

Lab 7

More on Timers Capture 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.”

A handwritten signature in black ink that reads 'Mohsin Sajjad'.

Student Signature: _____

Submitted to:

Engr. Faheem Jan

Month Day, Year (20 04, 2025)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

More on Timers Capture Mode

TASKS:

Capture event (button press on P1.6) and toggle LED on each capture use rising edge.

CODE:

```
*main.c
1#include <msp430fr4133.h>
2#include <stdint.h>
3uint16_t last_time = 0; // Last captured time
4uint16_t cap_diff, new_time = 0;
5int main(void) {
6    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
7    // Configure LED on P1.0
8    P1DIR |= BIT0; // Set P1.0 as output (LED)
9    P1OUT &= ~BIT0; // Ensure LED starts OFF
10
11    // Configure P1.6 for TA0.2 Capture Mode
12    P1DIR &= ~BIT6; // Set P1.6 as input
13    P1SEL0 |= BIT6; // Select Timer_A capture functionality for P1.6
14
15    // Disable high-impedance mode
16    PMSCTL0 &= ~LOCKLPM5;
17
18    // Configure Timer_A Capture Mode on TA0.2
19    TA0CTL2 = CM_1 | CCIS_0 | SCS | CAP | CCIE; // Capture rising edge, enable interrupt
20    TA0CTL = TASSEL_2 | MC_2 | TACLR; // SMCLK, Continuous mode, Clear timer
21
22    __bis_SR_register(LPM4_bits | GIE); // Enter low-power mode with interrupts
23    return 0;
24} // Timer_A Capture ISR
25#pragma vector = TIMER0_A1_VECTOR
26__interrupt void TIMER0_A1_ISR(void) {
27    if (TA0IV == TA0IV_TACCR2) { // Ensure it's TA0CCR2 interrupt
28        new_time = TA0CCR2;
29        cap_diff = new_time - last_time;
30        last_time = new_time;
31
32        P1OUT ^= BIT0; // Toggle LED on each button press
33
34        __bic_SR_register_on_exit(LPM4_bits); // Exit low-power mode
35    }
}
```

OUTPUT:



conclusion:

This code sets up a timer on the MSP430 to measure time between rising edges (like button presses) on pin P1.6. Each time an edge is detected, it calculates the time difference and toggles the LED on P1.0. It uses capture mode with interrupts and stays in low-power mode until triggered.

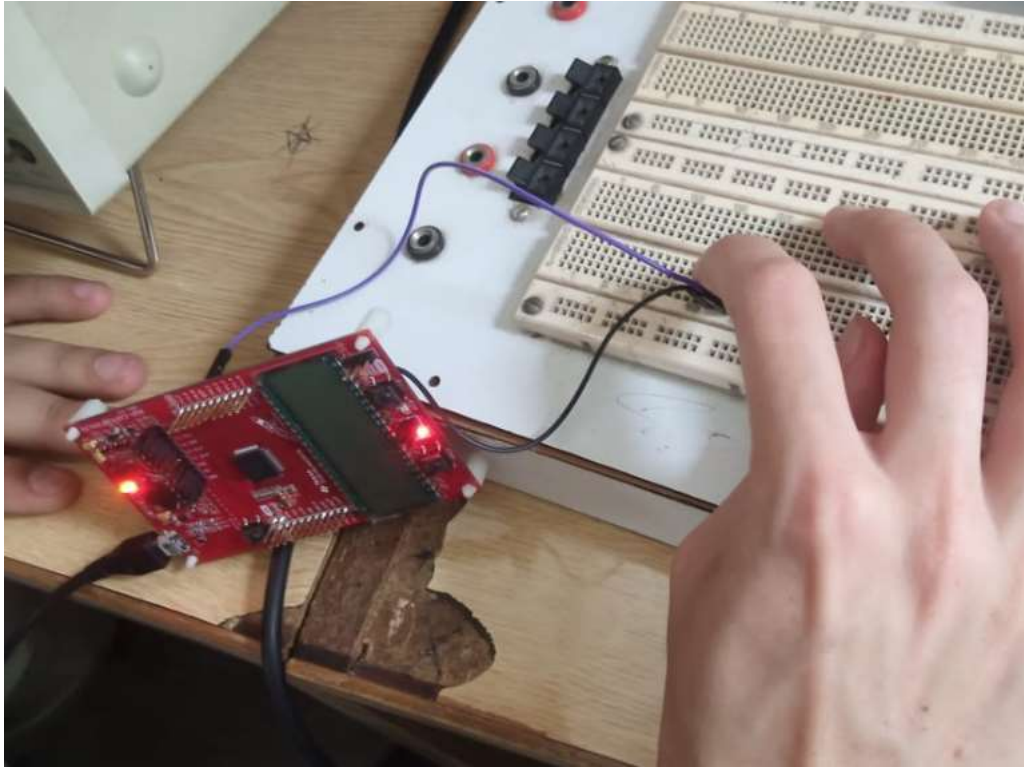
Task 2:

Capture event (button press on P1.6) and toggle LED on each capture use falling edge.

CODE:

```
1 #include <msp430fr4133.h>
2 #include <stdint.h>
3 uint16_t last_time = 0; // Last captured time
4 uint16_t cap_diff, new_time = 0;
5 int main(void) {
6     WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
7
8     // Configure LED on P1.0
9     P1DIR |= BIT0; // Set P1.0 as output (LED)
10    P1OUT &= ~BIT0; // Ensure LED starts OFF
11
12    // Configure P1.6 for TA0.2 Capture Mode
13    P1DIR &= ~BIT6; // Set P1.6 as input
14    P1SEL0 |= BIT6; // Select Timer_A capture functionality for P1.6
15
16    // Disable high-impedance mode
17    PM5CTL0 &= ~LOCKLPM5;
18
19    // Configure Timer_A Capture Mode on TA0.2
20    TA0CTL2 = CM_2 | CCIS_0 | SCS | CAP | CCIE; // Capture falling edge, enable interrupt
21    TA0CTL = TASSEL_2 | MC_2 | TACLK; // SMCLK, Continuous mode, Clear timer
22
23    __bis_SR_register(LPM4_bits | GIE); // Enter low-power mode with interrupts
24    return 0;
25}
26// Timer_A Capture ISR
27#pragma vector = TIMER0_A1_VECTOR
28__interrupt void TIMER0_A1_ISR(void) {
29    if (TA0IV == TA0IV_TACCR2) { // Ensure it's TA0CCR2 interrupt
30        new_time = TA0CCR2;
31        cap_diff = new_time - last_time;
32        last_time = new_time;
33        P1OUT ^= BIT0; // Toggle LED on each button press
34        __bic_SR_register_on_exit(LPM4_bits); // Exit low-power mode
35    }
```

OUTPUT:



CONCLUSION:

This code measures the time between falling edges (like button releases) on pin P1.6 using Timer_A in capture mode. Each time a falling edge is detected, it calculates the time difference and toggles the LED on P1.0. It runs in low-power mode and wakes up only when an interrupt occurs.

TASK 03:

Capture event (button press on P1.6) and toggle LED on each capture both rising and falling edge.

CODE:


```

1 #include <msp430fr4133.h>
2 #include <stdint.h>
3 uint16_t last_time = 0; // Last captured time
4 uint16_t cap_diff, new_time = 0;
5 int main(void) {
6     WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
7
8     // Configure LED on P1.0
9     P1DIR |= BIT0; // Set P1.0 as output (LED)
10    P1OUT &= ~BIT0; // Ensure LED starts OFF
11
12    // Configure P1.6 for TA0.2 Capture Mode
13    P1DIR &= ~BIT6; // Set P1.6 as input
14    P1SEL0 |= BIT6; // Select Timer_A capture functionality for P1.6
15    // Disable high-impedance mode
16    PM5CTL0 &= ~LOCKLPM5;
17    // Configure Timer_A Capture Mode on TA0.2
18    TA0CTL2 = CM_3 | CCIS_0 | SCS | CAP | CCIE; // Capture both edges, enable interrupt
19    TA0CTL = TASSEL_2 | MC_2 | TACLK; // SMCLK, Continuous mode, Clear timer
20
21    __bis_SR_register(LPM4_bits | GIE); // Enter low-power mode with interrupts
22    return 0;
23 } // Timer_A Capture ISR
24 #pragma vector = TIMER0_A1_VECTOR
25 __interrupt void TIMER0_A1_ISR(void) {
26     if (TA0IV == TA0IV_TACCR2) { // Ensure it's TA0CCR2 interrupt
27         new_time = TA0CCR2;
28         cap_diff = new_time - last_time;
29         last_time = new_time;
30
31         P1OUT ^= BIT0; // Toggle LED on each button press
32
33         __bic_SR_register_on_exit(LPM4_bits); // Exit low-power mode
34     }
35 }

```

OUTPUT:



CONCLUSION:

This code uses Timer_A to capture both rising and falling edges on pin P1.6. It measures the time between each edge and toggles the LED on P1.0. The microcontroller stays in low-power mode and wakes up only when an edge is detected.

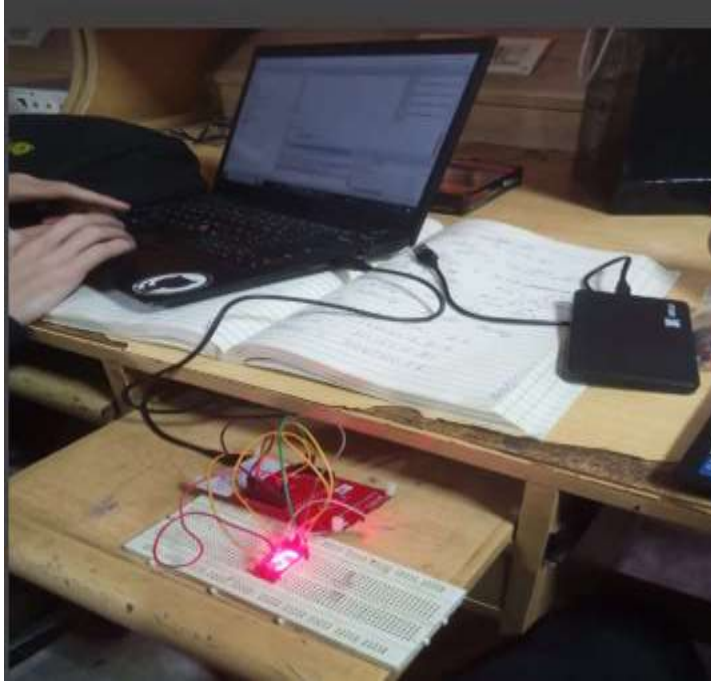
Task 04:

Display the captured event on seven segment display . If the event occur for the first time display 1 for the second time display 2 and capture upto 9 .

CODE:

```
1#include <msp430fr4133.h>
2#include <stdint.h>
3
4uint16_t last_time = 0;
5uint16_t cap_diff, new_time = 0;
6uint8_t event_count = 0;
7
8// Function to display number on 7-segment using P1.0 - P1.6
9void display_number(uint8_t num) {
10    switch(num) {
11        case 0: P1OUT = 0b01111011; break; // display 0
12        case 1: P1OUT = 0b00001010; break; // display 1
13        case 2: P1OUT = 0b10110011; break; // display 2
14        case 3: P1OUT = 0b10011011; break; // display 3
15        case 4: P1OUT = 0b11001010; break; // display 4
16        case 5: P1OUT = 0b11011001; break; // display 5
17        case 6: P1OUT = 0b11111001; break; // display 6
18        case 7: P1OUT = 0b00001011; break; // display 7
19        case 8: P1OUT = 0b11111011; break; // display 8
20        case 9: P1OUT = 0b11011011; break; // display 9
21        default: P1OUT &= 0x80; break; // turn off segments (preserve BIT7 if needed)
22    }
23}
24
25int main(void) {
26    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
27
28    // P1.0 - P1.6 for 7-segment output
29    P1DIR |= 0x7F; // Set P1.0 - P1.6 as output
30    P1OUT &= ~0x7F; // Clear output (all segments OFF)
31
32    // LED on P1.7 (optional for debugging toggle)
33    P1DIR |= BIT7;
34    P1OUT &= ~BIT7;
35
36    // Set P1.6 as input for Timer Capture
37    P2DIR &= ~BIT6;
38    P2SEL0 |= BIT6;
39
40    // Enable GPIOs
41    PM5CTL0 &= ~LOCKLPM5;
42
43    // Timer Capture Configuration
44    TA0CTL2 = CM_3 | CCIS_0 | SCS | CAP | CCIE; // Capture on both edges, enable interrupt
45    TA0CTL = TASSEL_2 | MC_2 | TACLR; // SMCLK, Continuous mode, clear timer
46
47    __bis_SR_register(LPM4_bits | GIE); // Low-power mode with global interrupt enable
48    return 0;
49}
50
51// Timer_A Capture ISR
52#pragma vector = TIMER0_A1_VECTOR
53__interrupt void TIMER0_A1_ISR(void) {
54    if (TA0IV == TA0IV_TACCR2) {
55        new_time = TA0CCR2;
56        cap_diff = new_time - last_time;
57        last_time = new_time;
58
59        if (event_count < 9) {
60            event_count++;
61            display_number(event_count);
62        }
63
64        P1OUT ^= BIT7; // Toggle LED on P1.7 to show event
65
66        __bic_SR_register_on_exit(LPM4_bits); // Wake from low-power mode
67    }
68}
```

Output:



Conclusion:

This code captures events on a rising or falling edge from a timer input (P1.6) and displays the count on a 7-segment display (P1.0 - P1.6). The event count is displayed from 1 to 9, and the LED (P1.7) toggles with each event. The system operates in low-power mode with interrupts enabled, efficiently counting and displaying without delays.