University of Texas at El Paso
Electrical and Computer Engineering Department

EE 3176 – Laboratory for Microprocessors I

Spring 2023

LAB 07
UART Communication

Goals:

- Learn about UART Communication and interface two microcontrollers together.
- The transmitting controller must translate data into a sequence of seven- or eight-bit values. The receiving controller must display received values as characters on an LCD Display.

Pre Lab Questions:

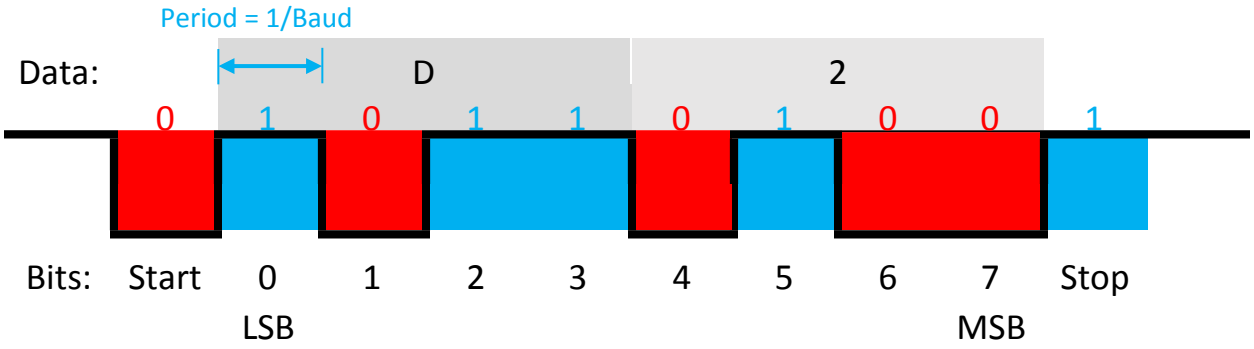
- Why use a UART?
- What is a baud rate?
- How does ASCII encoding work?
- Can you only transmit chars? Why?

UART Communication
Lab Guide

A **U**niversal **A**synchronous **R**eceiver **T**ransmitter (**UART**) is a common peripheral that provides asynchronous serial data communications capabilities. A UART offers two modes of operation: software mode and hardware mode. Both configurations are detailed in the various UART sections in the MSP432 User's guide. Using the software mode enables the receiver (RX) and the transmitter (TX) pins to communicate. This mode is useful for transmitting data between two MSP432s. The software mode is useful for setting up a so called "back channel" for communication between an MSP432 and a personal computer. You can send data stored in the microcontroller to the computer and vice versa.

UART Data Transmission

UARTs transmit data one serial frame at a time. For instance, data sent in a frame can have ten bits: one start bit, eight data bits, and one stop bit. Data bits can be used to transfer a character encoded using the ASCII standard. The data packet is transmitted at a speed specified as the UART baud rate, which in this case can be expressed in bits per second.



A start bit is a logical zero while stop bits are logical ones. Data shown above is being sent with the least significant bit (LSB) first. The data hex value is 2D, which is the value for the '-' character (dash or minus sign) in ASCII.

```
//******************************************************************************
//   MSP432P401 Demo - eUSCI_A0 UART echo at 9600 baud using BRCLK = 12MHz
//
//  Description: This demo echoes back characters received via a PC serial port.
//  SMCLK/ DCO is used as a clock source and the device is put in LPM0
//  The auto-clock enable feature is used by the eUSCI and SMCLK is turned off
//  when the UART is idle and turned on when a receive edge is detected.
//  Note that level shifter hardware is needed to shift between RS232 and MSP
//  voltage levels.
//
//               MSP432P401
//             -----------------
//         /|\|                 |
//          | |                 |
//          --|RST              |
//            |                 |
//            |                 |
//            |     P1.3/UCA0TXD|----> PC (echo)
//            |     P1.2/UCA0RXD|<---- PC
//            |                 |
//
//   William Goh
//   Texas Instruments Inc.
//   June 2016 (updated) | June 2014 (created)
//   Built with CCS v6.1, IAR, Keil, GCC
//******************************************************************************
#include "ti/devices/msp432p4xx/inc/msp.h"

int main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;

    CS->KEY = CS_KEY_VAL;           // Unlock CS module for configuration
    CS->CTL0 = 0;                   // Reset tuning parameters
    CS->CTL0 = CS_CTL0_DCORSEL_3;   // Set DCO to 12MHz
    CS->CTL1 = CS_CTL1_SELA_2 |     // Select ACLK = REFO
               CS_CTL1_SELS_3 |     // SMCLK = DCO
               CS_CTL1_SELM_3;      // MCLK = DCO
    CS->KEY = 0;                    // Lock CS module from unintended accesses
    // Configure Port 1 pins as UART TX and RX pins
    P1->SEL0 |= BIT2 | BIT3;        // set 2-UART pin as secondary function
    // Configure UART
    EUSCI_A0->CTLW0 |= EUSCI_A_CTLW0_SWRST; // Reset eUSCI
    EUSCI_A0->CTLW0 = EUSCI_A_CTLW0_SWRST | // Keep eUSCI in reset
                EUSCI_B_CTLW0_SSEL__SMCLK; // Use SMCLK as the eUSCI clock source
    // Baud Rate calculation
    // 12000000/(16*9600) = 78.125
    // Fractional portion = 0.125
    // User's Guide Table 21-4: UCBRSx = 0x10
    // UCBRFx = int ((78.125-78)*16) = 2
    EUSCI_A0->BRW = 78;                     // 12000000/16/9600
    EUSCI_A0->MCTLW = (2 << EUSCI_A_MCTLW_BRF_OFS) | EUSCI_A_MCTLW_OS16;

    EUSCI_A0->CTLW0 &= ~EUSCI_A_CTLW0_SWRST; // Initialize eUSCI
    EUSCI_A0->IFG &= ~EUSCI_A_IFG_RXIFG;    // Clear eUSCI RX interrupt flag
    EUSCI_A0->IE |= EUSCI_A_IE_RXIE;        // Enable USCI_A0 RX interrupt
```

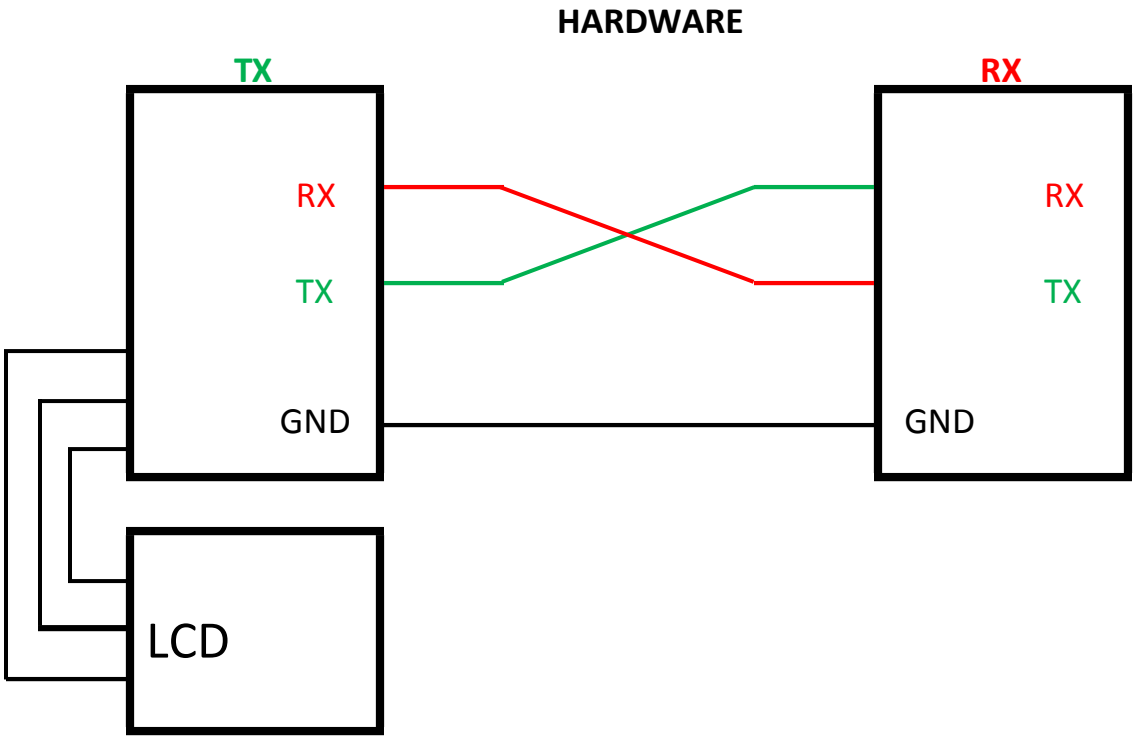
```
    // Enable global interrupt
    __enable_irq();

    // Enable eUSCIA0 interrupt in NVIC module
    NVIC->ISER[0] = 1 << ((EUSCIA0_IRQn) & 31);

    // Enable sleep on exit from ISR
    SCB->SCR |= SCB_SCR_SLEEPONEXIT_Msk;

    // Ensures SLEEPONEXIT occurs immediately
    __DSB();

    // Enter LPM0
    __sleep();
    __no_operation(); // For debugger
}

// UART interrupt service routine void
EUSCIA0_IRQHandler(void)
{
    if (EUSCI_A0->IFG & EUSCI_A_IFG_RXIFG)
    {
        // Check if the TX buffer is empty first
        while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG));

        // Echo the received character back
        EUSCI_A0->TXBUF = EUSCI_A0->RXBUF;
    }
}
```

UART Communication
Lab Guide

1. Create a new project and enter the sample program above.
2. Connect the microcontrollers as in the schematic below.

UART Connections

Two wires are needed for communication in software mode, three if a common ground wire is needed because one of the microcontrollers is not powered by the computer through a USB port. Only the USB cable is needed for hardware mode. Also, you must change the jumper configuration on the top right side of the Launchpad.



3. Add the LCD driver code.
4. Compile and run the program.
5. Modify the program to show on the LCD display both the transmitted and the received characters.
6. Make sure you test the system sending letters, numbers, and other printable characters.

UART Communication

Frequently Asked Questions

How can I receive a string? One frame is not enough.

You transfer strings one character (per frame) at a time. For each character to be received, wait in a loop until the UART is no longer busy by checking the UART IFG register. Once it is not busy, then retrieve a character from the Receive Buffer Register and store in a character array. Increment the array index in each loop iteration. Stop when you reach the end of the array, after receiving the number of characters you want in your string, or after getting a null string-terminating character.

…and to transmit a string?

Same idea but use instead the Transmit Buffer Register.

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

The software mode doesn't use the UART module. When using the UART module, we are using the hardware mode and the TX and RX pins. The TX pin will always be high when not transmitting any frame. The number of UARTs available in a microcontroller depends on its model number. See the User Guide for information about the model you are using.