EE 215 Homework 4 Solution

1. MSP430 hardware (7 pts) Answer the following questions:
   a. What are the operating modes of the MSP430, how many are there? How are the modes configured? [x5xx Family User's Guide, section 1.4] *The operating modes are for ultra-low-power applications; there are 8 different modes. The modes are configured by four bits in the SR.*
   b. What is a maskable interrupt? Why is it used? [x5xx Family User's Guide, section 1.3] *Maskable interrupts are caused by peripherals with interrupt capability. Each maskable interrupt source can be disabled individually by an interrupt enable bit, or all maskable interrupts can be disabled by the general interrupt enable (GIE) bit in the status register (SR).*
   c. What is an interrupt vector? *The interrupt vectors are located in the address range 0FFFFh to 0FF80h, for a maximum of 64 interrupt sources. A vector is programmed by the user and points to the start location of the corresponding interrupt service routine.*
   d. Why does the MSP430 have a watchdog timer? *How is the watchdog timer turned off? The primary function of the watchdog timer (WDT_A) module is to perform a controlled system restart after a software problem occurs. If the selected time interval expires, a system reset is generated. It is turned off by the C command: WDTCTL = WDTPW|WDTHOLD.*
   e. What is switch debounce? *A switch, when pushed, does not make a clean transition from 1 to 0, but bounces. There must be a hardware or software (or both) debounce to make sure the many switch contacts are seen as only one switch action.*

2. Bit operations (7 pts) If a = 10010111, b = 10001111, find the answers after the logic operation:

   a. a | b

```
a = 10010111
b = 10001111
    10011111
```

   b. a & b , bit wise AND

```
a = 10010111
b = 10001111
    10000111
```

   c. ~a , the binary effect of flipping bits

```
 a = 10010111
~a = 01101000
```

d. a ^ b, XOR, exclusive OR

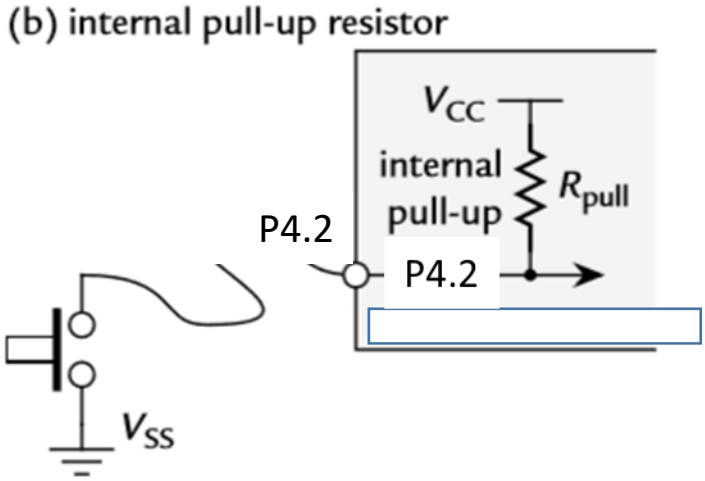
```
a = 10010111
b = 10001111
    00011000
```

e. a ^= b same as a = a ^ b (assignment) a = 00011000

f. a |= b same as a = a | b (assignment) a = 10011111

a &= b same as a = a & b (assignment) a = 10000111

3. Switch setup (6 pts) For a switch connected to P4.2 with an internal pullup resistor:



a. Draw the circuit.

b. write the C program lines to set up the bits in P4DIR, P4REN, and P4OUT, and comment these lines.

```
P4DIR &= ~ BIT2;  // set direction to input for BIT2
P4REN |= BIT2;   // pullup resistor
P4OUT |= BIT2;   // pullup resistor
```

4. C with Switch and LED (8 pts) Write a C program to turn on the LED (P1.0) only when the S1 (P2.1) is pushed. Configure the pullup resistor for the switch.

```
#include <msp430.h>
/**
 assume LED1=P1.0, S1=P2.1
 */
void main(void)
{

    WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer
// set up LED
     P1DIR|=BIT0;  // for LED1 set P1.0 to output
      P1OUT&=~BIT0; // turn off LED1
// set up switch
// PxDIR=0, PxREN=1, PxOUT=1, input with pullup resistor
      P2REN |=BIT1; //  P2.1 (S1) Pullup/Pulldown resistor enabled
      P2OUT |=BIT1; // P2.1 (S1) Pullup resistor enabled

      for (;;){
          if ((P2IN&BIT1)==0){  // button pressed
              P1OUT|=BIT0;}    // turn on LED1
              else{
              P1OUT&=~BIT0; }  // turn off LED1
      }
}
```

5. C with Switch and LED (12 pts) Write a C program for the MSP 430, using switch 1 (S1) to control LED1, switch 2 (S2) to control LED2 separately. Use pullup resistors for the switch. LED1 is P1.0, LED2 is P1.1, S1 is P1.6, S2 is P1.7 .
    a. if press S1, LED1 is on
    b. if press S2, LED2 is on;
    c. if press both, both LEDs are on;
    d. Make comments on each line of code to explain your program

```
#include <msp430.h>   //add head file
#define LED1 BIT0;   //define a symbol represent 0b00000001
#define LED2 BIT1;   //define a symbol represent 0b10000000
#define S1 BIT6;     //define a symbol represent 0b00000010
#define S2 BIT7;     //define a symbol represent 0b00000010
void main(void)
{
      WDTCTL = WDTPW | WDTHOLD;    //disable watchdog timer
// set up LEDs
      P1DIR |= LED1;
      P1DIR |= LED2;  //set P1.0 and P1.1 to output
      P1OUT &= ~LED1;
```

```
      P1OUT &= ~LED2;     //set LED1 and LED2 off initially (active high)
// set up switches
      P1REN |= S1;
      P1REN |= S2;  // pullup resistors
      P1OUT |= S1;
      P1OUT |= S2;  // pullup resistors

      for(;;){              //Loop forever
         if ( ((P1IN&BIT6)==0) && ((P1IN&BIT7)==0) ) { //are both buttons
down?
                  P1OUT |= LED1;
                  P1OUT |= LED2;} //Yes: both LEDs are on

         else if ((P1IN&BIT7)==0) {    // button2 down?

                  P1OUT |= LED2;}   // LED2 is on

         else if ((P1IN&BIT6)==0) {    //Is button1 pushed?

                  P1OUT |= LED1; }  //Yes: LED1 is on

         else {      //No: are both buttons up?
                  P1OUT &= ~LED1;
                  P1OUT &= ~LED2; //Yes: both LEDs are off
                   }
          }  // end for
} // end main


( ((P1IN&BIT6)==0) && ((P1IN&BIT7)==0) ) {   //No: are both buttons down?

                  P1OUT |= LED1;
                  P1OUT |= LED2;} //Yes: both LEDs are on
```