CPE 325: Intro to Embedded Computer System

Lab 08

UART Communication

Submitted by: Anshika Sinha

Date of Experiment:_____February 26, 2025_____

Report Deadline:_____March 4, 2025_____

Demonstration Deadline: ___March 17, 2025_____

Theory

Topic 1: Serial Communication and UART

a) Serial communication is a method of transferring data between two devices one bit at a time over a single communication line.

b) The MSP430 microcontroller supports both synchronous and asynchronous communication.

c) This lab focuses on asynchronous communication via UART (Universal Asynchronous Receiver-Transmitter).

d) Baud Rate Calculation: The system clock (1,048,576 Hz) is divided by the desired baud rate, with a modulation register adjusting for fractional values to minimize transmission errors.

e) The lab includes both polling and interrupt based communication. Using interrupts allows the MSP430 to enter low-power mode until data is received, increasing efficiency.

Topic 2: UAH Serial App

a) The UAH Serial Port Application (SPA) is a Windows-based tool for plotting and logging data received via a PC's serial COM port.
b) It is mainly used for visualizing and recording data from embedded systems connected via RS232 communication.
c) The UAH Serial App allows customizable data packet reception, supporting multiple channels, where each channel is plotted as a separate line.
d) It supports various data types such as uint8, uint16, and uint32 for flexibility in data representation.
e) Besides real-time plotting, SPA can load and display pre-recorded data from a file for further analysis.

Results & Observation

Program 1:

Program Description:

This C program implements MedBot, an interactive chatbot for prescribing medication via UART communication on an MSP430 microcontroller. It prompts the user for a patient's name and symptoms, offering prescriptions for headache, cough, or allergy. The bot calculates the total cost based on selected medications and dosages, ensuring valid input. Communication is handled via UART functions for sending and receiving data. The program continuously loops, allowing multiple users to interact with MedBot.

Program Flowchart:

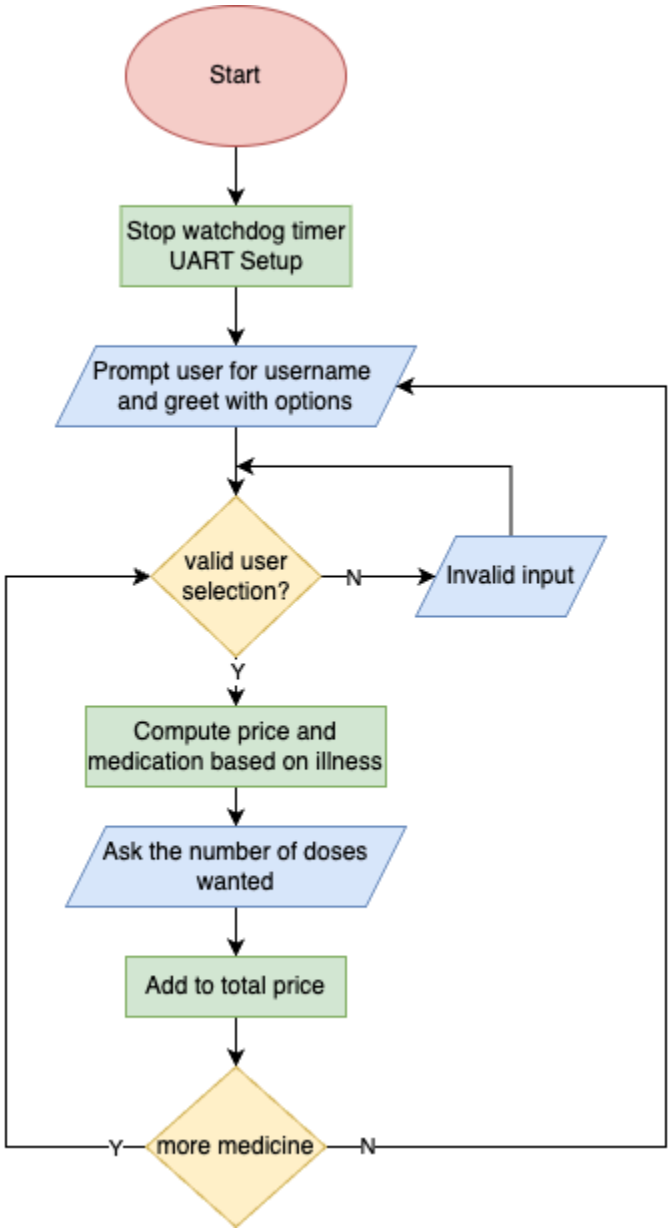

Figure 02: Program 1 Flowchart

Program 2:

Program Description:

This C program generates a triangular wave with an amplitude of 8 units and a frequency of 2.5 Hz. The waveform is transmitted via UART at 57,600 baud to the UAH Serial app for four full periods. The program uses Timer_A to generate and send the wave in 16 discrete steps per cycle. The UART module (USCI_A1) handles serial communication, while the Timer_A interrupt service routine (ISR) sends

waveform values at precise intervals. After four periods, the timer stops, completing the signal transmission.

Program Output:

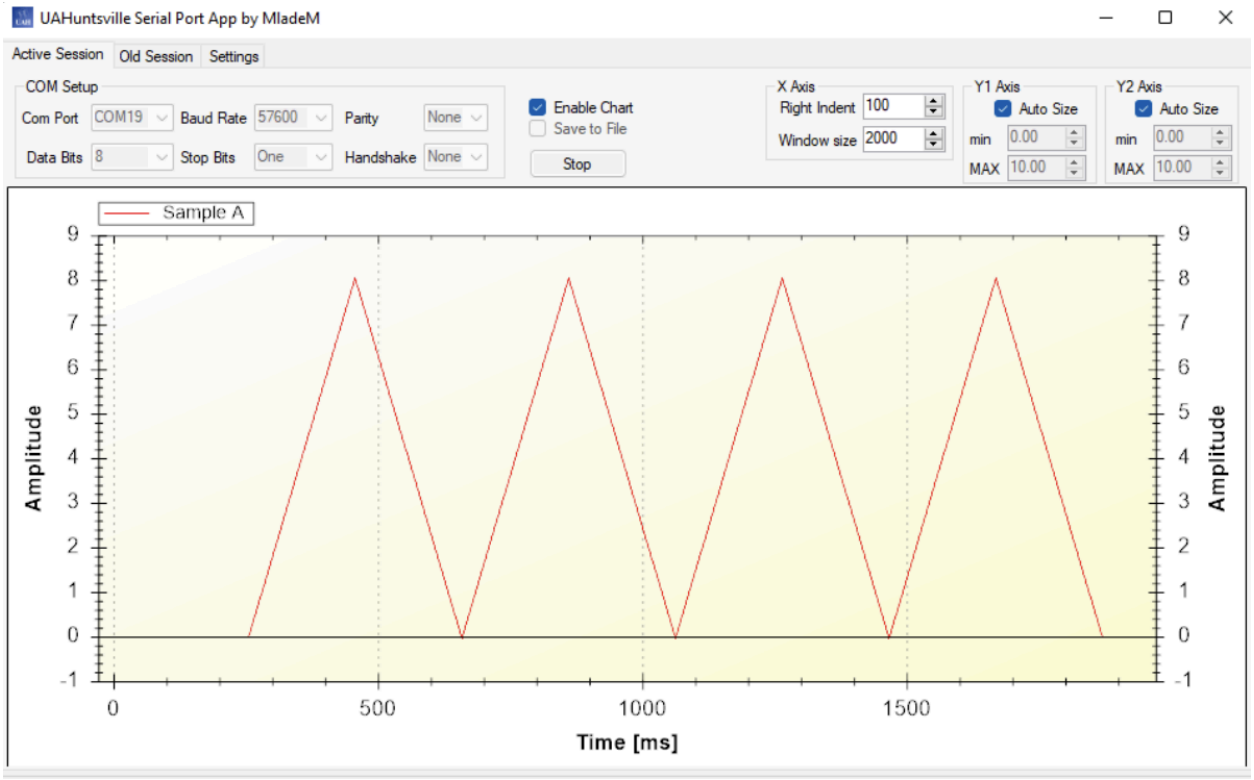

Figure 03: Program 2 Output

Appendix

Table 01: Program 1 source code

```
/*-------------------------------------------------------------------------
 * File:          Lab08_P1.c
 * Function:      Medbot Interactive chatbot
 * Description:   Medbot is an interactive chatbot that helps the user
*                 prescribe medication. It prompts the user for their        *
username, continuously asks what they need medication for, *
and calculates the total price for all their medicine.
 *
 * Input:         User input
 * Output:        Medbot chat
 *
 * Instructions:  Set the following parameters in putty/hyperterminal
 * Port: COM1
 * Baud rate: 115200
 * Data bits: 8
 * Parity: None
 * Stop bits: 1
 * Flow Control: None
 *
 * Author:      Anshika Sinha
 * Date:        March 4, 2025
-------------------------------------------------------------------------*/

#include <msp430.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <cstring.h>

// Define colors
#define colorReset   "\x1b[0m"
#define colorBot     "\x1b[31m"  // MedBot in red
#define colorUser    "\x1b[35m"  // User in purple

// Function prototypes
void UART_SETUP();
void UART_sendChar(char bip);
char UART_getChar();
void UART_sendString(char* str);
void UART_getLine(char* buf, int limit);

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;       // Stop watchdog timer
    UART_SETUP();                   // Setup UART

    // Initialize variables
    char userName[50];
    char item[10];
    char response[10];
    char msg[100];
    int totalPrice = 0;
```

```
    while(1) {
        // Prompt UserName
        UART_sendString(colorBot "[MedBot]: " colorReset);
        UART_sendString("\r\nHi I am MedBot. What is the name of the
patient?\r\n");

        UART_sendString(colorUser "[User]: " colorReset);
        UART_getLine(userName, 50);             // Get username

        // Greet user
        snprintf(msg, sizeof(msg),
                 "What is %s dealing with today? I can prescribe for headache,
cough, and allergy.\r\n",
                 userName);
        UART_sendString(colorBot "[MedBot]: " colorReset);
        UART_sendString(msg);

        totalPrice = 0;                         // Reset total price for new
prescription

        do {
            // User selects an item
            UART_sendString(colorUser "[User]: " colorReset);
            UART_getLine(item, 10);

            while (strcmp(item, "headache") != 0 && strcmp(item, "cough") != 0
&& strcmp(item, "allergy") != 0) {
                snprintf(msg, sizeof(msg),
                         "I cannot prescribe for %s. I can prescribe for
headache, cough, and allergy.\r\n",
                         item);
                UART_sendString(colorBot "[MedBot]: " colorReset);
                UART_sendString(msg);

                UART_sendString(colorUser "[User]: " colorReset);
                UART_getLine(item, 10);
            }

            // Handle item price and quantity
            string med = (strcmp(item, "headache") == 0) ? "aspirin" :
                    (strcmp(item, "cough") == 0) ? "cough drops" :
"antihistamines";
            int PRICE = (strcmp(item, "headache") == 0) ? 1 :
                    (strcmp(item, "cough") == 0) ? 2 : 3;

            snprintf(msg, sizeof(msg),
                     "%s costs $%d per dose. How many doses would you
like?\r\n", med, PRICE);
            UART_sendString(colorBot "[MedBot]: " colorReset);
            UART_sendString(msg);

            UART_sendString(colorUser "[User]: " colorReset);
            UART_getLine(response, 10);  // Get quantity as input
            int QUANTITY = atoi(response);  // Convert to integer

            totalPrice += PRICE * QUANTITY;  // Update total price
```

```
            // Ask if the User needs more medicine
            UART_sendString(colorBot "[MedBot]: " colorReset);
            UART_sendString("Do you need any more medicine?");

            UART_getLine(response, 10);  // Get yes/no response

            // Validate yes/no response
            while (strcmp(response, "yes") != 0 && strcmp(response, "no") !=
0) {
                UART_sendString(colorBot "[MedBot]: " colorReset);
                UART_sendString("Please respond with 'yes' or 'no'.\r\n");

                UART_sendString(colorUser "[User]: " colorReset);
                UART_getLine(response, 10);
            }

            // If "yes", re-prompt for the next item
            if (strcmp(response, "yes") == 0) {
                UART_sendString(colorBot "[MedBot]: " colorReset);
                UART_sendString("What is %s dealing with today? I can
prescribe for headache, cough, and allergy.\r\n");
            }

        } while (strcmp(response, "yes") == 0);

        // Show total price
        snprintf(msg, sizeof(msg),
                 "The total for all items is $%d. Thank you for prescribing
medicine for %s!\r\n\n",
                 totalPrice, userName);
        UART_sendString(colorBot "[MedBot]: " colorReset);
        UART_sendString(msg);
    }
}

// UART Setup
void UART_SETUP() {
    P3SEL |= BIT3 + BIT4;       // Set RXD/TXD pins
    UCA0CTL1 |= UCSWRST;        // Software reset
    UCA0CTL0 = 0;               // Control register
    UCA0CTL1 |= UCSSEL_2;       // Use SMCLK

    UCA0BR0 = 0x09;             // Baud rate 115200
    UCA0BR1 = 0x00;
    UCA0MCTL = 0x02;

    UCA0CTL1 &= ~UCSWRST;       // Initialize UART state machine
}

// Send a character via UART
void UART_sendChar(char bip) {
    while (!(UCA0IFG & UCTXIFG));  // Wait for TX buffer to be ready
    UCA0TXBUF = bip;
}

// Receive a character via UART
```

```
char UART_getChar() {
    while (!(UCA0IFG & UCRXIFG));  // Wait for RX buffer to be ready
    return UCA0RXBUF;
}

// Send a string via UART
void UART_sendString(char* str) {
    while (*str) {
        UART_sendChar(*str++);
    }
}

// Receive a line of input via UART with backspace handling
void UART_getLine(char* buf, int limit) {
    int i = 0;
    char ch;

    while (1) {
        ch = UART_getChar();

        if (ch == '\r' || ch == '\n') {  // Enter key pressed
            buf[i] = '\0';  // Null-terminate the string
            UART_sendString("\r\n");
            break;
        } else if (ch == '\b' || ch == 127) {  // Backspace pressed
            if (i > 0) {
                i--;  // Move buffer pointer back
                UART_sendString("\b \b");  // Clear character on screen
            }
        } else if (i < limit - 1) {  // Normal character input
            buf[i++] = ch;
            UART_sendChar(ch);  // Echo the character back
        }
    }
}
```

Table 02: Program 2 source code

```
/*-------------------------------------------------------------------------
----
 * File:          Lab08_P2.c
 * Function:      Generate triangular wave using serial
 * Description:   This program generates a triangular wave and displays it on
the UAH
 *                Serial App using a 57,600 baud rate. The amplitude of the
wave is 8 units
 *                and the frequency is 2.5 Hz. The signal is transmitted for 4
signal
 *                periods.
 *
 * Input:         None
 * Output:        Triangular wave on UAH Serial App
 *
```

```
 * Instructions:  Set the following parameters in putty/hyperterminal
 * Port: COM1
 * Baud rate: 57600
 * Data bits: 8
 * Parity: None
 * Stop bits: 1
 * Flow Control: None
 *
 * Author:      Anshika Sinha
 * Date:        March 4, 2025
-------------------------------------------------------------------------
--*/
#include <msp430.h>
#include <stdint.h>
#include <stdbool.h>

// Global variables
volatile float myData = 0.0;            // Stores the current value of the
waveform
int i = 0;                              // Counter to track the number of
completed periods (full wave cycles)
bool peak = true;                       // Flag to indicate if the waveform is
rising (true) or falling (false)

// Function to set up UART communication
void UART_setup(void) {
    P3SEL |= BIT3 | BIT4;               // Configure P3.3 (TXD) and P3.4 (RXD)
for UART functionality
    UCA0CTL1 |= UCSWRST;                // Put USCI_A0 in reset mode to
configure settings
    UCA0CTL1 |= UCSSEL_2;               // Use SMCLK (1 MHz) as the clock
source for UART
    UCA0BR0 = 18;                       // Set the lower byte of the baud rate
divider (1 MHz / 57600)
    UCA0BR1 = 0;                        // Set the upper byte of the baud rate
divider to 0
    UCA0MCTL = UCBRF_0 | UCBRS_2; // Configure modulation for accurate baud
rate timing
    UCA0CTL1 &= ~UCSWRST; // Release USCI_A0 from reset to begin operation
}
// Function to transmit a character over UART
void sendChar(char c) {
    while (!(UCA0IFG & UCTXIFG));       // Wait until the UART transmit buffer
is ready
    UCA0TXBUF = c;                      // Load the character into the
transmit buffer to send it
}
// Main function to set up the system and enter low power mode
int main() {
    WDTCTL = WDT_MDLY_32;               // Configure the Watchdog Timer for 32
ms interval interrupts
    UART_setup();                       // Initialize UART for communication
    SFRIE1 |= WDTIE;                    // Enable interrupts for the Watchdog
Timer
    myData = 0.0;                       // Initialize waveform value to 0
(start of rising phase)
```

```
    __bis_SR_register(LPM0_bits + GIE); // Enter Low Power Mode 0 with global
interrupts enabled
}
// Watchdog Timer ISR - Handles the generation and transmission of the
triangular wave
#pragma vector = WDT_VECTOR
__interrupt void watchdog_timer(void) {
    char *dataPtr = (char*)&myData;     // Pointer to access the bytes of the
floating-point waveform value
    sendChar(0x55); // Send a start byte (optional) to indicate the beginning
of the packet

    // Loop to send each byte of the floating-point value over UART
    int j;
    for (j = 0; j < 4; j++) {
        sendChar(dataPtr[j]);           // Send one byte of the floating-point
value
    }

    // Update the waveform value based on the current phase (rising or
falling)
    if (peak) {
        myData += 0.04;                 // Increase the value by 0.04 units in
the rising phase
        if (myData >= 8.0) {            // Check if the peak amplitude is
reached
            peak = false;               // Switch to the falling phase
        }
    } else {
        myData -= 0.04;                 // Decrease the value by 0.04 units in
the falling phase
        if (myData <= 0.0) {            // Check if the waveform has reached
the minimum value
            peak = true;                // Switch back to the rising phase
            i++;                        // Increment the period counter after
a complete up/down cycle
        }
    }
    // Stop the waveform generation after completing 4 full periods
    if (i == 4) {
        SFRIE1 &= ~WDTIE;               // Disable Watchdog Timer interrupts
to stop waveform generation
    }
}
```

Make sure to submit a single pdf file only