**NAME Mohsin Sajjad  
Section A:**

**Registeration Number: 22pwcse2149**

**Assignment MBSD**

**Question 01:**

Implement a breathing LED effect on the MSP430 using two hardware timers (Timer0 and Timer1). The LED should gradually increase in brightness (fade in), then decrease (fade out) repeatedly. Use Timer0 to alternate between LED ON and OFF states using variable durations, and Timer1 to gradually update the on/off durations to simulate a smooth breathing pattern. Configure the system using ACLK and low-power mode (LPM3). Provide options to apply this effect to one or two LED’s. (CLO3) Use Timer0 (TA0) and Timer1 (TA1) and ACLK (32.768 KHz) for Low Power operation LPM3.

**Answer:**

**CODE:**

**#include <msp430.h>**

**#define LED1 BIT0 // P1.0**

**#define LED2 BIT0 // P4.0 (optional second LED)**

**volatile unsigned int duty = 0; // Current ON-time (0…1000)**

**volatile int direction = 1; // 1 = fading in, –1 = fading out**

**void setupGPIO(void) {**

**// LEDs as outputs, start OFF**

**P1DIR |= LED1; P1OUT &= ~LED1;**

**P4DIR |= LED2; P4OUT &= ~LED2;**

**}**

**void setupTimer0(void) {**

**TA0CCR0 = duty; // initial ON-time**

**TA0CCTL0 = CCIE; // enable CCR0 interrupt**

**TA0CTL = TASSEL\_1 + MC\_1 // ACLK, up-mode**

**+ TACLR;**

**}**

**void setupTimer1(void) {**

**TA1CCR0 = 512; // ~15 ms ticks (32 768 Hz / 512)**

**TA1CCTL0 = CCIE; // enable CCR0 interrupt**

**TA1CTL = TASSEL\_1 + MC\_1 // ACLK, up-mode**

**+ TACLR;**

**}**

**int main(void) {**

**WDTCTL = WDTPW | WDTHOLD; // stop watchdog**

**PM5CTL0 &= ~LOCKLPM5; // activate GPIO**

**setupGPIO();**

**setupTimer0();**

**setupTimer1();**

**\_\_bis\_SR\_register(LPM3\_bits + GIE);**

**// CPU sleeps in LPM3, wakes only for TA0/TA1 ISRs**

**while (1);**

**}**

**// TA0 ISR: toggle LED with current duty cycle**

**#pragma vector = TIMER0\_A0\_VECTOR**

**\_\_interrupt void Timer0\_A0\_ISR(void) {**

**static unsigned char state = 0;**

**if (state == 0) {**

**// turn LEDs ON for “duty” ticks**

**P1OUT |= LED1;**

**P4OUT |= LED2;**

**TA0CCR0 = duty;**

**} else {**

**// turn LEDs OFF for “1000–duty” ticks**

**P1OUT &= ~LED1;**

**P4OUT &= ~LED2;**

**TA0CCR0 = 1000 - duty;**

**}**

**state ^= 1;**

**}**

**// TA1 ISR: step the duty up/down**

**#pragma vector = TIMER1\_A0\_VECTOR**

**\_\_interrupt void Timer1\_A0\_ISR(void) {**

**duty += direction;**

**if (duty >= 1000) {**

**duty = 1000;**

**direction = -1;**

**} else if (duty == 0) {**

**direction = 1;**

**}**

**}**

**Question 01:**

Write a code that turns on blinking when push button is pressed and released, turns off when push button is pressed and released again. (CLO 3)

**Answer:**

**Code:**

#include <msp430.h>

#define LED BIT0 // P1.0

#define BUTTON BIT2 // P1.2

volatile unsigned char blinkOn = 0;

void setupGPIO(void) {

// LED output, off

P1DIR |= LED;

P1OUT &= ~LED;

// Button input with pull-up

P1DIR &= ~BUTTON;

P1REN |= BUTTON;

P1OUT |= BUTTON;

// Button interrupt on high-to-low (press)

P1IE |= BUTTON;

P1IES |= BUTTON;

P1IFG &= ~BUTTON;

}

void setupTimer0(void) {

TA0CCR0 = 16384; // ~0.5 s (ACLK/2)

TA0CCTL0 = CCIE; // enable CCR0 interrupt

TA0CTL = TASSEL\_1 + MC\_1 // ACLK, up-mode

+ TACLR;

}

int main(void) {

WDTCTL = WDTPW | WDTHOLD; // stop watchdog

PM5CTL0 &= ~LOCKLPM5; // enable GPIO

setupGPIO();

setupTimer0();

\_\_enable\_interrupt();

while (1) {

\_\_bis\_SR\_register(LPM3\_bits + GIE);

// Sleep until button ISR wakes CPU

if (!blinkOn) {

// Ensure LED is off when blinking disabled

P1OUT &= ~LED;

}

}

}

// TA0 ISR: toggle LED only if blinkOn == 1

#pragma vector = TIMER0\_A0\_VECTOR

\_\_interrupt void Timer0\_A0\_ISR(void) {

if (blinkOn) {

P1OUT ^= LED;

}

}

// Port1 ISR: toggle blinking state on each press

#pragma vector = PORT1\_VECTOR

\_\_interrupt void Port\_1(void) {

if (P1IFG & BUTTON) {

blinkOn ^= 1;

P1IFG &= ~BUTTON; // clear flag

\_\_bic\_SR\_register\_on\_exit(LPM3\_bits);

}

}