Reading the Analog to Digital Converter in C

Objective: Example the operation and accuracy of the MC9S12XEP100's ADC. For this lab you will design and write a C language program that reads the potentiometer on Analog to digital converter (ATD) channel 2 and displays the converted digital value as binary output on the LEDs. You will analyze the ADC's linear input/output relationship performance and accuracy.
This is a two Part lab. For Pre-Lab and Part I you will work individually the first week and For Part II you will work in teams for the 2nd week. Here are the instructions for Pre-lab and Part I.

Terminology:
ADC: Analog to Digital Converter, converts analog input voltage to digital format. In the MC9S12XEP100 chip data sheet the ADC is referred to as A/D and/or ATD.
VDC: Voltage Direct Current
V_{ref}: Reference Voltage, range of possible input values the ADC can read.
V_{in}: Input Voltage, this is the analog input voltage reading
D_{value} or D: Digital Output Value, this is the input analog voltage reading converted to a digital value
Resolution or Bit mode conversion: the number of bits used to represent the converted value. This setting determines accuracy of conversion. For example: An 8-bit ADC with $5V_{ref}$ can represent 0 to 5V with 2^8 possible unique values (0 to 255) and a10-bit ADC with $5V_{ref}$ can represent the same 0 to 5V with 2^{10} possible unique values (0 to 1023).

Pre-Lab:
Our MC9S12XEP100 chip contains two 8-channel successive approximation ADC (ATD0 and ATD1), and each channel is capable of either 12, 10 or 8-bit mode conversions. Three channels of the ATD0 have been assigned to input devices on our training board and each have an input range V_{ref} of 0.0 to ~5.0 VDC.

- ***Analog Temperature Sensor***
  The Temperature sensor circuit is connected to A/D control register ATDCTL 0. The temperature is measured in 10mV/°F.
- ***Analog Pressure Sensor***
  The pressure sensor is connected to A/D control register ATDCTL 1. The sensor is a MPX5050GP sensor from Freescale Semiconductor. The sensor can measure up to 7.25 PSI and it outputs from 0 to ~4.7V.
- ***Analog Potentiometer***
  The potentiometer is connected to A/D control register ATDCTL 2. This simple analog input allows the user to vary the input voltage to the A/D converter between 0 to ~2.5V.

For example: An ADC channel, set to an 8-bit mode resolution, will convert measured input voltage levels (0.0V to 5.0V) in the form of a linear input/output relationship range of 0 to 255_{10}. You can calculate the theoretical linear input/output relationship and solve for the digital output as follows:

If an 8-bit ADC has a reference voltage (V_{ref}) of 5.0V and reads a $2.5V_{in}$ value, you can calculate the Digital output (D_{value}) by the following equation:

$$D = V_{in} / V_{ref} * (\text{resolution} - 1)$$

Then, $D = (2.5/5 * (2^8 - 1) \Rightarrow D = 127.5$ and since this is a digital value we must use a whole number, we would use normal rounding ~128 which is in 8-bit mode $1000\ 0000_2$ as the D value.

Pre-Lab questions:
1. For this lab; What is the source of the input voltage the ADC will read?
2. For this lab; to what ATD channel is the input voltage source connected?
3. For this lab; Where is the ADC 8-bit Digital output value displayed?
4. For this lab; How will you read the Digital output value, hex, decimal, or binary?
5. An 8-bit ADC has a V_{ref} of 5.0V and reads a analog potentiometer at $1.8V_{in}$

   a. Calculate the theoretical Digital output (D_{value})
   b. Calculate the quantized value per bit (step value)?
6. Flow chart Part I program

Instructions:
Part 1: (work individually 1st week)
Design and write a C language program that reads the potentiometer on Analog to digital converter (ATD) channel 2 and displays the converted digital value as binary output on the LEDs
Follow the slides from Lecture to:

1. Power-up Analog converter.
2. Wait for power to stabilize if necessary, in the application.
3. Set 8-bit mode, 4 conversions on a single channel (register)
4. Start conversion specifying channel (register).
5. Wait for conversion to complete.
6. Read ADC data. Then, output reading to LEDs

- The Analog to Digital *initialization* software (code) should be in a function (only the code to power up the ADC).
- The Analog to Digital *conversion and read* software (code) should be in another function that appropriately starts and waits for a reading and returns an 8-bit value (character) that will be directly written to the LED port.
- The main function software (code) should make function calls to the Analog to Digital functions
- The main function software (code) should write the returned conversion to the LEDs
- The program should execute in a continuous loop making conversions and displaying on the LEDs.
- When program is running: As you turn the potentiometer knob, you should get values from approximately 0 -127_{10} on the LEDs.

  ATD Channel 2 is wired to the potentiometer knob on the training board.

ATD Registers:	~~ADPU (PowerUp, in ATD0CTL2 register)~~*
	ATD0CTL1 (Control Register) set 8, 10, 12 bit resolution**
	ATD0CTL5 (Control Register) start conversion
	ATD0STAT0 (Conversion Complete Register SCF:bit 7)
	ATD0DR#H (Data Registers where # is the channel reading and H is formatting of the reading)
LEDs:	PORTC – LED output port
	DDRC – LED data direction port

*This register is set to default start-up values and we will not change the defaults. Therefore, we do not need to write to this register in this program. Meaning you may omit this line of code.

** Refer to:

- MC9S12XEP100 Data sheet Chapter 13: 8-bit mode: bits 5 and 6 set to zero.
- ATD Converter Register Datasheet: Reference posted in Canvas Resources Module
- I/O Ports Hardware Configuration: posted in Canvas Resources Module
- Lecture 9a Notes

Submit: Your completed Pre-Lab program outline including Program Development Cycle worksheet (flow chart), well formatted word document including all section as instructed in Pre-lab assignment. Your Part I solution main.c file completed, fully commented. Be sure your code is well organized, includes a commented header, and logical comments. And your entire zipped project folder.