CPE 325: Intro to Embedded Computer System

Lab06

Interrupt Service Routines with MSP430.

Submitted by: Dan Otieno

Date of Experiment: 02/15/22.

Report Deadline: 02/21/22.

Demonstration Deadline: 02/24/22.

Introduction

This lab involves testing ISR (Interrupt Service Routine) on the MSP430 and check reactions to external factors. The first part of this lab is written in assembly and the second part is in c language. In both cases we implement code to interface switches and observe outputs on LED 1 and LED 2. The codes are discussed further in this report.

Theory

Topics:

a) **Interrupts and Interrupt Vectors**: There's not much discussed in depth for this lab in the tutorial that was provided. In fact, we are advised that we will learn more about interrupts and interrupt vectors in future labs. But for the interrupts process in the MSP430, we must enable global interrupts in the status register, and then enable interrupts to occur for the bits on the desired port. Next, we specify whether the interrupt is called on a falling edge or rising edge and finally Initialize the interrupt flag by clearing it.

b) **Clock Module**: In the MSP430, when we change the content of the relevant clock module control registers, we can change the processor clock frequency and the frequency of other clock signals that are used for peripheral devices. MSP430 devices use an on-chip system clock called the FLL+ (frequency locked loop), which is an electronic control system that generates a signal locked to the frequency of an input or "reference" signal.

Results & Observation

Program 1:

Program Description:

This program required using assembly code to interface the LEDs on the MSP430. The first step was to blink LED 1 consistently and ensure that while SW2 was toggled to switch LED2 on and off, LED1 was not affected. My approach was to use the setup approach in the main loop that was shown as an example in the tutorial. I also decided, to use subroutines for the function to turn on LED 1 and to debounce switch 2. The function "SW1_sub" is called in main to blink LED1. And then I called the debounce function in the Interrupt Service Routine after checking if Switch 2 was pressed.

Program Output:

N/A: Ouputs to be demonstrated in lab, not to be observed in console.

Program 2:

Program Description:

Program 2 employed a slightly different approach to interfacing the LEDs, this time using c language. We set the microcontroller clock frequency to 2MHz and then toggle LED2 in the main loop at 1048576 cycles, this is achieved using the “delayed cycles” function inside an infinite loop. The next step was to set the frequencies for 4MHz and 8MHz in the ISR using the SCFQCTL clock function. The calculations to determine the frequencies are shown below:

- 2MHz: (63+1) * 32768.
- 4MHz: (127+1) * 32768.
- 8MHz: (255+1) * 32768

Program Output:

N/A: Ouputs to be demonstrated in lab, not to be observed in console.

Conclusion

In conclusion, these programs were a good way to be introduced into ISR. I would be looking forward to a more in-depth understanding of it in the MSP430. No flowcharts are included in this report, there were none requested.

Appendix

Your first code goes here, if any. Make sure you use a 1X1 table for this.

(Note: Make sure the code is readable, have comments. Also reduce spacing between lines to avoid lengthy reports.

Table 01: Program 1 MAIN CODE.

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.
                .ref  Debounce
                .ref  SW1_sub
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.
;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------
Setup:
                ; Configure Inputs: P1.0/P1.1.
                bic.b #002h, &P1DIR

                ; Configure Outputs: P2.1/P2.2.
            bis.b     #002h, &P2DIR                  ; LED 2(0x0002).

            bis.w     #GIE, SR                       ; Enable global interrupts.
            bis.b     #002h, &P1IES                  ; Set interrupts to call from hi to low
(SW1 and SW2).
            bis.b     #002h, &P1IE                   ; Enable port interrupts from bit 0.
                bic.b      #002h, &P1IFG                   ; Clear interrupt flag.
                call  #SW1_sub                  ; Call SW1 subroutine.

InfLoop:    jmp $                                          ; Loop until interrupt.
```

```
;-------------------------------------------------------------------------------
; P1_ISR: Interrupt Service Routine (ISR)
;-------------------------------------------------------------------------------
P1_ISR:
                bic.w      #GIE, SR                     ; Disable interrupts while
executing subroutine.
                bic.b      #002h, &P1IFG                ; Clear interrupt flag for SW2.

chkS2:          bit.b      #002h, &P1IN                 ; Check if SW2 is pressed
(0000_0002 on P1IN).
            jnz  l_exit                 ; If not zero, SW2 is not pressed, re-check after
loop.

            call #Debounce                    ; Call Debounce subroutine.

                bit.b      #002h, &P1IN                 ; Verify SW2 is still pressed.
            jnz  l_exit                 ; if not, wait until pressed.
LED_ON:         xor.b #002, &P2OUT                 ; Toggle LED 2.

l_exit:         reti                              ; Return from interrupt.

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
            .sect     ".int20"                   ;P1.x vector
            .short    P1_ISR

            .end
```

Table 02: Program 1: LED BLINK SUBROUTINE.

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file
```

```
;-------------------------------------------------------------------------------
            .def    SW1_sub

            .text                               ; Assemble into program memory.

;-------------------------------------------------------------------------------
; Switch 1 Subroutine
;-------------------------------------------------------------------------------
SW1_sub:
           bis.b #004h,&P2DIR               ; Set P2OUT to 0x0000_0100 (LEDS off)
InfLoop:   mov.w      #0xFFFF, R5                ; Software delay (65,535*16cc/2^20 ~ 1s)

SWDelay1:   nop                                 ; 1cc (total delay is 16 cc)
            nop
            nop
            nop
            nop
            nop
            nop
            nop
            nop
            nop
            nop
            nop
            nop
            dec.w   R5                          ; 1cc
            jnz     SWDelay1                    ; 2cc

            xor.b   #004h, P2OUT                ; toggle LED 1.
            jmp     InfLoop                     ; goto InfLoop

            ret                             ; return from Subroutine.

            .end
```

Table 03: Program 1: DEBOUNCE SUBROUTINE

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def Debounce
            .text                               ; Assemble into program memory.

Debounce:  mov.w      #2000, R15                     ; Set to (2000 * 10cc).

Db20ms:          dec.w      R15                      ; Debounce frequency = 0.5Hz = 2000ms = 
2s (1s on/1s off).
            nop                                 ;
            nop                                 ;
            nop                                 ;
            nop                                 ;
            nop                                 ;
            nop                                 ;
            nop                                 ;
            jnz  Db20ms                         ;;



                 ret

                      .end
```

Table 04: Program 2 Source Code

```
#include <msp430.h>
#define   SW1_PRESSED ((BIT0&P1IN)==0)
#define   SW2_PRESSED ((BIT1&P1IN)==0)

void main(void)
{
    WDTCTL = WDTPW+WDTHOLD;             // Stop WDT

    FLL_CTL0 |= XCAP18PF;              // Set load capacitance for xtal
    SCFI0 |= FN_2;                     // DCO range control
    SCFQCTL = 63;                      // (63+1) x 32768 = ~2Mhz
    P2DIR |= BIT1;                      // Set LED1 as output

    _EINT();                            // enable interrupts
    P1IE |= BIT0+BIT1;                       // P1.0 interrupt enabled
    P1IES |= BIT0+BIT1;                      // P1.0 hi/low edge
    P1IFG &= ~BIT0+BIT1;                     // P1.0 IFG cleared


    for(;;)
    {
        __delay_cycles(1048576);
        P2OUT ^= BIT1;             // LED1 is turned off
    }
}

//Port 1 interrupt service routine
#pragma vector = PORT1_VECTOR
__interrupt void Port1_ISR (void)
{
    P1IFG &= ~(BIT0+BIT1);                   // P1.0 IFG cleared
    if(SW1_PRESSED)
    {
        SCFQCTL = 127;               // (127+1) x 32768 = ~4Mhz
    }
    else if(SW2_PRESSED)
    {
        SCFQCTL = 255;              // (255+1) x 32768 = ~8Mhz
    }
}
```