COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

EGCO334: Microprocessor and Interfacing

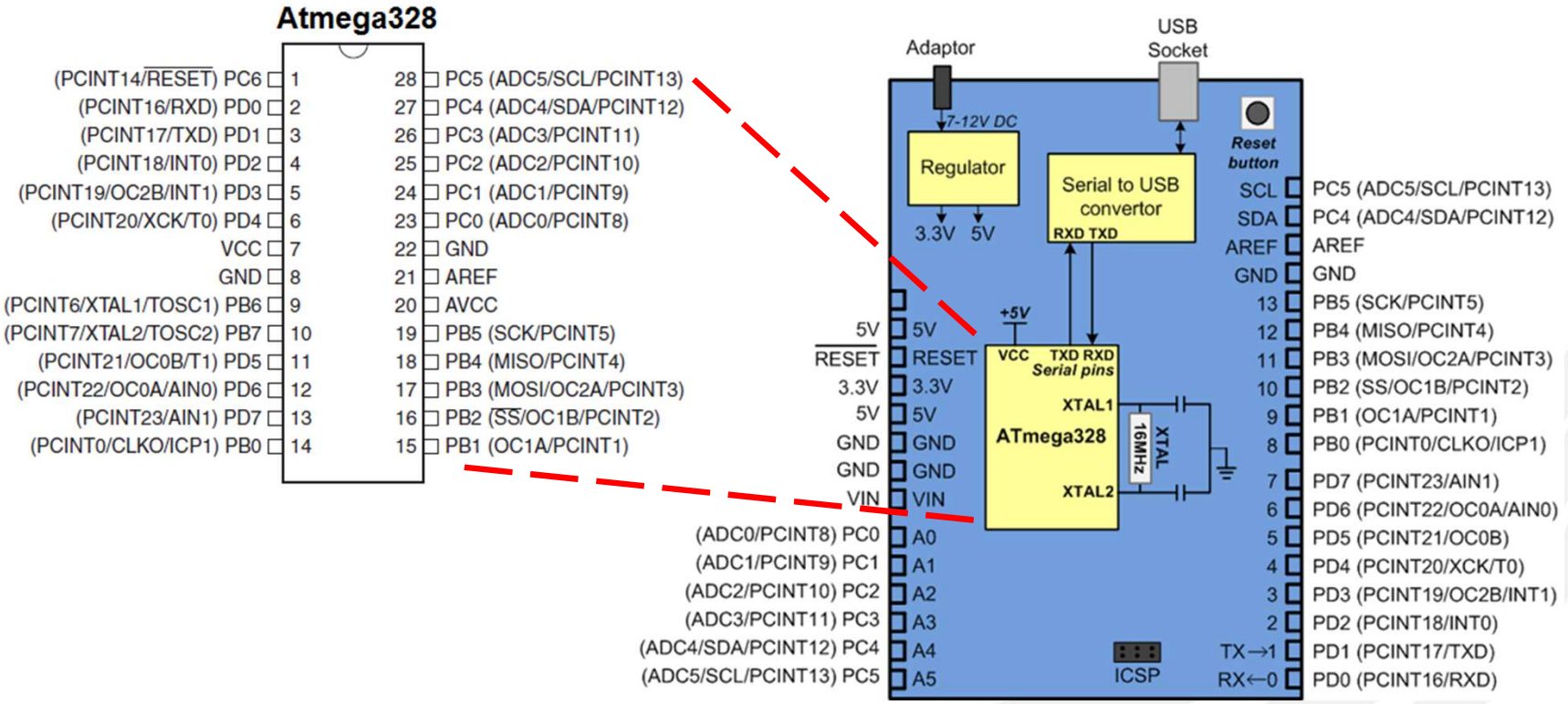
AVR Timer/Counter

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

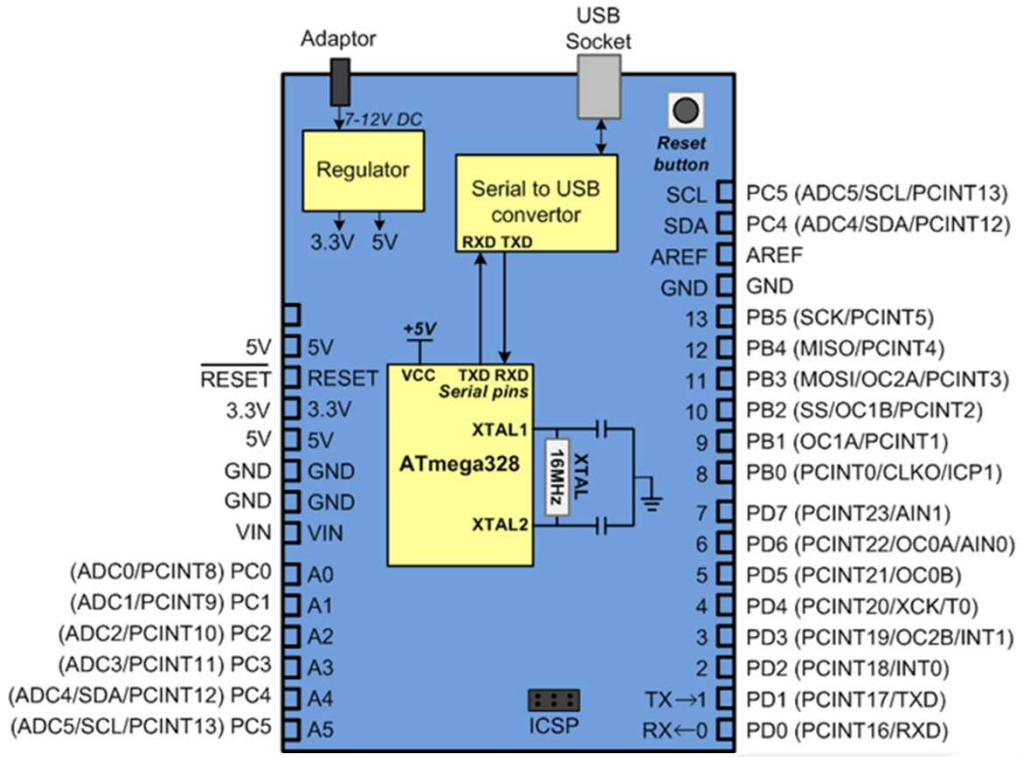
Outline

- Timer/Counter Concept
- AVR 328P Timer/Counter Interfaces
- AVR 328P Timer/Counter Registers
  - Prescaler
  - Timer0,2
  - Timer1
  - Counter

AVR 328P and Arduino HW



AVR 328P and Arduino HW



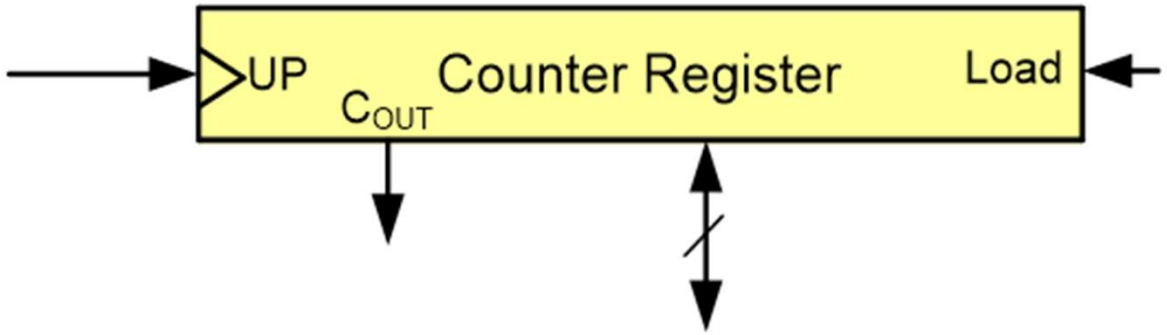
Timer/Counter

Timers in AVR328P

- 3 Timers
- Two 8-bit timers and ne 16-bit timer in ATmega328
  - Timer/Counter0 - 8-Bit
  - Timer/Counter1 - 16-bit
  - Timer/Counter2 - 8-bit

Timer/Counter

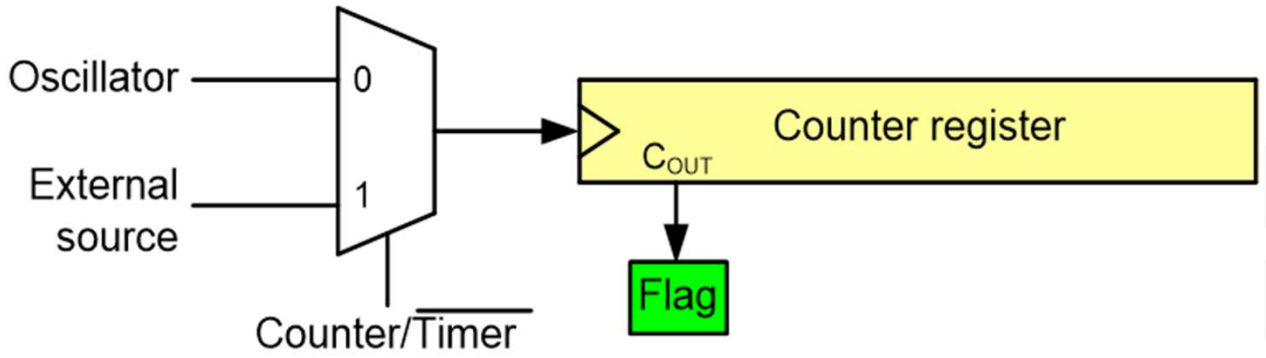
Basic Counter



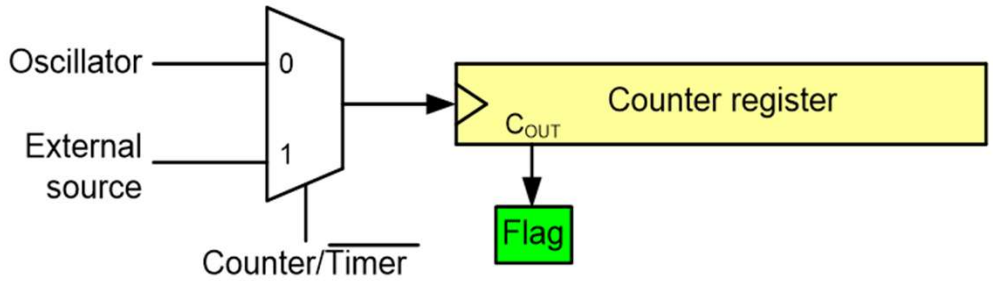
Timer/Counter

A generic timer/counter

- Delay generating
- Counting
- Wave-form generating
- Capturing



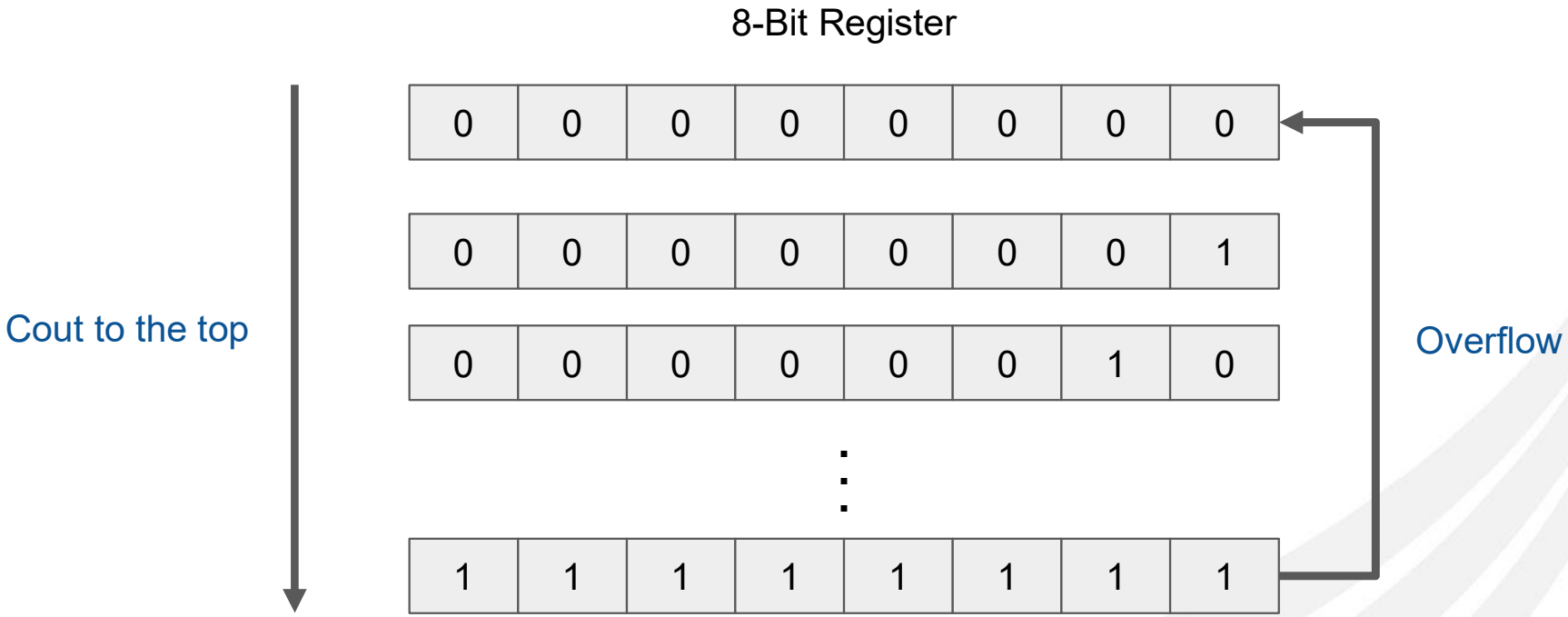
Timer/Counter



- 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 (Overflow) will be set and optionally an interrupt can be issued

Timer/Counter

Basic Counter/Timer Concept



Timer/Counter

Basic Counter/Timer Concept

- 8-Bit timer/counter starts counting from 0 and goes upto 255
- 16-Bit timer/counter starts counting from 0 to 65535
- Once the timer reaches the maximum value it “overflows” then restart from 0 (optionally overflow flag is set)

$$Time\ Period = \frac{1}{Frequency}$$

$$Timer\ Count = \frac{Required\ Delay}{Clock\ Time\ Period} - 1$$

What is the maximum delay for 4MHz processor

- **16-Bit timer?**
- **8-Bit timer?**

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

Timer/Counter

What will we do if we want 100ms delay?

- Reducing the clock frequency?
- Are there other solutions?

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

Timer/Counter

What will we do if we want 100ms delay?

- Reducing the clock frequency?
- Are there other solutions?

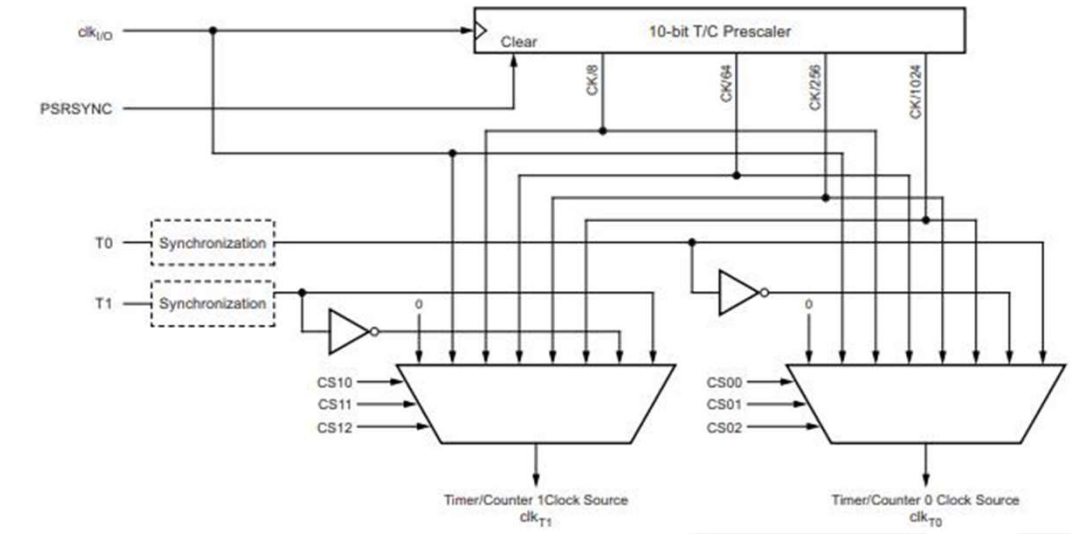
It came out requiring "Prescaler" to handle this role

Timer/Counter

Prescaler

Prescaler is the value is set by manipulate the bits in order to change the overflow timing

Figure 16-2. Prescaler for Timer/Counter0 and Timer/Counter1[1]



Timer/Counter

Prescaler

Timer0

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Timer1

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Timer/Counter

Prescaler

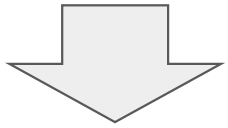
$$\text{interrupt frequency (Hz)} = \frac{\text{(Arduino clock speed 16,000,000Hz)}}{\text{(prescaler * (compare match register + 1))}}$$

Timer/Counter

Prescaler

if you wanted an interrupt every second (frequency of 1Hz), what is the value that should be set in match register?

$$\text{interrupt frequency (Hz)} = \frac{\text{(Arduino clock speed 16,000,000Hz)}}{\text{(prescaler * (compare match register + 1))}}$$



compare match register = [16,000,000Hz/ (prescaler * desired interrupt frequency)] - 1

compare match register = [16,000,000 / (prescaler * 1)] -1 => Which Timer will be used?

Timer/Counter

Prescaler

if you wanted an interrupt every second (frequency of 1Hz), what is the value that should be set in match register?

compare match register = [16,000,000 / (prescaler * 1)] -1

= [16,000,000 / (1024 * 1)] -1

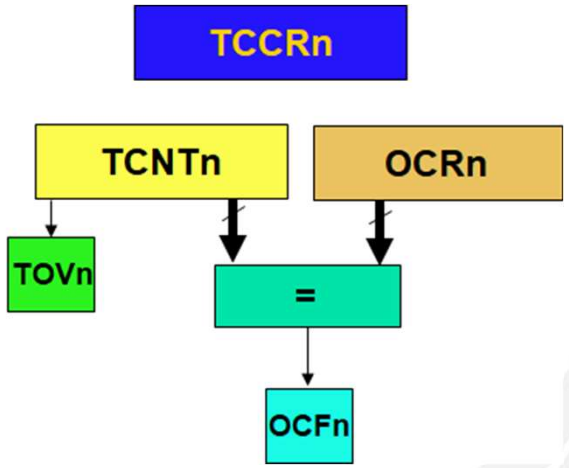
= 15,624

since 256 < 15,624 < 65,536, you must use Timer1 for this interrupt.

Timer/Counter

Timer/Counter Register

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

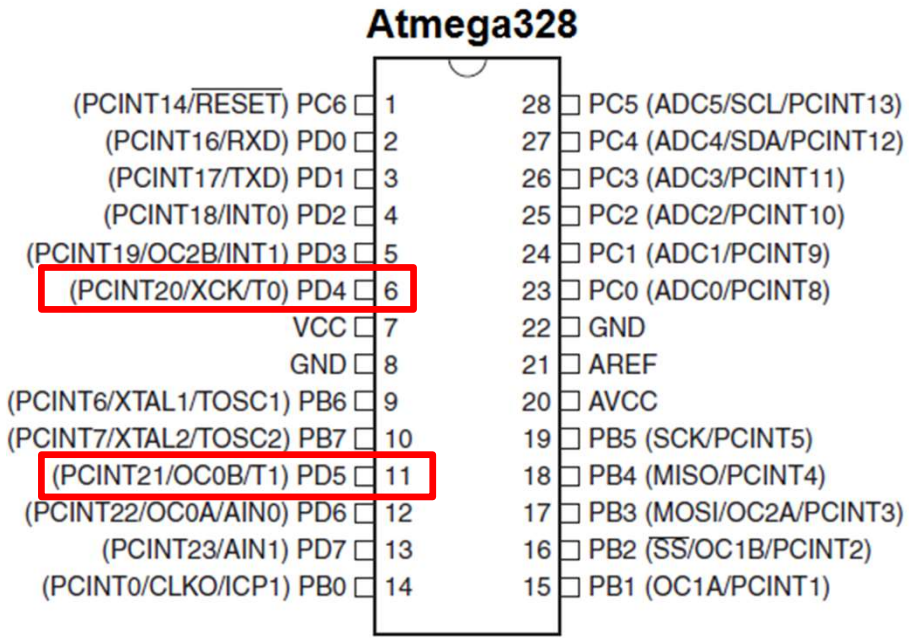


Note:

All of the timer registers are byte-addressable registers

Timer/Counter

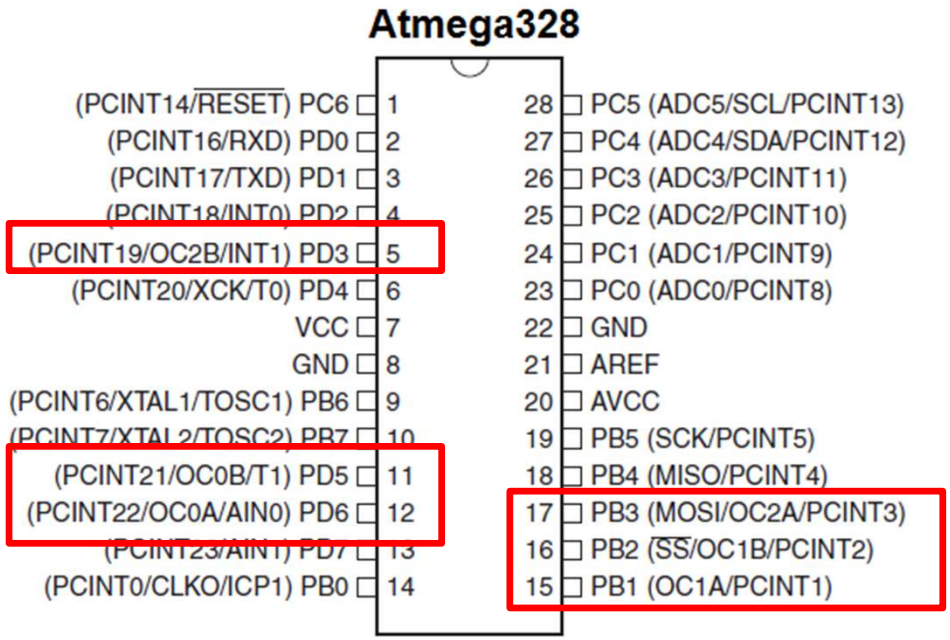
Timer/Counter External counter Input (Time/Counter 0,1)



PD4/ T0 Timer/Counter 0 external counter input
PD5/ T1 Timer/Counter 1 external counter input

Timer/Counter

Timer/Counter External compare matching Output



PD6/OC0A Timer/Counter 0 output compare match A output

PD5/OC0B Timer/Counter 0 output compare match B output

PB1/OC1A Timer/Counter 1 output compare match A output

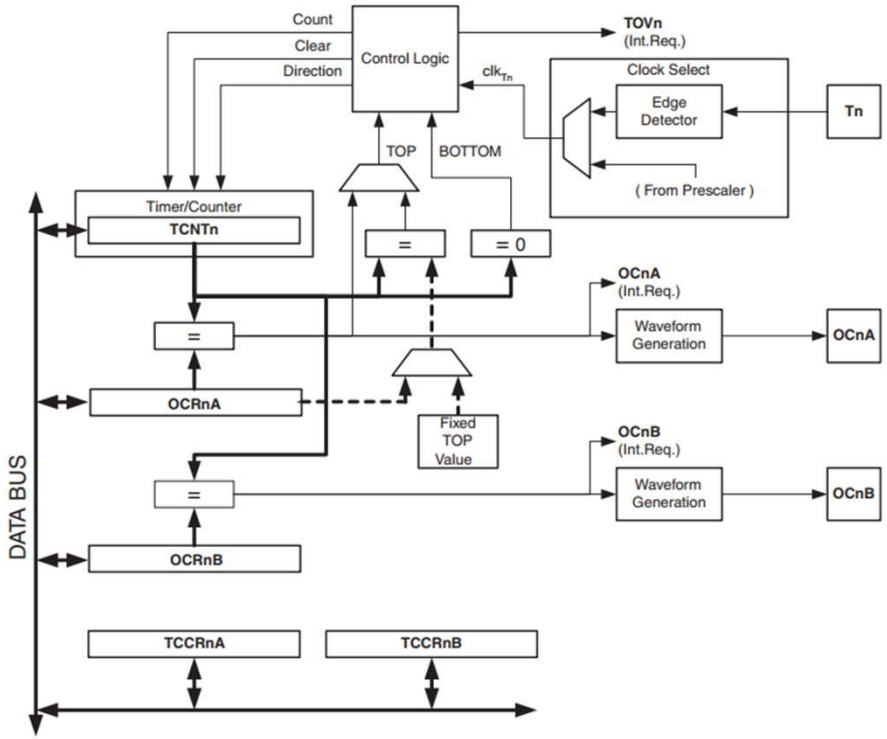
PB2/OC1B Timer/Counter 1 output compare match B output

PB3/OC2A Timer/Counter 2 output compare match A output

PD3/OC2B Timer/Counter 2 output compare match B output

Timer/Counter

Timer/Counter External compare matching Output (8-Bit)



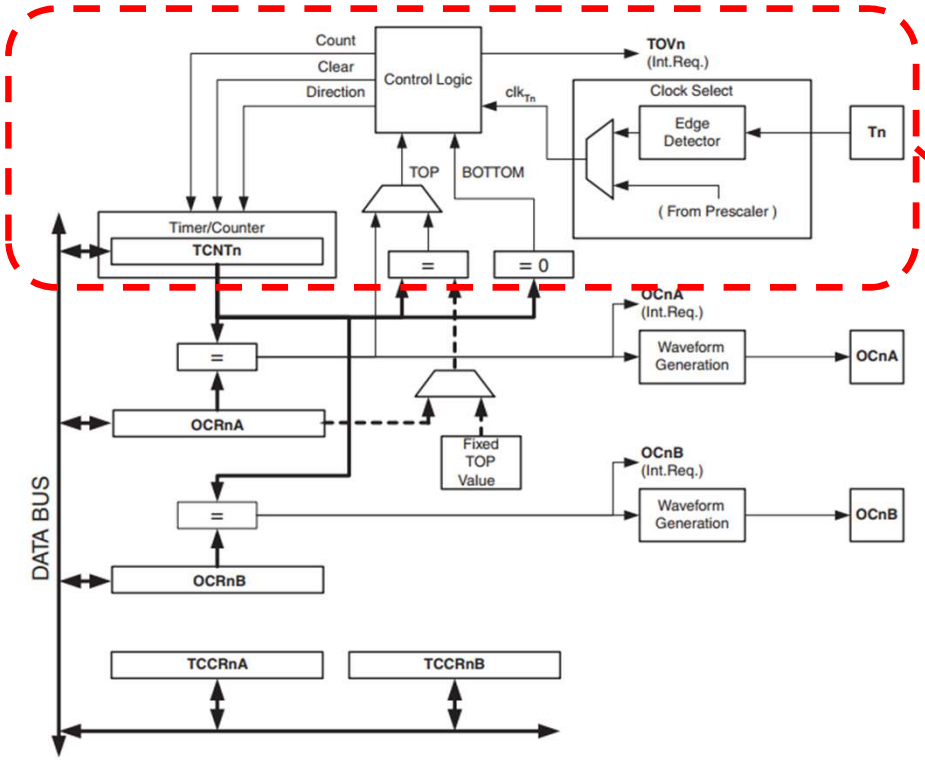
Timer/Counter 0,2 Input

Timer/Counter 0,2 Output Compare with A

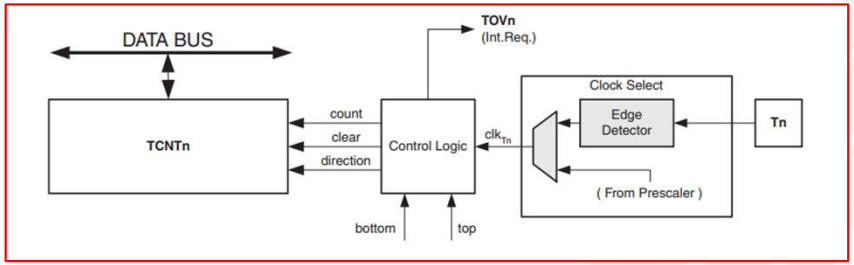
Timer/Counter 0,2 Output Compare with B

Timer/Counter

Timer/Counter External compare matching Output (8-Bit)

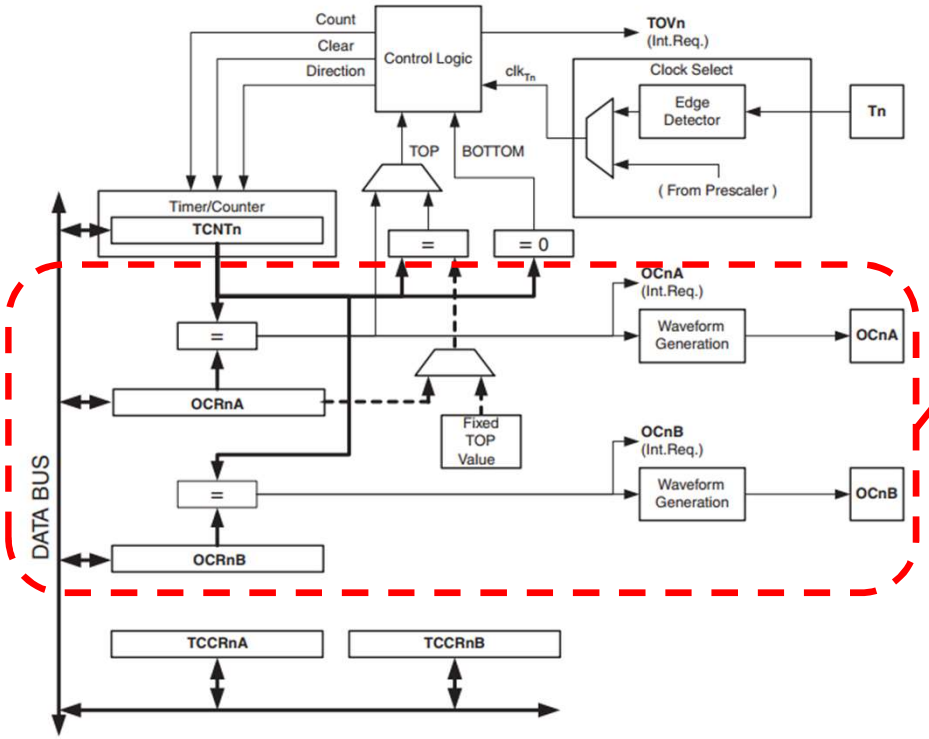


Counter Module

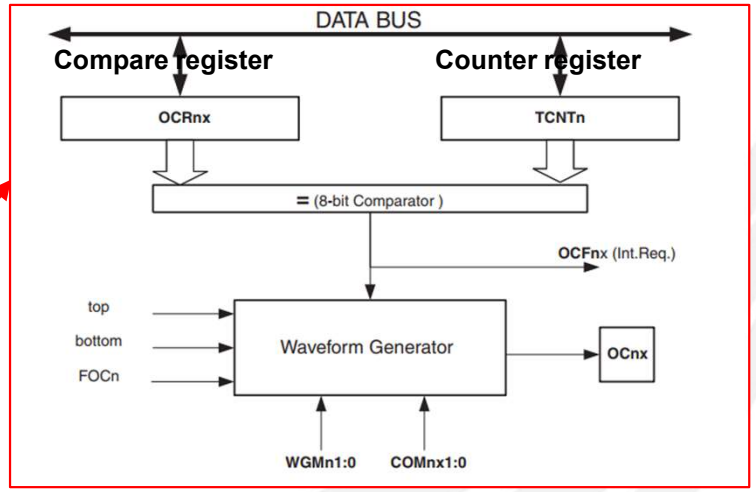


Timer/Counter

Timer/Counter External compare matching Output (8-Bit)

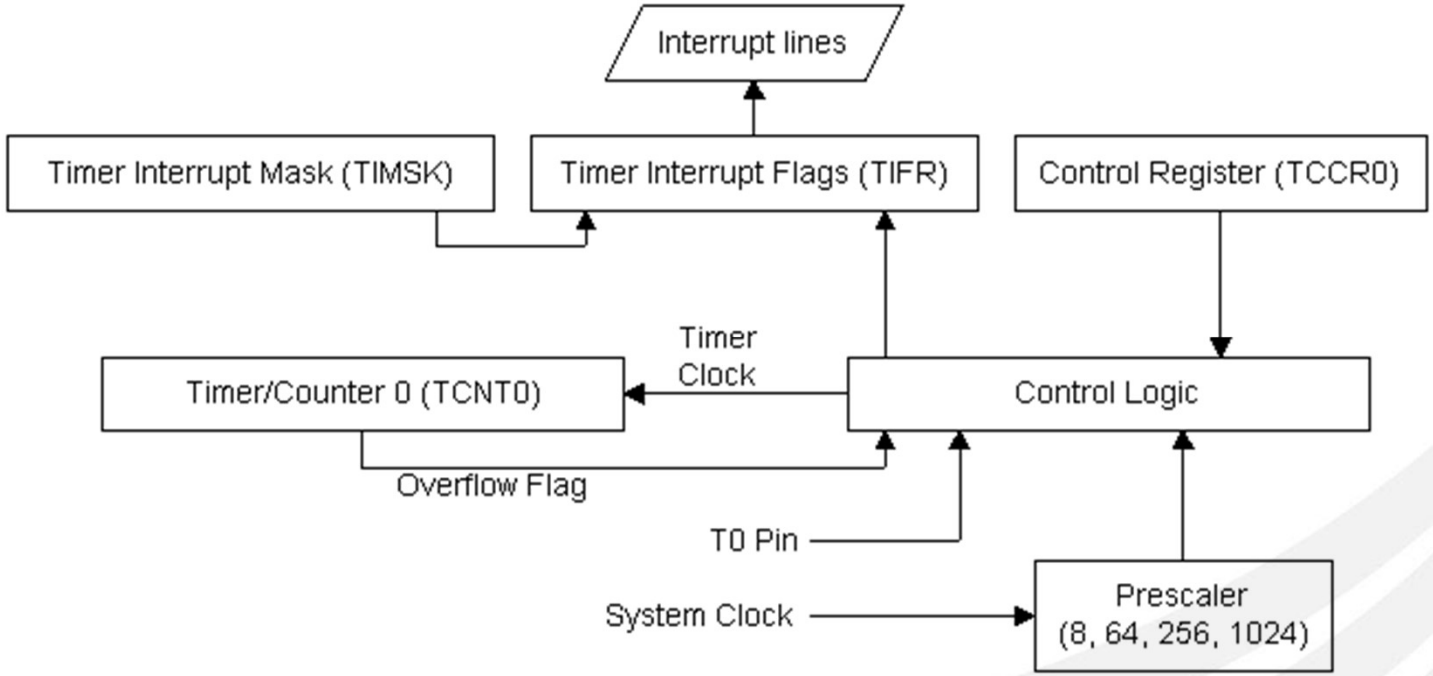


Compare Module



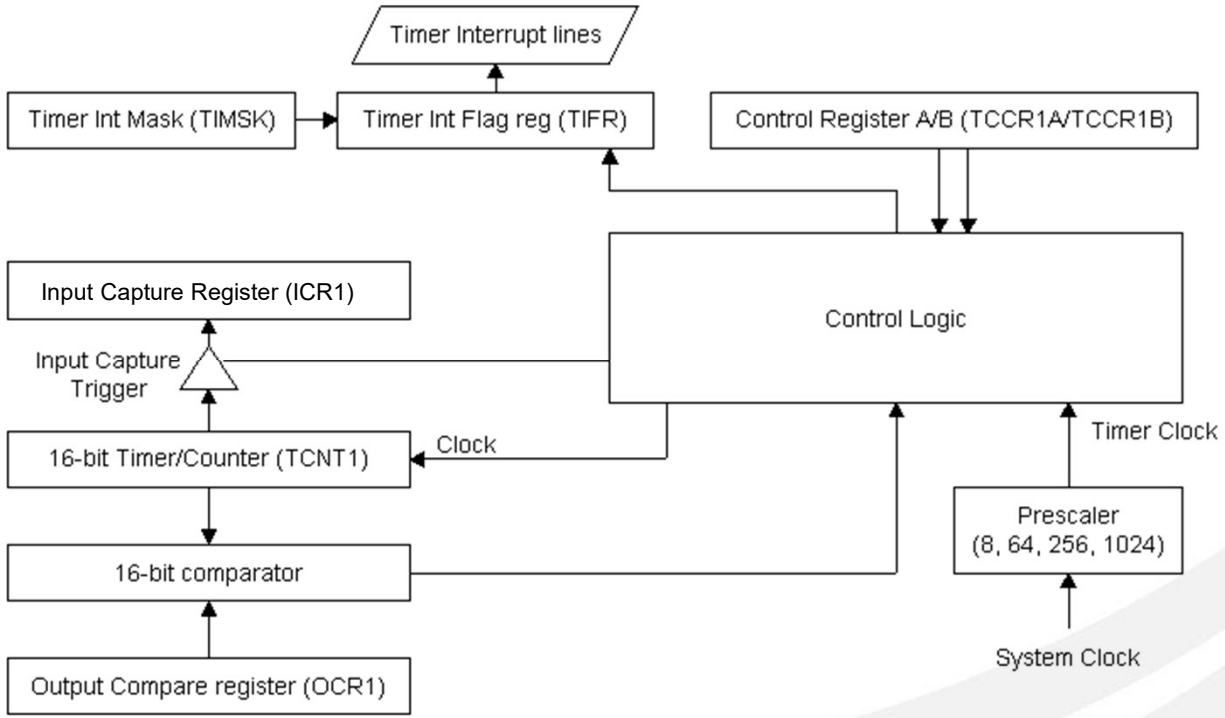
Timer/Counter

The 8-Bit Timer Control Register



Timer/Counter

The 16-Bit Timer Control Register



Timer/Counter

Timer Control Register

Counter0	Counter2	Description
TCCR0A	TCCR2A	Timer/Counter Control Register A
TCCR0B	TCCR2B	Timer/Counter Control Register B
TCNT0	TCNT2	Timer/Counter Register
OCR0A	OCR2A	Output Compare Register A
OCR0B	OCR2B	Output Compare Register B
TIMSK0	TIMSK2	Timer/Counter Interrupt Mask Register
TIFR0	TIFR2	Timer/Counter Interrupt Flag Register

Counter1	Description
TCCR1A	Timer/Counter 1 Control Register A
TCCR1B	Timer/Counter 1 Control Register B
TCCR1C	Timer/Counter 1 Control Register C
TCNT1H	Timer/Counter 1 High Register
TCNT1L	Timer/Counter 1 Low Register
OCR1AH	Output Compare Register 1 A High
OCR1AL	Output Compare Register 1 A Low
OCR1BH	Output Compare Register 1 B High
OCR1BL	Output Compare Register 1 B Low
ICR1H	Input Capture Register 1 High
ICR1L	Input Capture Register 1 Low
TIMSK1	Timer/Counter Interrupt Mask Register
TIFR1	Timer/Counter Interrupt Flag Register

Timer/Counter

Modes of Operation

- Normal
- CTC (Clear Timer on Compare Match)
- Fast PWM (Single Slope PWM)
- Phase Correct PWM (Double Slope PWM)

Mode of Operation is selected by WGM***xy*** registers, where ***x*** is Timer ID, and ***y*** is the WGM register ID

Modes of Operation (Timer0,2)

WGMX2	WGM0X	WGMX0	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM

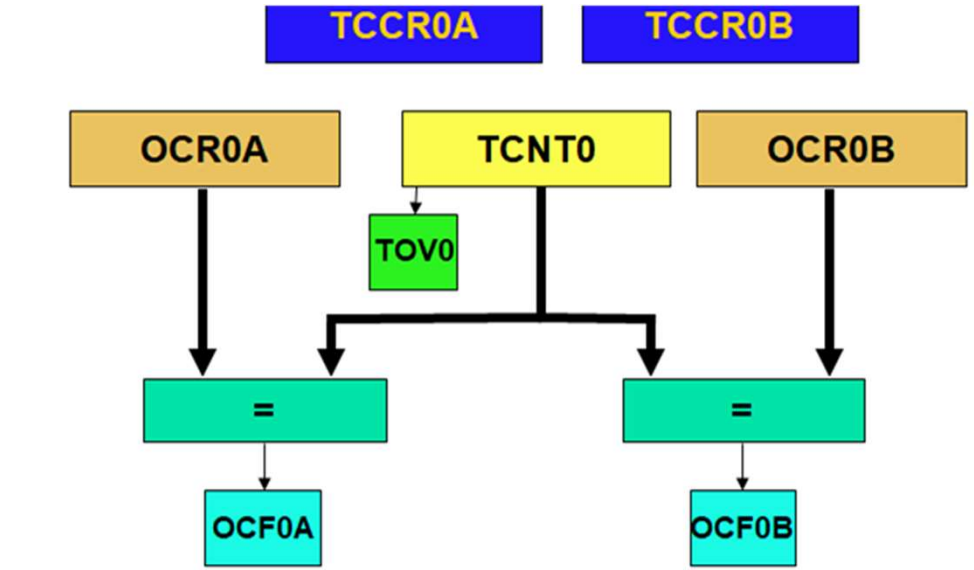
Timer/Counter

Modes of Operation (Timer1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

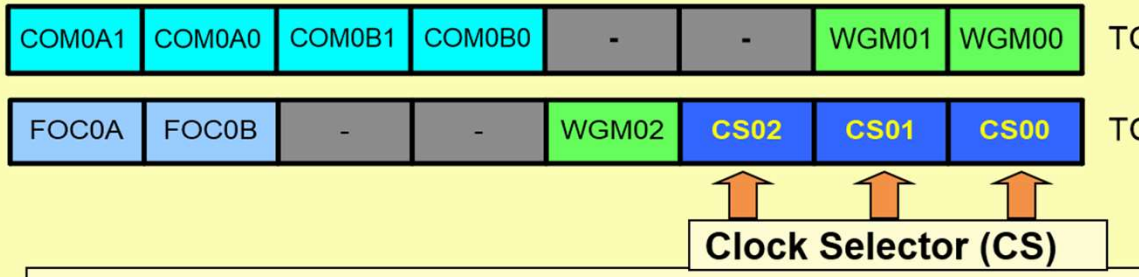
Timer0

Timer0 Registers



Timer0

Timer0 Registers



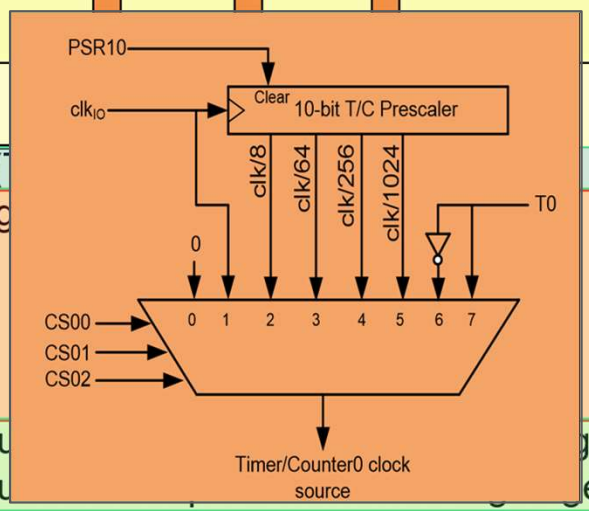
CS02	CS01	CS00	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Timer0

Timer0 Registers

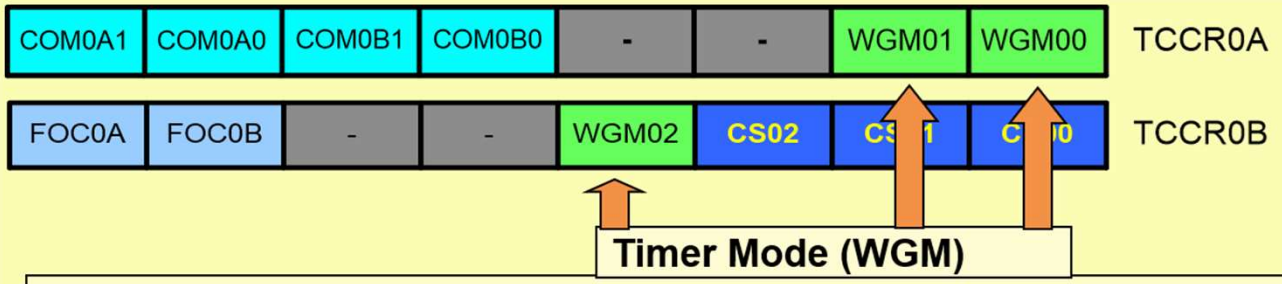
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	TCCR0A
FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	TCCR0B

CS02	CS01	CS00	Comment
0	0	0	No clock source (T
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	External clock sou
1	1	1	External clock sou



Timer0

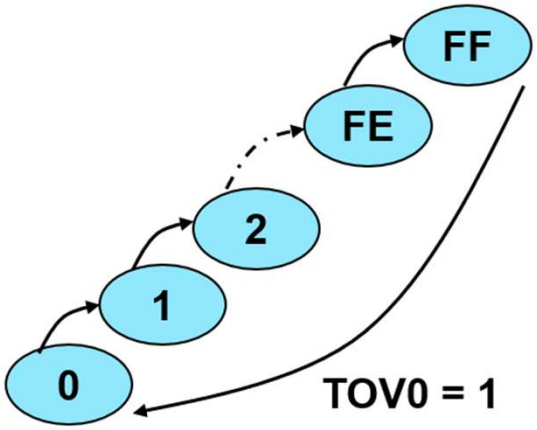
Timer0 Registers

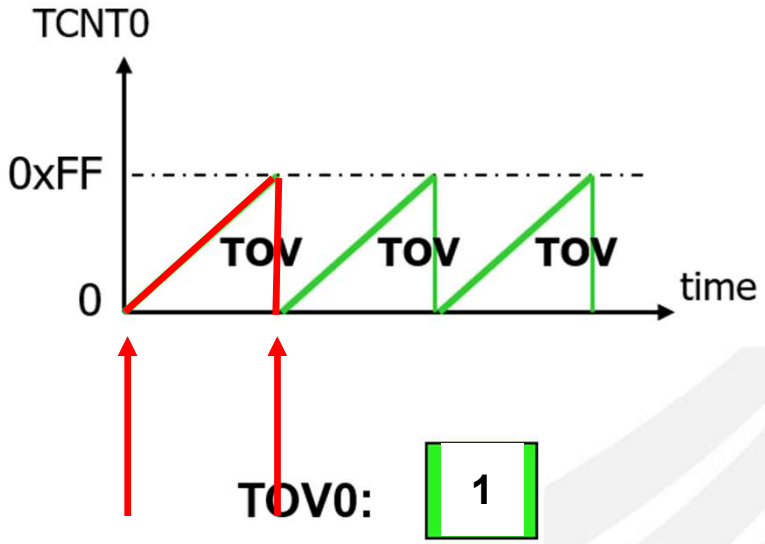


WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC (Clear Timer on Compare Match)
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM

Timer0

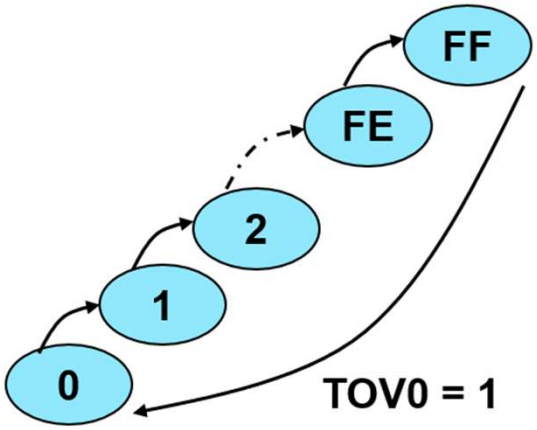
Timer0 Normal Mode

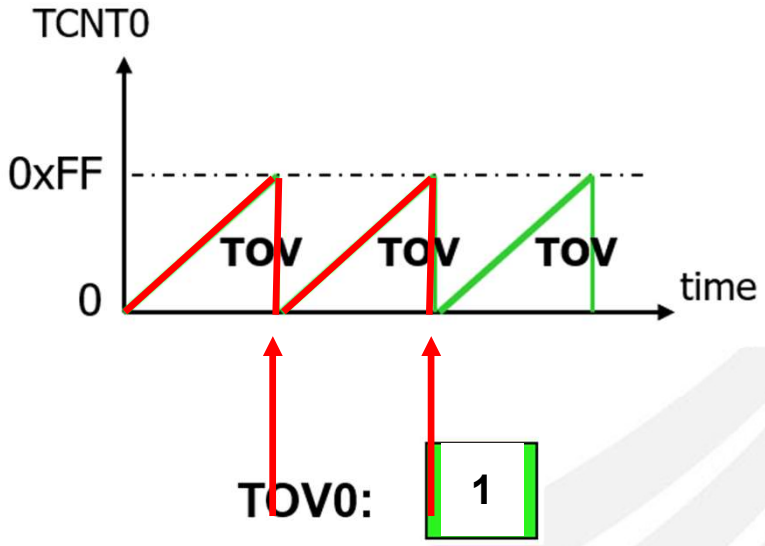




Timer0

Timer0 Normal Mode

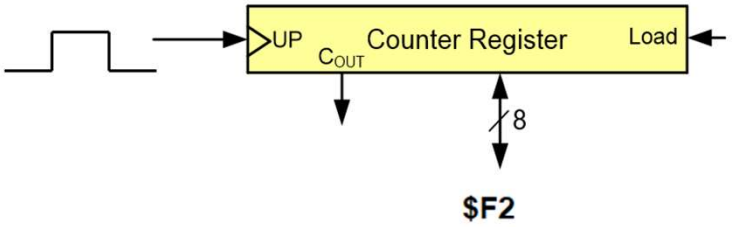




Timer0

Example 1: Write a program that waits 14 machine cycles in Normal mode

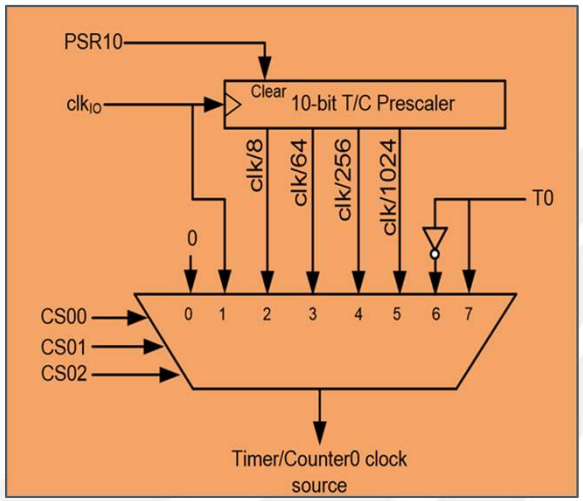
```
14 = $0E
  $100
  -$0E
  ----
   $F2
```



COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	TCCR0A

FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	TCCR0B
					0	0	1	

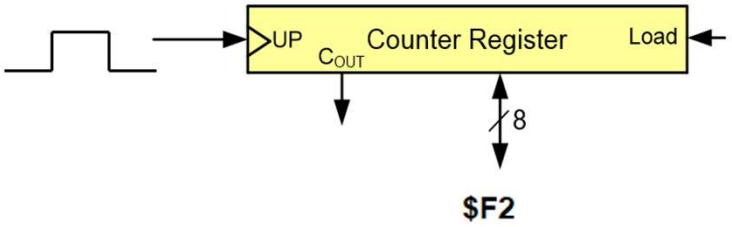
CS02	CS01	CS00	Comment
0	0	0	No clock source (Tim
0	0	1	clk (No Prescaling)

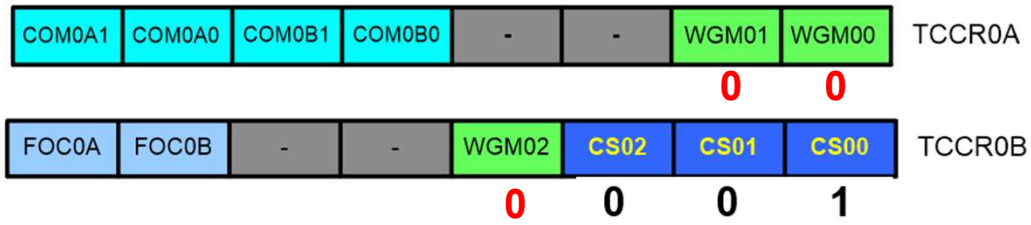


Timer0

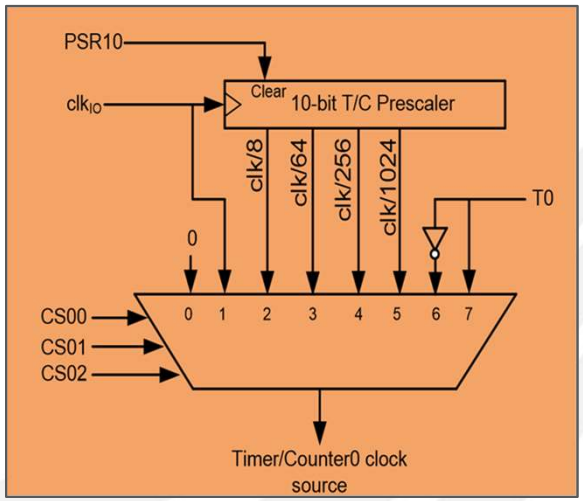
Example 1: Write a program that waits 14 machine cycles in Normal mode

```
14 = $0E
  $100
  -$0E
  ----
   $F2
```





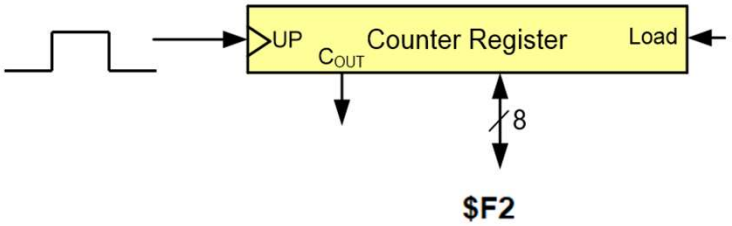
WGM02	WGM01	WGM00	Comment
0	0	0	Normal

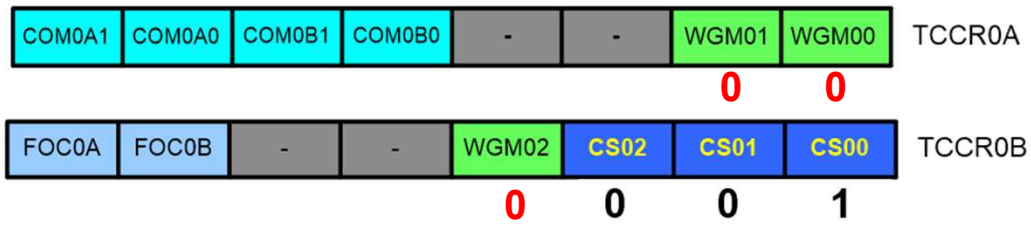


Timer0

Example 1: Write a program that waits 14 machine cycles in Normal mode

```
14 = $0E
  $100
  -$0E
  ----
   $F2
```






```
TCCR0A = 0x00;
TCCR0B = 0x03;
```

Timer0

Example 1: Write a program that waits 14 machine cycles in Normal mode


```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5   ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20          ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20         ;stop Timer0
        LDI     R20,(1<<TOV0)      ;R20 = 0x01
        OUT     TIFR0,R20          ;clear TOV0 flag

        EOR     R17,R16            ;toggle D5 of R17
        OUT     PORTB,R17          ;toggle PB5
        RJMP    BEGIN
```

```
DDRB = 1<<5;
PORTB &= ~(1<<5);  //PB5=0
while (1)
{
  TCNT0 = 0xF2;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&(1<<TOV0))==0)
  {}
  TCCR0B = 0;
  TIFR0 = (1<<TOV0);
  PORTB ^= (1<<5);
}
```

Timer0

Example 1: Write a program that waits 14 machine cycles in Normal mode

Calculate the delay, suppose XTAL = 10 MHz.

```
14 = $0E
  $100
  -$0E
  ----
   $F2
```

Solution 1 (inaccurate):

1) **Calculating T:**

T = 1/f = 1/10M = 0.1µs

2) **Calculating num of machine cycles:**

```
$100
-$F2
 $0E = 14
```

3) **Calculating delay**

14 * 0.1µs = 1.4 0µs

```
        LDI     R16,0x20
        SBI     DDRB,5   ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20         ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20        ;stop Timer0
        LDI     R20,(1<<TOV0)     ;R20 = 0x01
        OUT     TIFR0,R20         ;clear TOV0 flag

        EOR     R17,R16           ;toggle D5 of R17
        OUT     PORTB,R17         ;toggle PB5
        RJMP    BEGIN
```

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

Timer0

Example 1: Write a program that waits 14 machine cycles in Normal mode

Accurate calculating****

Other than timer, executing the instructions consumes time; so if we want to calculate the accurate delay a program causes **we should add the delay caused by instructions to the delay** caused by the timer

Timer0

Example 1: Write a program that waits 14 machine cycles in Normal mode

```
        LDI     R16,0x20
        SBI     DDRB,5
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2        1
        OUT     TCNT0,R20       1
        LDI     R20,0x00        1
        OUT     TCCR0A,R20      1
        LDI     R20,0x01        1
        OUT     TCCR0B,R20      1
AGAIN:  SBIS    TIFR0,TOV0      1 / 2
        RJMP    AGAIN           2
        LDI     R20,0x0         1
        OUT     TCCR0B,R20      1
        LDI     R20,0x01        1
        OUT     TIFR0,R20       1
        EOR     R17,R16         1
        OUT     PORTB,R17       1
        RJMP    BEGIN           2
                                18
```

Delay caused by timer = 14 * 0.1µs = 1.4 µs Delay caused by instructions = 18 * 0.1µs = 1.8
Total delay = 3.2 µs ➔ wave period = 2*3.2 µs = 6.4 µs ➔ wave frequency = 156.25 KHz

Finding values to be loaded into the timer

1. Calculate the period of clock source.
   a. Period = 1 / Frequency
      E.g. For XTAL = 16 MHz T = 1/16MHz
2. Divide the desired time delay by period of clock.
3. Perform 256 - n, where n is the decimal value we got in Step 2.
4. Set TCNT0 = 256 - n

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

Timer0

Example 2: Assuming that XTAL = 10 MHz, write a program to generate a square wave with a period of 10 us on pin PORTB.3.

For a square wave with T = 10 μs we must have a time delay of 5 μs. Because XTAL = 10 MHz, the counter counts up every 0.1 μs. This means that we need 5 μs / 0.1 μs = 50 clocks. 256 - 50 = 206.

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5   ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,206
        OUT     TCNT0,R20         ;load timer0

        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, int. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20        ;stop Timer0
        LDI     R20,(1<<TOV0)     ;R20 = 0x01
        OUT     TIFR0,R20         ;clear TOV0 flag

        EOR     R17,R16    ;toggle D5 of R17
        OUT     PORTB,R17  ;toggle PB5
        RJMP    BEGIN
```

```
DDRB = 1<<3;
PORTB &= ~ (1<<3);
while (1)
{
  TCNT0 = 206;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&0x01) == 0)
  {}
  TCCR0B = 0;
  TIFR0 = 1<<TOV0;
  PORTB = PORTB ^ (1<<3);
}
```

Timer0

Example 3: Modify TCNT0 in Example 2 to get the largest time delay possible with no prescaler. Find the delay in µs. In your calculation, do not include the overhead due to instructions.

To get the largest delay we make TCNT0 zero. This will count up from 00 to 0xFF and then roll over to zero

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5   ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0
        OUT     TCNT0,R20         ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, int. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20        ;stop Timer0
        LDI     R20,(1<<TOV0)     ;R20 = 0x01
        OUT     TIFR0,R20         ;clear TOV0 flag

        EOR     R17,R16    ;toggle D5 of R17
        OUT     PORTB,R17  ;toggle PB5
        RJMP    BEGIN
```

```
DDRB = 1 << 3;
PORTB &= ~(1<<3);
while (1)
{
  TCNT0 = 0x00;
  TCCR0A = 0x00;
  TCCR0B = 0x01;

while((TIFR0&(1<<TOV0))==0)
{}
  TCCR0B = 0;
  TIFR0 = 0x01;
  PORTB = PORTB^(1<<3);
}
```

Solution

1) Calculating T:

T = 1/f = 1/10MHz = 0.1µs

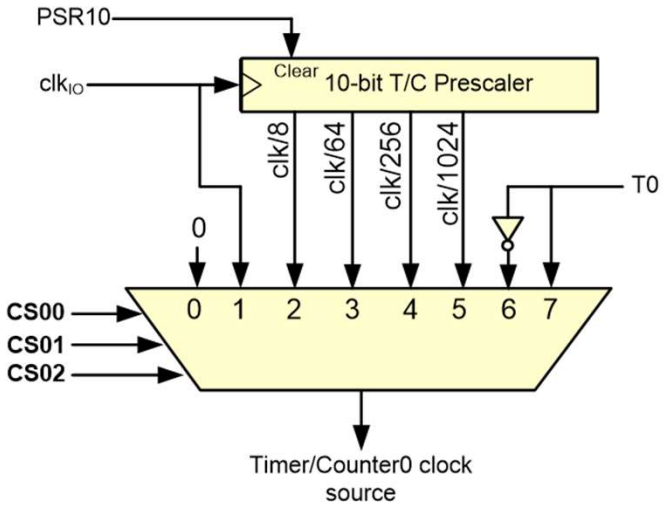
2) Calculating delay

256 * 0.1µs = 25.6µs

Timer0

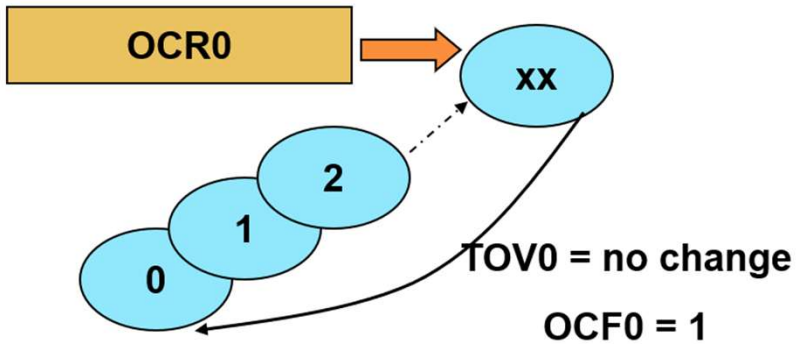
Generating Larger Delays

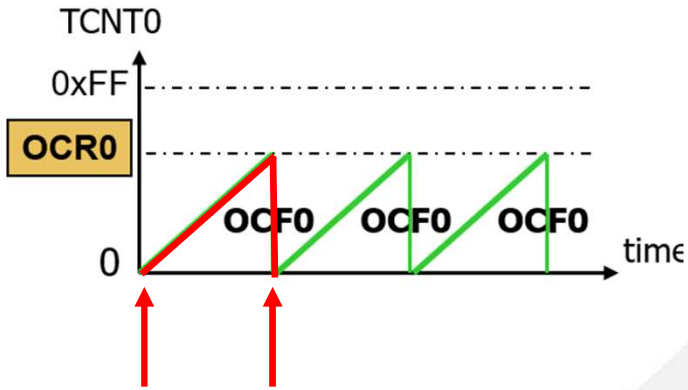
- Using loop
- Prescaler
- Get the Bigger counters



Timer0

CTC (Clear Timer on Compare match) mode





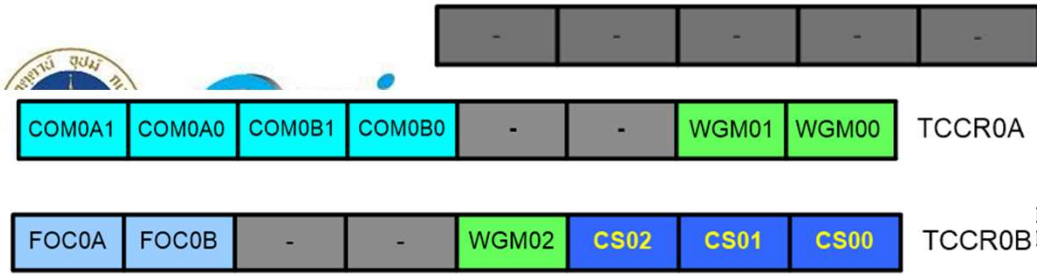
TOV0: 0

OCF0: 1

Timer0

CTC (Clear Timer on Compare match) mode

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



Timer0

CS02	CS01	CS00	Comment
0	0	0	No clock source (Tim
0	0	1	clk (No Prescaling)

Rewrite Example2 using CTC

For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. **Therefore, we have OCR0A= 49.**

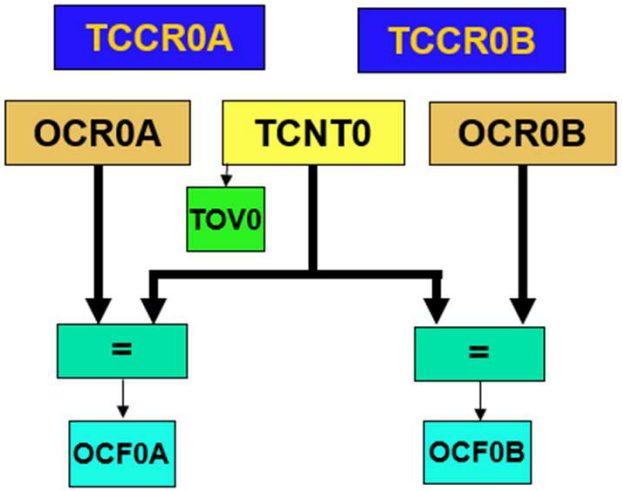
```
        LDI     R16,(1<<3)          ;R16 = 0x08
        SBI     DDRB,3              ;PB3 as an output
        LDI     R17,0
        OUT     PORTB,R17
        LDI     R20,49
        OUT     OCR0A,R20           ;load OCR0A
BEGIN:  LDI     R20,(1<<WGM01)      ;TCCR0A=0x02
        OUT     TCCR0A,R20  ;CTC mode, int. clk
        LDI     R20,1               ; TCCR0B = 1
        OUT     TCCR0B,R20          ; N = 1
AGAIN:  SBIS    TIFR0,OCF0A ;if OCF0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,1<<OCF0A
        OUT     TIFR0,R20           ;clear OCF0A flag
        EOR     R17,R16             ;toggle D3 of R17
        OUT     PORTB,R17           ;toggle PB3
        RJMP    BEGIN
```

```
DDRB |= 1<<3;
while (1)
{
  OCR0A = 49;
  TCCR0A = (1<<WGM01); //CTC
  TCCR0B = 1;  //N = 1
 while((TIFR0&(1<<OCF0A))==0)
  { }
  TCCR0B = 0; //stop timer0
  TIFR0 = 1<<OCF0A;
  PORTB ^= 1<<3; //toggle PB3
}
```

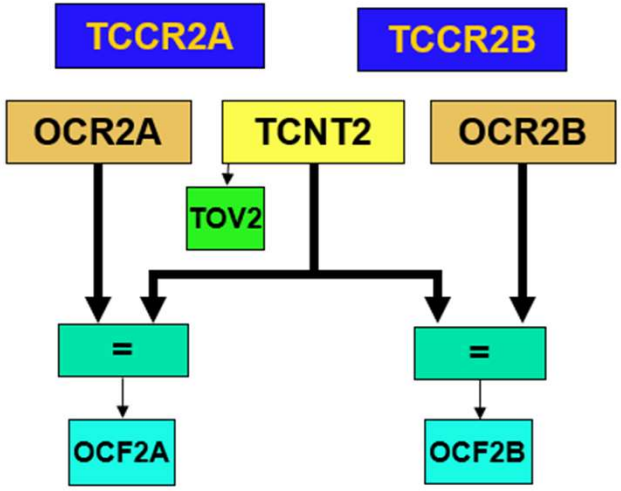
COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

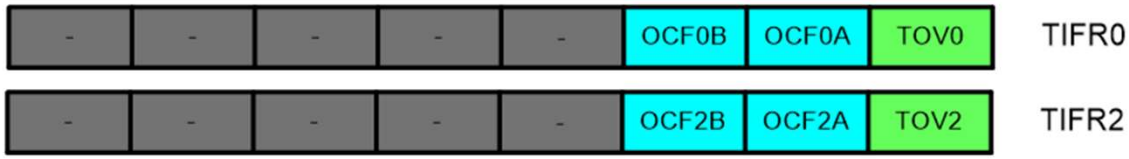
Timer2 vs. Timer0

- Timer0



- Timer2





COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

Timer2 vs. Timer0

The difference between Timer0 and Timer2

Timer0

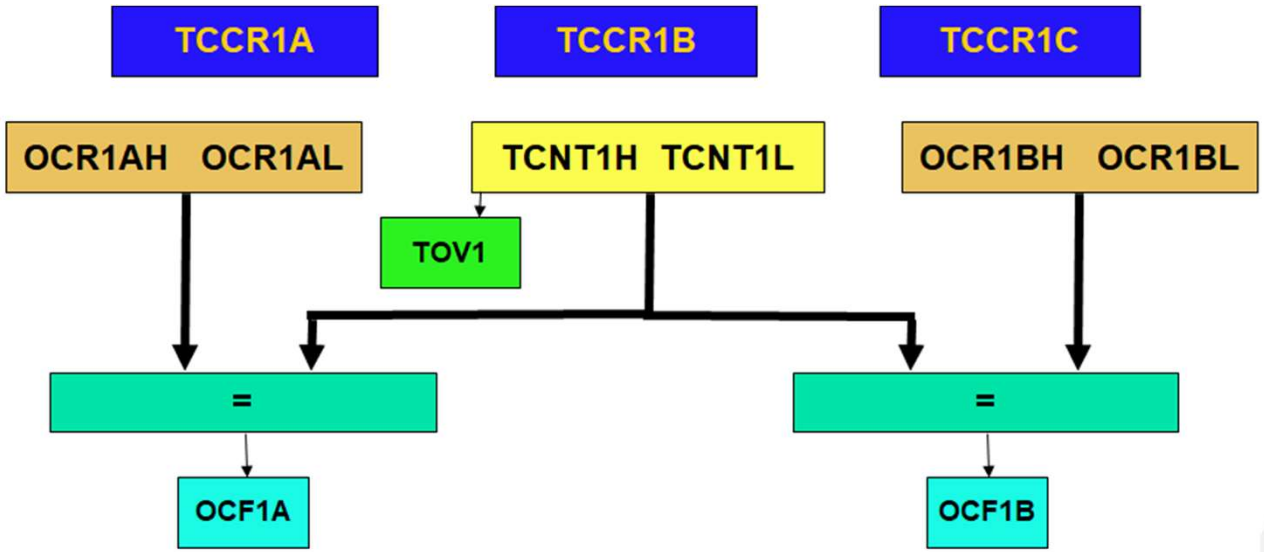
CS02	CS01	CS00	Comment
0	0	0	Timer/Counter stopped
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock (falling edge)
1	1	1	External clock (rising edge)

Timer2

CS22	CS21	CS20	Comment
0	0	0	Timer/Counter stopped
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 32
1	0	0	clk / 64
1	0	1	clk / 128
1	1	0	clk / 256
1	1	1	clk / 1024

Timer1

Timer1 Registers



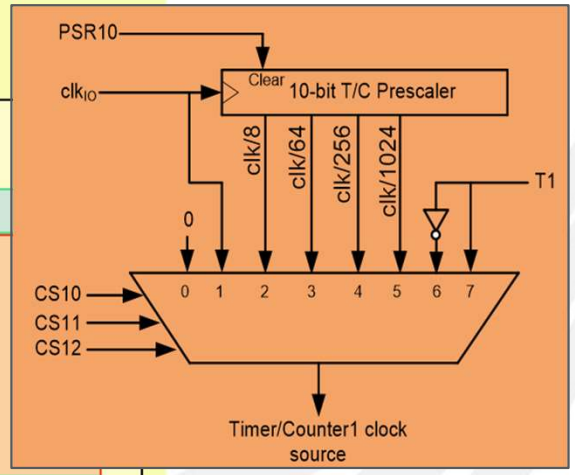
Timer1

Timer1 Registers

TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10
TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

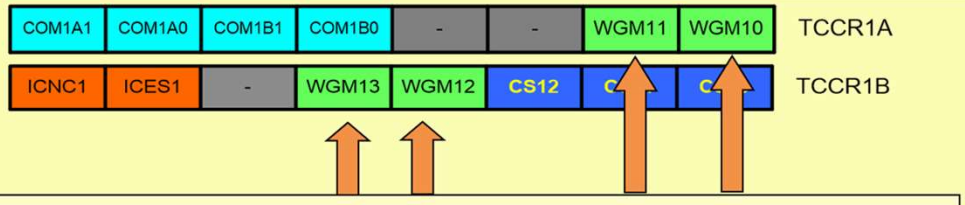
Clock Selector (CS)

CS12	CS11	CS10	Comment
0	0	0	No clock source (Timer/Counter stopped)
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	External clock source on T1 pin. Clock on falling edge
1	1	1	External clock source on T1 pin. Clock on rising edge



Timer1

Timer1 Registers



Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Timer1

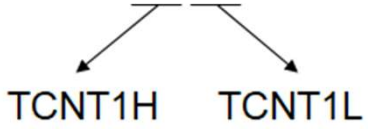
Assuming XTAL = 10 MHz write a program that toggles PB5 once per millisecond, using Normal mode.

XTAL = 10 MHz 1/10 MHz = 0.1 µs

Num. of machine cycles = 1 ms / 0.1 µs

TCNT1 = 65,536 – 10,000 = 55,536

= $D8F0




```
    LDI   R16,HIGH(RAMEND)    ;init stack pointer
    OUT   SPH,R16
    LDI   R16,LOW(RAMEND)
    OUT   SPL,R16
    SBI   DDRB,5              ;PB5 as an output
BEGIN:SBI PORTB,5             ;PB5 = 1
    RCALL DELAY_1ms
    CBI   PORTB,5             ;PB5 = 0
    RCALL DELAY_1ms
    RJMP  BEGIN

DELAY_1ms:
    LDI   R20,HIGH(-10000)
    STS   TCNT1H,R20
    LDI   R20, ,LOW(-10000)
    STS   TCNT1L,R20          ;Timer1 overflows after 10000 machine cycles
    LDI   R20,0x0
    STS   TCCR1A,R20          ;WGM11:10=00
    LDI   R20,0x1
    STS   TCCR1B,R20          ;WGM13:12=00,CS=CLK
AGAIN:SBIS TIFR1,TOV1         ;if TOV1 is set skip next instruction
    RJMP  AGAIN
    LDI   R20,1<<TOV1
    OUT   TIFR1,R20           ;clear TOV1 flag
    LDI   R19,0
    STS   TCCR1B,R19          ;stop timer
    STS   TCCR1A,R19          ;
    RET
```

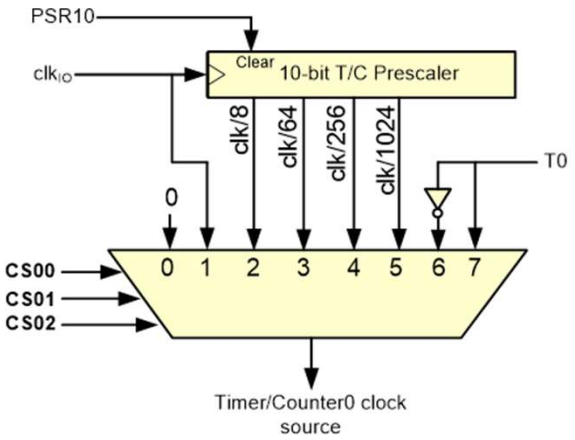
Timer1

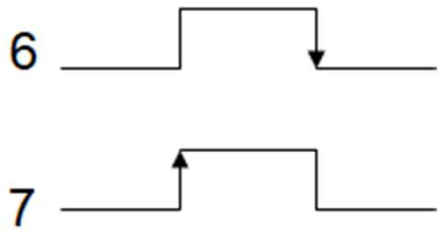
Assuming XTAL = 10 MHz, write a program that toggles PB5 once per millisecond, using CTC mode.

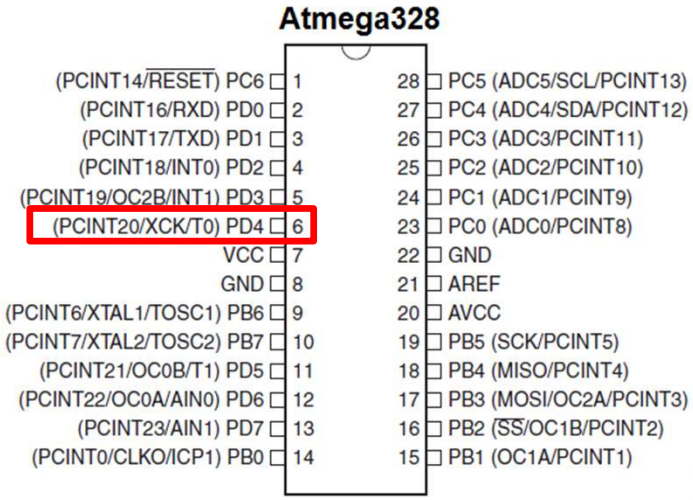
```
     LDI     R16,HIGH(RAMEND)
     OUT     SPH,R16
     LDI     R16,LOW(RAMEND)
     OUT     SPL,R16

     SBI     DDRB,5       ;PB5 as an output
BEGIN:SBI    PORTB,5      ;PB5 = 1
     RCALL   DELAY_1ms
     CBI     PORTB,5      ;PB5 = 0
     RCALL   DELAY_1ms
     RJMP    BEGIN

DELAY_1ms:
     LDI     R20,0x00
     STS     TCNT1H,R20   ;TEMP = 0
     STS     TCNT1L,R20   ;TCNT1L = 0, TCNT1H = TEMP

     LDI     R20,0x27
     STS     OCR1AH,R20   ;TEMP = 0x27
     LDI     R20,0x0F
     STS     OCR1AL,R20   ;OCR1AL = 0x0F, OCR1AH = TEMP

     LDI     R20,0x00
     STS     TCCR1A,R20   ;WGM11:10=00
     LDI     R20,0x09
     STS     TCCR1B,R20   ;WGM13:12=01,CS=CLK
AGAIN:
     SBIS    TIFR1,OCF1A ;if OCF1A is set skip next instruction
     RJMP    AGAIN
     LDI     R19,0
     STS     TCCR1B,R19   ;stop timer
     STS     TCCR1A,R19   ;
     LDI     R20,1<<OCF1A
     OUT     TIFR1,R20    ;clear OCF1A flag
     RET
```

```
#include <avr/io.h>
void  delay1ms();
int main(){
 DDRB |= 1<<5;
 while (1) {
 delay1ms();
 PORTB ^= (1<<5); //toggle PB5
 }
}
void  delay1ms()
{
 TCNT1 = 0;
 OCR1A = 10000-1;
 TCCR1A = 0;     //WGM=0100 (CTC)
 TCCR1B = 0x09;  //N = 1
 while((TIFR1&(1<<OCF1A))==0)
 { } //wait until OCF1A is set
 TCCR1B = 0; //stop timer1
 TIFR1 = 1<<OCF1A;//clear flag
}
```

Counter


PSR10
clk$_{IO}$
Clear
10-bit T/C Prescaler
clk/8
clk/64
clk/256
clk/1024
T0
0
0 1 2 3 4 5 6 7
CS00
CS01
CS02
Timer/Counter0 clock source


6
7

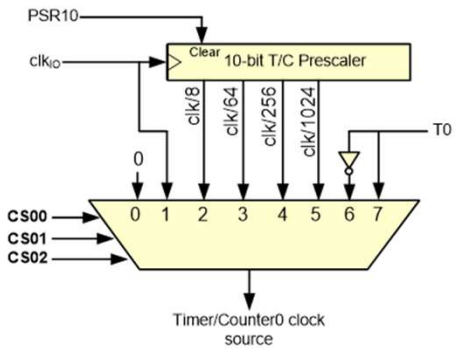

Atmega328
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

Counter

Assuming that clock pulses are fed into pin T0, write a program for counter 0 in normal mode to count the pulses on falling edge and display the state of the TCNT0 count on PORTC.

```
    CBI   DDRD,4            ;make T0 (PD4) input
    LDI   R20,0xFF
    OUT   DDRC,R20          ;make PORTC output
    LDI   R20,0x00
    OUT   TCCR0A,R20
    LDI   R20,0x06
    OUT   TCCR0B,R20        ;counter, falling edge
AGAIN:
    IN    R20,TCNT0
    OUT   PORTC,R20         ;PORTC = TCNT0
    RJMP  AGAIN             ;keep doing it
```

```
#include <avr/io.h>
int main()
{
 DDRC = 0xFF;
 TCCR0A = 0;     //WGM=0000 (Normal)
 TCCR0B = 0x06;  //CS=6 (Count on fall)
 while (1)
 {
   PORTC = TCNT0; //read TCNT0
 }
}
```



CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

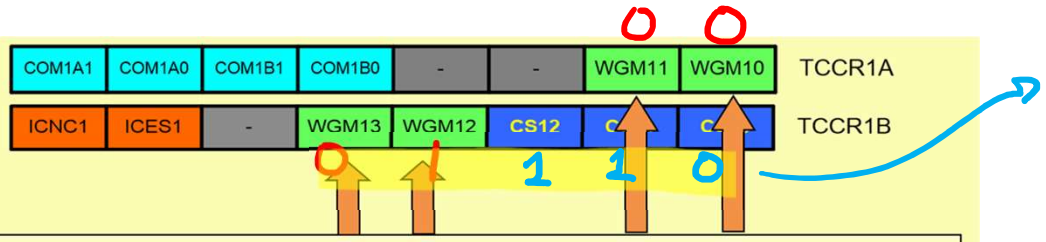
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	TCCR0A
FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	TCCR0B

Counter

Assuming that clock pulses are fed into pin T1. Write a program for Counter1 in CTC mode to make PORTC.0 high every 100 pulses.

```
1       CBI DDRD, 5
2       SBI DDRC, 0
3
4       LDI R20, 0x00
5       STS TCCR1A, R20
6       LDI R20, 0x0E
7       STS TCCR1B, R20
8
9   AGAIN:
10      LDI R20, 0
11      STS OCR1AH, R20
12      LDI R20, 99
13      STS OCR1AL, R20
14
15  L1:
16      SBIS TIFR1, OCF1A
17      RJMP L1
18      LDI R20, 1<<OCF1A
19      OUT TIFR1, R20
20
21      SBI PORTC, 0
22      CBI PORTC, 0
23      RJMP AGAIN
24
```

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

int main ( )
{
    DDRD &= ~(1<<5);
    DDRC |= 1<<0;
    TCCR1A = 0;
    TCCR1B = 0X0E;
    OCR1A = 99;
    while(1) {
        while((TIFR1 & (1<<OCF1A)) == 0)
        {}
        TIFR1 = (1<<OCF1A);
        PORTC |= (1<<0);
        PORTC &= ~(1<<0);
    }
    return 0;
}
```



0 1 1 1 0 = 0E

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

T1 (PD5)

Exercise 9

1. We would like to toggle pin PB5 every 960 microseconds. Assume that the clock speed is 16 MHz and the prescaler value is 64. Use Timer1.
    1.1 Calculate the number that Timer1 must count to get the desired delay.
    1.2 Write the code in C.

Exercise 9

2.1 Write C program that use external clock source T0 to toggle PC.0 every time the counter count to 10.

2.2 Wire T0 clock source to push button in PicSimLab and verify that PC.0 toggle when the button is pressed 10 times.