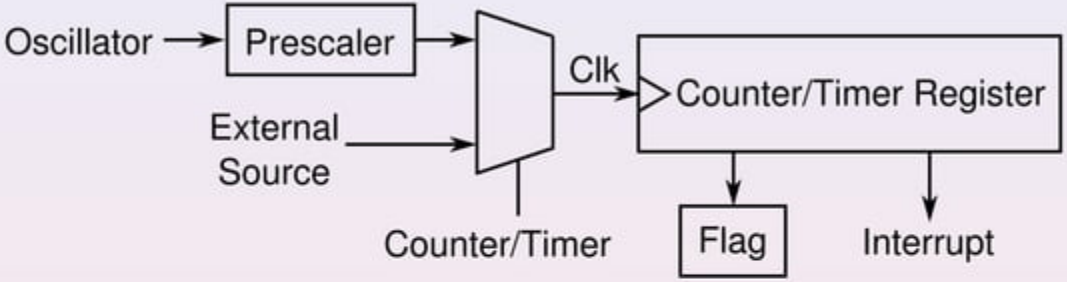
Timer/Counters in Atmega328P

- Timer/Counter0 – 8 bit.
- Timer/Counter1 – 16 bit.
- Timer/Counter2 – 8 bit.

Timer/Counter can be used to cause time delay, to count events, or to generate PWM signal.

General Features of Timer/Counter

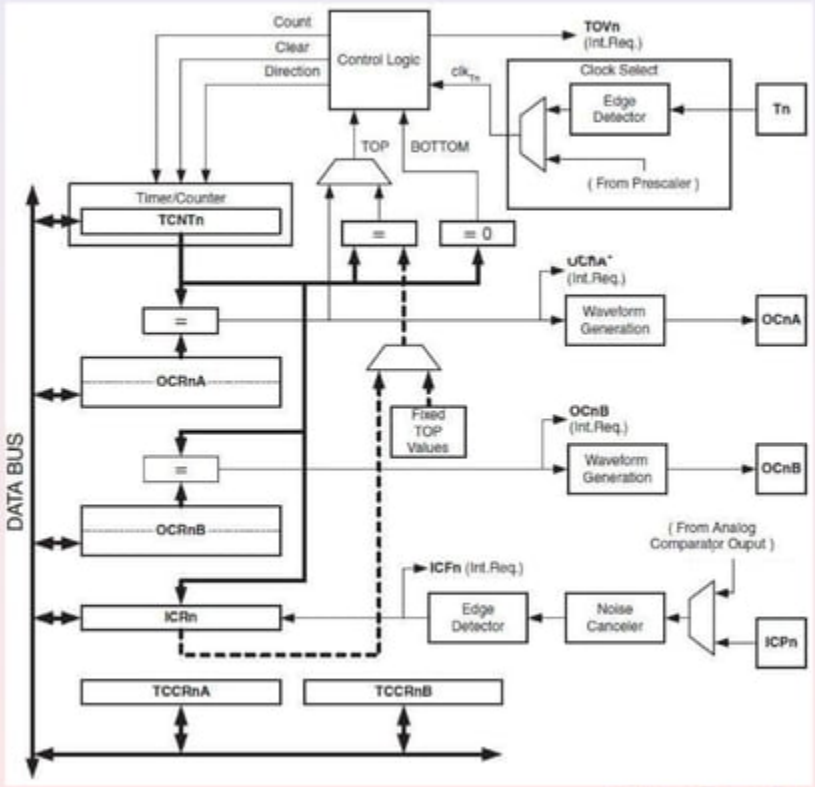


- External Source is fed as Clk input to Timer/Counter Register when used as counter.
- Oscillator is fed as Clk input to Timer/Counter Register when used as timer.
- Prescaler divides the oscillator frequency by a specific constant.
- When Timer/Counter register reaches a specific value,a flag will be set and optionally an interrupt can be issued.

Features of 16-bit Timer/Counter1 with PWM

- Permits 16-bit PWM
- Two independent Output Compare Units
- Double Buffered Output Compare Registers
- One Input Capture Unit
- Input Capture Noise Canceler
- Clear Timer on Capture Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator
- Variable PWM period
- Frequency Generator
- External Event Counter
- Four independent interrupt sources (TOV1, OCF1A, OCF1B, ICF1)

Timer/Counter1 Block Diagram



Registers Pertaining to Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter1 High and Low bytes

Bit	7	6	5	4	3	2	1	0	
(0x89)	OCR1A[15:8]								OCR1AH
(0x88)	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register1A High and Low Bytes

Bit	7	6	5	4	3	2	1	0	
(0x8B)	OCR1B[15:8]								OCR1BH
(0x8A)	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register1B High and Low Bytes

Registers Pertaining to Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter Control Register1A

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter Control Register1B

Bit	7	6	5	4	3	2	1	0	
(0x82)	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter Control Register1C

Registers Pertaining to Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter1 Interrupt Flag Register1

- By setting appropriate bits in TIMSK1 the corresponding interrupt can be enabled.
- When ISR is executed corresponding to a timer/counter1 event, the corresponding flag is cleared in the TIFR1.
- By setting appropriate bits in TIFR1 the corresponding flag can be cleared.

Timer/Counter1 Mode 0 and Mode 4 Interrupt Vectors

ISR Name to be used in C	Source of Interrupt
TIMER1_CAPT_vect	Timer/Counter1 Capture Event
TIMER1_COMPA_vect	Timer/Counter1 Compare Match A
TIMER1_COMPB_vect	Timer/Counter1 Compare Match B
TIMER1_OVF_vect	Timer/Counter1 Overflow

Modes of Operation of Timer/Counter1

- Normal Mode (also known as Mode 0)
- Clear Timer on Compare (CTC) Match Mode (also known as Mode 4 and Mode 12)
  - In Mode 4 comparison is done with OCR1A.
  - In Mode 12 comparison is done with ICR1.
- Fast PWM Mode
- Phase Correct PWM Mode
- Phase and Frequency Correct PWM Mode

The last three modes are PWM modes. Here we discuss only Normal Mode and CTC mode. Mode selection can be made by setting appropriate bits in TCCR1A and TCCR1B registers.

Setting Timer/Counter1 Clock Source and Prescaler

CS12	CS11	CS10	Effect
0	0	0	No clock (Timer/Counter stops)
0	0	1	clk (no prescaling)
0	1	0	clk/8
0	1	1	clk/64
1	0	0	clk/256
1	0	1	clk/1024
1	1	0	Ext. clk on T1 pin, falling edge
1	1	1	Ext. clk on T1 pin, rising edge

- CS12, CS11 and CS10 bits are in TCCR1B register.
- Timer/Counter1 is inactive when no clock source is selected.
- Timer/Counter1 starts when clock source is set.

Normal Mode (Mode 0)

- This mode can be selected by making WGM13 = 0, WGM12 = 0 in TCCR1B and WGM11 = 0, WGM10 = 0 in TCCR1A.
- In this mode there is no comparison with either ICR1 or OCR1A.
- In this mode the timer counts up for every clock cycle until it reaches $FFFF (also referred to as MAX) and then resets to $0000 in the next clock cycle.
- When the timer value changes from $FFFF to $0000, the TOV1 flag is set.
- An interrupt can be generated when TOV1 flag is set by setting TOIE1 bit in TIMSK1.

CTC Mode (Mode 4)

- This mode can be selected by making WGM13 = 0, WGM12 = 1 in TCCR1B and WGM11 = 0, WGM10 = 0 in TCCR1A.
- In this mode the timer counts up until it reaches the value stored in OCR1A (also referred to as TOP) and then resets from TOP to $0000.
- When the timer resets from TOP to $0000, the OCF1A flag will be set.
- An interrupt can be generated when OCF1A flag is set by setting OCIE1A bit in TIMSK1.

Example C Statements to Set Up Timer1

```
//EXAMPLE-1
//Write C statements to setup Timer/Counter1 in Mode 0
//as a timer. Select a prescaler value of 64. It is not necessary
//to cause interrupt because of timer overflow.
TCCR1A=0x00;TIMSK1=0x00;
TCCR1B=0x03;

//EXAMPLE-2
//Write C statement to setup Timer/Counter1 in Mode 0 as a timer.
//Select a prescaler value of 64. Enable timer overflow interrupt.
TCCR1A=0x00; TIMSK1=0x01; sei();
TCCR1B=0x03;

//EXAMPLE-3
//Write C statement to setup Timer/Counter1 in Mode 4 with prescaler
//value 8; Interrupt not necessary. Assume that the number to be
//loaded in OCR1 is 3D08.
TCCR1A=0x00; OCR1BH=0x3D;
OCR1BL=0x08; TIMSK1=0x00; TCCR1B=0x0A;
```

Maximum Delay using Timer 1 in Mode 0

Calculate maximum time delay that can be realized using Timer 1 in Mode 0 with prescaler value of 1024. Assume that MCU clock frequency is 16 MHz.

Frequency of clock signal applied to Timer 1 is given by

$$f_{clk} = \frac{16 \times 10^6}{1024} = 15625 \text{ Hz}$$

Therefore maximum delay possible is

$$t_{delay(max)} = \frac{65536}{15625} = 4.1943 \text{ s}$$

Timer1 Initial Count in Mode 0 for a Given Delay

Assuming MCU clock frequency to be 16 MHz and prescaler value of 1024, find the initial number N to be loaded in TCNT1H and TCNT1L to get a time delay close to 1 second in Mode 0.

No. of steps required to cause a delay of 1 s is given by

$$n = 15625 \times 1 = 15625 = 65535 - N + 1$$

Therefore

$$N = (49911)_{10} = 0xC2F7$$

So, TCNT1H = C2 and TCNT2L = F7.

Using Timer1 to cause Delay

Write C code to setup Timer1 (16-bit timer) of Atmega328P MCU to give a 1 s time delay assuming that the MCU is operating at 16 MHz clock with prescaler set to 1024. Use timer interrupt to blink an LED connected to an output port line. Show timer calculations and hardware connections clearly.

The number to be loaded in Output Compare Register1 (OCR1) is given by

$$\text{OCR1} = \left(\frac{16 \times 10^6}{1024} \times 1\right) - 1 = (15624)_{10} = (3D08)_{16}$$

Timer/Counter1 Program Using Interrupt

```
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
    OCR1A = 0x3D08;
    TCCR1B |= (1 << WGM12);    // Mode 4, CTC on OCR1A
    TIMSK1 |= (1 << OCIE1A); //Set interrupt on compare match
    TCCR1B |= (1 << CS12) | (1 << CS10); //Prescaler to 1024.
    sei();     // enable interrupts

    while (1)
    {
        // we have a working Timer
    }
}

ISR (TIMER1_COMPA_vect)
{
    // action to be done every 1 sec
}
```

Using Timer0 as Counter

Assuming that a 1 Hz clock pulse is fed into pin T0 of Atmega328P MCU, write C code to use the TOV0 flag to extend Timer0 to a 16-bit counter and display the counter on PORTC and PORTD.

```
#include <avr/io.h>

int main (void)
{
    PORTB = 0x01;    //Enable pull-up resistor of PB0
    DDRC = 0xFF;     //Configure Port C as output.
    DDRD = 0xFF;     //Configure Port D as output.
    TCCR0 = 0x00;    //Initialize clock source.

    while(1)
    {
        do
        {
            PORTC = TCNT0;
        }while(TIFR & (0x1<<TOV0)==0);//Wait for TOV0 to become 1.

        TIFR = 0x1<<TOV0; //Clear TOV0.
        PORTD ++;         //Increment PORTD.
    }
    return 0;
}
```