ECGR 4101/5101 - Fall 2014

Lab 3: Creating Your Own Library and Analog to Digital Converters

Objective:

In this lab, you will learn how to create your own C library files, and call an ADC reading from a function contained within it. The initialization and reading code will be created by setting individual registers without the use of any driver functions provided by Renesas.

Assignment:

1. Begin by creating an empty project workspace. Consult the supplemental instruction from the previous lab if you need help doing this. If you do, at the step where you select “Tutorial Project”, select “Empty Application” instead and click “Finish.”
2. Once the project is set up, go to file->new to open a blank file. We will save this file as “customADC.h” under the src directory in your workspace. We will then open another blank document and save it in the same directory as “customADC.c”. We do this because libraries will usually have a .h file to accompany a .c file.
3. Now we will populate the .h files with the function prototypes. Header files contain #define macros, function prototypes, and sometimes, but not often, entire functions. Under normal circumstances, functions should only be written within the .c file which the header file corresponds to.
4. In your header file, include the board’s header file
    a. #include "platform.h"
5. In your header file, create 2 function prototypes:
    a. void adcInit(void); // This function initializes the ADC for us
    b. int adcRead(int channel); // This function will read an analog value from the specified channel

        adcInit() will be used to initialize the 12 bit ADC by starting the peripheral clock, selecting alignment settings, mode, etc. adcRead() will be used to initiate an ADC conversion on the channel specified by the “int channel” parameter, wait for the conversion to finish, then return the result.
6. Now in the customADC.c file, we will implement these functions.
    a. In adcInit() you will need to do the following:
        i. Power on the 12 bit ADC peripheral by enabling it’s clock
        ii. Set up the I/O pins the ADC inputs are attached to
        iii. Configure the 12 bit ADC Control Register for single cycle mode, no interrupts
        iv. Configure the data alignment register
        v. Configure the ADC trigger to use software triggering
    b. In adcRead() you will need to do the following:
        i. Select the 12 bit ADC channel to read from
        ii. Begin conversion
        iii. Wait for conversion to finish
        iv. Return result

To learn about the 12-bit ADC register, be sure to consult the data sheet, the course textbook, and even the tutorial code for information on how to create these functions.

7. Once you have finished creating the functions, add the adcInit() function somewhere in your main() before the while(1). Add adcRead() in the while loop and pass it 0x0004 to read from the channel that the potentiometer is attached to. Take the ADC value returned by the adcRead() function, convert it to a string, and write it to the display using the lcd_display() function. To do this, you will need to learn how to use the sprintf() function.
8. Now you can test your custom ADC functions. Be sure everything is working before continuing on from this point.
9. We will do a little more abstraction before moving on from this point. In customADC.h, let's add some defined macros to pass to the adcRead() function for channel selection. Make definitions for AN0, AN1, … AN15. That way we can call adcRead(AN2); to read from channel 2. Creating definitions removes "magic numbers" from your code and makes it easier to read.
10. Let's add another function to our .c and .h files to convert the digital value to the voltage value. Create a convertADC() function. This function will be passed the ADC value as an int, and return the voltage as a float. Make the function flexible, so that if in the future we can use this function for converting ADC values from the 10 bit ADC as well, along with various reference voltages. We want to create robust functions so that we will be able to use them for various purposes in the future. You will be graded on how well you design your function.
11. For the final demonstration, your program will need to have the following requirements:
    a. Display your team's name on the first line of the LCD
    b. On the second line, display "ADC Test"
    c. When switch 1 is pressed, read the voltage from channel 2 (the potentiometer's channel) and have the third line display "Channel 2:" and the fourth line display the converted ADC value.
    d. When switch 2 is pressed, take another reading from channel 2 and display the same thing, except on the fourth line display the actual voltage value using your convertADC() function.
    e. When switch 3 is pressed, take a reading from an ADC channel of your choice and display the channel number on the third line and the ADC reading from that channel on the fourth line. Make sure you select an ADC channel that has a pin available on one of the board's headers so we can connect a voltage supply to test.

Graduate Students Only:

1. Now we will add a little more functionality to our library. We will add support for continuous sampling as well. Change your adcInit() function to allow you to pass a parameter to it to select between single cycle (one shot) mode or continuous scan. Then, you will need to create two more functions: beginScan() and readChannel().
   beginScan() should be able to accept multiple channel parameters which are of the same type that were passed to adcRead(). For example, if I wanted to set up continuous scans on channels 1 and 2, I can call beginScan(AN1 | AN2), where AN1 and AN2 are the same defined values already created in our header file. readChannel() should be created such that I pass it the value from a single channel that I want to read.

2. Modify your main() function such that you set up a continuous scan on 2 other channels. Display the channels selected on lines 5 and 7 or your LCD. Under each of these lines, read and display the current adc value for each respective channel.
3. For the final demonstration, your program will need to have the following requirements:
   a. Display the current adc value for the 2 channels you have selected on lines 5-8.

To Submit:

- A pdf containing your project report. Paragraphs describing the objective, what you did to complete that objective, and code snippets throughout the description as they are relevant. When describing the library function you created, be sure to explain any design decisions you made.
- Your main.c, customADC.c, and customADC.h files
- A zip file containing the 4 files mentioned above
- Your lab check-off sheet at the demonstration

Grading will be changed such that 40% is your demonstration, 20% will be your code structure, and the remaining 40% will be your report. Your code will be graded based on comments, proper alignment (indentations), function design, proper capitalization of definitions and variables, and design of custom functions.

Embedded Systems Lab Demonstration Validation Sheet

This sheet should be modified by the student to reflect the current lab assignment being demonstrated

Lab Number:	Type the Lab Number Here
Team Members	Team Member 1 Name Here Team Member 2 Name Here
Date:	MM/DD/YYYY

Lab Requirements

Obtain a list of the Lab requirements from the end of the lab handout and type them here, perform a self-review and indicate with an X if you met each requirement or not.

REQ Number	Objective	Self-Review	TA Review
1	Team name and ADC Test is displayed		
2	SW1 reads the correct ADC value from channel 2		
3	SW2 reads the correct ADC value from channel 2 and displays the converted voltage value		
4	SW3 reads from a different channel and displays the correct voltage		
G1	Graduate Students Only: Two channel's current values are displayed on LCD line 5 - 8		

Comments

If you were unable to meet a particular requirement or if a requirement required a custom implementation describe briefly why you were unable to meet the requirement or what your custom implementation was in the box below, also if you have any additional considerations that you would like to be taken into account while grading you may type them here as well.
