Microprocessor Systems Laboratory (ELCE333)

Laboratory Experiment No.4

MARKING SHEET

Student Names and ID No's: **Amal AlQasimi 100036830**
Shaikha AlBesher 100037064

Laboratory Section: Wednesday **Experiment Date: February 25, 2015**

No.	Criteria	Description	Weight %	Mark	Comments
1	Pre-lab	A mark will be allocated to each student that reflects his preparations for the lab.	20		
2	Performance In the lab	A mark will be allocated to each student individually that reflects his performance in the lab	10		
3	Results and Analysis	Documentation and analysis of the results for each task performed in the lab	30		
4	Summary/ Conclusions	Conclusions for each task performed in the lab	10		
5	Assignment Questions	Answers to assignment questions	20		
6	Report Presentation	Overall presentation of the report including proper layout and clarity of figures, tables, and graphs. Correct use of English language.	10		
	Total		**100**		

ELCE 333: Microprocessor Systems Laboratory

Lab Report- Experiment No.5

Experiment Title: HCS12 Input and Output Ports

Completed by:

Amal Al Qasimi (100036830)

Shaikha Al Besher (100037064)

Lab instructors:

Mahmoud Khonji

Mohammed Al Zaabi

Spring 2015

Table of Contents

Summary 4

1. Introduction 5

2. Aims and Objectives: 6

   2. 1 Aim 6

   2.2 Objectives 6

3. Lab tasks 6

   TASK-1: Transmit Characters 6

   TASK-2 Transmit String 7

   TASK-3: ASCII Data and Serial I/O 8

   TASK-4: Dip Switched Calculator+ BONUS 10

4. Analysis and Interpretation 13

5. Conclusions and Recommendations 13

6. Assignment Questions 14

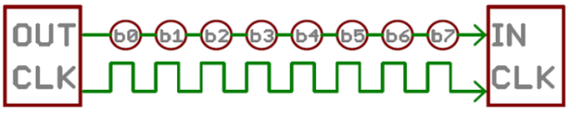
References 15

Summary

The following lab report will include the details of the fifth Microprocessor System's Laboratory lab session. It will first start off by stating the aims and objectives of the tasks assigned to us throughout the lab session. Following the aims and objectives, the report will discuss the purpose of each task separately with the results obtained upon the completion of the task. Based on the observations of the tasks completed, the analysis and interpretation section will include the explanation of why we obtained the results we did. Finally the report will end with a brief conclusion. The assignment questions given to us as part of the lab are answered after the lab repot.

1. Introduction

Embedded electronics is all about interlinking circuits (processors or other integrated circuits) to create a symbiotic system. In order for those individual circuits to swap their information, they must share a common communication protocol. Hundreds of communication protocols have been defined to achieve this data exchange, and, in general, each can be separated into one of two categories: parallel or serial. Parallel interfaces transfer multiple bits at the same time. They usually require buses of data - transmitting across eight, sixteen, or more wires. Data is transferred in huge, crashing waves of 1's and 0's. Whereas Serial interfaces stream their data, one single bit at a time. These interfaces can operate on as little as one wire, usually never more than four.



Example of a serial interface transmitting one bit every clock pulse.

Asynchronous Serial: Over the years, dozens of serial protocols have been crafted to meet particular needs of embedded systems. USB, and Ethernet, are a couple of the more well-known computing serial interfaces. Other very common serial interfaces include SPI, I^2C. Each of these serial interfaces can be sorted into one of two groups: synchronous or asynchronous.

A synchronous serial interface always pairs its data line(s) with a clock signal, so all devices on a synchronous serial bus share a common clock. This makes for a more straightforward, often faster serial transfer, but it also requires at least one extra wire between communicating devices. Examples of synchronous interfaces include SPI, and I^2C.Asynchronous means that data is transferred without support from an external clock signal. This transmission method is perfect for minimizing the required wires and I/O pins, but it does mean we need to put some extra effort into reliably transferring and receiving data.

Rules of Serial: The asynchronous serial protocol has a number of built-in rules - mechanisms that help ensure robust and error-free data transfers. These mechanisms, which we get for eschewing the external clock signal, are:

- Data bits
- Synchronization bits
- Parity bits
- Baud rate

Through the variety of these signaling mechanisms, there's no one way to send data serially. The protocol is highly configurable. The critical part is making sure that both devices on a serial bus are configured to use the exact same protocols.

Baud Rate: The baud rate specifies how fast data is sent over a serial line. It's usually expressed in units of bits-per-second (bps). This value determines how long the transmitter holds a serial line high/low or at what period the receiving device samples its line. [4]

2. Aims and Objectives:

2. 1 Aim: To be familiarized with the programming and utilize the asynchronous serial communication interface in HCS12 microcontroller.

2.2 Objectives:

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+ Trainer board.

3. Lab tasks

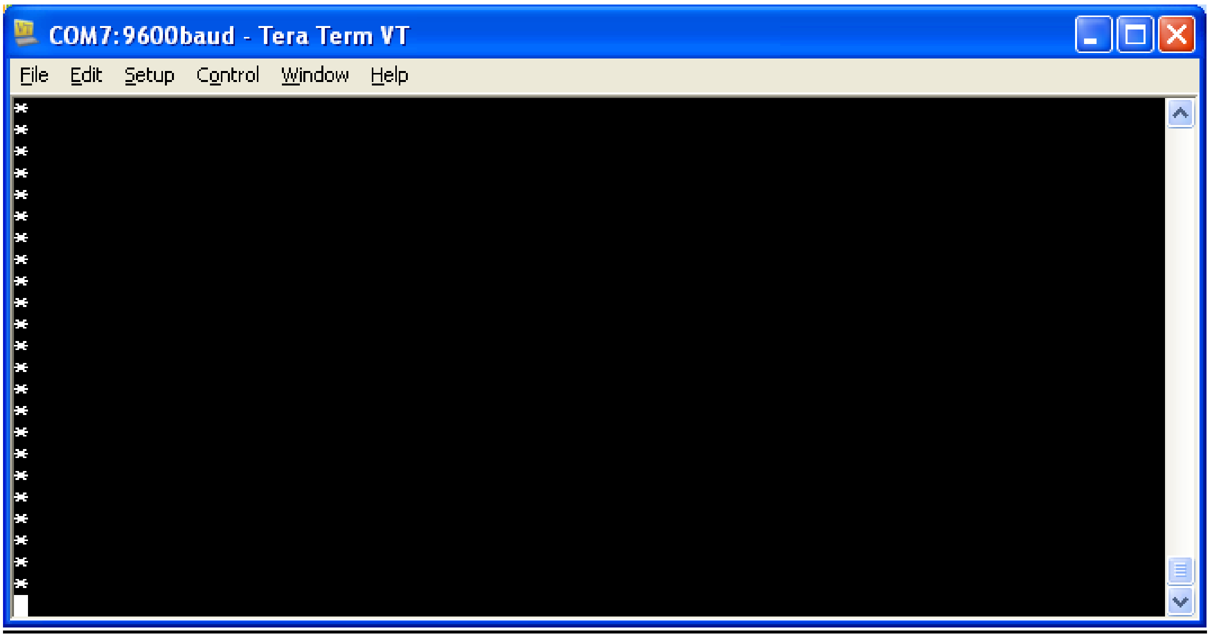
TASK-1: Transmit Characters

In this task we were asked to use the serial communication interface in HCS12run in order to run a given program and check the output check the output on the HyperTerminal or Tera Term display. The logic behind this code is to print a single defined character continuously each character on a separate line. The defined character in this code is '*'.

The code is:

```
#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 } }
```

The following figure shows the output:



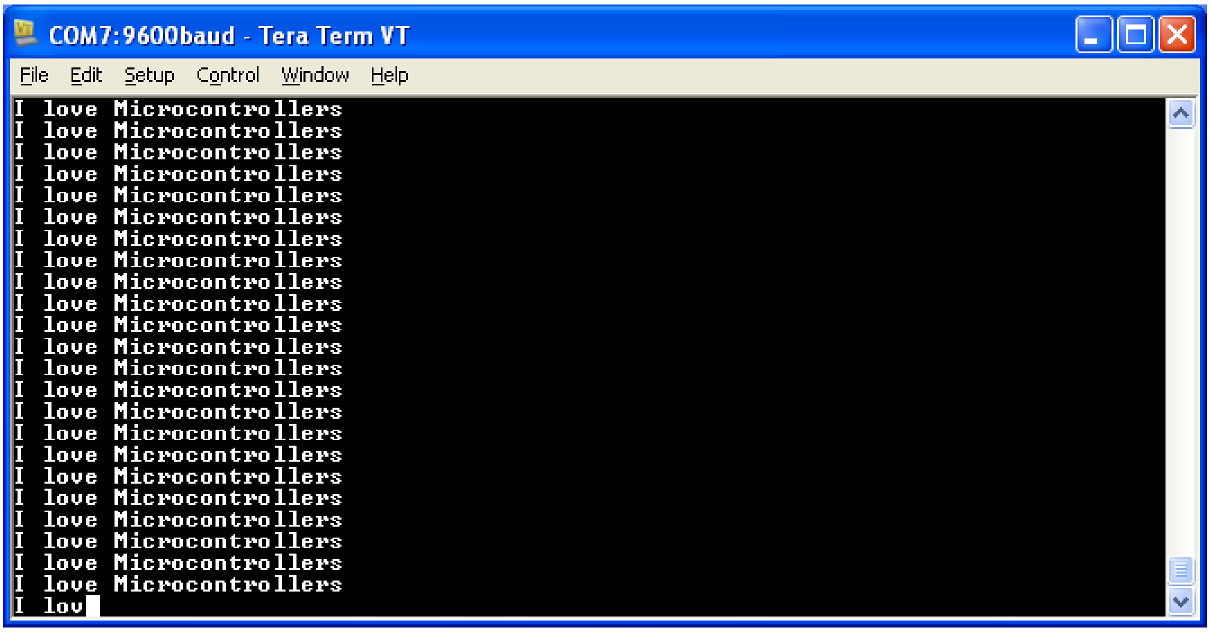
TASK-2 Transmit String

The aim of this task to modify the code used in task 1, in order to transmit the following string “I love Microcontrollers” continuously. Each string transmission should be displayed on the terminal on the start of a new line. The modification was done by replacing SCI1_OutChar function with SCI1_OutString function.

The modified code can be seen below:

```
#include <hidef.h>
#include "derivative.h"
#include "sci1.h"


void main(void) {

SCI1_Init(BAUD_9600);
for(;;) {
SCI1_OutString ("I love Microcontrollers"); // transmits I love Microcontrollers to SCI1
SCI1_OutChar (0x0A); // new line

SCI1_OutChar (0x0D); // carriage return
}
}
```

The output is shown in the figure below:



TASK-3: ASCII Data and Serial I/O

This task requires writing a code that:

1- Prints out the following text: **The ASCII code for**.

2- Allows the user to input a single character on the keyboard.

3- Prints out the following text: **is**

4- Prints out the ASCII code for the character typed.

5- Prints a carriage return, line feed and repeats starting at step 1.

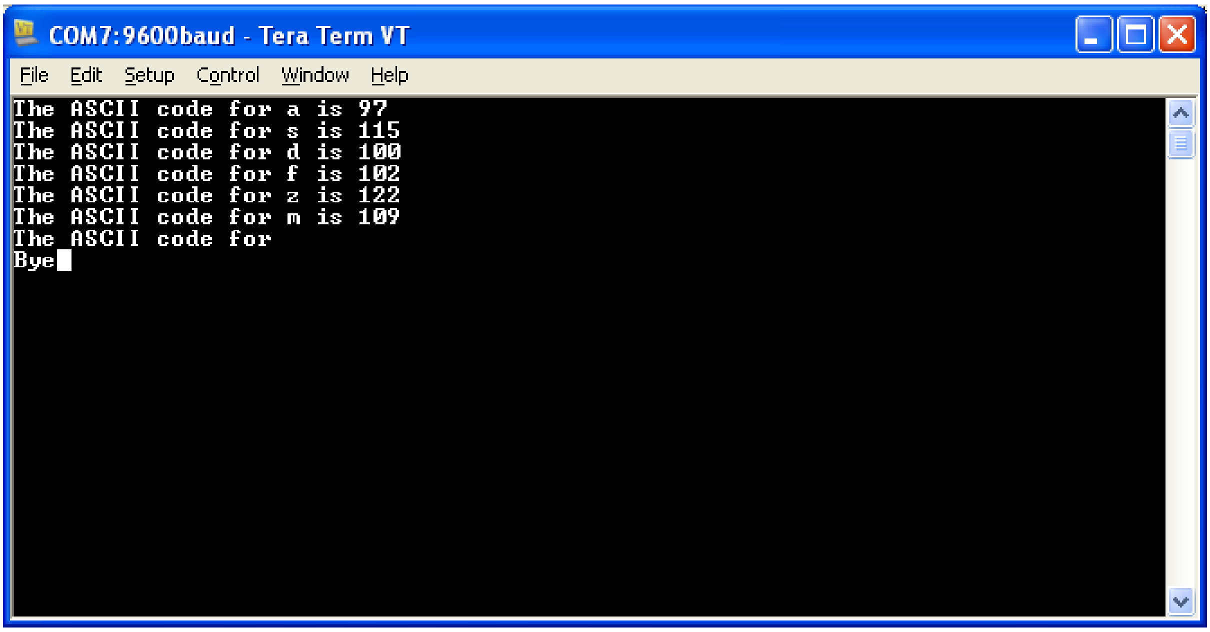
6- And finally displays **Bye** in new line if the user hits the Esc instead of the character, and the transmission session should be halted.

The created code for this program is shown below:

```
#include <hidef.h>
#include <hidef.h> /* common defines and macros */
#include "derivative.h" /* derivative-specific definitions */
#include "sci1.h"

void main(void) {
char x;
/* put your own code here */
SCI1_Init(BAUD_9600);
```

```
for(;;){
SCI1_OutString("The ASCII code for ");
x=SCI1_InChar();
if(x==0x1B){
SCI1_OutChar (0x0A);
SCI1_OutChar (0x0D);
SCI1_OutString("Bye");
break;
} else{

SCI1_OutChar (x);
SCI1_OutString(" is ");
SCI1_OutUDec(x);
SCI1_OutChar (0x0A);
SCI1_OutChar (0x0D);
}
}
}/* loop forever */
/* please make sure that you never leave main */
```

Samples of output:



In this task, we used if-else statements. It starts by checking whether the user input is Esc which has the value of "1B" in hexadecimal, or any other character. For any character inserted by user, its ASCII value will be calculated and displayed. If the user inters "Esc", its ASCII will also be displayed, but printing the word "Bye" would halt the transmission.

TASK-4: Dip Switched Calculator+ BONUS

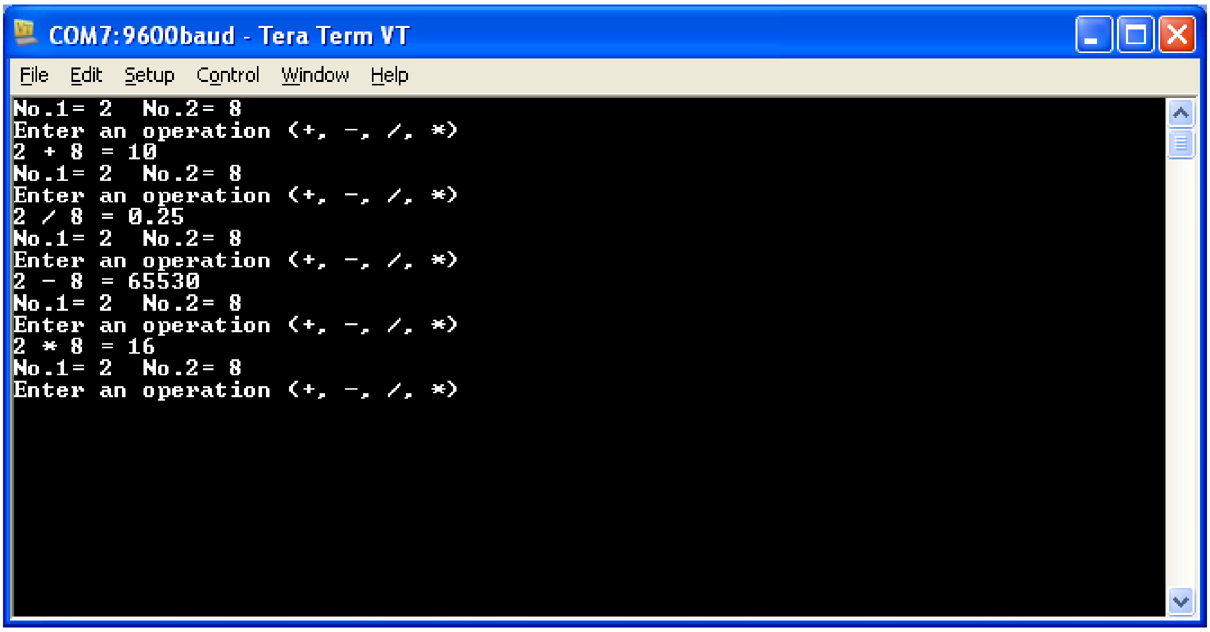
The aim behind this task is to start by reading 2 integer numbers from the 8 Dip switches, and then perform arithmetic operations including addition, subtraction, division, and multiplication. The first 4 switches represent the first number, and the second 4 represent the second number. Both numbers will be displayed on the Serial Terminal and the user must choose the operation to be performed, taking in consideration that when subtracting a large number from a small number the result will has a negative sign.

The code is shown below:

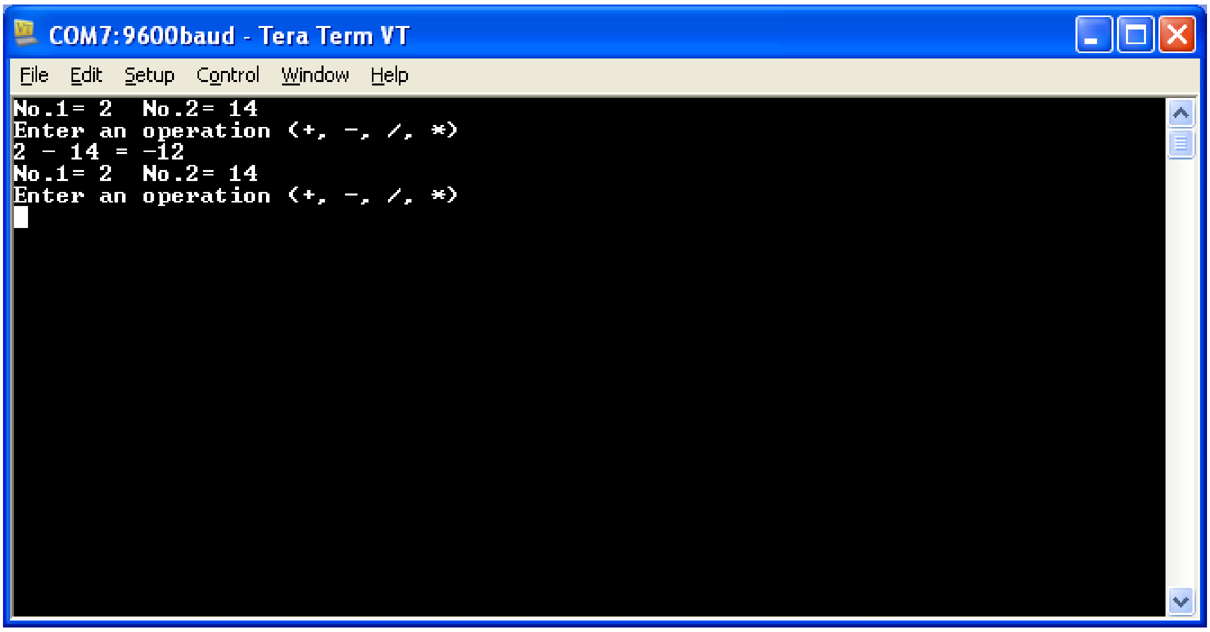
```
#include <hidef.h> /* common defines and macros */
#include "derivative.h" /* derivative-specific definitions */
#include "sci1.h"

void main(void) {
int num1, num2;
float n;
char op;
SCI1_Init(BAUD_9600);
for(;;){
num1=PTH_PTH0*1+ PTH_PTH1*2+ PTH_PTH2*4+PTH_PTH3*8;
num2=PTH_PTH4*1+ PTH_PTH5*2+ PTH_PTH6*4+PTH_PTH7*8;

SCI1_OutString("No.1= ");
SCI1_OutUDec (num1);
SCI1_OutString("  No.2= ");
SCI1_OutUDec (num2);
SCI1_OutChar (0x0A);
SCI1_OutChar (0x0D);
SCI1_OutString("Enter an operation (+, -, /, *)");
SCI1_OutChar (0x0A);
SCI1_OutChar (0x0D);
op=SCI1_InChar();
switch (op){
case (0x2B):{
   //+

  SCI1_OutUDec (num1);
  SCI1_OutString(" + ");
  SCI1_OutUDec (num2);
  SCI1_OutString(" = ");
  SCI1_OutUDec(num1+num2);
break;
 }

case (0x2D):{
   //-
if (num2>num1) {
  SCI1_OutUDec (num1);
```

```
 SCI1_OutString(" - ");
 SCI1_OutUDec (num2);
 SCI1_OutString(" = -");
 SCI1_OutUDec(num2-num1);


 } else {
 SCI1_OutUDec (num1);
 SCI1_OutString(" - ");
 SCI1_OutUDec (num2);
 SCI1_OutString(" = ");
 SCI1_OutUDec(num1-num2);

 }
break;
 }

case (0x2A):{
  //*
 SCI1_OutUDec (num1);
 SCI1_OutString(" * ");
 SCI1_OutUDec (num2);
 SCI1_OutString(" = ");
 SCI1_OutUDec(num1*num2);
break;
 }

case (0x2F):// /
 {
 SCI1_OutUDec (num1);
 SCI1_OutString(" / ");
 SCI1_OutUDec (num2);
 SCI1_OutString(" = ");
 SCI1_OutFloat(n,num2,num1);
break;
}

}
SCI1_OutChar (0x0A);
SCI1_OutChar (0x0D);

}
}
```

The Figures bellow show how the program is working:



Example of subtracting a larger number from a small number:



4. Analysis and Interpretation

Clearly, it has been understood that the lab was concentrated around the topic of asynchronous serial communication interface in HCS12 microcontroller in all four tasks. Task 1 consisted of running a given code and observing the output on the HyperTerminal or Tera Term display. The code contained instructions for a new line and carriage which returned to the start of the next line otherwise it kept printing the given sample side by side infinitely. The difference between Task 2 and Task 1 was that in Task 2 we were required to write a sentence and display it on the screen. In Task 3, the only difference was that a string code must be used where it was expected from the user to enter a character then the program will get then display the ASCII code of. Furthermore, the program kept asking the user to enter a letter until the user pressed the "ESC" button and ended the program. Task 4 was a bit more complex, where it required a code that read numbers and read an operation then performed that operation and displayed the result. The bonus task involved adding the functionality of being able to subtract numbers and displaying negative results.

5. Conclusions and Recommendations

Aiming to understand how to use asynchronous serial communication interface in HCS12 microcontroller, this laboratory composed of four different tasks where each of them was designed to learn and apply a unique skill in writing C language on the CodeWarrior program. In addition, we learnt how to develop a program which performed basic, yet essential mathematical calculations and explored the special case of negative numbers when subtracting two numbers. Overall, students were able to compile, download and debug/test a C program using CodeWarrior C compiler and Dragon12+ Trainer board. Having such knowledge will definitely come in handy in the future when completing larger projects in the future.

6. Assignment Questions

1)

```
#include <hidef.h>
#include "derivative.h"
#include "sci1.h"

void main(void) {

SCI1_Init(BAUD_19200);


for(;;) {
SCI1_OutChar ('*');          // transmits * to SCI1
SCI1_OutChar (0x0A);         // new line
SCI1_OutChar (0x0D);         // carriage return
} /* loop forever */ }
```

2)

```
#include <hidef.h>
# include “derivative.h”
#include “scil.h”

 Void main(void) {

Int  c=0;
Int  s=0;
Intnum=0;
Char stg[30];

SCI1_Init(BAUD_19200);
for(;;)
{
SCI1_OutString (“Enter a sentence: ”);
SCI1_InString(stg, 30);
SCI1_OutString( stg+1);
While(stg[num]!* $00) {
```

```
If(stg[num] == $44)
s++;
Else
c++;
num++;

SCI1_OutString (“ has ”);
SCI1_OutUDec(c);
SCI1_OutString (“ characters& ”) ;
SCI1_OutString(s);
SCI1_OutString (“ spaces”);
SCI1_OutChar (0x0A);// new line
SCI1_OutChar (0x0D);// carriage return

}/* loop forever */
}
```

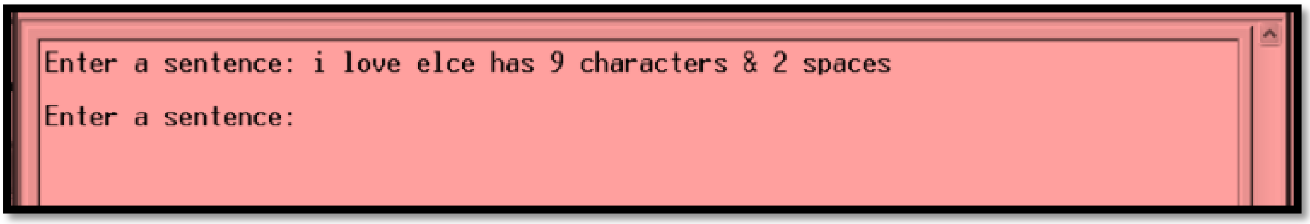


Figure 1 Assignment question 2 results in the hyper terminal

References

1. Course TextBook
2. http://www.evbplus.com/
3. MC9S12DP256 User'sManual
4. https://learn.sparkfun.com/tutorials/serial-communication/all.pdf