KPIT

Session-VI
Timers

Session outline

- Role of timing in Automotive subsystems
- Working principle of Timer
- Various Timer modes in Atmega328P
- Creating delays using timer

Why timing is critical ?

- **Role of timing in Fuel Injection System:**
    - **The main purpose of the fuel injection system is to deliver fuel into the cylinders of an engine.**
- Fuel must be injected at the proper time, that is, the injection timing must be controlled and
- The correct amount of fuel must be delivered to meet power requirement, that is, injection metering must be controlled.

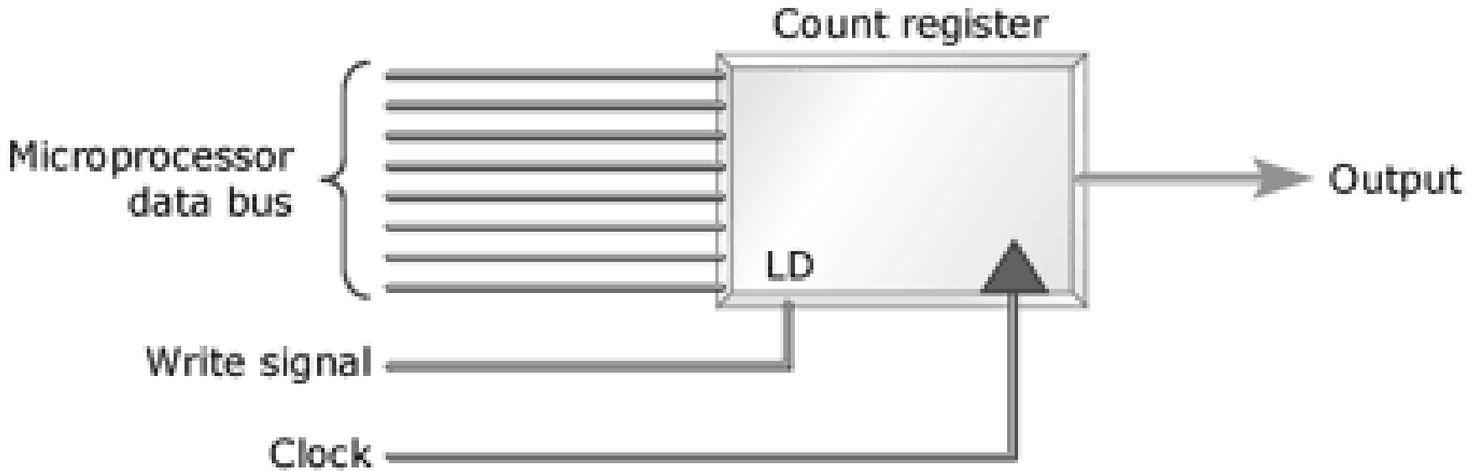
- **Role of timing in Battery management system:**
- We need to detect a period such as how long a car was not used, or a battery was not charged

- **Automated Wiper Control applications:**
- Sense rain sensor data every 5 second

Components of timer module

Counter/timer hardware is a crucial component of most embedded systems. In some cases, a timer is needed

- to measure elapsed time;
- count some external events.
- to generate PWM



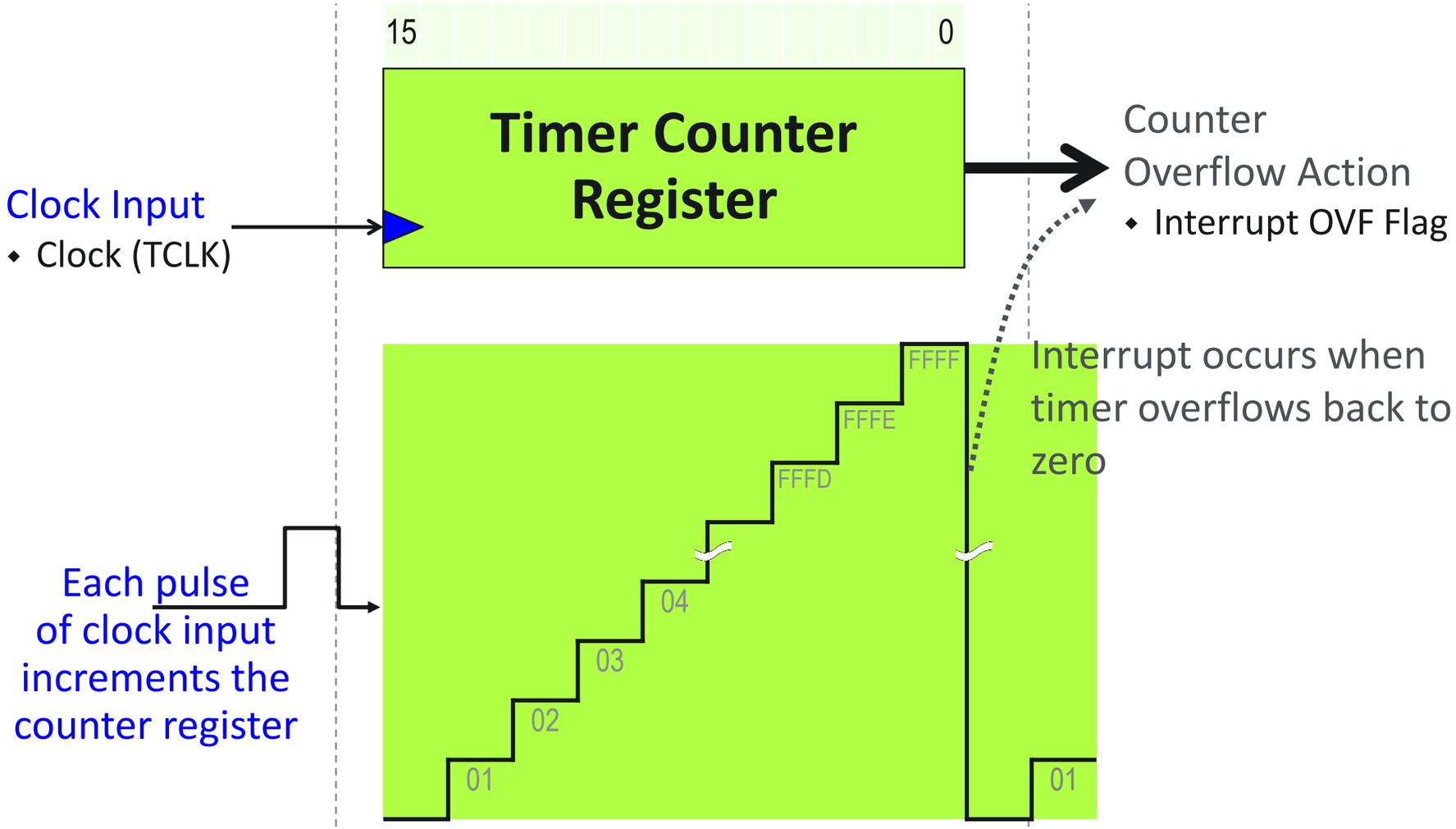
A simple counter/timer hardware

Timer/Counter Basics

Input → Counter → Action

Notes
- Timers are often called “Timer/Counters” as a counter is the essential element
- “Timing” is based on counting inputs from a known clock rate

Timer/Counter Basics



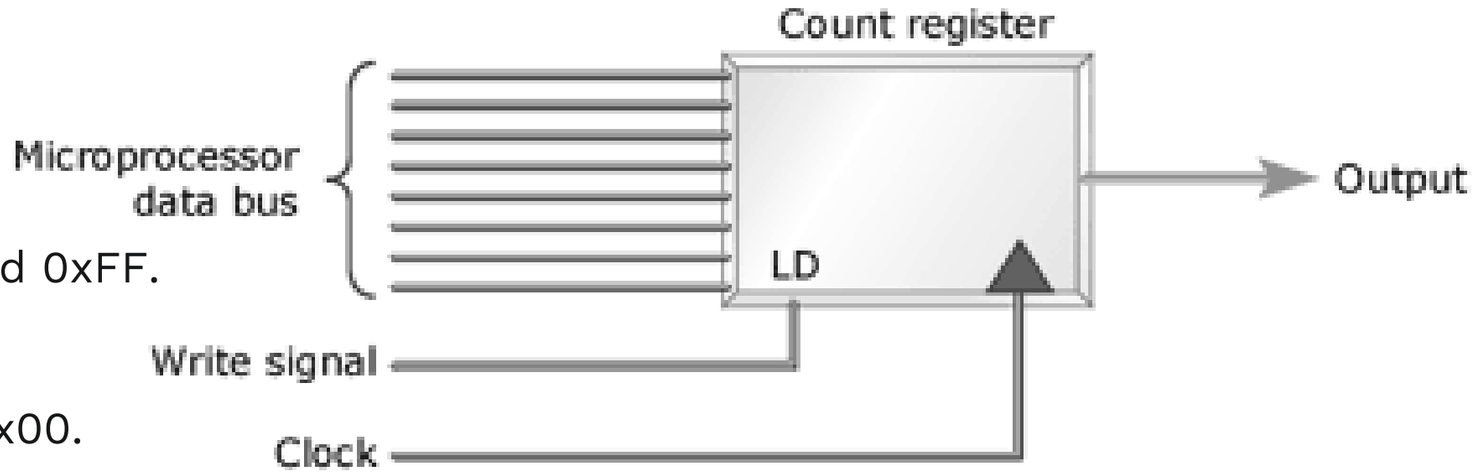
- Timers are often called “Timer/Counters” as a counter is the essential element
- “Timing” is based on counting inputs from a known clock rate
- Actions don’t occur when writing value to counter

Working principle of timer

Software loads the count register with an initial value between 0x00 and 0xFF (in case of 8-bit). Each subsequent transition of the input clock signal increments that value.

Up mode
it counts up from the initial value toward 0xFF.

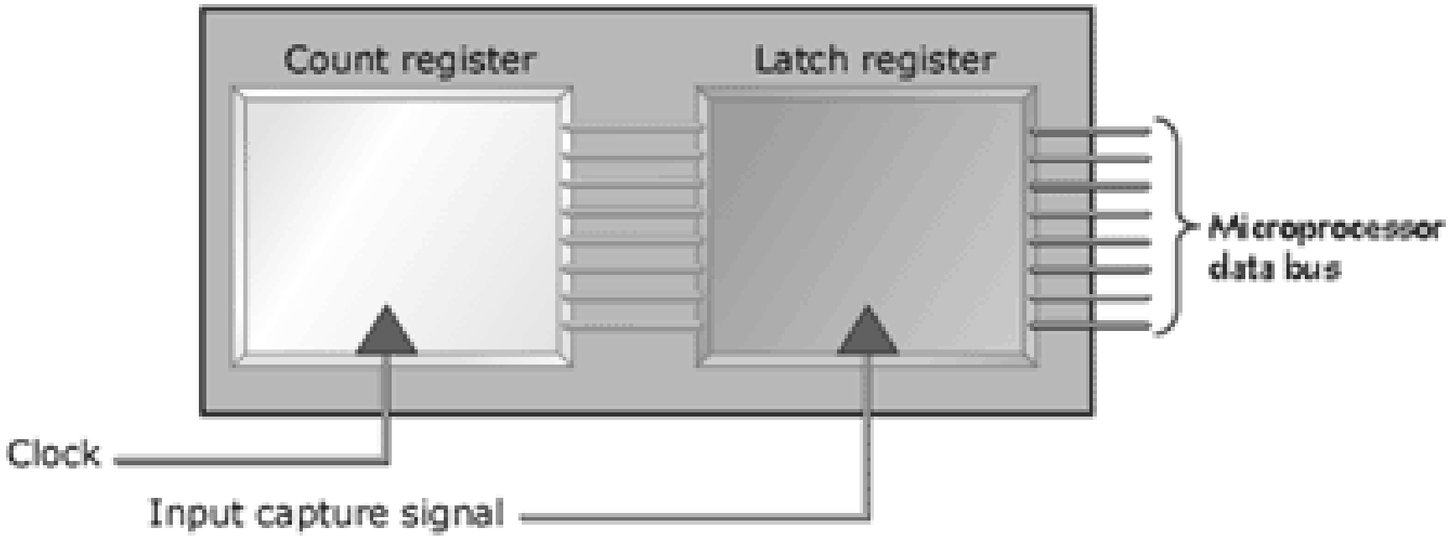
Down mode
A down counter counts down, toward 0x00.



When the counter overflows or reaches 0x00, the output signal is asserted. The output signal may thereby trigger an interrupt at the processor or set a bit/Flag that the processor can read. To restart the timer, software reloads the count register with the same or a different initial value.

Timers and its applications

- A timer with automatic reload capability will have a latch register to hold the count written by the processor.
- After overflow, count register automatically reloads the contents of the latch into the count register. Since the latch still holds the value written by the processor, the counter will begin counting again from the same initial value.



Various applications of auto reload mode:

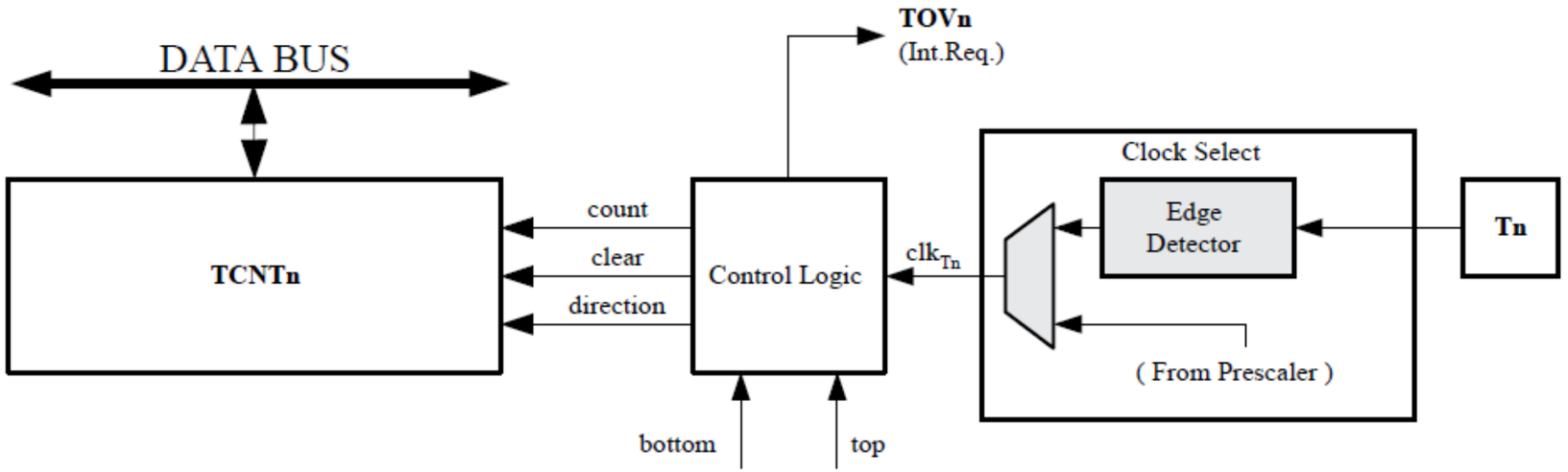
generate a periodic interrupt like a real-time operating system (RTOS) timer tick,

provide a baud rate clock to a UART,

or drive any device that requires a regular pulse.

Timer module support in Atmega328P

- There are three types of timers –
- TIMER0: 8- bit timer
- TIMER1: 16-bit timer
- TIMER2: 8-bit timer



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

Various timer operation modes in Atmega328P

- Normal Mode of operation
- Clear Timer on Compare Match (CTC) Mode
- Fast PWM Mode, Phase correct PWM mode

For more information; Please refer ATmega328/P datasheet

Example

Write a Program for flashing an LED (connected to PD7 pin) every 8 ms

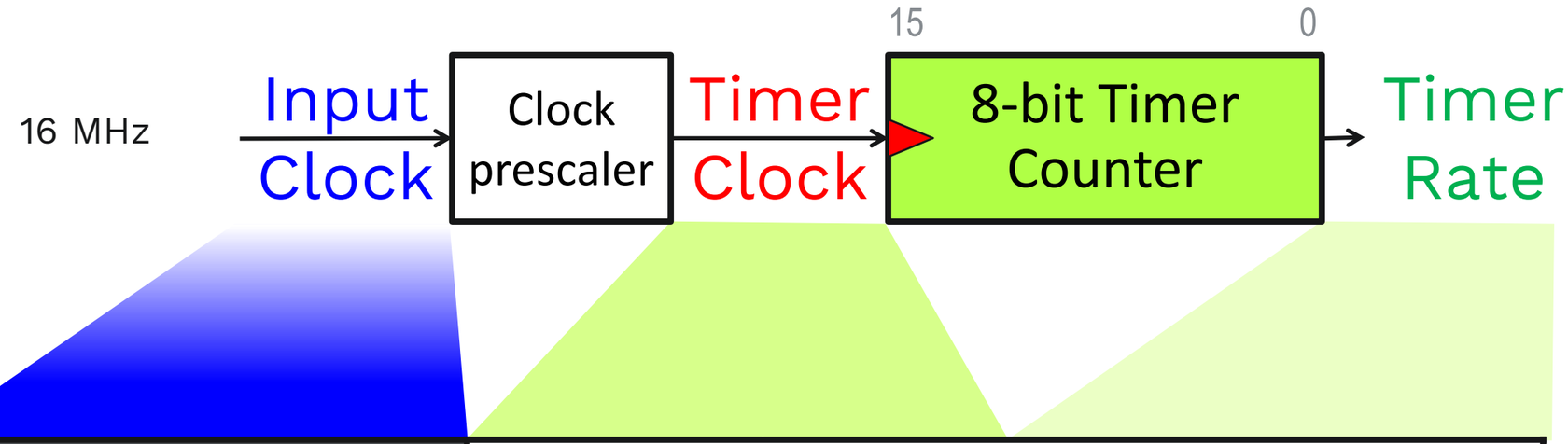
Timer count value = (delay required/clock time period) - 1

Representational model

Our task is to know the timer count value at 8 msec

As timer count increases by 1 per timer tick, Hence count value will be different under various timer clock frequency.

Understanding timer clock



System Clock 16 MHZ	Clock Pre-scaler	Timer Module	
	Divider	Timer clock Frequency	Timer count for delay of 8 ms
	8	2 MHz	15999
	64	250 KHz	1999
	256	62.5 KHz	499
	1024	15625 Hz	124

Our approach for example

- Hence for a delay of 8 ms, we need a timer count of 124. This can easily be achieved with an 8-bit counter (MAX = 255). Hence 8-bit timer (Timer0) can be used.

- We need to keep a track of the counter value. As soon as it reaches 124, we toggle the LED value and reset the counter

- STEP-I: Configure clock for timer by configuring Timer pre-scaler register TCCR0
    - TCCR0 |= (1 << CS02)|(1 << CS00);

- STEP-II: Initialize timer count to zero i.e. TCNT0 = 0;

- STEP-III: Monitor TCNT0 , (TCNT0 >= 124) in while(1) loop; If TRUE
- Toggle LED;
- Clear Timer counter to zero; TCNT0=0

Sample C program

```
int main(void)
{
    // connect led to pin PORTD pin 7
    DDRD = 0xFF;
    // initialize timer
    Timer0_init();
    // loop forever
    while(1)
    {
        // check if the timer count reaches 124
        if (TCNT0 >= 124)
        {
            PORTC ^= (1 << 0);    // toggles the led
            TCNT0 = 0;            // reset counter
        } } }
```

```
void Timer0_init()
{
 // set up timer with pre-scaler = 1024
    TCCR0B |= (1 << CS02)|(1 << CS00);

    // initialize counter
    TCNT0 = 0;
}
```

Exercise 1

- Write a Program for flashing an LED (connected to PD7 pin) every 1 sec

Hint:

Calculate timer count value for 1 sec delay for various clock prescalers.

Chose appropriate counter (8-bit or 16-bit)

Decide the timer clock and Timer module (Timer0, Timer1 or Timer2)