CPE 325: Embedded Systems Laboratory

Lab06

MSP430 Interrupts and Universal Clock Subsystem

Submitted by: Esther Shore

Date of Experiment: June 21, 2023

Report Deadline: June 27, 2023

Demonstration Deadline: July 5, 2023

**Introduction**

This lab introduces the topics of interrupts and interrupts vectors in addition to the clock module in MSP430. Software interrupts are implemented in the programs in order to handle the switch presses/releases. Both assembly and C were used to demonstrate how interrupts and software delays work. Changing the clock frequency of the MSP430 is also discovered and examined how it affects a blinking LED at a constant number of cycles.

**Theory Topics**

1. Interrupts and interrupt vectors

Interrupts are essentially a break from the main program when certain conditions are met, executing the interrupt service routine, and then immediately transferring control back where the main program left off. They are able to handle events asynchronously, effectively managing events that are reliant on time. Interrupts are often used to detect events such as switch presses.

Interrupt vectors point to the corresponding ISR for the interrupt source. These are often established for a specific port. Some of the I/O ports on the MSP430 have interrupt capabilities that are configurable, which detect interrupts on that port. Interrupts can be enabled or disabled for specific bits and configured to detect rising or falling edges for the specified bit(s).

1. Clock module in MSP430

The clock module in MSP430 is a dedicated module that provides the option to control the processor clock frequency and other clock signals that are used for peripheral devices. It can be used to change clock frequencies by setting the relevant clock module control register(s). Specifically for the MSP430, the system clock control register is used to change the processor clock frequency and the system clock frequency integrator register is used to set the DCO control range. Use of the FLL+ control register is also necessary to specify oscillator capacitor and DCO output divided/not divided. There are four clock signals available from the FLL+ module on the MSP430.

**Program 1 (Interfacing with Interrupts)**

***Program Description:***

This assembly program toggles LED1 every second, resulting in a blink frequency of 0.5Hz. The LED continues to blink at this frequency no matter what, remaining unaffected by any switch presses. This was implemented by creating a main loop with a 1ms delay and having counters for the loop, press debounce, and release debounce. The main loop counter is set for 1000 iterations or 1000ms, while the press and release debounce counters are set for 20 iterations or 20ms when instantiated. The interrupt is used to detect SW2 changes, which toggles LED2 when pressed. By not using any delays within the interrupt itself, the program is able to handle the switch press asynchronously and return to the main loop as soon as possible.

**Program 2 (Changing clock frequencies)**

***Program Description:***

This C program changes the clock frequency of the MSP430FG4618 experimenter board based on switch presses. The LED2 is also set to toggle approximately every 1M clock cycles, resulting in it blinking based at different rates depending on the processor clock frequency. When SW1 is pressed, the clock frequency is changed to ~4MHz and when SW2 is pressed, the clock frequency is changed to ~8MHz. The default clock frequency for the program is also set to ~2MHz. Higher clock frequencies result in a higher blink frequency for the LED. It uses interrupts to detect the switch presses and releases.

***Program Output:***

Referring to the following two lines of C code:

FLL\_CTL0 |= DCOPLUS + XCAP18PF; // DCO+set, freq = xtal x D x N+1

SCFI0 |= FN\_2 + FLLD\_2; // DCO range control

Since the DCOPLUS bit in the FLL\_CTL0 register and FLLD\_2 bit in the SCFI0 register are set, it is specified that f\_DCOCLK = D \* (N + 1) \* f\_ACLK where D = 2. By rearranging the equation, the N multiplier can be found to set the SCFQCTL register based on the desired clock frequency.

Calculations:

Set frequency ~2MHz: N = 2097152/(32768\*2) - 1 = 31

Set frequency ~4MHz: N = 4194304/(32768\*2) - 1 = 63

Set frequency ~8MHz: N = 8388608/(32768\*2) - 1 = 127

Since the delay between the LED toggle is set to 1048576 clock cycles, the blinking frequency can be calculated given each clock frequency as follows:

Clock freq ~2MHz: 1048576/2097152 = 0.5s ON and 0.5s OFF => 1 cycle/1 sec = 1Hz

Clock freq ~4MHz: 1048576/4194304 = 0.25s ON and 0.25s OFF => 1 cycle/0.5 sec = 2Hz

Clock freq ~8MHz: 1048576/8388608 = 0.125s ON and 0.125s OFF => 1 cycle/0.25 sec = 4Hz

**Program 3 (Bonus)**

***Program Description:***

This program extends Program 1 by rewriting it from assembly to C and enabling toggling of LED1 blinking through S1 press. When S1 is pressed, the program pauses the blinking of LED1 at whatever the last state was (ON/OFF) and resumes once S1 is pressed again. A second interrupt for Port 2 was utilized to handle the switch.

**Conclusion**

This was probably the most difficult lab yet in terms of content, implementation, and necessary time taken. Primarily, interrupts were a very foreign topic and being able to detect switch presses from different switches without affecting the main loop posed a harsh learning curve. Despite these difficulties, it was beneficial to be exposed to the concepts of interfacing with interrupts as well as changing clock frequencies.

***Appendix:***

**Table 1:** Program 1 Source Code

| ;-------------------------------------------------------------------------------  ; File: Lab6\_P1.asm  ; Description: Program blinks LED1 at constant frequency 0.5Hz and pressing  ; S2 toggles LED2. Uses ISR to handle switch and debounce delay  ; in main loop.  ; Clocks: ACLK = 32.768kHz, MCLK = SMCLK = default DCO = 2^20=1,048,576 Hz  ; Platform: TI LaunchPad MSP430F5529  ; Author: Esther Shore  ; Date: June 27, 2023  ;-------------------------------------------------------------------------------  .cdecls C,LIST,"msp430.h" ; Include device header file  ;-------------------------------------------------------------------------------  .def RESET ; Export program entry-point to  ; make it known to linker.  .def S2\_ISR  ;-------------------------------------------------------------------------------  .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 stack pointer  StopWDT: mov.w #WDTPW|WDTHOLD, &WDTCTL ; Stop watchdog timer  ;-------------------------------------------------------------------------------  Setup:  bis.b #01h, &P1DIR ; set P1.0 to output LED1  bic.b #01h, &P1OUT ; clear P1OUT BIT0, LED1 OFF  bis.b #80h, &P4DIR ; set P4.7 to output LED2  bic.b #80h, &P4OUT ; clear P4OUT BIT7, LED2 OFF  bic.b #02h, &P1DIR ; set P1.1 as input for S2  bis.b #02h, &P1REN ; enable pull-up resistor at P1.1  bis.b #02h, &P1OUT ; required for proper I/O setup  bis.w #GIE, SR ; enable Global Interrupts  bis.b #02h, &P1IE ; enable Port 1 interrupt from bit 1  bis.b #02h, &P1IES ; set interrupt to call from hi to low  bic.b #02h, &P1IFG ; clear interrupt flag  mov.w #1000, R5 ; 1s delay counter for LED1 (1000ms = 1s)  clr.w R9 ; press debounce counter  clr.w R10 ; release debounce counter  InfLoop:  mov.w #100, R4 ; 100 \* 10cc = 1000cc = 1ms  Loop1ms: dec.w R4 ; decrement delay counter  nop  nop  nop  nop  nop  nop  nop  jnz Loop1ms ; continue to loop until 1ms  dec.w R5 ; decrement LED1 delay  jz LED1tog ; once zero, toggle LED1  ChckPress: cmp.w #0, R9 ; check if press debounce zero  jne Press ; if not zero, go to Press func  ChckRelease:cmp.w #0, R10 ; check if release debounce  jeq InfLoop ; if zero, loop back to top  bit.b #02h, &P1IN ; else, check if S2 released  jz InfLoop ; if not released (0), loop  dec.w R10 ; decrement release counter  cmp.w #0, R10 ; check if release debounce zero  jne InfLoop ; if not zero, loop  bic.b #02h, &P1IFG ; clear interrupt flag for S2  bis.b #02h, &P1IE ; enable Port 1 interrupt from bit 1  jmp InfLoop ; main loop  LED1tog: xor.b #01h, P1OUT ; toggle LED1  mov.w #1000, R5 ; reset delay counter  jmp ChckPress ; continue to check press  Press: dec.w R9 ; decrement press debounce counter  cmp.w #0, R9 ; check if press debounce zero  jne ChckRelease ; if not zero, go to ChckRelease  mov.w #20, R10 ; set release debounce to 20ms  jmp ChckRelease ; jump to ChckRelease label  ;----------------------------------------------------------------------------  ; P1.1 (S2) interrupt service routine (ISR)  ;----------------------------------------------------------------------------  S2\_ISR: bit.b #02h, &P1IN ; check if S2 is pressed  jnz LExit ; if not zero, S2 is not pressed  bic.b #02h, &P1IE ; forbid interrupts for S2  xor.b #80h, P4OUT ; toggle LED2  bic.b #02h, &P1IFG ; clear interrupt flag for S2  mov.w #20, R9 ; set press debounce counter to 20  LExit: reti ; return from interrupt  ;-------------------------------------------------------------------------------  ; Stack Pointer definition  ;-------------------------------------------------------------------------------  .global \_\_STACK\_END  .sect .stack  ;-------------------------------------------------------------------------------  ; Interrupt Vectors  ;-------------------------------------------------------------------------------  .sect ".reset" ; MSP430 RESET Vector  .short RESET  .sect ".int47" ;P1.x Vector  .short S2\_ISR  .end |
| --- |

**Table 2:** Program 2 Source Code

| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* File: Lab6\_P1.c  \* Description: The program detects when SW1 or SW2 are pressed and changes  \* the microcontroller frequency accordingly.  \* P1\_ISR is used to detect both switch presses and releases.  \* Board: MSP430FG461x/F20xx Experimenter Board  \* Clocks: ACLK = 32.768kHz, MCLK = SMCLK = default DCO  \* Author: Esther Shore  \* Date: June 27, 2023  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <msp430.h>  unsigned char SW1pressed = 0; // SW1 status (0 not pressed, 1 pressed)  unsigned char SW2pressed = 0; // SW2 status (0 not pressed, 1 pressed)  void main(void) {  WDTCTL = WDTPW+WDTHOLD; // Stop WDT  P2DIR |= BIT2; // Set LED1 as output  P2DIR |= BIT1; // Set LED2 as output  P2OUT = 0x00; // Clear LED1, LED2 status  SW1pressed = 0; // Initialize SW1 status  SW2pressed = 0; // Initialize SW2 status  \_EINT(); // Enable interrupts  P1IE |= BIT0 | BIT1; // P1IE.BIT0 and BIT1 interrupt enabled  P1IES |= BIT0 | BIT1; // P1IES.BIT0 and BIT1 detect hi->low edge  P1IFG &= ~(BIT0 | BIT1); // P1IFG.BIT0 and BIT1 cleared  FLL\_CTL0 |= DCOPLUS + XCAP18PF; // DCO+set, freq = xtal x D x N+1  SCFI0 |= FN\_2 + FLLD\_2; // DCO range control  SCFQCTL = 31; // set frequency ~2MHz  // 2097152/(32768\*2) - 1 = 31  while(1) { // infinite loop  P2OUT ^= BIT1; // toggle LED2  \_\_delay\_cycles(1048576); // delay by ~1M clock cycles  }  }  // Port 1 interrupt service routine  #pragma vector = PORT1\_VECTOR  \_\_interrupt void Port1\_ISR (void) {  if (P1IFG&BIT0) { // if interrupt from P1.0 (SW1)  if (SW1pressed == 0) { // if SW1 used to be not pressed  SW1pressed = 1; // now pressed, toggle switch flag  SCFQCTL = 63; // set ~4MHz, 4194304/(32768\*2) - 1 = 63  P1IFG &= ~BIT0; // P1IFG.BIT0 is cleared  P1IES &= ~BIT0; // P1IES.BIT0 low/high edge  } else if (SW1pressed == 1) { // SW1 used to be pressed  SW1pressed = 0; // now set released state  SCFQCTL = 31; // back to default frequency  P1IFG &= ~BIT0; // P1IFG.BIT0 is cleared  P1IES |= BIT0; // P1IES.BIT0 hi/low edge  }  }  else if (P1IFG&BIT1) { // if interrupt from P1.1 (SW2)  if (SW2pressed == 0) { // if SW2 was previously not pressed  SW2pressed = 1; // now set flag pressed  SCFQCTL = 127; // set ~8MHz, 8388608/(32768\*2) - 1 = 127  P1IFG &= ~BIT1; // P1IFG.BIT1 is cleared  P1IES &= ~BIT1; // P1IES.BIT1 low/high edge  } else if (SW2pressed == 1) { // if SW2 previously pressed  SW2pressed = 0; // now released  SCFQCTL = 31; // back to default frequency  P1IFG &= ~BIT1; // P1IFG.BIT1 is cleared  P1IES |= BIT1; // P1IES.BIT1 hi/low edge  }  }  } |
| --- |

**Table 3:** Program 3 Bonus Source Code

| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* File: Lab6\_P3.c  \* Description: The program extends Lab06\_P1.asm in C and enables toggling of  \* the blinking LED1 by S1 press.  \* Port1\_ISR is used to detect both switch presses and releases.  \* Board: MSP430F5529 LaunchPad  \* Clocks: ACLK = 32.768kHz, MCLK = SMCLK = default DCO  \* Author: Esther Shore  \* Date: June 27, 2023  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <msp430.h>  #define S1 P2IN&BIT1  #define S2 P1IN&BIT1  unsigned int loop\_counter = 1000; // loop counter for 0.5Hz blink (1000ms = 1s)  unsigned int press\_db\_counter = 0; // initialize press debounce counter  unsigned int release\_db\_counter = 0; // intialize release debounce counter  unsigned int blink = 1; // blink state: 1 is True, 0 is False  void main(void) {  WDTCTL = WDTPW + WDTHOLD; // stop WDT  P1DIR |= BIT0; // set LED1 as output  P1OUT &= ~BIT0; // clear Port1 status  P4DIR |= BIT7; // set LED2 as output  P4OUT &= ~BIT7; // clear Port4 status  P1DIR &= ~BIT1;  P1REN |= BIT1;  P1OUT |= BIT1; // setup S2  P2DIR &= ~BIT1;  P2REN |= BIT1;  P2OUT |= BIT1; // setup S1  \_EINT(); // enable interrupts  P1IE |= BIT1; // P1IE.BIT1 interrupt enabled  P1IES |= BIT1; // P1IES.BIT1 detect hi->low edge  P1IFG &= ~BIT1; // P1IFG.BIT1 is cleared  P2IE |= BIT1; // P2IE.BIT1 interrupt enabled  P2IES |= BIT1; // P2IES.BIT1 detect hi->low edge  P2IFG &= ~BIT1; // P2IFG.BIT1 is cleared  while(1) { // infinite loop  \_\_delay\_cycles(1000); // delay by 1000cc = 1ms at 1MHz clock freq  loop\_counter--;  if (loop\_counter == 0) { // once 1000 loops of 1ms delays  if (blink == 1) {  P1OUT ^= BIT0; // toggle LED1  }  loop\_counter = 1000; // reset loop counter  }  if (press\_db\_counter > 0) {  press\_db\_counter--;  if (press\_db\_counter == 0) {  release\_db\_counter = 20;  }  }  if (release\_db\_counter > 0 && S2 != 0) {  release\_db\_counter--;  if (release\_db\_counter == 0) {  P1IFG &= ~BIT1; // S2 interrupt is cleared  P1IE |= BIT1; // S2 interrupt enabled  P2IFG &= ~BIT1; // S1 interrupt is cleared  P2IE |= BIT1; // S1 interrupt enabled  }  }  }  }  // Port 1 interrupt service routine  #pragma vector = PORT1\_VECTOR  \_\_interrupt void Port1\_ISR (void) {  if (S2) {  P1IE &= ~BIT1; // S2 interrupt disabled  P4OUT ^= BIT7; // toggle LED2  P1IFG &= ~BIT1; // S2 interrupt is cleared  press\_db\_counter = 20;  }  }  // Port 2 interrupt service routine  #pragma vector = PORT2\_VECTOR  \_\_interrupt void Port2\_ISR (void) {  if (S1) {  P2IE &= ~BIT1; // S1 interrupt disabled  P2IFG &= ~BIT1; // S1 interrupt is cleared  press\_db\_counter = 20;  if (blink == 1) { // set new blink state  blink = 0;  } else if (blink == 0) {  blink = 1;  }  }  } |
| --- |