***Efficient Embedded Course***

**LAB 5**

**ADC LAB EXERCISE:**

**VOLTAGE MONITOR**

Note. The figures shown in solutions may vary subject to different experimental environments

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc87592405)

[1.1 Lab overview 1](#_Toc87592406)

[2 Learning Objectives 1](#_Toc87592407)

[3 Requirements 1](#_Toc87592408)

[4 Details 2](#_Toc87592409)

[4.1 Hardware 2](#_Toc87592410)

[4.1.1 Connections 3](#_Toc87592411)

[4.2 Procedure 3](#_Toc87592412)

# Introduction

## Lab overview

In this project you will use the ADC to measure the voltage of the coin cell BT1 and the USB bus supply voltage rail. This can be useful for systems which rely on the battery as a back-up power source.

# Learning Objectives

* Implement an ADC based voltage monitor and compile C program to read the ADC input.

# Requirements

In this lab, we will be using the following hardware and software:

* **Keil µVision5 MDK IDE**
  + Please see the included Getting Started with Keil guide on how to download and install Keil.
* **STM32 Nucleo-L552ZE-Q**
  + For more information, click [here](https://www.st.com/en/evaluation-tools/nucleo-l552ze-q.html).
* **Coin cell**

# Details

## Hardware

Please see the included Nucleo-L552ZE-Q pins legend (NUCLEO\_L552ZE\_pins.docx) for the pinout of the Arduino-included Zio connectors for CN7, CN8, CN9 and CN10.

Figure . Typical lithium coin cell discharge voltage curve.

Figure 2 shows the voltage of a lithium CR2032 coin cell as it is discharged with a constant resistance load. We can connect the battery voltage directly to an ADC input. However, we cannot do this to measure the USB bus voltage for two reasons. First, connecting a voltage higher than VDD + 0.3V to the input will damage the ADC input. Second, the ADC will return a reading of all ones if the input voltage is above the reference voltage (3.3V here).

Diagram

Description automatically generated

Figure . Voltage divider for battery and USB bus monitoring.

We will use the resistive voltage divider shown in Figure 3 to reduce the input voltage to a level somewhat below the reference voltage. Choosing the component values R1 = 1 M and R2 = 1 M will scale the voltage by a factor of 1/2 = 0.5. A 3.0V input will be scaled down to 1.5V. We will need to compensate for this scaling in our code.

Note that we choose high-resistance values for R1 and R2 so the total current through them is small (1.5 A), limiting additional battery discharge. Also, note that we can change the resistor ratio to read the different voltages from different types of battery (e.g. lithium ion, lead acid, nickel metal hydride) or buses.

### Connections

Connect the switch signals to GPIO port signals on the MCU as shown in Table 1 below. This matches the pins used in the furnished code.

Table . Switch signals and connections

|  |  |  |  |
| --- | --- | --- | --- |
| Signal name | Description | Direction | MCU |
| VBattery | Battery voltage | Supply |  |
| 5V\_USB | USB supply voltage | Supply |  |
| VADCIn | Divided input voltage | Input to MCU | PA\_3 |
| GND | Ground | Power |  |

Build the circuit shown on your breadboard.

* To monitor the USB bus voltage, connect the voltage divider to the ADC channel.
* To monitor the battery voltage, connect a test clip or solder a small wire to a positive terminal (1 or 2) of a battery holder. Connect this wire to the ADC input.

## Procedure

1. Measure the voltage of the battery (VBattery) with a multimeter. VBattery = \_\_\_\_\_\_\_\_\_\_\_\_\_V
2. Measure the divided voltage of the battery (VADCIn) with a multimeter. VADCIn = \_\_\_\_\_\_\_\_\_\_\_V
3. Measure the 3V3 supply rail, which is connected to the ADC’s high reference voltage.   
   VRefH. = \_\_\_\_\_\_\_\_\_\_\_V
4. Run the lab code and set a breakpoint in the main function to examine the ADC output (res) and the computed battery voltage vbat. What is vbat, and how closely does it match the battery voltage you measured with the multimeter? vbat = \_\_\_\_\_\_\_\_\_\_\_\_\_. \_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Measure the resistance of R1 and R2 with a multimeter. What are the actual resistances?   
   R1 = \_\_\_\_\_\_\_\_. R2 = \_\_\_\_\_\_\_\_\_\_\_.
6. Based on R1 and R2, what should the scale factor in the code be? \_\_\_\_\_\_\_\_\_\_\_.
7. Update the code to use the correct values of R1 and R2. Is the vbat result more accurate now, and how much error remains? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
8. Run the main loop five times and record vbat each time below.   
   vbat(1) = \_\_\_\_\_\_\_V

vbat(2) = \_\_\_\_\_\_\_V

vbat(3) = \_\_\_\_\_\_\_V

vbat(4) = \_\_\_\_\_\_\_V

vbat(5) = \_\_\_\_\_\_\_V

1. Was vbat constant, or were there variations?
2. Modify the software to enable software averaging over 10 samples (hint: use a for loop and calculate the mean of vbat). Run the loop five times and record vbat each time below.   
   vbat(1) = \_\_\_\_\_\_\_V

vbat(2) = \_\_\_\_\_\_\_V

vbat(3) = \_\_\_\_\_\_\_V

vbat(4) = \_\_\_\_\_\_\_V

vbat(5) = \_\_\_\_\_\_\_V

1. Was vbat constant, or were there variations? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. How do these results compare to those without averaging? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Measure the voltage of the USB bus (VBattery) with a multimeter. VUSB = \_\_\_\_\_\_\_\_\_\_\_\_\_V
4. Measure the divided voltage of the battery (VBatDiv) with a multimeter. VBatDiv = \_\_\_\_\_\_\_\_\_\_\_V
5. Optional: Why won’t this system work if the circuit runs off of the battery? Extra Credit: Modify the code to support running off the battery.

The battery voltage is not regulated, so the P3V3 supply rail voltage is unknown. As a result the reference voltage will be unknown, so you won’t be able to scale the ADC output. To solve this problem use an external fixed bandgap voltage reference. Using this you can calibrate your ADC. The process involves measuring the bandgap with the ADC. Using the result nband-gap, the program can compute the input voltage vin =Vband-gap \* nin/nband-gap.