Khalifa University of Science, Technology and Research
Electronic Engineering Department

ELCE333Microprocessor Systems laboratory

Laboratory Experiment 4

HCS12 INPUT AND OUTPUT PORTS

Lab Partners

Leena ElNeel	100037083
Muna Darweesh	100035522
Shamsah AlNabooda	100036984

Date Experiment Performed: 18/2/2015
Date Lab Report Submitted: 25/2/2015

Lab Instructors:

Mahmoud Khonji

Mohammed Ali Saif Al Zaabi

Spring 2015

Table of Contents

List of Figures and Tables

Summary

This report sums up the main aim of this laboratory experiment which to learn how to read and write data from input and output ports and using loop to implement delays

This experiment focuses on introducing the basics of C language and continues on teaching us the HCS12 ports utilization. This lab session includes many tasks where C language will be used to write the programs through all the tasks. A brief introduction and theoretical explanation will be presented in the first part of the report. The second part includes the tasks, codes and simulation results. And finally all the results is analyzed in the conclusion.

1. Introduction

C is one of the most widely used programming languages of all time, and there are very few computer architectures for which a C compiler does not exist.

The Keypad of the HCS12

The HCS12 board has a 4 x 4 keypad which is connected to PORT A. The rows and columns connection of the Key Pad is shown in figure 1. PORT A is structured as bidirectional register so that PA0-PA3 are output columns. PA4-PA7 are input rows, so, DDRA is equal 0x0F. The pull up resistors for the pins associated with the core of PORT A should be on by setting PUCR to 0x01.

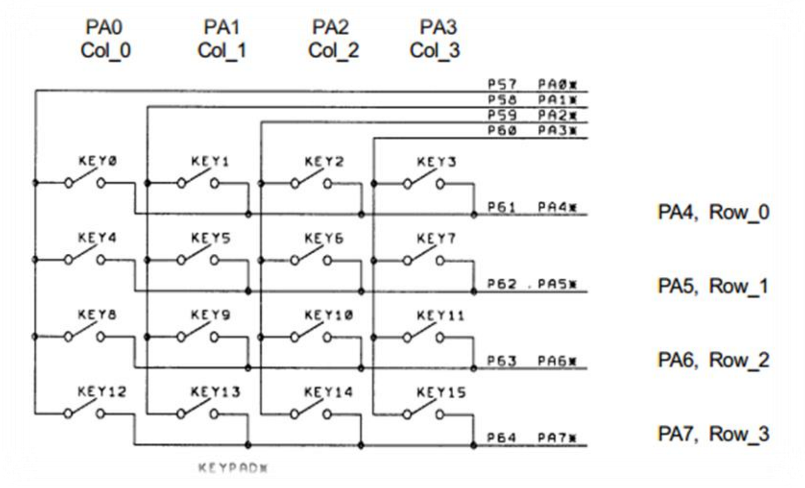


Figure 1: Port A connection with the keypad

The function below checks which key is pressed; it keeps looping until a key is pressed and return the key value to the main function.

```
int key_pad(void)
{
int X;
while(1)
{
PORTA = 0XFE;
X = PORTA;
if (X == 0xEE)return 0x01;
if (X == 0xDE)return 0x04;
if (X == 0xBE)return 0x07;
if (X == 0x7E)return 0x0E;
PORTA = 0XFD;
X = PORTA;
if (X == 0xED)return 0x02;
if (X == 0xDD)return 0x05;
if (X == 0xBD)return 0x08;
if (X == 0x7D)return 0x00;
PORTA = 0XFB;
X = PORTA;
if (X == 0xEB)return 0x03;
if (X == 0xDB)return 0x06;
if (X == 0xBB)return 0x09;
if (X == 0x7B)return 0x0F;
PORTA = 0XF7;
X = PORTA;
if (X == 0xE7)return 0x0A;
if (X == 0xD7)return 0x0B;
if (X == 0xB7)return 0x0C;
if (X == 0x77)return 0x0D;
}
```

Some pre-written C library functions to initialize and access the LCD display. The LCD library functions are shown in table 1 and are written in a header file (lcd.h) that has to be included in the main program using: #include "lcd.h"

Table 1: LCD library functions

Function	Passing Parameters Example	Description
void LCD_Init(void)	-	Initialize the LCD Controller
void LCDWriteChar(byte d)	Char: 'A'	Write 2 nibbles character to LCD
void LCDWriteLine(byte line, char* d);	Line: 1 or 2 String: “Hi”	Print ASCII string on the 1st or 2nd line of LCD display.
void LCDWriteInt(Int d);	Int: 1	Print integer on the 1st or 2nd line of LCD display.
void LCDWriteFloat(float d);	Float: 1.1000	Print float on the 1st or 2nd line of LCD display.
void LCD_clear_line(int line);	Line: 1 or 2	Clear either line 1 or 2
void LCD_clear_disp(void);	-	Clear both lines 1 and 2
void delay(byte ms)	Delay time	delay in msec

1.1 Aim

To introduce the students to the read and write data from the input and output ports and how delays can be implemented using loops.

1.2 Objectives

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

1- Learn how a C program can access I/O registers.

2- Develop simple programs for an embedded system.

3- Use the CodeWarrior Integrated Development Environment for the development of HCS12 microcontroller C programs.

4- Writing simple C language programs for interfacing DIP switches and Key Pad with LCD.

5- To assemble, download and run a C program using CodeWarrior C compiler and Dragon12 plus Trainer board

2. Design and Results

Task 1: Reading DIP Switches and Writing them to LEDs

In the first task, we are asked to write a program code in C language in order to take the inputs from the DIP switches and output in the LEDs. The code works by loading the data, adding accumulators and transferring the data into the LEDs

```
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific
definitions */

void main() {
          EnableInterrupts;

  DDRH =0x00;  // set port H as input
  DDRB= 0xFF; // set port B as an output
  DDRJ= 0xFF;
  DDRP=0x0F;
  PTP=0x0F;
  PTJ= 0;
          PTH =0;
  PORTB =0xFF; // for hex value

  for(;;)
   {
    PORTB= PTH;
  }// end of loop
} // end of main
```

As port H is controlling the DIP switches we assigned to be an input at the port H and for the output port B. To disable the 7-segmnet display ports J and P were assigned to be output (for port P 4-LSB assigned for output, whereas 4-MSB were assigned for input) and they were set to 0x00 and 0x0F respectively). After that, in the loop, the output were assigned to the input to display the values on DIP switches on LEDs.

The figures below shows the result obtained when the input was 1101 0110:

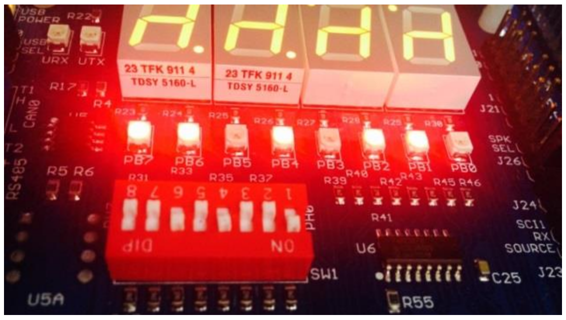
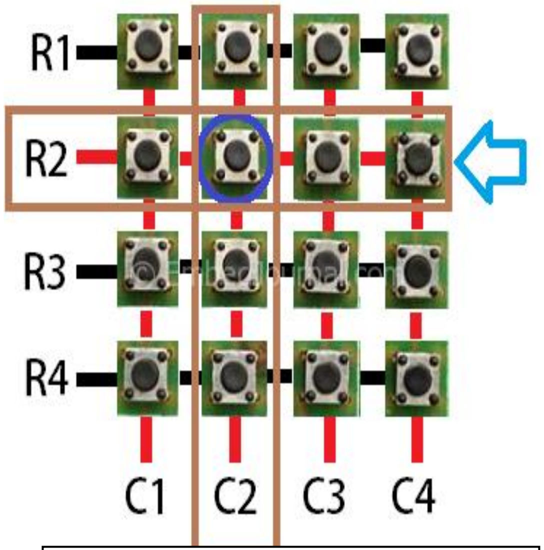


Figure 2: Result of 1101 0110 input

Task 2: Reading keypad and writing it to LEDs

In the second task of this lab the aim was to make LED's display the equivalent binary number of the button that is being pressed in the keypad that have a range from [0,F] plus '*' and '#'.

Matrix keypad is used to reduce pin count in which there are many digital inputs to be covered by the available pins in the microcontroller. Therefore, the number of pins that are required to interface a given number of inputs decreases with increase in the order of the matrix. Inside the matrix, columns and rows are scanned to identify the pressed button and retrieve its value.



In order to display the digital value that is been retrieved from the keypad on the LEDs, ports B and J firsts was assigned to be an output whereas port P was set to 0x0F to disable the 7 segment display. Then an integer return function *key_pad()* (that includes many if cases) was used to match the retrieved logical values from the keypad to its numerical value. The last step was assigning the output to the input *PORTB =key_pad()*.

```
#include <hidef.h>
#include "derivative.h
int key_pad();
void main() {
             EnableInterrupts;
  DDRB= 0xFF; // set port B as an output
  DDRJ= 0xFF;
  DDRA= 0x0F;
  DDRP=0x0F;
  PTP=0x0F;
  PUCR= 0x01;
  PORTA = 0x00;
  PTJ= 0x00;
 for(;;) {    PORTB =key_pad();
              _FEED_COP();}
}// end of main
```

```
int key_pad(){
int X;
while(1) {
    PORTA = 0XFE;
    X = PORTA;
    if (X == 0xEE)return 0x01;
    if (X == 0xDE)return 0x04;
    if (X == 0xBE)return 0x07;
    if (X == 0x7E)return 0x0E;
    PORTA = 0XFD;
    X = PORTA;
    if (X == 0xED)return 0x02;
    if (X == 0xDD)return 0x05;
    if (X == 0xBD)return 0x08;
    if (X == 0x7D)return 0x00;
    PORTA = 0XFB;
    X = PORTA;
    if (X == 0xEB)return 0x03;
    if (X == 0xDB)return 0x06;
    if (X == 0xBB)return 0x09;
    if (X == 0x7B)return 0x0F;
    PORTA = 0XF7;
    X = PORTA;
    if (X == 0xE7)return 0x0A;
    if (X == 0xD7)return 0x0B;
    if (X == 0xB7)return 0x0C;
    if (X == 0x77)return 0x0D; }
  return X;  }
```

The result of this task is shown in the figure below:

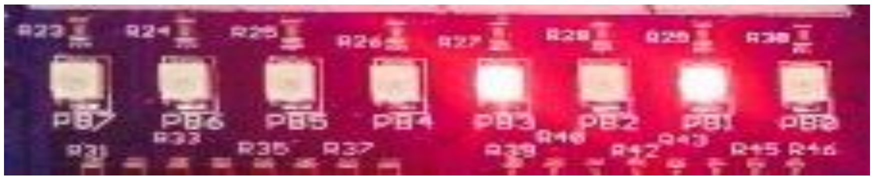


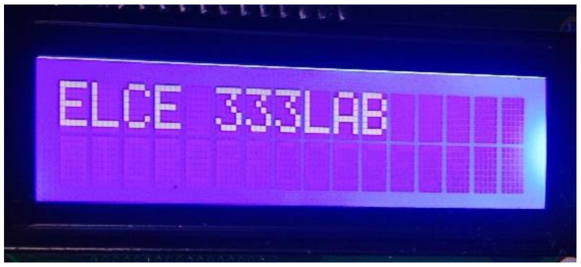
Figure 3: Reading hexadecimal 'A' form keypad and writing it to LED

Task 3: Writing to LCD

In this task it is required to write a C language program using the LCD function calls to print the following text message "ELCE 333 Lab" on the first line of Dragon 12 LCD and "Microprocessor" on the second line. This is done using the command "LCDWriteLine". Before displaying the second line, it is required to erase the first message using the command "LCD_clear_line" .The two messages should be displayed for 0.5 second and cleared for another 0.5 second and so on as a flashing text. This is done using delay command which is "delay (500)". To display and remove the text message we need to use the delay command twice, between that the command that clear the line which is " LCD_clear_disp()" is used.
The flashing should be done alternatively so that when the first line appears, the second disappear and vice versa. The code for this task is shown below.

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"
int main() {
            EnableInterrupts;
      LCD_Init();

  for(;;) {

   LCDWriteLine( 1, "ELCE 333LAB");
    delay(500) ;
    LCD_clear_disp();
     delay(500) ;
   LCDWriteLine( 2, "MICROPROCESSOR");
   delay(500)   ;
   LCD_clear_disp();

     delay(500) ;

    _FEED_COP(); /* feeds the dog */
  } /* loop forever */
 return 0;}
```

The results are shown as the following. The first one shows the text message "ELCE 333 Lab" and the second one shows the text message "Microprocessor". Clearly, by display the second text message, the first text message will disappear, so the line is cleared, and the second one will be at the second line.



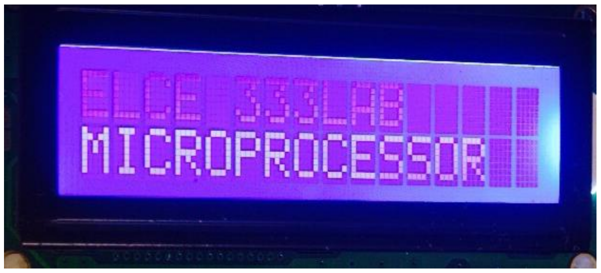


Figure 4: Text message "ELCE 333 Lab"

Figure 5: Text message "Microprocessor".

Task 4:Reading DIP Switches and Writing them to LED’s and LCD

In this task we were asked to modify program written in task one so that the DIP switches input would be displayed in both the LEDs and the LCDs. Similar to what’ve been done before output was assigned to input (PORTB = PTH). The new addition here is that value of PORTB was rewritten as input into the LCD using LCD function “LCDWriteInt(PORTB)”.The input result from the switches was 1110 0000 which is equivalent to 224 as seen is displayed on the LED and LCD.

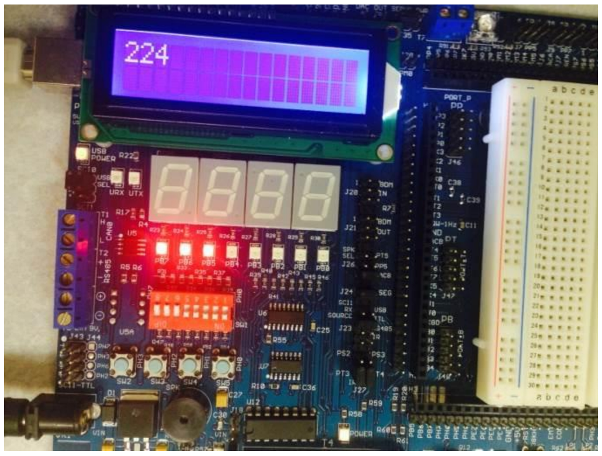


Figure 6: Task 4 LCD and LED results.

```
#include <hidef.h>      /* common defines
and macros */
#include "derivative.h"      /* derivative-
specific definitions */
 #include "lcd.h"

void main() {
            EnableInterrupts;

  DDRH =0x00;  // set port H as input
  DDRB= 0xFF; // set port B as an output
  DDRJ= 0xFF;
  DDRP=0x0F;
  PTP=0x0F;
 PTH =0;
  PORTB =0xFF; // for hex value
 LCD_Init();
  for(;;)
   {
    PORTB= PTH;
    LCDWriteInt(PORTB);
    LCDWriteLine(1,"");
    delay(100);
  }// end of loop
} // end of main
```

Task 5: Interfacing LCD with key pad and push buttons.

The last task of this lab contains a portion of each of the previous tasks in which it sum up the practical knowledge learned in this lab. This task aimed to display an input read from the keypad in the second line of the LCD in case SW5 wasn't pressed. In case SW5 was pressed the displayed number should be the number read from the keypad divided by two.

In order to do that, we had first to do the steps for assigning outputs to LCD similar to previous tasks. Then, we defined two float variables one (*Y*) is being assigned to the number that is been retrieved from the keypad and the other (*Z)* for dividing it by 2. After that, IF ELSE condition testing were used whether there is any switch in port B that is on and in case this statement is evaluated to true then the function LCDWriteFloat(y) is used to display the number of LCD. On the other hand, if this statement is evaluated false then the value *Z* is displayed instead of *Y*.

The code of this task is as the following.

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"

int key_pad();
void main()
 {
        float y;
        float z;
   DDRA= 0x0F;
   PUCR= 0x01;
        EnableInterrupts;
        LCD_Init();
   for(;;)
   {
     y= key_pad();
     delay (50);
     LCD_clear_disp();
      if (PTH_PTH0 ==1)
       {
        LCDWriteLine( 1, "num= ") ;
        LCDWriteFloat(y);
       }
      if (PTH_PTH0 ==0)
      {
        LCDWriteLine( 1, "num= ") ;
        z=y/2;
        LCDWriteFloat(z); // write the numebr as it is
      }
   }// end of loop
} // end of main
```

```
int key_pad()
{
 int X;
 while(1) // loop forever
   {
     PORTA = 0XFE;
     X = PORTA;
     if (X == 0xEE)return 0x01;
     if (X == 0xDE)return 0x04;
     if (X == 0xBE)return 0x07;
     if (X == 0x7E)return 0x0E;
     PORTA = 0XFD;
     X = PORTA;
     if (X == 0xED)return 0x02;
     if (X == 0xDD)return 0x05;
     if (X == 0xBD)return 0x08;
     if (X == 0x7D)return 0x00;
     PORTA = 0XFB;
     X = PORTA;
     if (X == 0xEB)return 0x03;
     if (X == 0xDB)return 0x06;
     if (X == 0xBB)return 0x09;
     if (X == 0x7B)return 0x0F;
     PORTA = 0XF7;
     X = PORTA;
     if (X == 0xE7)return 0x0A;
     if (X == 0xD7)return 0x0B;
     if (X == 0xB7)return 0x0C;
     if (X == 0x77)return 0x0D;
   } // end of loop
   return X;
```

Assignment Questions

1)Create a C language program that checks the state of PH2 (while all DIP switches at their high position) and flashes all the eight LEDs of Port B from left to right if PH2 is high. And flashes them from right to left (like they do when the board is first powered on) if PH2 was low. Hint: Use the keyword >> to shift right or << to shift left.

In order to do that we first determined the inputs and outputs. Since we were using LEDs, we had to assigned port H as input and ports B, J and P as output in which they control the switching LEDs on without the 7 segments display.

After that, we used IF ELSE condition testing to determine the output in each of the two cases. In case all switches on port H were in a high position (up) and PH2 (pressed button) was high (i.e. 1111 1111), then all LEDs will flash from left to right with a delay of 0.1 seconds. Once all LEDs are flashed and lsb of LEDs is on, PORTB will be reassigned to the msb ($2^7 = 128$)of the LEDs so we can have a sequence of continuous flashing LEDs in that order. For the other case, same steps will be repeated except that this time the flashing sequence will be from right to left case PH2 is low; hence this condition is true when its equal (1111 1011). Also, when LEDs flashing sequence reaches the msb (the last bit on LEDs) it reassign PORTB to 1 so starts the flashing process from the lsb in same order.

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"
void main(void)
{
            EnableInterrupts;
   DDRH = 0x00 ;      //Set PORTH as input
   DDRB = 0xFF;       //set PORTB as Output
   DDRJ = 0xFF ;      //set PTJ as Output
   DDRP = 0xFF ;      //set PTP as Output
   PTP = 0x0F  ;
   PORTB = 1;
   for(;;)
   {
     if ( PTH == 0xFF )
      {
        PORTB = PORTB>>1 ;
        delay (50);
        if (PORTB == 0)
         {
           PORTB =  128;
           delay(50);
         }
       }
        if ( PTH == 0xFB )
         {

           PORTB =PORTB<<1 ;
           delay (50);
           if (PORTB == 128)
           {
             PORTB =  1;
             delay(50);
           }
       } // end of if
 }//end of loop
} //end of main
```

2)Write a C language program to perform (addition, subtraction, multiplication, or division) of two numbers entered by the user and display the result on the LCD. The first number is read from the four most significant bits of PTH (higher 4 DIP Switches). The second number is entered by the user using the key Pad. The mathematical operation is to be determined using the four least significant bits of PTH (lower 4 DIP switches) as shown in the table below.

PH3	PH2	PH1	PH0
Division	Multiplication	Subtraction	Addition

The first step in writing such code is to define variables for inputs (num1 and num2) and for the variables that will hold the result of the operation to be performed. Also, PORT A has to be configured as a bidirectional register so that PA0-PA3 are output columns and PA4-PA7 are input rows, as a result DDRA should equal 0x0F. Then, we assigned num1 to be taken from the msb bits in PTH and num2 from the value returned from the keypad reading function. After that, by using IF condition to check which bit is active within the 4-lsb of PTH (BIT0= addition, BIT1= subtraction, BIT2= multiplication, BIT3=division) and perform the corresponding operation then display the result on LCD.

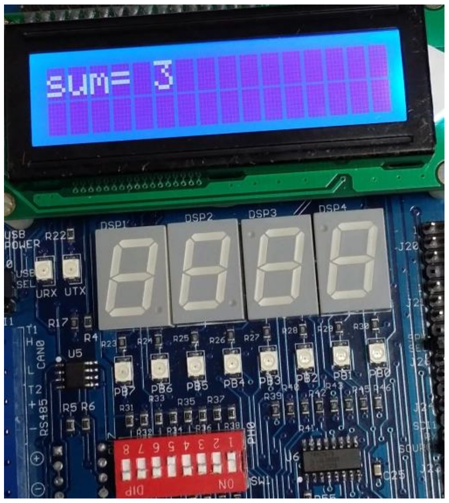


Figure 7: Case: num1=1, num2=2, op=sub

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"

void main()

{
  int num1,num2;
  int sum,sub, mul,div;

              EnableInterrupts;
  DDRH = 0x00 ;      //Set PTH as input
  DDRA = 0x0F;       // for keypad usage 4-lsb output and
4-msb input
  PUCR = 0x01;       //set th pull up control register
  LCD_Init ();      // initilizing LCD

  for(;;)
   {
    delay (100);
    num1 = ((PTH & 0xF0 )>> 4);
    delay (100);
    num2 = (key_pad());
    delay (100);
    if (PTH & 0x01)
      {
       LCD_clear_disp();
       LCDWriteLine (1, "sum= ");
       sum= (num1 + num2) ;
       LCDWriteInt (sum);
       LCD_clear_line( 2);
      }

    if (PTH & 0x02)
      {
       LCD_clear_disp();
       LCDWriteLine (1, "sub= ");
       sub= (num1 - num2) ;
       LCDWriteInt (sub);
       LCD_clear_line( 2);
       }

    if (PTH & 0x04)
      {
       LCD_clear_disp();
       LCDWriteLine (1, "mul= ");
       mul= (num1 * num2) ;
       LCDWriteInt (mul);
       LCD_clear_line( 2);
       }
```

```
    if (PTH & 0x08)
      {
       LCD_clear_disp();
       LCDWriteLine (1, "div= ");
       div= (num1 / num2) ;
       LCDWriteInt (div);
       LCD_clear_line( 2);
      }

 } //end of loop
}  // end of main
int key_pad(void)
{
  int X;
  while(1)
  {

    PORTA = 0XFE;
    X = PORTA;
    if (X == 0xEE)return 0x01;
    if (X == 0xDE)return 0x04;
    if (X == 0xBE)return 0x07;
    if (X == 0x7E)return 0x0E;

    PORTA = 0XFD;
    X = PORTA;
    if (X == 0xED)return 0x02;
    if (X == 0xDD)return 0x05;
    if (X == 0xBD)return 0x08;
    if (X == 0x7D)return 0x00;

    PORTA = 0XFB;
    X   = PORTA;
    if (X == 0xEB)return 0x03;
    {
    if (X == 0xDB)return 0x06;
    if (X == 0xBB)return 0x09;
    if (X == 0x7B)return 0x0F;

    PORTA = 0XF7;
    X = PORTA;
    if (X == 0xE7)return 0x0A;
    if (X == 0xD7)return 0x0B;
    if (X == 0xB7)return 0x0C;
    if (X == 0x77)return 0x0D;
    return X;
    } // end of if
   }  // end of loop
 } // end of key_pad()
```

Conclusion

This experiment introduced the structure of C language programs and their format, and the HCS12 ports and there usage. The first task, it was about writing lab 3 program that takes the inputs from the DIP switches and outputs their status in the LEDs by C language. That is done by assigning port H, as and input and port B as and output. In the second task we were asked to write a program that takes the inputs from the given KeyPad and outputs their binary equivalent status in the LEDs. For the third task we have to write simple message in two lines to be displayed on the LCD and clear them after certain time alternatively. This can be done with special commands such as “LCDWriteLine” to write to the LCD display. Before displaying the second line, it is required to erase the first message using the command “LCD_clear_line” and introducing the delay with the delay command.

Task four, the input was inserted using the DIP switches and the output was displayed on the LEDs and the LCD. Finally, in task five we are required to write a program that prompts the user to enter a number which should be displayed as is on the LCD display by using Key Pad and push buttons. However, the difference in this task is that the inputs are taken from two different ports (PORTA and PTH).