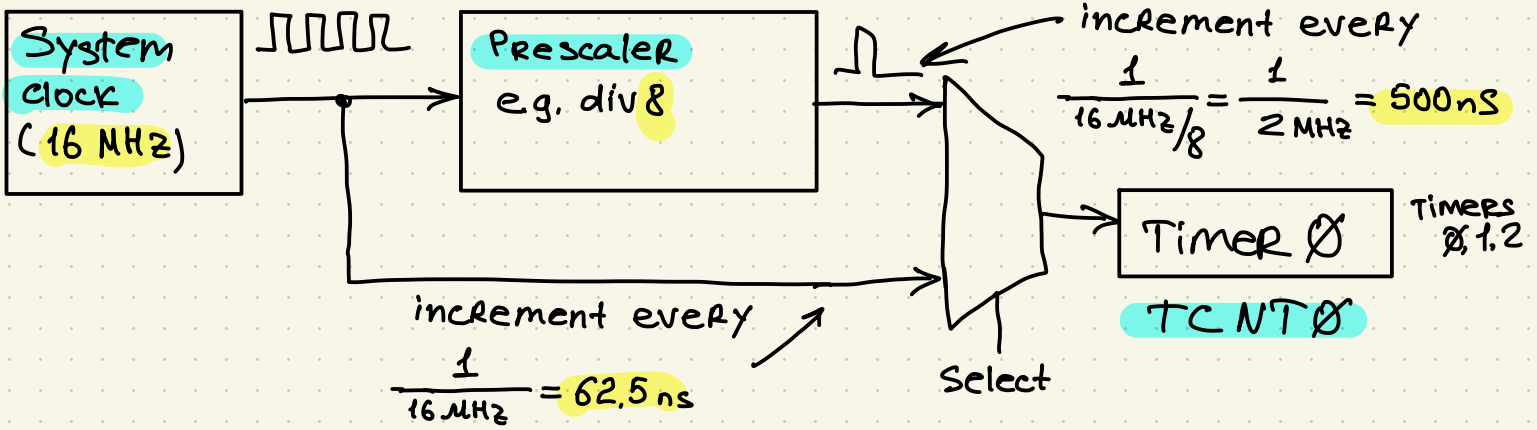
24-OCT-23

Arduino Timers



Timer1 count

65529
65530
65531 == (OCR1A: 65531) → Compare match interrupt (TIMER1_COMPA)
65532
65533
65534
65535
0 → overflow interrupt (TIMER1_OVF)
1
2 .... ICP1 (PB0) logic change → ISR1 = 3 → input capture interrupt (TIMER1_CAPT)
3
...

1) Timer0 - 8 bit (0..255)
   - compare match
   - overflow
2) Timer1 - 256 bit (0..65535)
   - compare match
   - overflow
   - input capture (?)
3) Timer2 - 8 bit (0..255)
   - compare match
   - overflow

TIMER Interrupts (PRESCALERS)

1 → 16 MHZ (62.5 ns)
8 → 2 MHZ (500 ns)
64 → 250 kHz (4 μs)
256 → 62.5 kHz (16 μs)
1024 → 15.625 kHz (64 μs)

} microseconds

= In register TCCR1B (Timer counter/control Register)

CS12, CS11, CS10 ← bites

1 0 1 = 1024

NOTE that TCCR1A register must set to 0: TCCR1A=0;

TCCR1B Register:


```
TCCR1B |= (1 << CS11);                  //CLK/8
TCCR1B |= (1<<CS10) | (1<<CS11);        //CLK/64
TCCR1B |= (1 << CS12);                  //CLK/256
TCCR1B |= (1<<CS10) | (1<<CS12);        //CLK/1024
```

16 MHZ → for 256 bit register (Timer 0 & Timer 2)

16.000.000 / 256 = 62 500 tick per second (overflow, i.e. ISR)

(256 bits)

↳ 16 µs (1/62.500)

Prescaler

16.000.000 / 64 = 250.000 Hz (tick per second)

↳ 4 µs (1/250.000) 4 microseconds for one tick.

If 4 microseconds multiple to 250 = 1 ms : 0,000004 · 250 = 0,001 sec.

So, to get 1 ms time interval, save 249 to OCR1A and allow interrupts if timer count match that 249 value.

```
TIMSK1 |= (1<<OCIE1A);
```

Register TIMSK1:

Setting bits 1 and 2 we can enable time compare interrupt on the value defined in registers OCR1A and OCR1B

OCR = Output Compare Register

Compare Match Interrupt

How to use PRESCALAR

i.e: We need a LED to blink every 500 ms. So we need to trigger the Timer interrupt each 500 ms. The system clock is 16 MHz, so each pulse is 62 ns. In order to count up to 500 ms with 62 nanosecond step, we would need to count up to 8.000.000. Which is not enough for 16-bit Register (65535). But, if we use prescaler 256, each pulse will be 16 microseconds long.

500 000 (microseconds) : 16 = 31.250 pulses for

500 ms = 500.000.000 nanoseconds

save 31.250 - 1 to Register OCR1A. → 31249

- Timer1 will compare his value to OCR1A and after Timer 1 reaches 31249 ticks, interrupt will occur:

TIMER1_COMPA_vect () { ... add your code }

FORMULA TO calculate:

Compare Match Register Value:

for time in seconds:

$$\left(\frac{16\,MHz \cdot T_{sec}}{Prescalar} - 1\right) = OCR\ Register$$

for time as Frequency:

$$\left(\frac{16\,MHz}{\left(\frac{Prescalar}{Freq\ (Hz)}\right)}\right) - 1 = OCR\ Register$$

1) Timer1 call ISR every 100 ms with Prescalar = 256. 100 ms = 0.1 s

$$\frac{16.000.000 \cdot 0,1}{256} - 1 = 6249$$

100 ms → 10 Hz.

$$\frac{16.000.000}{\left(\frac{256}{10}\right)} - 1 = 6249$$

2) Timer1 call ISR every 500 ms Prescalar = 256 500 ms = 0,5 s

$$\frac{16.000.000 \cdot 0,5}{256} - 1 = 31249$$

500 ms → 2 Hz.

$$\frac{16.000.000}{\left(\frac{256}{2}\right)} - 1 = 31249$$

Code example for Compare Match Register:

```
void setup(){
    cli(); // disable interrupts till we make settings
    TCCR1A=0; // reset entire Register
    TCCR1B=0 // reset entire Register
    TCCR1B |= b00000100; //set CS12=1, Presciler=256
    TCNT1 =0; //Reset TimerI value to 0.
    TimsK1 |= b00000010; //enable compare matchA.

    OCR1A = 31249 ; //compare RegisterA to this value
    SEi(); //enable interrupts
}
ISR(TIMER1_COMPA_vect) {
  TCNT1 =0; //set timer to 0, to stop count after ISR!
  LED_STATE=!LED_STATE;
}
```

note: for automated reset Timer1 to 0 after it match the Register value, use CTC mode (reset on match)

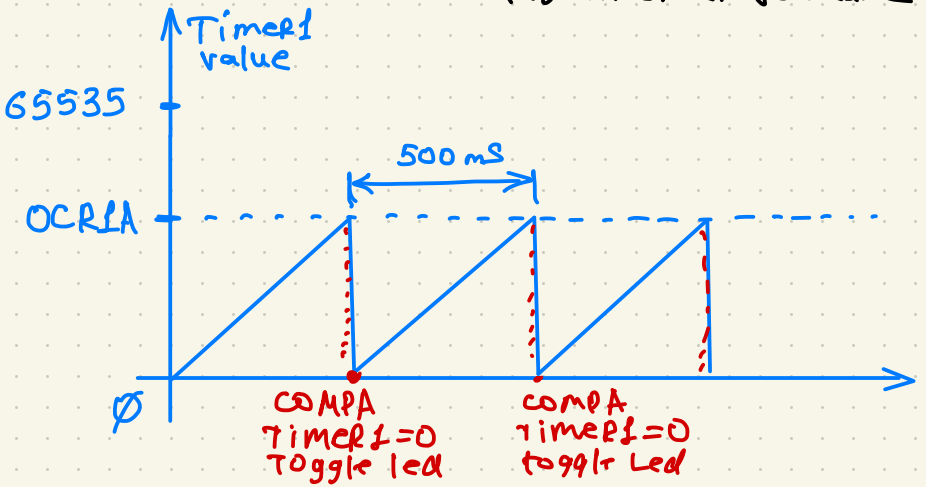
- need to set bit WGM12 to 1 in TCCR1B register:

```
TCCR1B |= (1 << WGM12);
```

- note that it is same as use

```
TCNT1 = 0; // set Timer to 0 again and again
```

every time inside the ISR, but if every line is counted in order to make ISR as short as possible...



Timer overflow

main idea: We need to set the TimerStartValue to a number. So it start to count from it to 65535 (max value) and then overflow interrupt occurs after defined period of time.

```
void setup()
{
  TCCR1A = 0;
  TCCR1B = 0;
  TCCR1B |= 0b00000011; // Prescalar = 64
  TCNT1 = 40536; // see formula
  TIMSK1 |= 0b00000001; // enable Timer OVF ISR
}
```

CS11 CS10

CS12 = 0

TOIE1 = 1.

```
ISR(Timer1_OVF_vect)
{
  TCNT1 = 40535; // set Timer init value again!
  ... // do something
}
void loop() { // do nothing! }
```

FORMULA TO CALCULATE

$$TCNT1 = 65536 - \frac{F_{clock}\ (Hz)}{Prescalar \times F_{target}\ (Hz)}$$

i.e. we need 100ms to call interrupt.

100 ms → 10 Hz

Prescaler = 64 (for example)

$$TickCount = \frac{16.000.000}{\left(\frac{64}{10}\right)} = 25.000$$

$$\underline{Timer\ Preload} = 65535 - 25.000 = 40.535$$

↳ Timer start value.

---

2 Hz — 500 ms (0,5s)
10 Hz — 100 ms (0,1s)
50 Hz — 20 ms (0,02s)
100 Hz — 10 ms (0,01s)
1000 Hz — 1 ms (0,001s)

$$TCNT1 = 65536 - \frac{F_{clock} \cdot Timerval}{Prescalar}$$

i.e. we need 100ms interval to call interrupt.

100 ms = 0,1s.

Prescaler = 64 (for example)

$$TickCount = \frac{16\ MHz \cdot 0,1}{64} = 25.000$$

The TickCount = 25.000, this is number of ticks to count 100ms. So, if Timer start to count from 65535 - 25000, it will take 100ms to call overflow interrupt:

$$\underline{Timer\ Preload} = 65535 - 25.000 = 40.535$$

↳ Timer start value.