Embedded Systems Lab

CPE 325-02

Introduction to Serial Communication

By: David Thornton

Lab Date: October 20, 2020

Lab Due: October 28, 2020

Demonstration Due: October 28, 2020

Introduction

This lab introduces serial communication and UART with the MSP430.

Theory

Topic 1: Serial Communication and UART

- “[UART] is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. The electric signaling levels and methods are handled by a driver circuit external to the UART. A UART is usually an individual (or part of an) integrated circuit (IC) used for serial communications over a computer or peripheral device serial port. One or more UART peripherals are commonly integrated in microcontroller chips. A related device, the universal synchronous and asynchronous receiver-transmitter (USART) also supports synchronous operation.” - Source

Topic 2: UAH Serial App

- “The UAHuntsville Serial Port Application (SPA) is a Windows application for plotting and logging data received via a PC’s serial COM port. It is typically used to plot and record data coming from an embedded platform to the PC using a serial port. The Serial Port Application can be configured to receive data packets with a varying number of channels (each channel is plotted as a line on the graph), with a configurable type of data (e.g., uint8, uint16, uint32, etc). In addition to the real-time plotting and recording, the SPA supports loading of pre-recorded data from a file and plotting them on a graph.” - Source

Lab Assignment

1. Write and test a program in C that displays the following message in HyperTerminal (You may use Putty, realterm, MobaXterm etc).
   “Please enter the username:”
   The program should then wait for the user to enter user-name. Your program should display the following message after the user-name is entered.
   “Enter the password:”

After the user is done entering password (you can check if the user pressed “Enter” key to determine the end of user-name and password), you should display following messages:

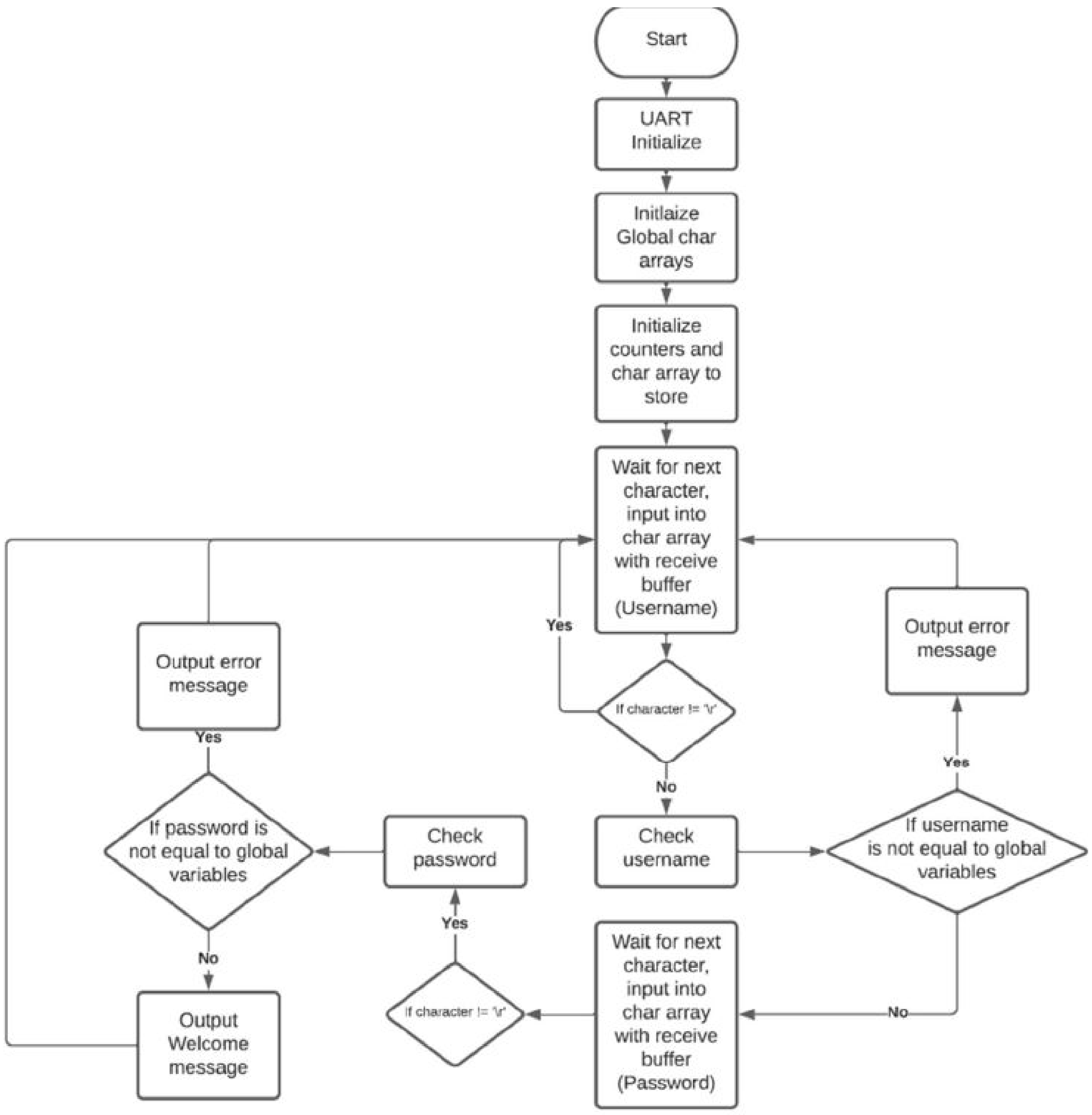
“Welcome to CPE 325!!!” if the combination of username and password is correct
“Incorrect username or password!!!” if the username or password is incorrect.

After any of the above outputs, you should go back to displaying the initial message prompting for user-name. Hint: Your program must have at least two predefined usernames and passwords in your code. Instructors may make appropriate deductions in the points awarded if the solutions to compare the user-input password and stored password are deemed not robust.

2. Update your code in Q1 above to use a timer to timeout the input of passwords. [As soon as the user enters the username, you can enable a timer (may be watchdog timer or timer A), and if certain duration passes set a flag that displays the original prompt to enter the user name. This should automatically disable the user to input password]
3. Extra 5 points will be awarded if you echo * when the user types to enter the password. i.e. consider the user’s password is “quiz”. As the user types ‘q’, an * should be displayed in HyperTerminal screen, as the user types ‘u’, another * should be displayed and so on. Your goal is that the password should not be revealed to any onlookers.

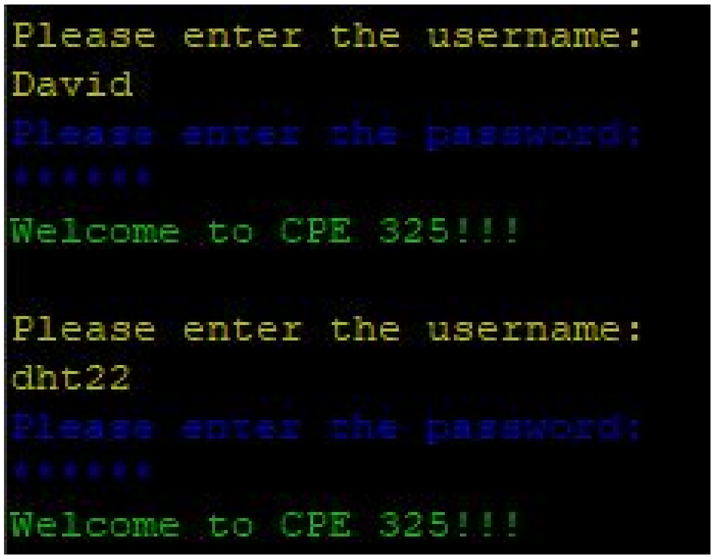
4. Extra 5 points will be awarded if you print the text in colors. For example: The message “Welcome to CPE325!!!” can be printed in green text while the message “Incorrect username or password” can be printed in red text. The prompt to enter user-name and password can be printed in blue, etc. You are free to make your own choice, but use of multiple colors will be given more value.
5. Extra 10 points will be awarded to students who integrate following capabilities:
    a. Change the password of a current username. Hint: You may need to create a simple menu offering for a password change after displaying the message “Welcome to CPE325!!!”. [5 points]
    b. Add a new user to the list of users. Hint: You may need to add another option to add a new user. Make sure you take as input a new password for this new user and test if these new credentials work for a new login. [5 points]

Flow Chart


Start
UART Initialize
Initlaize Global char arrays
Initialize counters and char array to store
Wait for next character, input into char array with receive buffer (Username)
If character != '\r'
Yes
No
Check username
If username is not equal to global variables
Yes
Output error message
No
Wait for next character, input into char array with receive buffer (Password)
If character != '\r'
Yes
Check password
If password is not equal to global variables
Yes
Output error message
No
Output Welcome message

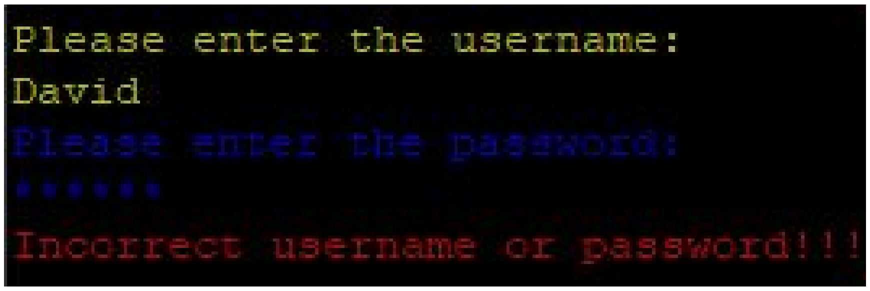
Observations

Question 1 works correctly. I could not get the timer (question 2) working. I was successful in getting bonus questions 3 and 4 working. I did not attempt bonus question 5. The code as it stands now is not very strong, and has several weaknesses. Perhaps more time with this assignment would benefit students in the future, as this was the lab I have found most enjoyable thus far, despite being the most difficult.



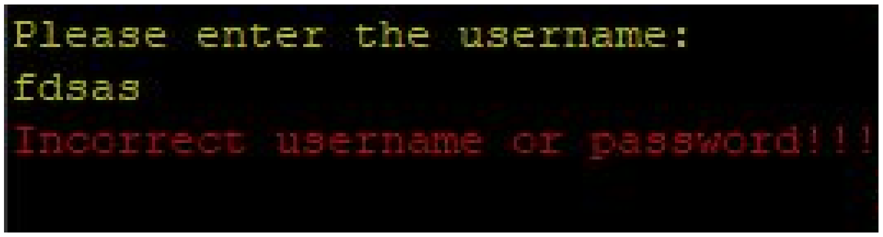
```
Please enter the username:
David
Please enter the password:
******
Welcome to CPE 325!!!

Please enter the username:
dht22
Please enter the password:
******
Welcome to CPE 325!!!
```

Figure 1: Correct input of both usernames and passwords



```
Please enter the username:
David
Please enter the password:
******
Incorrect username or password!!!
```

Figure 2: Correct input of a username, incorrect input of a password



```
Please enter the username:
fdsas
Incorrect username or password!!!
```

Figure 3: Incorrect input of a username

Conclusion

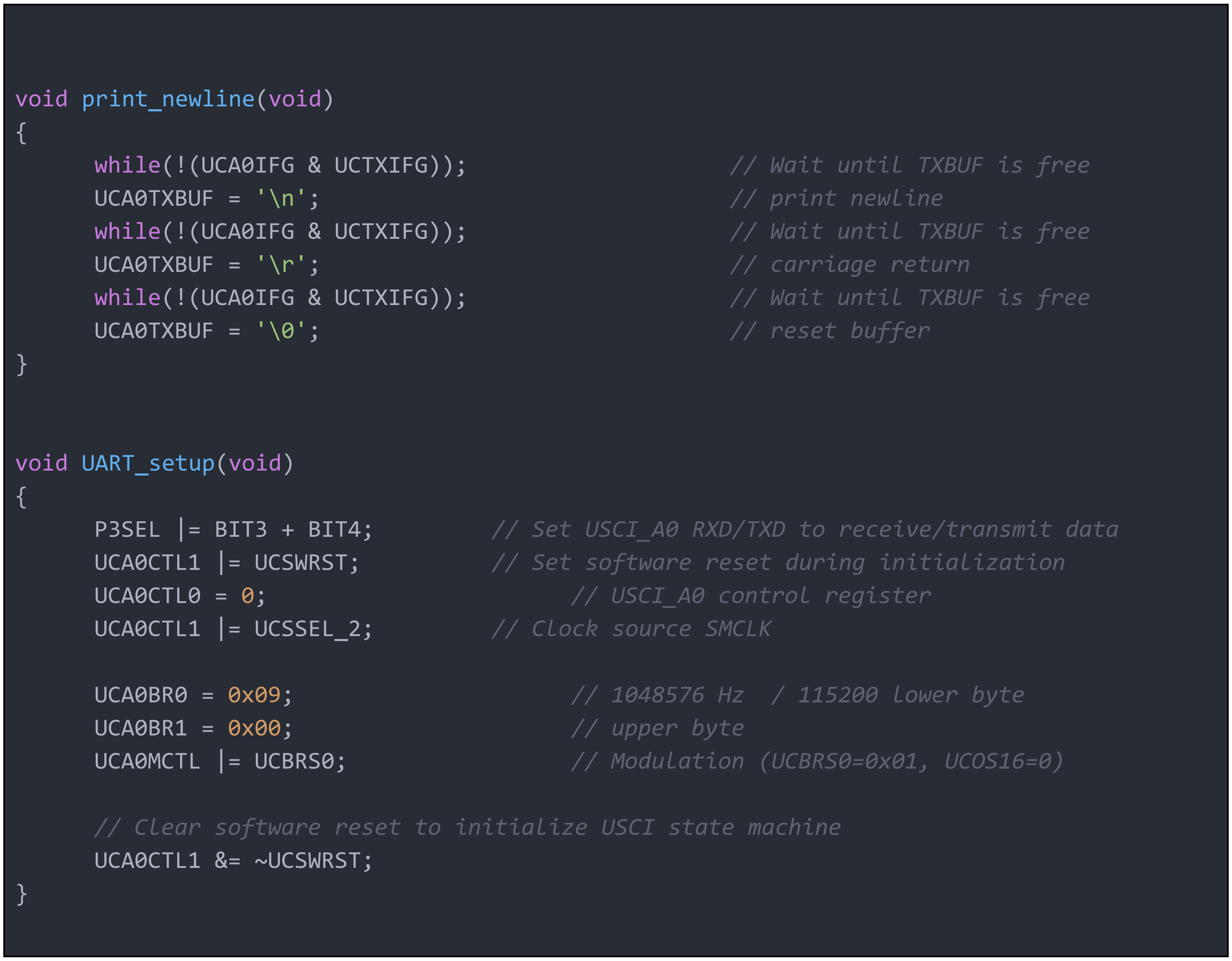
This lab was successful in introducing me to using serial communication with the MSP430.
Demo link

Appendix

Appendix 1: Lab_8.c

```c
/*---------------------------------------------------------
* File:           Lab_8.c
* Description:    This program user UART communication to
*                       test usernames and passwords. This program
*                       uses a baud rate of 115200.
* Input:          Typing into hyperterminal
* Output:         Hyperterminal text
* Author:         David Thornton
* Lab Section:    2
* Date:           October 28, 2020
* *-------------------------------------------------------*/
#include <stdio.h>
#include <msp430.h>

char username_prompt[35] = "\033[33mPlease enter the username: ";
char password_prompt[35] = "\033[34mPlease enter the password: ";
char incorrect_login[41] = "\033[31mIncorrect username or password!!!";
char login_success[41] = "\033[32mWelcome to CPE 325!!!";

char username_1[5] = "David";
char username_2[5] = "dht22";
char password_1[6] = "David1";
char password_2[6] = "abc123";

void print_username_prompt(void);
void print_password_prompt(void);
void print_incorrect_login(void);
void print_login_success(void);
void print_newline(void);
void check_username(char any_user[]);
void check_password(char any_user[], char any_pass[]);

void UART_setup(void);

void main(void)
{
      WDTCTL = WDTPW + WDTHOLD;     // Stop watchdog timer
      UART_setup();                       // Initialize UART
      while(1)
      {
            unsigned int i = 0;
            unsigned int j = 0;
            char username[5];
            char password[6];

            print_username_prompt();
            do
            {
                  if(i >= 5) break;
                  while(!(UCA0IFG & UCRXIFG));  // Wait for a new character
                  while(!(UCA0IFG & UCTXIFG));  // Wait until TXBUF is free
                  UCA0TXBUF = UCA0RXBUF;              // TXBUF <= RXBUF (echo)
                  username[i] = UCA0RXBUF;      // username[i] <- user input
                  i++;
            } while(UCA0RXBUF != '\r');
            check_username(username);

            print_password_prompt();
            do
            {
                  if(j >= 6) break;
                  while(!(UCA0IFG & UCRXIFG));  // Wait for a new character
                  while(!(UCA0IFG & UCTXIFG));  // Wait until TXBUF is free
                  UCA0TXBUF = '*';                    // TXBUF <= * (echo)
                  password[j] = UCA0RXBUF;      // password[j] <- user input
                  j++;
                  // if timeout flag is set, restart
                  // if timeout flag is not set, keep going
            } while(UCA0RXBUF != '\r');
            check_password(username, password);
      }
}


void print_username_prompt(void)
{
      print_newline();
      unsigned int i = 0;
      for(i = 0; i <= 35; i++)
      {
            while (!(UCA0IFG & UCTXIFG));             // Wait for previous character to
transmit
            UCA0TXBUF = username_prompt[i];           // Transmit a byte
      }
      print_newline();
}


void print_password_prompt(void)
{
      print_newline();
      unsigned int i = 0;
      for(i = 0; i <= 35; i++)
      {
            while (!(UCA0IFG & UCTXIFG));             // Wait for previous character to
transmit
            UCA0TXBUF = password_prompt[i];           // Transmit a byte
      }
      print_newline();
}


void print_incorrect_login(void)
{
      print_newline();
      unsigned int i = 0;
      for(i = 0; i <= 41; i++)
      {
            while (!(UCA0IFG & UCTXIFG));             // Wait until TXBUF is free
            UCA0TXBUF = incorrect_login[i];           // Transmit a byte
      }
      print_newline();
}


void print_login_success(void)
{
      print_newline();
      unsigned int i = 0;
      for(i = 0; i <= 41; i++)
      {
            while (!(UCA0IFG & UCTXIFG));             // Wait for previous character to
transmit
            UCA0TXBUF = login_success[i];       // Transmit a byte
      }
```

```c
      print_newline();
}


void check_username(char any_user[])
{
      unsigned int i = 0;
      for (i = 0; i < 5; i++)
      {
            if ((any_user[i] != username_1[i]) && (any_user[i] != username_2[i]))
            {
                  print_incorrect_login();
                  main();
            }
      }
      while(!(UCA0IFG & UCTXIFG));                    // Wait until TXBUF is free
}


void check_password(char any_user[], char any_pass[])
{
      unsigned int i = 0;
      for (i = 0; i < 6; i++)
      {
            if ((any_pass[i] != password_1[i]) && (any_pass[i] != password_2[i]))
            {
                  print_incorrect_login();
                  main();
            }
            else if((any_user[i] == username_1[i]) && (any_pass[i] == password_2[i]))
            {
                  print_incorrect_login();
                  main();
            }
            else if((any_user[i] == username_2[i]) && (any_pass[i] == password_1[i]))
            {
                  print_incorrect_login();
                  main();
            }
      }
      print_login_success();
}
```



```
void print_newline(void)
{
      while(!(UCA0IFG & UCTXIFG));                      // Wait until TXBUF is free
      UCA0TXBUF = '\n';                                 // print newline
      while(!(UCA0IFG & UCTXIFG));                      // Wait until TXBUF is free
      UCA0TXBUF = '\r';                                 // carriage return
      while(!(UCA0IFG & UCTXIFG));                      // Wait until TXBUF is free
      UCA0TXBUF = '\0';                                 // reset buffer
}


void UART_setup(void)
{
      P3SEL |= BIT3 + BIT4;         // 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 = 0x09;                     // 1048576 Hz  / 115200 lower byte
      UCA0BR1 = 0x00;                     // upper byte
      UCA0MCTL |= UCBRS0;                 // Modulation (UCBRS0=0x01, UCOS16=0)

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