HACETTEPE UNIVERSITY

ELECTRICAL AND ELECTRONICS ENGINEERING

ELE417 – EMBEDDED SYSTEM DESIGN

EXPERIMENT I – BASIC I/O USAGE AND EMBEDDED C PROGRAMMING

PRELIMINARY WORK I

2021-2022 SPRING

Student

Name: Egemen Can

Surname: Ayduğan

ID: 21728036

Date of Submission: 22.04.2022

Q1.)

I obtained The MSP-EXP430G2ET LaunchPad Development Kit.

My criteria for getting the MSP-EXP430G2ET LaunchPad Development Kit is an easy-to-use microcontroller development board for low-power and low-cost MSP430G2x MCUs. It has built-in emulation for programming and debugging and has a 14/20-pin DIP socket, built-in buttons and LEDs, and BoosterPack™ Plug-in Module pinouts that support a wide range of modules for additional functionality such as wireless, displays.

The model of the microcontroller in the launchpad is MSP430G2553. To summarize its general features, it has 16 MHz MCU with 16KB Flash, 512B SRAM, comparator, UART/SPI/I2C, timer.

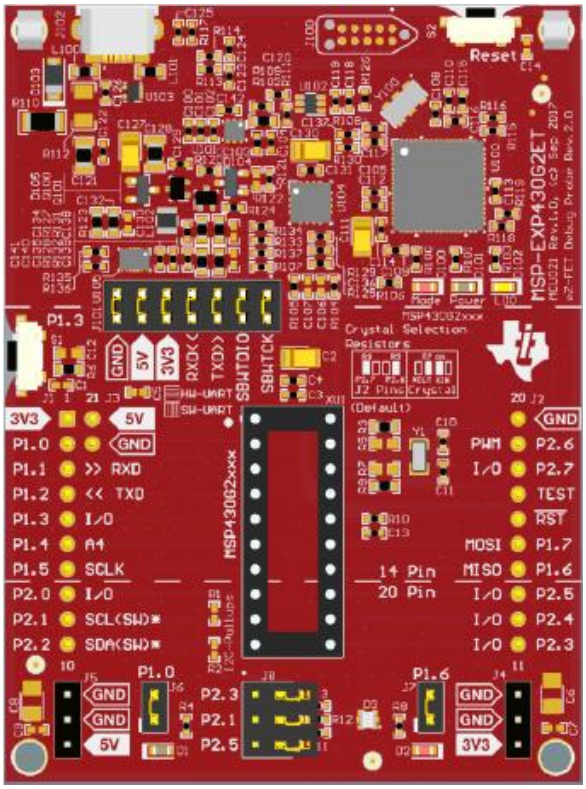


Fig.1. MSP-EXP430G2ET LaunchPad Development Kit.

Q2.)

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Table 1. Truth Table for XOR Operation

As can be seen from the table, when "XORing with 1" is performed, the 2's complement of the desired bits is obtained. In other words, if the input is high, the output is low, and if the input is low, the output is low.

```
int main(void){
    unsigned int MyNumber = 0x00;
    MyNumber ^= 0x01;               // MyNumber = 00000000 XOR 00000001 = 00000001
    MyNumber ^= 0x01;               // MyNumber = 00000001 XOR 00000001 = 00000000
    return(0);
}
```

Q3.)

To understand the state of a Bit, we can "AND with 1" because if we "AND with 1 on a bit" the result will be whatever the number is.

```
    MyNumber &= 1;              // MyNumber = MyNumber AND 1 = MyNumber

    // TO GET THE STATUS OF MULTIPLE BITS
    MyNumber &= 0x03;           // MyNumber = MyNumber AND 00000101
    MyNumber &= (BIT0|BIT2);    // MyNumber = MyNumber AND 00000101
```

Q4.)

Bouncing is the tendency of any two metal contacts in an electronic device to generate multiple signals as the contacts close or open; **debouncing** is any kind of hardware device or software that ensures that only a single signal will be acted upon for a single opening or closing of a contact.

In some applications, a return operation may be needed because a wavy signal may be harmful to our device, and it may also prevent us from getting the expected results even if the algorithm we have installed is correct.

```
#include <msp430.h>

#define Switch   BIT3                   // Switch -> P1.3
#define GreenLED BIT0                   // Green LED -> P1.0

void main(void) {
    WDTCTL = WDTPW | WDTHOLD;           // Stop Watchdog Timer

    P1DIR |= GreenLED;          // Set LED pin -> Output
    P1DIR &= ~Switch;           // Set SW pin -> Input
    P1REN |= Switch;            // Enable Resistor for SW pin
    P1OUT |= Switch;            // Select Pull Up for SW pin
    P1OUT &= Switch;            // P1 is zero without P1.3 because of Pull Up Conf.
    while(1){
        if(!(P1IN&Switch)){             // If SW is Pressed
            while(!(P1IN & Switch));    // Wait till Switch Released
            P1OUT ^= GreenLED;          // Toggle LED
        }
    }
}
```

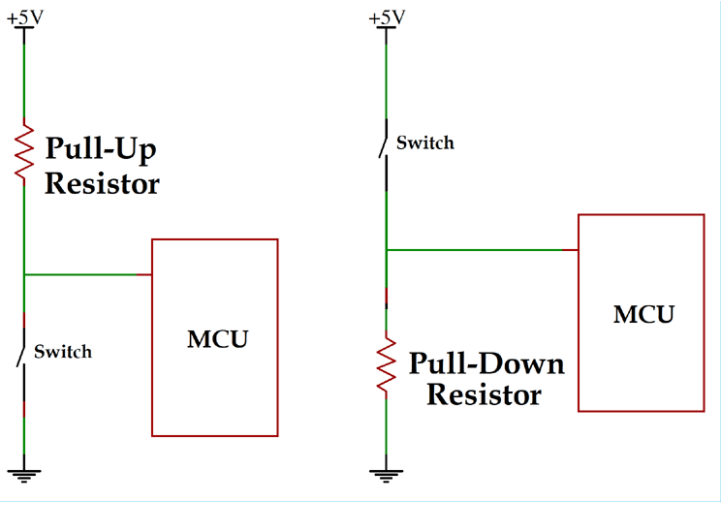
P1OUT	0x08
P7	0
P6	0
P5	0
P4	0
P3	1
P2	0
P1	0
P0	0

When the Green Led is OFF

P1OUT	0x09
P7	0
P6	0
P5	0
P4	0
P3	1
P2	0
P1	0
P0	1

When the Green Led is ON

Q5.)



A microcontroller in any embedded system utilizes I/O signals to communicate with the external devices. The simplest form of I/O is usually stated to as GPIO(General Purpose Input/Output). When the GPIO voltage level is low, then it is in high or high impedance state, then the pull up and pull-down resistors are used to ensure GPIO which is always in a valid state.

Some microcontrollers, including the MSP430, have these pull-up, pull-down resistors built-in. Thus, the operation can be performed using the software without the need for any external components.

```
P1REN |= Switch;          // Pull-Up/Pull-Down Resistors enabled for Button.
P1OUT |= Switch;          // Button is enabled as Pull-Up.
```

Q6.)

```
int main(void){
    WDTCTL = WDTPW | WDTHOLD;           // Stop Watchdog Timer
    volatile unsigned int Aydugan;
    volatile unsigned int Egemen_Can;
    volatile unsigned int i;

    Aydugan = 0;
    Egemen_Can = 19;                // ÇORUM  (0001 0011)

    for(i=0; i<4;i++){              // Final Egemen_Can = 0001 0011 0000 = 304
        Egemen_Can = Egemen_Can << 1;
    }

    Aydugan = Egemen_Can & 0x30;
    // 0001 0011 0000 And 0000 0011 0000 = 0011 0000 = 0x30
    return 0 ;
}
```

(x)= Aydugan	unsigned int	0	0x03F8
(x)= Egemen_Can	unsigned int	19	0x03FA
(x)= i	unsigned int	0	0x03FC

Initial Values

(x)= Aydugan	unsigned int	0	0x03F8
(x)= Egemen_Can	unsigned int	304	0x03FA
(x)= i	unsigned int	4	0x03FC

After Shifting Left 4 Times

Egemen_Can = (0001 0011)b << 4 = (0001 0011 0000)b = 0x130

(x)= Aydugan	unsigned int	48	0x03F8
(x)= Egemen_Can	unsigned int	304	0x03FA
(x)= i	unsigned int	4	0x03FC

Final Values

Aydugan = (0001 0011 0000)b AND (0000 0011 0000)b = (0011 0000)b = (0011 0000)b = 0x30

Q7.)

```
int main(void){

    WDTCTL = WDTPW | WDTHOLD;           // Stop watchdog timer
    P1SEL = 0x0000;                     // P1 became GPIO.
    P1DIR |= 0x0001;                    // P1.0 pin is output.
    P1OUT= 0x0000;                      // Starts with P1.0 = 0.

    volatile unsigned int i=0;
    while(1){
        if(i<2){
            P1OUT = P1OUT ^ 0001;       // 1sn ON, 1sn OFF
            __delay_cycles(1000000);    // Delay 1 second (Short Delay)
            i++;
        }
        if(i>=2){
            P1OUT = P1OUT ^ 0001;       // 3sn ON, 3sn OFF
            __delay_cycles(3000000);    // Delay 3 second (Long Delay)
            i++;
            if(i==4){
                i=0;
            }
        }
    }
}
```