Lab 6
Timers

Engr.Shahzada Fahim Jan

Timers:

- Timers can be used either as
  - ➢Timer to generate a timer delay
  - ➢Event counter to count events happening outside the microcontroller

Mux (Multiplexer) - Clock Source Selector

- The MUX allows selecting between multiple clock sources.
- The selection is controlled by a "Clock Select" input.
- The chosen clock signal is passed to the next stage.

Prescaler - Clock Frequency Divider

- The prescaler divides the input clock frequency by a selectable factor.
- This allows adjusting the clock rate that drives the binary counter.
- The division factor is determined by the "Prescaler Select" input.

Binary Counter - n-bit Counter

- It increments on each clock pulse received from the prescaler.
- It generates an **overflow signal** when it reaches its maximum value.

Comparator - Output Compare Function

- The counter value is continuously compared with a **Compare Register**.
- When the counter matches the compare register, an **Output Compare** signal is generated.
- This is useful for generating periodic events or triggering interrupts.

1. Overflow Output Operation

- The **Timer Clock** provides the clock signal that increments the counter.
- The **Counter** is a 16-bit timer, meaning it can count from 0x0000 to 0xFFFF (65535 in decimal).
- When the counter reaches its maximum value (0xFFFF),
- it **overflows** back to 0x0000 and restarts counting.
- The **Overflow Output** signal generates a pulse when the counter rolls over from 0xFFFF to 0x0000.
- This overflow event can be used for:
    - Periodic interrupts
    - Controlling PWM waveforms

2. Output Compare Operation

- The **Clock (Clk)** signal drives the timer.
- The **Output Compare** signal is generated when the counter value matches a predefined **Compare Register** value.
- example, the compare register is set to **3**, so every third clock cycle, an **Output Compare** event occurs.
- This operation is useful for:
  - Generating precise timing signals
  - Creating square waves or PWM signals

Key Features of Timer_A

1. **16-bit Timer/Counter**
    1. Provides higher precision in timing and counting applications.
2. **3-bit Prescaler**
    1. Allows dividing the clock input by 1, 2, 4, or 8 to adjust timing resolution.
3. **3 Capture/Compare Registers (CCR0, CCR1, CCR2)**
    1. Can store timer values for **event capturing** or **output comparison**.
    2. Used in generating **PWM signals**, event timing, and interrupts.
4. **Four Timer Modes**
    1. **Stop Mode** → Timer is halted.
    2. **Up Mode** → Counts from 0 to a specified value.
    3. **Continuous Mode** → Counts from 0 to 65535 (max value).
    4. **Up/Down Mode** → Counts up, then down to create a symmetric waveform.

Clock Selection (TASSELx)

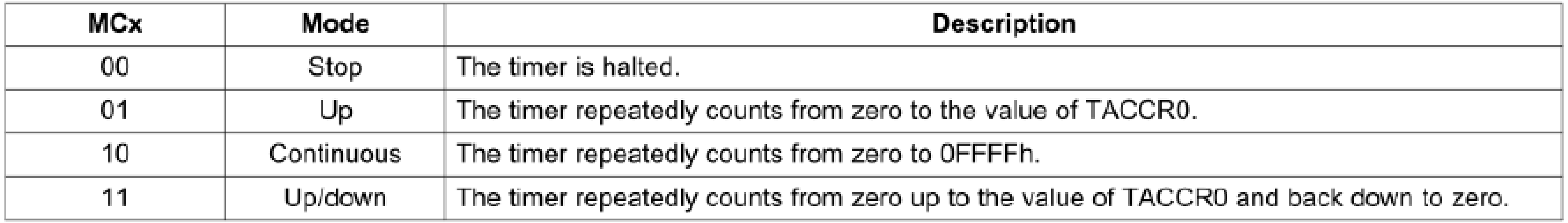
- Allows selecting the timer clock source.
- The prescaler (IDx) divides the input clock.

- **16-bit Timer Register (TAR)**
  - Stores the current timer value.
  - Can be cleared/reset using TACLR.
- **Capture/Compare Logic (CCR0, CCR1, CCR2)**
  - Stores values for comparison.
  - Generates interrupts and toggles output based on comparisons.
- **Capture Mode**
  - Captures external events using CCI (Capture/Compare Input).
  - Can synchronize with external signals.
- **Output Unit**
  - Generates output signals based on the configured mode (OUTMODx).
  - Can generate PWM or toggle signals.

Applications of Timer_A

- Generating **PWM** signals.
- Measuring **pulse widths** of external signals.
- Implementing **time delays**.
- **Counting events** from an external source.
- Controlling **servo motors and LEDs**.

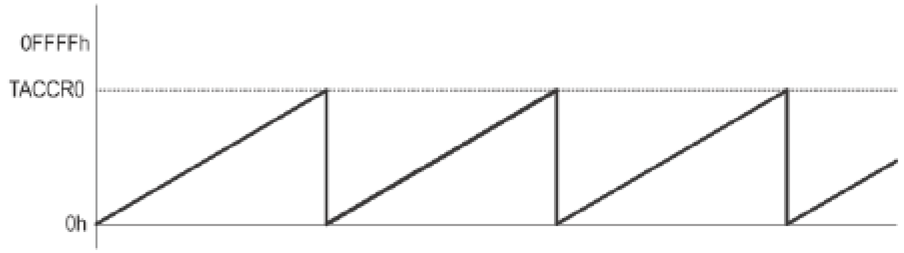
Timer Mode Control

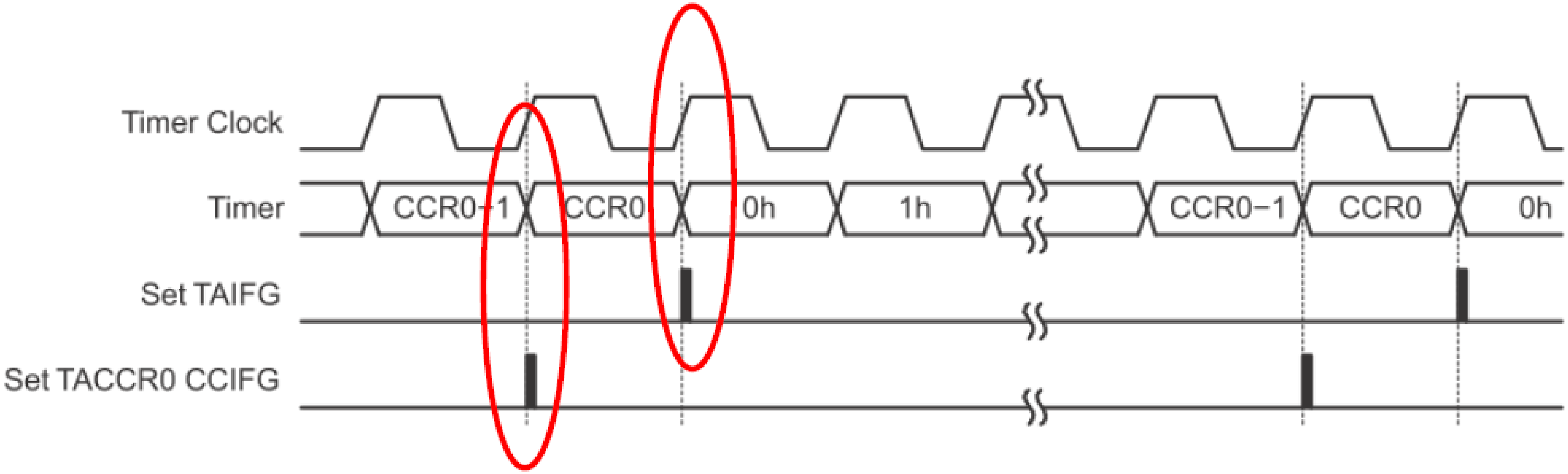


MCx	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of TACCR0.
10	Continuous	The timer repeatedly counts from zero to 0FFFFh.
11	Up/down	The timer repeatedly counts from zero up to the value of TACCR0 and back down to zero.

Up-Mode Flag setting

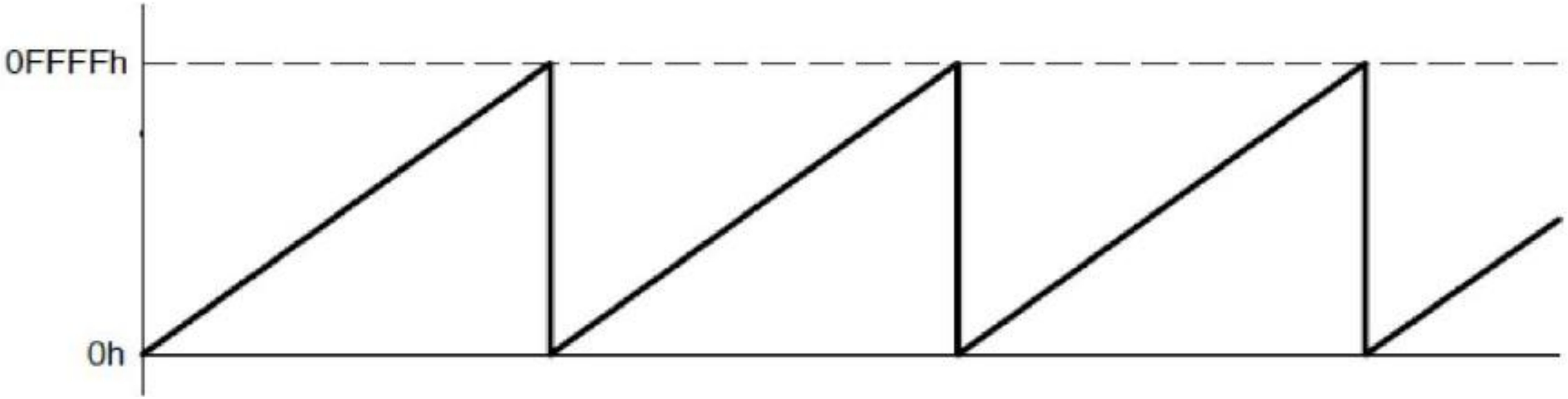
- The TACCR0 CCIFG interrupt flag is set when the timer counts to the TACCR0 value.
- The TAIFG interrupt flag is set when the timer counts from TACCR0 to zero.





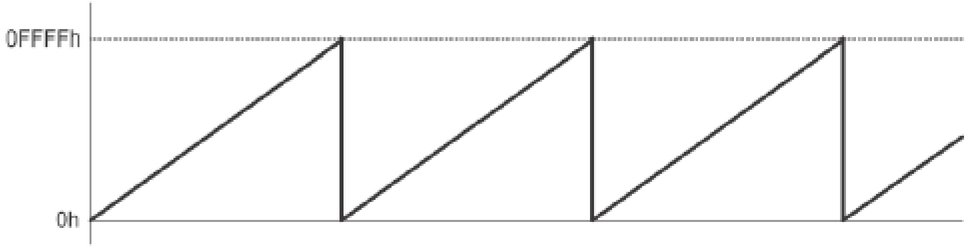
Continuous Mode

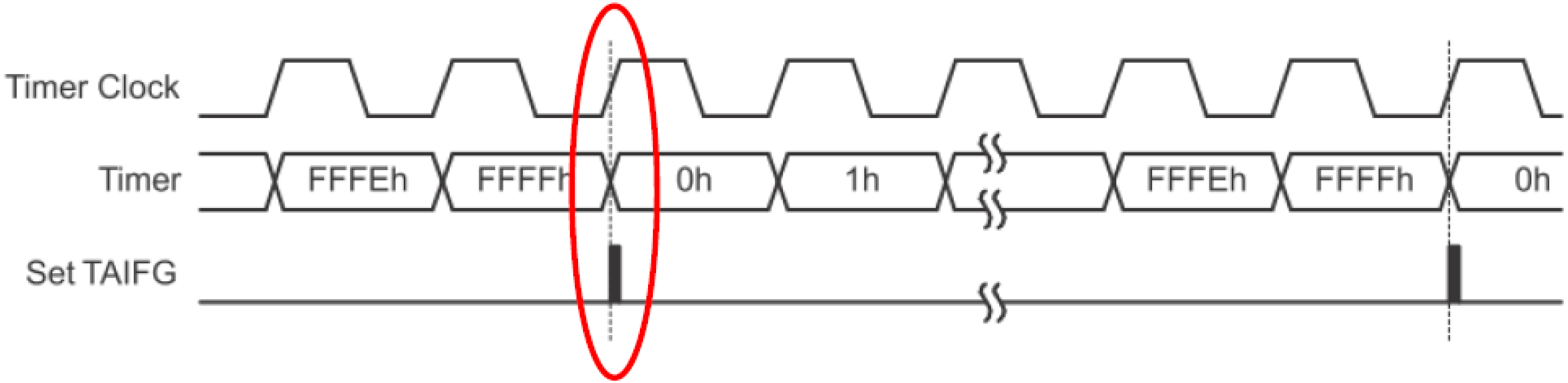
- In the continuous mode, the timer repeatedly counts up to 0FFFFh and restarts from zero as shown in the figure.
- The number of timer counts in the period is 0xFFFF + 1
- The capture/compare register TAxCCR0 works the same way as the other capture/compare registers.



Continuous Mode Flag setting

- The TAIFG interrupt flag is set when the timer counts from 0FFFFh to zero





```c
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;                     // Stop WDT

    // Configure GPIO
    P1DIR |= BIT0;                                // P1.0 output
    P1OUT |= BIT0;                                // P1.0 high

    // Disable the GPIO power-on default high-impedance mode to activate
    // previously configured port settings
    PM5CTL0 &= ~LOCKLPM5;

    TA0CCTL0 |= CCIE;                             // TACCR0 interrupt enabled
    TA0CCR0 = 50000;
    TA0CTL |= TASSEL__SMCLK | MC__CONTINOUS;      // SMCLK, continuous mode

    __bis_SR_register(LPM0_bits | GIE);           // Enter LPM3 w/ interrupts
    __no_operation();                             // For debugger
}
```

```c
// Timer A0 interrupt service routine
volatile unsigned int counter = 0;

#pragma vector = TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
    counter++;
    if (counter >= 20)  // 20 * 50ms = 1 second
    {
        P1OUT ^= BIT0;
        counter = 0;
    }
    TA0CCR0 += 50000;
}
```

TASKS:

Create a delay of 2.5 sec .. the LED should ON after 2.5 sec and OFF for 2.5 sec …

Create a delay of 1 sec .. the LED should ON after 1 sec and OFF for 1 sec …

Create a delay of 500 msec .. the LED should ON after 500 msec and OFF for 500 msec …