Serial Communication Using MSP430

Objective: This lab exercise provides introductory exposure to serial communication between a microcontroller and PC, and to rudimentary control of external circuitry via digital ports.

Requirements:

- Initialize the UART and demonstrate serial communication by echoing commands to the host computer.
- Build a simple external circuit to interface with the controller.
- Configure the I/O ports and demonstrate external circuit control based on serial commands.

Prelab: Draw a functional block diagram showing the requirements to implement a system that accepts serial commands and then controls two wheel motors in response to those commands. Include code snippets, flow charts, or any other details that would help implement this design.

Install the CCS IDE from ti.com and obtain an MSP430G2553 evaluation board to ensure connection between the IDE and board for programming.

On a breadboard, build the circuit shown in figure 1.

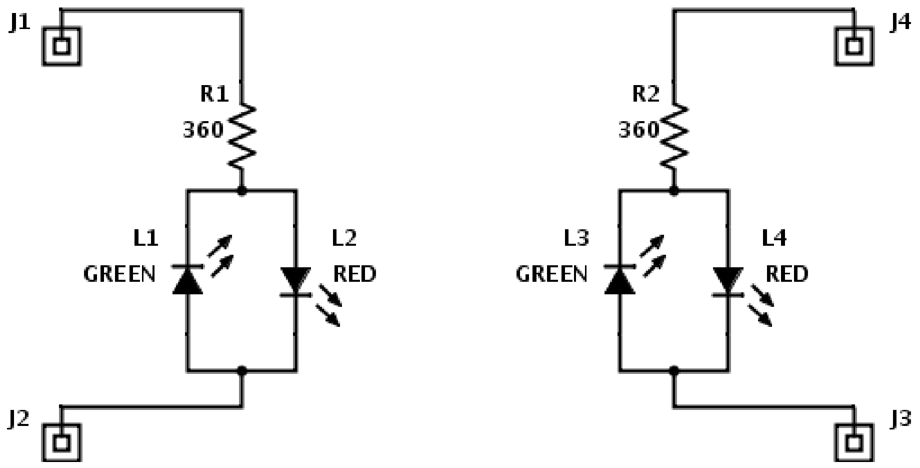

Figure 1. External circuit for MSP430 control

The Jumpers will connect to the headers on either side of the PCB in accordance with Table 1. LEDs should illuminate at approximately 4 mA, dropping approximately 2 V.

External Jumper	Microcontroller Port
J1	P2.3
J2	P2.2
J3	P2.0
J4	P2.1

Table 1. Jumper – port connectivity to external circuit

Background: The MSP430 communicates serially via RS-232 using one of its UARTs. In this lab, we will use Hyperterm with the following settings: 9600 baud, eight data bits, no parity, one stop bit, no flow control. This connection is made via USB with the aid of the "emulation portion" of the evaluation PCB depicted in Figure 2. Refer to Reference 1, pp. 81-95, for additional information related to the MSP430 UART.

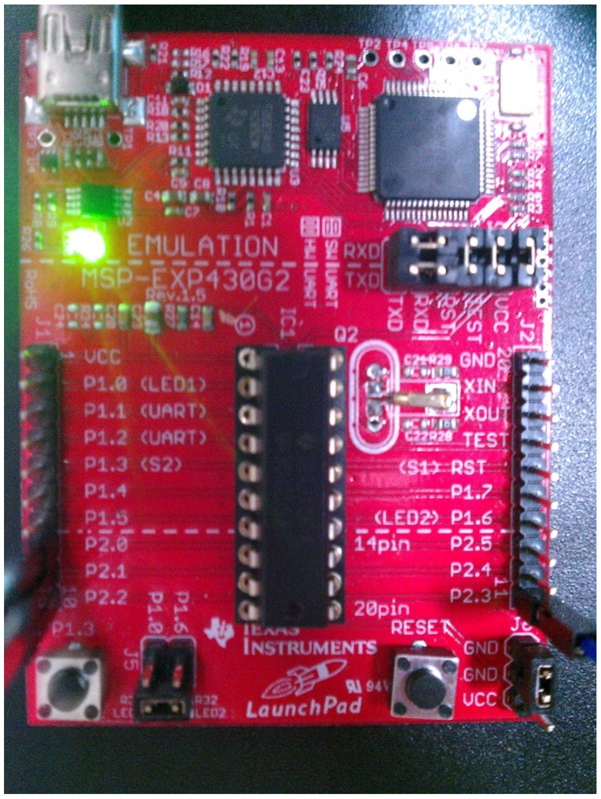

Figure 2. MSP430 evaluation board

Procedure: The objective will be accomplished in three steps:

- Establish serial communication.
- Code input processing logic.
- Interface to external circuit and debug.

1. Establish serial communication.

   Ensure the four pin jumpers are installed as shown in the upper-right corner of the image in Figure 2!

   Take the following steps to establish serial communication (this is exemplified in the subsequent code):

   a. Selecting the UART function for Pins
      P1SEL & P1SEL2 registers control the function of each Pin in port 1.Their precise combination is essential to enable the UART function on Pins P1.1 and P1.2.(Pg 42 Datasheet).

   b. Clock Selection
      There are three clock signals ACLK(Auxiliary Clock),MCLK(Master Clock) & SMCLK (Sub Master Clock) available as the basic clock signals. SMCLK provides the clock speed of 1MHZ.

   c. Baud rate selection:
      To set the baud rate the clock speed (1MHz) is divided by the baud rate and the register UCA0BR0 is loaded with integral part obtained after this division . UCA0BR0 holds the lower byte whereas UCA0BR1 hold the upper byte.

   d. Enabling Interrupts:
      Bytes are commonly received using interrupts and these must be enabled.

   After creating a new project with main.c file, add the following code to set up the peripherals:

```
#include "msp430g2553.h"    //Header file for the µC.

void send_data_to_uart(unsigned char *data_str, unsigned char
                       str_len);

void send_data_to_uart(unsigned char *data_str, unsigned char
                       str_len)
{
    while(str_len--)
    {
        while(!(IFG2 & UCA0TXIFG));
        UCA0TXBUF = *data_str;
        data_str++;
    }
}

static unsigned char data;

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;     // Stop WatchDogTimer
    UCA0CTL1 |= UCSWRST;         // Reset state
    BCSCTL1 = CALBC1_1MHZ;        // Set Basic CLK System Control
    DCOCTL = CALDCO_1MHZ;         // Set DCO to 1MHz,Pg23 Data St.
    /** Configure hardware UART **/
    P2DIR=0x0F;                   // P2 configured as output port.
    P2OUT=0x00;                   // P2 initialized to zero
    /******* Pin Selection *******/
    P1SEL = BIT1 + BIT2;         // P1.1 = RXD, P1.2 = TXD
    P1SEL2 = BIT1 + BIT2;         // P1.1 = RXD, P1.2 = TXD
    UCA0CTL1 |= UCSSEL_2;         // Use SMCLK
    /* Set baud rate to 9600 with 1MHz clock
     * (Datasheet 15.3.13) *******/
    UCA0BR0 = _;                 // To be determined by student
    UCA0BR1 =_;                   // 9600 baud with 1MHz clock
    UCA0MCTL = UCBRS0;            // Modulation UCBRSx = 1
    UCA0CTL1 &= ~UCSWRST;         // Initialize  state machine
    IE2 |= UCA0RXIE;              // Enable USCI_A0 RX interrupt
    /* Enter LPM0 (Low Power Mode)
     * & enable interrupts *******/
    __bis_SR_register(LPM0_bits + GIE);

    while(1);
}
```

Next, below the above code, we add an interrupt service routine (ISR) which is called any time the UART receives data. A function which sends this received data to the UART can be implemented for debugging.

```
/* Echo received character to the host computer.
 * First Ensure TX buffer is ready*/
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
    data = UCA0RXBUF;             // Initialize Port2


    send_data_to_uart("Rx: ", 4);
    send_data_to_uart(&data, 1);
    send_data_to_uart("\n", 1);


    switch(data)

    {
       case 'a':            // Left
       {  P2OUT=0x00;  }    // Value to be determined by student
       break;

       case 'd':            // Right
       {  P2OUT=0x00;  }    // Value to be determined by student
       break;
```

```
            case 'w':             // Forward
            {  P2OUT=0x00;  }     // Value to be determined by student
            break;

            case 's':             // Backward
            {  P2OUT=0x00;  }     // Value to be determined by student
            break;

            default:
            {  P2OUT=0x00;  }     // Stop
            break;
        }

    }
```

Ensure the controller is connected to the PC via USB cable, and then “build all” and run.

Connect to the UART using Hyperterm or a similar serial terminal. The microcontroller should respond to any key press with “Rx: “ followed by the letter that was sent.

Demonstrate this behavior to the TA.

TA Initials:_______

2. Code input processing logic.

   The switch statement above sets the bits associated with P2 in accordance with the received data.

   Change the hexadecimal values for case statements ‘a’, ‘d’, ‘w’, and ‘s’ to set (to 1) one of the four lower bits of P2, depending on the data received. Use Tables 2 and 3 for reference.

Command	J1-J2	J3-J4
Left	Off	Green
Right	Green	Off
Forward	Green	Green
Backward	Red	Red
Stop	Off	Off

Table 2. LED states for given commands

Binary	Hexadecimal
0000 0000	0x00
0000 0001	0x01
0000 0010	0x02
0000 0011	0x03
0000 0100	0x04
0000 0101	0x05
0000 0110	0x06
0000 1000	0x08
0000 1001	0x09
0000 1010	0x0A
0000 1100	0x0C

Table 3. Selected conversions between binary and hexadecimal

3. Interface to external circuit and debug.

   Connect the circuit to the control board according to Table 1.

   Demonstrate operation consistent with Table 2 to the TA.

   TA Initials:_______

Postlab: Write a report fully explaining the requirements and the steps necessary to fully achieve the requirements.

References

1. Litovsky, G., “Beginning Microcontrollers with the MSP430 Tutorial” [Online]. Available: http://www.argenox.com/design/Tutorialv0_4.pdf

2. Msp430g2553 Datasheet: http://www.ti.com/lit/ds/symlink/msp430g2553.pdf

3. User Guide MSP430G2533: http://www.ti.com/lit/ug/slau144j/slau144j.pdf

Sample Code :

```
#include "msp430g2553.h"  //Header file for the µC.

void send_data_to_uart(unsigned char *data_str, unsigned char
   str_len);

void send_data_to_uart(unsigned char *data_str, unsigned char
                       str_len)
{
while(str_len--)
{
while(!(IFG2 & UCA0TXIFG));
UCA0TXBUF = *data_str;
data_str++;
}
}

static unsigned char data;

void main(void)
{
WDTCTL = WDTPW + WDTHOLD;         // Stop WatchDogTimer
UCA0CTL1 |= UCSWRST;              // Reset state
BCSCTL1 = CALBC1_1MHZ;            // Set Basic CLK System Control
DCOCTL = CALDCO_1MHZ;             // Set DCO to 1MHz
/** Configure hardware UART **/
P2DIR=0x0F;                       // P2 configured as output port.
P2OUT=0x00;                       // P2 initialized to zero
/******* Pin Selection *******/
P1SEL = BIT1 + BIT2;              // P1.1 = RXD, P1.2 = TXD
P1SEL2 = BIT1 + BIT2;             // P1.1 = RXD, P1.2 = TXD
UCA0CTL1 |= UCSSEL_2;             // Use SMCLK
/* Set baud rate to 9600 with 1MHz clock
 * (Datasheet 15.3.13) *******/
UCA0BR0 = 104;
UCA0BR1 = 0;                      // 9600 baud with 1MHz clock
UCA0MCTL = UCBRS0;                // Modulation UCBRSx = 1
UCA0CTL1 &= ~UCSWRST;             // Initialize  state machine
IE2 |= UCA0RXIE;                  // Enable USCI_A0 RX interrupt
/* Enter LPM0 (Low Power Mode)
 * & enable interrupts *******/
__bis_SR_register(LPM0_bits + GIE);

while(1);
}

/* Echo received character to the host computer.
 * First Ensure TX buffer is ready*/
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
```

```
data = UCA0RXBUF;                // Initialize Port2


send_data_to_uart("Rx: ", 4);
send_data_to_uart(&data, 1);
send_data_to_uart("\n", 1);

switch(data)
{
case 'a':                       // Left
{   P2OUT=0x01;   }
break;

case 'd':                       // Right
{   P2OUT=0x04;   }
break;

case 'w':                       // Forward
{   P2OUT=0x05;   }
break;

case 's':                       // Backward
{   P2OUT=0x0A;   }
break;

default:
{   P2OUT=0x00;   }            // Stop
break;
}

}
```