Khalifa University of Science, Technology and Research
Electronic Engineering Department

ELCE333Microprocessor Systems laboratory

Laboratory Experiment 8

Analog to Digital Converter

Lab Partners

Leena ElNeel	100037083
Muna Darweesh	100035522
Shamsah AlNabooda	100036984

Date Experiment Performed: 25/3/2015
Date Lab Report Submitted: 1/4/2015

Lab Instructors:

Mahmoud Khonji

Mohammed Ali Saif Al Zaabi

Spring 2015

Table of Contents

List of Figures and Tables

Summary

This experiment introduces the students to the concept of Analog to Digital Converter. The first task analyzes the difference between 8 and 10 resolutions. Then, the next task is to use the temperature sensor to sense the temperature and display them on the LCD. Finally, the speed of the flashed LEDs is altered by using the light sensor.

1. Introduction

In the final Microprocessor Systems Laboratory, we are taking the topic of changing analog to digital signals. Microcontrollers are involved with pressure and temperature measuring through sensors using a method of quantization. A signal must be converted so the microcontroller can read and process it. ATDs have 8, 10, 12, 16 or 24 bits. A note to be known is that the higher the resolution is the faster the conversion time. In our laboratory we use the HCS12 chip which comes with 16 channels of ATD. Each of the ATD0 and the ATD1 has 8 channels and both makeup the ATD. ATD control registers have the following functions:

Control register 2: enable ATD convertor

Control register 4: set the clock frequency, length of sample time, resolution

Control register 5: select type of conversion and input channel

On the dragon board, the following peripherals are connected to the channels:

Channel 7: potentiometer

Channel 5: temperature sensor

Channel 4: light sensor

***Aim**:* To introduce the students to use of analog to digital (ATD) converters.

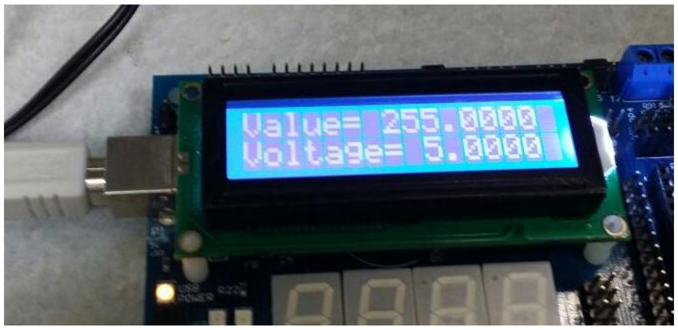
Objectives:

- Understand the concept of analog to digital (ATD) converters.
- Get introduced to the concept of ATD subsystem initialization and conversion.
- Gain the experience using the 16 channels A/D converter of the DRAGON12.
- Use onboard peripherals related to ATD
- Use the CodeWarrior IDE for the development of HCS12 microcontroller C programs.
- To compile, download and debug/test a C program using CodeWarrior C compiler and Dragon12 Plus Trainer board.

2. Design and Results

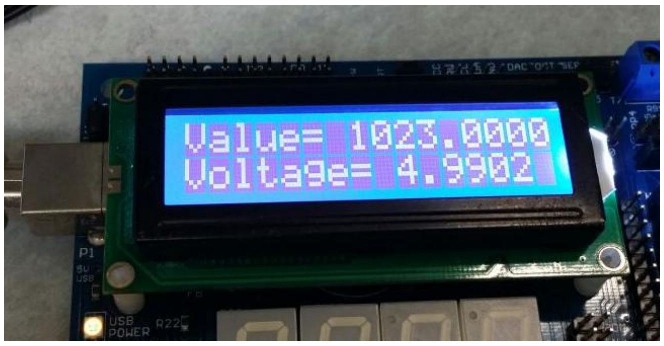
Task1: 10-Bit resolution

In this task, we were asked to implement the given task in the introduction and by connecting the potentiometer which is found in channel 7.The output we got was a voltage of 5 and a value of 255. The result was showed in Figure 1 below. Then, we were asked to modify the program so we have a 10 bit resolution. The edited code is shown below. The output result was a value of 1023 and the voltage of 4.9902 which was viewed on the LCD and shown below in Figure 2. The results match the expected value since the maximum output is in the range between (0-5)V.



```
Value= 255.0000
Voltage= 5.0000
```

Figure 2: Task1 part 1 result



```
Value= 1023.0000
Voltage= 4.9902
```

Figure 2: Task1 part 2 result

```
#include <hidef.h> /* common defines and macros */
#include "derivative.h" /* derivative-specific definitions
*/
#include "lcd.h"
void ATD_init(void)
{
ATD0CTL2_ADPU = 1; // power up ATD channel 0,
disable interrupts
delay(1); // wait for ADC to warm up
ATD0CTL4 = 0x05; // 8-bit,sample time 2 ADC clock,
prescale of 5,
}
int ATD_CONVERT()
{
ATD0CTL5 = 0x87; // channel no. 7 and right justified
while(!(ATD0STAT0 & 0x80)); // wait for conversion to
finish
return(ATD0DR0); // get and return the value to the caller
}
void main(void) {
int val;
float out;
LCD_Init();
ATD_init();
for(;;) {
val=ATD_CONVERT();
out=((val*1.0)/205); //output in Voltage = conversion
result *step size
//out =conversion result *((VrefH-VrefL)/(2^resolution-
1))
LCDWriteLine(1,"Value= ");
LCDWriteFloat(val);
LCDWriteLine(2,"Voltage= ");
LCDWriteFloat(out);
delay(100);
LCD_clear_disp();
_FEED_COP(); /* feeds the dog */
} /* loop forever */
/* please make sure that you never leave main */
}
```

Task 2: Temperature sensor

In this task, it is required to write a C program in order for it to read the temperature sensor output then display it as a voltage on the LCD. In the first line of the LCD the temperature will be written as a voltage whereas in the second line it will be represented in Celsius. The resolution is used is 10-bits. As the temperature changes the readings on the LCD should be changed too. It is given that the sensor's resolution is 10 m V/(Celsius degree). The code is as the following. The code follows the same concept as what is in task1. We modified the channel to be 5; hence, the ATD0CTL5 register should be modified to 0x85; to enable the channel number 5 and 8 to enable the MSB which is called DJM to 1. The extra line is the part that identifies the resolution of the temperature sensor, the command tempc= out /0.01, and this will present the values of the temperature in Celsius.

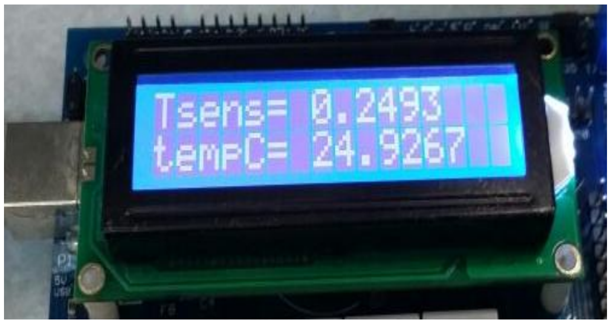


Figure 4: Task2 result

Then we heat the temperature by our hands and the result is shown in the figure above. The first line is the value in voltages and the second is the temperature which the room temperature.

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"
void ATD_init(void)
{ ATD0CTL2_ADPU = 1;
 // power up ATD channel 0, disable interrupts
  delay(1); // wait for ADC to warm up
  ATD0CTL4 = 0x05;
 // 10-bit,sample time 2 ADC clock, prescale of 5,}
int ATD_CONVERT()
{ATD0CTL5 = 0x85;
 // channel no. 5 and right justified
  while(!(ATD0STAT0 & 0x80));
// wait for conversion to finish
  return(ATD0DR0);
// get and return the value to the caller}
void main(void) {
  int val;
  float tempc , out;
  LCD_Init();
  ATD_init();
  for(;;) {
    val=ATD_CONVERT();
    out=((val*5.0)/1023);
//output in Voltage = conversion result *step size
//out =conversion result *((VrefH-VrefL)/(2^resolution-
1))
    tempc= out /0.01;   // the sensor resolution
    LCDWriteLine(1,"Tsens= ");
// Display at the first line of LCD
    LCDWriteFloat(out);
    LCDWriteLine(2,"tempC= ");
// Display at the second line of LCD
    LCDWriteFloat(tempc);
    delay(100);
    LCD_clear_disp();
    _FEED_COP(); /* feeds the dog */
   }//end of loop
} // end of main
```

Task 3: Light sensor

In the final task of this lab we were asked to write a C code that reads a light sensor output and flash LEDs accordingly. This program should also alter the speed of flashing LEDs according to the proximity of your hand to the sensor.

In order to so that, we've started the code in a similar way as in previous tasks. The first step in writing this code was powering up ATD channel 0 and disable the interrupts. We also have set the sample bit to be 10, the prescalar to 5, justification to right and channel to 4 (channel for the light sensor). After that, we've set the necessary ports for lighting up the LEDS and set the output retrieved from the sensor to a variable "*val*". This variable is used as input in the function *delay(int ms)* that is to be occurring between the process of flashing the LEDs and turning them off. At first, PORTB LEDS is flashed by setting it to 0xFF then a delay will occur before the LEDs are turned off by setting them to 0x00 in which the delay will vary according to the value retrieved from the sensor and hence your hand position form the sensor.

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"
void ATD_init(void)
{  ATD0CTL2_ADPU = 1; // power up ATD channel 0, disable interrupts
  delay(1); // wait for ADC to warm up
  ATD0CTL4 = 0x05; }
int ATD_CONVERT()
{ATD0CTL5 = 0x84; // channel no. 4 and right justified
  while(!(ATD0STAT0 & 0x80)); // wait for conversion to finish
  return(ATD0DR0); // get and return the value to the caller
}
void main(void)
{ long int val;
  int  mul;
  DDRB= 0xFF;
  DDRJ= 0xFF;
  DDRP= 0x0F;
  PTP= 0x0F;
  PTJ= 0x00;
  PORTB= 0xFF;
  LCD_Init();
  ATD_init();
  for(;;)
  {val=ATD_CONVERT();
    PORTB =0x0FF;
    delay (val);
    PORTB= 0x00;
    delay (val);
    _FEED_COP(); /* feeds the dog */  } //end of loop
} //end of main
```

Figure 5: Flashing lights according to light sensor

3. Assignment Questions

1) Write a C language program to change the speed of flashing of the PORTB LEDs based on the potentiometer value.

The answer for this question is similar to the code in task 3 in which the input retrieved from the potentiometer by the function *ATD_CONVERT( )* is used as input into the delay function that occur between flashing the LEDs and turning them off.

Also, a slight change have been made to the code of task 3 so the required functionality can work properly and that is changing the channel from 5 to 4 (as it is the corresponding channel to the potentiometer).

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"
void ATD_init(void)
{
  ATD0CTL2_ADPU = 1;
 // power up ATD channel 0, disable interrupts
  delay(1); // wait for ADC to warm up
  ATD0CTL4 = 0x05;
// 10-bit,sample time 2 ADC clock, prescale of 7,
}
int ATD_CONVERT()
{
  ATD0CTL5 = 0x87; // channel no. 7 and right justified
  while(!(ATD0STAT0 & 0x80));
 // wait for conversion to finish
  return(ATD0DR0); // get and return the value to the caller
}
void main(void)
{
  Long  int val;
  int  mul;
  DDRB= 0xFF;
  DDRJ= 0xFF;
  DDRP= 0x0F;
  PTP= 0x0F;
  PTJ= 0x00;
  PORTB= 0xFF;
  LCD_Init();
  ATD_init();
  for(;;)
  {
    val=ATD_CONVERT();
    PORTB =0x0FF;
    delay (val);
    PORTB= 0x00;
    delay (val);
    _FEED_COP(); /* feeds the dog */
    } //end of loop
} //end of main
```

2) Modify the program in task 2 to display the temperature on the second line of the LCD in Kelvin if SW4 is pressed and in Fahrenheit if SW3 is pressed.

In this question, the temperature channel will be assigned to channel 5 for temperature channel. The SW4 corresponds to Kelvin temp and SW3 corresponds to Fahrenheit. After getting the sensed value (tempc), the following equation is used

Temperatuer (K)=273.0+ tempc;

Temoeratuer in Fahrenheit = tempc * ((9.0/5.0)+32.0);

The results are as the following.

The temperature around 22℃

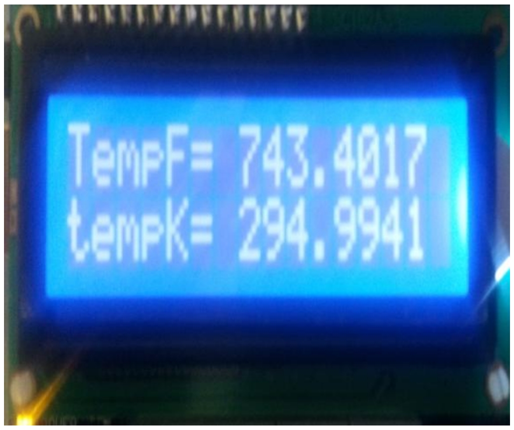


Figure 6: Assignment question 2

```
#include <hidef.h> /* common defines and macros */
#include "derivative.h" /* derivative-specific definitions */
#include "lcd.h"
void ATD_init(void)
{
  ATD0CTL2_ADPU = 1; // power up ATD channel 0, disable
interrupts
  delay(1); // wait for ADC to warm up
  ATD0CTL4 = 0x05; // 10-bit,sample time 2 ADC clock,
prescale of 5,
}
int ATD_CONVERT()
{
  ATD0CTL5 = 0x85; // channel no. 5 and right justified
  while(!(ATD0STAT0 & 0x80)); // wait for conversion to
finish
  return(ATD0DR0); // get and return the value to the caller
}
void main(void) {
  int val;
  float tempc ,tempk, tempf, out;
  DDRH= 0xFF;  // set DIp switch high to be able to use push
buttons
  PTH=0x0F;
  LCD_Init();
  ATD_init();
  for(;;) {
    val=ATD_CONVERT();
    out=((val*5.0)/1023); //output in Voltage = conversion result
*step size
    //out =conversion result *((VrefH-VrefL)/(2^resolution-1))
    tempc= out /0.01;

    tempk=273.0+ tempc;
    tempf= tempc * ((9.0/5.0)+32.0);

    if ( PIFH_PIFH2== 1) { // SW3

    LCDWriteLine(1,"TempF= ");
    LCDWriteFloat(tempf);
    }
     if ( PIFH_PIFH1 == 1) {  // SW4
    LCDWriteLine(2,"tempK= ");
    LCDWriteFloat(tempk);
     }
    _FEED_COP(); /* feeds the dog */
    } /* loop forever */
  /* please make sure that you never leave main */
}
```

4. Conclusion

This lab presents the ATD converter concept. The first task was basically about knowing how to change the resolution of a given code that discussing the way of changing the potentiometer position. This is done by modifying the LSB of ATD0CTL2 LSB which is called SRES8 to 0 if 8 bit resolution is required or 1 for 10 bit resolution. Also, the voltage is calculated using a formula that take the product of the conversion result and the difference between the high and low voltage then, divide that by ($2^{resolution} - 1$). The second task depended on the use of the temperature sensor and to display its voltage and Celsius value on the LCD using 10-bit resolution as well. However, the channel will be modified in ATD0CTL5 control register. The last task is to flash all LEDs and alter their speed using the light sensor. Their speeds will vary according to proximity of the hand to the sensor.