

CPE 325: Intro to Embedded Computer Systems

Lab08

UART Serial Communications

Submitted by: Michael Agnew

Date of Experiment: 3/6/2024

Report Deadline: 3/7/2024

Demonstration Deadline: 3/18/2024

Introduction:

The main objective of this lab was to understand serial communication with the MSP430 and whatever code associated with it. There were two different objectives that needed to be reached for this lab, the first of which was to piggyback a program from CCS to MobaXTerm and display an output and to the terminal. This program was also meant to take in input from the user and perform certain actions based on this input. The second code utilized the UAH Serial App, and allowed the user to make a program that displays a waveform graph onto it by means of serial communication once again.

Theory:

Serial Communication: There are two different ways that one can connect the MSP430 to a different system. The first of these is synchronously, which requires the MSP430 and the device it wants to communicate with to have the same clock signal. This is not something that is very easy to do, so instead for this lab, the MSP430 was connected asynchronously using UART serial communication. In this UART mode, the divider located internally must be initialized by finding the calculation of the clock by the baud rate. This quotient will rarely be a whole number which means that the modulation register must be used to approximate the baud rate. In a more concise way, serial communication is a way that embedded systems can communicate with one another regardless of whether or not they share the same clock signal. This can be done with a very limited amount of wiring which is very useful and cost-effective.

UAH Serial App: The UAH serial app is an application that allows one to display a waveform graph using a CCS program and the MSP430. A serial connection is made between the two using a cable, and the serial app is set to “connect” in order for it to take in the signal. The program

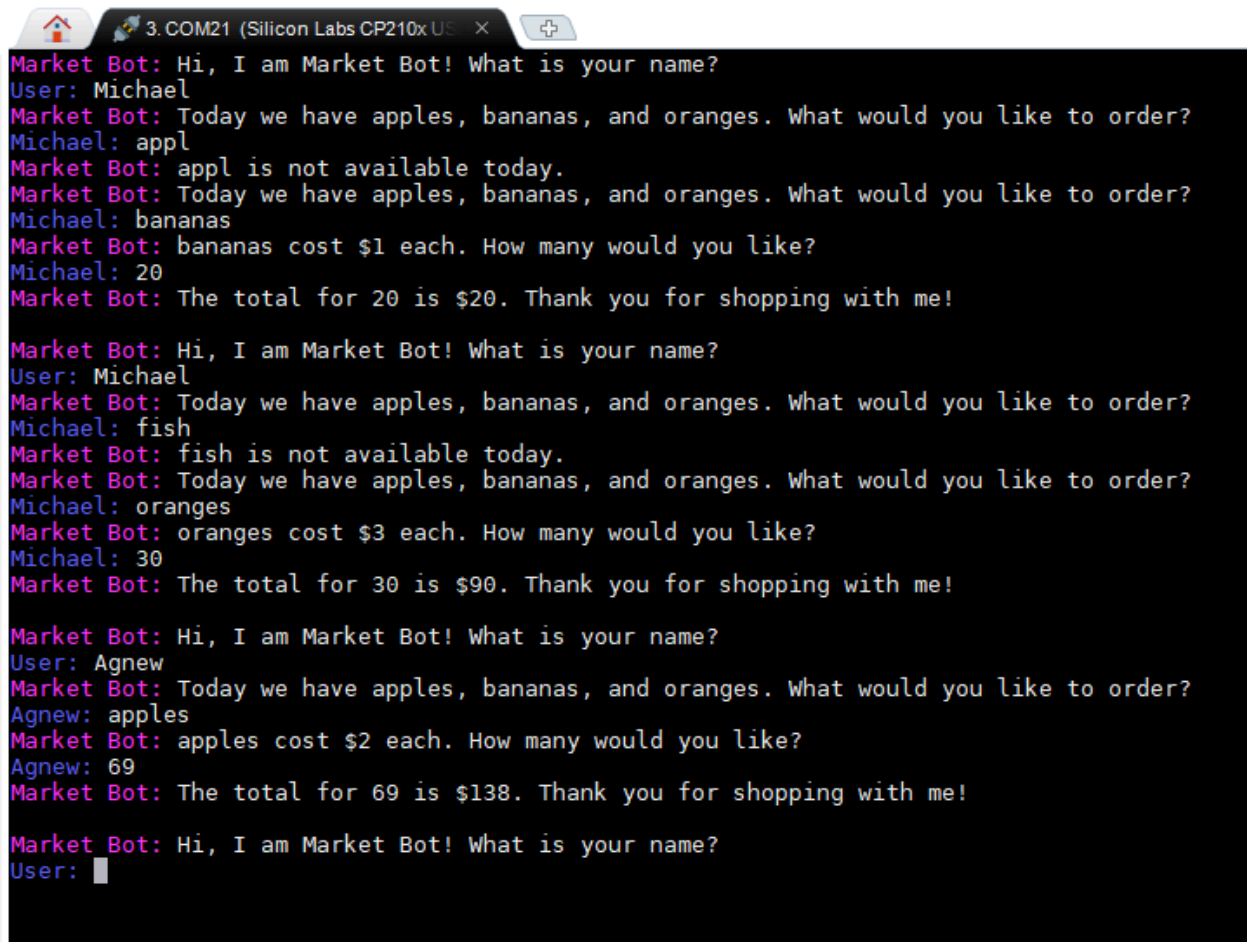
that is loaded into CCS is then run and the waveform is then produced on the app. Because this is a serial connection, the correct baud rate must also be set for this in order for it to work properly.

Problem 01:

Description: The first problem that was listed for this assignment was making a Market Bot in the MobaXTerm terminal using UART communication. The way that this was meant to be made was that the bot would greet the user and then prompt the user for their name. After receiving the name, the bot would then inform the user what fruits it had available in stock. After choosing a fruit, the market bot would then tell the user how much that fruit costs and prompt the user for how many they want. After the user inputs the number of that fruit they want, the market bot performs a basic calculation in order to determine the total price. This total price is then output to the user before the program loops and greets the user once again. In the event that the user types in the name of a fruit that is not in stock, the market bot will inform the user that that fruit is not available and then resend the message explaining which fruits are available.

Program Execution: The way that this program works is, it utilizes a few different functions: setup, sendChar, sendString, getLine, and simply main. These functions work in unison in order to complete the overall program. sendChar, getChar, sendString, and getLine are all instrumental in sending strings/characters to the terminal as well as receiving and processing them. The setup function initializes the needed hardware registers and functions that allow the serial communication that the program needs, and main controls the majority of the output by using the aforementioned functions. With all of these functions working in conjunction with each other, the goal that was explained in the description was achieved in the proper way.

Bonus: The bonus for this assignment revolved around the code assigning a color to the user's name, and implementing a method to use backspace in the terminal in order to delete mistakes made while typing. These were both done and can be seen in the output generated by the code when run through the necessary evaluations.



```
3. COM21 (Silicon Labs CP210x USB to UART Bridge) x
Market Bot: Hi, I am Market Bot! What is your name?
User: Michael
Market Bot: Today we have apples, bananas, and oranges. What would you like to order?
Michael: appl
Market Bot: appl is not available today.
Market Bot: Today we have apples, bananas, and oranges. What would you like to order?
Michael: bananas
Market Bot: bananas cost $1 each. How many would you like?
Michael: 20
Market Bot: The total for 20 is $20. Thank you for shopping with me!

Market Bot: Hi, I am Market Bot! What is your name?
User: Michael
Market Bot: Today we have apples, bananas, and oranges. What would you like to order?
Michael: fish
Market Bot: fish is not available today.
Market Bot: Today we have apples, bananas, and oranges. What would you like to order?
Michael: oranges
Market Bot: oranges cost $3 each. How many would you like?
Michael: 30
Market Bot: The total for 30 is $90. Thank you for shopping with me!

Market Bot: Hi, I am Market Bot! What is your name?
User: Agnew
Market Bot: Today we have apples, bananas, and oranges. What would you like to order?
Agnew: apples
Market Bot: apples cost $2 each. How many would you like?
Agnew: 69
Market Bot: The total for 69 is $138. Thank you for shopping with me!

Market Bot: Hi, I am Market Bot! What is your name?
User: 
```

Figure 1. Market Bot Program Output and Interaction with User.

There was also a flowchart that was required to be made for this program, it can be seen below:

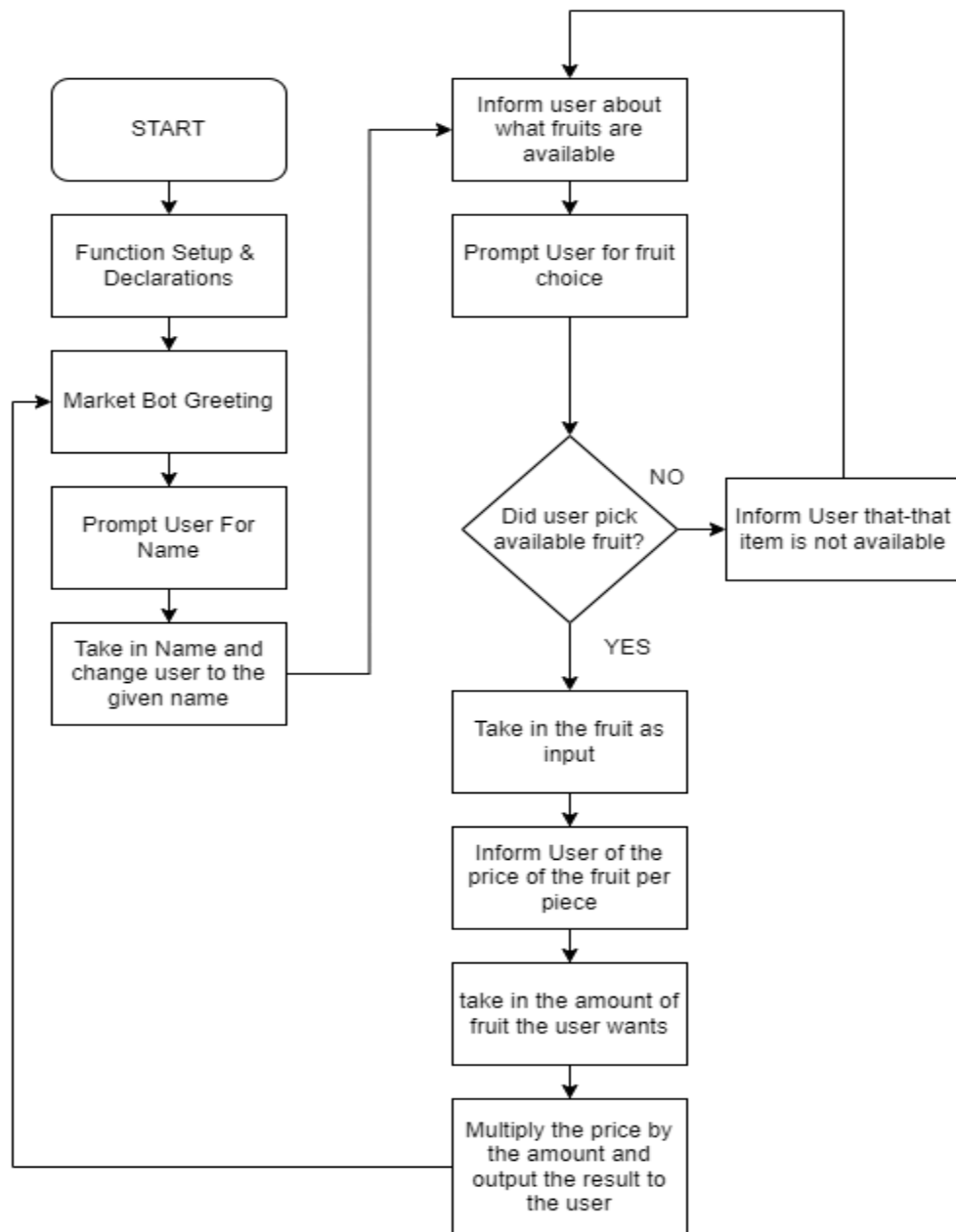


Figure 2. Flowchart for Problem 01.

Problem 02:

Description: Problem 2 revolved around the idea of making a C language program that makes a waveform graph out of a triangular wave and displays the effect in the UAH Serial App. This was to be done with a 57,600 baud connection and an amplitude (height) of 9. The final requirement was that the frequency of this wave be 4hz.

Program Execution: The way that this was done was by enabling the watchdog timer interrupt to go off every 250 ms, which comes out to be 4hz as the assignment described. This then enters into the watchdog timer ISR where a loop is entered in order for the waveform to be made equal as it rises and falls in order to make an even triangle. The amplitude is also bottlenecked to make sure that it does not grow higher than 9. In terms of the baud rate, $2^{20}/57,600$ is ~ 18 , so the baud lower byte is set to 18. The output of this program is as shown below:

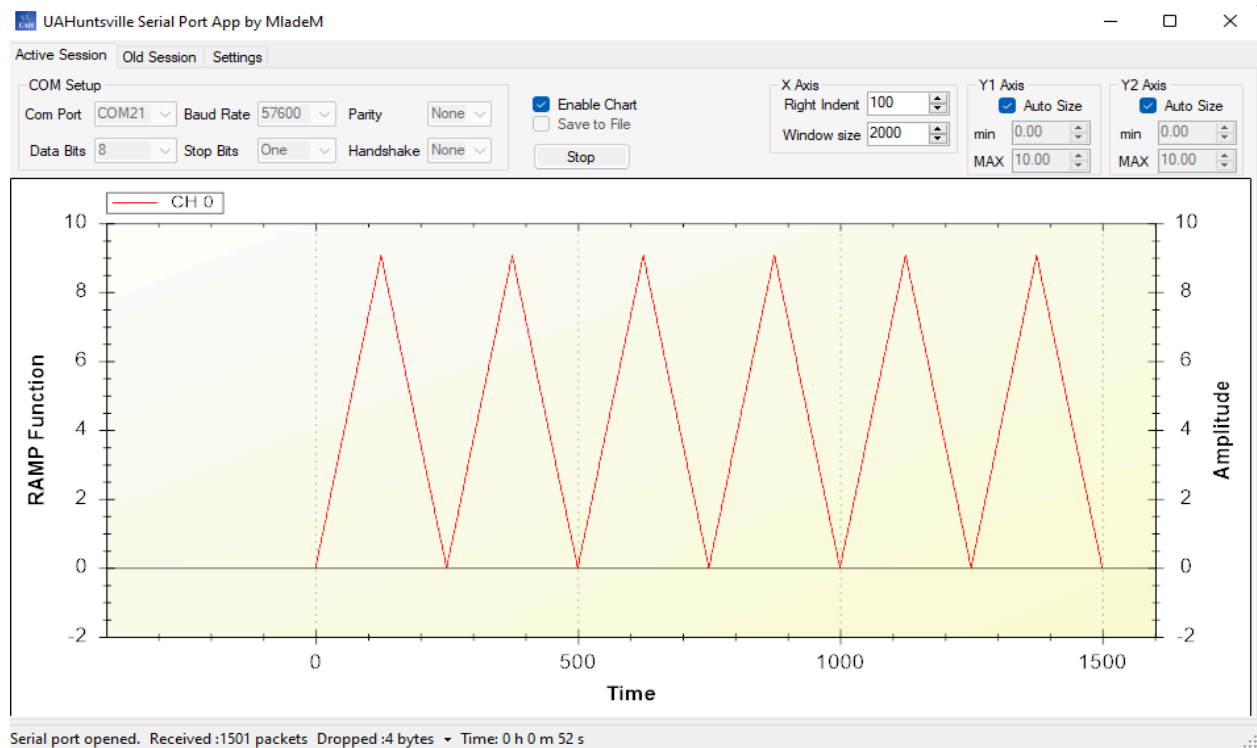


Figure 3. Serial App Triangular Waveform.

In order for there to only be 6 triangles for this code, a counter needed to be made and incremented when the triangle was on its rising side and its falling side. This means that a threshold of 12 was set in order for the loop to be exited and the program to stop.

Tutorial Questions:

1. What clock signals are stopped in LPM0?

ANS: There are two different clock cycles that are stopped in LPM0, these are CPU and MCLK.

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

ANS: The maximum amount of time that you can have for the real-time clock is 99999999 because 8 is the amount of characters that it can handle which means that the max time is 99999999.

Conclusion:

Fortunately, there were no difficulties that were encountered over the course of the lab.

Everything that was set out to be completed was done so in the allotted time and in the way that it was meant to be done. The means of connecting the MSP430 to another system by means of serial communication was learned as well as the use of the UAH Serial App. The serial communication connections were made properly and both assignments were completed. The market bot worked properly and the bonus was implemented as well. The serial app was also configured to output the waveform triangles properly and the output was gathered for the results. Overall, the lab was a success and everything was implemented properly.

Appendix:

Problem 01:

```
/*-----  
* File:      Lab8CodeFinal.c  
* Function:   Makes a market bot that allows the user the buy fruits from it  
* Description: Market Bot sells fruits to user  
* Input:      User Input from MobaXTerm  
* Output:     MobaXTerm Terminal  
* Author(s):  Michael Agnew, ma0133@uah.edu  
* Date:       3/6/2024  
*-----*/  
  
#include <msp430.h>  
#include <stdio.h>  
#include <string.h>  
  
// ANSI escape codes for colors  
#define ANSI_COLOR_RESET "\x1b[0m"  
#define ANSI_COLOR_AUTHOR1 "\x1b[35m" // Makes a pink color for Market Bot  
#define ANSI_COLOR_AUTHOR2 "\x1b[36m"  
#define ANSI_COLOR_USER "\x1b[34m" // Makes a blue color for user  
  
void UART_sendChar(char thing) // function for sending character to terminal  
{  
    while (!(UCA0IFG & UCTXIFG));  
    UCA0TXBUF = thing;  
}  
  
void UART_sendString(const char* str) // sends a string to the terminal  
{  
    while (*str != '\0') // while not at the end of the string  
    {  
        UART_sendChar(*str++); // output the string to the terminal  
    }  
}  
  
char UART_getChar(void) // gets the character from the terminal
```



```

{
    while (!(UCA0IFG & UCRXIFG));
    return UCA0RXBUF;
}

void UART_getLine(char* bufferArray, int limit, const char* inputName) // get the line input
from the terminal
{
    int i = 0;
    char c;

    // Display the input name prompt before waiting for user input
    UART_sendString(ANSI_COLOR_USER);
    UART_sendString(inputName);
    UART_sendString(": " ANSI_COLOR_RESET);

    while (1)
    {
        c = UART_getChar(); // initialize c as a character from getChar

        if (c == '\b' && i > 0)
        {
            UART_sendChar('\b');
            UART_sendChar(' '); // Use this to overwrite the character
            UART_sendChar('\b');
            i -= 1; // decrement the length of the buffer array
            bufferArray[i] = '\0'; // get rid of the character from the buffer array
            continue;
        }

        if ((c == '\r') || (i == limit - 1))
        {
            break; // bweak hehe
        }

        bufferArray[i++] = c; // set the new character in the buffer array to c

        UART_sendChar(c); // send c to the terminal
    }
    bufferArray[i] = '\0'; // set the end of the buffer array to the null character
}

```

```

void UART_setup(void) // setup for the serial connection
{
    P3SEL |= BIT3 + BIT4; // Set USCI_A0 RXD/TXD to receive/transmit data
    UCA0CTL1 |= UCSWRST; // Set software reset during initialization
    UCA0CTL1 |= UCSSEL_2; // Clock source SMCLK
    UCA0BR0 = 0x09; // 1048576 Hz / 115200 lower byte
    UCA0BR1 = 0x00; // upper byte
    UCA0MCTL |= UCBRS_1 + UCBRF_0; // Modulation (UCBRS0=0x01,
UCOS16=0)
    UCA0CTL1 &= ~UCSWRST; // Clear software reset to initialize USCI state machine
}

```

```

int main(void)
{
    WDTCTL = WDTPW + WDTHOLD; // stop the watchdog timer
    UART_setup(); // set up the UART

    char userInput[100]; // make an array to store the user input
    char userName[100]; // make an array to store the user name

    while (1)
    {

        UART_sendString(ANSI_COLOR_AUTHOR1 "Market Bot: "
ANSI_COLOR_RESET); // output market bot in pink
        UART_sendString("Hi, I am Market Bot! What is your name?\r\n"); // output the
market bot greeting
        UART_getLine(userName, 100, "User"); // output the name of the user
        UART_sendString("\r\n"); // Use this to make a new space

        while (1)
        {

            UART_sendString(ANSI_COLOR_AUTHOR1 "Market Bot: "
ANSI_COLOR_RESET); // output the market bot in pink
            UART_sendString("Today we have apples, bananas, and oranges. What would you like
to order?\r\n"); // output the options for fruit
            UART_getLine(userInput, 100, userName); // output the name of the user
            UART_sendString("\r\n"); // add an extra line

            if (strcmp(userInput, "apples") == 0 || strcmp(userInput, "bananas") == 0 ||
strcmp(userInput, "oranges") == 0) // see if the fruit match

```

```

    {
        break; // bnweak out heuueh
    }

    UART_sendString(ANSI_COLOR_AUTHOR1 "Market Bot: "
ANSI_COLOR_RESET); // output the market bot in pink
    UART_sendString(userInput); // output what the user said
    UART_sendString(" is not available today.\r\n"); // tell the user that the fruit is not
available
    }

    int fruitAmount;
    int price;
    if (strcmp(userInput, "apples") == 0)
    {
        price = 2; // if apples, price is 2
    }
    else if (strcmp(userInput, "bananas") == 0)
    {
        price = 1; // if banana, price is 1
    }
    else if (strcmp(userInput, "oranges") == 0)
    {
        price = 3; // orange is 3 bruh
    }

    char costArray[100]; // make an array for the cost
    sprintf(costArray, "%d", price); // print the price

    UART_sendString(ANSI_COLOR_AUTHOR1 "Market Bot: "
ANSI_COLOR_RESET); // print market bot in pink
    UART_sendString(userInput); // print user input
    UART_sendString(" cost $"); // print the cost
    UART_sendString(costArray); // print the cost but like for real this time
    UART_sendString(" each. How many would you like?\r\n"); // asking how many
FRUIT BOYS the user wants
    UART_getLine(userInput, 100, userName); // get the user's input

    sscanf(userInput, "%d", &fruitAmount); // get fruit or something
    int total = price * fruitAmount; // I remember when the MSP430 said "it's mathing
time" and then it math'd all over the place
    char total_str[10]; // string time
    sprintf(total_str, "%d", total); // print total

```

```

        UART_sendString("\r\n"); // Add new line for spacing
        UART_sendString(ANSI_COLOR_AUTHOR1 "Market Bot: "
ANSI_COLOR_RESET); // print name of market bot in pink
        UART_sendString("The total for "); // total
        UART_sendString(userInput); // user stuff
        UART_sendString(" is $"); // money money
        UART_sendString(total_str); // total_str dont ask
        UART_sendString(". Thank you for shopping with me!\r\n"); // thanks man you too
        UART_sendString("\r\n"); // new line new me
    }
}

```

Problem 02:

```

/*-----
* File:      lab8part2.c
* Function:   Send floating data to Serial port
* 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: COM1
* Baud rate: 115200
* Data bits: 8
* Parity: None
* Stop bits: 1
* Flow Control: None
*
*      MSP430F5529
*      -----
* /\ |      XIN|-
* | |      | 32kHz
* |--|RST   XOUT|-
*
*      |      |
*      | P2.4/UCA0TXD|----->
*      |      | 115200 - 8N1

```

```

*      | P2.5/UCA0RXD|<-----
*      |
* Input:      None
* Output:     Ramp signal in UAH Serial app
* Author:     Michael Agnew, ma0133@uah.edu
* Date:       March 6th, 2024
*-----*/
#include <msp430.h>
#include <stdint.h>
#include <stdbool.h>

volatile float myData = 0.0; // data
int i = 0; // loop counter
bool peak = 1; // boolean that keeps track of whether or not the waveform is rising or falling

void UART_setup(void)
{
    P3SEL |= BIT3 | BIT4; // Make sure that these are capable of transmitting and
receiving data
    UCA0CTL1 |= BIT0; // I think this is for software reset
    UCA0CTL0 = 0; // Control register
    UCA0CTL1 |= UCSSEL_2; // SMCLK
    UCA0BR0 = 18; // 1048576 Hz/57,600 lower byte
    UCA0BR1 = 0x00; // This is for the upper byte
    UCA0MCTL = 0x02; // do the mod thing
    UCA0CTL1 &= ~BIT0; // UCSWRST software reset
}

void sendChar(char c)
{
    while (!(UCA0IFG & UCTXIFG)); // Wait for previous character to transmit
    UCA0TXBUF = c; // Put character into tx buffer
}

int main()
{
    WDTCTL = WDT_MDLY_32; // 32 becuase 1000 sucks, also MDLY
    UART_setup(); // Initialize USCI_A0 module in UART mode
    SFRIE1 |= WDTIE; // Enable watchdog interrupts

    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)
{
    char *pntr = (char*)&myData;
    char next = 0;
    sendChar(0x55);
    for (next = 0; next < 4; next++) // iterate 4 times essentially
    {
        sendChar(pntr[next]); // send character
    }
    if (myData < 0.0) // while the data is 0, move it move it
    {
        peak = 1; // set the peak to true
        i += 1; // inc
    }
    else if (myData >= 9.0)
    {
        peak = 0; // set peak to false
        i += 1; // incing time
    }
    if (peak) // if peak is true ( which means it's like going to the top or something )
    {
        myData += 0.1; // increase
    }
    else
    {
        myData -= 0.1; // if peak is not true, then it must be false duh so go down
    }
    if (i == 12) // once it reaches 12, you know that you have 6 triangles and we only want
6 triangles please dont have more triangles than me
    {
        SFR1E1 &= ~ WDTIE; // disable watchdog interrupts
    }
}

```