Timer/counter

The AVR microcontroller
and embedded
systems
using assembly and c

Guess What ?

Gift Quiz

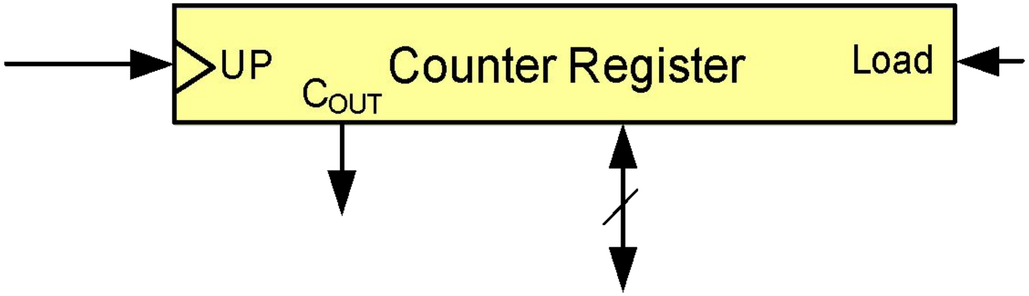
Can we get one Today ?

Why Not !

What if I am not lucky enough to get this gift?

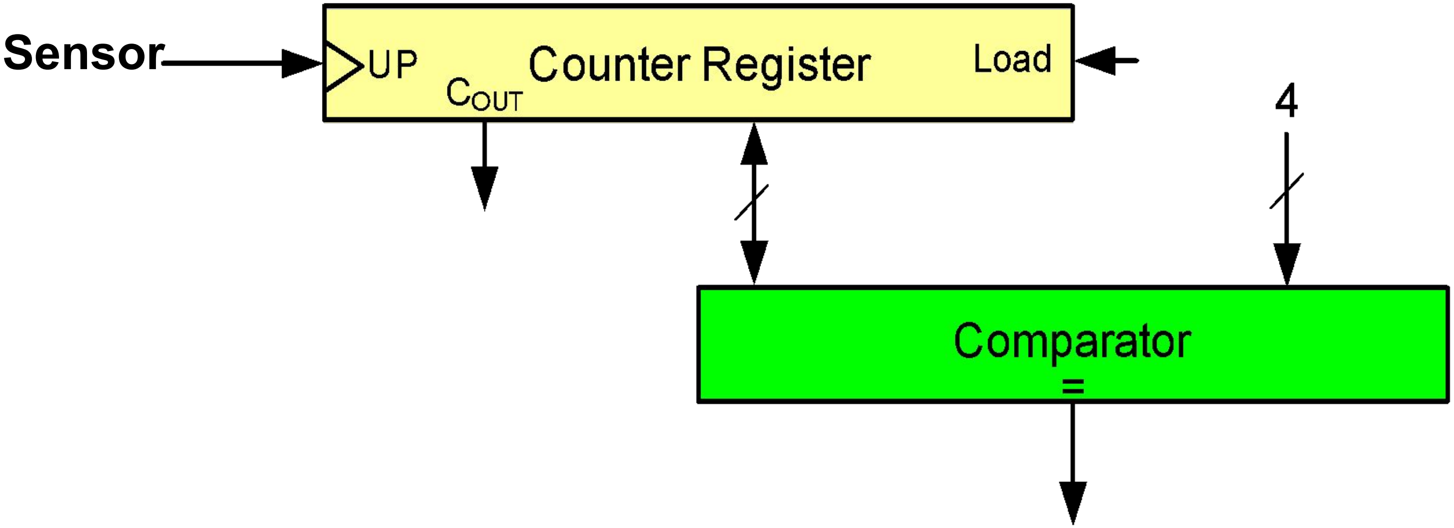
You will get 10/10 for that QUIZ ☺

A counter register



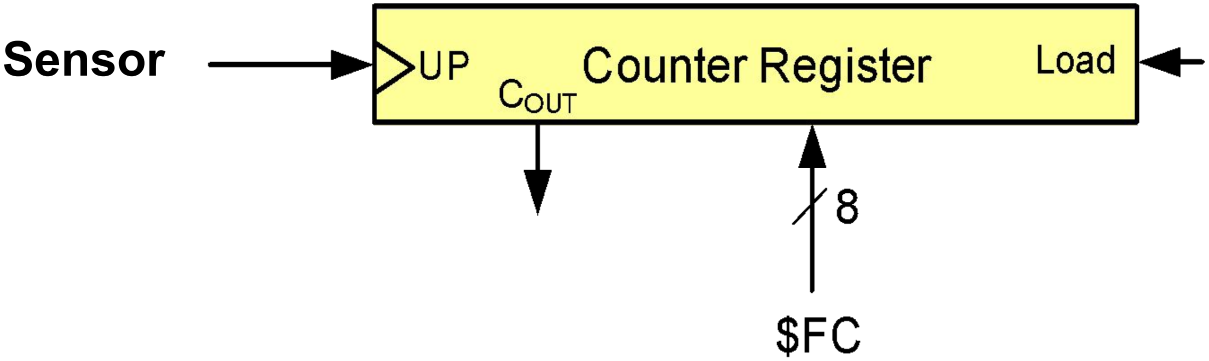
A simple design (counting people)

First design

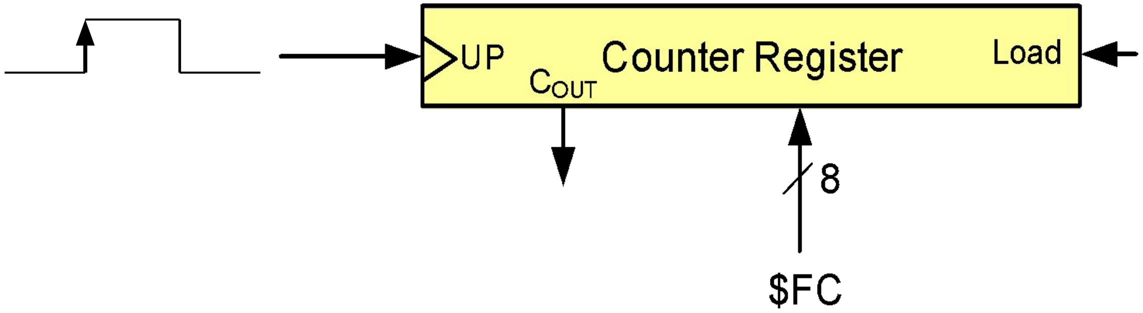


A simple design (counting people)

Second design

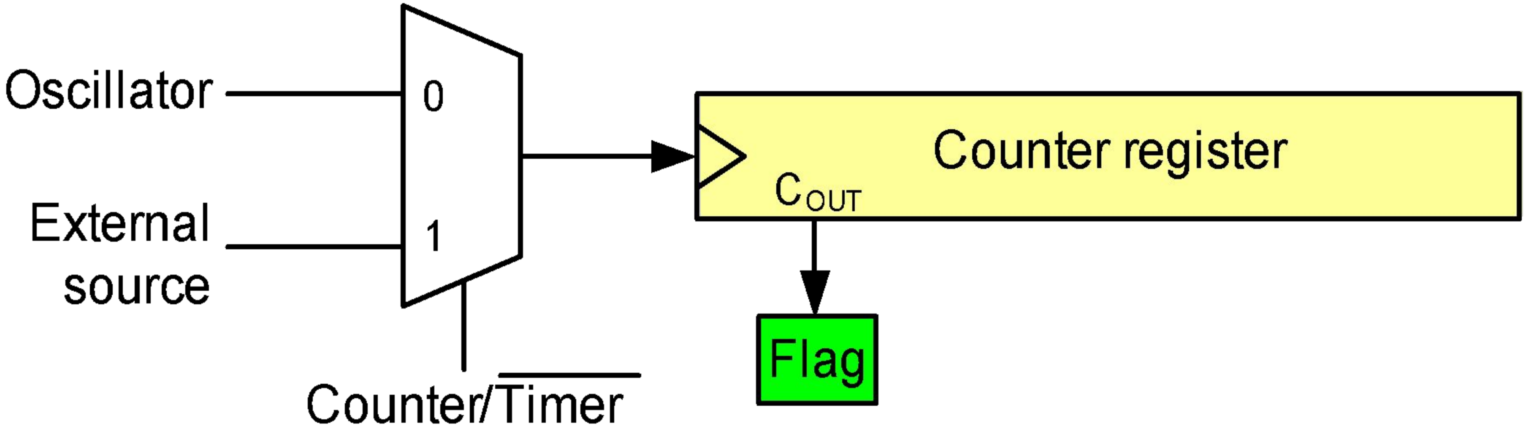


A simple design (making delay)



A generic timer/counter

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

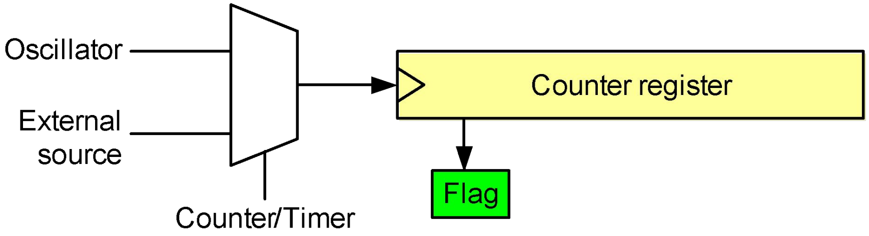


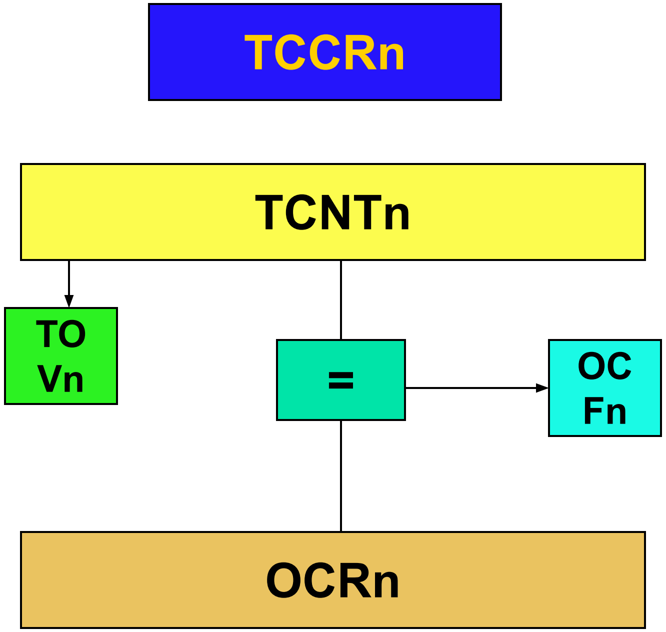
Timers in AVR

- 1 to 6 timers
  - 3 timers in ATmega32
- 8-bit and 16-bit timers
  - two 8-bit timers and one 16-bit timer in ATmega32

Timer in AVR

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





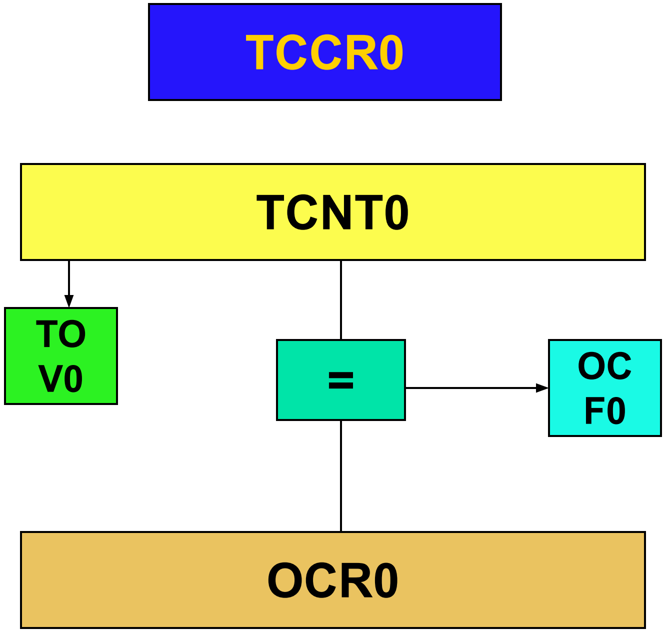
Comment:

All of the timer registers are byte-addressable I/O registers

Timer 0 (an 8-bit timer)

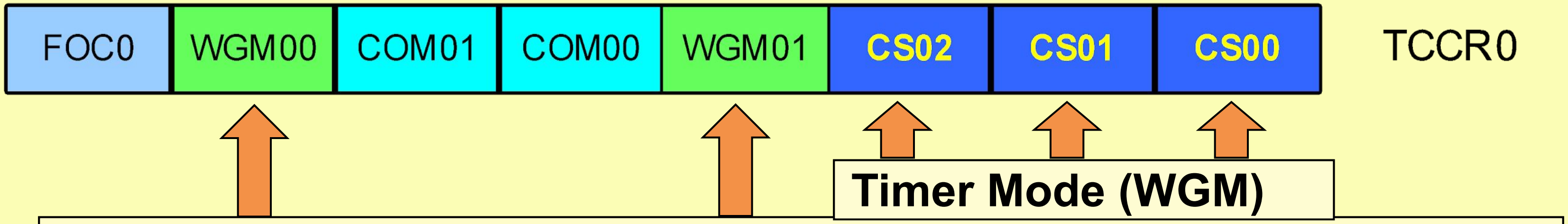
The AVR microcontroller
and embedded
systems
using assembly and c

Timer 0



OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

TIFR

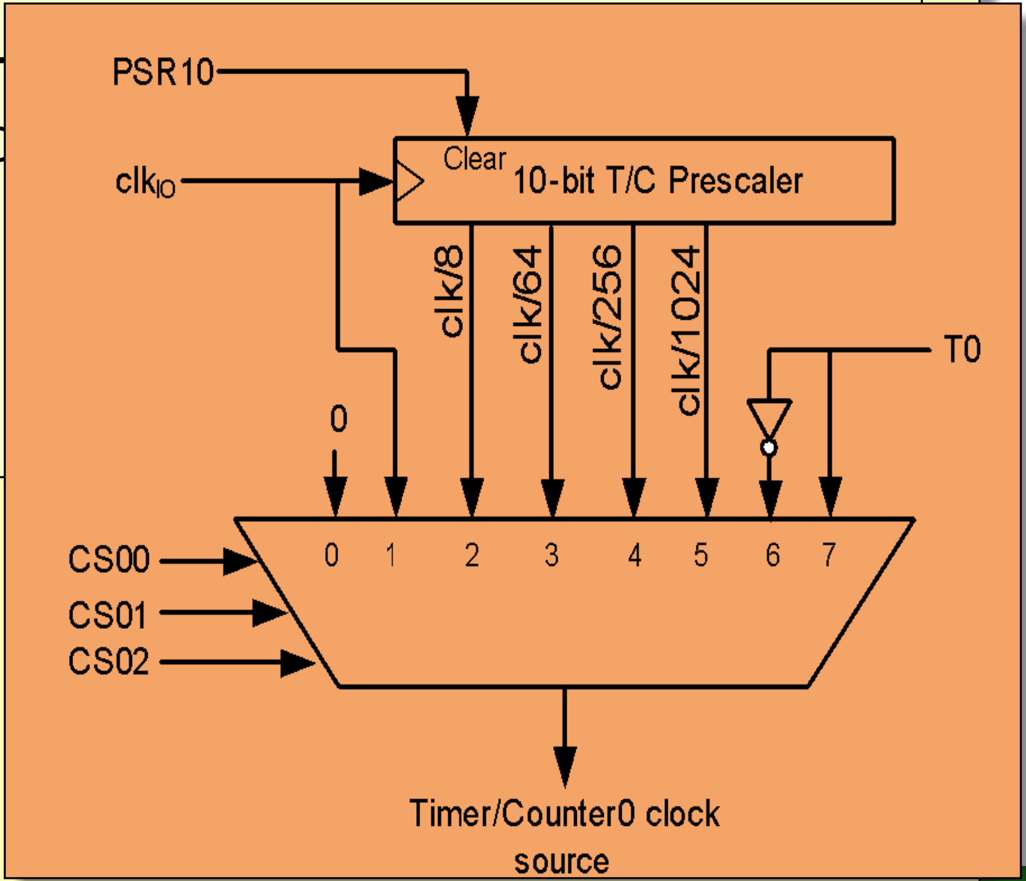


FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

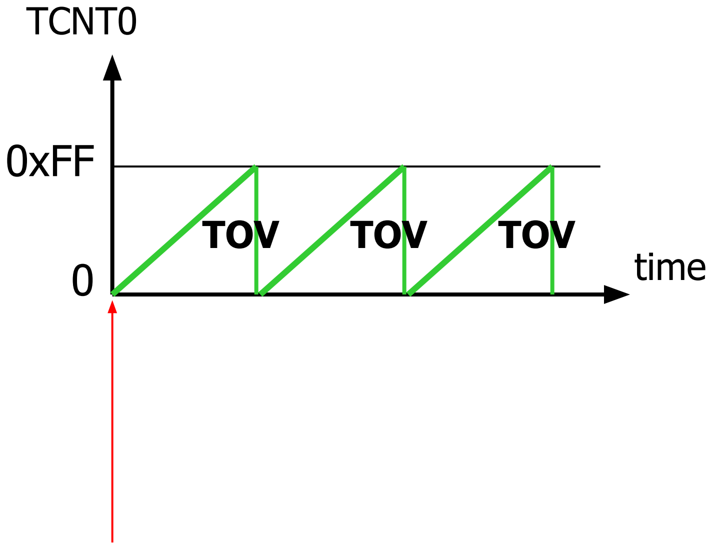
TCCR0

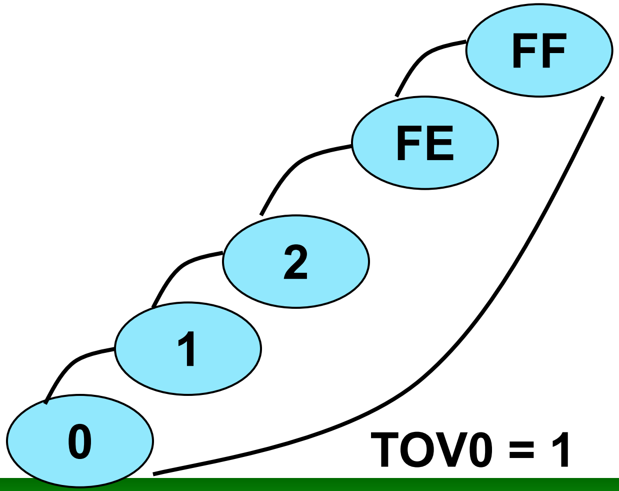
Timer Mode (WGM)

WGM00	WGM01	Comment
0	0	Normal
0	1	CTC (Clear Timer on C
1	0	PWM, phase correct
1	1	Fast PWM



Normal mode



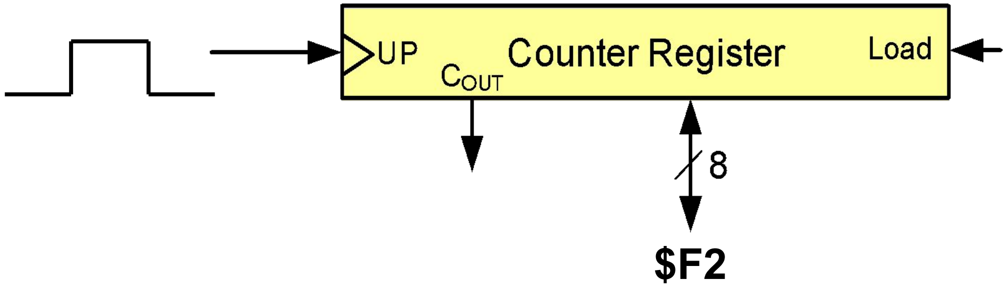


TOV0: 1

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

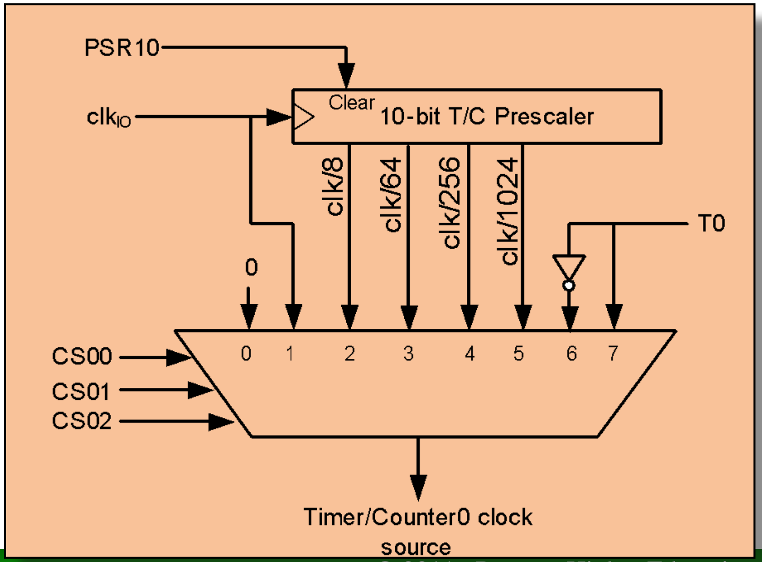
14 = $0E

$100
-$0E
$F2



FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
	0			0	0	0	1	

WGM00	WGM01	Comment
0	0	Normal
0	1	CTC
1	0	PWM, phase correct
1	1	Fast PWM



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

```
  $100
 -$0E
 ------
   $F2
```

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR

```
.INCLUDE "M32DEF.INC"

        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  R20,0x01
        OUT  TCCR0,R20 ;Timer0,Norm
AGAIN:  IN   R20,TIFR       ;read
        SBRS R20,0 ;if TOV0 is set
        RJMP AGAIN
        LDI  R20,0x0
        OUT  TCCR0,R20      ;stop Tim
        LDI  R20,(1<<TOV0)     ;R20
        OUT  TIFR,R20       ;clear TO

        EOR  R17,R16        ;toggle D
        OUT  PORTB,R17      ;toggle P
        RJMP     BEGIN
```

```
DDRB = 1<<5;
PORTB &= ~(1<<5);  //PB5=0
while (1)
{
  TCNT0 = 0xF2;
```

Question: How to calculate the delay generated by the timer?

Answer:

1) Calculate how much a machine clock lasts.
  T= 1/f

2) Calculate how many machine clocks it waits.

3) Delay = T * number of machine cycles

In example 1 calculate the delay. Imagine XTAL = 10 MHz.

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

```
.INCLUDE "M32DEF.INC"

        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  R20,0x01
        OUT  TCCR0,R20 ;Timer0,Normal mode,int clk
AGAIN:  IN   R20,TIFR      ;read TIFR
        SBRS R20,0 ;if TOV0 is set skip next inst.
        RJMP AGAIN
        LDI  R20,0x0
        OUT  TCCR0,R20     ;stop Timer0
        LDI  R20,0x01
        OUT  TIFR,R20      ;clear TOV0 flag

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

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

```
        LDI     R16,0x20
        SBI     DDRB,5
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:          LDI     R20,0xF2                     1
        OUT     TCNT0,R20                       1
        LDI     R20,0x01                        1
        OUT     TCCR0,R20                       1
AGAIN:          IN      R20,TIFR                     1
        SBRS R20,0                          1 / 2
        RJMP AGAIN                              2
        LDI     R20,0x0                         1
        OUT     TCCR0,R20                       1
        LDI     R20,0x01                        1
        OUT     TIFR,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

Finding values to be loaded into the timer

1. Calculate the period of clock source.
   - Period = 1 / Frequency
     - E.g. For XTAL = 8 MHz □ T = 1/8MHz
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

Example 2: Assuming that XTAL = 10 MHz, write a program to generate a square wave with a period of 10 ms 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.

```
.INCLUDE "M32DEF.INC"

        LDI  R16,0x08
        SBI  DDRB,3   ;PB3 as an output
        LDI  R17,0
        OUT  PORTB,R17
BEGIN:  LDI  R20,206
        OUT  TCNT0,R20     ;load timer0
        LDI  R20,0x01
        OUT  TCCR0,R20 ;Timer0,Normal mode,int clk
AGAIN:  IN   R20,TIFR      ;read TIFR
        SBRS R20,TOV0 ;if TOV0 is set skip next
        RJMP AGAIN
        LDI  R20,0x0
        OUT  TCCR0,R20     ;stop Timer0
        LDI  R20,0x01
        OUT  TIFR,R20      ;clear TOV0 flag
        EOR  R17,R16       ;toggle D3 of R17
        OUT  PORTB,R17     ;toggle PB3
        RJMP     BEGIN
```

```
DDRB = 1<<3;
PORTB &= ~ (1<<3);
while (1)
{
  TCNT0 = 206;
  TCCR0 = 0x01;
  while((TIFR&0x01) == 0);
  TCCR0 = 0;
  TIFR = 1<<TOV0;
  PORTB = PORTB ^ (1<<3);
}
```

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.

```
.INCLUDE "M32DEF.INC"

    LDI  R16,1<<3
    SBI  DDRB,3     ;PB3 as an output
    LDI  R17,0
    OUT  PORTB,R17
BEGIN:   LDI  R20,0x0
    OUT  TCNT0,R20     ;load Timer0
    LDI  R20,0x01
    OUT  TCCR0,R20 ;Timer0,Normal mode,int clk
AGAIN:   IN   R20,TIFR
    SBRS R20,TOV0  ;if T
    RJMP AGAIN
    LDI  R20,0x0
    OUT  TCCR0,R20     ;s
    LDI  R20,0x01
    OUT  TIFR,R20      ;c
    EOR  R17,R16       ;t
    OUT  PORTB,R17     ;t
    RJMP     BEGIN
```

```
DDRB = 1 << 3;
PORTB &= ~(1<<3);
while (1)
{
   TCNT0 = 0x0;
   TCCR0 = 0x01;

while((TIFR&(1<<TOV0))==0);
   TCCR0 = 0;
   TIFR = 0x01;
   PORTB = PORTB^(1<<3);
}
```

Solution

1) Calculating T:

T = 1/f = 1/10MHz = 0.1µs

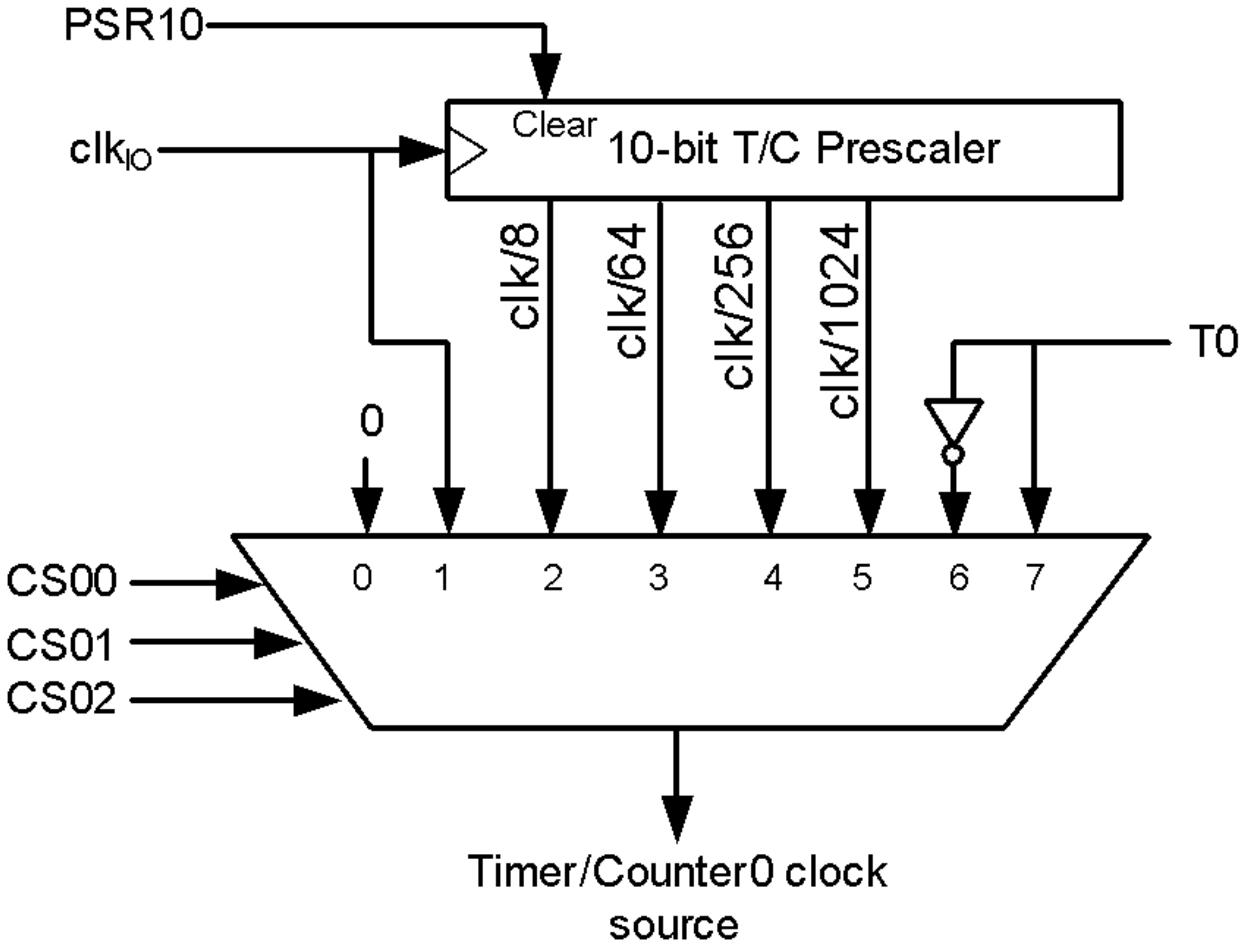
2) Calculating delay

256 * 0.1µs = 25.6µs

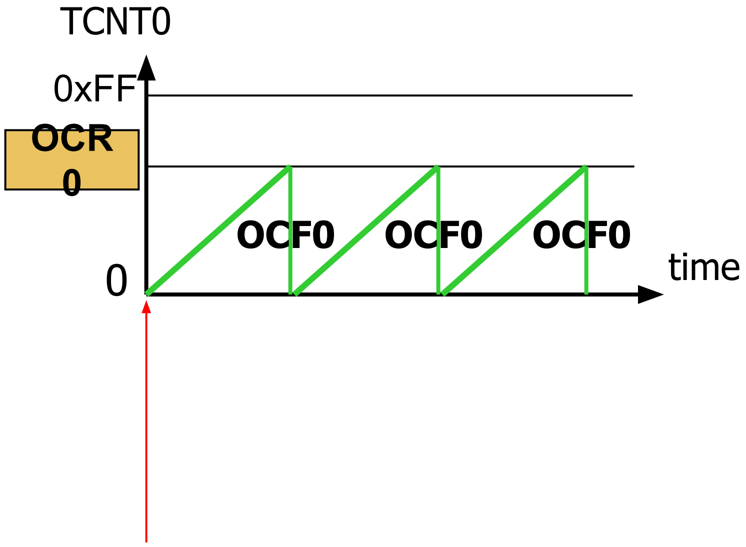
Generating Large Delays

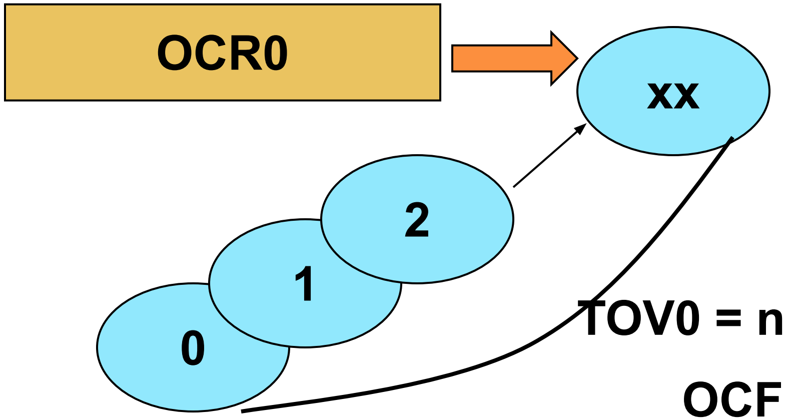
- Using loop
- Prescaler
- Bigger counters

Prescaler and generating a large time delay



CTC (Clear Timer on Compare match) mode





TOV0 = no change

OCF0 = 1

TOV0: 0

OCF0: 1

Rewrite example 2 using CTC

FOC0	WGM00	COM01	COM00	WGM01	**CS02**	**CS01**	**CS00**	TCCR0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR

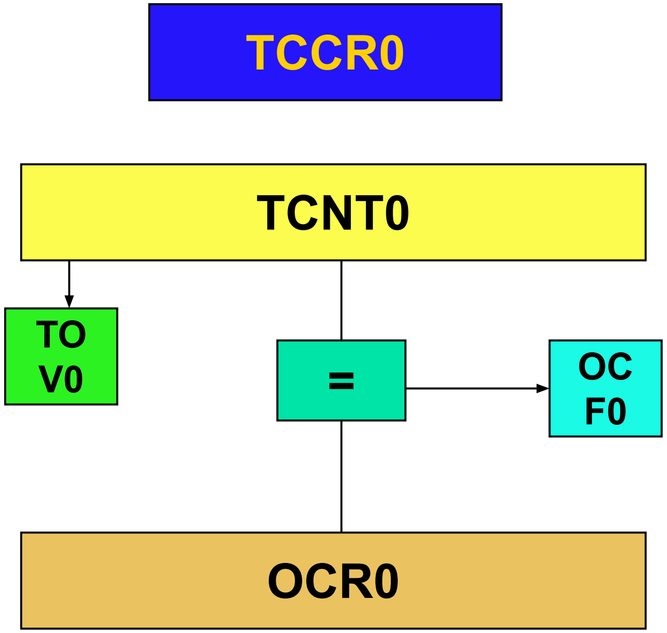
- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. Therefore, we have OCR0= 49.

```
.INCLUDE "M32DEF.INC"
        LDI  R16,0x08
        SBI  DDRB,3   ;PB3 as an output
        LDI  R17,0
        OUT  PORTB,R17
        LDI  R20,49
        OUT  OCR0,R20 ;load timer0
BEGIN:  LDI  R20,0x09
        OUT  TCCR0,R20 ;Timer0,CTC mode,int clk
AGAIN:  IN   R20,TIFR      ;read TIFR
        SBRS R20,OCF0 ;if OCF0 is set skip next
        RJMP AGAIN
        LDI  R20,0x0
        OUT  TCCR0,R20     ;stop Timer0
        LDI  R20,0x02
        OUT  TIFR,R20      ;clear TOV0 flag
        EOR  R17,R16       ;toggle D3 of R17
        OUT  PORTB,R17     ;toggle PB3
        RJMP     BEGIN
```

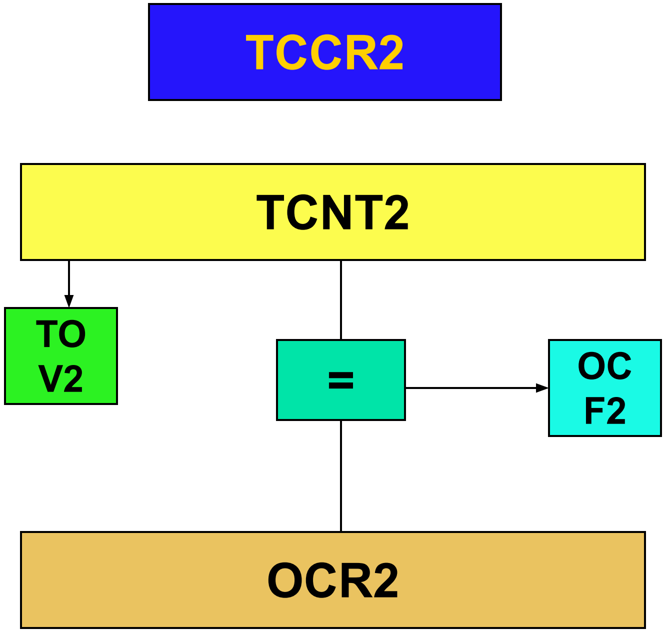
```
DDRB |= 1<<3;
PORTB &= ~(1<<3);
while (1)
{
   OCR0 = 49;
   TCCR0 = 0x09;

   while((TIFR&(1<<OCF0))==0);
   TCCR0 = 0; //stop timer0
   TIFR = 0x02;
   PORTB.3 = ~PORTB.3;
}
```

Timer2

- Timer0

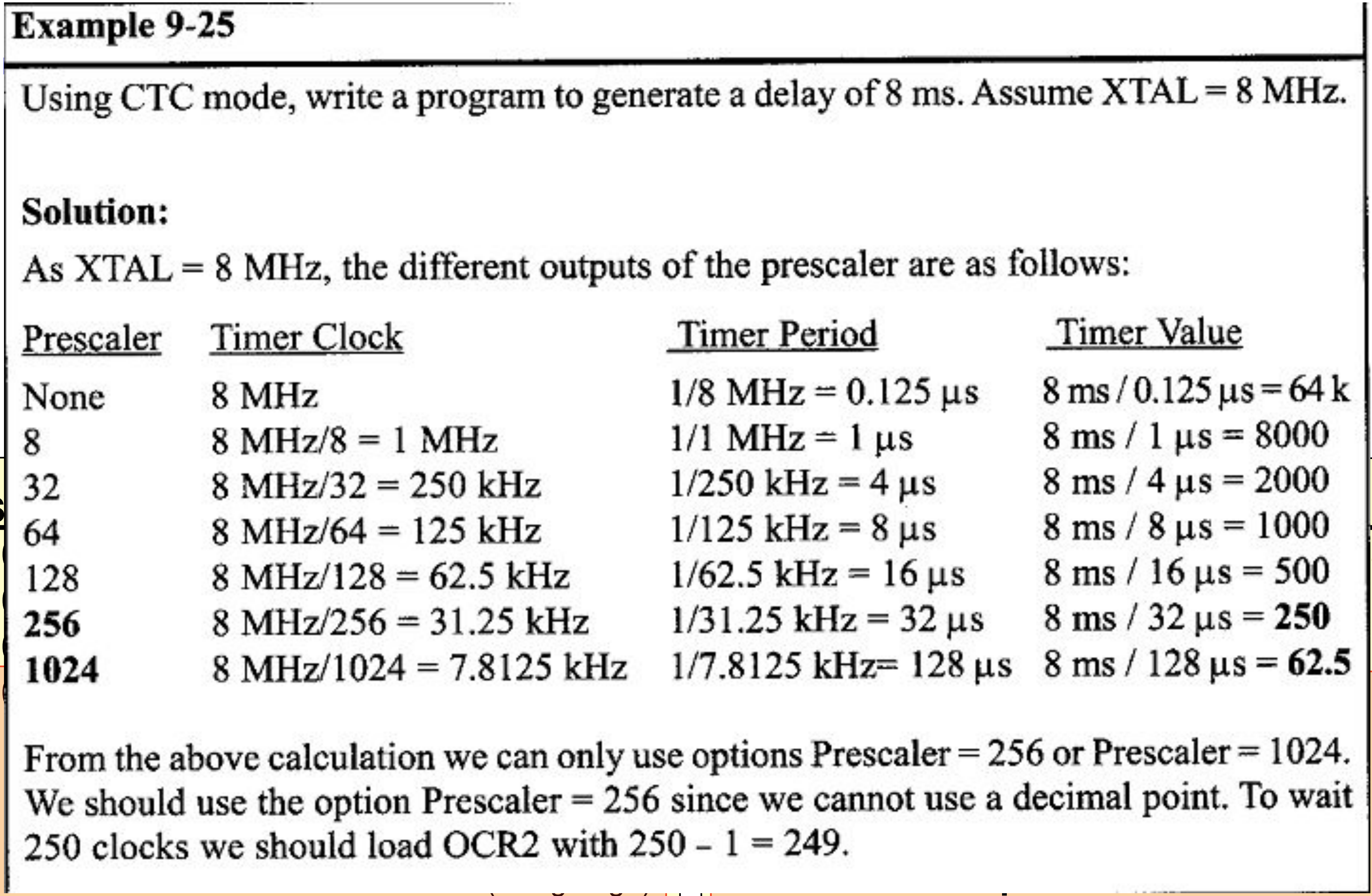


- Timer2



OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR

The difference between Timer0 and Timer2



Example 9-25

Using CTC mode, write a program to generate a delay of 8 ms. Assume XTAL = 8 MHz.

Solution:

As XTAL = 8 MHz, the different outputs of the prescaler are as follows:

Prescaler	Timer Clock	Timer Period	Timer Value
None	8 MHz	1/8 MHz = 0.125 µs	8 ms / 0.125 µs = 64 k
8	8 MHz/8 = 1 MHz	1/1 MHz = 1 µs	8 ms / 1 µs = 8000
32	8 MHz/32 = 250 kHz	1/250 kHz = 4 µs	8 ms / 4 µs = 2000
64	8 MHz/64 = 125 kHz	1/125 kHz = 8 µs	8 ms / 8 µs = 1000
128	8 MHz/128 = 62.5 kHz	1/62.5 kHz = 16 µs	8 ms / 16 µs = 500
256	8 MHz/256 = 31.25 kHz	1/31.25 kHz = 32 µs	8 ms / 32 µs = **250**
1024	8 MHz/1024 = 7.8125 kHz	1/7.8125 kHz= 128 µs	8 ms / 128 µs = **62.5**

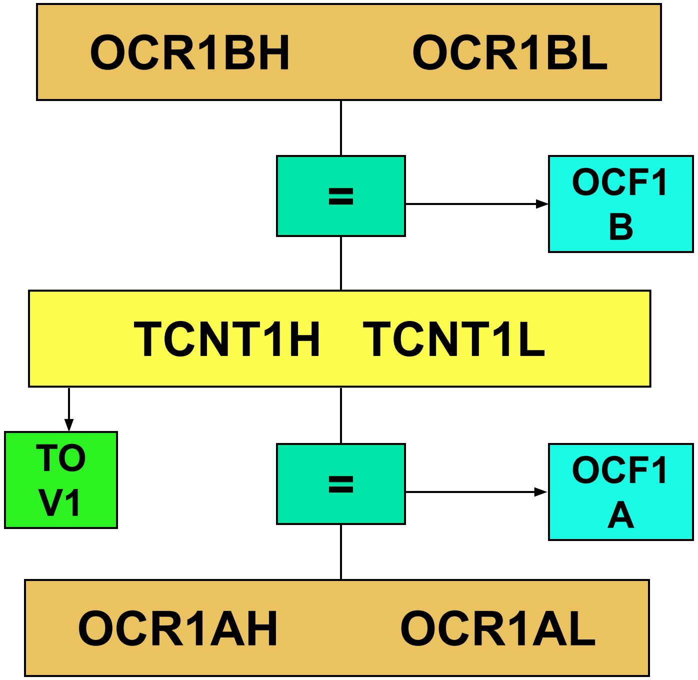
From the above calculation we can only use options Prescaler = 256 or Prescaler = 1024. We should use the option Prescaler = 256 since we cannot use a decimal point. To wait 250 clocks we should load OCR2 with 250 – 1 = 249.

cs

ed

Timer 1

TCCR1A TCCR1B

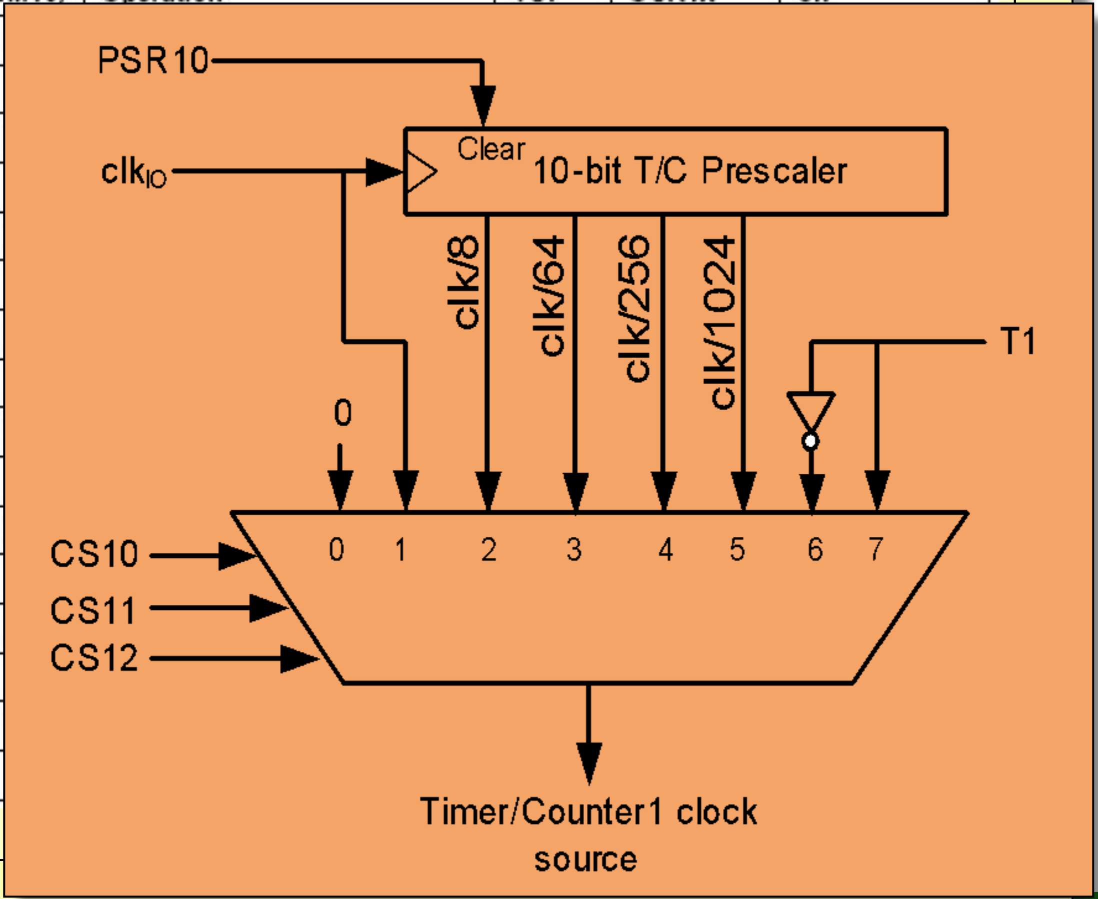


OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR

COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
ICNC1	ICES1	-	WGM13	WGM12	CS12	C 1	C 0	TCCR1B

Clock Se ctor (S)

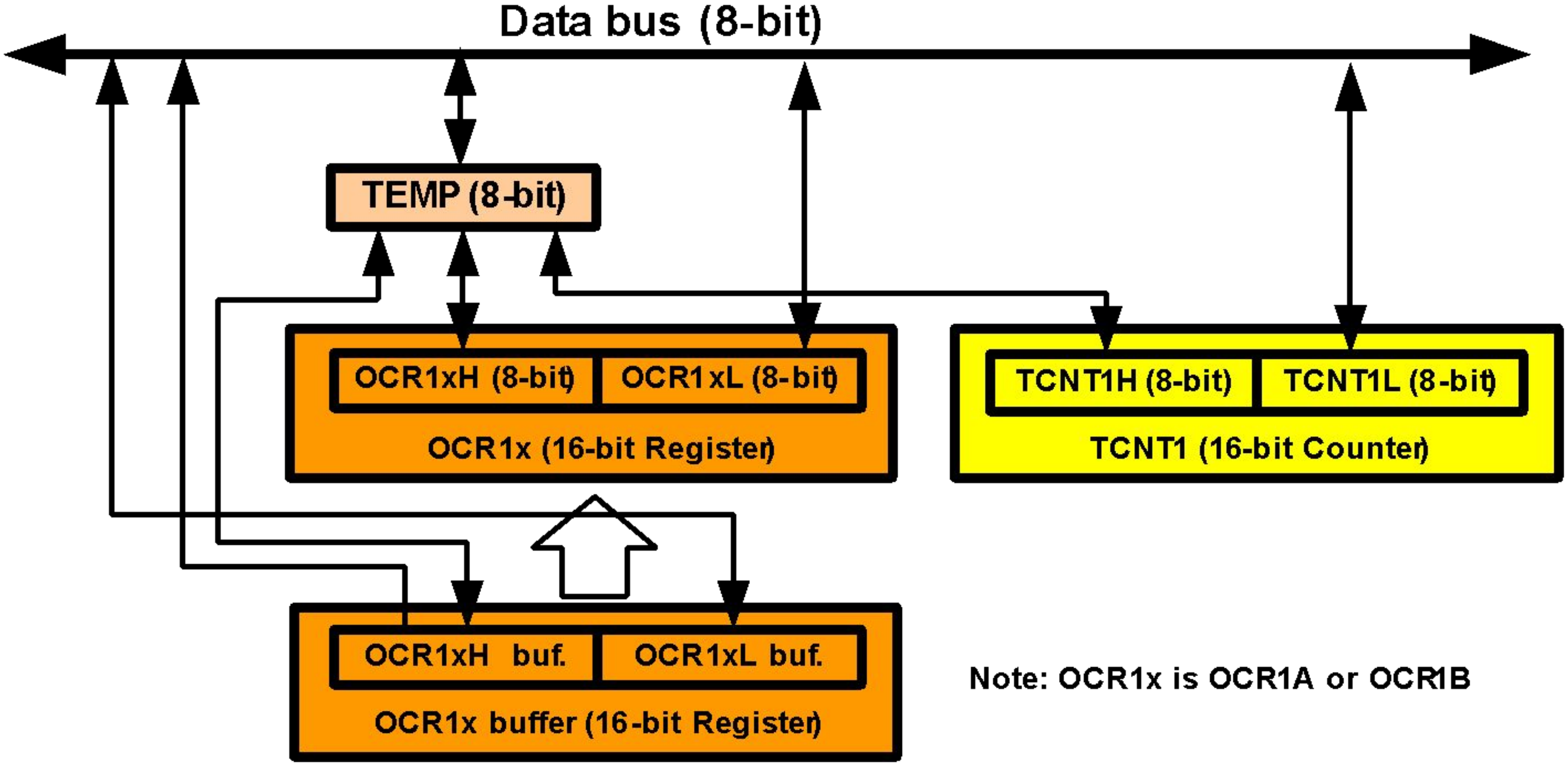
Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0					
1	0	0	0					
2	0	0	1					
3	0	0	1					
4	0	1	0					
5	0	1	0					
6	0	1	1					
7	0	1	1					
8	1	0	0					
9	1	0	0					
10	1	0	1					
11	1	0	1					
12	1	1	0					
13	1	1	0					
14	1	1	1					
15	1	1	1					



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

```
.INCLUDE "M32DEF.INC"
    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)
    OUT    TCNT1H,R20
    LDI    R20, ,LOW(-10000)
    OUT    TCNT1L,R20          ;Timer1 overflows after 10000 machine cycles
    OUT    TCCR1A,R20      ;WGM11:10=00
    LDI    R20,0x1
    OUT    TCCR1B,R20      ;WGM13:12=00,CS=CLK
AGAIN:IN   R20,TIFR        ;read TIFR
    SBRS   R20,TOV1        ;if OCF1A is set skip next instruction
    RJMP   AGAIN
    LDI    R20,1<<TOV1
    OUT    TIFR,R20        ;clear TOV1 flag
    LDI    R19,0
    OUT    TCCR1B,R19      ;stop timer
    OUT    TCCR1A,R19      ;
    RET
```

TEMP register



Note: OCR1x is OCR1A or OCR1B

```
LDI   R20,0xF3
OUT   TCNT1H,R20
LDI   R20,0x53
OUT   TCNT1L,R20
```

```
IN   R20,TCNT1L
IN   R21,TCNT1H
```

```
TCNT1H = 0xF3;
TCNT1L = 0x53;
```

```
a = TCNT1L;
b = TCNT1H;
```

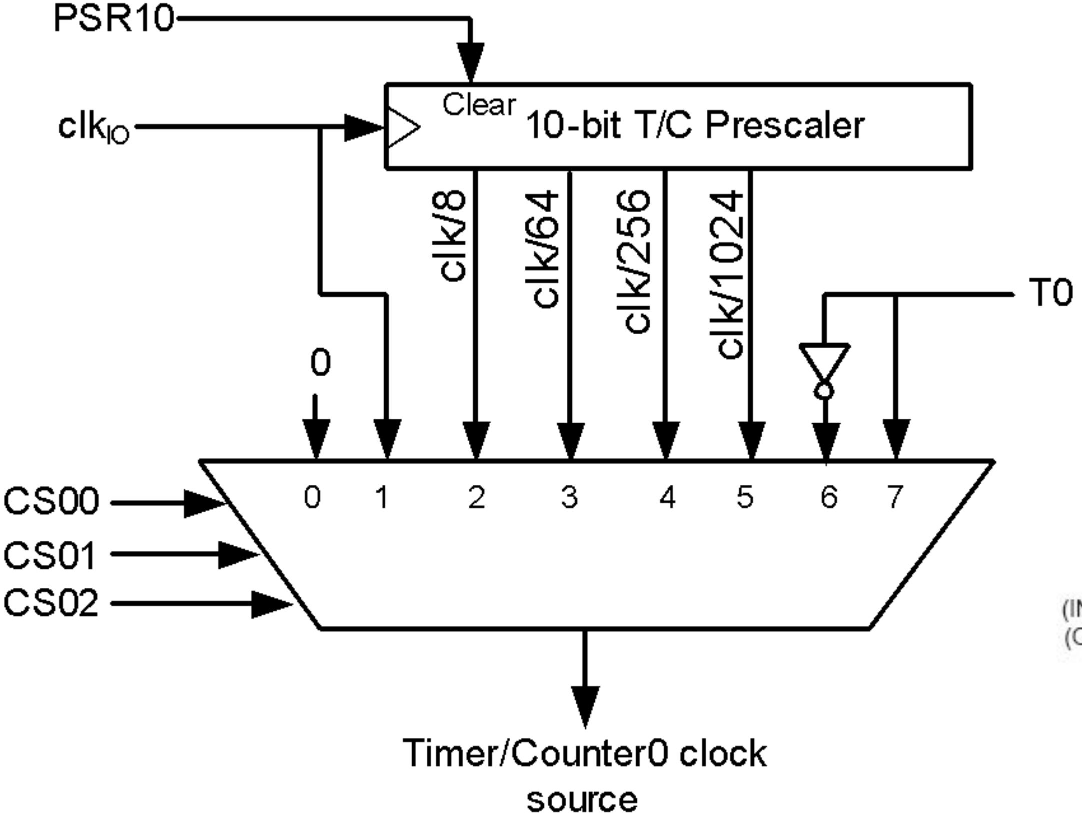
Assuming XTAL = 10 MHz write a program that toggles PB5 once per millisecond, using CTC mode.

```
.INCLUDE "M32DEF.INC"
     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
     OUT    TCNT1H,R20        ;TEMP = 0
     OUT    TCNT1L,R20        ;TCNT1L = 0, TCNT1H = TEMP

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

     LDI    R20,0x3
     OUT    TCCR1A,R20        ;WGM11:10=11
     LDI    R20,0x19
     OUT    TCCR1B,R20        ;WGM13:12=11,CS=CLK
AGAIN:
     IN     R20,TIFR          ;read TIFR
     SBRS   R20,OCF1A         ;if OCF1A is set skip next instruction
     RJMP   AGAIN
     LDI    R20,1<<OCF1A
     OUT    TIFR,R20          ;clear OCF1A flag
     LDI    R19,0
     OUT    TCCR1B,R19        ;stop timer
     OUT    TCCR1A,R19        ;
     RET
```

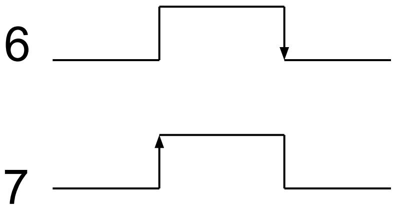
Counting

The AVR microcontroller
and embedded
systems
using assembly and c

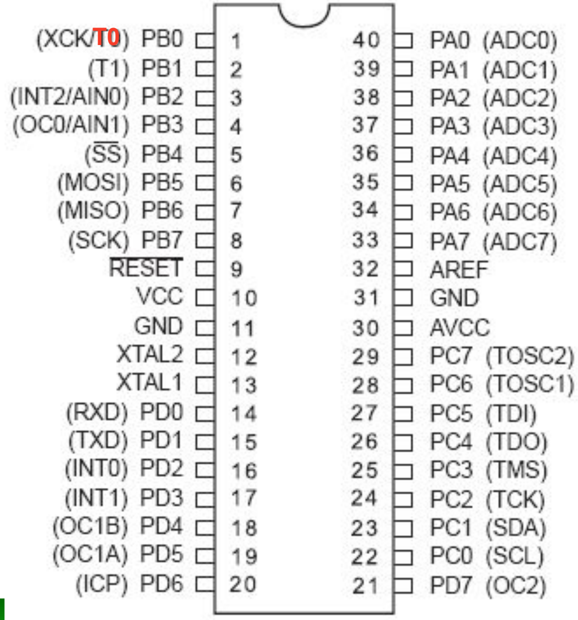
Counting



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



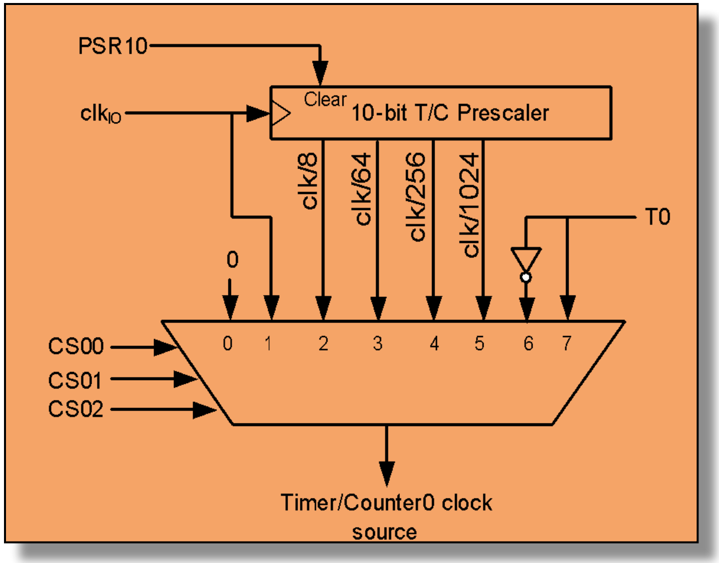
6
7



(XCK/T0) PB0 1
(T1) PB1 2
(INT2/AIN0) PB2 3
(OC0/AIN1) PB3 4
(SS) PB4 5
(MOSI) PB5 6
(MISO) PB6 7
(SCK) PB7 8
RESET 9
VCC 10
GND 11
XTAL2 12
XTAL1 13
(RXD) PD0 14
(TXD) PD1 15
(INT0) PD2 16
(INT1) PD3 17
(OC1B) PD4 18
(OC1A) PD5 19
(ICP) PD6 20
40 PA0 (ADC0)
39 PA1 (ADC1)
38 PA2 (ADC2)
37 PA3 (ADC3)
36 PA4 (ADC4)
35 PA5 (ADC5)
34 PA6 (ADC6)
33 PA7 (ADC7)
32 AREF
31 GND
30 AVCC
29 PC7 (TOSC2)
28 PC6 (TOSC1)
27 PC5 (TDI)
26 PC4 (TDO)
25 PC3 (TMS)
24 PC2 (TCK)
23 PC1 (SDA)
22 PC0 (SCL)
21 PD7 (OC2)

Example 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.

```
.INCLUDE "M32DEF.INC"
    CBI   DDRB,0         ;make T0 (PB0) input
    LDI   R20,0xFF
    OUT   DDRC,R20       ;make PORTC output
    LDI   R20,0x06
    OUT   TCCR0,R20      ;counter, falling edge
AGAIN:
    IN    R20,TCNT0
    OUT   PORTC,R20      ;PORTC = TCNT0
    IN    R16,TIFR
    SBRS  R16,TOV0
    RJMP  AGAIN          ;keep doing it
    LDI   R16,1<<TOV0
    OUT   TIFR, R16
    RJMP  AGAIN          ;keep doing it
```



FOC0	WGM00	COM01	COM00	WGM01	**CS02**	**CS01**	**CS00**

TCCR0

Assuming that clock pulses are fed into pin T1. Write a program for counter 1 in CTC mode to make PORTC.0 high every 100 pulses.

```
.INCLUDE "M32DEF.INC"

   CBI  DDRB,1          ;make T1 (PB1) input

   SBI  DDRC,0          ;PC0 as an output

   LDI  R20,0x0
   OUT  TCCR1A,R20
   LDI  R20,0x0E
   OUT  TCCR1B,R20      ;CTC, counter, falling edge
AGAIN:
   LDI  R20,0
   OUT  OCR1AH,R20      ;TEMP = 0
   LDI  R20,99
   OUT  OCR1AL,R20      ;ORC1L = R20, OCR1H = TEMP
L1:IN   R20,TIFR
   SBRS R20,OCF1A
   RJMP L1          ;keep doing it
   LDI  R20,1<<OCF1A ;clear OCF1A flag
   OUT  TIFR, R20

   SBI  PORTC,0         ;PC0 = 1
   CBI  PORTC,0         ;PC0 = 0
   RJMP AGAIN           ;keep doing it
```