ECSE444 Microprocessors

Winter 2023

Lab 2: Basic I/O and ADC-based Readout of Temperature and Voltage

You have developed in the previous laboratory exercise a complete C/assembly code for the STM32L4+ processor. This lab exercise will teach you to use the General-Purpose Input/Output (GPIO) pins of the processor, as well as some of the built-in blocks that track the physical status of the processor. There are two readily available physical parameters that the STM32L4+ processor can read: its core temperature and its voltage.

The program that you are asked to build will allow you to switch between updating the readout of the temperature and voltage each time the blue button is pressed on your development kit. The two variables expressing these two physical values will be observable by means of "watched variables" that the processor and the IDE allow you observing during the ongoing processor operation

This exercise relies on the previous laboratory exercise, all the classes and tutorials, and it will focus on the use of basic hardware blocks within the processor. In addition to consulting the class notes, you should consult the processor documentation to complete this exercise. Some specific hints will be given in Tutorial 2 and the lectures leading to this lab exercise.

Background

GPIO Pins Use

You saw in the class that GPIO pins can be programmed to perform digital bit input reading, digital pin output, various special functions, as well as the analog quantity reading/writing. You will exercise the first two capabilities in this lab.

Output PIN – LED Driving

The simplest GPIO operation is that of outputting a zero or one, and you can test this operation by driving the green LED light on your board. From section 6.12 of the board user manual available on MyCourses, you will see the pin PB14 drives LED2. To drive that pin, you have to enable it in Cube MX (started by double-clicking the .ioc file of your project) and declare as an output (by selecting System Core->GPIO, and then the pin PB14). The tool will allow you to label this pin with a useful mnemonic, as well as set some other parameter (e.g., pull-up or -down). Consider carefully the suitable values for those by consulting the board schematic and the basic electric circuits knowledge.

Input PIN – Button Readout

Similarly, you can locate the blue button pin and configure it as an input. Again, consult the circuit schematic to deduce the default state (i.e., when button is not pressed) and the potential need for

pull resistors to be included in the pin configuration.

Analog to Digital Converter (ADC)
STM32L4+ processors are equipped by flexible high-performance ADC converters, as described in detail in Sec. 21.2 of the STM32L4+ Reference Manual, available on MyCourses. Commonly, an ADC would read analog quantities via external pins, but it is simpler to first focus on reading out a) the internal reference voltage and b) the internal temperature sensor.

The simplest way to obtain a correct value from an ADC is to start conversion and then read out the value after the conversion is completed, and we will use this method.

Lab 2 - Exercise

You will perform this exercise in several stages and record the results obtained at each step in your lab report.

Step 1
First, you will write a main C program that detects the pressing of the button and tests the LED2 display by toggling the value every time the button pressed. It is perfectly acceptable to have this program run in an infinite loop generated by Cube MX after you configured the two GPIO pins. You should take note of the commands added for initializing the GPIOs, as it illustrates a way by which the Cube MX GUI generates correct code for your program.

Step 2
You will configure the ADC1 to read out the reference voltage and augment your code to convert the value read to a correct voltage value. After generating the code from Cube MX, please take notice of the control structures instantiated for ADC1 and the values generated (especially the assignment of the reference voltage as the source for ADC conversion). You can test the obtained values by observing that variable in your code – no need to interrupt the execution.

Step 3
Similar to Step 2, you should configure the ADC1 to read out the temperature, and then add a correct conversion into your program variable, such that the value of that variable is the temperature in degrees Celsius.

Step 4
This is the “detective” part of your exercise, where you will compare the code obtained in steps 2 and 3, such that your program control can switch the readouts between the voltage and the temperature. It helps to consult the reference manual to find how to correctly initialize the readings “on fly” to avoid the errors in the procedure.

Step 5
Produce the final program that alternates between the ADC readings and prepare it for the demonstration to a TA. The LED light should help in understanding what the processor is doing.

Useful Notes

In realizing your code, you are free to use good C coding practices, such as the use of conditional compiling for retargeting to different execution cases given above.

If you start your project by using the board support package (BSP) for your board, the clock and pins will be properly selected for you. It is a good practice to check the clock and

The Debugger

While developing your code, you will spend substantial time using the debugger. Please follow the instruction from your tutorials to ensure proper development and debug practices.

Experimental Results to Report

You are asked to reach the following milestones and include the results in your report.

1. Describe the configuration of pins such that the Step 1 is correctly run on the processor,
2. Describe the ADC1 configuration for Step 2,
3. Describe the ADC1 configuration for Step 3,
4. Document the code for programmable reconfiguration of ADC1.
5. Readable and concise code/pseudo-code for your final program execution
6. Test the temperature sensor readings by (non-destructive!) heating and cooling of the processor. Fingers pressed on the processor can raise temperature a bit, and natural convection will reduce it. You can also consider the use of a hair fan.

Final Report

Once you have all the parts working, include all the relevant data to your report, which will be due first day in a week following the Lab 2 demonstration, but not later than 4 days after the demo.

Demonstration

The demonstration includes showing your source code and demonstrating a working program. Your program should be capable of manipulating a variety of test cases and should flag errors appropriately.

Report

The report should concisely explain your solution to the problem given, including the final code. All code should be well documented. Your report should contain a performance evaluation and correctness validation. More detail on the report will be given out in the class.

Due Dates

The first two labs will be completed in several phases, over the three weeks. First, you should take time to understand the lab and ask any questions in regular lab sessions or through discussion groups.

There will be the first lab demonstration on

Feb. 14-16th,

by which time you should be able to explain to the TAs how you approach the exercise.

The final demonstration in which the parts are put together and run on test cases will be on

Feb. 22nd and 23th

and will include showing your source code and demonstrating a working program for all test cases that we will post.

The final report will be due on

Friday, Feb. 24th.