KHALIFA UNIVERSITY

Khalifa University of Science, Technology and Research
Electronic Engineering Department

ELCE333Microprocessor Systems laboratory

Laboratory Experiment 4

SERIAL COMMUNICATION INTERFACE

Lab Partners
Dana Bazazeh, 100038654
Moza Al Ali 100038604

Date Experiment Performed: 18/02/2015
Date Lab Report Submitted: 25/02/2015

Lab Instructor: Dr.Mohamed Al Zaabi/ Dr.Mahmoud Khonji

SPRING 2015

Table of Contents

List of Figures and Tables

Summary

In this lab report students include the primary purpose of this experiment, which is mainly an introduction about the programming and usage of the asynchronous serial communication interface in HCS12 microcontroller. The experiment has five tasks. In the first task students are asked to transmit switches state to LEDs while in the second task they were asked to transmit the key value pressed on the keypad to LEDs. The third task is about writing to the LCD and creating effects while the fourth task includes combining the functions of the DIP switches, the LEDs and the LCD. Finally, the last task considered Dip Switches and keypadsimple calculator. Later in this report there will be a detailed explanation on each task with provided codes and analysis of simulated results.

1. Introduction

Mainly C program is used in this session. The C programming language was devised in the early 1970s as a system implementation language for the nascent Unix operating system. Derived from the type less language BCPL, it evolved a type structure; created on a tiny machine as a tool to improve a meager programming environment, it has become one of the dominant languages of today. This paper studies its evolution. Many later languages have borrowed directly or indirectly from C, including C#, D, Go, Java, JavaScript, Limbo, LPC, Perl, PHP, Python, and Unix's C shell. The most pervasive influence on these languages has been syntactical, and they tend to combine the recognizable expression and statement syntax of C with underlying type systems, data models, and semantics that can be radically different. C++ started as a preprocessor for C and is currently nearly a superset of C.Before there was an official standard for C, many users and implementers relied on an informal specification contained in a book by Ritchie and Brian Kernighan; that version is generally referred to as "K&R" C. In 1989 the American National Standards Institute published a standard for C (generally called "ANSI C" or "C89"). The next year, the same specification was approved by the International Organization for Standardization as an international standards .ISO later released an extension to the internationalization support of the standard in 1995, and a revised standard (known as "C99") in 1999. The current version of the standard (now known as "C11") was approved in December of 2011.

The Keypad of the HCS12

PORT A of the microcontroller board HCS12 connects to a 4x4. The data direction register of this board is configured as a bidirectional register so that PA0-PA3 are output columns and PA4-PA7 are input rows. In addition, in order to set the pull up resistor of PORT A pins ON, we should set PUCR to 0x01.

16 x 2 LCD

PORT K of HCS12 is connected to an LCD screen that contains two lines each with 16 characters for each line. To write to the LCD, some pre-written C Functions can be used.

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- Write a C program that can access I/O registers.

2- Create programs that represent an embedded system.

3- Use the CodeWarrior IDEto develop and simulate C programs for the HCS12 microcontroller.

4- Writesimple C language programs for connecting DIP switches and Key Pad with LCD.

5- To be able to debug, download and test the written c programs on the Dragon12+ Board.

2. Design and Results

Task 1: Reading DIP Switches and Writing them to LEDs

In this task it is required to use the information that have been learnt in Lab-3 and the C Tutorial in order to write a program that takes the inputs from the DIP switches and outputs their status in the LEDs. The code for this task is shown below.

This task asks us to read the status of DIP switches and write it to the LEDs. This is the same as what we did in the previous lab but instead of using assembly language, we will use the C language. The complete code is shown below in figure 1.

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

void main(void) {
EnableInterrupts;

DDRB = 0XFF;
DDRJ  = 0XFF;
PTJ = 0;
DDRP = 0XFF;
PTP = 0X0F;
DDRH = 0;


for(;;) {

PORTB =PTH;
}
}
```

Figure 1: C Program of task1

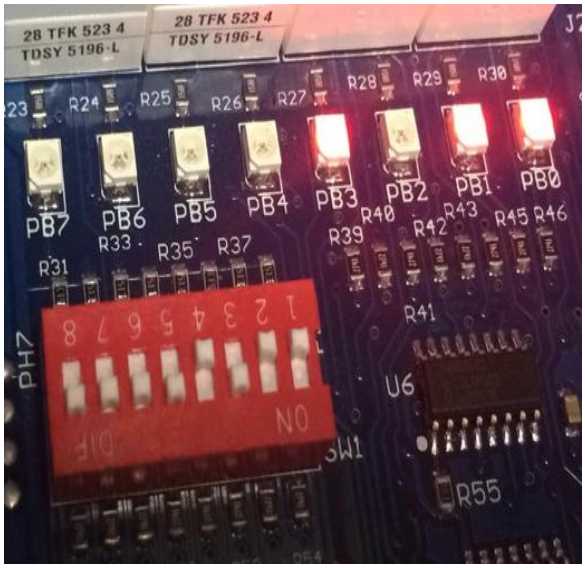

Figure 2: reading 0x0B from switches and writing to LEDs

As can be seen from the code above, our whole code goes in the main function. This language is simpler than the assembly in a way that we can directly set the direction registers as inputs or outputs in one step. Here we set PORTB, PORTJ and PORTP to output mode by initializing their data direction register with 0XFF which stands for hexadecimal FF. Also, we set PTH to input mode in order to read the switches state by setting its data direction register to 0. An infinite loop is used to continuously read the state of the DIP switches and load the value to the LEDs using PORTB =PTH

Task 2: Reading Key Pad and Writing it to LEDs

This task asks us to read input from the keypad and write to the LEDs. Again, we use the C language to write a program. It is important to note that the 4x4 keypad is connected to PORTA, which should be configured as bidirectional, meaning that part of it will be used as input and another part will be used as output. The full c program is shown below in figure 3.

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


int key_pad(void);


void main(void) {

EnableInterrupts;

   DDRB = 0XFF;
  DDRJ  = 0XFF;
  PTJ = 0;
  DDRP = 0XFF;
  PTP = 0X0F;
  DDRA = 0X0F;
  PUCR = 0X01;

  for(;;) {

  PORTB =key_pad();

  }
}
```

Figure 3: C Program of task2

```
intkey_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; }
```

Figure 4: the key_pad() function definition

We can see from figure 2 above that we set the data direction register of PORTA to 0X0F to attain the bidirectional configuration mentioned earlier. In addition, we set PUCR to 0X01 to to turn on the pull up resistors for PORTA pins. Inside the infinite loop, we store the value of the pressed button from the keypad to the LEDs PORTB by calling the key_pad() function. The definition of this function is shown in figure 4. It is used to check all columns and rows on the keypad and return a hexadecimal integer representingthe key that is currently pressed. The LEDs will be updated every time a new key is pressed.

Task 3: Writing to LCD

This task asks us to write specific text on the LCD display. The LCD display consists of 2 lines. In the first line, we are asked to write the text "ELCE 333 Lab" and on the second line "Microprocessor". Also, these lines of texts should alternate, each displayed for 0.5 seconds, causing a flashing effect. The complete Cprogram is shown below in figure 5.

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

void main(void) {
            EnableInterrupts;

LCD_Init ();

for(;;) {

LCDWriteLine (1, "ELCE 333 Lab");
delay(500);
LCD_clear_line(1);

LCDWriteLine (2, "Microprocessor");
delay(500);
LCD_clear_line(2);

  //  _FEED_COP(); /* feeds the dog */
  }
}
```

Figure 5: C Program of task3

Figure 6: LCD text effect

As can be seen in the code above, we included a file “lcd.h”. This file contains ready functions that will allow us to access and alter the LCD display easily. The first function call LCD_Init () is used to initialize the connection to the LCD. After the connection is made, we use the LCDWriteLine (1, "ELCE 333 Lab") to write the text in line 1 and LCDWriteLine (2, "Microprocessor") to write in line 2. Moreover, after writing each line, we need to display it for 0.5 seconds and therefore we use the function delay(500) which takes the delay time in msecs as a parameter. Finally, we use the function LCD_clear_line(int line) to clear the line passed as a parameter. This code is placed in an infinite loop to create a continuous flashing text effect.

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

This task asks us to modify the code from Task 1 and display the value of the DIP switches in the LEDs as well as the LCD. Since Task 1 already displays the value on the LEDs, we need to just add the code for the LCD. The complete code is shown below in figure 7.

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

void main(void) {

EnableInterrupts;

   LCD_Init() ;

  DDRB = 0XFF;
  DDRJ  = 0XFF;
  PTJ = 0;
  DDRP = 0XFF;
  PTP = 0X0F;
  DDRH = 0;

  for(;;) {

  PORTB =PTH;
  LCDWriteInt(PTH);
  delay(100) ;
LCDWriteLine(1,"");
  } }
```

Figure 7: C Program of task4

The modification we did above is that we use initialized a connection with the LCD port and used LCDWriteInt(PTH) function that takes an integer as a parameter and write it to the LCD. Here, we will pass port H (PTH) to the function, as it has the value of the DIP switches state. In order to avoid writing the value of the DIP switches more than once on the LCD, we have to clear the line after each write. For this to work properly, we have to delay for a short time (0.1 secs) and then use LCDWriteLine(1,"") to clear the first line by keeping the string parameter empty. Writing to the LEDs is the same as in task 1.

Task-5: Interfacing LCD with Key Pad and Push Buttons

This task asks us to write a c program that will prompt the user to enter a number using the keypad and then depending on whether the right-most DIP switch PTH0 is lifted or not, it will either display the number pressed on the keypad, or half that number on the LCD. If the PTH0 is lifted (bit =1), we should display the number as it is, otherwise (bit = 0) we display the number divided by 2. The complete program is shown below in figure 8.

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

int key_pad(void);
void main(void) {
float  x1;
int x2;
    LCD_Init() ;
    EnableInterrupts;

  DDRH = 0;
  DDRA = 0X0F;
  PUCR = 0X01;

  for(;;) {

   LCDWriteLine(1,"Enter a number:");

  if(PTH_PTH0 == 1) {
  delay(100) ;
LCDWriteLine(2,"");
    x1 = key_pad()/2.0;

  LCDWriteFloat(x1);
  }

  else {
  delay(100) ;
LCDWriteLine(2,"");

  x2 = key_pad();
  LCDWriteInt(x2);
```

Figure 8: C Program of task5

As we can see in the code above, we initialize PORTA as bidirectional port as we did before and initialize the LCD connection by calling its appropriate function. Moreover, we define 2 variables, x1 and x2, 1 for each case. In order to prompt the user to press a key, we print “Enter a number” on the 1st line of the LCD. Next, we use the conditional if statement if(PTH_PTH0 == 1) to check the state of the PTH0 switch. If the switch represents a value of 1, we will initialize x1 with the keypad value divided by 2 and print the float result on the 2nd line of the LCD using the function LCDWriteFloat(x1). Otherwise, we will initialize x2 with the direct value of the keypad key that is pressed and display this integer using LCDWriteInt(x2). It is important to clear the second line before writing a new value so that we do not have a history of keys pressed. The definition of the key_pad() function used to attain the value of the key pressed was shown previously in figure 4.

3. 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.

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"
void main(void) {

            EnableInterrupts;

   DDRB = 0xFF;
   DDRJ = 0xFF ;
   DDRP = 0xFF ;
   PTP = 0x0F  ;
   DDRH = 0x00 ;

   PORTB = 1;

for(;;) {

 if ( PTH == 0xFB ) {

    PORTB =PORTB<<1 ;
    delay(1000);

if (PORTB == 0){
     PORTB =  1;
     delay(1000);
    }
  } else if ( PTH == 0xFF ) {

    PORTB = PORTB>>1 ;
    delay(1000);

if (PORTB == 0){
     PORTB =  0x80;
     delay(1000);

    }
  }
}}
```

First, we initialize PORTB with a value of 1, which is equivalent to turning on LED0. From there, we can either shift this 1bit right or left, depending on the state of switch PH2. Therefore, in the infinite loop, we first check the case where the switch PH2 has a value 0 while all other switches have a value of 1. This is represented by a hex of $FB. If this is true, then we use the logical shift left operator << as follows PORTB<<1 to shift the 1 bit stored in PORTB 1 position to the left. Also, after the bit shifts fromthe leftmost position, PORTB will become 0 and we have to re-initialize it to 1. On the other hand, we check the second case where PH2 and all other switches have a value of 1, representing a hex value of $FF. If this case is true, we logically shift the 1bit to the right using PORTB>>1, again checking when it reaches the rightmost position and re-initializing PORTB with a hex of $80, causing the bit to rotate to the leftmost position and then continue shifting left. We should use delay after shifting PORTB to make the effect slower and more visible.

Figure 9: C Program of assignment1

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)

```
#include <hidef.h>
#include "derivative.h"
#include "lcd.h"
void main(void) {
int num1;
int num2;
EnableInterrupts;
  DDRH = 0x00 ;
  DDRA = 0x0F;
  PUCR = 0x01;
LCD_Init ();

for(;;)
 {
num1 =  ((PTH &0xF0 )>> 4);
num2 = key_pad();

LCDWriteLine(1,"");
delay(100) ;
LCDWriteLine (1, "no.1=");
LCDWriteInt (num1);
LCDWriteChar(' ');
LCDWriteChar('n');
LCDWriteChar('o');
LCDWriteChar('.');
LCDWriteChar('2');
LCDWriteChar('=');
LCDWriteInt (num2);
delay (100);

LCDWriteLine(2,"");
delay(100) ;

if ((PTH & 0x0F) == 0x01)
{  LCDWriteLine (2, "Op= + result =");
LCDWriteInt (num1+num2);
 } else if ((PTH & 0x0F) == 0x02)
{  LCDWriteLine (2, "Op= - result =");
LCDWriteInt (num1-num2);
 } else if ((PTH & 0x0F) == 0x04)
{  LCDWriteLine (2, "Op= * result =");
LCDWriteInt (num1*num2);
 } else if ((PTH & 0x0F) == 0x08)
{  LCDWriteLine (2, "Op= / result =");
LCDWriteInt (num1/(num2));
 }
 }}
```

Figure 10: C Program of assignment 2

Here, we start by initializing 2 integer variables, 1 to store the state of the 4 most significant switches and the other to read the key pressed on the key pad. In order to read the 4 left most switches, we have to AND PTH with the hex $F0, to give us only the values of the first 4 significant switches. Then, we shift the result 4 bits to the right to remove the 4 zeros in the lest significant positions. To get the second number, we just use the key_pad() function as previously done. Next, we have to use conditional if statements to check which operation we need to carry out. Here, we have to AND PTH with the hex $0F in order to get the 4 least significant positions. Then we check the 4 cases where the result can give a hex of $01, $02, $04 or $08 and carry out the operation assigned to each case. The last condition results in a division operation so we must use LCDWriteFloat(float) instead of LCDWriteInt(int).

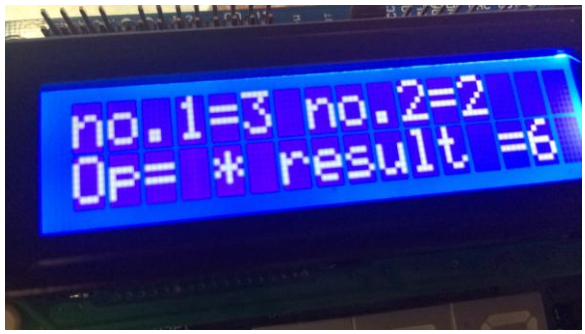

Figure 11: LCD sample result for assignment2

4. Conclusions and Recommendations

After the completion on the provided tasks in this lab session, students got the expected results after testing their code on the dragon board. Moreover, they are now confident that the theoretical knowledge that they own from the class courses matches the implementation results that they achieved in the laboratory. Good background knowledge is gained in writing c language program and using the DIP switches and the key pad with LCD. In addition, students learned how to How to assemble, download and run a C program using CodeWarrior C compiler and Dragoon12Plus Trainer board. Finally, they are able to Download, run, and test code on a Dragoon12+Board. Students got maximum benefit from this lab session and many new techniques were introduced to them during the lab session, this widens their knowledge and skills in programming.