CPE 325: Intro to Embedded Computer System

Lab08

UART Serial Communications.

Submitted by: Dan Otieno

Date of Experiment: 03/11/22

Report Deadline: 03/11/22.

Introduction

Write a brief discussion on what the lab is about. (Use the tutorial and write in your own words. DO NOT copy text).

This lab involved writing a program to enable communication between the MSP430 and a HyperTerminal. This also served as an introduction to UART serial interfacing. The screenshots for program testing for this lab are not included in this report, neither is a flowchart. Due to time constraints, the code attached to this report is inconclusive and only reflects the stage reached during work on the assignment. I will complete the code and hopefully successfully demonstrate it in the lab.

Theory

Write short notes on each topic discussed in lab.

Topic 1: **Serial Communication and UART.**

This is the means by which an MSP430-based platform can communicate with another system such as a personal computer, using either the synchronous or asynchronous communication mode. Two devices establish synchronous communication by sharing a common clock. The internal divider (determined by dividing the clock by the baud rate) and modulation register in the MSP430 are initialed to configure the MSP430 in UART mode. The modulation register accounts for the fraction part of that division. As stated, an MSP430 can be connected to a PC using a HyperTerminal application (e.g. MobaXterm) in Windows.

Topic 2: **UAH Serial App.**

a) The UAH serial app was developed by a former student to send and view other types of data, besides character variables, by translating data packets sent into it. Once the application translates the data, it graphically presents the data versus time graph. A packet is a set of predetermined bytes that we can instruct the software application how to interpret. As stated in the tutorial for this lab, "The software must be told how many bytes of information to expect as well as the type and number of data was sending and how it's ordered. To send the data from the MSP430, we first send our header byte followed by our data that has been broken up into 1-byte chunks. The USCI UART buffer will then be fed each byte at a time. It is important in this process to ensure that the packet that you are sending has the same structure that the receiving device is expecting."

Results & Observation

Program 1:

Program Description:

Explain your approach in solving the problem.

Program One, when completed and debugged, displays a message in the HyperTerminal and waits for the user to input their response. The response needs to match the phrase “Let’s go!” and if the user does not input that phrase, the program keeps looping into the welcome message until that phrase is typed in. Once done, the program displays a string for the user to retype, if the user types a wrong character, the program counts the incorrect characters and displays the total in the HyperTerminal. If user retypes sentence correctly, they are provided with a second string, and finally the sentence “Correct” is displayed if they match all characters. This is done by writing various UART functions and using buffers to get characters. These functions are called in main. As of the time of this report, my program is not completed and does not provide the desired outputs.

Program Output:

WILL BE DEMONSTRATED IN THE LAB.

Program Flowchart:

A FLOWCHART WILL BE COMPLETED WHEN SOURCE CODE IS DONE.

Figure ##: Lorem ipsum dolor sit amet

Program 2:

Program Description:

Explain your approach in solving the problem.

This problem requested writing a program that generates a triangular wave and displays it in the UAH serial app. As of the time of this report, the second part of the lab is not yet completed, my intention is to complete it and demo in the next lab class.

Program Output:

WILL BE DEMONSTRATED IN LAB.

Program Flowchart:

SAME AS #1

Conclusion

This lab was important for experimenting with Serial communication and understanding how the MSP430 interacts with a PC or personal device.

Appendix

Your first code goes here, if any. Make sure you use a 1X1 table for this.

(Note: Make sure the code is readable, have comments. Also reduce spacing between lines to avoid lengthy reports.

Table 01: Program one source code: Inconclusive.

```
#include <msp430xG46x.h>
#include <stdio.h>
#include <string.h>

char* wlcm = "\033[1;33mWelcome to typing practice! Are you ready to
start?";
char go[25];
char* StrOne = "The quick brown fox jumps over the lazy dog.";
char* StrTwo = "This is a second string.";

void UART_setup();
void UART_Send_Character(char);
void UART_send_String(char*);
void UART_get_word(char*, int);
void Incorrect(char*, char*, int);

void main(void)
{
      WDTCTL = WDTPW | WDTHOLD;     // stop watchdog timer
      UART_setup();

      int compare, comparetwo, comparethree, limit1 = 10, limit2 = 45,
limit3 = 30;
      char* respone, resptwo;

      while(1)
      {
          UART_send_String(wlcm);
          UART_get_word(go, limit1);
          UART_send_String("\r\n");         // Send carriage, new line.

          compare = strcmp("Let's go!", go);

          while(compare != 0)
          {
              UART_send_String(wlcm);
              UART_get_word(go, limit1);
              strcmp("Let's go!", go);
          }

          UART_send_String("\r\n");
          if(compare = 0)
```

```
            {
                UART_send_String(StrOne);
                UART_get_word(respone, limit2);
                comparetwo = strcmp(respone, StrOne);

                if(comparetwo != 0)
                {
                    UART_send_String("\033[0;32mInCorrect!");
                }
            }
            UART_send_String("\r\n");

            if(comparetwo = 0)
            {
                UART_send_String(StrTwo);
                UART_get_word(resptwo, limit3);      // Get user's entries.
                comparethree = strcmp(resptwo, StrTwo);

                if(comparethree != 0)
                {
                    UART_send_String("\033[0;32mInCorrect!");
                }
                else
                {
                    UART_send_String("\033[0;32mCorrect!");
                    //new_line();
                }
            }
        }
}

void UART_setup(void)
{
    P2SEL |= BIT4 + BIT5;   // Set USCI_A0 RXD/TXD to receive/transmit data
    UCA0CTL1 |= UCSWRST;    // Set software reset during initialization
    UCA0CTL0 = 0;           // USCI_A0 control register
    UCA0CTL1 |= UCSSEL_2;   // Clock source SMCLK

    UCA0BR0 = 0x12;         // 1048576 Hz  / 57600 lower byte
    UCA0BR1 = 0;          // upper byte
    UCA0MCTL = 0x02;        // Modulation (UCBRS0=0x01, UCOS16=0)

    UCA0CTL1 &= ~UCSWRST;   // Clear software reset to initialize USCI
state

}

void UART_Send_Character(char my_char)
{
    while(!(IFG2&UCA0TXIFG));   // Wait until can transmit.
```

```
    UCA0TXBUF = my_char;        // Tx Buffer gets my_char variable.
}

void UART_send_String(char* string)
{
    int i = 0;

    while(string[i] != '\0')  // Send character by character
    {
        UART_Send_Character(string[i]);
        i++;
    }
    UART_Send_Character('\r');
    UART_Send_Character('\n');
}

void UART_get_word(char* buffer, int limit)
{
    int counter = 0;
    char my_char = 0;
    while(my_char != '\r' && counter < limit)
    {
        my_char = UCA0RXBUF;
        while(!(IFG2&UCA0RXIFG));         // Wait until character can be
read from RXBUF.
        UART_Send_Character(my_char);
        *buffer = my_char;
        buffer++;
        counter++;
    }
    *buffer = '\0';
}

void Incorrect(char* buffer, char* msg, int length)
{
    int count = 0, i;

    for(i=0; i<length; i++)
    {
        if(buffer[i] != msg[i])
            count++;
    }

    return count;
    //snprintf(buffstring, 15, "d%", numcount);
}
```

Your next code goes here, if any.

Table 02: Program 2 source code

```
Will be demonstrated in lab.
```