Goals:

- Learn about UART Communication and the 12 Button Keypad and interface two micros together.
- The transmitting micro (or receiving micro) should translate keypresses into characters like a phone. The receiving micro should display the received characters on the LCD Display.

Bonus: (Choose 1)

Display the changing characters in real time: the current character being changed should blink until the user has not pushed any buttons after a while. +10

Instead of transmitting to another micro, transmit the characters to a terminal program on the PC. Indicate data is being transmitted from the micro with the Green LED and that data is being received with the Red LED. +10

Pre Lab Questions:

- Why use UART? What baud rate did you use? Why?
- How does ASCII encoding work?
- Can you only transmit chars; why or why not?
UART Communication comes in 2 modes: software and hardware. Both configurations are detailed in the various UART sections in the User’s guide, but for simplicity’s sake, UART in software mode allows us to use the RX and TX pins (P1.1 and P1.2, respectively) for communication. This mode is useful for transmitting info between two connected MSP430s. UART in hardware mode is useful for using the so called ‘Back Channel’, i.e. communication between the MSP430 and the PC. You can send data stored in the micro to the PC and vice versa.

UART Data Transmission

Data is sent in 10bit packets: one start bit, one stop bit and 8 character bits, which are encoded using the ASCII standard. The data packet is transmitted at a speed known as the Baud Rate, or bits per second. See the diagram below:

Start bits are low while stop bits are high. Notice how the data is sent LSB first. The hex value is 2Dh, the value for the ‘-’ (minus sign) in ASCII.
**UART Setup**

Use the following settings in the supplied code to setup the **UART**:

**UCA0CTL0**: USCI_A0 control Register 0 - Configure the register as such: No parity, LSB first, 8-bit data, 1 stop bit, **UART**, Asynchronous.

**UCA0CTL1**: USCI_A0 control Register 1 - Use SMCLK as USCI clock source and enable software interrupts. Refer to Section 16.4.2 in the User’s Guide.

**UCA0BR0**: USCI_A0 Baud Rate Control Register 0 - Configure this 16-bit register to have a 1MHz frequency and 9600 Baud Rate. This value can be easily obtained from Table 15-4 in the MSP430 User’s Guide.

**UCA0MCTL**: USCI_A0 Modulation Control Register - Configure this register as such: 2nd Stage modulation = 1, Oversampling off. You can find the details for this register from Section 15.4.5 from the User’s Guide.

**IFG2**: Interrupt Flag Register 2 - Clear the flags. See Section 15.4.13.

**IE2**: Interrupt Enable Register 2 - Enable **UART** interrupts. Section 15.4.12.

**P1DIR**: This register controls all 8 pins of the Port 1, make the built-in LEDs to be outputs.

**P1OUT**: The register that determines the value of all pins that are programmed as outputs by P1DIR.

**P2IN**: The register that can be read to see the value of every pin; make sure to use P2DIR to make Port 2 pins as inputs.

**P1SEL & P1SEL2**: These 2 registers control the function of each Pin in Port 1 (depending on the current module being used); you are responsible for finding the correct combination (from the datasheet) to enable the **UART** function on Pins **P1.1** and **P1.2** (**RX** and **TX**, respectively).

You will be given a sample program with these settings already set. See how it functions and then apply the idea to this lab assignment. Be sure to have two separate projects; one for the receiving micro and one for the transmitting micro.
UART Connections

Connection is simple. Only two wires are needed for communication in software mode, three if you want to include a ground wire so the receiving micro can be used without it being powered. Only the USB cable is needed for hardware mode. Also, you must change the jumper configuration on the top right side of the micro. See the appropriate diagrams below for the sample program:
Here you should see the characters you are pressing in the terminal program on the PC and the characters being received by the micro. The LEDs on the micro should react to characters being received from the PC.

Notes:

- A sample program has been provided. Pressing Switch 0 should turn on the Red LED on the receiving micro; Switch 1 will turn on the Green LED; Switch 2 will turn on BOTH.
- When connecting to the PC: Switch 0 sends ‘d’; Switch 1 sends ‘i’; Switch 2 sends ‘r’.
- The Interrupt Vectors for the UART are only used for receiving and transmitting, but you can change the state of the LEDs in their ISRs.
- You can access the necessary Receive and Transmit Buffer Registers, UCA0RXBUF and UCA0TXBUF, in any fashion, but only using the ISR will allow you to have your code be non-blocking.
UART Communication

Lab Hints

There is no sample code to start with; instead you must know enough by now to be able to write the program from scratch. However, the general program flow will be described below:

- Include necessary headers
- Declare necessary global variables
- Initialize and configure necessary modules (Ports, Interrupts)
- You may poll or use interrupts to capture keypresses
- You can use interrupts to execute short code and have your main loop handle the main functions, such as Receive and Transmit, and to display your data on the LCD screen.
- Use the Receive and Transmit Buffer Registers. Fill the Transmit Buffer Register to transmit a character; when a character is received, the Receive Buffer Register will be filled and you can retrieve the character from it.

Check the comments in source files/libraries provided. Make sure you understand what is happening in the sample programs provided. Below is a reference of how letters are derived from the key/button presses:

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHI</td>
<td>ABC</td>
<td>DEF</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>PQRS</td>
<td>JKL</td>
<td>MNO</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>#, *, 0</td>
<td>TUV</td>
<td>WXYZ</td>
</tr>
</tbody>
</table>
```
Lab Notebook Questions:

Answer the following questions in your lab notebook:

- Explain in detail the 3 different types of communication protocols used by the MSP430 (SPI, IIC (aka I2C) and UART) and in what applications it would be advantageous to use one over the other.
UART Communication
Frequently Asked Questions

How can I receive a string; a character is not enough?

Simply use a loop to wait until the UART is no longer busy by checking the UART IFG. If it is not busy, then retrieve a character from the Receive Buffer Register and store in your character array by incrementing the array index in each iteration of the loop. Stop when you reach the end of the array or after however many characters you want your string to be.

...and to transmit a string?

Same idea but instead use the Transmit Buffer Register.

Can I use more/other pins for UART; how many UARTs can I use?

First, to clarify, true ‘software’ UART mode doesn’t use the UART module. Using the UART module, we are technically always using hardware mode, meaning pins P1.1 and P1.2 are almost always high. To keep them from interfering from other applications when not using the UART module, the first two jumpers from the left (the ones we rotate in the diagrams) should be removed.

You can only use the UART module for one setup. In theory, you can use delays and maybe even the timer to create your own ‘UART’ implemented in software, much like how you can generate a PWM with delays and setting a pin high or low, or also timer. But this would be more trouble than it be worth trying to get the timing perfect or even usable.