Lab 8

LOW POWER MODE

Spring 2025

Submitted by: Mohsin Sajjad

Registration No: 22pwsce2149

Class Section: A

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Mohsin Sajjad

Student Signature: ______________

Submitted to:

Engr. Faheem Jan

Month Day, Year (04 05, 2025)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

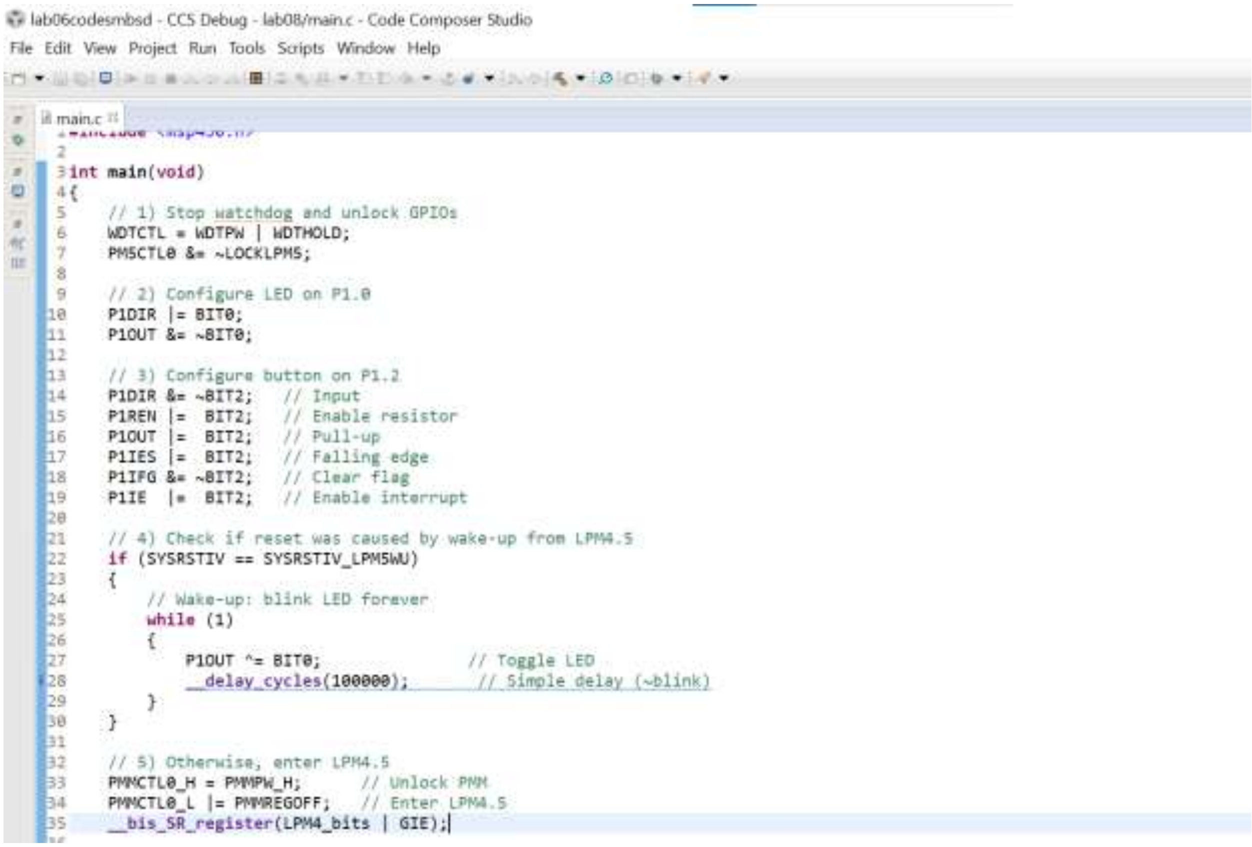
LOW POWER MODE

TASKS:

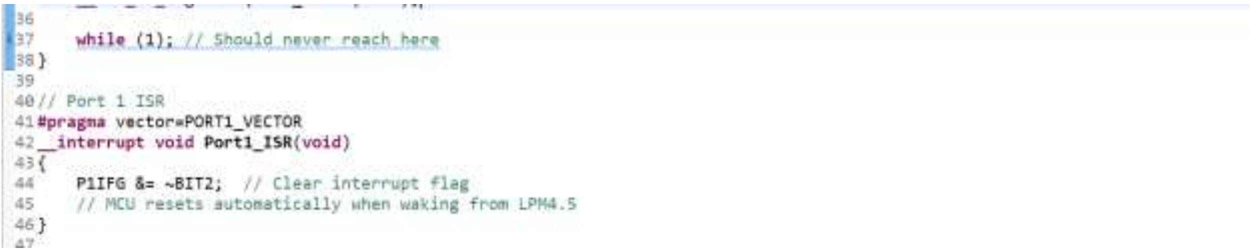
Task 01:

Configure Watchdog Timer (WDT) for 1-second interval. Enter LPM3.

CODE:

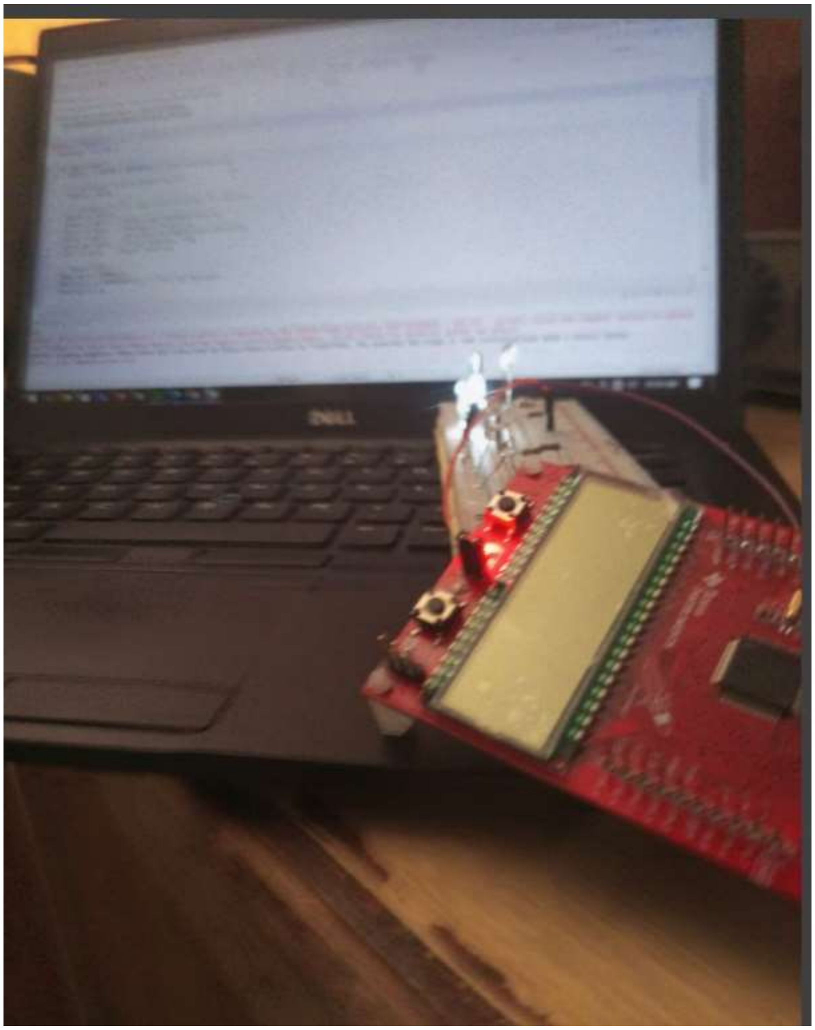

lab06codesmbsd - CCS Debug - lab08/main.c - Code Composer Studio
File Edit View Project Run Tools Scripts Window Help
main.c

```c
#include <msp430.h>

int main(void)
{
    // 1) Stop watchdog and unlock GPIOs
    WDTCTL = WDTPW | WDTHOLD;
    PM5CTL0 &= ~LOCKLPM5;

    // 2) Configure LED on P1.0
    P1DIR |= BIT0;
    P1OUT &= ~BIT0;

    // 3) Configure button on P1.2
    P1DIR &= ~BIT2;   // Input
    P1REN |=  BIT2;   // Enable resistor
    P1OUT |=  BIT2;   // Pull-up
    P1IES |=  BIT2;   // Falling edge
    P1IFG &= ~BIT2;   // Clear flag
    P1IE  |=  BIT2;   // Enable interrupt

    // 4) Check if reset was caused by wake-up from LPM4.5
    if (SYSRSTIV == SYSRSTIV_LPM5WU)
    {
        // Wake-up: blink LED forever
        while (1)
        {
            P1OUT ^= BIT0;              // Toggle LED
            __delay_cycles(100000);      // Simple delay (~blink)
        }
    }

    // 5) Otherwise, enter LPM4.5
    PMMCTL0_H = PMMPW_H;       // Unlock PMM
    PMMCTL0_L |= PMMREGOFF;   // Enter LPM4.5
    __bis_SR_register(LPM4_bits | GIE);
```



```c

    while (1); // Should never reach here
}

// Port 1 ISR
#pragma vector=PORT1_VECTOR
__interrupt void Port1_ISR(void)
{
    P1IFG &= ~BIT2;  // Clear interrupt flag
    // MCU resets automatically when waking from LPM4.5
}
```

OUTPUT:



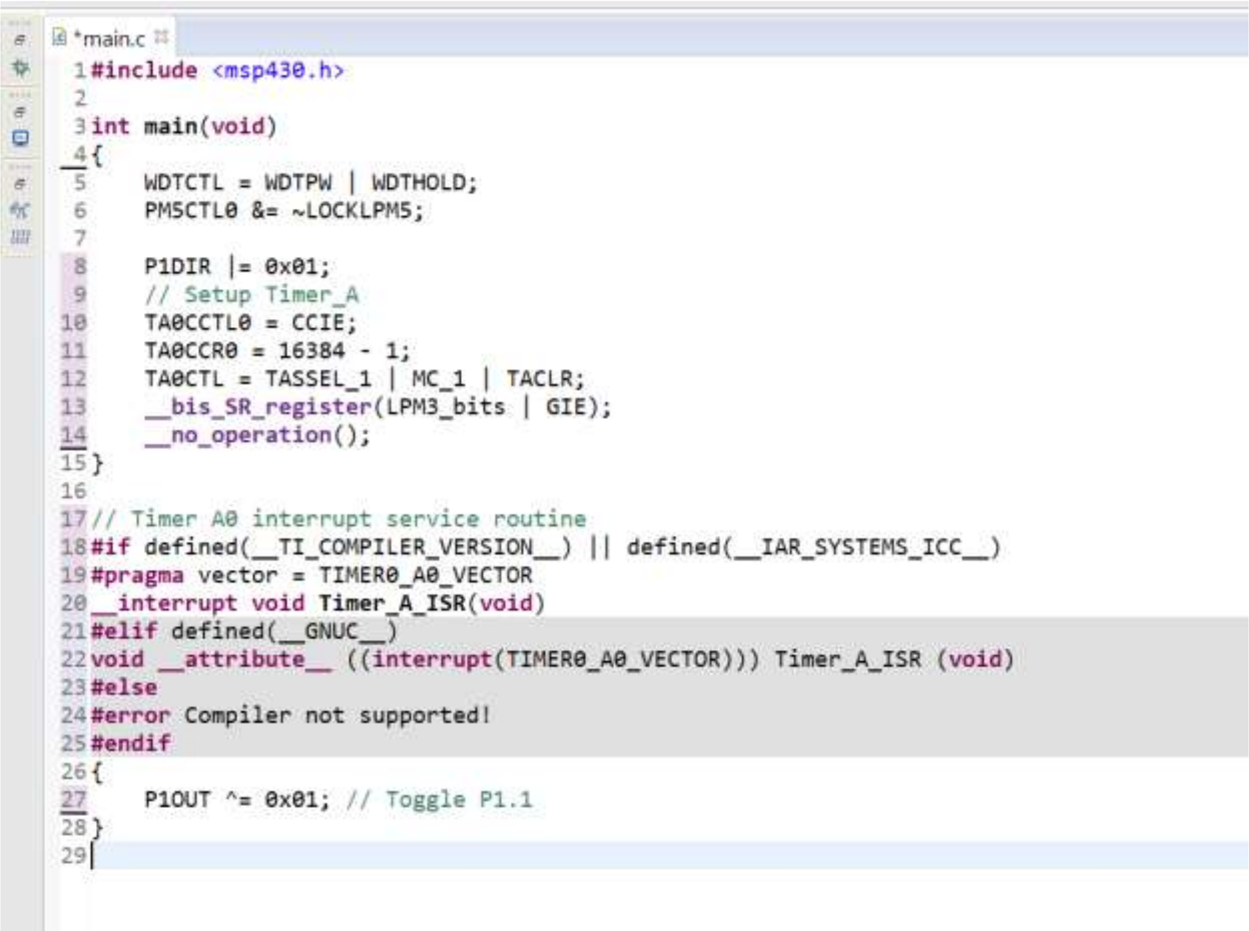
conclusion:

In this programe the MSP430 microcontroller turns off everything to save power (LPM4.5) and waits for a button press. The button is connected to pin P1.2 and the LED to P1.0. When the button is pressed, the microcontroller wakes up and starts blinking the LED. If the button is not pressed, it stays in low-power mode. This shows how to use low power and wake-up features in a simple way.

Task 2:

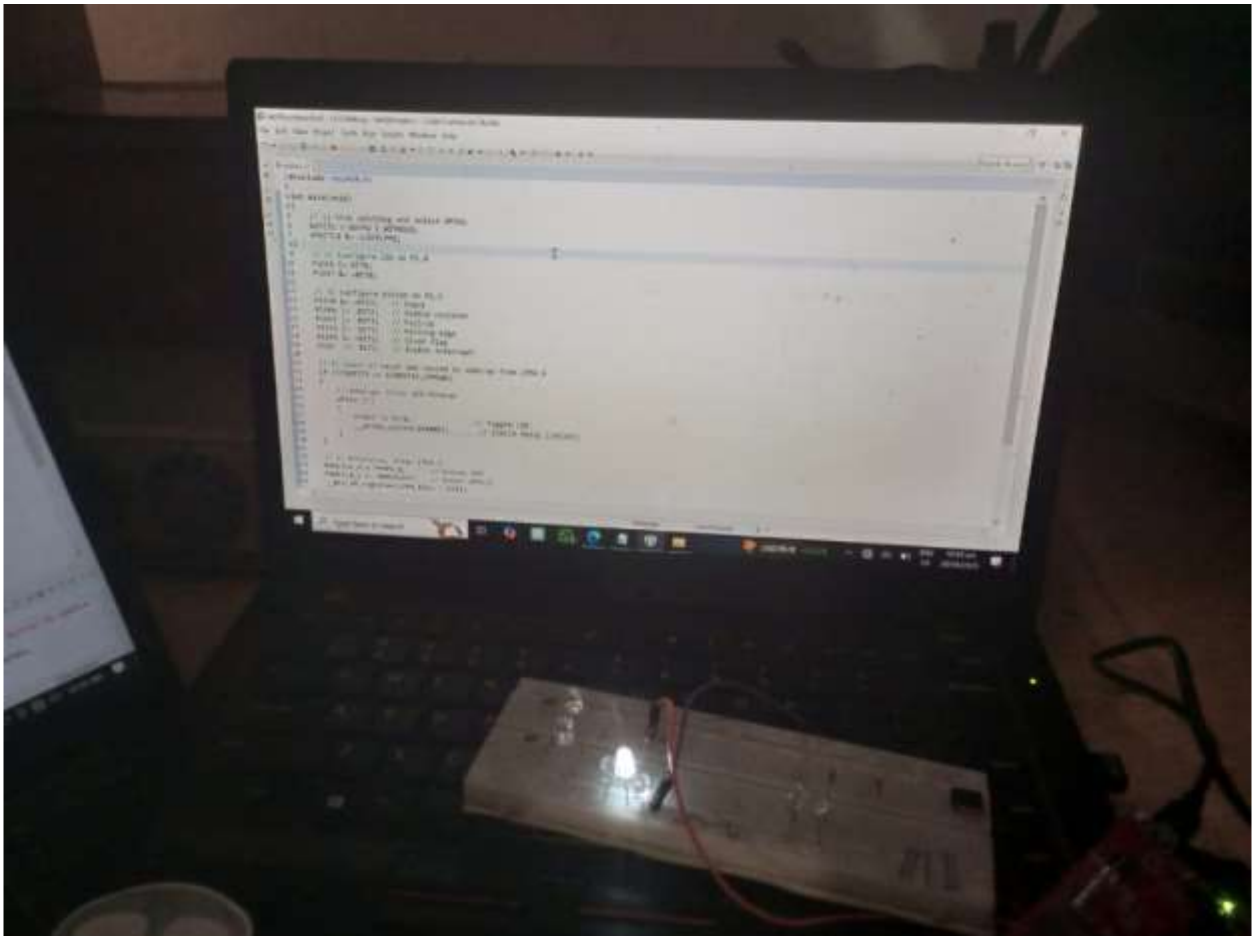
Use Timer_A(instead of WDT) to generate an interrupt every 500 ms. Enter LPM3. Toggle P1.1 LED every 500 ms when Timer_Ainterrupt fires.

CODE:



```c
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;
    PM5CTL0 &= ~LOCKLPM5;

    P1DIR |= 0x01;
    // Setup Timer_A
    TA0CCTL0 = CCIE;
    TA0CCR0 = 16384 - 1;
    TA0CTL = TASSEL_1 | MC_1 | TACLR;
    __bis_SR_register(LPM3_bits | GIE);
    __no_operation();
}

// Timer A0 interrupt service routine
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector = TIMER0_A0_VECTOR
__interrupt void Timer_A_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(TIMER0_A0_VECTOR))) Timer_A_ISR (void)
#else
#error Compiler not supported!
#endif
{
    P1OUT ^= 0x01; // Toggle P1.1
}
```

OUTPUT:



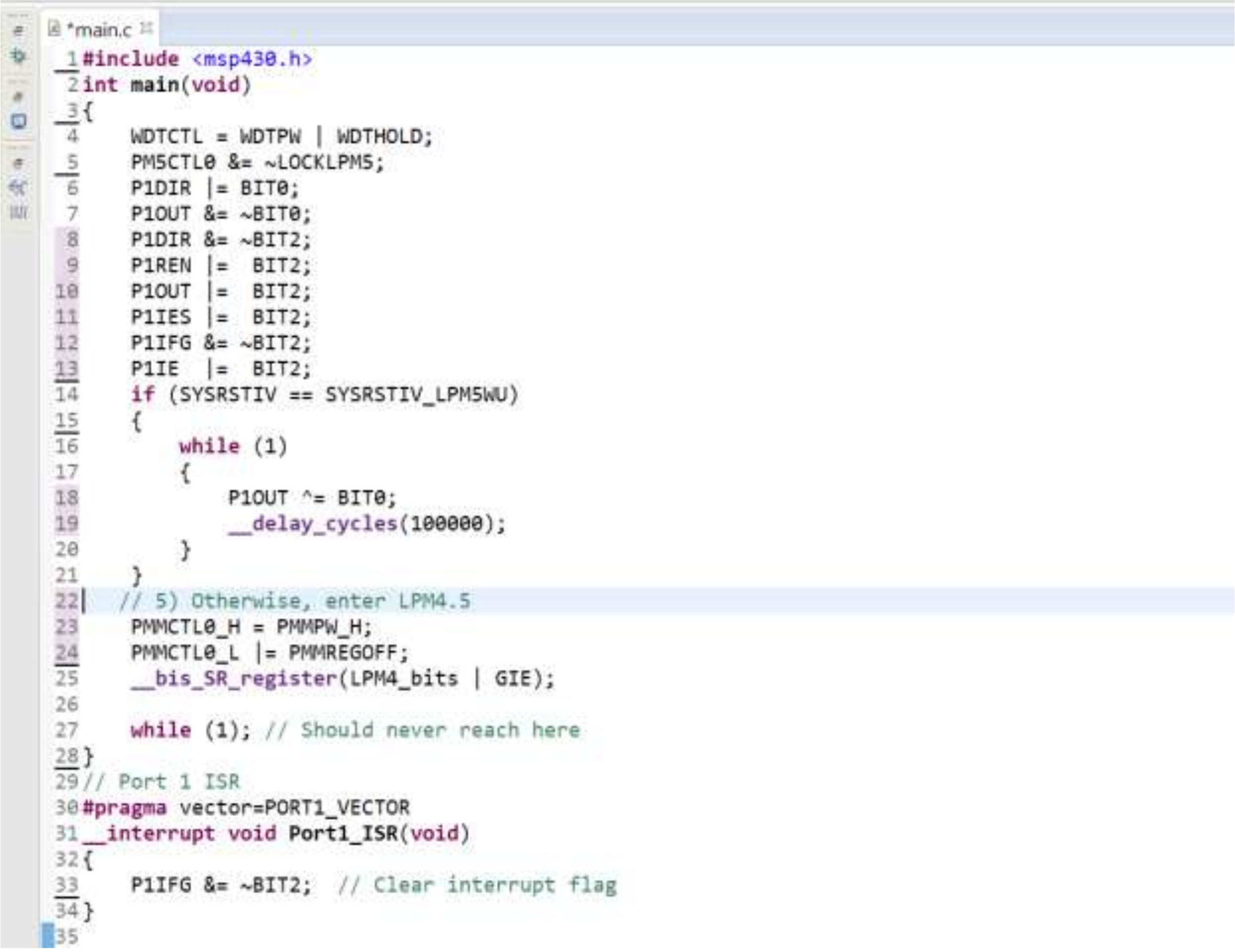
CONCLUSION:

This MSP430 program uses Timer_A to blink an LED connected to P1.0. The timer is set to generate an interrupt every 0.5 seconds (using ACLK). When the timer interrupt occurs, it toggles the LED. The microcontroller stays in Low Power Mode 3 (LPM3) to save energy and wakes up only for the interrupt. This shows how to use timers and low power together for energy-efficient LED blinking.

TASK 03:

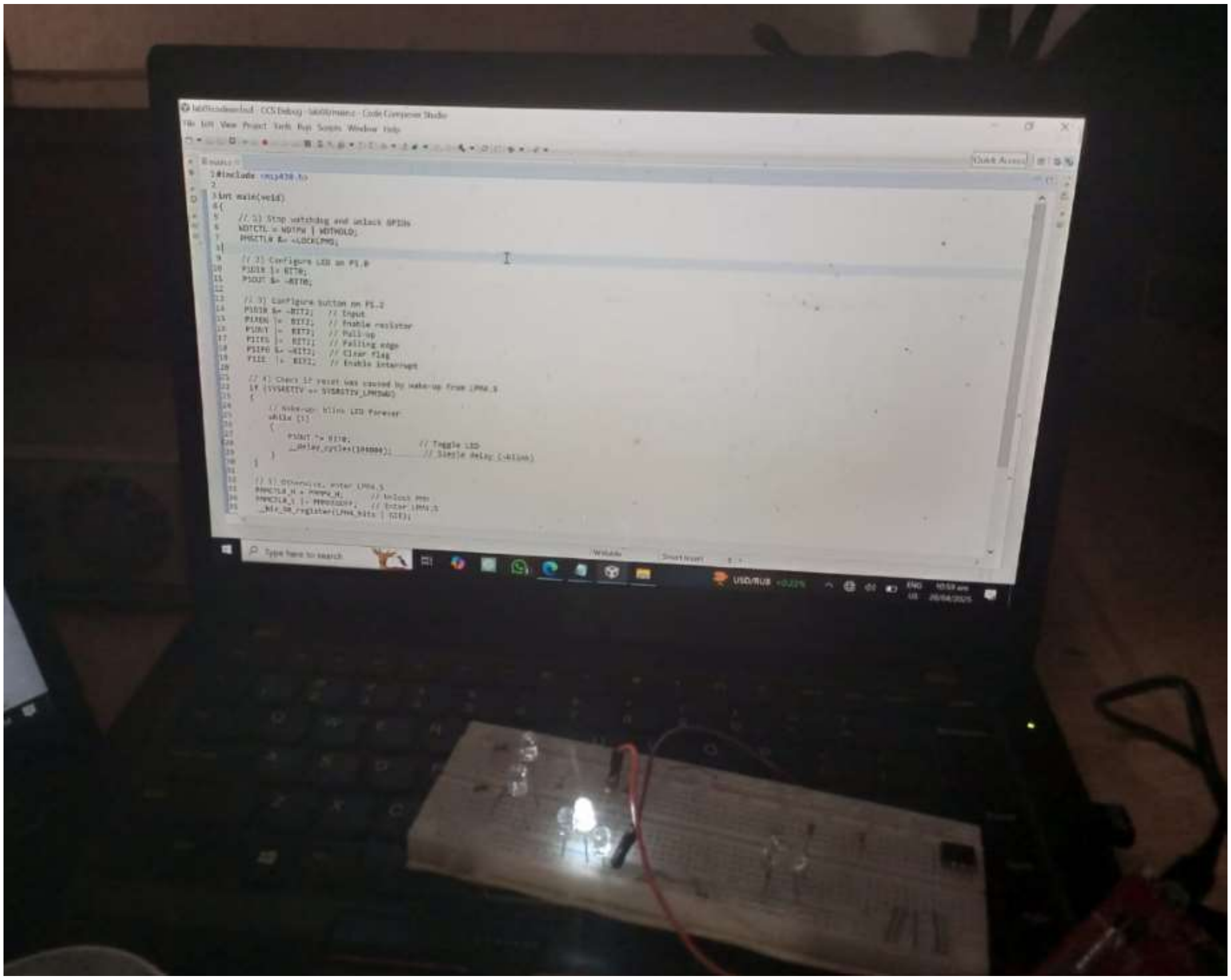
Configure a push-button on P1.2 as wake-up pin. •Enter LPM4.5. •When the button is pressed, MCU resets and blinks P1.0 continuously. (You must configure the wake-up interrupt before entering LPM4.5.)

CODE:



```c
#include <msp430.h>
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;
    PM5CTL0 &= ~LOCKLPM5;
    P1DIR |= BIT0;
    P1OUT &= ~BIT0;
    P1DIR &= ~BIT2;
    P1REN |=  BIT2;
    P1OUT |=  BIT2;
    P1IES |=  BIT2;
    P1IFG &= ~BIT2;
    P1IE  |=  BIT2;
    if (SYSRSTIV == SYSRSTIV_LPM5WU)
    {
        while (1)
        {
            P1OUT ^= BIT0;
            __delay_cycles(100000);
        }
    }
   // 5) Otherwise, enter LPM4.5
    PMMCTL0_H = PMMPW_H;
    PMMCTL0_L |= PMMREGOFF;
    __bis_SR_register(LPM4_bits | GIE);

    while (1); // Should never reach here
}
// Port 1 ISR
#pragma vector=PORT1_VECTOR
__interrupt void Port1_ISR(void)
{
    P1IFG &= ~BIT2;  // Clear interrupt flag
}
```

Output:

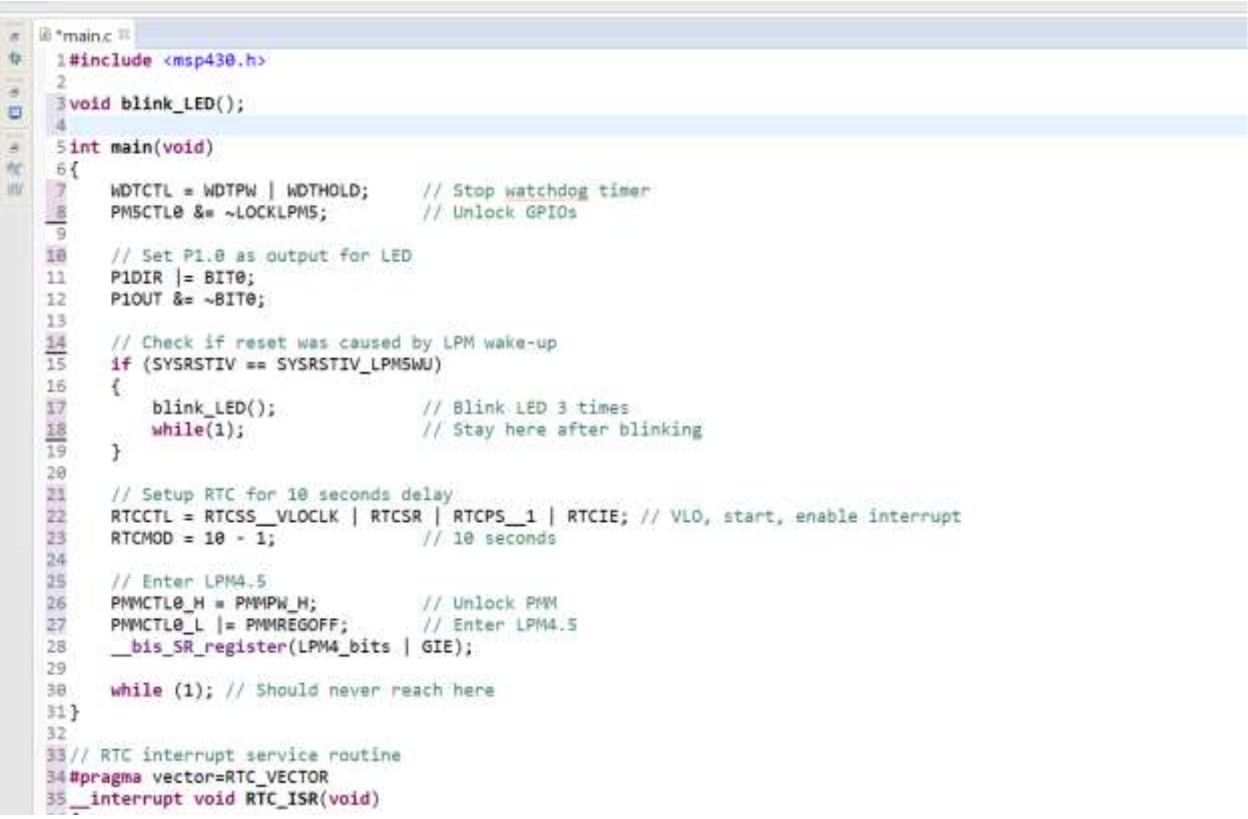


```
#include <msp430.h>

int main(void)
{
    // 1) Stop watchdog and unlock GPIOs
    WDTCTL = WDTPW | WDTHOLD;
    PM5CTL0 &= ~LOCKLPM5;

    // 2) Configure LED on P1.0
    P1DIR |= BIT0;
    P1OUT &= ~BIT0;

    // 3) Configure button on P1.2
    P1DIR &= ~BIT2;   // Input
    P1REN |=  BIT2;   // Enable resistor
    P1OUT |=  BIT2;   // Pull-up
    P1IES |=  BIT2;   // Falling edge
    P1IFG &= ~BIT2;   // Clear flag
    P1IE  |=  BIT2;   // Enable interrupt

    // 4) Check if reset was caused by wake-up from LPM4.5
    if (SYSRSTIV == SYSRSTIV_LPM5WU)
    {
        // Wake-up: blink LED forever
        while (1)
        {
            P1OUT ^= BIT0;                 // Toggle LED
            __delay_cycles(100000);        // Simple delay (~blink)
        }
    }

    // 5) Otherwise, enter LPM4.5
    PMMCTL0_H = PMMPW_H;       // Unlock PMM
    PMMCTL0_L |= PMMREGOFF;    // Enter LPM4.5
    __bis_SR_register(LPM4_bits | GIE);
```

Task 04:

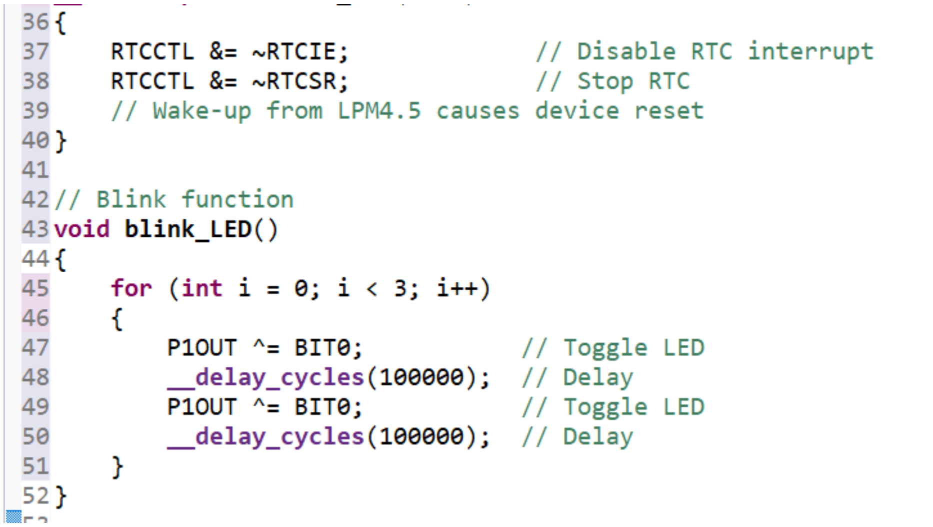
Setup a timer or RTC (Real Time Clock) to wake device after 10 seconds. Enter LPM4.5. When it wakes up, blink P1.0 LED three times.

Code:



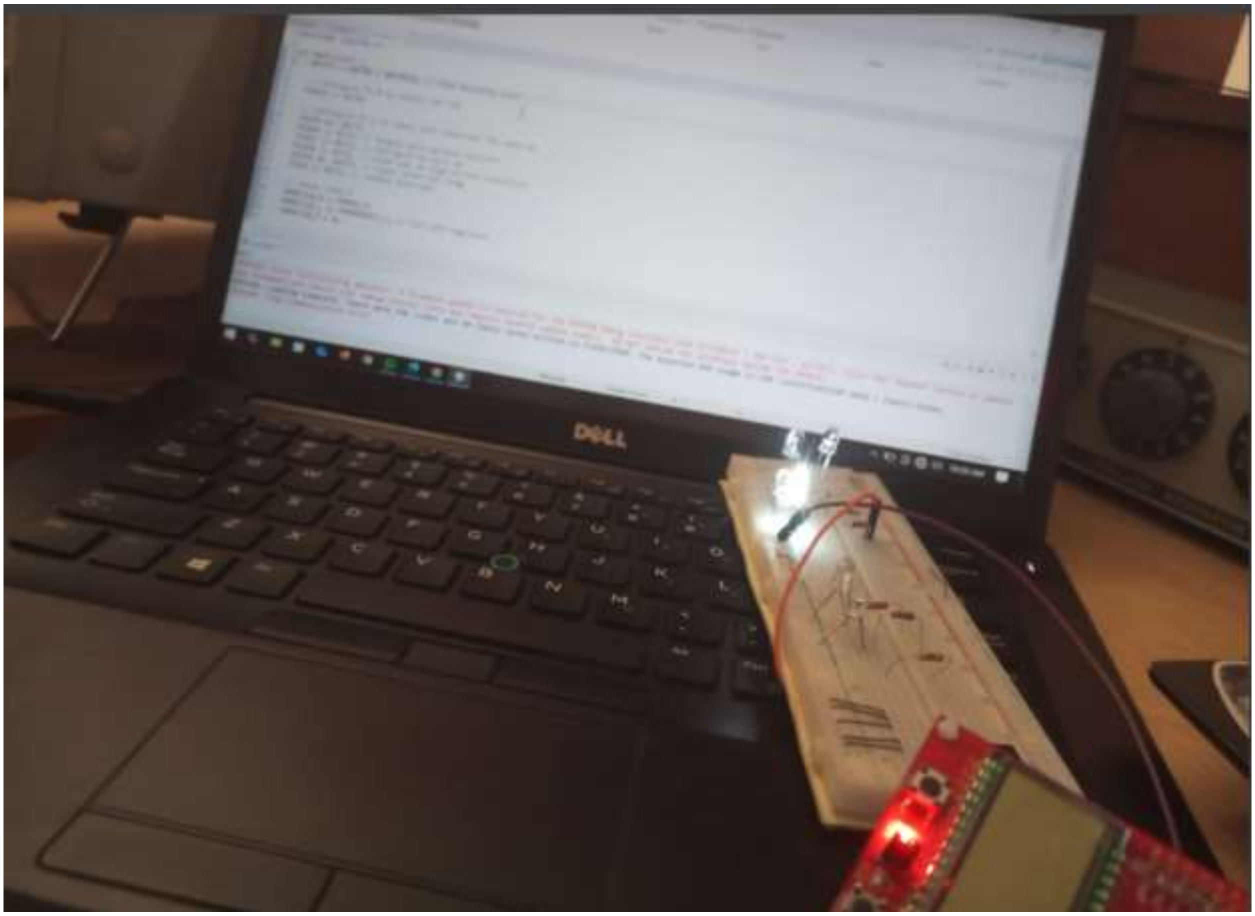
*main.c

```
#include <msp430.h>

void blink_LED();

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;     // Stop watchdog timer
    PM5CTL0 &= ~LOCKLPM5;         // Unlock GPIOs

    // Set P1.0 as output for LED
    P1DIR |= BIT0;
    P1OUT &= ~BIT0;

    // Check if reset was caused by LPM wake-up
    if (SYSRSTIV == SYSRSTIV_LPM5WU)
    {
        blink_LED();              // Blink LED 3 times
        while(1);                 // Stay here after blinking
    }

    // Setup RTC for 10 seconds delay
    RTCCTL = RTCSS__VLOCLK | RTCSR | RTCPS__1 | RTCIE; // VLO, start, enable interrupt
    RTCMOD = 10 - 1;              // 10 seconds

    // Enter LPM4.5
    PMMCTL0_H = PMMPW_H;          // Unlock PMM
    PMMCTL0_L |= PMMREGOFF;       // Enter LPM4.5
    __bis_SR_register(LPM4_bits | GIE);

    while (1); // Should never reach here
}

// RTC interrupt service routine
#pragma vector=RTC_VECTOR
__interrupt void RTC_ISR(void)
```



```
{
    RTCCTL &= ~RTCIE;               // Disable RTC interrupt
    RTCCTL &= ~RTCSR;               // Stop RTC
    // Wake-up from LPM4.5 causes device reset
}

// Blink function
void blink_LED()
{
    for (int i = 0; i < 3; i++)
    {
        P1OUT ^= BIT0;              // Toggle LED
        __delay_cycles(100000);   // Delay
        P1OUT ^= BIT0;              // Toggle LED
        __delay_cycles(100000);   // Delay
    }
}
```

OUTPUT:



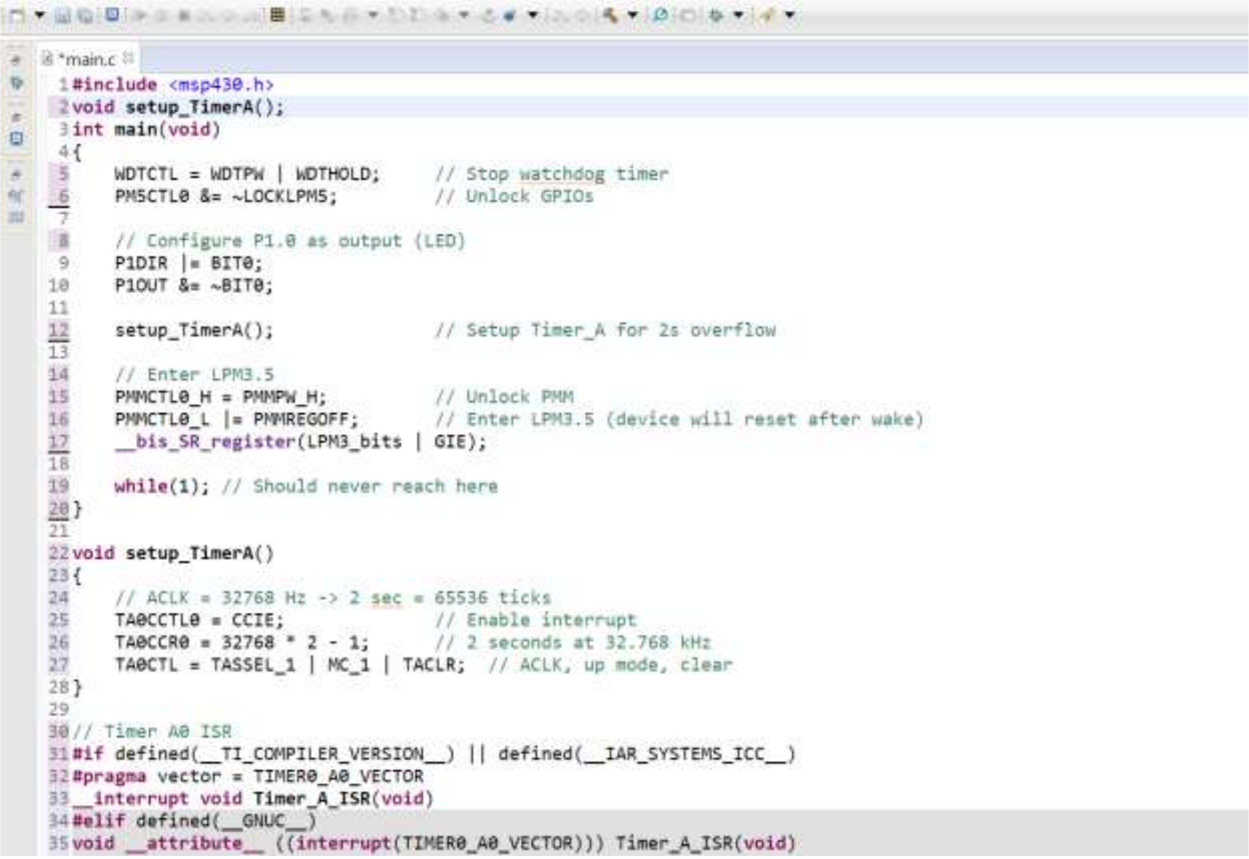
CONCLUSION:

This program uses the RTC to wake up the MSP430 from deep sleep (LPM4.5) after 10 seconds. When it wakes up, it blinks the LED on P1.0 three times. The device then stays in an idle state. This helps save power while performing a task after a timed delay.

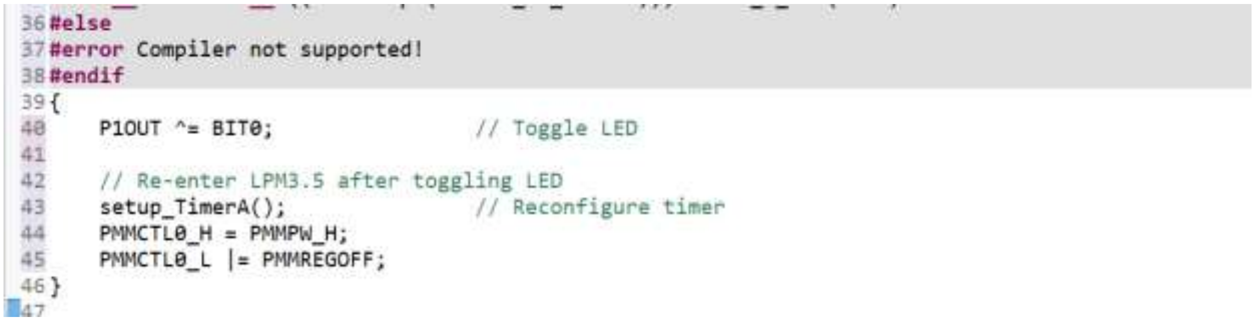
Task 05:

•Configure a Timer_A to overflow every 2 seconds. •Enter LPM3.5. •On wakeup, toggle an LED and go back to sleep automatically.

CODE:

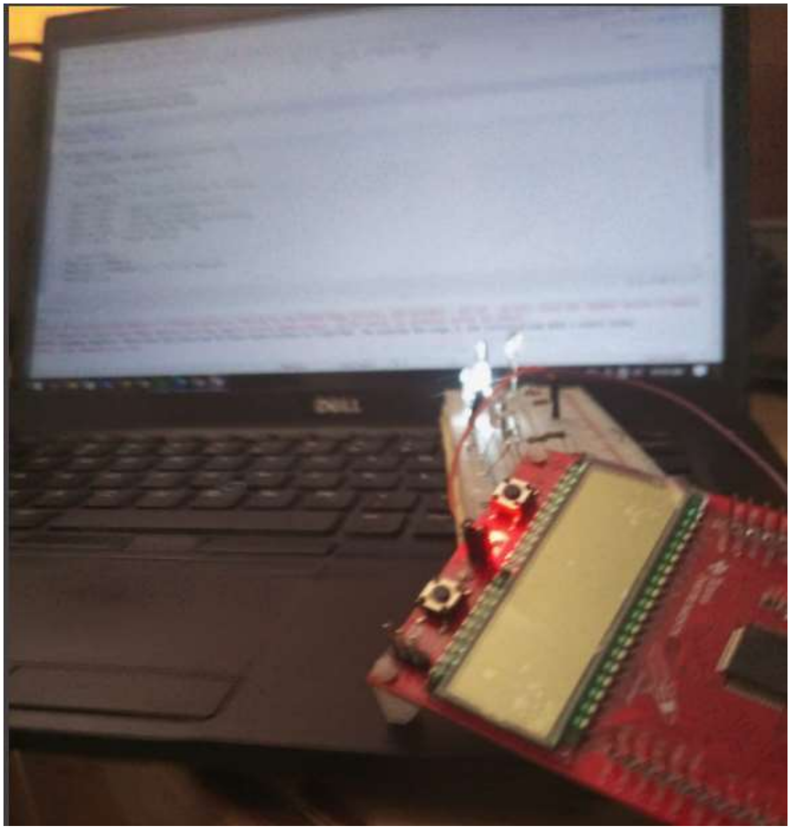


```c
#include <msp430.h>
void setup_TimerA();
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;      // Stop watchdog timer
    PM5CTL0 &= ~LOCKLPM5;          // Unlock GPIOs

    // Configure P1.0 as output (LED)
    P1DIR |= BIT0;
    P1OUT &= ~BIT0;

    setup_TimerA();                // Setup Timer_A for 2s overflow

    // Enter LPM3.5
    PMMCTL0_H = PMMPW_H;           // Unlock PMM
    PMMCTL0_L |= PMMREGOFF;        // Enter LPM3.5 (device will reset after wake)
    __bis_SR_register(LPM3_bits | GIE);

    while(1); // Should never reach here
}

void setup_TimerA()
{
    // ACLK = 32768 Hz -> 2 sec = 65536 ticks
    TA0CCTL0 = CCIE;               // Enable interrupt
    TA0CCR0 = 32768 * 2 - 1;       // 2 seconds at 32.768 kHz
    TA0CTL = TASSEL_1 | MC_1 | TACLR;  // ACLK, up mode, clear
}

// Timer A0 ISR
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector = TIMER0_A0_VECTOR
__interrupt void Timer_A_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(TIMER0_A0_VECTOR))) Timer_A_ISR(void)
```



```c
#else
#error Compiler not supported!
#endif
{
    P1OUT ^= BIT0;                 // Toggle LED

    // Re-enter LPM3.5 after toggling LED
    setup_TimerA();                // Reconfigure timer
    PMMCTL0_H = PMMPW_H;
    PMMCTL0_L |= PMMREGOFF;
}
```

Output:



Conclusion:

This program sets up a timer to overflow every 2 seconds using ACLK. The MSP430 enters low power mode (LPM3.5) to save energy. When the timer overflows, the device wakes up, toggles the LED, and goes back to sleep automatically. This cycle repeats every 2 seconds.