

**Table of Contents**

***Table of Contents...…………………………………….………………………………….…2***

***List of Figures and Tables ...…………………………………….…………………………3***

***Summary ...…………………………………….………………………………….…………..4***

***1. Introduction ...…………………………………….………………………………….…….5***

**1.1** Aim ***……………………………………….…………………….………………….6***

**1.2** Objectives ***…………………………………………………………………………..6***

***2. Design and Results …………..……………………………………….…………...…….7***

**2.1**Task 1:10- Bit Resolution ***……………………….…………………….………………7***

**2.2**Task 2: Temperature sensor ***……………………….…………………….…………….8***

**2.3**Task 3: Light sensor ***……………………….…………………….…………………..9***

***3. Assignment Questions ……………………………………….……………….………11***

***4. Conclusions and Recommendations ……………………………………….……….13***

**List of Figures and Tables**



Figure 1: 4-channel stereo multiplexed analog-to-digital converter 

**Summary**

This lab report  discuss the aim and the objective of this laboratory experiment which is about  the usage of analog to digital (ATD)

There are three main tasks in this lab that should be achieved. First, we should do a 10- Bit Resolution. Second, we should do a temperature sensor. Finally,  task is about Light sensor. To achieve this, we should have read the introduction about this laboratory experiment to know how to use the sensors in the board.

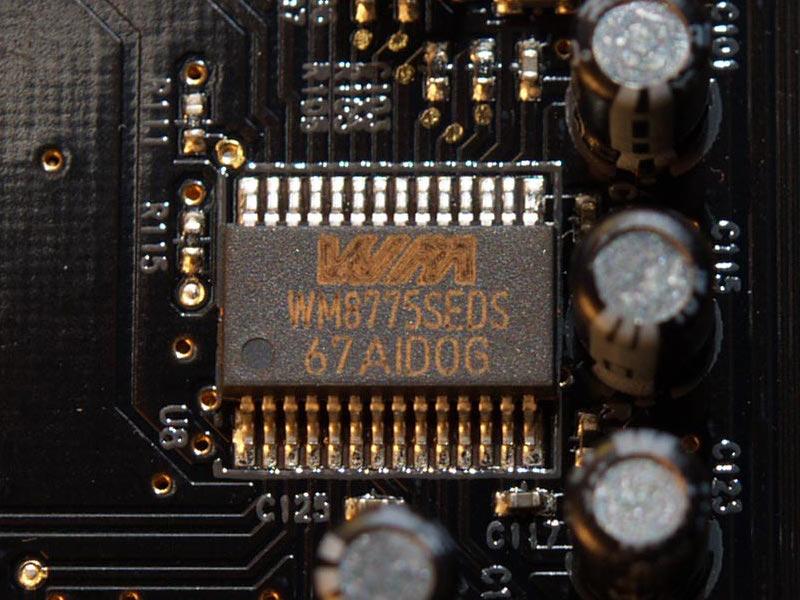
**1. Introduction**

Analog-to-digital conversion is considered as an electronic process in which a continuously variable [analog](http://searchcio-midmarket.techtarget.com/definition/analog) signal is changed, without altering its essential content, into a multi-level [digital](http://searchcio-midmarket.techtarget.com/definition/digital) signal.

The input to an *analog-to-digital converter (ADC)* consists of a [voltage](http://searchcio-midmarket.techtarget.com/definition/voltage) that varies among a theoretically infinite number of values. Examples are sine waves, the waveforms representing human speech, and the signals from a conventional television camera. The output of the ADC, in contrast, has defined levels or states. The number of states is almost always a power of two -- that is, 2, 4, 8, 16, etc. The simplest digital signals have only two states, and are called [binary](http://searchcio-midmarket.techtarget.com/definition/binary). All whole numbers can be represented in binary form as strings of ones and zeros.

Digital signals propagate more efficiently than analog signals, largely because digital impulses, which are well-defined and orderly, are easier for electronic circuits to distinguish from noise, which is chaotic. This is the chief advantage of digital modes in communications. Computers "talk" and "think" in terms of binary digital data; while a microprocessor can analyze analog data, it must be converted into digital form for the computer to make sense of it.

A typical telephone [modem](http://searchmobilecomputing.techtarget.com/definition/modem) makes use of an ADC to convert the incoming audio from a twisted-pair line into signals the computer can understand. In a digital signal processing system, an ADC is required if the signal input is analog.



***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***

1. The first requirement in this task is to make a CodeWarrior project for the program shown in the introduction in the laboratory script, execute the program in the Dragon12 Plus Trainer and finally change the potentiometer position and verify the LCD readings using a DMM.

The code of this task is shown as follows:

|  |
| --- |
| #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 = 0x85; // 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 \*/  } |

1. In the second sub-task, it is required to modify the previous program to set the ATD resolution to 10 bit and display the result on the LCD.

The code of this sub-task is shown below.

|  |
| --- |
| #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; // power up ATD channel 0, disable interrupts  delay(1); // wait for ADC to warm up  ATD0CTL4 = 0x85; // 8-bit, sample time 2 ADC clock, prescale of 5  }  int ATD\_CONVERT()  {ATD0CTL5 = 0x87; // channel no. 7 and right justified (the potentiometer channel)  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(); //initiate the LCD controller  ATD\_init(); //initiate the ATDcontroller  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= "); // display “Value= “ on LCD line 1  LCDWriteFloat(val); // display the value of val  LCDWriteLine(2,"Voltage= "); // display “Voltage= “ on LCD line 2  LCDWriteFloat(out); // display the value of out  delay(100);  LCD\_clear\_disp(); // clear the LCD screen  \_FEED\_COP(); /\* feeds the dog \*/  } /\* loop forever \*/  /\* please make sure that you never leave main \*/  } |

* Comment: The difference between part (a) and (b) is that at the maximum value of the potentiometer, the digital value (converted value) in part (a) is 256 while in part (b) is 1023 which is due to the following equation: 2x where x is the resolution.

***Task 2: Temperature sensor***

In this task it is required 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. Use 10 bit resolution. Then, try increase 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 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;         // 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 out1;  float temp;  LCD\_Init();  ATD\_init();  for(;;) {  val=ATD\_CONVERT();  out1=((val\*1.0\*5.0)/1023); //output in Voltage = conversion result \*step size  //out =conversion result \*((VrefH-VrefL)/(2^resolution-1))  temp=((out1/10)\*1000); // 10 mV/°C  LCDWriteLine(1,"voltage= ");  LCDWriteFloat(out1);  LCDWriteLine(2,"teampC= ");  LCDWriteFloat(temp);  delay(100);  LCD\_clear\_disp();  \_FEED\_COP(); /\* feeds the dog \*/  } /\* loop forever \*/  /\* please make sure that you never leave main \*/  } |

* Comment: It is clear that whenever we press on the temperature sensor, the temperature changes as well as the voltage.

***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 your hand to the sensor.

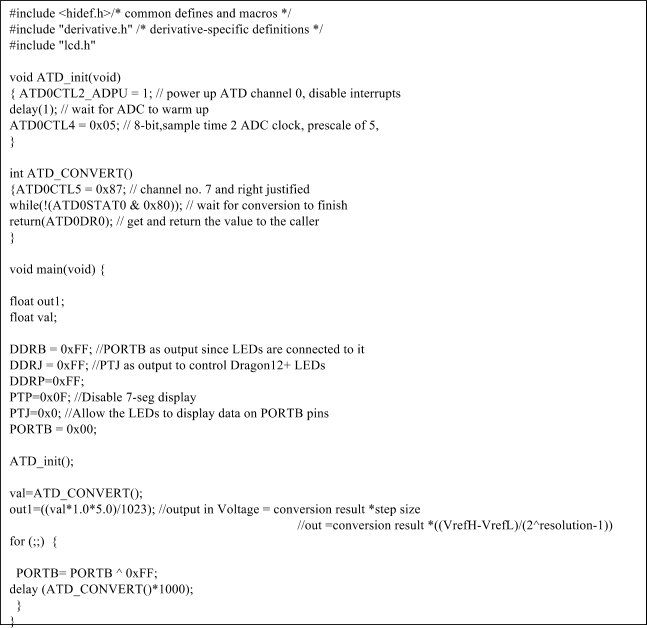
The code of this task is shown below.

|  |
| --- |
| #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; // 10-bit,sample time 2 ADC clock, prescale of 5  }  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) {     int val;     float out;     LCD\_Init();     ATD\_init();  DDRB = 0xFF; //PORTB as output since LEDs are connected to it  DDRJ = 0xFF; //PTJ as output to control Dragon12+ LEDs  DDRP=0xFF;  PORTB = 0x00;       for(;;) {     val=ATD\_CONVERT();     out=((val\*1.0 \* 5 )/1023); //output in Voltage = conversion result \*step size     LCDWriteLine(1,"Value= ");     LCDWriteFloat(val);     LCDWriteLine(2,"light= ");     LCDWriteFloat(out \* 100);     delay(100);     LCD\_clear\_disp();    PORTB=PORTB ^ 0xFF;      delay(out \* 1000 ) ;  PORTB=PORTB ^ 0xFF;  delay(out \* 1000 ) ;       \_FEED\_COP();     }  } |

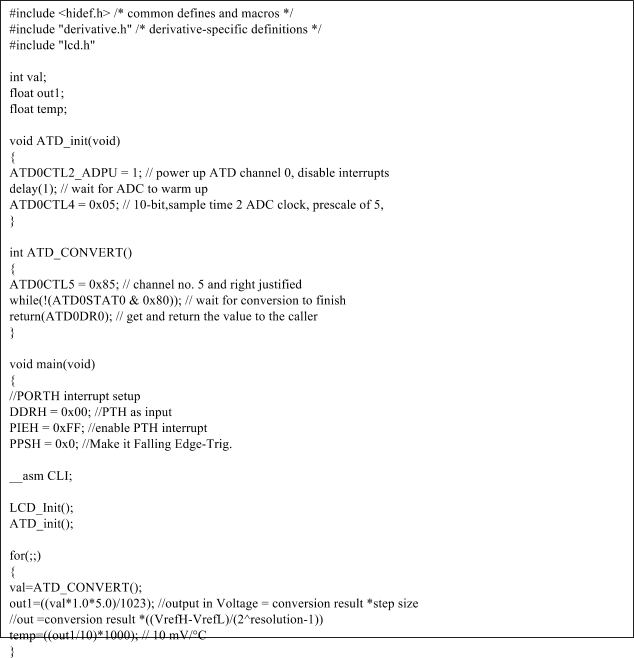
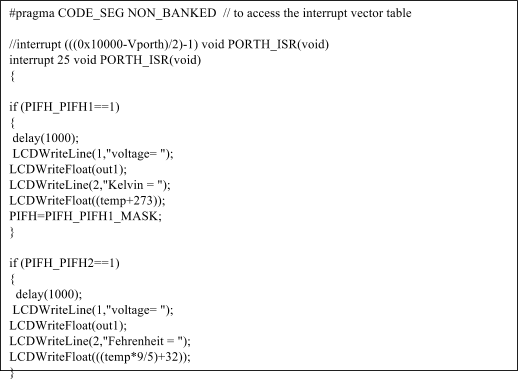
* Comment:  whenever the light that is absorbed by the light sensor decreases, the speed of the flashing of the LEDs increases. In other words, there is an inversely proportional relationship between the flashing speed and the light intensity.

**3. Assignment Questions**

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



**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.**

**4. Conclusions and Recommendations**

After performing the required tasks of this experiment properly and that can be interpreted by the results that we got, we can say confidently that the theoretical knowledge that we own from the class courses matches the implementation results that we achieved in the laboratory. The following achievements have been gained:

* Realize the concept of analog to digital (ATD) converters.
* How to use the 16 channels A/D converter of the DRAGON12.
* How to use onboard peripherals that are related to ATD.
* How to use the CodeWarrior IDE for the development of HCS12 microcontroller C programs.
* Download, run, and test code on a Dragoon12+Board.

Recommendations that should be taken into account in order to improve the procedures of the laboratory experiments are mutual between students and laboratory instructors. Students have to acquire the proper knowledge of theoretical concepts, build a proper understanding of what is going to be achieved in the laboratory by performing the pre-laboratory section which gives a brief introduction of procedures that should be followed in order to achieve the objectives of the experiment. While laboratory instructors play an essential role in introducing techniques that could be useful for students to do their work professionally and assist students when they face difficulties in implementing the laboratory task properly.