

Lab 8. Temperature Monitoring Through Analog to Digital Conversion (C)

Preparation

You will need a LaunchPad and a laptop/computer with Keil uVision5 installed. Download the Lab8 starter project from Canvas.

Starter project Lab8

Purpose

The purpose of this lab is for you to learn how to use the ADC to convert an analog voltage to a digital value.

Introduction

For this lab you will be modifying a given source project in C for temperature monitoring. You will need to include your Lab 7 code for initializing UART0 and reading a character input. Your project will first print the temperature in Celsius. Next, you will modify the print function to also display the temperature in Fahrenheit. The temperature is measured using a temperature sensor that is internal to the CPU on the board. After that, you will modify your code so that the user can choose whether to output the temperature in Celsius or Fahrenheit. Lastly, you will use the disassembly window in the debugger to observe how the compiler converts your C code into ARM Assembly instructions.

Resources

PuTTY – An open source program that will allow serial communication between your board and your computer. You can download the installer file or a copy of the executable from the following link: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>. You will need to configure your connection when you first launch PuTTY. Choose Serial as your connection type. Next, select your COM port and a speed of 9600. You can determine the COM port by checking your Device Manager to see which one your board is connected to. A terminal window will be launched after clicking “Open”.

ADC – Be sure to watch the “ADC Initialization Ritual” and “Capturing a Sample” videos on the main page of Canvas. Use the ADC chapter from the actual physical version of the textbook or the ADC slides from the main course page on Canvas. You can also refer to Chapter 13 of the Tiva’s Data Sheet (<http://www.ti.com/lit/ds/spms376e/spms376e.pdf>) for information about the on-chip temperature sensor.

Procedure

1. Begin by Downloading the project files and inserting your code from Lab 7 for InitConsole() and myGetChar(). Skeletons for those functions are located in the file lab7.c. Next, you need to configure the ADC to read from the temperature sensors. Most of this code is provided, you just need to choose a sampling rate of 500K samples/sec.

The major component for this part is completing the function Get_TempC() which will trigger the ADC and return the temperature measurement in Celsius. To do this, you should complete the functions ADC0_In(), Convert_Raw_To_V(), and Convert_V_To_C() and then call them within Get_TempC(). More information about each function can be found in main.c.

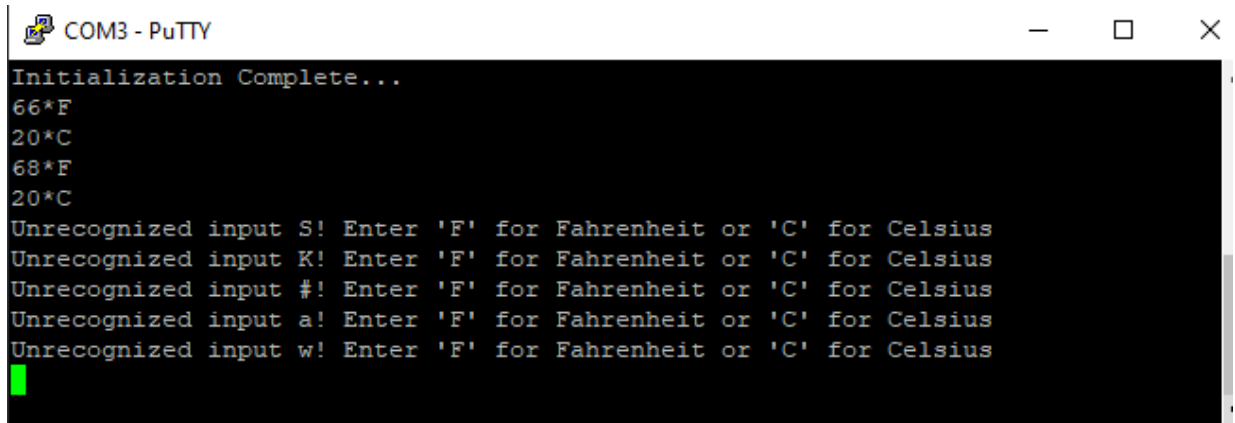
Once completed, your program should output the current temperature to the terminal every 100ms. This should be a normal room temperature value. You can increase the temperature by heating up the main chip through methods such as holding it against your hand. If you are not initially getting a value around room temperature, then you need to determine if the issue is with your ADC0_In() or one of the conversion functions. You can hardcode the raw ADC output as 1746 rather than calling your ADC0_In(). This should produce a temperature reading of 42C if your conversion functions are correct. Demonstrate your working code to the TA.

Signature_____ Date_____

2. Now you will add some functionality to the program. First, fill in the function `Convert_C_To_F()` so that it converts a given temperature from Celsius to Fahrenheit. Next, you need to modify the `Print_Temps()` functions so that it calls `Convert_C_To_F()` and displays the temperature in both Celsius and Fahrenheit. Skeletons of these functions are located in `main.c`. Demonstrate your modified program to the TA.

Signature_____ Date_____

3. Next modify the program so that rather than streaming the temperature values it instead only transmits the temperature value when it is prompted. Your program should listen for incoming characters. If the character is “f” or “F”, then it should respond with the temperature value in Fahrenheit. If the character is “c” or “C”, then it should respond with the temperature value in Celsius. For any other character your program should respond with some sort of error message showing what key was pressed while also describing acceptable inputs. An example is shown below. Demonstrate this behavior to the TA.



```
COM3 - PuTTY
Initialization Complete...
66°F
20°C
68°F
20°C
Unrecognized input S! Enter 'F' for Fahrenheit or 'C' for Celsius
Unrecognized input K! Enter 'F' for Fahrenheit or 'C' for Celsius
Unrecognized input #! Enter 'F' for Fahrenheit or 'C' for Celsius
Unrecognized input a! Enter 'F' for Fahrenheit or 'C' for Celsius
Unrecognized input w! Enter 'F' for Fahrenheit or 'C' for Celsius
█
```

Signature_____ Date_____

4. Lastly, we will use the debugging and disassembly features to investigate the C compiler. To begin, place a breakpoint at the beginning of the `Convert_C_To_F()` function. Now launch the debugger and run your code. The debugger should stop at the breakpoint inside the function. You will need to enter either a “f” or “F” on your keyboard (if you are doing the lab steps in order) before your running program will execute that function. Observe the disassembly window. How does the conversion function compare in C and ARM Assembly? Is it possible to change your C code to require fewer assembly instruction? What happens if you calculate the temperature using decimal numbers? Without using decimal numbers? **Be prepared to discuss your answers with the TA and to explain why you think you observed these results..**

Signature_____ Date_____