Khalifa University of Science, Technology and Research

Electronic Engineering Department

ELCE333Microprocessor Systems laboratory

Laboratory Experiment 5

SERIAL COMMUNICATION INTERFACE

Lab Partners

Leena ElNeel	100037083
Muna Darweesh	100035522
Shamsah AlNabooda	100036984

Date Experiment Performed: 25/2/2015
Date Lab Report Submitted: 4/3/2015

Lab Instructors:

Mahmoud Khonji

Mohammed Ali Saif Al Zaabi

Spring 2015

Table of Contents

List of Figures and Tables

List of figures

List of tables

Summary

This report illustrates the concept of using the programming with C languages and use of the asynchronous serial communication interface in HCS12 microcontroller.

In this experiment contains different tasks. Some of them is about is Transmitting Characters and String. Others are is about ASCII Data and Serial I/O and Dip Switches Calculator. This report contains the results and the conclusion from the lab

1. Introduction

A serial communication interface (SCI) is an interface designed to transfer data only in asynchronous mode that utilizes the EIA 232 standard. Our microcontroller has two asynchronous serial Communications interfaces which are SCI0 an SCI1 and an extra synchronous serial peripheral interface, SPI. The extra chip allows the microcontroller to communicate with deceive that implements RS-232 standard. Port S of HCS12 is connected to two signal pins for each SCI channel.

-SCI0 - shares the use of Port S pins PS0 (RxD0) and PS1 (TxD0)
-SCI1 - shares the use of Port S pins PS2 (RxD1) and PS3 (TxD1)

There are 10 registers for status, control and data transfers. For examples:

- **SCnDRL** : is Data Register Low in which the character is transmitted or received.
- **SCnCR2** :is Control Register 2, **TE**& **RE** enable transmitting & receiving data
- **SCnCR1** : is Control Register 1 has advanced features as well as parity, bit length, and stop bits setting.
- **SCnBDH** & **SCnBDL** is SCIn Baud Rate Control which are double registers for setting the baud rate.
- **SCnSR1:is** SCIn Status Register 1 - **TDRE** responsible for stating that data has been transmitted or **RDRF** responsible for stating that data has been received respectively.

*The setup for asynchronous serial port is the following:

1. Set baud rate with **SBR12-0** bits of **SCnBDH** and **SCnBDL**. Specfic values are calculated based on the equation SBR = $f_E \div 16 \div$ baud rate where fE is the processor clock speed.
2. Enable transmitter and receiver (**TE** and **RE** bits of **SCnCR2**)
3. Select 8-bit or 9-bit mode (**M** bit of **SCnCR1**)
4. If using parity, enable parity (**PE** bit of **SCnCR1**) and choose odd or even (**PT** bit of **SCnCR1**)
5. If using interrupts enable **TIE** or **RIE** interrupt in **SCnCR2**

C library functions can be used to start then access the Serial Interface. The SCI1 library functions are presented in table 1 and are written in a header file (sci1.h) which is included in the main program by the aid of this command: #include "sci1.h"

Table 1: SCI interface library functions

Function	Input Parameters	Description
void SCI1_Init(unsigned short baudRate);	Baud Rate Ex (BAUD_9600)	Initialize Serial port SCI1 with baud rate in bits/sec
char SCI1_InStatus(void);	-	Checks if new input is ready
char SCI1_OutStatus(void);	-	Checks if output data buffer is empty
char SCI1_InChar(void);	-	Reads 2 nibbles character from SCI1
SCI1_InString(char *, unsigned short);	1) Empty string “” 2) String Length	Reads in a String until a carriage return is inputted or until max length of the string is reached
unsigned short SCI1_InUDec(void);	-	Reads ASCII input in unsigned decimal format
void SCI1_OutChar(char);	Char: 'A'	Writes 2 nibbles character to SCI1
void SCI1_OutString(char *pt);	String “Hi”	Writes a string of characters character to SCI1
void SCI1_OutUDec(unsigned short);	Int: 1	Writes ASCII output in unsigned decimal format

1.1 Aim

To introduce the students to the programming and the use of the asynchronous serial communication interface in HCS12 microcontroller.

1.2 Objectives

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

1- Understand serial I/O and its parameters

2- Write an HCS12 program to send and receive data from and to a PC.

3- To compile, download and debug/test a C program using CodeWarrior C compiler and Dragon12 Plus Trainer board.

2. Design and results

Task 1: Transmit Characters

In the first task of this lab we explored HCS12DG256 on board asynchronous serial communication SCI1 in which we were given the following code to run it and debug it:

```
#include <hidef.h>
#include "derivative.h"
#include "sci1.h"
void main(void)
{
SCI1_Init(BAUD_9600);
for(;;)
 {
 SCI1_OutChar ('*');// transmits * to SCI1
 SCI1_OutChar (0x0A);// new line
 SCI1_OutChar (0x0D);// carriage return
 } /* loop forever */
}// end of main
```

The difference in this code than previous lab is that the header file of SCI1 was added to enable usage of functions related to serial port SCI1. Also, in the main body, the serial port SCI1 was initialized with a baud rate of 9600 bps. In addition to that, in the loop, the function "*void SCI1_OutChar(char)"* is used to display a character output to Tera Term display.

As we can notice in figure 1that the character '*' was printed in each line separately (because of new line and carriage return) and continuously (because of the forever running loop).

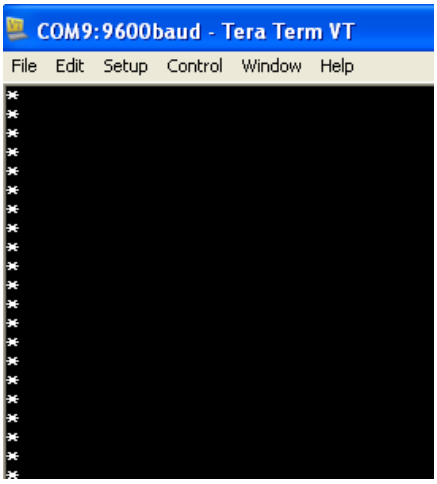

Figure 1: Transmit char '*' to serial port

Task 2: Transmit String

In this task, we were asked to modify the given code in task 1 but declare a string output. Instead of the stars we want to output "I love microcontroller" endlessly. The modified code is shown below:

```
#include <hidef.h> /* common defines and macros */
#include "derivative.h" /* derivative-specific definitions */
#include "sci1.h"
void main(void)
{SCI1_Init(BAUD_9600);
for(;;)
  {
  SCI1_OutString ("I love microcontroller");// transmits string to
SCI1
  SCI1_OutChar (0x0A);// new line
  SCI1_OutChar (0x0D);// carriage return
  } /* loop forever */
/* please make sure that you never leave main */}
```

The difference in this code is that we print a string using the function "*void SCI1_OutString()*" to display a string output to Tera Term display. As we can notice in figure 2 the string was printed in each line separately (because of new line and carriage return) and continuously (because of the forever running loop).

The figure below shows the output after running the program:

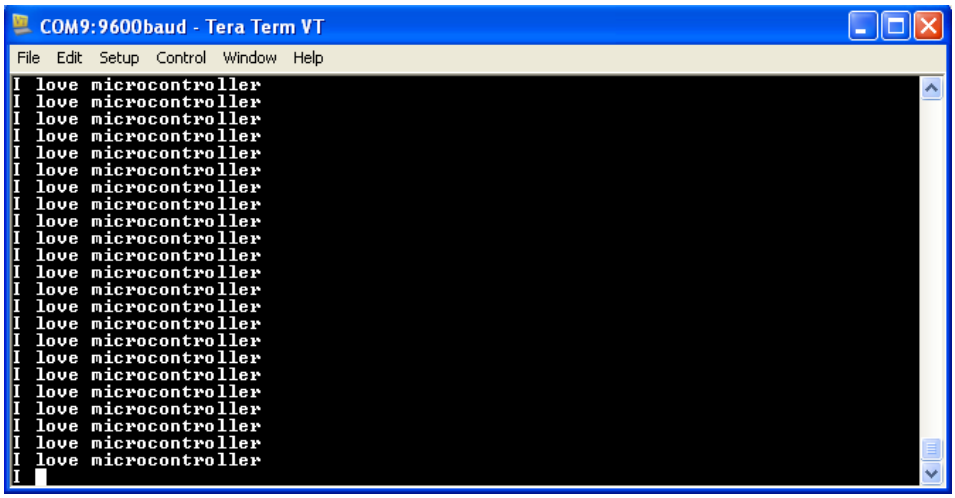

Figure 2: Transmitting I love microcontroller

Task 3: ASCII Data and Serial I/O

In the third task, we are asked to write a code that allows the user to enter a character and displays the ASCII equivalent on the screen in the following format **"The ASCII code for A is 65"** but however when the ESC button is pressed the program should display **"Bye"** and it should stop. The written code is shown below:

```
#include <hidef.h>
#include "derivative.h"
#include "sci1.h"
void main(void)
{char y;
SCI1_Init(BAUD_9600);
for(;;)
 { if (y == 0x1B)
 {
 SCI1_OutString  ("Bye");// transmits * to SCI1
 SCI1_OutChar (0x0A);// new line
 SCI1_OutChar (0x0D);// carriage return
  break;
 } //end of IF
 else {
    SCI1_OutString ("The ASCII code for ");// transmits * to SCI1
   y= SCI1_InChar();
   SCI1_OutChar(y);
   SCI1_OutString (" is ");// transmits * to SCI1
   SCI1_OutUDec(y);
   SCI1_OutChar (0x0A);// new line
   SCI1_OutChar (0x0D);// carriage return
 }//end of else
```

In this code, we had to define two cases; the first case is when the user enters a valid letter. The second is for when the user presses ESC. We defined *if (y == 0x1B)* which represents ESC in ASCII code, the code will output "Bye" on a new line and terminate. However, for when the user enters a character, the code outputs "The ASCII code for" and the entered character. Then, the code uses the function *SCI1_OutUDec(y);* to display the ASCII equivalent of the entered character. When a valid character is entered, the program will loop motivating the user to enter another value.

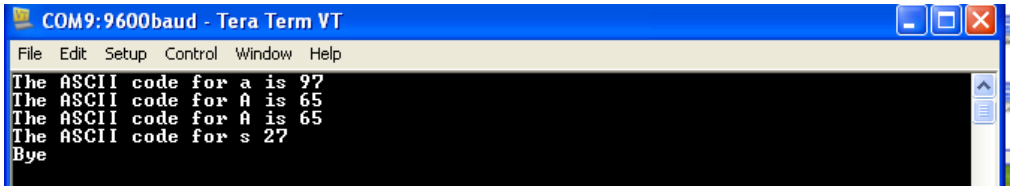

Figure 3: HyperTerminal Screen display of task 3

Task 4: DIP switches calculator

In this task we were asked to write a C# code for calculator using the first 4 DIP switches from the right for the first number and the other left 4 DIP switches for the second number whereas the operation is entered by user using the serial terminal.

In order to write such code, we had first to define all variables that are needed (i.e. variables for operands (int), operations (char) and operation functions (int & float). Then we assigned 4-LSB of PTH (that controls the DIP switches) to be used for the num1 and the 4-MSB of PTH to be used for num2. In the serial terminal, the decimal equivalent representation of binary num1 and num2 in PTH is displayed followed by message that pulls the user to enter an operation (+, -, *, /) to be performed. The code include an IF condition statement that compares the user entered operation and do the relevant arithmetic operation and result displaying using the function "void SCI1_OutUDec(unsigned short)". The addition and the multiplication operations are performed ordinarily without any special cases; however, the subtraction operation contains a case of a negative result when the second operand is larger than the first one. Therefore, before performing the subtraction num1 and num2 are checked such that if num2 > num1 then sub= num2 – num1 with adding a negative sign to the number, else perform an ordinary subtraction operation (i.e. sub= num1-num2). The other special case we dealt with in this task is the case of non-integer division result and to overcome this problem "*div*" variable was defined as float and the following function was used to represent the division result : "*SCI1_OutFloat(div, num1,num2)*".

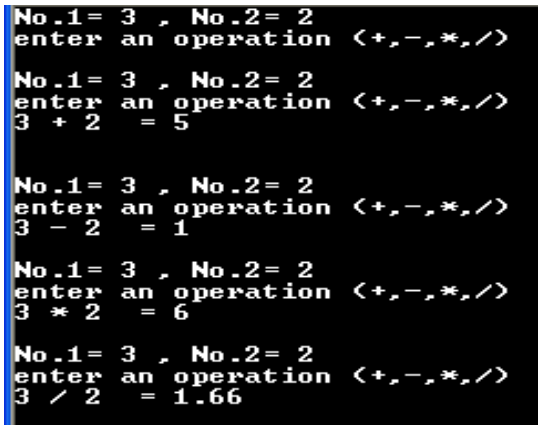

Figure 4: DIP switch calculator results

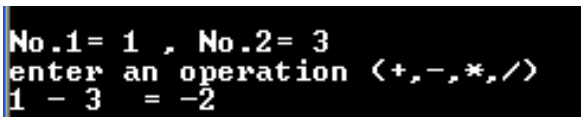

Figure 5: Case of negative results

The following is the code for this task:

```
#include <hidef.h>
#include "derivative.h"
#include "sci1.h"
void main()
{
int num1, num2;
int sum, sub, mul,subn;
float div;
char op;
SCI1_Init(BAUD_9600);
for(;;)
 { // 4-LSB of DIP switches assigned to No.1
   num1= PTH_PTH0*1 +PTH_PTH1*2 +
PTH_PTH2*4 + PTH_PTH3*8;
   // 4-MSB of DIP switches assigned to No.1
   num2= PTH_PTH4*1 +PTH_PTH5*2 +
PTH_PTH6*4 + PTH_PTH7*8;
   op= SCI1_InChar();
   SCI1_OutString ("No.1= ");
   SCI1_OutUDec(num1);
   SCI1_OutString (" , No.2= ");
   SCI1_OutUDec(num2);
   SCI1_OutChar (0x0A);// new line
   SCI1_OutChar (0x0D);// carriage return
   SCI1_OutString ("enter an operation (+,-,*,/)");
   SCI1_OutChar (0x0A);// new line
   SCI1_OutChar (0x0D);// carriage return
if (op =='+')
 {
  sum= num1+num2;
  SCI1_OutUDec(num1);
  SCI1_OutString (" + ");
  SCI1_OutUDec(num2);
  SCI1_OutString ("  = ");
  // display decimal value of sum on serial terminal
  SCI1_OutUDec(sum);
  SCI1_OutChar (0x0A);// new line
  SCI1_OutChar (0x0D);// carriage return
 SCI1_OutChar (0x0A);// new line
   SCI1_OutChar (0x0D);// carriage return
 } if (op =='-')
 {
  sub= num1-num2;
  SCI1_OutUDec(num1);
  SCI1_OutString (" - ");
  SCI1_OutUDec(num2);
  SCI1_OutString ("  = ");
  // To handle negative results (i.e. num2 >num1)
  if (num2>num1)
  {
   subn=num2-num1;
   SCI1_OutChar ('-');
   SCI1_OutUDec(subn);
  } else {
// when subtraction result is positive (i.e. num1>num2)
   SCI1_OutUDec(sub);
  }//end of else
  SCI1_OutChar (0x0A);// new line
  SCI1_OutChar (0x0D);// carriage return
 }
 if (op =='*')
 {
  mul= num1* num2;
  SCI1_OutUDec(num1);
  SCI1_OutString (" * ");
  SCI1_OutUDec(num2);
  SCI1_OutString ("  = ");
  SCI1_OutUDec(mul);
  SCI1_OutChar (0x0A);// new line
  SCI1_OutChar (0x0D);// carriage return
 }
 if (op =='/')
 {
  div= num1/num2;
  SCI1_OutUDec(num1);
  SCI1_OutString (" / ");
  SCI1_OutUDec(num2);
  SCI1_OutString ("  = ");
  /* as the division result can sometimes be float
   we use the following function */
  SCI1_OutFloat(div, num1,num2);
  SCI1_OutChar (0x0A);// new line
   SCI1_OutChar (0x0D);// carriage return
  }
   SCI1_OutChar (0x0A);// new line
   SCI1_OutChar (0x0D);// carriage return
   break;
 } //end of loop
} // end of main
```

3. Assigned question

1) Write an HCS12 program in C to receive bytes serial and put them on PORTB. Set the baud rate at 19200, 8-bit data.

The first step in this code is initializing the 4- LSB LEDs for displaying the output by assigning the outputs and inputs to corresponding ports. Then, in the loop section, we wrote functions for reading the character input from the serial ports. In case of reading numbers from 10 to 15 we converted it corresponding letter representation (i.e. A to F) to and then display its binary code in PORTB.

```
#include <hidef.h>
#include "derivative.h"
#include "SCI1.h"
void main(void) {
char x;
int n;
SCI1_Init(BAUD_9600);
   // initilizing LEDs for displaying the results
   DDRB = 0x0F;
// becuase we will only represent 4 bits [0,F]
   DDRJ = 0xFF;
   DDRP=  0x0F;
   DDRT = 0xFF;
   PTJ = 0x00;
   PTP= 0x0F;
   PORTB =0;
for(;;) {
   n= 0;
   SCI1_OutString ("Enter a Character : ");
   x= SCI1_InChar ();
   if (( x=='a') || (x=='A')) n=10 ;
   if (( x=='b') || (x=='B')) n=11;
   if (( x=='c') || (x=='C')) n=12;
   if (( x=='d') || (x=='D')) n=13;
   if (( x=='e') || (x=='E')) n=14;
   if (( x=='f') || (x=='F')) n=15;
```

```
   if (n != 0)
  {
    SCI1_OutUDec(n);
    PORTB =n;
   }
   else
   {
    SCI1_OutChar(x);
    PORTB = x;
   }

   SCI1_OutChar (0x0A);// new line
   SCI1_OutChar (0x0D);// retutn carriage

   } //end of loop
}//end of main
```

Figure 7: Assignment Question 1 results

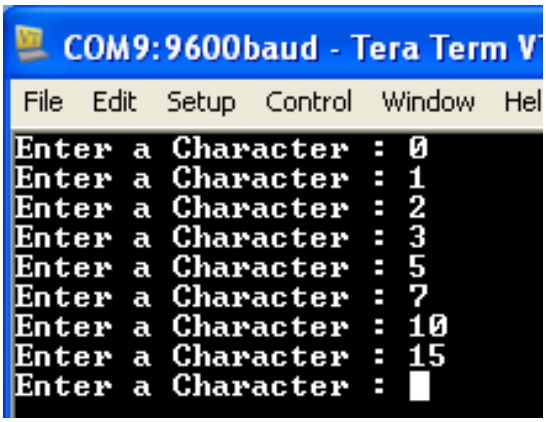

Figure 6: Assignment Question: TeraTerm results

2) Write a program to prompt the user to enter a sentence and then count the number of characters excluding spaces as well as the number of spaces in the entered sentence. Then display the numbers on the serial terminal. Run the program and provide snap shot of your serial terminal.

In this task we define an array that takes each character and stored. Then a while loop that checks for the Null character, if not detected so checks for the number for spaces and characters. Then we display the number of the character-1 to remove the effect of the Enter key. Then the variables are cleared to read new sentence.

```
#include <hidef.h>
#include "derivative.h"
#include "SCI1.h"
void main(void) {


  int n=0;
        int charsp=0;
        int Charnum2=0;
        int less=0;
        char Array[30];
        SCI1_Init(BAUD_9600);

for(;;)
{
        SCI1_OutString ("Please Enter a
sentence: ");
  SCI1_InString(Array, 30);
  SCI1_OutString( Array+1);

 while(Array [n]!= 0x00)
 {
  if(Array[n] == 0x20)
   charsp++;
  else
        Charnum2++;
    n++;
}
  less= Charnum2-1;
SCI1_OutString (" has ");
SCI1_OutUDec(less);
SCI1_OutString ("characters &  ") ;
SCI1_OutUDec(charsp);
SCI1_OutString (" Spaces");
   n=0;
  charsp=0;
  Charnum2=0;
  less=0;
SCI1_OutChar (0x0A);
SCI1_OutChar (0x0A);
SCI1_OutChar (0x0D);
}
}
```

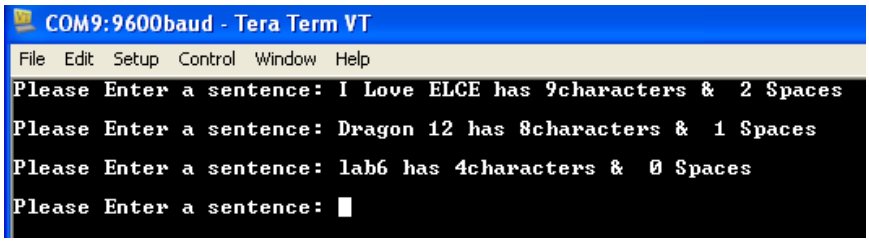

Figure 8: Results for assigned question

3. Conclusion

The aim of this lab session was to introduce students by the use of asynchronous serial communication interface in HCS12 and how to program it. Task one, requires to transmit the character * to SCI1 using the command "*SCI1_OutChar(char)"* and observes the output obtained in HyperTerminal or Tera Term display. In the second task, modifies the first task to display a string, where each string must be displayed in a new line. In the third task, it was required to designed a program. The program allows the user to enter a character which will be converted to its ASCII value than display it. This is done using the instruction "*SCI1_OutUDec(y)"*. If the user pressed "Esc" instead of the character the string "Bye" should be displayed in a new line and the transmission session. The fourth task asked us to write a C language program that reads two numbers from the Dip switches then do a specific operation such as (+, -, /, *), to display the whole operation and the result on serial terminal. In order to display the results if we have fraction such as the division operation it is required need to use the command "*SCI1_OutFloat (results, num1,num2)"* to display the full result. Also, the program should be able to give the right result in the subtraction operation in the case the first number is less than the second number. To do so, an if statement will be defined.