

# Microprocessor Systems Lab 4

## Checkoff and Grade Sheet

Partner 1 Name: Yineng Li

Partner 2 Name: Yuchen Wang

Grade Component	Max.	Points Awarded		TA Init.s	Date
		Partner 1	Partner 2		
Performance Verification: Task 1	10 %	10		KW	10/25
Task 2	10 %	10			
Task 3	10 %	10	10	TAC	10/28
Task 4	10 %	10		TAC	10/28
Task 5 [Depth]	10 %				
Documentation and Appearance	50 %				
Total:					



# Task 1: Simple Voltmeter

- Introduction and High-Level Description
  - The purpose of task 1 is to let one of the analog pins on the DISCO board sample a voltage.
  - Sampling mode will be set to single channel / single conversion, using full 12-bit ADC resolution.
  - The hexadecimal value as well as the decimal value of the voltage with at least 6 significant digits will be printed out.
  - The indicator of High, Low, and Average indicator will be displayed.
  - The program will record the last 16 values and calculate the average, maximum and minimum.
  - Flowchart in appendix.

Table 1: Task 1 wire connection

	Connected to	Connected to
Wave generator	Disco board Pin A0	Analog discovery V+
ground	Disco board gnd	Analog discovery gnd

- Low Level Description
  - Enable clock by `HAL_RCC_ADC1_CLK_ENABLE();`.
  - For setting to proper clock and resolution, `hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV6;` and `hadc1.Init.Resolution = ADC_RESOLUTION_12B;` are used.
  - ADC channel is configured by `sConfig.Channel = ADC_CHANNEL_6;`, `sConfig.Rank = 1;`, `sConfig.SamplingTime =`

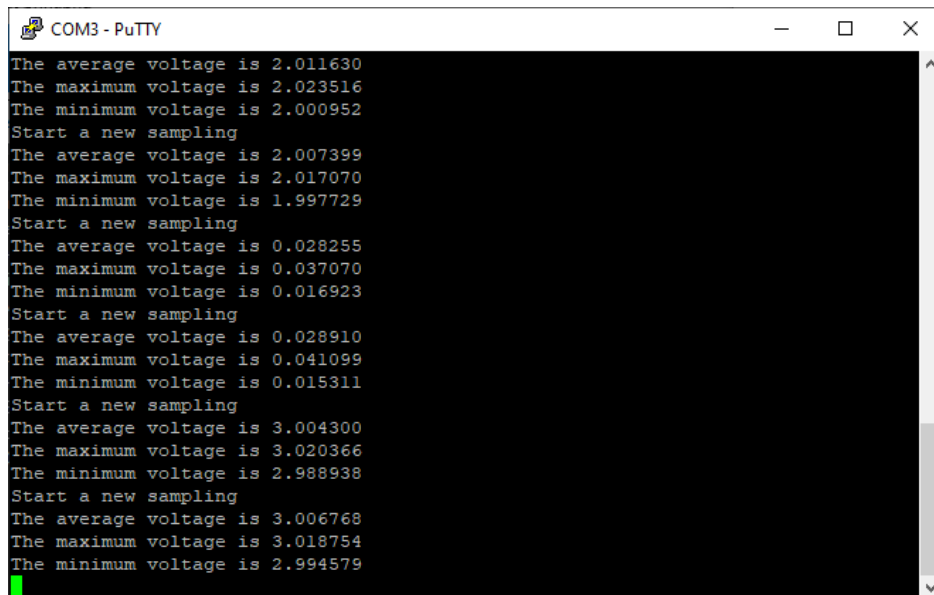
```
ADC_SAMPLETIME_15CYCLES;], and
```

```
HAL_ADC_ConfigChannel(&hadc1,&sConfig);.[1]
```

- To convert value that read from adc to voltage value, use `tmp = 3.3*convResult/4095;.[2]`
- Properly set the GPIOs as did in previous labs.
- To read the value, Interrupt EXTI0 is used here, can be triggered by the User pushbutton. After pushing this button, the Disco board will print the value in the terminal.

- Results and Analysis

- The program worked as expected. When the button is pressed, the DISCO board would read the adc input and display the voltage value. Also, the max, min, and average value of the last 16 attempts are displayed.
- What's new in this lab is how to select the right mode for the ADC to read the voltage value.
- Here are the results.



```
COM3 - PuTTY
The average voltage is 2.011630
The maximum voltage is 2.023516
The minimum voltage is 2.000952
Start a new sampling
The average voltage is 2.007399
The maximum voltage is 2.017070
The minimum voltage is 1.997729
Start a new sampling
The average voltage is 0.028255
The maximum voltage is 0.037070
The minimum voltage is 0.016923
Start a new sampling
The average voltage is 0.028910
The maximum voltage is 0.041099
The minimum voltage is 0.015311
Start a new sampling
The average voltage is 3.004300
The maximum voltage is 3.020366
The minimum voltage is 2.988938
Start a new sampling
The average voltage is 3.006768
The maximum voltage is 3.018754
The minimum voltage is 2.994579
```

Figure 1: Terminal output

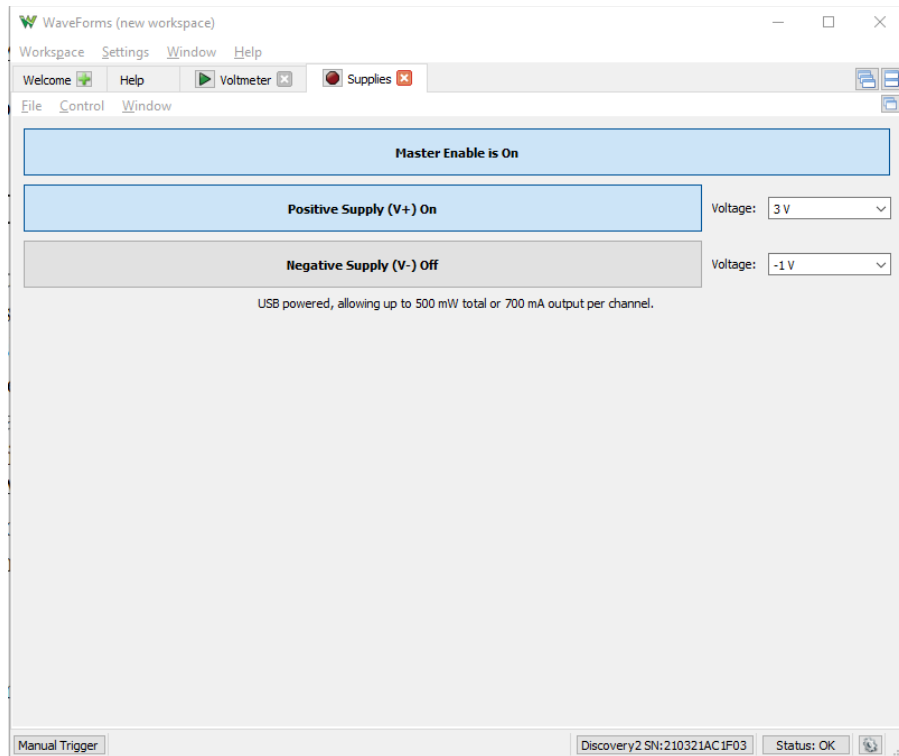


Figure 2: Analog discovery voltage supply

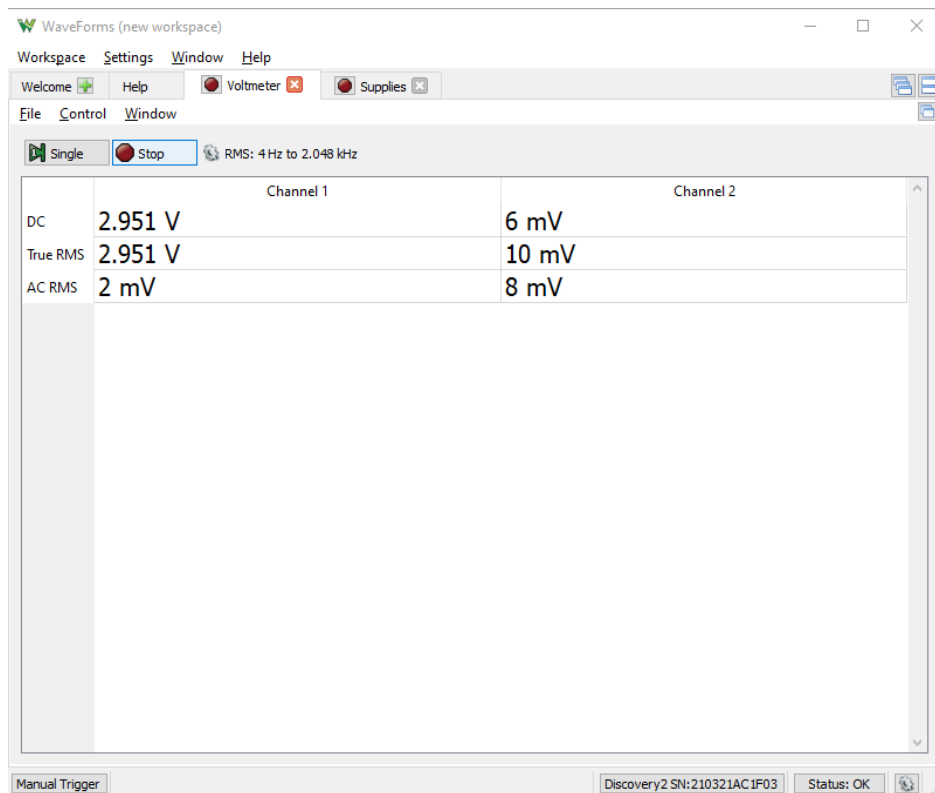


Figure 3: Analog discovery voltmeter

## Task 2: DAC Output

- Introduction and High-Level Description
  - In this task, the DISCO board will be connected to a signal generator, which will keep sending a sine wave. Meantime, the board will use ADC1 to read the value and use DAC to output the value.
  - The program will keep polling the ADC value and keep sending the DAC output.
  - Flowchart in appendix

Table 2: Task 2 wire connection

	Connected to	Connected to
Wave generator	Disco board pin A0	Analog discovery w1
DAC output	Disco board Pin A1	Analog discovery Oscilloscope 1
ground	Disco board GND	Analog discovery GND

- Low Level Description
  - The ADC part of this program is quite like the part in task1. But instead of reading the input only when the button is pressed, ADC will keep reading and converting the input this time.
  - Use `__HAL_RCC_DAC_CLK_ENABLE();` to enable the clock.
  - Use `DAC_ChannelConfTypeDef ChannelConfig;` to declare the `DAC_ChannelConfTypeDef`. To config channel out 1 for DAC, use `ChannelConfig.DAC_Trigger = DAC_TRIGGER_NONE;` and `ChannelConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;`
  - To properly set up the DAC Handler, `DACHandler.Instance = DAC;` and `HAL_DAC_Init(&DACHandler);` are used.

- And finally, to configure the DAC channel,

```
HAL_DAC_ConfigChannel(&DACHandler, &ChannelConfig,  
DAC_CHANNEL_1);
```

 is used.[3]

- Results and Analysis

- The program worked as expected. When the DISCO board detects any voltage input, it will automatically output the same voltage. So when the input is a sine wave, DAC will take the conversions generated from ADC and output them through DAC1.
- What's new in this lab is how to generate output voltage through DAC.
- Here are the results.

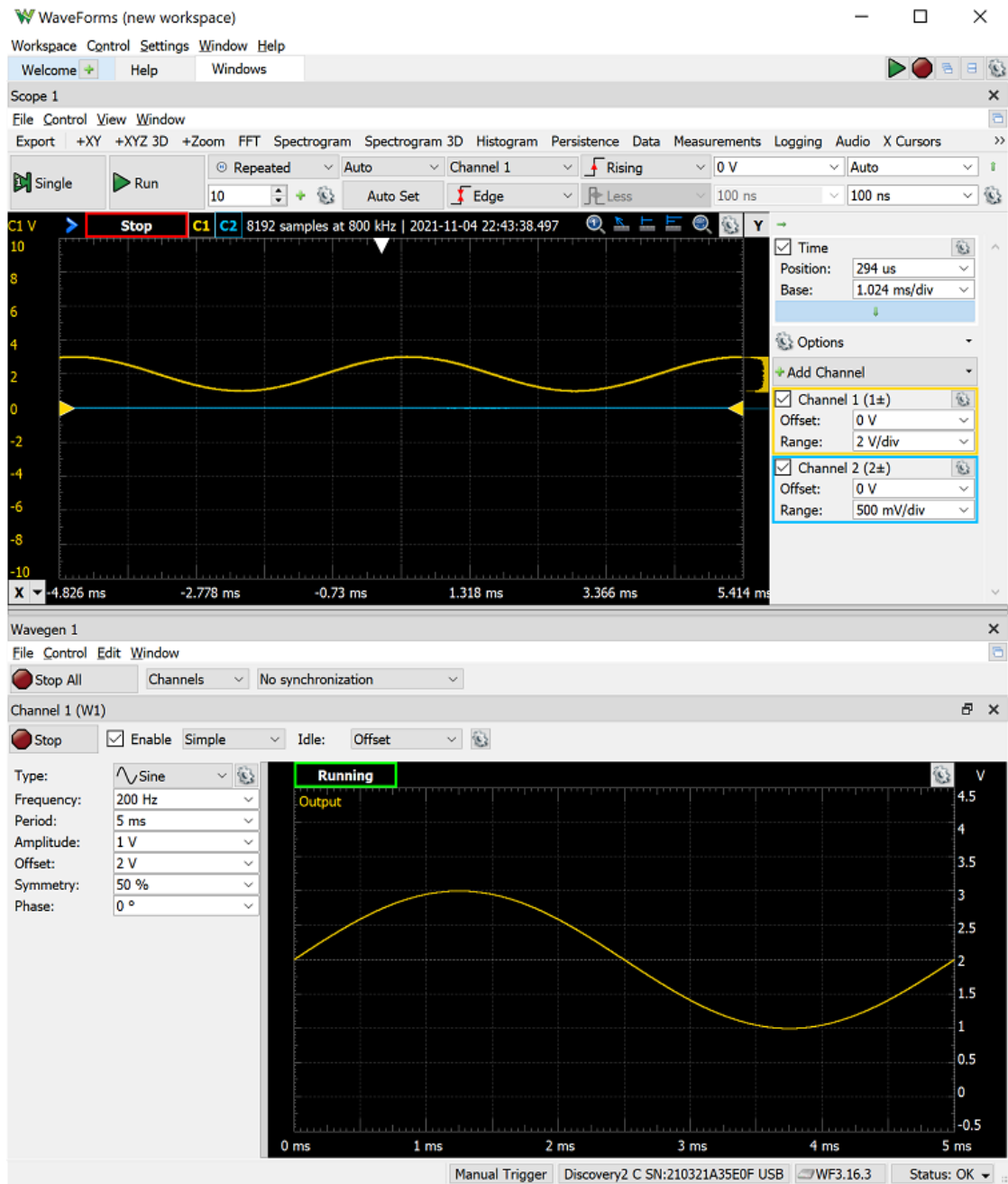


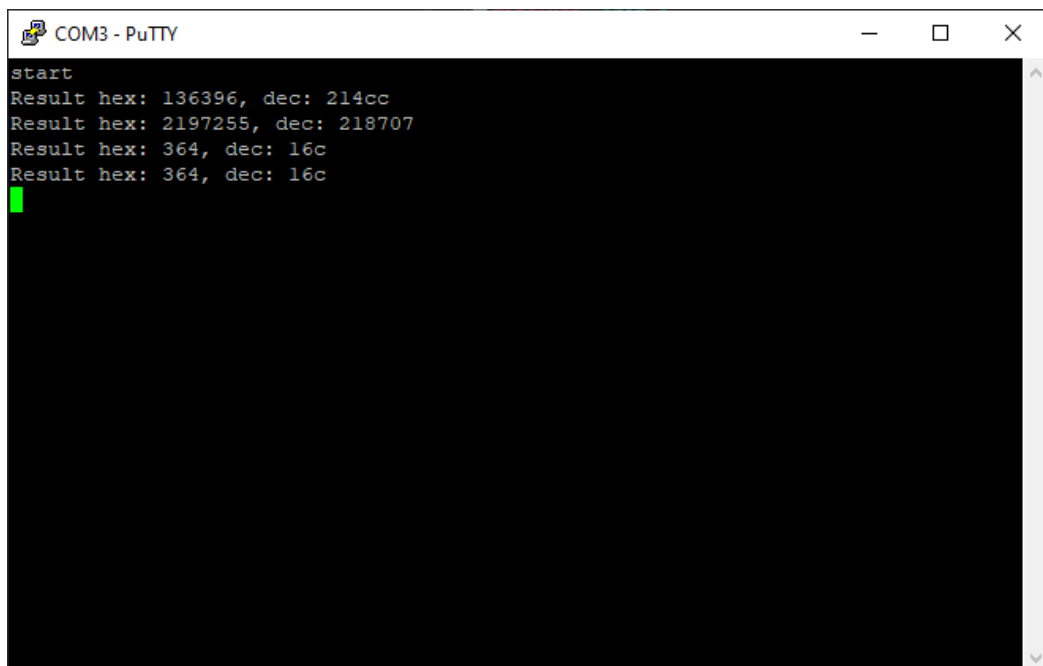
Figure 4: Task 2 Result

# Task 3: Simple Assembly Math

- Introduction and High-Level Description
  - There are 2 parts in this task. Both tasks have to use basic inline assembly to first load and then add integers together. Then use extended assembly to pass the result to a C variable and then print it out.
  - The reset part in these 2 tasks sets are slightly different. In the first task set, 2 `int32_t` variables have to be multiplied using extended assembly when 2 single precision floats have to be multiplied using extended assembly in the other task set.
  - Then using extended assembly to evaluate one equation:  $2x/3 + 5$ , the first task set asks to solve this equation using 32-bit integer math and the second task set asks to solve this equation using floating point addition.
  - The final one is also using extended assembly. Both of the task sets will use MAC commands to evaluate the previous equation again. However, the first task set will still use 32-bit integers and the second task will still use floating points.
  - Since the program is only using Disco board to calculating, there is no flow chart for this task.
- Low Level Description
  - Use LDR to load into registers.
  - Use ADD to add two targets together and use STR to convert the result into string. And finally use printf to print out the result.
  - In the multiply part, using MUL to multiply two targets for the `int32_t`, and using VMUL for the single precision floats.



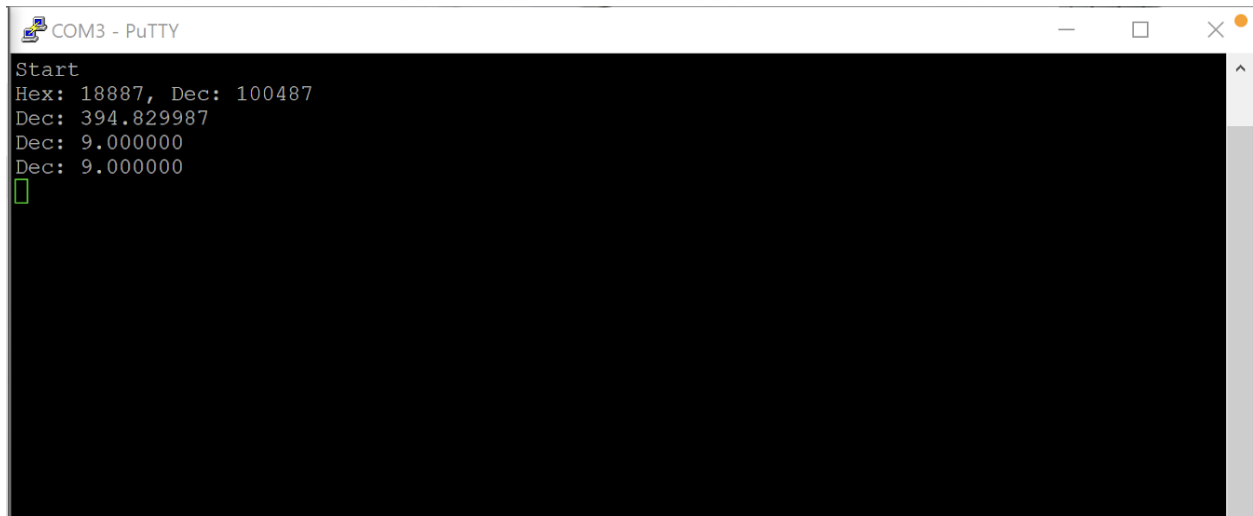
- Then for the evaluation part. Using MLA and SDIV for the 32-bit integer and using VMLA for the floating points.
- For floating point assembly, it is necessary to declare the type of calculation after the commands, here students used F32 for 32-bit floating point.
- Results and Analysis
  - The program worked as expected.
  - Here are the results



A screenshot of a PuTTY terminal window titled "COM3 - PuTTY". The window has a black background with white text. The text shows the output of an assembly program. It starts with "start" on a new line. Then, four lines of output are shown, each starting with "Result hex:" followed by a hexadecimal value, a comma, and "dec:" followed by a decimal value. The values are: 136396 (214cc), 2197255 (218707), 364 (16c), and 364 (16c). A green cursor is visible on the line following the last result.

```
start
Result hex: 136396, dec: 214cc
Result hex: 2197255, dec: 218707
Result hex: 364, dec: 16c
Result hex: 364, dec: 16c
```

Figure 5: Task 3 result 1 interger



```
COM3 - PuTTY
Start
Hex: 18887, Dec: 100487
Dec: 394.829987
Dec: 9.000000
Dec: 9.000000
█
```

Figure 6: Task 3 result float

## Task 4: IIR Filter Implementation

- (a) *Introduction and High-Level Description*
  - The purpose of this is to build a IIR filter using the ADC and DAC function in the DISCO board. The board will read a input signal and process it. The output is a processed signal.
  - The filter equation is given in the lab instruction, here student needs to implement it in two ways, c equation and MAC assembly.
  - The basic idea of this IIR filter is the output is a sum of current input, last two input, and the last output signal voltage. Each with different coefficient.
  - Flow chart is attached in the appendix
  - Wire connection:

Table 2: Task 4 schematic

	Connected to	Connected to
ADC	A0 DISCO board	Discovery board wave generator W1
DAC	A1 DISCO board	Discovery board oscilloscope 2
Ground	GND DISCO board	Discovery board ground
Oscilloscope 2	GND DISCO board	Oscilloscope 2-
Oscilloscope 1	Discovery board wave generator W1	GND

- *(b) Low Level Description*
  - The ADC and DAC setting are the same as task 2

- Students used the converted result directly for the signal processing, so there is a possibility of overflow. To solve this problem, students used double variable for c equation. And device the coefficient by 1000 for the MAC assembly. After the calculation, the result will multiply by 1000.
- To verify the filter, network function in the Analog discovery board is used. It automatically set the wave generator and use two oscilloscope to read the filter output and the wave generator output.
- For each loop, the program will get the current value and calculate the output value with all previous data, then update the previous data.
- User pushbutton will be used for interrupt to choose the running model.

Interrupt EXTI0 is used here for interrupt.

- *(c) Results and Analysis:*

- The program worked as expected
- MAC assembly works better than c equation. Directly use FPU has a better performance.
- The result are as followed. Zeros showed in the figure.

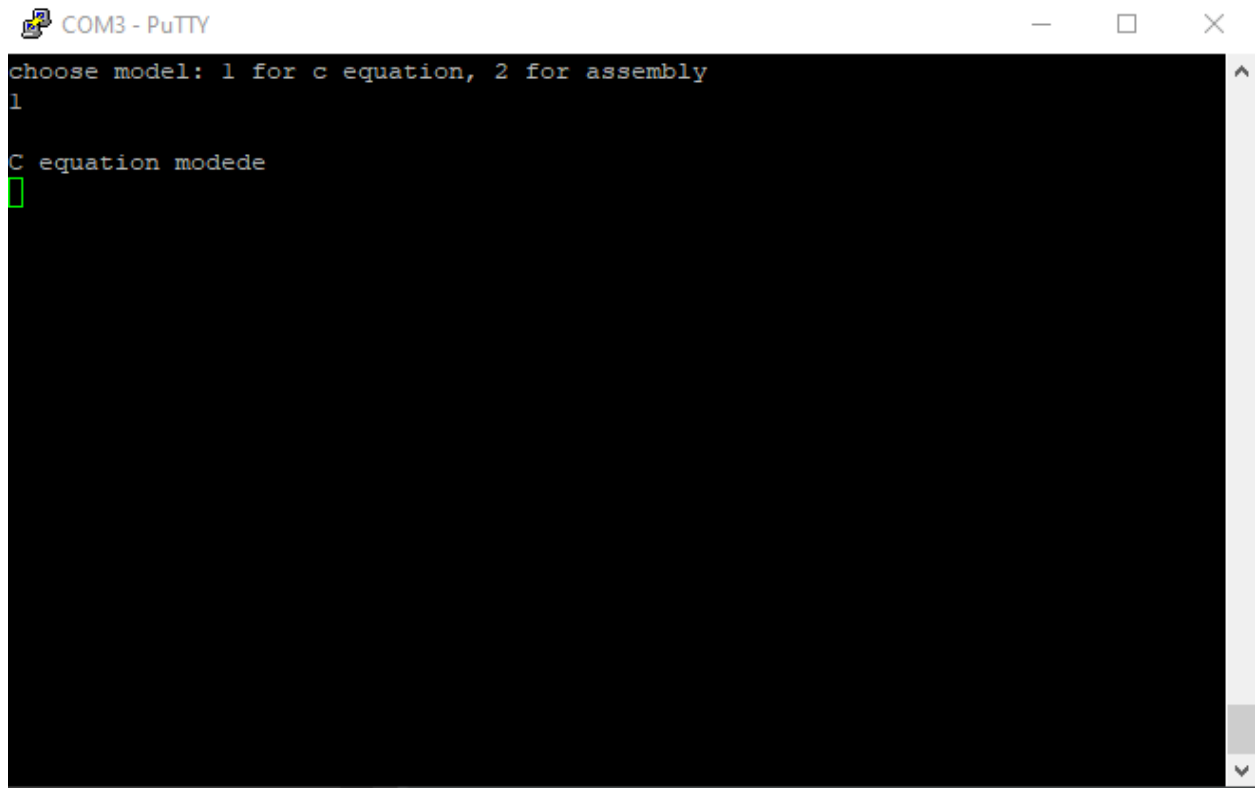


Figure 7: Task 4 c equation model

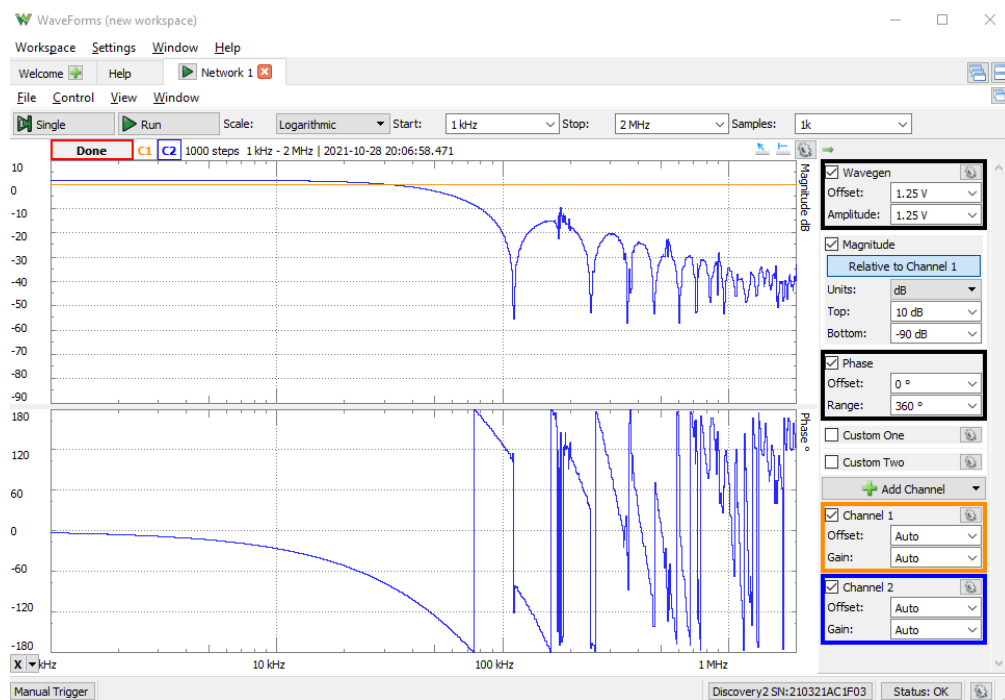


Figure 8: Task4 analog discovery result c equation



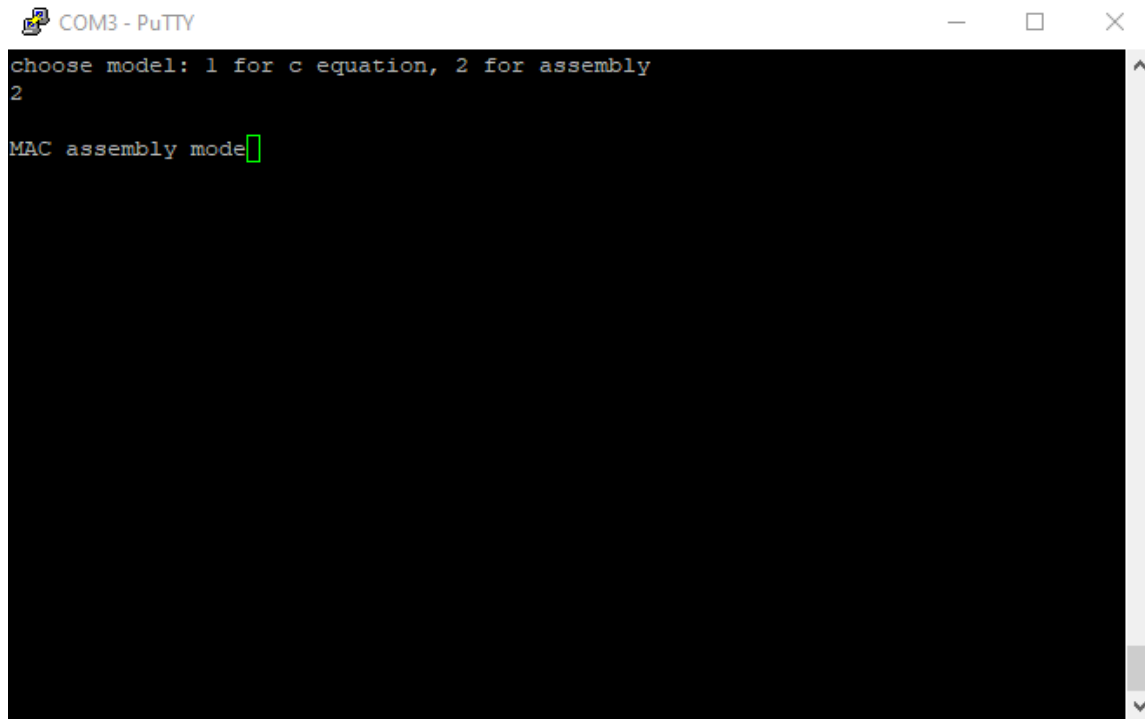


Figure 9: Task 4 MAC assembly model

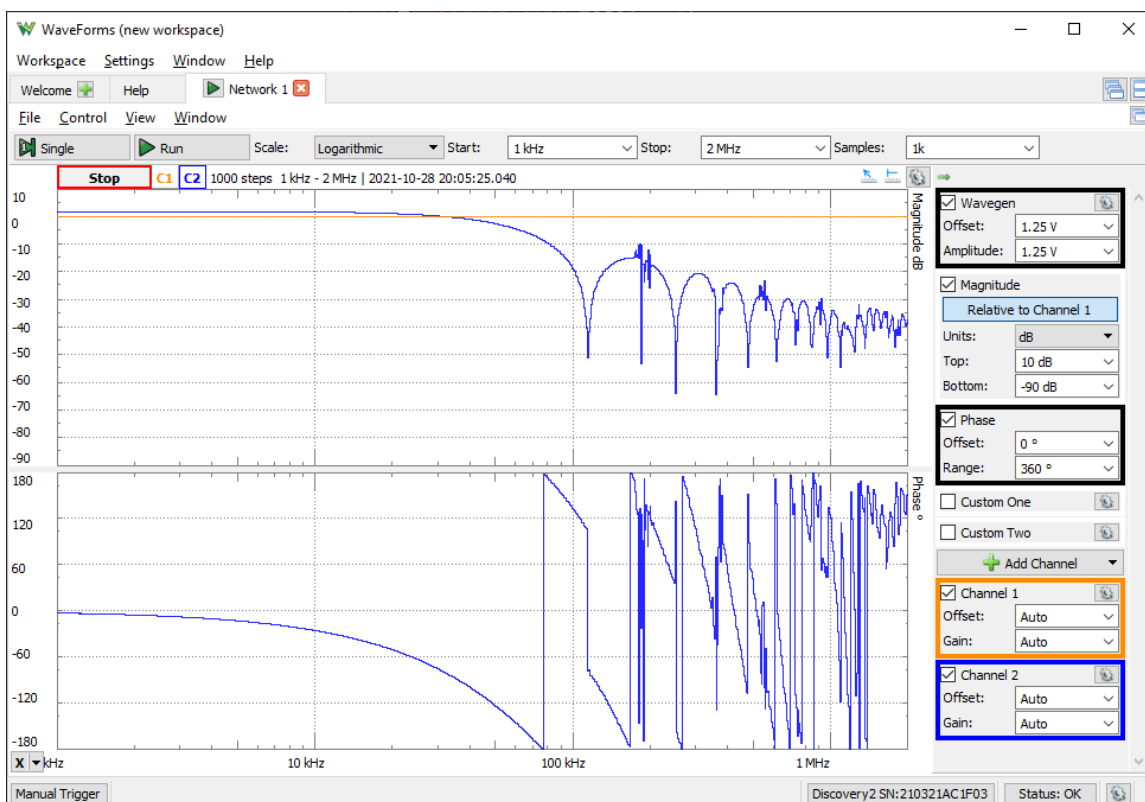


Figure 10: Task 4 MAC assembly model

# Conclusion

In this lab, students learned how to use ADC and DAC peripherals on the disco board. Open and set the ADC parameter to make it work as expected. Use ADC to read the input signal and process it. DAC setting and running is also learnt in this lab. Using DAC to output a signal.

Students also learned how to use arm assembly code, like MIPS, this basic assembly code can help improve the efficiency of calculation. This type of language directly orders the microprocessor to do the calculation using FPU. This can be used on processing signals.

The combination of these functions can make the microprocessor to be a signal processor. In task 4 students make a IIR filter by using ADC, DAC and arm assembly code.

The real world applications are variable for this lab. The ADC function can be a oscilloscope.

The multiple ADC channel can help the board do multiple measurements. The DAC can works as a wave generator. Using the ARM assembly code, the calculation can be fast. So the combination of their functions can make a low pass filter or high pass filter.

# Reference

- [1] *Mastering STM32*, Carmine Novello, USA, 2018, pp. 390.
- [2] *RM0410 Reference manual*, STMicroelectronics, NV, USA, 2018, pp. 438-444.
- [3] *RM1905 User manual*, STMicroelectronics, NV, USA, 2017, pp. 198-209.

# Appendix

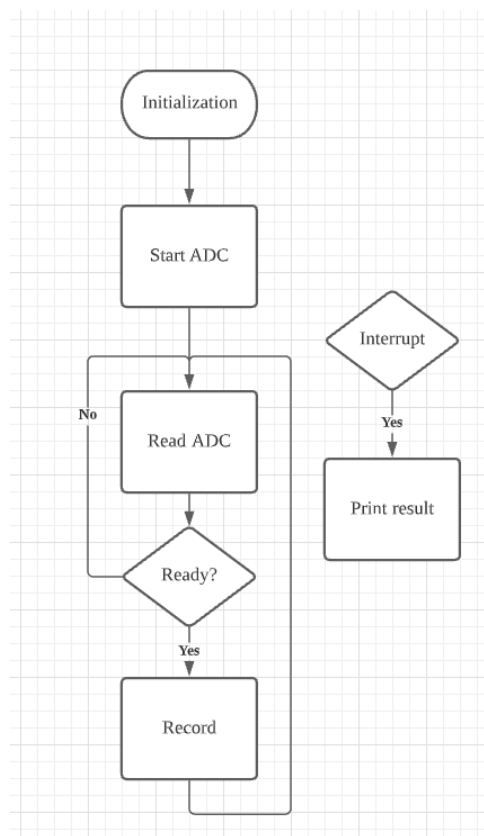


Figure 11: Task 1 Flowchart

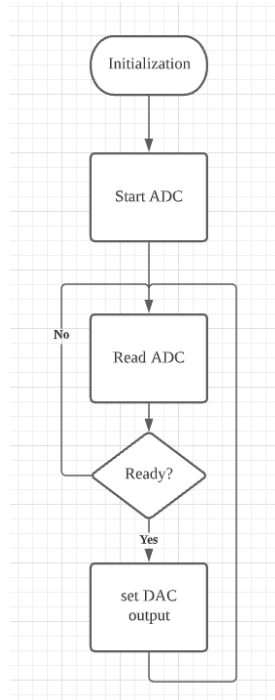


Figure 12: Task 2 Flowchart

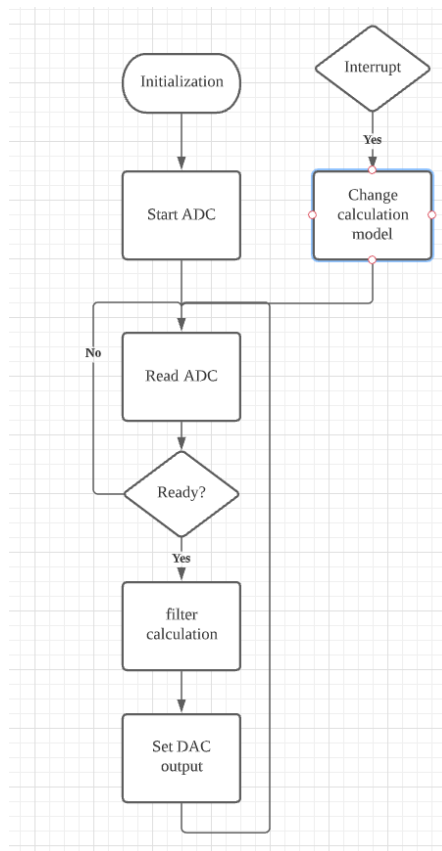


Figure 13: Task 4 Flowchart