|  |
| --- |
| EEL 4742C-12 |
| EEL 4742 Laboratory |
| Experiment #3 |
|  |
| **Daniel Franco, Andrew Mendez**  **2/25/2014** |
|  |

|  |
| --- |
|  |

**Objective:** To develop C language and assembly language programs that converts ASCII characters to binary and hexadecimal numbers. The UART on the MSP430FG4618 will be used via HyperTerminal to enter in ASCII data and to display ASCII data.

**Apparatus List:**

* Dell Computer
* Monitor
* Keyboard
* Mouse
* CCS software
* MSP430FG4618

**Procedure and/or Design Methodology:**

This experiment had two separate parts to it, which we will describe in order as we worked on them

Part One, C code:

1. Write a C language program that inputs two hexadecimal ASCII numbers. Next, the program should convert these ASCII numbers to an 8 bit binary number. Finally, this program should display the equivalent ASCII character on a new line in the HyperTerminal window. The program should then print a new line and start over waiting for user input.
2. Write a C language program that inputs one ASCII character and displays on a new line in the HyperTerminal window its two digit hexadecimal value. The program should then print a new line and start over waiting for user input.
3. Write a C language program that inputs one ASCII character (“A – Z” or “a – z”). Next, the program should convert the uppercase letters to a lowercase and convert the lowercase letters to uppercase. The program should display on a new line in the HyperTerminal window the converted letter. The program should then print a new line and start over waiting for user input.
4. Write a C language program that inputs 8 ASCII “0” or “1” digits. Next, the program should convert this value to an eight bit binary number. Finally, the program should display on a new line in the HyperTerminal window, the hexadecimal value of this eight bit binary number. The program should then print a new line and start over waiting for user input.
5. Write a C language program that inputs ASCII characters and stores them in a character array until the enter key is hit (0x0D) or until 32 characters have been entered. Next, end the character array with the null character (0x00). Display this character (string) array on a new line in the HyperTerminal window. Finally, have the program sort these ASCII characters from lowest to highest and display on a new line this sorted character array.

Part Two, Assembly code:

1. Write an assembly language program that inputs two hexadecimal ASCII numbers. Next, the program should convert these ASCII numbers to an 8 bit binary number. Finally, this program should display the equivalent ASCII character on a new line in the HyperTerminal window. The program should then print a new line and start over waiting for user input.
2. Write an assembly language program that inputs one ASCII character and displays on a new line in the HyperTerminal window its two digit hexadecimal value. The program should then print a new line and start over waiting for user input.
3. Write an assembly language program that inputs one ASCII character (“A – Z” or “a – z”). Next, the program should convert the uppercase letters to a lowercase and convert the lowercase letters to uppercase. The program should display on a new line in the HyperTerminal window, the converted letter. The program should then print a new line and start over waiting for user input.
4. Write an assembly language program that inputs 8 ASCII “0” or “1” digits. Next, the program should convert this value to an eight bit binary number. Finally, the program should display on a new line in the HyperTerminal window, the hexadecimal value of this eight bit binary number. The program should then print a new line and start over waiting for user input.
5. Write an assembly language program that inputs ASCII characters and stores them in a character array until the enter key is hit (0x0D) or until 32 characters have been entered. Next, end the character array with the null character (0x00). Display this character (string) array on a new line in the HyperTerminal window. Finally, have the program sort these ASCII characters from lowest to highest and display on a new line this sorted character array.

**Design Specification Plan:**

C- Programs:

For our C programs we decided to take the approach of creating functions that would complete the task assigned, we felt like it would help us keep our main code looking organized and clean while still accomplishing the set goal. We heavily relied on the discussion that was presented in the pre-laboratory assignment to program our equations and arithmetic. We have provided screenshots of both, our main programs and the functions which we make calls to when the procedure is completed on the variables we input into the HyperTerminal.

Assembly programs:

Since the assembly programs were designed to complete the same tasks as the C programs, we followed the same plan, with the difference that this time we would be programming in a different language at a level closer to the hardware. The discussion provided to us in the pre-laboratory assignment helped us thoroughly because it gave us a direct idea of what we should do with the hardware in order to get it to complete the assigned tasks.

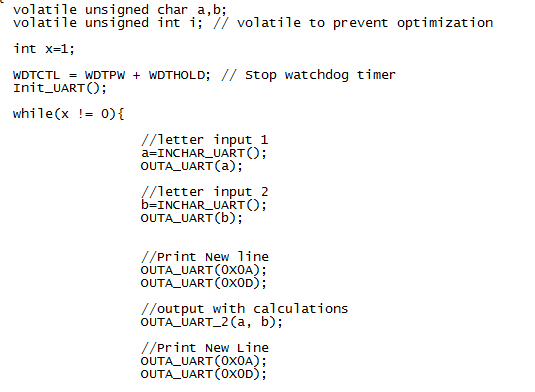
**Test Plan:**

In order to test our code we would be using the HyperTerminal and the UART debugger that comes along with the MSP-430FG4618. By running the code we had created in the Code Composer Studio, we would be able to communicate with the hardware through the Terminal and make sure that the input we had was dealt with correctly.

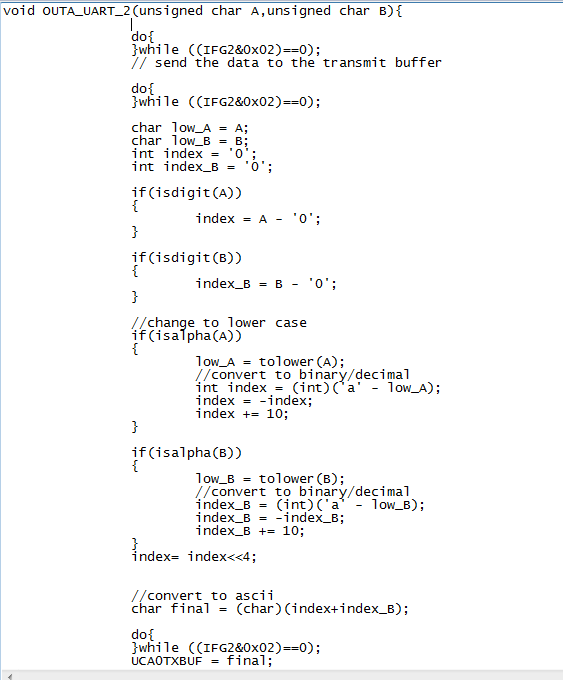
**Source Code:** (We will only be including the primary parts of the codes, if we copy the whole code into here, the report will be too long and we would not want to print too many pages)

**Part One:**

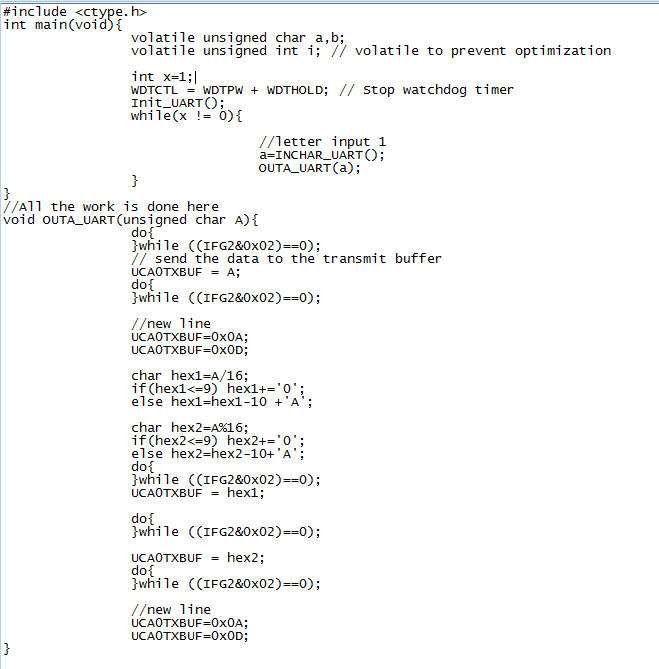
1.1) Main Code



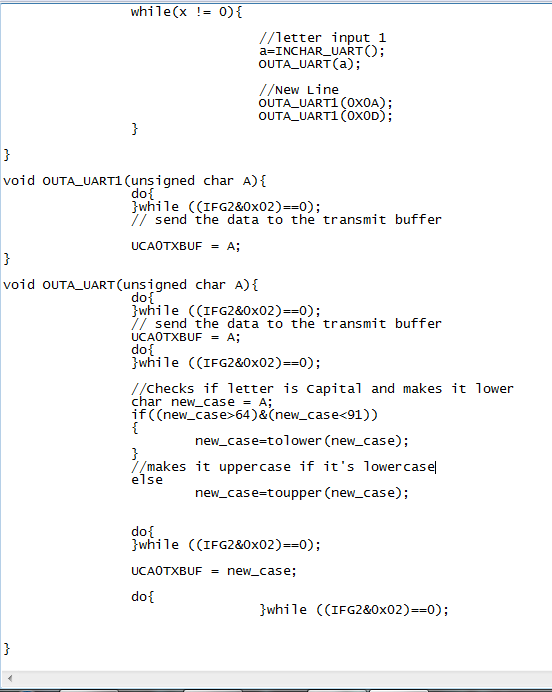
The following is the function that does the work and prints the output characters:



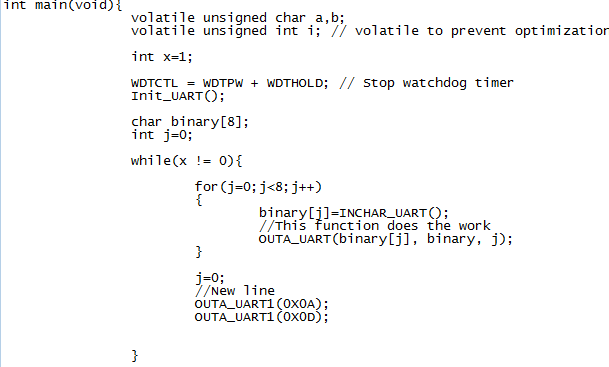
1.2)



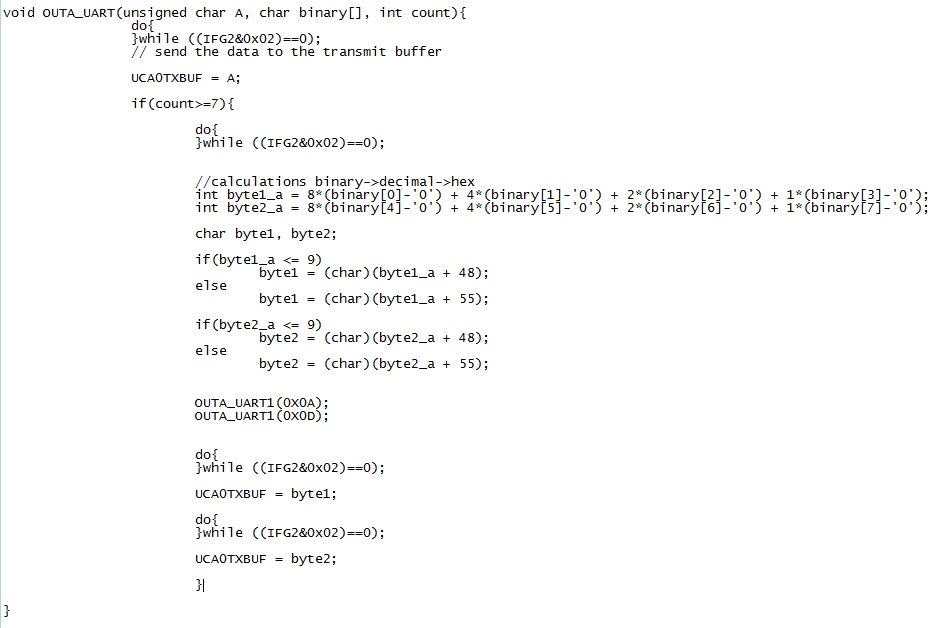
1.3)



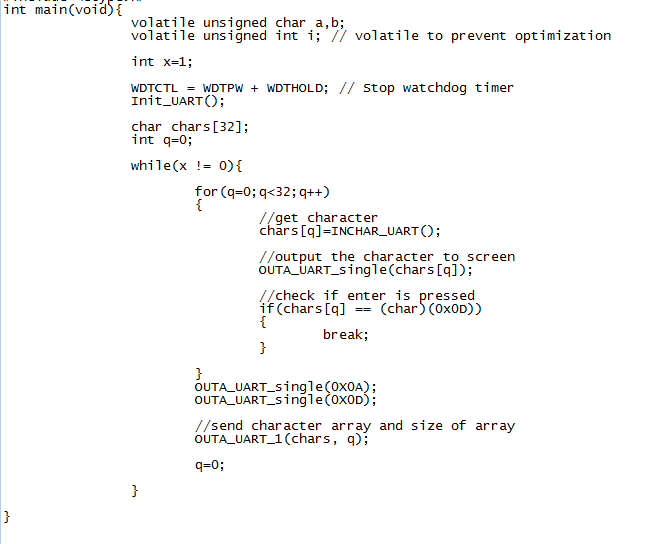
1.4)

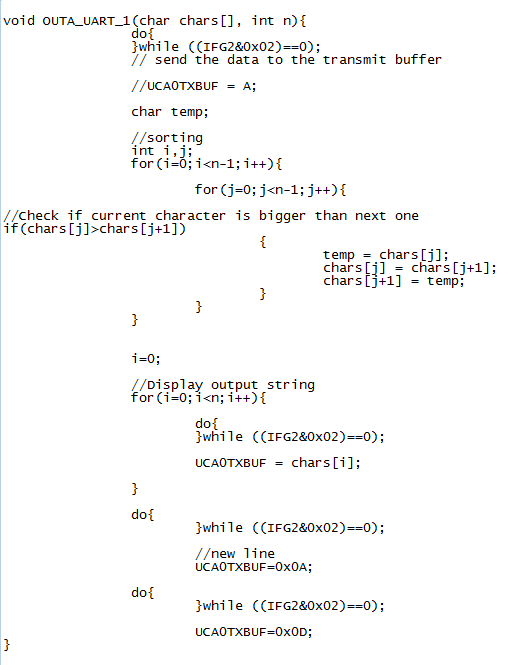
Main:  
 

Function that does the work:



1.5)

Main:  


Function that does the work:  


**Part Two:**

**Conclusion:** This was a challenging experiment that took us a long time to complete, the six hours that were assigned to us to complete the codes turned out to not be enough and we had to go back to complete it in our own time. After completing all of the codes we learned how the hardware operates to convert from one base to another, concept we could use, for example, to convert units in a program. The assembly code was the most challenging to complete, but being able to do it supplied us with a valuable piece of knowledge and a better relationship with the hardware.