CPE 325: Intro to Embedded Computer System

Lab08

UART Serial Communications

Submitted by: Nolan Anderson

Date of Experiment: 10/28/2020

Report Deadline: 10/28/2020

Introduction

This lab covers communication protocols on the MSP430 which includes configuring the UART parameters, baud rates, input and output through serial communications in a console, and parameter checking.

Theory

Serial Communication and UART :

Serial communication is essentially communication with another system, in this case our personal PC's. You can communicate either synchronously or asynchronously and they must share a common clock source. This lab does it asynchronously. Speed of communication between devices should also match.

UART (universal asynchronous receiver transmitter): This enables serial communication between MSP430 and another decide. You must set up uart to communicate with your PC at the same baud rate:

```
177 void UART_setup(void)
178 {
179     P3SEL |= BIT3 + BIT4;       // Set USCI_A0 RXD/TXD to receive/transmit data
180     UCA0CTL1 |= UCSWRST;        // Set software reset during initialization
181     UCA0CTL0 = 0;               // USCI_A0 control register
182     UCA0CTL1 |= UCSSEL_2;       // Clock source SMCLK
183     UCA0BR0 = 0x09;             // 1048576 Hz  / 115200 lower byte
184     UCA0BR1 = 0x00;             // upper byte
185     UCA0MCTL |= UCBRS0;         // Modulation (UCBRS0=0x01, UCOS16=0)
186     UCA0CTL1 &= ~UCSWRST;       // Clear software reset to initialize USCI state machine
187 }
```

UAH Serial App:

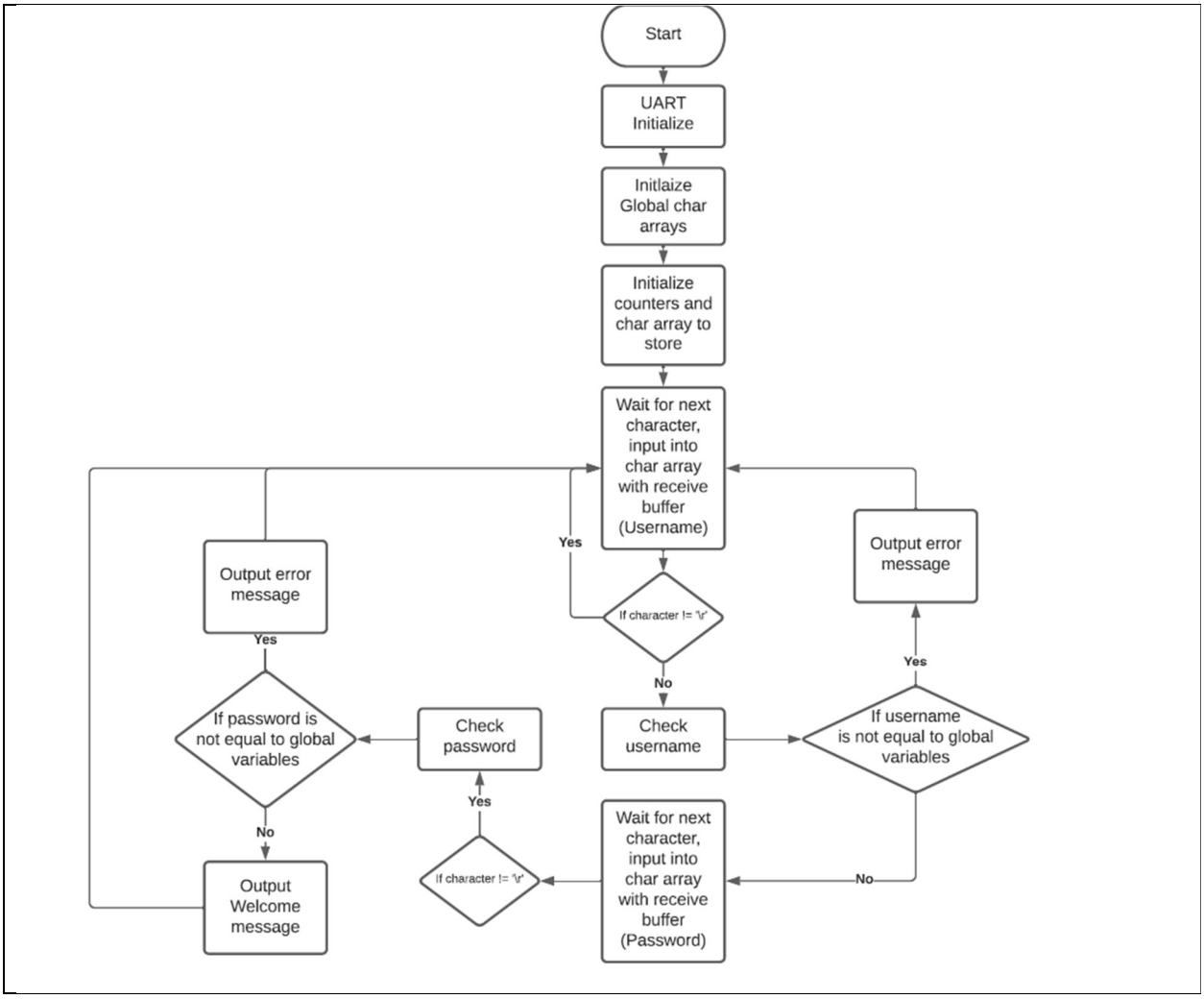
This serial app translates serials packets that are sent to it. It graphically represents the data versus time. The serial app takes in packets that consist of predetermined bytes. It expires a packet that has a 1-byte header followed by the data followed by an optional checksum.

Results & Observation

1. In a single demonstration, show the full operation of Q1 and Q2 above.

In demonstration. Note that I did not get #2 to work, but I did leave my code at the bottom of Appendix 1 of my attempt. I simply could not figure it out.

Flow Charts:



Results Screenshots/Pictures:

Observations:

Serial communication is extremely difficult and this was by far the hardest lab I have done. Serial communication is extremely useful but extremely hard to get working correctly. After finishing the lab, it does make a lot of sense but figuring it out initially is difficult.

Conclusion:

UART Communication is very useful but extremely cumbersome to get correct output and data correct.

Video Link:

https://drive.google.com/file/d/1GcLRE6uhtOQrk5ZG6vRCFKn7cX7K58F-/view?usp=sharing

Appendix

Appendix 1

```
/*-------------------------------------------------------------------------------
* Student:      Nolan Anderson
* Program:      Lab_8_1.c
* Date:         Oct 28, 2020
* Input:        Input through console
* Output:       Outputs information about username and password and checks it
* Description:  This file uses UART Serial Communication to check usernames
*               and passwords.
*-------------------------------------------------------------------------------*/
#include <stdio.h>
#include <msp430.h>
// ***** FUNCTION DECLARATION ***** //
void buffers(void);                 // Function for correct line spacing dellimeters.
void user_prompt(void);             // Outputs the username prompt
void pass_prompt(void);             // Outputs the password prompt
void error_prompt(void);            // Outputs the error prompt
void welcome_prompt(void);          // Outputs the welcome prompt
void user_check(char un[]);         // Username check
void pass_check(char un[], char pw[]); // Password check against username
void UART_setup(void);              // UART Communication setup

// ***** VARIABLES ***** //
char u1[5] = "npa00";       // Username 1
char p1[6] = "2129ke";      // Password 1
char u2[5] = "Nolan";       // Username 2
char p2[6] = "Nolan1";      // Password 2

// ***** PROMPTS ***** //
char user_msg[35] = "\033[33mPlease enter the username: ";
char pass_msg[35] = "\033[35mPlease enter the password: ";
char error_msg[41] = "\033[31mIncorrect username or password!!!";
char login_success[41] = "\033[32mWelcome to CPE 325!!!";
char timeout[44] = "You did not enter a password quickly enough.";

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

        user_prompt();      // Output the user prompt
        do
        {
            if(i >= 6) break;           // If username is greater than 5 characters..
            while(!(UCA0IFG & UCRXIFG)); // Wait for a new character
            while(!(UCA0IFG & UCTXIFG)); // Wait until TXBUF is free
            UCA0TXBUF = UCA0RXBUF;      // TXBUF <= RXBUF (echo)
            user[i] = UCA0RXBUF;        // user(i) = receive character.
            i++;
        } while(UCA0RXBUF != '\r');     // While the character is not return line
        user_check(user);               // Check the username

        pass_prompt();                  // Output the password prompt.
        do
        {
            if(j >= 7) break;
            while(!(UCA0IFG & UCRXIFG)); // Wait for a new character
            while(!(UCA0IFG & UCTXIFG)); // Wait until TXBUF is free
            UCA0TXBUF = '*';            // TXBUF <= RXBUF (echo)
            pass[j] = UCA0RXBUF;        // pass(j) = receive character.
            j++;
        } while(UCA0RXBUF != '\r');     // While the character is not the return line.
        pass_check(user, pass);         // Check the password against username.
    }
}
```

```
// ================================WELCOME PROMPT======================================== //

void welcome_prompt(void)
{
    buffers();
    int i = 0;
    for(i = 0; i <= 27; i++)
    {
        while (!(UCA0IFG & UCTXIFG));   // Wait for previous character to transmit
        UCA0TXBUF = login_success[i];   // Transmit a byte
    }
    buffers();
}

// ================================USERNAME PROMPT======================================== //

void user_prompt(void)
{
    buffers();
    int i = 0;
    for(i = 0; i <= 33; i++)
    {
        while (!(UCA0IFG & UCTXIFG));   // Wait for previous character to transmit
        UCA0TXBUF = user_msg[i];        // Transmit a byte
    }
    buffers();
}

// ================================PASSWORD PROMPT======================================== //

void pass_prompt(void)
{
    buffers();
    int i = 0;
    for(i = 0; i <= 33; i++)
    {
        while (!(UCA0IFG & UCTXIFG));   // Wait for previous character to transmit
        UCA0TXBUF = pass_msg[i];        // Transmit a byte
    }
    buffers();
}

// ================================ERROR PROMPT======================================== //

void error_prompt(void)
{
    buffers();
    int i = 0;
    for(i = 0; i <= 39; i++)
    {
        while (!(UCA0IFG & UCTXIFG));   // Wait until TXBUF is free
        UCA0TXBUF = error_msg[i];       // Transmit a byte
    }
    buffers();
}

// ================================USERNAME CHECK======================================== //

void user_check(char un[])
{
    int i = 0;
    for (i = 0; i < 5; i++)
    {
        if ((un[i] != u1[i]) && (un[i] != u2[i]))       // If the usernames dont match
        {
            error_prompt();                             // Output the error prompt
            main();
        }
    }
    while(!(UCA0IFG & UCTXIFG));  // Wait until TXBUF is free
}

// ================================PASSWORD CHECK======================================== //

void pass_check(char un[], char pw[])
{
    int i = 0;
    for (i = 0; i < 6; i++)
```

```
    {
        if ((pw[i] != p1[i]) && (pw[i] != p2[i]))        // If the passwords don't match
        {
            error_prompt();                              // Output the error prompt
            main();
        }
        else if((un[i] == u1[i]) && (pw[i] == p2[i]))    // If username 1 and password 2 are entered, fail.
        {
            error_prompt();
            main();
        }
        else if((un[i] == u2[i]) && (un[i] == p1[i]))  // If username 2 and password 1 are entered, fail.
        {
            error_prompt();
            main();
        }
    }
    welcome_prompt();
}

void buffers(void)
{
    while(!(UCA0IFG & UCTXIFG));  // Wait until TXBUF is free
    UCA0TXBUF = '\n';
    while(!(UCA0IFG & UCTXIFG));  // Wait until TXBUF is free
    UCA0TXBUF = '\r';
    while(!(UCA0IFG & UCTXIFG));  // Wait until TXBUF is free
    UCA0TXBUF = '\0';
}

// ==============================UART SETUP FUNCTION======================================== //

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

// Here I attempted a timer but ran out of time to get it working.
//unsigned int sec = 0;           // Seconds
//unsigned int tsec = 0;          // 1/10 second
//// ==============================TIMER A SETUP======================================== //
//
//void TimerA_setup(void)
//{
//    TA0CTL = TASSEL_2 + MC_1 + ID_3;    // Select SMCLK/8 and up mode
//    TA0CCR0 = 13107;                    // 100ms interval
//    TA0CCTL0 = CCIE;                    // Capture/compare interrupt enable
//}
//// ==============================PASSWORD TIMER======================================== //
//
//void PassTimer(void)
//{
//    while(1)
//    {
//        tsec++;
//        if (tsec == 1)
//        {
//            tsec = 0;
//            sec++;
//            buffers();
//            int i = 0;
//            for(i = 0; i <= 44; i++)
//            {
//                while (!(UCA0IFG & UCTXIFG));   // Wait for previous character to transmit
//                UCA0TXBUF = timeout[i];  // Transmit a byte
//            }
//            buffers();
//        }
//    }
//}
```