

**Khalifa University of Science, Technology and Research**

**Electronic Engineering Department**

**ELCE332 Microprocessor Systems laboratory**

**Laboratory Experiment 8**

**HCS12 Analog to Digital Convertor**

**Lab Partners**

Hamad Eisa Alhazami 100036985

Jasim Alhammadi 100035510

**Date Experiment Performed**: 25-3-2015

**Date Lab Report Submitted**: 8-4-2015

**Lab Instructor: Mr. Mohammed Alzaabi/ Mr. Mahmoud Khounji**

**Spring 2015**

# Summary

This report discusses the concept of using Analog to Digital (ATD) converters and how it can be implemented on HCS12 board using C programming. There are three main tasks. The first task is about implementing a program to read the changes in the potentiometer and displaying them using the LCD with different resolutions (8 bits and 10 bits). The second task required writing a program to read the temperature sensor output and display it as a voltage and as temperature in Degrees Celsius on the LCD. The third task required writing a program to read the light sensor output and alter the speed of the flash of the PORTB LEDs based on the light sensor output.

# 

Contents

[Summary 2](#_Toc416261049)

[1. Introduction 4](#_Toc416261050)

[1.1 Aim 4](#_Toc416261051)

[1.2 Objectives 4](#_Toc416261052)

[2. Design and Results 5](#_Toc416261053)

[Task 1: 10 – bit Resolution 5](#_Toc416261054)

[Task 2: temperature sensor 7](#_Toc416261055)

[Task 3: Light sensor 9](#_Toc416261056)

[3. Analysis and Interpretation 11](#_Toc416261057)

[4. Assignment questions 12](#_Toc416261058)

[5. Conclusions and Recommendations 15](#_Toc416261059)

# 1. Introduction

Real life signals, such as temperature, pressure (wind or liquid), humidity, and velocity are physical analog quantities. In order to allow the microcontroller to interact with these quantities, they are converted to electrical (voltage) signals using a sensors. Then, an analog-to-digital converter translates analog signals to digital signals for the microcontroller to read and analyze them.

Quantizing the analog signal into finite states then assigning each state to a digital word is a way of translating analog signal to digital signal. The resolution of the converter depends on the number of these states. ATDs have *n*-bit resolution; where *n* can be 8, 10, 12, 16, or even 24 bits.

The smaller the ATD step size the higher the resolution the better is the conversion result. Another factor that determines the conversion step size is the reference voltage supplied to the converter. Higher resolutions may lead to higher conversion time.

1.1 Aim

The aim of this report is to introduce the students to use of analog to digital (ATD).

1.2 Objectives

On completion of this experiment the student should be able to:

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

# 2. Design and Results

## **Task 1: 10 – bit Resolution**

We were asked to make a CodeWarrior project for the program shown in the introduction of the laboratory script, then execute the program in the Dragon12 Plus Trainer, change the potentiometer position and verify the LCD readings using a DMM. The following code does the required task:

The code is shown below:

**#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)/51); //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 \*/**

**}**

The code is modified by setting the prescaler to the first three pins of DIP switches in order to be able to set the prescaler values manually. The modification is set into a loop in order to make the prescaler dynamic and flexible to instant changes. The code is run on the board, and examination is made on the following pre-scalar values:

## **Task 2: temperature sensor**

We were asked to write a C language program to read the temperature sensor output and display it as a voltage on the first line of the LCD and as a temperature in Celsius on the second line of the LCD using 10 bit resolution. Then, icreasing the temperature of the sensor by pressing on it to check the changes on the LCD display. Note: the sensor’s resolution is 10 mV/°C. The code is shown below:

**#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 = 0x85; // 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) {**

**float val;**

**float out;**

**LCD\_Init();**

**ATD\_init();**

**for(;;) {**

**val=(ATD\_CONVERT())/204.6;**

**out=((val\*1000.0)/10); //output in Voltage = conversion result \*step size**

**//out =conversion result \*((VrefH-VrefL)/(2^resolution-1))**

**LCDWriteLine(1,"Voltage= ");**

**LCDWriteFloat(val);**

**LCDWriteLine(2,"Temperature= ");**

**LCDWriteFloat(out);**

**delay(100);**

**LCD\_clear\_disp();**

**\_FEED\_COP(); /\* feeds the dog \*/**

**} /\* loop forever \*/**

**/\* please make sure that you never leave main \*/**

**}**

Slight modifications were done on Task 1’s program to activate the use of the temperature sensor. Since channel 5 is connected to a built-in Precision Centigrade Temperature Sensor U14 (LM45) the channel has to be changed by changing the ATD control register 5 (ATDxCTL5) which is used to select the analog input channels by Bits (2-0) (CC, CB, and CA) so to use channel 5 these 3 bits are assigned the value 5. Obviously, whenever temperature sensor is pressed the temperature and the voltage will change.

## **Task 3: Light sensor**

In this task it is required to write a C language program to read the light sensor output and alter the speed of the flash of the PORTB LEDs based on the proximity of the human hand to the sensor. The code of this task is shown below.

**#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,**

**}**

**void delay2(unsigned long int ms)**

**{ unsigned long int i,j;**

**for(i=0;i<(ms\*100);i++)for(j=0;j<14;j++) asm("nop\n");**

**}**

**int ATD\_CONVERT()**

**{ATD0CTL5 = 0x84; // 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) {**

**float val;**

**float out;**

**ATD\_init();**

**DDRB = 0xFF; //PORTB as output since LEDs are connected to it**

**DDRJ = 0xFF; //PTJ as output to control Dragon12+ LEDs**

**DDRP=0xFF;**

**PTP=0x0F; //Disable 7-seg display**

**PTJ=0x00; //Allow the LEDs to display data on PORTB pins**

**for(;;) {**

**val=(ATD\_CONVERT());**

**//val=(ATD\_CONVERT())/204.6;**

**PORTB = 0xFF;**

**delay2(val);**

**PORTB = 0x00;**

**delay2(val);**

**\_FEED\_COP(); /\* feeds the dog \*/**

**} /\* loop forever \*/**

**/\* please make sure that you never leave main \*/**

**}**

Channel 4 is connected to a light sensor so the ATD control register 5 (ATDxCTL5) Bits (2-0) (CC, CB, and CA) are assigned the value 4. Whenever the human hand approaches the light sensor, it absorbs less light and the speed of the flashing of the LEDs increases. Hence they are inversely proportional.

# 3. Analysis and Interpretation

In this lab learned the idea of analog to digital ATD converter was learned. In task 1, the ATD converter was used to observe the changes in the potentiometer using 8 and 10 bits resolutions. In task 2, the ATD converter was used to read the temperature sensor results and show them using the LCD screen. In task 3, the ATD converter was used to read the light sensor output. Overall ATD have a very important role since they allow the microcontroller to read and process the physical analog quantities through converting them into digital signals.

# 4. Assignment questions

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

**#include <hidef.h> /\* common defines and macros \*/**

**#include "derivative.h" /\* derivative-specific definitions \*/**

**#include "lcd.h" /\* include lcd.h to use functions for LCD \*/**

**void ATD\_init(void)**

**{ ATD0CTL2\_ADPU = 1;**

**delay(1);**

**ATD0CTL4 = 0x05;**

**}**

**int ATD\_CONVERT()**

**{ATD0CTL5 = 0x87;**

**while(!(ATD0STAT0 & 0x80));**

**return(ATD0DR0); }**

**void main(void) {**

**float out1;**

**float val;**

**DDRB = 0xFF;**

**DDRJ = 0xFF;**

**DDRP=0xFF;**

**PTP=0x0F;**

**PTJ=0x0;**

**PORTB = 0x00;**

**ATD\_init();**

**val=ATD\_CONVERT();**

**out1=((val\*1.0\*5.0)/1023);**

**for(;;) {**

**PORTB= PORTB ^ 0xFF;**

**delay (ATD\_CONVERT()\*1000);**

**\_FEED\_COP(); /\* feeds the dog \*/**

**} /\* loop forever \*/**

**}**

**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.**

**#include <hidef.h> /\* common defines and macros \*/**

**#include "derivative.h" /\* derivative-specific definitions \*/**

**#include "lcd.h" /\* include lcd.h to use functions for LCD \*/**

**void ATD\_init(void)**

**{ ATD0CTL2\_ADPU = 1;**

**delay(1);**

**ATD0CTL4 = 0x05; }**

**int ATD\_CONVERT()**

**{ATD0CTL5 = 0x85;**

**while(!(ATD0STAT0 & 0x80));**

**return(ATD0DR0); }**

**int val;**

**float out1;**

**float temp;**

**void main(void){**

**//PORTH interrupt setup**

**DDRH = 0x00; //PTH as input**

**PIEH = 0xFF; //enable PTH interrupt**

**PPSH = 0x0; //Make it Falling Edge-Trig.**

**\_\_asm CLI;**

**LCD\_Init();**

**ATD\_init();**

**for(;;)**

**{val=ATD\_CONVERT();**

**out1=((val\*1.0\*5.0)/1023);**

**temp=((out1/10)\*1000);**

**\_FEED\_COP(); /\* feeds the dog \*/**

**} /\* loop forever \*/**

**/\* please make sure that you never leave main \*/}**

**#pragma CODE\_SEG NON\_BANKED // to access the interrupt vector table**

**interrupt 25 void PORTH\_ISR(void) // interrupt (((0x10000-Vporth)/2)-1) void PORTH\_ISR(void)**

**{ if (PIFH\_PIFH1==1) // if SW4 is pressed{**

**// display the value of out**

**// display “temp= “ on LCD line 2**

**delay(1000);**

**LCDWriteLine(1,"voltage= "); // display “Voltage= “ on LCD line 1**

**LCDWriteFloat(out1); // display the value of out1**

**LCDWriteLine(2,"Kelvin = "); // display “Kelvin= “ on LCD line 2**

**LCDWriteFloat((temp+273)); // display the value of temperature in Kelvin**

**PIFH=PIFH\_PIFH1\_MASK; //clear PTH Interrupt Flag introduced by SW4}**

**if (PIFH\_PIFH2==1) // if SW3 is pressed**

**{**

**delay(1000);**

**LCDWriteLine(1,"voltage= "); // display “Voltage= “ on LCD line 1**

**LCDWriteFloat(out1); // display the value of out1**

**LCDWriteLine(2,"Fehrenheit = "); // display “Fahrenheit = “ on LCD line 2**

**LCDWriteFloat(((temp\*9/5)+32)); // display the value of temperature in Fahrenheit**

**PIFH=PIFH\_PIFH1\_MASK; //clear PTH Interrupt Flag introduced by SW3}}**

# 5. Conclusions and Recommendations

To conclude, the concept of ATD converter in an embedded system was understood. In addition to that, the students were able to writing programs and to use the 16 channels A/D converter of the Dragon12 Plus Trainer Board. Furthermore, the students were able to compile, download and test a C program using CodeWarrior C compiler and to use Dragon12 Plus Trainer board onboard peripherals related to ATD.