CPE 325: Embedded Systems Laboratory

Lab08

UART Serial Communications

Submitted by: Esther Shore

Date of Experiment: July 5, 2023

Report Deadline: July 12, 2023

Demonstration Deadline: July 17, 2023

**Introduction**

This lab is all about UART serial communications and utilizing the UAH Serial App, using the USCI peripheral on the MSP430. It introduces how to transmit and receive data through UART, creating interactive programs to demonstrate how to harness the power of serial communication protocols.

**Theory Topics**

1. Serial Communication and UART

Serial communication involves two devices communicating with each other via transmission line, sending and receiving data bit by bit. UART is a hardware communication protocol that is asynchronous and stands for Universal Asynchronous Receiver/Transmitter. Each byte of data that is sent is framed by a start and stop bit in addition to an optional parity bit. The speed of data transfer is configurable by adjusting the baud rate (bits/sec). This method is often used for serial communication between microcontrollers and sensors or peripherals.

1. UAH Serial App

The UAH Serial App is a software application that allows graphical visualization of data received via UART by translating the received serial packets. This is an advantage over something like the Putty application because it allows better interpretation of different types of data beyond ASCII characters such as floating point numbers.

**Lab Questions**

1. What clock signals are stopped in LPM0?

In LPM0, CPU and MCLK are disabled while SMCLK and ACLK are active.

1. What is the maximum time you can have on the real-time clock (demo #3)?

The format of the time displayed is given as sssss:tsec (5 digits, 1 digit). Since the variable sec is an unsigned int, the length is 4 bytes or 16 bits and the max value it can hold is 2^16 - 1 = 65535. The tsec variable represents tenths of a second and thus the max value is 9. Therefore, the maximum time the real-time clock can show from Demo #3 is 65535:9.

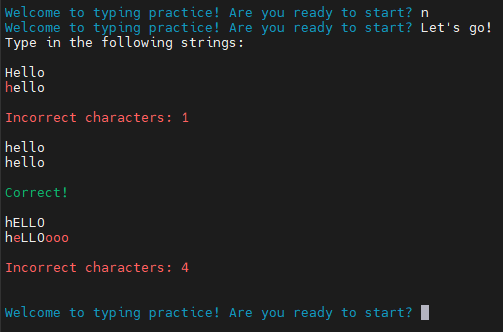
**Program 1 (Typing Practice)**

***Program Description:***

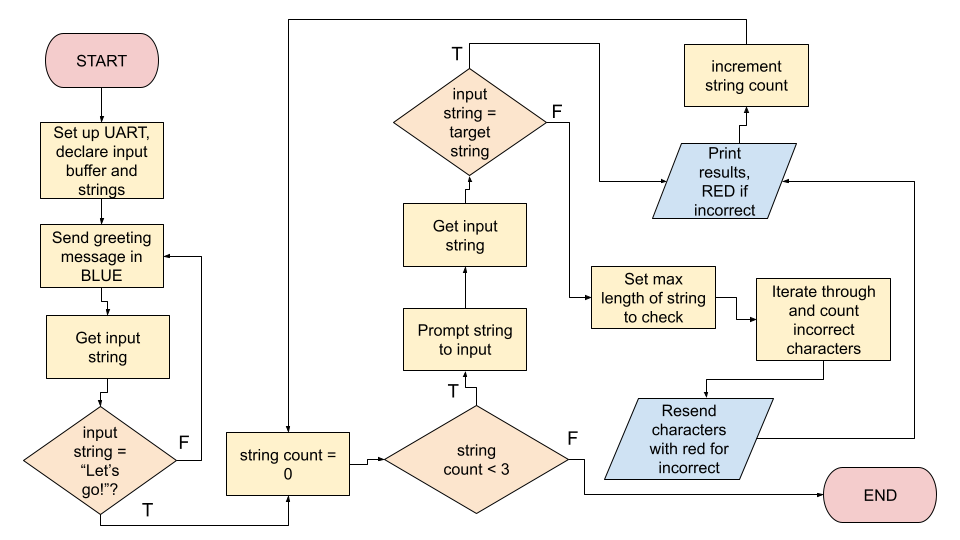
This C program guides the user through a series of three strings for typing practice, displaying the characters in MobaXterm through UART transmission.

The welcome prompt will display in blue and continue to appear until the proper input of “Let’s go” is entered. Then, one at a time, the user will be prompted with a string to replicate as part of the typing test. After entering the attempt, it will display the results of the comparison between the prompted string and the submitted string. If it is incorrect, the incorrect characters will display in red and a count of the number of incorrect characters will also be displayed in red. If it is correct, then a green message saying such will be displayed.

***Program Output:***



***Program Flowchart:***

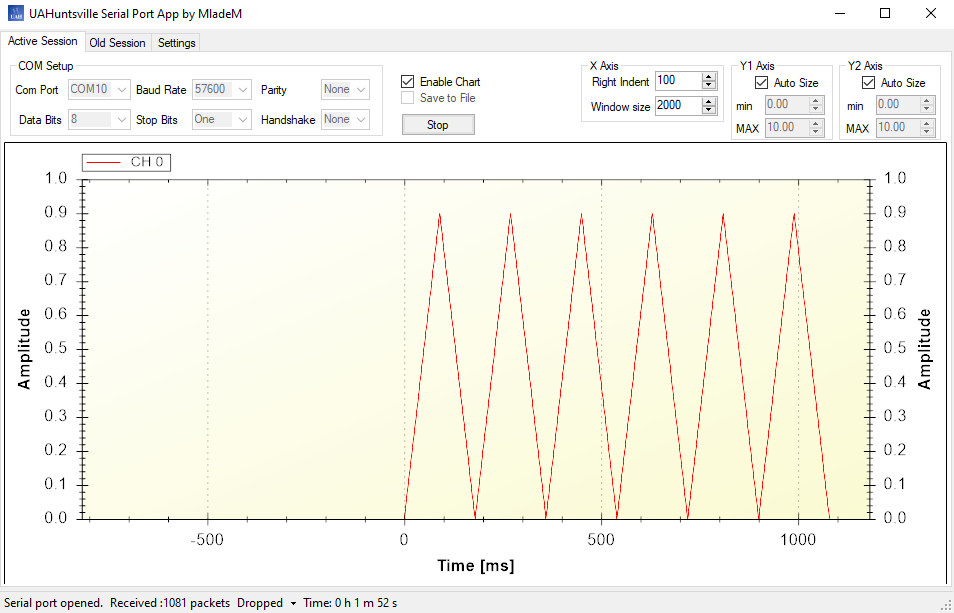
******

**Program 1 (Triangle Wave Generator)**

***Program Description:***

This C program generates a triangular waveform and displays it on the UAH Serial App using a 57600 baud connection. The signal will be transmitted for a duration of 6 signal periods with target amplitude 9 units and frequency 4 Hz. To handle the constrained duration of the signal, a counter was set to stop sending data on the 12th change of direction (or vertex in the graph). The frequency of the signal was altered by changing the number of points, or samples, that constitute one period of the waveform. At a constant sample rate, increasing the number of samples would decrease the frequency while decreasing the number of samples would increase the frequency.

***Program Output:***



**Conclusion**

It was difficult at first understanding how to set up and configure UART to work properly, but after working with it for a while, it started to become easier. The only challenge in the end was figuring out how to adjust the frequency of the generated triangular wave. This was accomplished by adjusting the number of samples to complete one period.

***Appendix:***

**Table 1:** Program 1 Source Code

| /\*------------------------------------------------------------------------------  \* File: Lab08\_P1.c  \* Function: Typing practice  \* Description: Prompts input for three different strings and counts/shows the  \* incorrect characters if there are any  \* Input: Characters entered via MobaXterm received via UART  \* Output: Results of string comparisons sent over UART  \* Author: Esther Shore  \* Date: July 2023  \*----------------------------------------------------------------------------\*/  #include <msp430.h>  #include <stdio.h>  #include <string.h>  // ANSI escape codes for text color  #define COLOR\_RED "\x1b[31m"  #define COLOR\_GREEN "\x1b[32m"  #define COLOR\_BLUE "\x1b[34m"  #define COLOR\_RESET "\x1b[0m"  void UART\_setup();  void UART\_send\_character(char);  void UART\_send\_string(char\* string);  void UART\_get\_word(char\* buffer, int limit);  void main(void) {  WDTCTL = WDTPW + WDTHOLD; // Stop WDT  UART\_setup();  char input[50];  char strings[3][50] = {"Hello", "hello", "hELLO"};  while (1) {  do {  UART\_send\_string(COLOR\_BLUE);  UART\_send\_string("\r\nWelcome to typing practice! Are you ready to start? ");  UART\_send\_string(COLOR\_RESET);  UART\_get\_word(input, sizeof(input));  } while (strcmp(input, "Let's go!") != 0);  UART\_send\_string("\r\nType in the following strings:\r\n\r\n");  int i = 0;  while (i < 3) {  UART\_send\_string(strings[i]);  UART\_send\_string("\r\n");  UART\_get\_word(input, sizeof(input));  if (strcmp(input, strings[i]) == 0) {  UART\_send\_string(COLOR\_GREEN);  UART\_send\_string("\r\n\r\nCorrect!\r\n\r\n");  UART\_send\_string(COLOR\_RESET);  } else {  int incorrect\_chars = 0;  int max\_index;  int j = 0;  if (strlen(strings[i]) > strlen(input)) {  max\_index = strlen(strings[i]) - 1;  } else {  max\_index = strlen(input) - 1;  }  while (j <= max\_index) {  if (strings[i][j] != input[j]) {  incorrect\_chars++;  UART\_send\_string(COLOR\_RED);  UART\_send\_character(input[j]);  UART\_send\_string(COLOR\_RESET);  } else {  UART\_send\_character(input[j]);  }  j++;  }  char result[50];  UART\_send\_string(COLOR\_RED);  snprintf(result, sizeof(result), "\r\n\r\nIncorrect characters: %d\r\n\r\n", incorrect\_chars);  UART\_send\_string(result);  UART\_send\_string(COLOR\_RESET);  }  i++;  }  }  } |
| --- |

**Table 2:** Program 2 Source Code

| /\*------------------------------------------------------------------------------  \* File: Lab08\_P2.c  \* Function: Send floating data to Serial port in form of triangular wave  \* Description: UAH serial app expects lower byte first so send each byte at a  \* time sending Lowest byte first  \* Clocks: ACLK = LFXT1 = 32768Hz, MCLK = SMCLK = default DCO  \*  \* Instructions: Set the following parameters in putty  \* Port: COM10  \* Baud rate: 57600  \* Data bits: 8  \* Parity: None  \* Stop bits: 1  \* Flow Control: None  \*  \* Input: None  \* Output: Triangular wave signal in UAH Serial app  \* Author: Esther Shore  \* Date: July 2023  \*----------------------------------------------------------------------------\*/  #include <msp430.h>  #include <stdint.h>  volatile float myData;  void UART\_setup(void) {  P3SEL |= BIT3+BIT4; // P3.3,4 = USCI\_A0 TXD/RXD  UCA0CTL1 |= UCSWRST; // \*\*Put state machine in reset\*\*  UCA0CTL1 |= UCSSEL\_2; // SMCLK  UCA0BR0 = 18; // 1MHz / 57600 = 18.2(see User's Guide)  UCA0BR1 = 0; // 1MHz 57600  UCA0MCTL |= UCBRS\_2 + UCBRF\_0; // Modulation UCBRSx=1, UCBRFx=0  UCA0CTL1 &= ~UCSWRST; // \*\*Initialize USCI state machine\*\*  }  void UART\_putCharacter(char c) {  while (!(UCA0IFG&UCTXIFG)); // USCI\_A0 TX buffer ready? Wait for previous character to transmit  UCA0TXBUF = c; // Put character into tx buffer  }  int main() {  WDTCTL = WDT\_MDLY\_32;  UART\_setup(); // Initialize USCI\_A0 module in UART mode  SFRIE1 |= WDTIE;  myData = 0.0;  \_\_bis\_SR\_register(LPM0\_bits + GIE);  }  // Sends a ramp signal; amplitude of one period ranges from 0.0 to 9.9  #pragma vector = WDT\_VECTOR  \_\_interrupt void watchdog\_timer(void) {  static int counter = 0;  if (counter < 12) {  static int multiplier = 0;  static int sign = 1;  multiplier += sign;  if (multiplier == 0 || multiplier == 90) {  sign = -sign;  counter++;  }  char index = 0;  // Use character pointers to send one byte at a time  char \*myPointer = (char\* )&myData;  UART\_putCharacter(0x55); // Send header  for(index = 0; index < 4; index++) { // Send 4-bytes of myData  UART\_putCharacter(myPointer[index]);  }  // Update myData for next transmission  myData = multiplier \* 0.01;  } else if (counter == 12) {  counter++;  char index = 0;  // Use character pointers to send one byte at a time  char \*myPointer = (char\* )&myData;  UART\_putCharacter(0x55); // Send header  for(index = 0; index < 4; index++) { // Send 4-bytes of myData  UART\_putCharacter(myPointer[index]);  }  }  }  void UART\_setup() {  P3SEL |= BIT3+BIT4; // P3.3,4 = USCI\_A0 TXD/RXD  UCA0CTL1 |= UCSWRST; // \*\*Put state machine in reset\*\*  UCA0CTL1 |= UCSSEL\_2; // SMCLK  UCA0BR0 = 18; // 1MHz / 57600 = 18.2(see User's Guide)  UCA0BR1 = 0; // 1MHz 57600  UCA0MCTL |= UCBRS\_2 + UCBRF\_0; // Modulation UCBRSx=1, UCBRFx=0  UCA0CTL1 &= ~UCSWRST; // \*\*Initialize USCI state machine\*\*  UCA0IE |= UCRXIE; // Enable USCI\_A0 RX interrupt  }  void UART\_send\_character(char my\_char) {  while (!(UCA0IFG&UCTXIFG)); // USCI\_A0 TX buffer ready?  UCA0TXBUF = my\_char; // TX -> RXed character  }  void UART\_send\_string(char\* string) {  int i;  for (i = 0; i < strlen(string); i++) {  UART\_send\_character(string[i]);  }  }  void UART\_get\_word(char\* buffer, int limit)  {  int i = 0;  char received\_char;  while (i < limit - 1) {  while (!(UCA0IFG & UCRXIFG)); // USCI\_A0 RX buffer ready?  received\_char = UCA0RXBUF; // RX -> Get received character  UART\_send\_character(received\_char);  if (received\_char == '\r') {  break;  }  buffer[i] = received\_char;  i++;  }  buffer[i] = '\0'; // Terminate the string with null character  } |
| --- |