

ECSE 444 Lab 2 Report

Basic I/O and ADC-based Readout of Temperature and Voltage

Group 11: Byron Chen, 260892558, Kevin Li, 260924898

Abstract—In this lab, a program that can observe and record core temperature and voltage of STM32 is designed by using ADC1. According to the the chip document and STM32 data sheet, the internal voltage is recorded by ADC1 and converted into actual temperatures and voltages by the formulae provided. By overwriting the write function, the program is able to print out the actual temperatures and voltages in the ITM console. Furthermore, by pressing the blue button on the board, the user is able to switch the data between temperature and voltage.

Index Terms—I/O, ADC1, Core Temperature, Internal Voltage, STM32

I. INTRODUCTION

CORE temperature and voltage of STM32 are two readily available physical parameters. In this lab, a program that can observe and record these two physical values is designed by using ADC1. Furthermore, this project allows switches between updating the readout of the temperature and voltage each time the blue button is pressed on the STM32 board. LED1 on the board will be used as an indicator light for noticing the user if it in the voltage mode or the temperature mode.

II. MATERIALS AND METHODS

This lab can be mainly divided into 5 steps, which are [1]:

- 1) Detecting the pressing of the blue button and testing the LED2 display by toggling the value every time the button pressed.
- 2) Read the reference voltage of the board by configuring the setting of ADC1.
- 3) Configure the setting of ADC1 in order to read properly the internal temperature of the board with ADC's temperature sensor.
- 4) Print out the data supposed to read.
- 5) Configure the code such that the program can switch the readout between the voltage and the temperature.

For step 1, the pin for the functional button (blue button) is read in the while loop. Whenever it is pressed, the status of the pin will turned to zero which can be the signal to tell the LED pin to toggle by using the function `HAL_GPIO_TogglePin`.

For step 2, ADC's channel must be set to a specific channel for reading the reference voltage at the beginning. There are two different methods to switch the readout channel of ADC. Either change it in the CubeMX or change it directly in the CubeIDE. Since the channel needs to be switched frequently between temperature readout and voltage readout, CubeIDE is chosen in this project.

First of all, a global variable, namely flag, is declared to signify the desired output ('0' for temperature, '1' for voltage). Since the calculation of internal temperature requires the value

of reference voltage, the flag is set to be 1 as default. Channel switching can be done by changing the value of the flag. Besides the channel switching, Sampling time needs to be calculated as well. The data sheet [2] provides the formula to calculate the sampling time(in cycle):

$$\text{convolution time} = \frac{\text{sampling time} + 12.5}{\text{ADC clock(Hz)}} \quad (1)$$

The data sheet indicates that the conversion time for reference voltage is 4us and the ADC clock is 80MHz [2]. So the required sampling time for the reference voltage readout will be $80\text{Mhz} * 4\text{us} - 12.5\text{cycles} = 307.5\text{cycles}$. From the provided sampling time's setting, the closest one with a sampling time of 247.5 cycles is chosen. The sampling time and channel of ADC are set by changing respectively `sConfig.SamplingTime` and `sConfig.Channel` in CubeIDE.

After setting the configurations, the reference voltage can be read correctly. However, a conversion needs to be done to get the actual value of reference voltage. The conversion formula provided in the chip document[3] is shown as below:

$$\text{reference voltage} = V_{\text{ref_charac}} * \frac{V_{\text{refint_cal}}}{V_{\text{refint_data}}} \quad (2)$$

Where $V_{\text{ref_charac}}$ is defined as 3000mV; $V_{\text{refint_cal}}$ is stored in the memory with an address of 0x1FFF75AA; $V_{\text{refint_data}}$ is the value obtained by ADC. And all the data is shown in the data sheet[2].

For step 3, the configuration of ADC is similar to the ADC's configuration of step 2. The global variable will be set to '0' and since the required conversion time for the temperature sensor is 5 us. ADC's sampling time of temperature sensor is $5\text{us} * 80\text{Mhz} - 12.5\text{cycles} = 387.5\text{cycles}$. The closest sampling time's setting is 247.5 cycles. The configuration is set in the same way as step 2 in CubeIDE. Conversion also needs to be done to obtain the actual internal temperature. The conversion formula provided in Chip document is shown as below:

$$T = \frac{(ts_cal1_T - ts_cal2_T)}{ts_cal2 - ts_cal1} * (ts_data - ts_CAL1) + 30 \quad (3)$$

Where ts_cal1_T and ts_cal2_T are 130 and 30 degrees celsius respectively; the values of ts_cal2 and ts_cal1 are stored in the memory with addresses 0x1FFF75A8 and 0x1FFF75CA; TS_data is the value obtained by the temperature sensor. And all the data is shown in the data sheet[2].

However, the formula (3) is only valid when the measurement of ts_data is done with the reference voltage of 3000mV. As a result, the formula needs to be adjusted. ts_data needs to

be multiplied by the ratio between the actual reference voltage and the default reference voltage. This actual reference voltage is calculated in step 2 and the default reference voltage is 3000mV.

For step 4, *printf* is used for printing out the needed to be read out after overwriting the built-in function *_write* and it is shown below.

```
1 /**
2  * Overwrite the write function for ITM
3  */
4  int _write(int file, char *ptr, int len){
5      int i = 0;
6
7      for (i = 0; i < len; i++){
8          ITM_SendChar((*ptr++));
9      }
10
11     return len;
12 }
```

Listing 1. Overwrite the Write Function

For step 5, the global variable, namely flag, and the blue button are used to switch the readouts of the values to be monitored. If the blue button is pressed, the value of the flag will be toggled. In function *MX_ADC1_Init*, there is a if-else selection is used to determine the current channel used. If the flag equals to '0', temperature sensor will be used; otherwise, ADC's configuration for reference voltage will be used. Since the calculation of the actual temperature needs the value of the reference voltage, the flag is initialized with value '1'. To ensure the read from the button pin to be discrete, which can prevent multiply readings by only one pressing action, a delay of 200 ms is set by using function *HAL_Delay*.

III. RESULTS

By pressing the blue button, the LED1 shown in Figure 1 is toggling and the print out for the data is changing. When the LED is off, Voltage is printed out as shown in Figure 2 and when it is on, temperature is printed out as shown in Figure 3.

In order to see if the temperature sensor working properly, a hair heater is used to rise the temperature of the core and the outputs are shown in Figure 4.

IV. DISCUSSION & SUMMARY

As shown in Figure 4, we can see that the temperature sensor is used properly. Also, by pressing the blue buttons, we can see that the LED pin-out and Button pin work properly as well. When doing this lab, it is important to notice that when converting the raw data from the ADC to the actual temperature, a factor as discussed in Section 2 can never be ignored.

V. WORK DISTRIBUTION

Both Byron and Kevin participated all parts through the lab. Byron is mainly responsible for implementing the toggle of the LED and the read out for the internal voltage; Kevin is mainly responsible for implementing the read out for the temperature and the modes switching between voltage and temperature.

APPENDIX A PICTURES

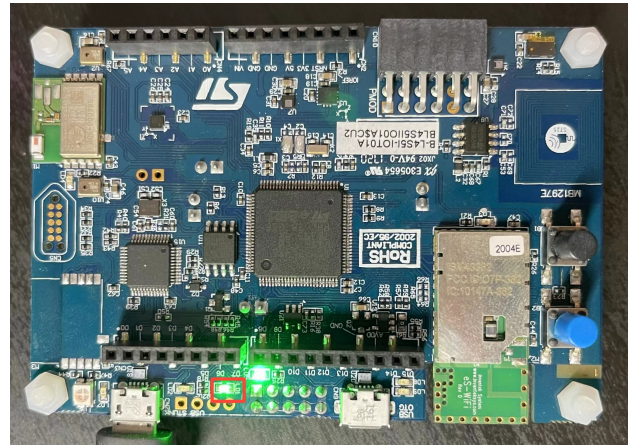


Fig. 1. Illustration of the LED that Shows the Mode (The LED Squared by Red Box)

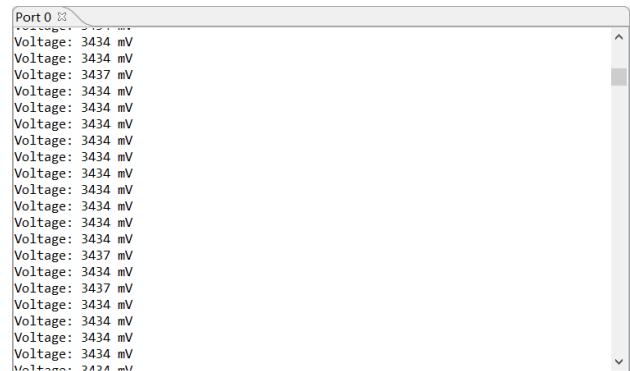


Fig. 2. The Print Out of the Voltage

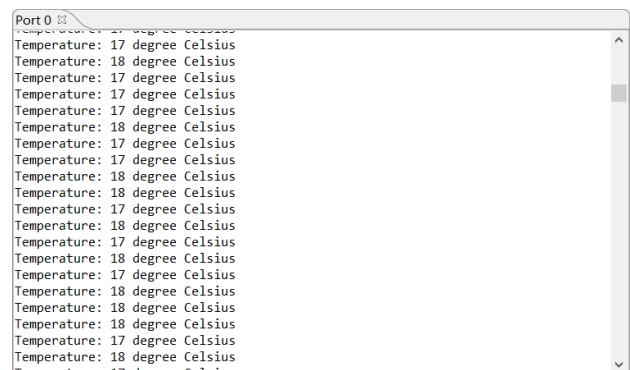
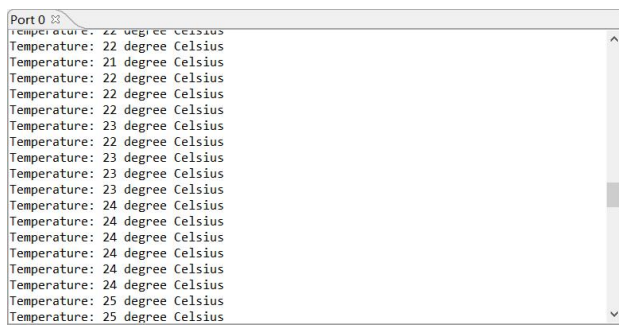


Fig. 3. The Print Out of the Temperature



```
Port 0
Temperature: 22 degree Celsius
Temperature: 22 degree Celsius
Temperature: 21 degree Celsius
Temperature: 22 degree Celsius
Temperature: 22 degree Celsius
Temperature: 22 degree Celsius
Temperature: 23 degree Celsius
Temperature: 22 degree Celsius
Temperature: 23 degree Celsius
Temperature: 23 degree Celsius
Temperature: 23 degree Celsius
Temperature: 24 degree Celsius
Temperature: 24 degree Celsius
Temperature: 24 degree Celsius
Temperature: 24 degree Celsius
Temperature: 24 degree Celsius
Temperature: 24 degree Celsius
Temperature: 25 degree Celsius
Temperature: 25 degree Celsius
```

Fig. 4. The Print Out of the Temperature When Heating the Core

REFERENCES

- [1] Project Description.
- [2] Datasheet - STM32L476xx - Ultra-low-power Arm® Cortex®-M4 32-bit MCU+FPU, 100DMIPS, up to 1MB Flash, 128 KB SRAM, USB OTG FS, LCD, ext. SMPS. STM.
- [3] STM32L4+ Series advanced Arm®-based 32-bit MCUs. STM.