CPE 325: Intro to Embedded Computer System

Lab 06

Interrupts in C, MSP430 Clock Subsystem

Submitted by: Anshika Sinha

Date of Experiment:__February 12, 2025_____

Report Deadline:____February 18, 2025_____

Demonstration Deadline: ___February 24, 2025_________

Theory

Topic 1: Interrupts and Interrupt Vectors

a) Interrupts allow a microcontroller to temporarily pause its main execution and handle specific events automatically.

b) When an interrupt occurs, the processor jumps to a predefined Interrupt Service Routine (ISR) to handle the event.

c) After execution, it resumes from where it left off using the RETI (Return from Interrupt) instruction.

d) To set up an interrupt for an I/O port, the following steps must be performed:

   a. Enable Global Interrupts: Set the GIE (General Interrupt Enable) bit in the status register (SR).

   b. Enable Specific Interrupts: Enable interrupts for the required port and pin

   c. Specify the Interrupt Edge: Define whether the interrupt triggers on a falling edge (high-to-low transition) or rising edge (low-to-high transition) using P1IES |= BIT1;.

   d. Clear Interrupt Flag: Ensure the interrupt flag is cleared at initialization

e) The Interrupt Vector Table (IVT) stores addresses of ISRs. The PORT1 Vector is ".int47" and PORT2 Vector is ".int42" for MSP430.

Topic 2: Clock module in MSP430

a) The MSP430 microcontroller family provides flexible clocking options through its Unified Clock System (UCS). The UCS allows users to control processor and peripheral clock frequencies by configuring various clock sources and signals.

b) The MSP430F5529 uses a digitally controlled oscillator (DCO) to generate clock frequencies. The clock system consists of UCS (Unified Clock System), where UCSCTL1 controls the DCO range and UCSCTL2 sets the DCO multiplier.

c) Clock Sources

   1. XT1CLK:
      - Low or high-frequency oscillator (32.768 kHz crystal or 4-32 MHz external source).
      - Can serve as a reference for the Frequency Locked Loop (FLL).
   2. VLOCLK (Very Low Power Oscillator):
      - Internal, low-power oscillator (~10 kHz typical).
   3. REFOCLK (Reference Oscillator):
      - Internal oscillator with a fixed 32.768 kHz frequency.
      - Used as a reference for FLL.
   4. DCOCLK (Digitally Controlled Oscillator):
      - Internal clock, stabilized by FLL, with software-configurable frequency.

5. XT2CLK:
    - ○ Optional high-frequency oscillator (4-32 MHz).
    - ○ Used for high-speed applications.

d) Clock sources are used to generate three primary clock signals:

1. ACLK (Auxiliary Clock)
    - ○ Can be sourced from XT1CLK, REFOCLK, VLOCLK, or DCOCLK.
    - ○ Often used for low-power peripherals like timers.
2. MCLK (Master Clock)
    - ○ Drives the CPU and system.
    - ○ Selectable from XT1CLK, REFOCLK, VLOCLK, DCOCLK, or XT2CLK.
3. SMCLK (Subsystem Master Clock)
    - ○ Used for peripherals such as timers and communication modules.
    - ○ Configurable similar to MCLK.

e) Configuring the UCS Module

a. UCS settings are controlled using UCSCTL0 - UCSCTL8 registers.
b. Digital Controlled Oscillator (DCO) Configuration
c. Frequency Locked Loop (FLL) Behavior

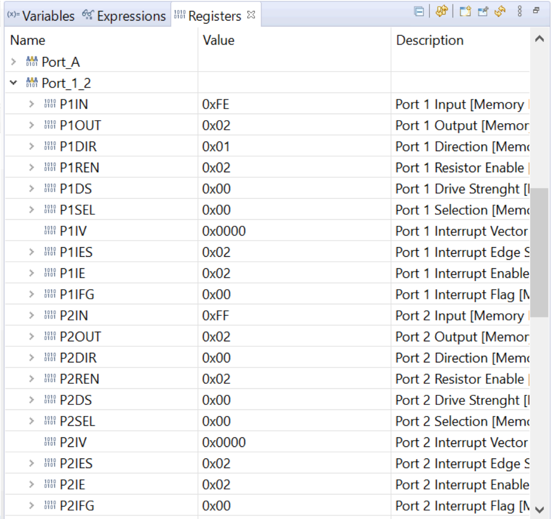
Results & Observation

Program 1:

Program Description:

This MSP430 assembly program interfaces Switch 1, Switch 2, and LED1 and LED2. An interrupt service routine is used to interface the switches. Initially, both LED1 and LED2 start in the OFF state. When SW1 is first pressed, LED2 turns on and changes state with each subsequent press. When SW2 is pressed, LED1 blinks 5 times at 5 Hz, then changes the state of LED2.

Program Output:



Name	Value	Description
Port_A		
Port_1_2		
P1IN	0xFE	Port 1 Input [Memory
P1OUT	0x02	Port 1 Output [Memor
P1DIR	0x01	Port 1 Direction [Mem
P1REN	0x02	Port 1 Resistor Enable
P1DS	0x00	Port 1 Drive Strenght [
P1SEL	0x00	Port 1 Selection [Mem
P1IV	0x0000	Port 1 Interrupt Vector
P1IES	0x02	Port 1 Interrupt Edge S
P1IE	0x02	Port 1 Interrupt Enable
P1IFG	0x00	Port 1 Interrupt Flag [M
P2IN	0xFF	Port 2 Input [Memory
P2OUT	0x02	Port 2 Output [Memor
P2DIR	0x00	Port 2 Direction [Mem
P2REN	0x02	Port 2 Resistor Enable
P2DS	0x00	Port 2 Drive Strenght [
P2SEL	0x00	Port 2 Selection [Mem
P2IV	0x0000	Port 2 Interrupt Vector
P2IES	0x02	Port 2 Interrupt Edge S
P2IE	0x02	Port 2 Interrupt Enable
P2IFG	0x00	Port 2 Interrupt Flag [M

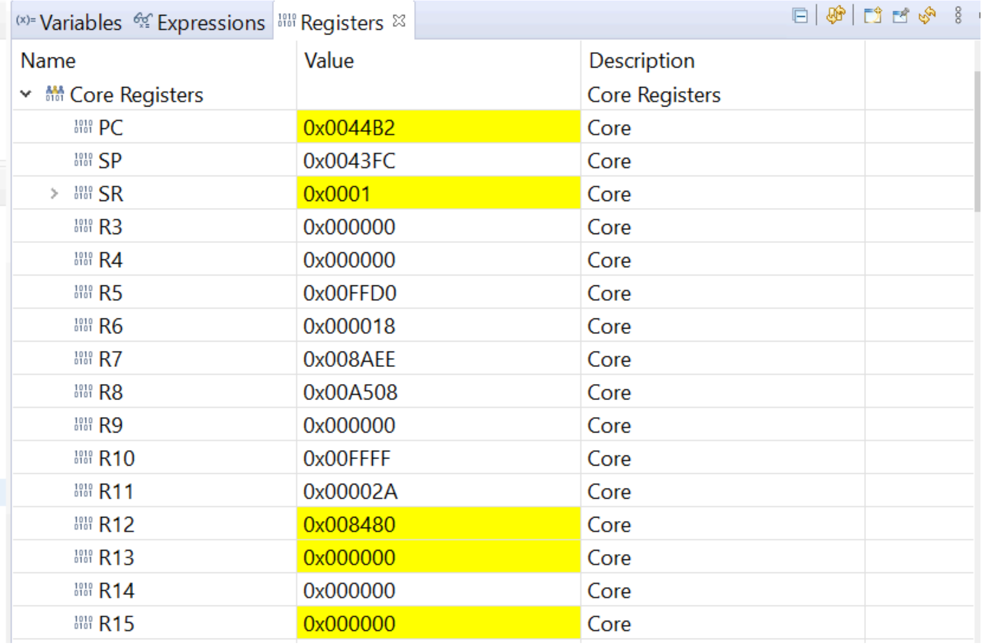
Figure 01: Program 1 output

Program 2:

Program Description:

This C program interfaces Switch 1, Switch 2, and the two LEDs. Initially, LED1 is on and LED2 is off. The clock frequency is set to 2 MHz. LED1 and LED2 blink using a 50,000 interaction loop delay. Every time SW2 is pressed, the clock frequency is set to 10 MHz. Everytime SW1 is pressed, the clock frequency is halved, and it cannot go under 1 MHz.

Program Output:



Name	Value	Description
Core Registers		Core Registers
PC	0x0044B2	Core
SP	0x0043FC	Core
SR	0x0001	Core
R3	0x000000	Core
R4	0x000000	Core
R5	0x00FFD0	Core
R6	0x000018	Core
R7	0x008AEE	Core
R8	0x00A508	Core
R9	0x000000	Core
R10	0x00FFFF	Core
R11	0x00002A	Core
R12	0x008480	Core
R13	0x000000	Core
R14	0x000000	Core
R15	0x000000	Core

Figure 02: Program 2 output

Program Flowchart:

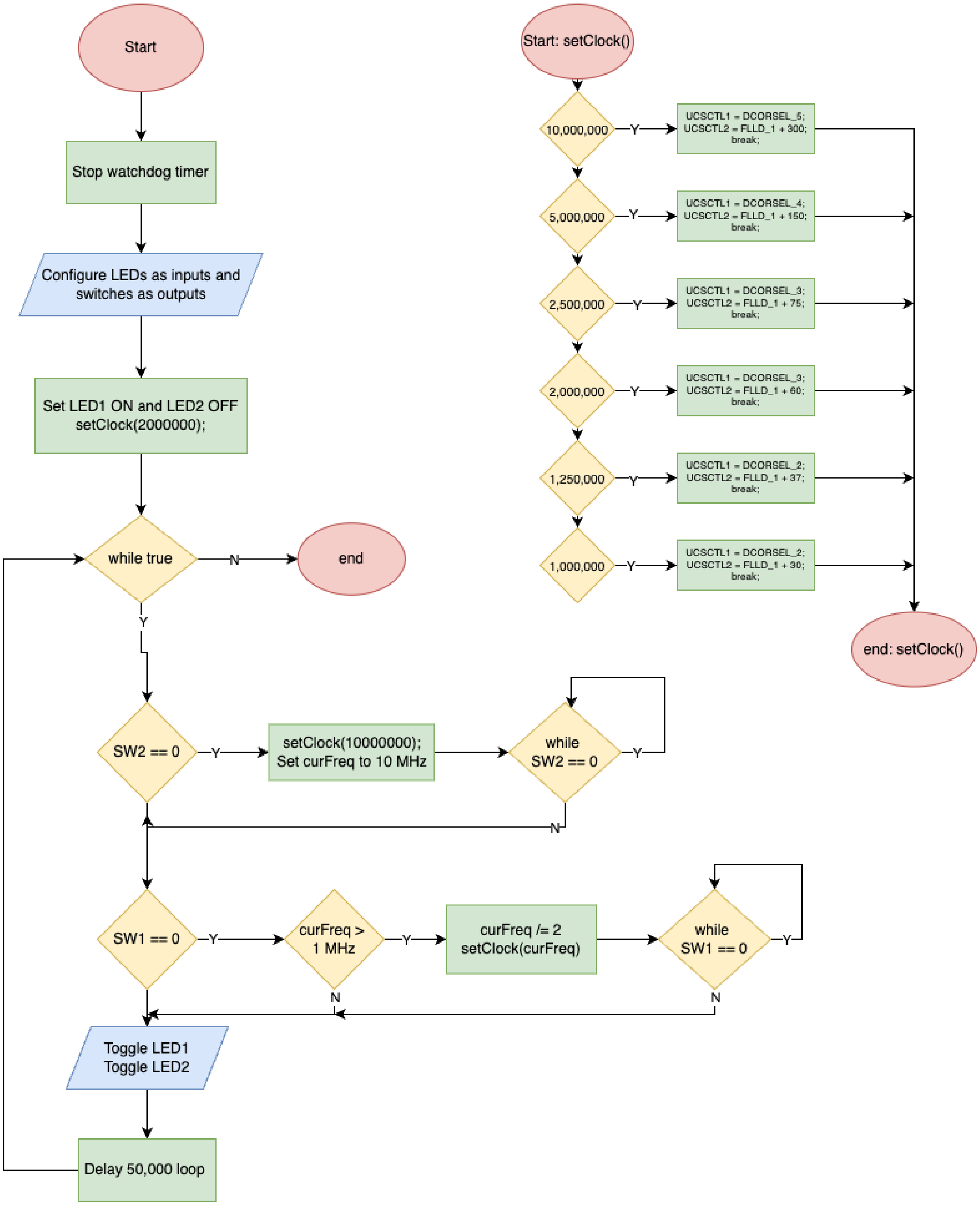


Figure 03: Program 2 Flowchart

a. **Calculate LEDs blinking rate for each clock frequency and show your work.**

Each clock frequency setting in setClock() corresponds to specific values for UCSCTL1 (DCO range) and UCSCTL2 (DCO multiplier settings).

Formula: fDCOCLK = fREFCLK × ((N+1) / D), where

- fREFCLK = 32.768 kHz (default reference clock)
- N = DCO multiplier
- D = DCO divider

- 1 MHz:
    - UCSCTL1 = DCORSEL_2
    - 32768 × (30+1)/1 ≈ 1 MHz
    - UCSCTL2 = FLLD_1 + 30
- 1.25 MHz
    - UCSCTL1 = DCORSEL_2
    - 32768 × (37+1)/1 ≈ 1.25 MHz
    - UCSCTL2 = FLLD_1 + 37
- 2 MHz
    - UCSCTL1 = DCORSEL_3
    - 32768 × (60+1)/1 ≈ 2 MHz
    - UCSCTL2 = FLLD_1 + 60
- 2.5 MHz
    - UCSCTL1 = DCORSEL_3
    - 32768 × (75+1)/1 ≈ 2.5 MHz
    - UCSCTL2 = FLLD_1 + 75
- 5 MHz
    - UCSCTL1 = DCORSEL_4
    - 32768 × (150+1)/1 ≈ 5 MHz
    - UCSCTL2 = FLLD_1 + 150
- 10 MHz
    - UCSCTL1 = DCORSEL_5
    - 32768 × (300+1)/1 ≈ 10 MHz
    - UCSCTL2 = FLLD_1 + 300

Appendix

Table 01: Program 1 source code

```
;-----------------------------------------------------------------------------
;       File:           Lab6_P1.asm
;
;       Description:    An interrupt service routine is used to interface the
switches.
;                                       Initially, both LED1 and LED2 start in
the OFF state. When SW1 is
;                                       first pressed, LED2 turns on and
changes state with each subsequent
;                                       press. When SW2 is pressed, LED1
blinks 5 times at 5 Hz, then changes
;                                       the state of LED2.
;
;   Author:                     Anshika Sinha
;   Date:                       February 18, 2025
;-----------------------------------------------------------------------------

;-----------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-----------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.

            .def    SW_ISR
            .def        delay
;-----------------------------------------------------------------------------
            .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:
                        bis.b   #001h, &P1DIR                  ; Set P1.0 to
output direction
                        bis.b   #080h, &P4DIR                  ; Set P4.7 to
output direction
                        bic.b   #001h, &P1OUT                  ; Set P1OUT to
0x0000_0001 (off)
                        bic.b   #080h, &P4OUT                  ; Set P4OUT to
0x1000_0000 (off)
```

```
                       bic.b   #002h, &P2DIR                  ; Set P2.1 as
input for SW1
                       bis.b   #002h, &P2REN                  ; Enable
pull-up resistor at P2.1
                       bis.b   #002h, &P2OUT                  ; Required for
I/O setup

                       bic.b   #002h, &P1DIR                  ; Set P1.1 as
input for SW2
                       bis.b   #002h, &P1REN                  ; Enable
pull-up resistor at P1.1
                       bis.b   #002h, &P1OUT                  ; Required for
I/O setup

                       bis.w   #GIE, SR                               ;
Enable global interrupts

                       bis.b   #002h, &P1IE                   ; Enable Port1
interrupt
                       bis.b   #002h, &P1IES                  ; Set interrupt
call from hi to low
                       bic.b   #002h, &P1IFG                  ; Clear
interrupt flag

                       bis.b   #002h, &P2IE                   ; Enable Port2
interrupt
                       bis.b   #002h, &P2IES                  ; Set interrupt
call from hi to low
                       bic.b   #002h, &P2IFG                  ; Clear
interrupt flag

InfLoop:
                       nop
                       jmp            $
               ; Loop until interrupt

;----------------------------------------------------------------------------
; P1_0 (SW2) and P4_7 (SW1) interrupt service routine (ISR)
;----------------------------------------------------------------------------
SW_ISR:
                       bic.b   #002, &P2IFG                   ; Clear
Interrupt flag
            bit.b   #002h, &P2IN           ; Check SW2
            jz         CheckSW1                               ; If not, check
SW1
            bic.b      #002h, &P1IFG                  ; Clear interrupt flag

CheckSW2:   bit.b   #02h, &P1IN             ; Check if SW2 is pressed
                                            ; (0000_0010 on P1IN)
            jnz     lExit                   ; If not zero, SW is not pressed
                                            ; loop and check again
Debounce2:   mov.w   #2000, R15              ; Set to (2000 * 10 cc )
SW2D20ms:    dec.w   R15                     ; Decrement R15
            nop
            nop
            nop
            nop
            nop
            nop
            nop
            jnz     SW2D20ms                 ; Delay over?
            bit.b   #00000010b,&P1IN        ; Verify SW2 is still pressed
            jnz     lExit                   ; If not, wait for SW2 press
LED1on:      mov.w     #9, R4                                 ; Blink 5 times
blinkLoop:
                       xor.b   #0x01,&P1OUT           ; Toggle LED1

                       call    #delay
                       dec.w   R4
                       jnz             blinkLoop
                       xor.b   #0x80, &P4OUT                  ; Toggle LED2

SW2wait:    bit.b   #002h,&P1IN             ; Test SW2
            jz      SW2wait                 ; Wait until SW2 is released
            bic.b   #001,&P1OUT             ; Turn off LED1
            jmp        lExit

CheckSW1:       bit.b   #002h, &P2IN                   ; Check if SW1 is
pressed
                       jnz             lExit


Debounce1:  mov.w   #2000, R15              ; Set to (2000 * 10 cc )
SW1D20ms:   dec.w   R15                     ; Decrement R15
            nop
            nop
            nop
            nop
            nop
            nop
            nop
            jnz     SW1D20ms                 ; Delay over?
            bit.b   #002h,&P2IN                ; Verify SW2 is still pressed
            jnz     lExit                   ; If not, wait for SW2 press

LED2on:     xor.b      #0x80, &P4OUT                   ; Toggle LED2
SW1wait:    bit.b   #002h,&P1IN             ; Test SW1
            jz      SW1wait                 ; Wait until SW1 is released
            jmp        lExit

lExit:          reti                                                  ;
return from interrupt

;------------------------------------------------------------------------------
--
; Delay subroutine
;------------------------------------------------------------------------------
--
delay:          mov            #50000, R5
                       jmp     delayLoop
delayLoop:
                       dec     R5
                       jnz             delayLoop
                       ret
```

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

;-------------------------------------------------------------------------------
--
; Interrupt Vectors
;-------------------------------------------------------------------------------
--
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
            .sect   ".int47"                ;PORT1_VECTOR
            .short     SW_ISR
            .sect      ".int42"              ;PORT2_VECTOR
            .short  SW_ISR
            .end
```

Table 02: Program 2 source code

```
/***************************************************************************
 *   File:        Lab6_P2.c
 *   Description: Initially, LED1 is on and LED2 is off. The clock frequency
is set to 2 MHz.
 *                LED1 and LED2 blink using a 50,000 interaction loop delay.
Every time SW2 is pressed,
 *                the clock frequency is set to 10 MHz. Every time SW1 is
pressed, the clock
 *                frequency is halved, and it cannot go under 1 MHz.
 *   Input:       Press SW1 or SW2
 *   Output:      LED1 and LED2 blink at various different frequencies
 *   Board:       MSP430F5529 Experimenter Board
 *   Author:      Anshika Sinha
 *   Date:        February 18, 2025
 ***************************************************************************/

#include <msp430.h>

// Interface inputs and outputs
#define SW1 P2IN&BIT1                           // Switch 1 at P2.1
#define SW2 P1IN&BIT1                           // Switch 2 at P1.1
#define LED1 0x01                               // Mask for BIT0 = 0000_0001b
#define LED2 0x80                               // Mask for BIT7 = 1000_0000b

void setClock(unsigned long freq);
unsigned long curFreq = 2000000;            // Set inital frequency to 2 MHz

int main(void)
{
        WDTCTL = WDTPW | WDTHOLD;                      // Stop watchdog timer
        // Configure LEDs as outputs
```

```
    P1DIR |= LED1;                           // Set P1.0 to output direction
    P4DIR |= LED2;                           // Set P4.7 to output direction

    // Configure switches as inputs
    P2DIR &= ~BIT1;                             // Set P2.1 as input for SW1
input
    P2REN |= BIT1;                              // Enable pull-up register at
P2.1
    P2OUT |= BIT1;

    P1DIR &= ~BIT1;                             // Set P1.1 as input for SW2
input
    P1REN |= BIT1;                              // Enable pull-up register at
P1.1
    P1OUT |= BIT1;

    P1OUT |= LED1;                             // LED1 is ON
    P4OUT &= ~LED2;                              // LED2 is OFF

    setClock(2000000);                          // Set clock frequency to 2 Hz

    while(1) {
        if ((SW2) == 0)
        {
            setClock(10000000);             // Set clock frequency to 10 MHz
            curFreq = 10000000;             // Update current frequency
            while ((SW2) == 0);             // Wait for release
        }
        if ((SW1) == 0)                         // If SW1 is pressed
        {
            if (curFreq > 1000000) {
                curFreq /= 2;
                setClock(curFreq);
            }
            while ((SW1) == 0);             // Wait for release
        }

        // Toggle LEDs
        P1OUT ^= LED1;
        P4OUT ^= LED2;

        unsigned int i = 0;
        for (i = 50000; i>0; i--);           // 50,000 delay loop
    }
}

void setClock(unsigned long freq) {
    switch (freq) {
            case 10000000:
                UCSCTL1 = DCORSEL_5;
                UCSCTL2 = FLLD_1 + 300;
                break;
            case 5000000:
                UCSCTL1 = DCORSEL_4;
                UCSCTL2 = FLLD_1 + 150;
                break;
            case 2500000:
                UCSCTL1 = DCORSEL_3;
                UCSCTL2 = FLLD_1 + 75;
                break;
            case 2000000:
                UCSCTL1 = DCORSEL_3;
                UCSCTL2 = FLLD_1 + 60;
                break;
            case 1250000:
                UCSCTL1 = DCORSEL_2;
                UCSCTL2 = FLLD_1 + 37;
                break;
            case 1000000:
                UCSCTL1 = DCORSEL_2;
                UCSCTL2 = FLLD_1 + 30;
                break;
    }
}
```

Make sure to submit a single pdf file only