bug simply has to do with how the compiler decides to compile the printf statement.

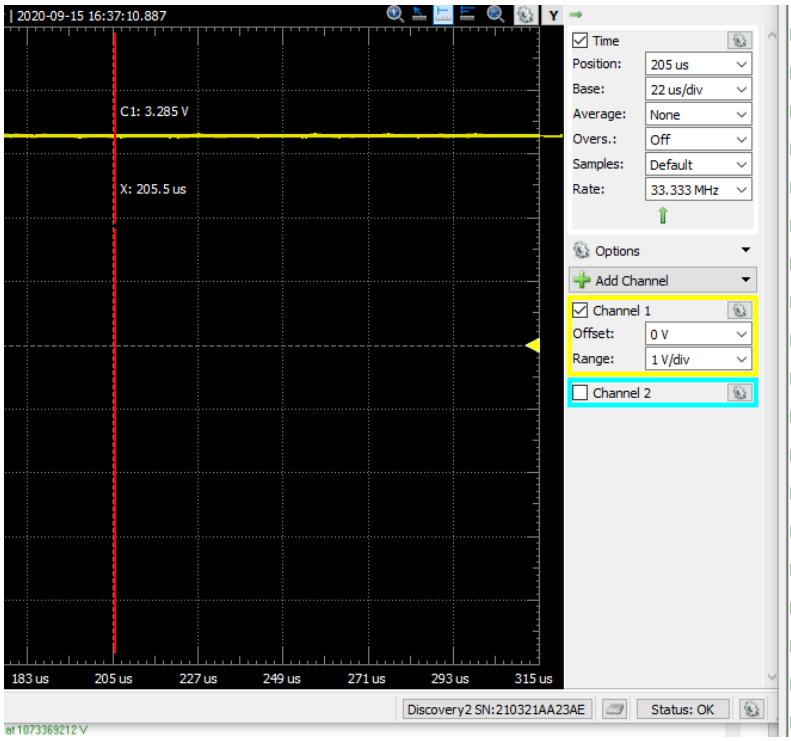


## BUG 4

21. Now that the printf statement works properly, the value is still off. After the screenshot, the value converged to ~2.3227V.

```
P3V3 Rail is at 1.7885 V
P3V3 Rail is at 2.0818 V
P3V3 Rail is at 1.6821 V
P3V3 Rail is at 2.1317 V
P3V3 Rail is at 2.4073 V
P3V3 Rail is at 1.7728 V
P3V3 Rail is at 2.2499 V
P3V3 Rail is at 2.8820 V
P3V3 Rail is at 3.1958 V
P3V3 Rail is at 3.1477 V
P3V3 Rail is at 3.5204 V
P3V3 Rail is at 3.4934 V
P3V3 Rail is at 3.7567 V
P3V3 Rail is at 2.3023 V
P3V3 Rail is at 2.4073 V
P3V3 Rail is at 1.7785 V
P3V3 Rail is at 2.1986 V
P3V3 Rail is at 1.8572 V
P3V3 Rail is at 1.5652 V
P3V3 Rail is at 1.5117 V
P3V3 Rail is at 1.3757 V
```

22. and 23. Voltage on COM port says slightly above 3.3V (~3.325V) while voltage on oscilloscope says slightly below 3.3V (~3.285). Roughly equal!

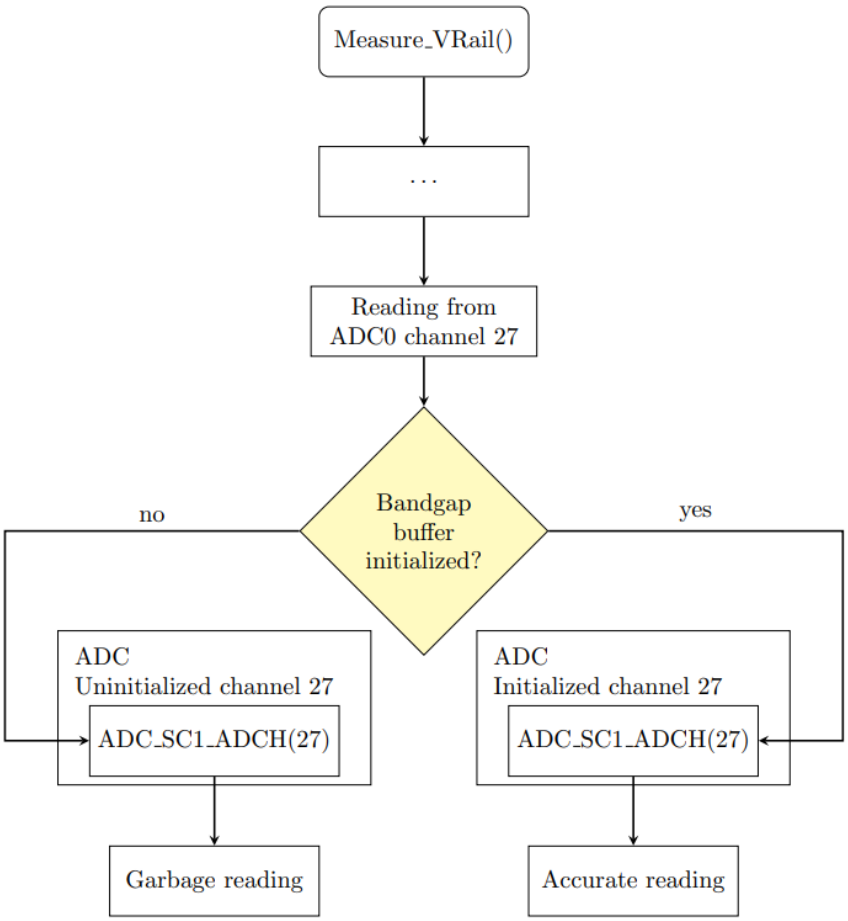



```
P3V3 Rail is at 3.3215 V
P3V3 Rail is at 3.3287 V
P3V3 Rail is at 3.3208 V
P3V3 Rail is at 3.3211 V
P3V3 Rail is at 3.3262 V
P3V3 Rail is at 3.3274 V
P3V3 Rail is at 3.3208 V
P3V3 Rail is at 3.3233 V
P3V3 Rail is at 3.3274 V
P3V3 Rail is at 3.3267 V
P3V3 Rail is at 3.3247 V
P3V3 Rail is at 3.3227 V
P3V3 Rail is at 3.3233 V
P3V3 Rail is at 3.3267 V
P3V3 Rail is at 3.3208 V
P3V3 Rail is at 3.3235 V
P3V3 Rail is at 3.3153 V
P3V3 Rail is at 3.3260 V
P3V3 Rail is at 3.3208 V
```

24. Since we are reading from ADC0 channel 27 (as given by the line in Measure_VRail(): ADC0->SC1[0] = ADC_SC1_ADCH(27);) we are reading from the bandgap reference voltage channel. As given in the documentation, you can see that the bandgap buffer must be enabled before reading from it. Therefore, the program was trying to read from an uninitialized channel.

| | | | |
|---|---|---|---|
| 11011 | AD27 | Bandgap (Diff)[2] | Bandgap (S.E)[2] |
| 11100 | AD28 | Reserved | Reserved |
| 11101 | AD29 | -VREFH (Diff) | VREFH (S.E) |
| 11110 | AD30 | Reserved | VREFL |
| 11111 | AD31 | Module Disabled | Module Disabled |

1. ADCx_CFG2[MUXSEL] bit selects between ADCx_SEn channels a and b. Refer to MUXSEL description in ADC chapter for details.
2. This is the PMC bandgap 1V reference voltage. Prior to reading from this ADC channel, ensure that you enable the bandgap buffer by setting the PMC_REGSC[BGBE] bit. Refer to the device data sheet for the bandgap voltage ($V_{BG}$) specification.




25. We can find that the maximum error of the ADC reading from channel 27 is 3%, since the bandgap voltage reference tends around 1V with a minimum of 0.97V and maximum of 1.03V.

| | | | | | |
|---|---|---|---|---|---|
| $V_{BG}$ | Bandgap voltage reference | 0.97 | 1.00 | 1.03 | V |

The error from my reading can be calculated by taking the magnitude of the difference of the two voltage readings, and then dividing it by the average of the two readings. The reason the divisor is the average, is because the oscilloscope might have some error as well, so we use the average of the two as our reference voltage. So: (3.325 – 3.285)/(3.305) = 1.21%, which is better than the worst case scenario of 3%.