Austin Bumbalough
CPE 325-08
Lab 06
10/08/2019

Lab 6 Solution

Part 1 (Assembly)

In lab 6 part 1, I wrote an assembly program that uses interrupts triggered by on-board switches to manipulate two on-board LEDs.

When SW2 is pressed while LED1 is blinking nothing happens. Pressing SW2 does not disrupt the blinking LED. SW2 is functional, however, and this is not a program error. By default, when the MSP 430 enters an interrupt service routine (ISR) all other interrupts are disabled for the duration of the subroutine. The programmer has the option to explicitly enable interrupts during an ISR if such functionality is required.

Source Code

```
            .cdecls C,LIST,"msp430.h"       ; Include device header file
;-------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.
;-------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that hav
                                            ; 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:      bic.b #0x03, &P1DIR         ; Configure P1.0 and P1.1 as inputs
            bis.b #0x06, &P2DIR         ; Configure P2.1 and P2.2 as ouputs
            bic.b #0xFF, &P2OUT         ; Initialize LEDs to off
            bis.w #GIE, SR              ; Enable global interrupts
            bis.b #0x03, &P1IES         ; Trigger interrupts on falling edge s
            bic.b #0x03, &P1IFG         ; Clear any pending interrupts
            bis.b #0x03, &P1IE          ; Enable interrupts on P1.0 and P1.1
            jmp $
;-------------------------------------------------------------------------
; Port 1 Interrupt Service Routine
;-------------------------------------------------------------------------
P1_ISR:     push R15
            bic.w #GIE, SR              ; Disable interrupts while executing s
            bit.b #0x01, &P1IFG         ; Check for SW1 interrupt
            jnz S1                      ;
            bit.b #0x02, &P1IFG         ; Check for SW2 interrupt
            jnz S2                      ;
            bic.b #0xFF, &P1IFG         ; Clear P1IFG
            jmp ISR_exit
S1:         bit.b #0x01, &P1IN          ; Check if SW1 is pressed
            jnz ISR_exit                ;
            mov.w #2000, R15            ; (2 * 100) * 10 cc per loop
DbncSW1:    dec.w R15                   ; 2 ms debounce
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            jnz DbncSW1                 ;
            bit.b #0x01, &P1IN          ; Check if SW1 is still pressed
            jnz ISR_exit                ;
            call #SW1                   ;
            xor.b #0x02, &P2OUT         ;
            bic.b #0x01, &P1IFG         ; Clear interrupt flag for SW1
            jmp ISR_exit                ;
S2:         bit.b #0x02, &P1IN          ; Check if SW2 is pressed
            jnz ISR_exit                ;
            mov.w #2000, R15            ;
DbncSW2:    dec.w R15                   ;
            nop                         ;
            nop                         ;
```

```
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            jnz DbncSW2                 ;
            bit.b #0x02, &P1IN          ; Check if SW2 is still pressed
            jnz ISR_exit                ;
            xor.b #0x02, &P2OUT         ; Toggle LED 2
            bic.b #0x02, &P1IFG         ;
ISR_exit:   pop R15                     ;
            reti                        ;
;-------------------------------------------------------------------------------
; Switch 1 Subroutine
;-------------------------------------------------------------------------------
SW1:        push R14                    ;
            push R15                    ;
            mov.w #6, R14               ; LED 1 toggle counter
sw1_loop:   mov.w #16667, R15           ;
sw1_delay:  dec.w R15                   ; 167 ms delay
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            nop                         ;
            jnz sw1_delay               ;
            xor.b #0x04, &P2OUT         ; Toggle LED 1
            dec.w R14                   ;
            jnz sw1_loop                ;
            pop R15                     ;
            pop R14                     ;
            ret                         ;
;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"            ; MSP430 RESET Vector
```

```
.short  RESET
.sect   ".int20"
.short  P1_ISR
.end
```

Part 2 (C)

In lab 6 part 1, I wrote a C program that uses interrupts triggered by on-board switches to manipulate two on-board LEDs.

LED Blinking Rate

Since the delay loop is a constant 50,000 iterations, the delay time is a function of the processor clock speed. The delay in milliseconds is $D(f) = \frac{5 * 10^8}{f}$

Clock Frequency (MHz)	Delay Time (ms)	LED Frequency (Hz)
1	500	1
2	250	2
4	125	4
8	75	8

Source Code

```
int clkFreq = 1;

int main(void)
{
    // Setup Ports and Registers
    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer
    P1DIR = 0x00; // P1 set to input
    P2DIR |= BIT2+BIT1; // P2.1 and P2.2 set to output
    P2OUT = BIT2; // LED 1 on, LED 2 off
```

```
    P1IES |= BIT1+BIT0; // Interrupt triggered on falling edge for P1.1 and P1
    P1IE |= BIT1+BIT0; // Interrupts enabled on P1.1 and P1.0
    P1IFG &= ~(BIT1+BIT0); // Clear any pending interrupts
    _EINT(); // Enable global interrupts

    while(1) {
        for (long int i=50000;i>0;i--);
        P2OUT ^= BIT2+BIT1;
    }

    return 0;
}

#pragma vector = PORT1_VECTOR
__interrupt void P1ISR(void) {
    switch (P1IFG & (BIT1+BIT0)) {
        case (BIT0): // SW1 pressed, frequency increases
            switch (clkFreq) {
                case 1:
                    clkFreq = 2;
                    SCFQCTL = 60; // f = (60+1) * 32768 = 1.99 MHz
                    break;
                case 2:
                    clkFreq = 4;
                    SCFQCTL = 121; // f = (121+1) * 32768 = 3.99 MHz
                    break;
                case 4:
                    clkFreq = 8;
                    SCFI0 |= BIT2; // Adjust DCO range up to 1.3-12.1 MHz
                    FLL_CTL0 |= DCOPLUS; // Enable FLL+ loop divider, doubles
                    break;
                case 8:
                    // Do nothing, frequency is at maximum value
                    break;
            }
            P1IFG &= ~BIT0;
            break;
        case(BIT1): // SW2 pressed, frequency decreases
            switch (clkFreq) {
                case 1:
                    // Do nothing, frequency is at minimum value
                    break;
                case 2:
                    clkFreq = 1;
```

```
                    SCFQCTL = 30; // f = (30+1) * 32768 = 1.01 MHz
                    break;
                case 4:
                    clkFreq = 2;
                    SCFQCTL = 60; // f = (60+1) * 32768 = 1.99 MHz
                    break;
                case 8:
                    clkFreq = 4;
                    SCFI0 &= ~BIT2; // Adjust DCO range down to 0.65-6.1 MHz
                    FLL_CTL0 &= ~DCOPLUS; // Disable FLL+ loop divider, halves
                    break;
            }
            P1IFG &= ~BIT1;
            break;
    }
}
```

Part 3 (Bonus)

In lab 6 part 3, I reimplemented the part 1 solution using C instead of assembly. I also extended the program so that pressing both switches at the same time would cause both LEDs to light up and stay lit while the switches were held. Upon releasing the switches, the LEDs go to the off state.

Source Code

```
int main(void) {
    // Setup Ports and Registers
    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer
    P1DIR = 0x00; // P1 set to input
    P2DIR |= BIT2+BIT1; // P2.1 and P2.2 set to output
    P2OUT &= ~(BIT2+BIT1); // LED 1 and LED 2 off initially
    P1IES |= BIT1+BIT0; // Interrupt triggered on falling edge for P1.1 and P1
    P1IE |= BIT1+BIT0; // Interrupts enabled on P1.1 and P1.0
    P1IFG &= ~(BIT1+BIT0); // Clear any pending interrupts
    _EINT(); // Enable global interrupts

    while(1);
    return 0;
```

```
}

#pragma vector = PORT1_VECTOR
__interrupt void P1ISR(void) {
    for (int i=5000;i>0;i--); // Hard to press both buttons at the exact same
    switch (P1IFG & (BIT1+BIT0)) { // so add 5 ms delay before processing inte
        case (BIT0): // SW1 pressed
                for (int i=10000;i>0;i--);
                if ((SW1) == 0) {
                    for (int j=3;j>0;j--) {
                        for (int k=16667;k>0;k--);
                        P2OUT ^= BIT2;
                        for (int k=16667;k>0;k--);
                        P2OUT ^= BIT2;
                    }
                    P2OUT ^= BIT1;
                }
                P1IFG &= ~BIT0; // Clear interrupt flag
                break;
        case(BIT1): // SW2 pressed
                for (int i=10000;i>0;i--);
                if ((SW2) == 0) {
                    P2OUT ^= BIT1;
                }
                P1IFG &= ~BIT1; // Clear interrupt flag
                break;
        case(BIT1+BIT0): // Both switches pressed
                for (int i=10000;i>0;i--);
                if (((SW1) == 0) & ((SW2) == 0)) {
                    P2OUT |= BIT2+BIT1;
                }
                while (((SW1) == 0) & ((SW2) == 0));
                P2OUT &= ~(BIT2+BIT1);
                P1IFG &= ~(BIT1+BIT0);
                break;
    }
}
```