Microprocessors, Lecture 5:

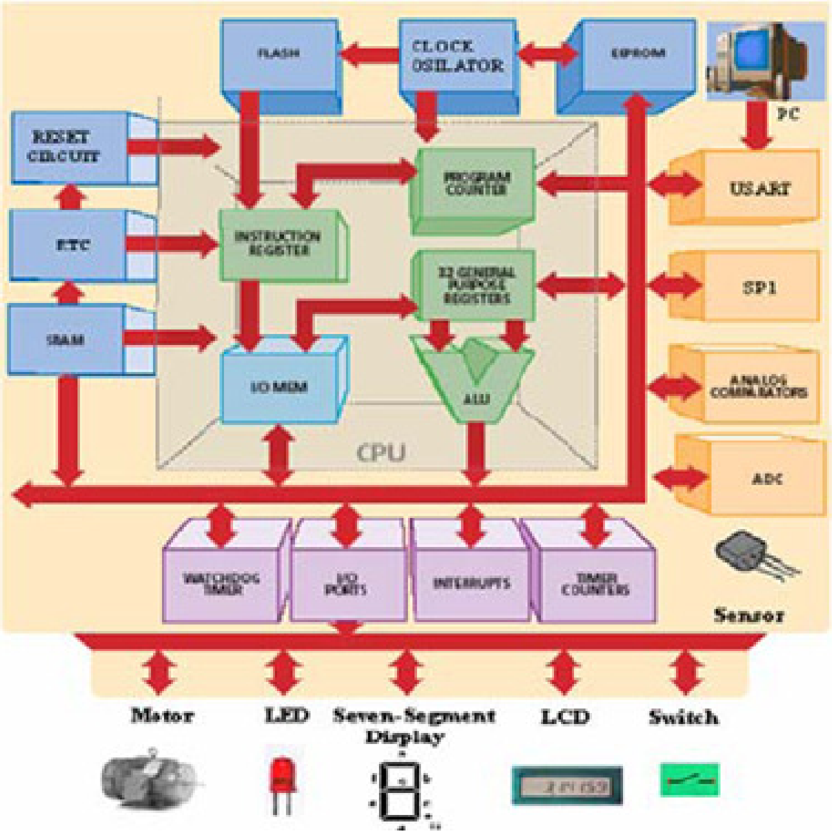
AVR Microcontrollers
-Timers

(Chapter 9 of the text book)

Contents

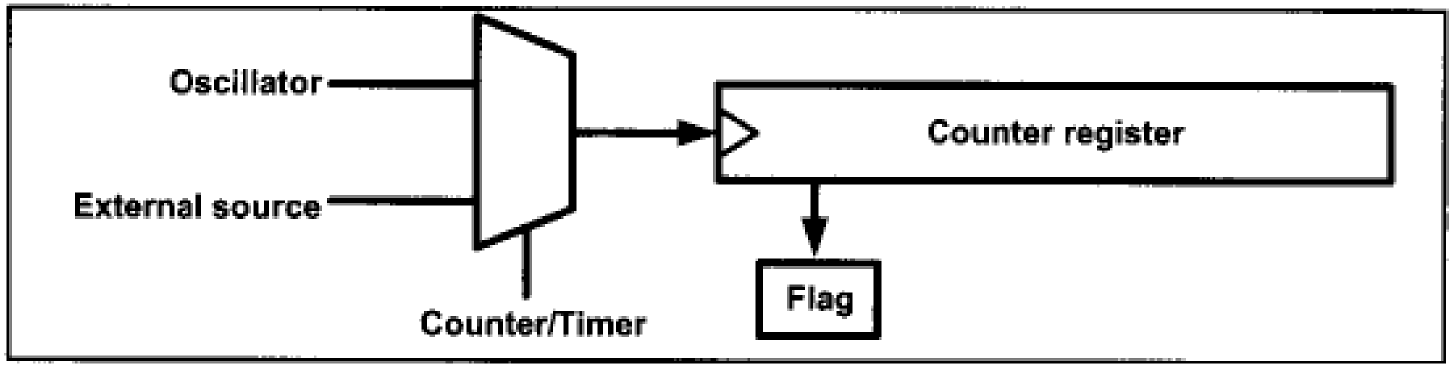
- Timers 0 and 2 of ATmega32
- Timer programming in C

Timers in AVR



Timer/Counter

- What is a timer?
  - To count an event
  - To generate delay

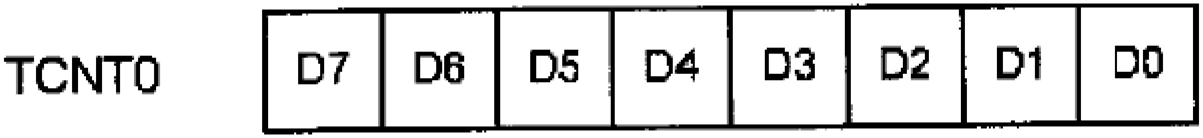


Timers in AVR

- ATmega32: 3 timers
  - Timer0 (8-bit)
  - Timer1 (16-bit)
  - Timer2 (8-bit)

Timers in AVR

- Basic registers:
  - TCNTx (x=0,1,2)= timer/counter register
  - Keeps the timer/counter value
  - On reset, contains 0
  - Counts up with each pulse



TCNT0	D7	D6	D5	D4	D3	D2	D1	D0

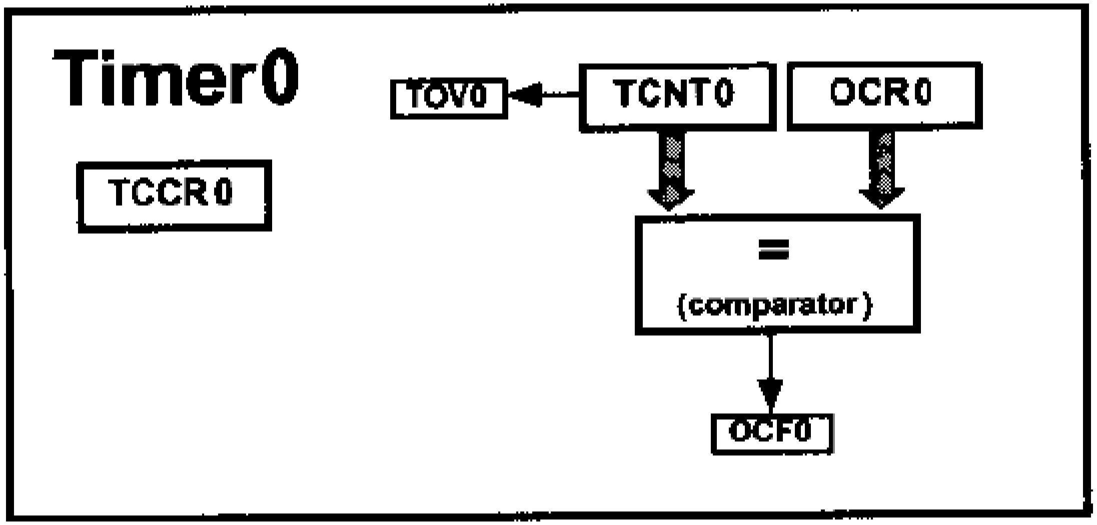
Timers in AVR

- Basic registers:
  - TOVx(x=0,1,2)= timer/counter overflow flag
  - TOVx Becomes 1 when TCNTx overflows
    - » switches from 0xFF to 0x00
  - Should be reset by software

Timers in AVR

- Basic registers:
  - OCRx (x=1,2,3)= output compare register
  - Another way to count
  - The contents of OCRx are compared to TCNTx
    - OCFx is set if they are equal

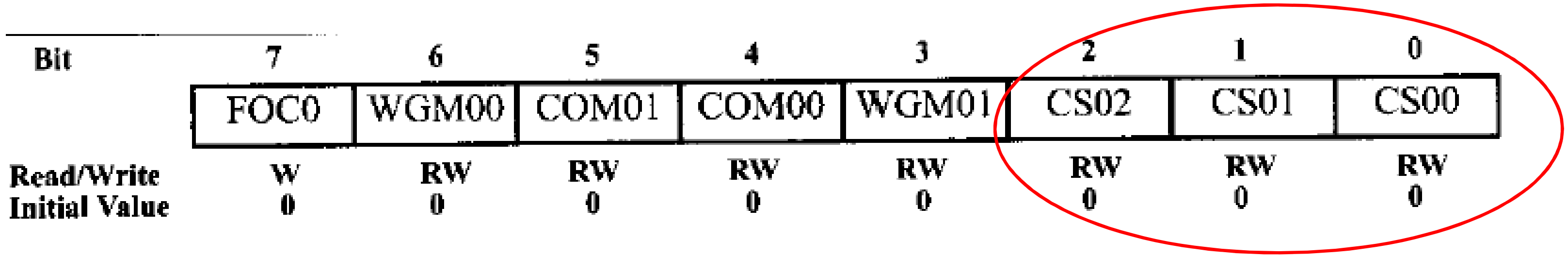
Timers in AVR



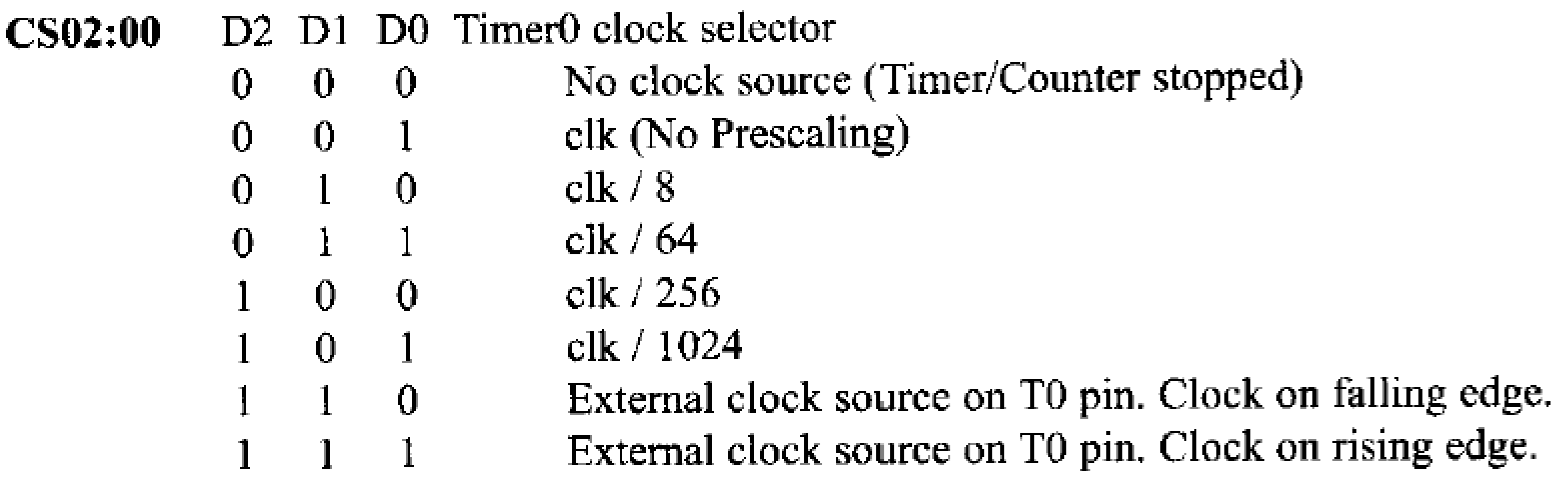
Timers in AVR

- Basic registers:
  - TCCRx (x=1,2,3)= timer/counter control register
  - Setting modes of operation

TCCR0 in AVR



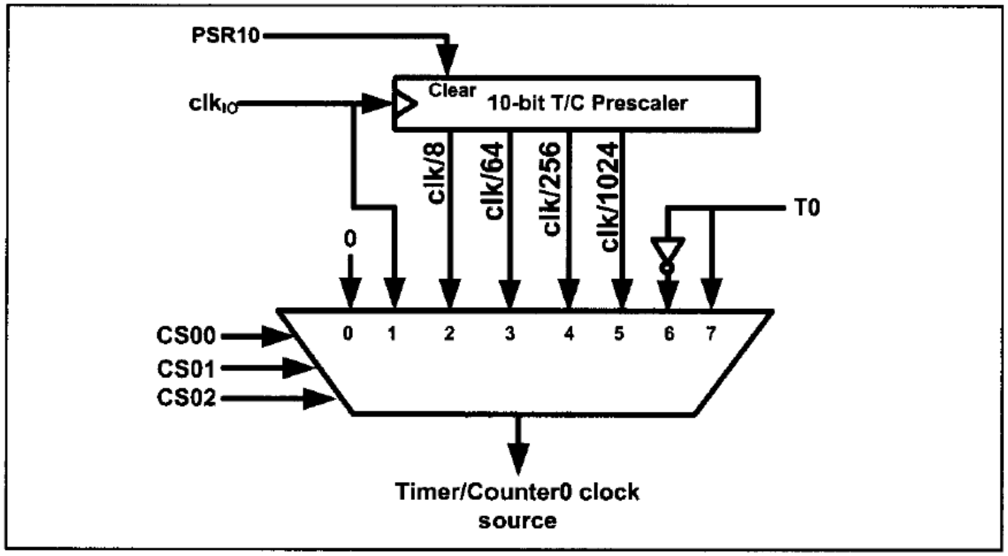
Bit	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Read/Write	W	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

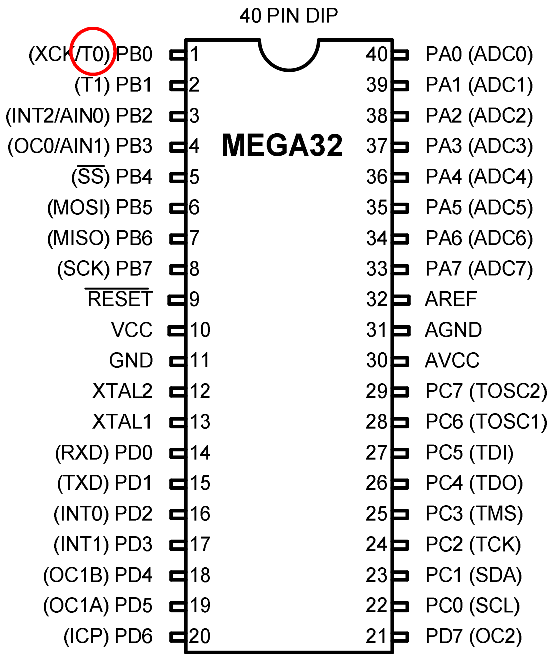


CS02:00	D2	D1	D0	Timer0 clock selector
	0	0	0	No clock source (Timer/Counter stopped)
	0	0	1	clk (No Prescaling)
	0	1	0	clk / 8
	0	1	1	clk / 64
	1	0	0	clk / 256
	1	0	1	clk / 1024
	1	1	0	External clock source on T0 pin. Clock on falling edge.
	1	1	1	External clock source on T0 pin. Clock on rising edge.

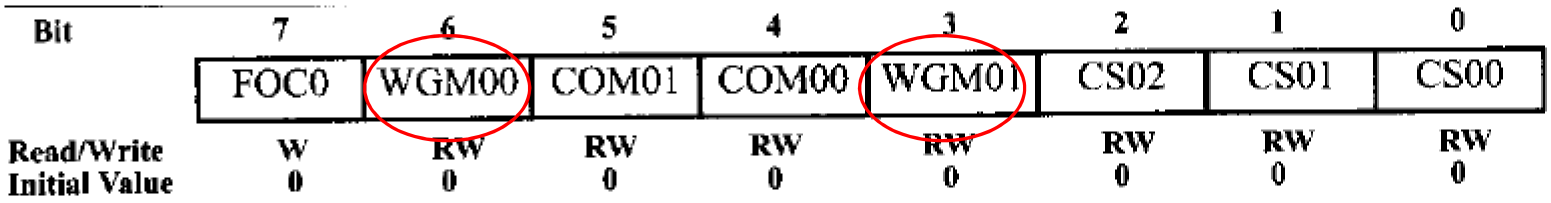
TCCR0 in AVR

CS02:00	D2	D1	D0	Timer0 clock selector
	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.

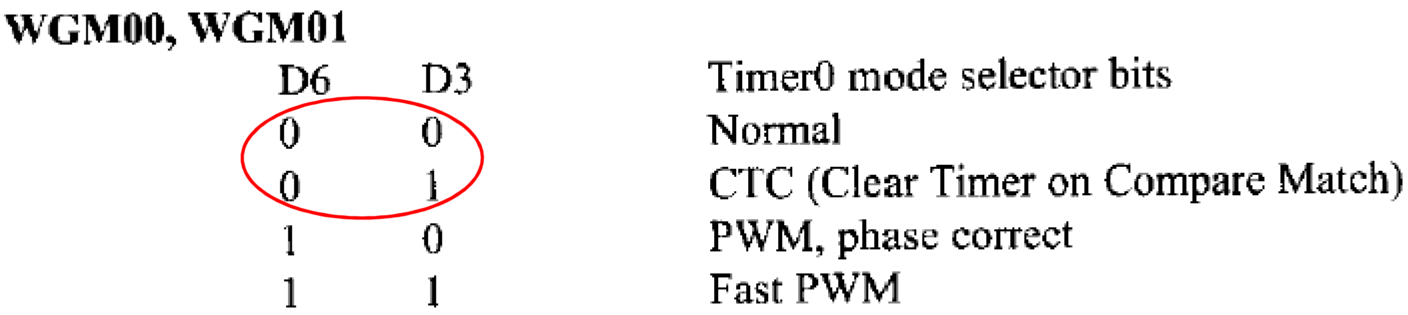




TCCR0 in AVR



Bit	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Read/Write	W	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0



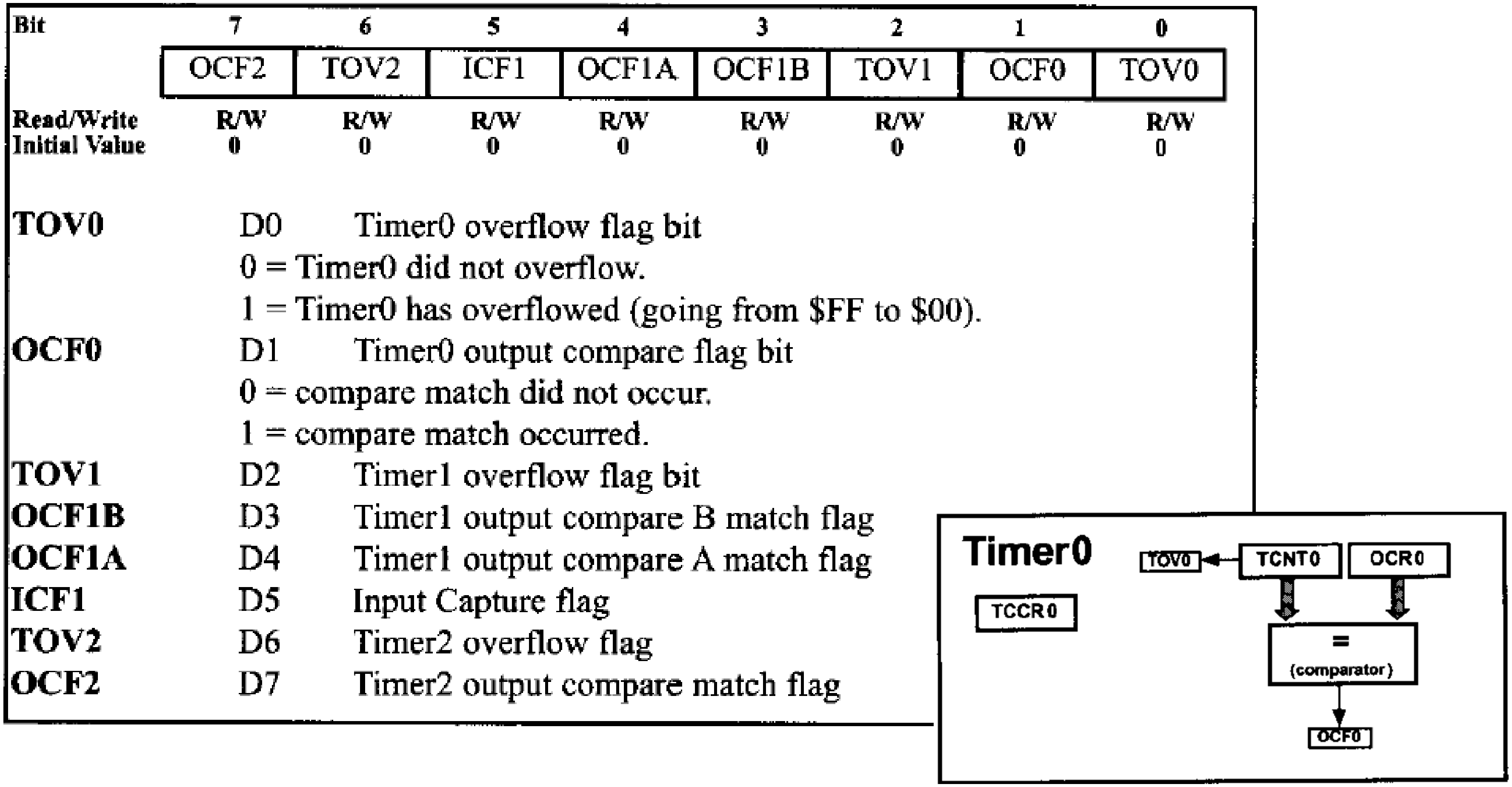
WGM00, WGM01

D6	D3	Timer0 mode selector bits
0	0	Normal
0	1	CTC (Clear Timer on Compare Match)
1	0	PWM, phase correct
1	1	Fast PWM

Timers in AVR

- TIFR (timer/counter interrupt flag register)
- To keep the state of the counters
- One register for all counter/timers

TIFR

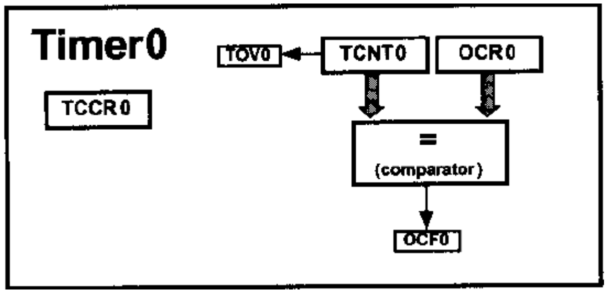


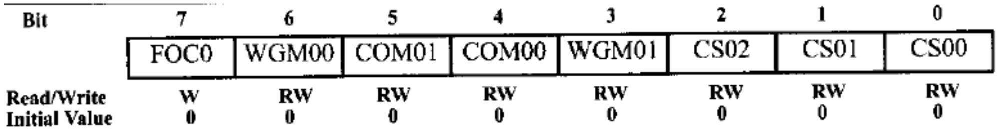
Bit	7	6	5	4	3	2	1	0
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TOV0 D0 Timer0 overflow flag bit
0 = Timer0 did not overflow.
1 = Timer0 has overflowed (going from $FF to $00).
OCF0 D1 Timer0 output compare flag bit
0 = compare match did not occur.
1 = compare match occurred.
TOV1 D2 Timer1 overflow flag bit
OCF1B D3 Timer1 output compare B match flag
OCF1A D4 Timer1 output compare A match flag
ICF1 D5 Input Capture flag
TOV2 D6 Timer2 overflow flag
OCF2 D7 Timer2 output compare match flag

Timer0 in normal mode

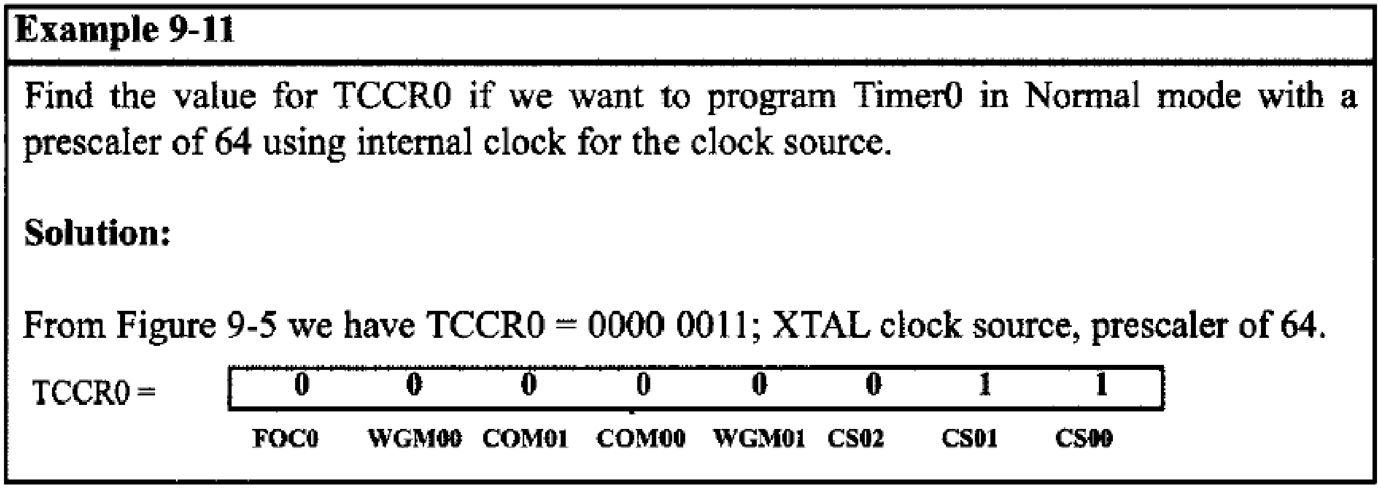
- Set TCNT0 with proper value
- Set TCCR0: which clock source? Which prescalar?
  - When is set, the timer starts
- Keep monitoring TOV0
- Stop timer Set TCCR0
- Clear TOV0





Bit	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Read/Write	W	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

Timers in AVR



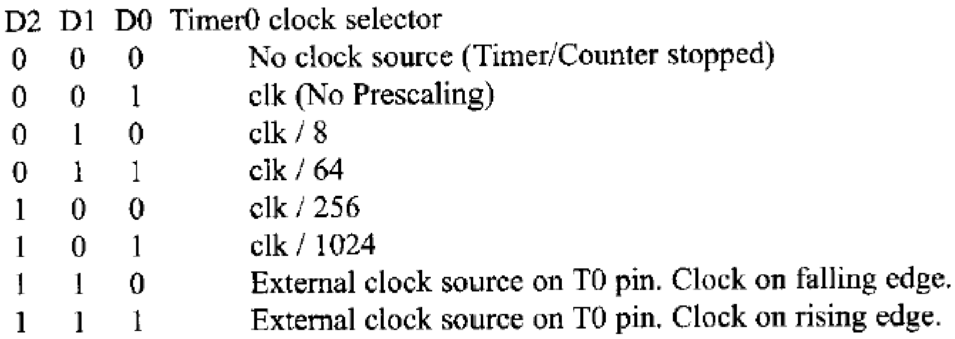
Example 9-11

Find the value for TCCR0 if we want to program Timer0 in Normal mode with a prescaler of 64 using internal clock for the clock source.

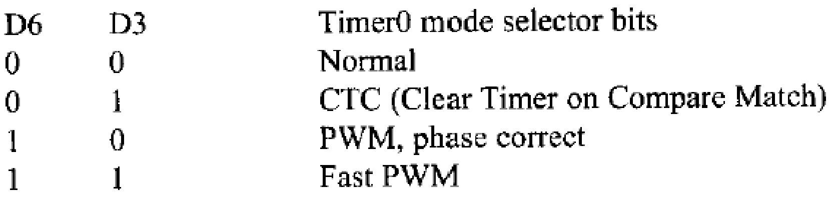
Solution:

From Figure 9-5 we have TCCR0 = 0000 0011; XTAL clock source, prescaler of 64.

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

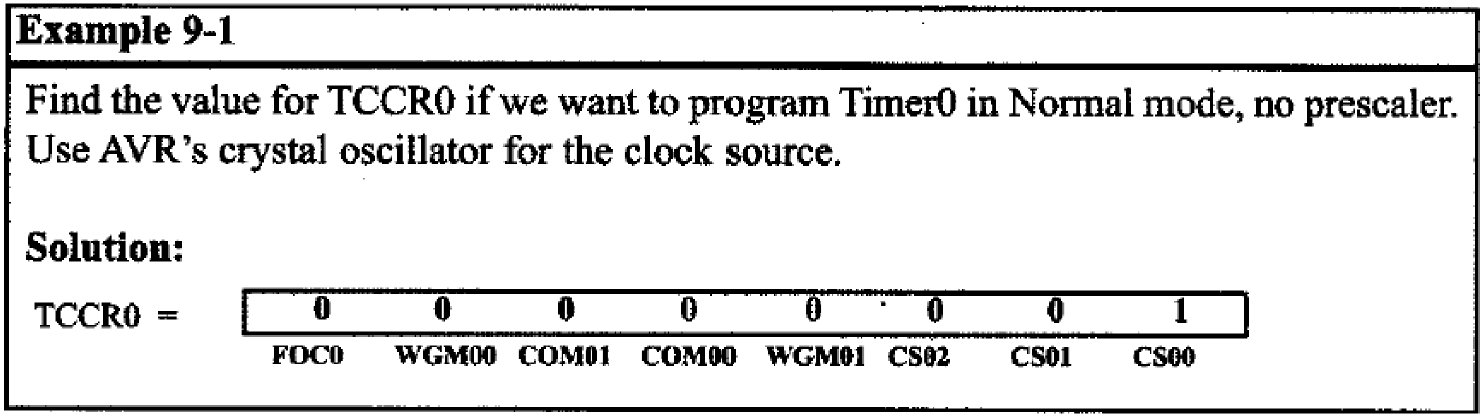


D2	D1	D0	Timer0 clock selector
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.



D6	D3	Timer0 mode selector bits
0	0	Normal
0	1	CTC (Clear Timer on Compare Match)
1	0	PWM, phase correct
1	1	Fast PWM

Timers in AVR



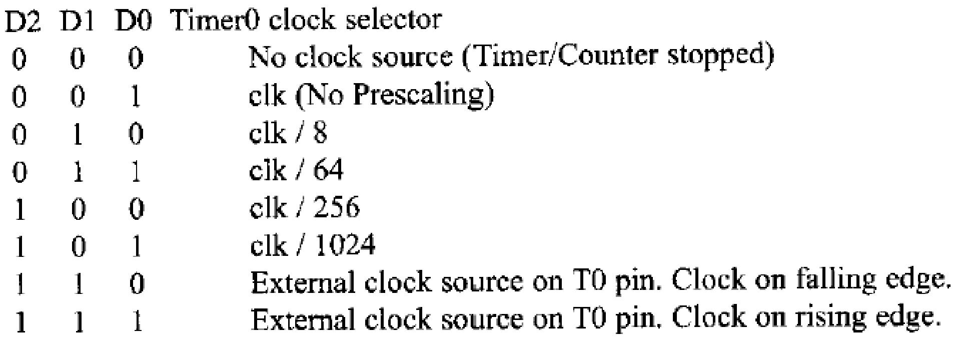
Example 9-1

Find the value for TCCR0 if we want to program Timer0 in Normal mode, no prescaler. Use AVR's crystal oscillator for the clock source.

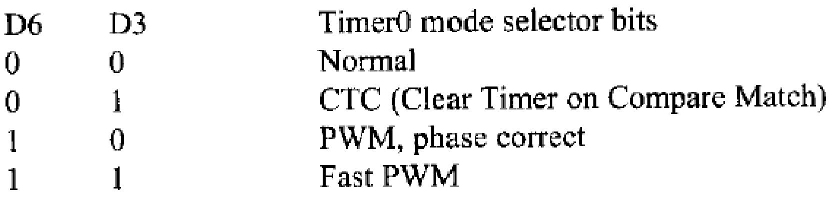
Solution:

TCCR0 =

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

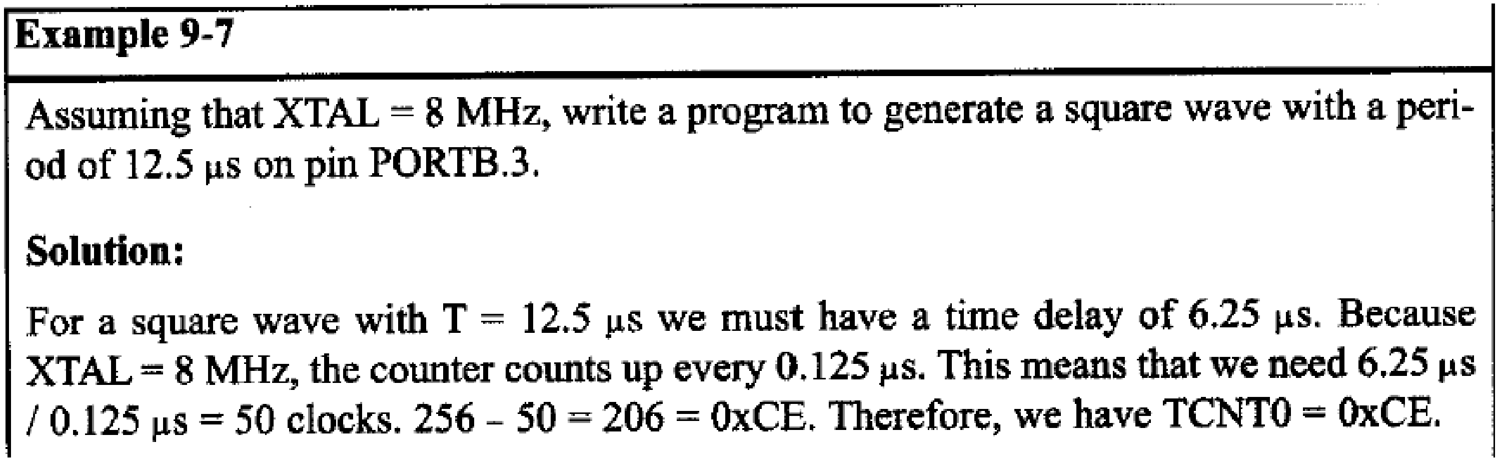


D2	D1	D0	Timer0 clock selector
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.



D6	D3	Timer0 mode selector bits
0	0	Normal
0	1	CTC (Clear Timer on Compare Match)
1	0	PWM, phase correct
1	1	Fast PWM

Timers in AVR



Example 9-7

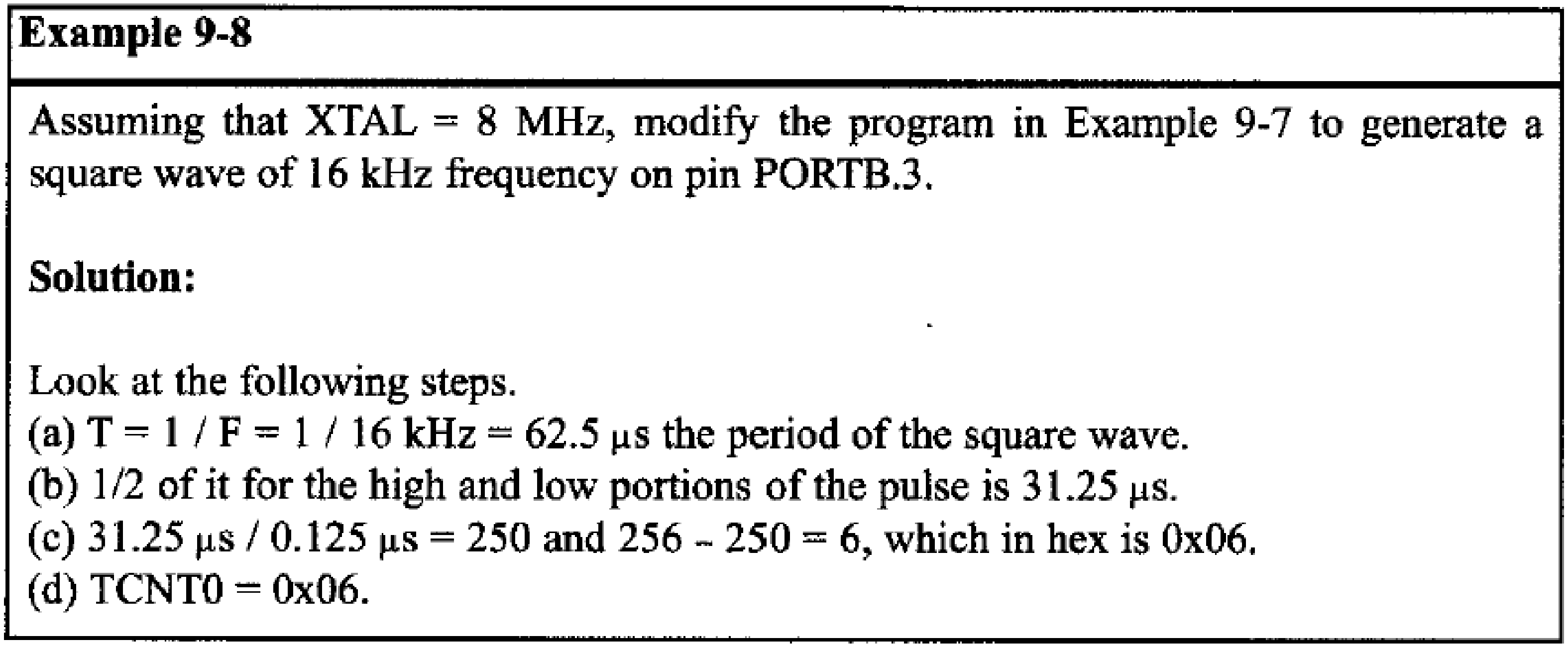
Assuming that XTAL = 8 MHz, write a program to generate a square wave with a period of 12.5 µs on pin PORTB.3.

Solution:

For a square wave with T = 12.5 µs we must have a time delay of 6.25 µs. Because XTAL = 8 MHz, the counter counts up every 0.125 µs. This means that we need 6.25 µs / 0.125 µs = 50 clocks. 256 – 50 = 206 = 0xCE. Therefore, we have TCNT0 = 0xCE.

```
TCCR0=0x01    //normal mode, no prescaling
TCNT0=0xCE
```

Timers in AVR



Example 9-8

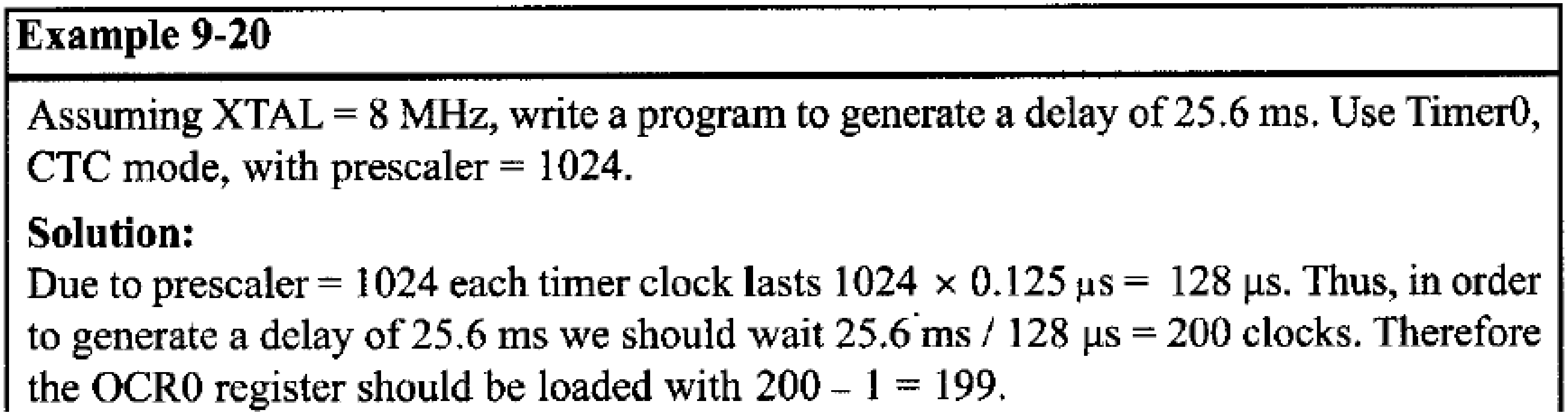
Assuming that XTAL = 8 MHz, modify the program in Example 9-7 to generate a square wave of 16 kHz frequency on pin PORTB.3.

Solution:

Look at the following steps.
(a) T = 1 / F = 1 / 16 kHz = 62.5 μs the period of the square wave.
(b) 1/2 of it for the high and low portions of the pulse is 31.25 μs.
(c) 31.25 μs / 0.125 μs = 250 and 256 - 250 = 6, which in hex is 0x06.
(d) TCNT0 = 0x06.

Timers in AVR-CTC mode

- Compare mode (clear timer o compare)
- Another way to count
  1. Increment TCNT at each clock cycle
  2. OCFn=1 when OCRn=TCNTn



Example 9-20

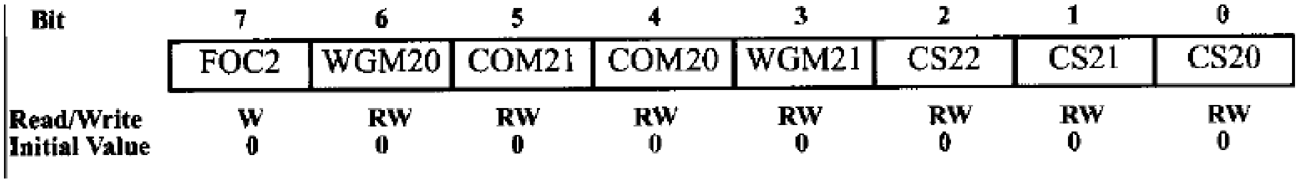
Assuming XTAL = 8 MHz, write a program to generate a delay of 25.6 ms. Use Timer0, CTC mode, with prescaler = 1024.

Solution:

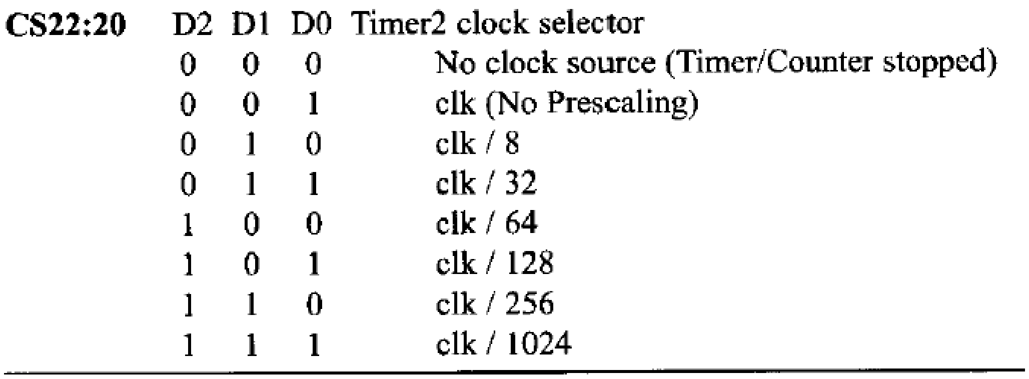
Due to prescaler = 1024 each timer clock lasts 1024 × 0.125 μs = 128 μs. Thus, in order to generate a delay of 25.6 ms we should wait 25.6 ms / 128 μs = 200 clocks. Therefore the OCR0 register should be loaded with 200 – 1 = 199.

Timer 2 in ATmega32

- Just like timer 0, but no external clock
  - Timer only
- TCCR2:



Bit	7	6	5	4	3	2	1	0
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20
Read/Write	W	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

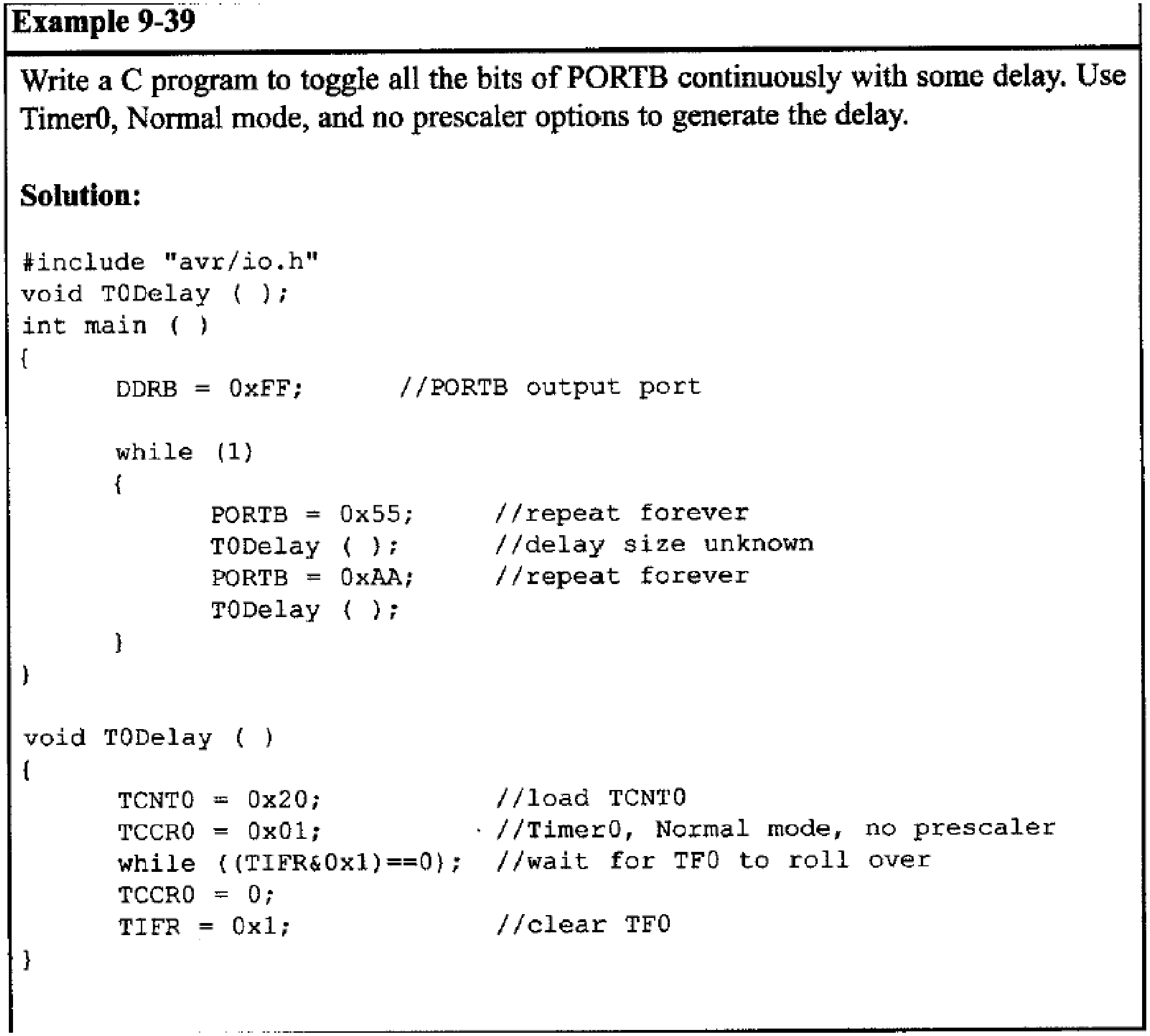


CS22:20	D2	D1	D0	Timer2 clock selector
	0	0	0	No clock source (Timer/Counter stopped)
	0	0	1	clk (No Prescaling)
	0	1	0	clk / 8
	0	1	1	clk / 32
	1	0	0	clk / 64
	1	0	1	clk / 128
	1	1	0	clk / 256
	1	1	1	clk / 1024

Timer programming in C

- We can use the register names in C codes:
  - TCNT0, TCNT1, TCNT2
  - TIFR0,…
  - TCCR0,…
  - ….

Timer programming in C



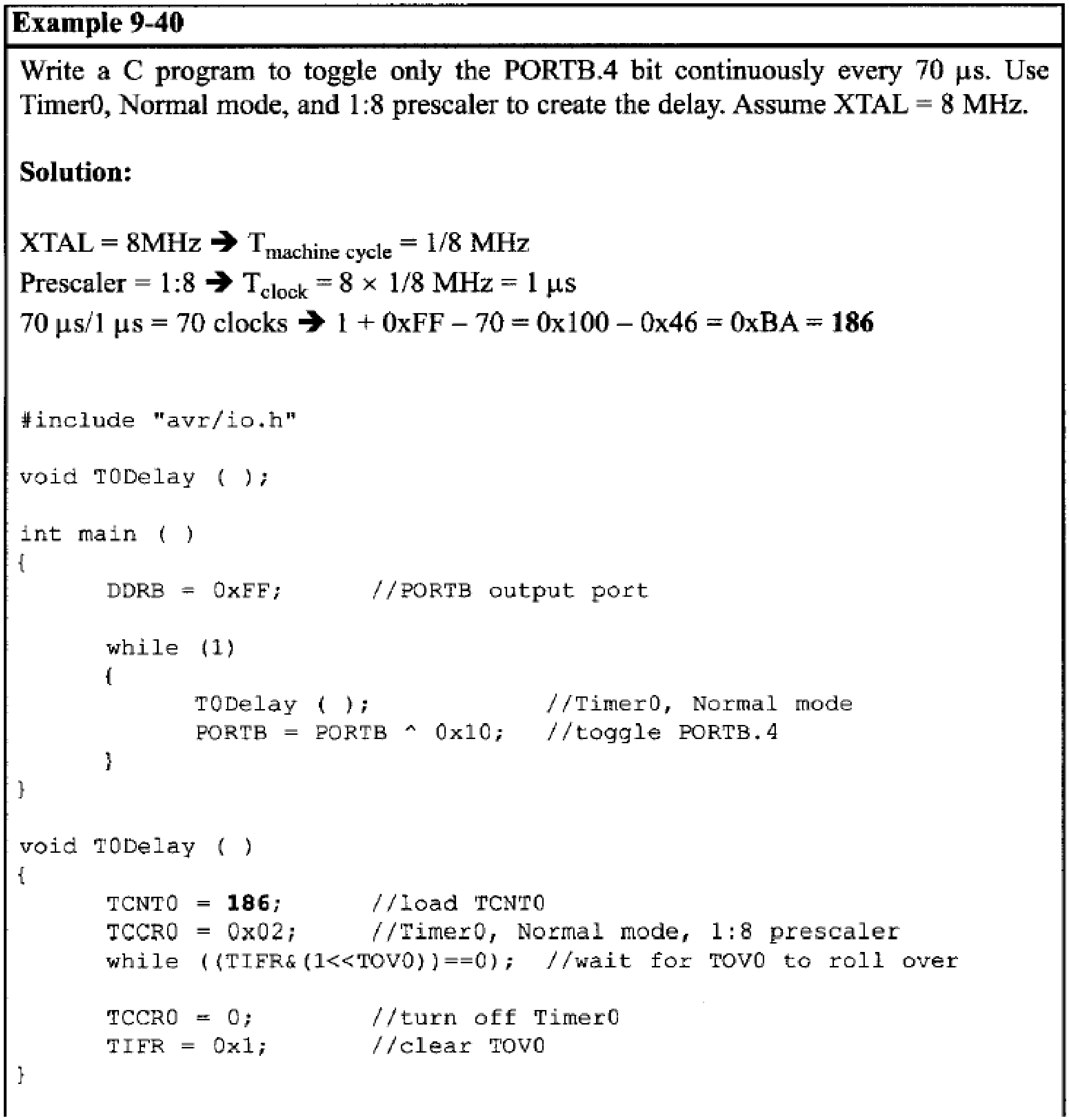
Example 9-39

Write a C program to toggle all the bits of PORTB continuously with some delay. Use Timer0, Normal mode, and no prescaler options to generate the delay.

Solution:

```
#include "avr/io.h"
void T0Delay ( );
int main ( )
{
      DDRB = 0xFF;        //PORTB output port

      while (1)
      {
            PORTB = 0x55;       //repeat forever
            T0Delay ( );        //delay size unknown
            PORTB = 0xAA;       //repeat forever
            T0Delay ( );
      }
}

void T0Delay ( )
{
      TCNT0 = 0x20;             //load TCNT0
      TCCR0 = 0x01;             //Timer0, Normal mode, no prescaler
      while ((TIFR&0x1)==0);    //wait for TF0 to roll over
      TCCR0 = 0;
      TIFR = 0x1;               //clear TF0
}
```

Timer programming in C



Example 9-40

Write a C program to toggle only the PORTB.4 bit continuously every 70 µs. Use Timer0, Normal mode, and 1:8 prescaler to create the delay. Assume XTAL = 8 MHz.

Solution:

XTAL = 8MHz ➔ $T_{\text{machine cycle}}$ = 1/8 MHz

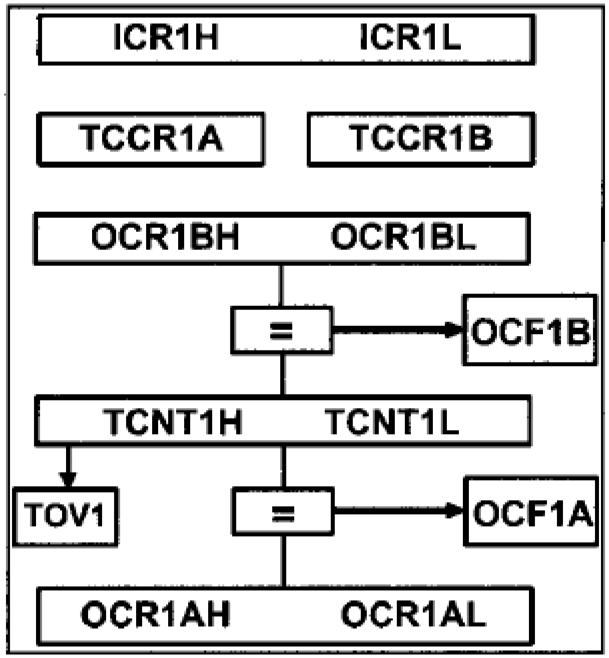
Prescaler = 1:8 ➔ T_{clock} = 8 × 1/8 MHz = 1 µs

70 µs/1 µs = 70 clocks ➔ 1 + 0xFF – 70 = 0x100 – 0x46 = 0xBA = **186**

```
#include "avr/io.h"

void T0Delay ( );

int main ( )
{
      DDRB = 0xFF;       //PORTB output port

      while (1)
      {
            T0Delay ( );            //Timer0, Normal mode
            PORTB = PORTB ^ 0x10;   //toggle PORTB.4
      }
}

void T0Delay ( )
{
      TCNT0 = 186;       //load TCNT0
      TCCR0 = 0x02;      //Timer0, Normal mode, 1:8 prescaler
      while ((TIFR&(1<<TOV0))==0);  //wait for TOV0 to roll over

      TCCR0 = 0;         //turn off Timer0
      TIFR = 0x1;        //clear TOV0
}
```

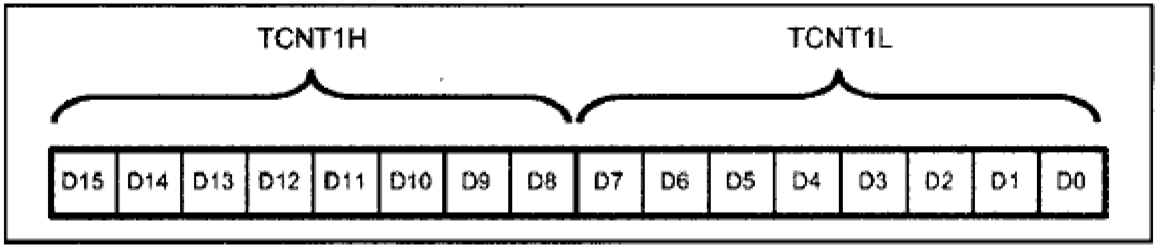
Timer 1

- **16-bit counter/timer**
- **TCNT1L and TCNT1H**
- **2 8-bit registers to control timer 1**
  - TCCR1L and TCCR1H
- **2 registers in compare mode**
  - OCR1A and OCR1B



Timer 1

TCNT1



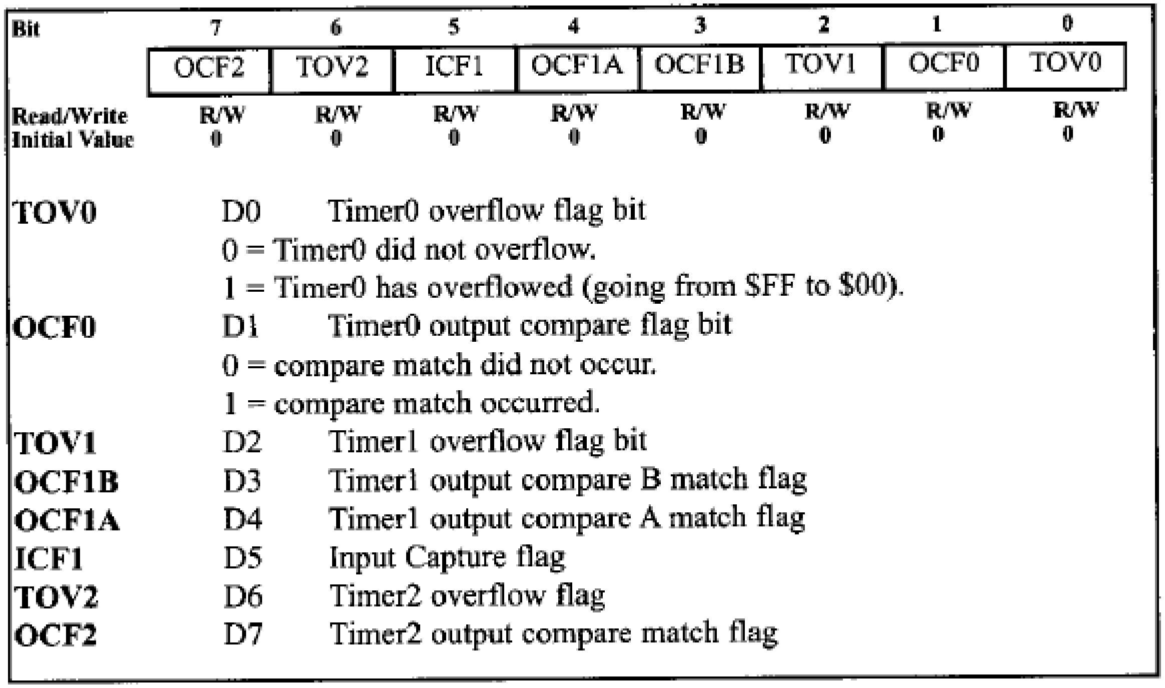
TCNT1H								TCNT1L							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

3 flags in TIFR:

TOV1

and

OCF1A-OCF1B



Bit	7	6	5	4	3	2	1	0
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
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

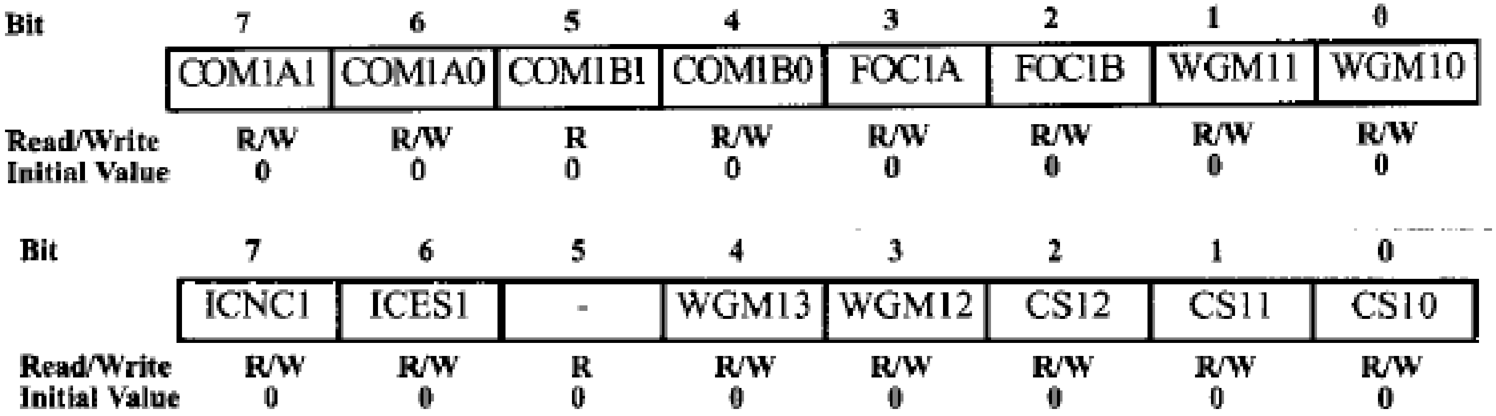
TOV0 D0 Timer0 overflow flag bit
0 = Timer0 did not overflow.
1 = Timer0 has overflowed (going from $FF to $00).

OCF0 D1 Timer0 output compare flag bit
0 = compare match did not occur.
1 = compare match occurred.

TOV1	D2	Timer1 overflow flag bit
OCF1B	D3	Timer1 output compare B match flag
OCF1A	D4	Timer1 output compare A match flag
ICF1	D5	Input Capture flag
TOV2	D6	Timer2 overflow flag
OCF2	D7	Timer2 output compare match flag

Timer 1 control registers

- **2 registers**
- **Plenty of operation modes**



TCCR1A

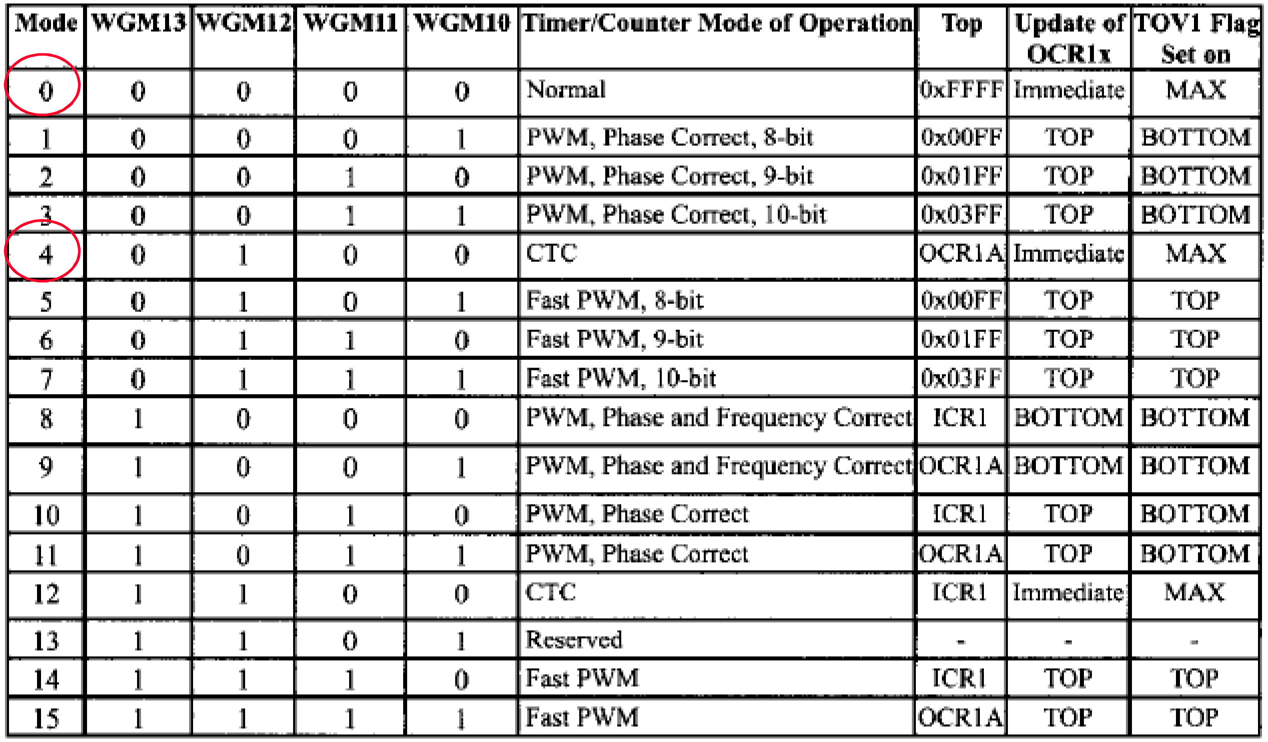
Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TCCR1B

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
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

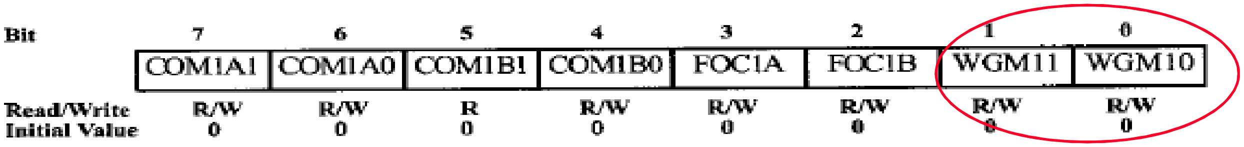
Timer 1 control registers

In this course, we focus on modes 0 and 4



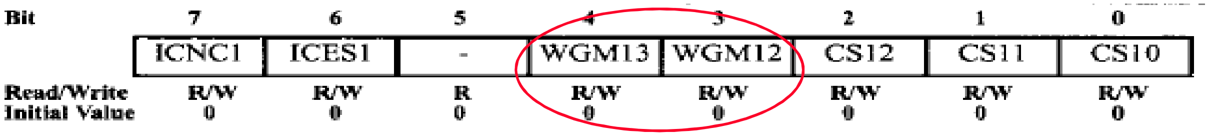
Mode	WGM13	WGM12	WGM11	WGM10	Timer/Counter Mode of Operation	Top	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	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

TCCR1A



Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
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

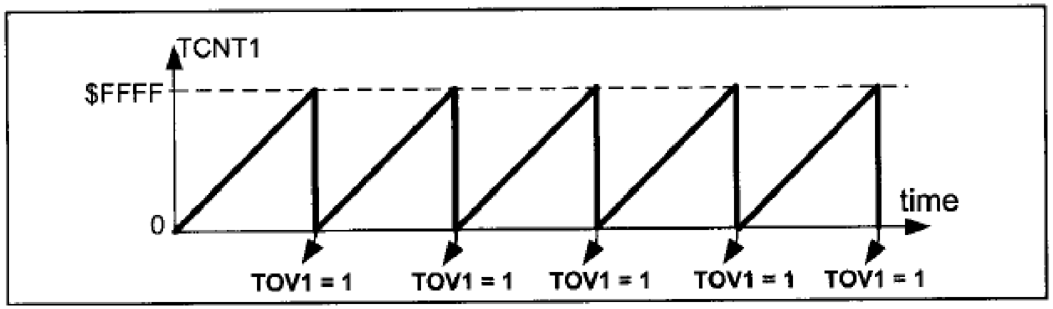
TCCR1B



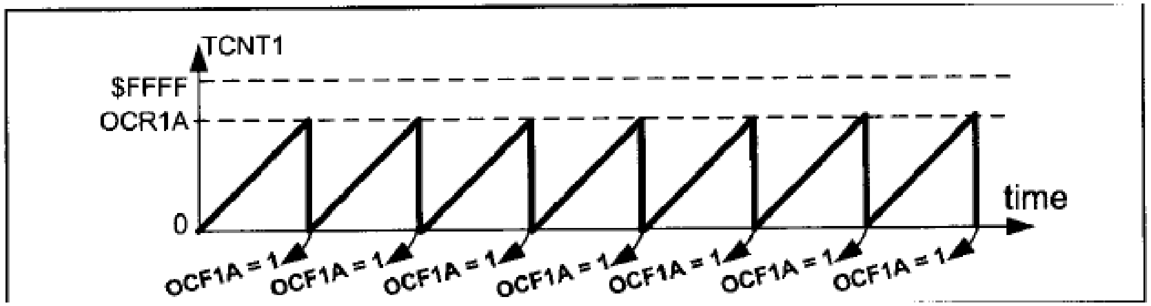
Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Timer 1 modes

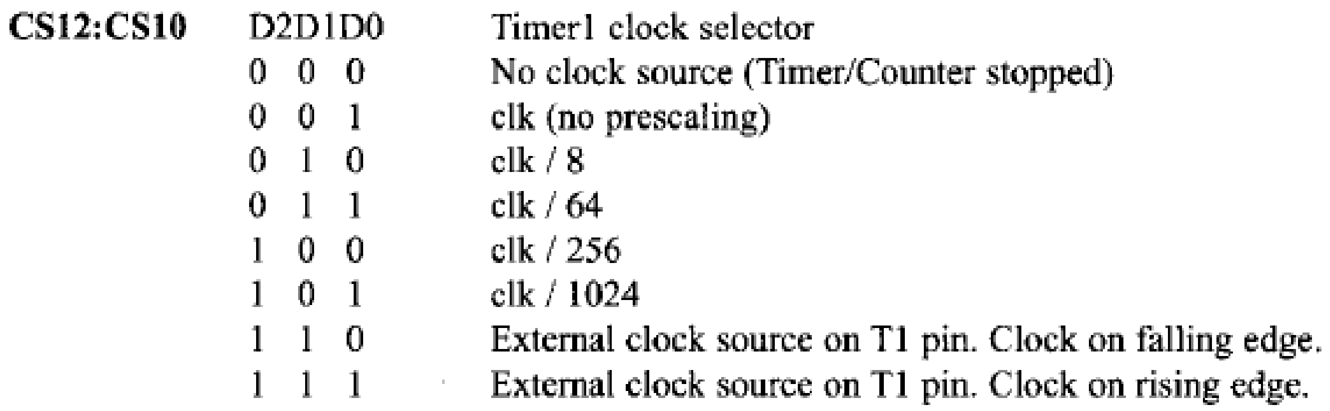
- 16 modes, we use 2 modes in this chapter:
- Normal mode:



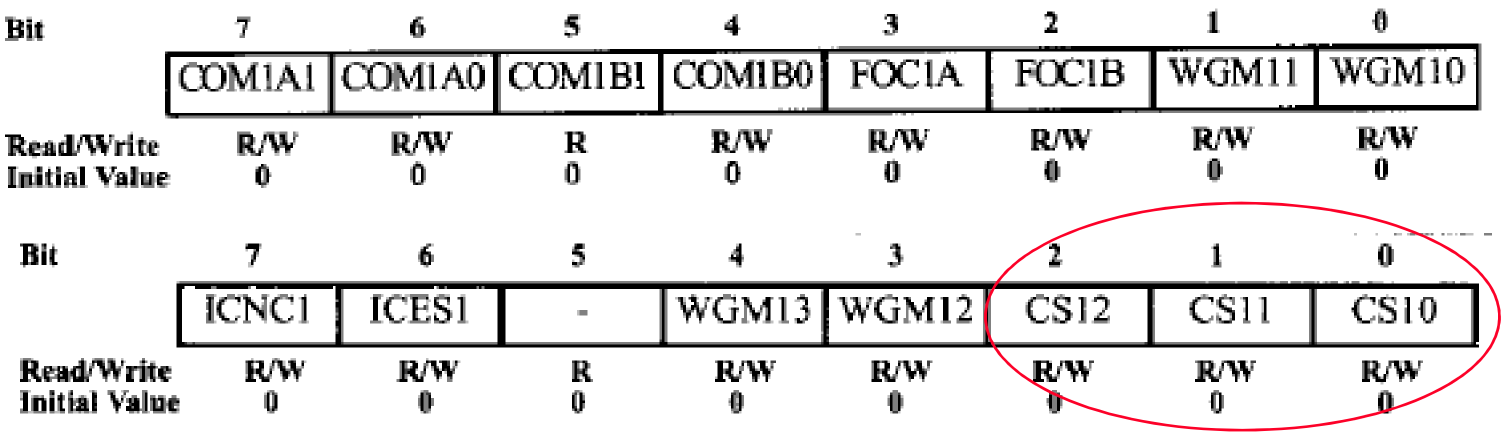
- CTC mode



Timer 1 control registers



CS12:CS10	D2D1D0	Timer1 clock selector
	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.



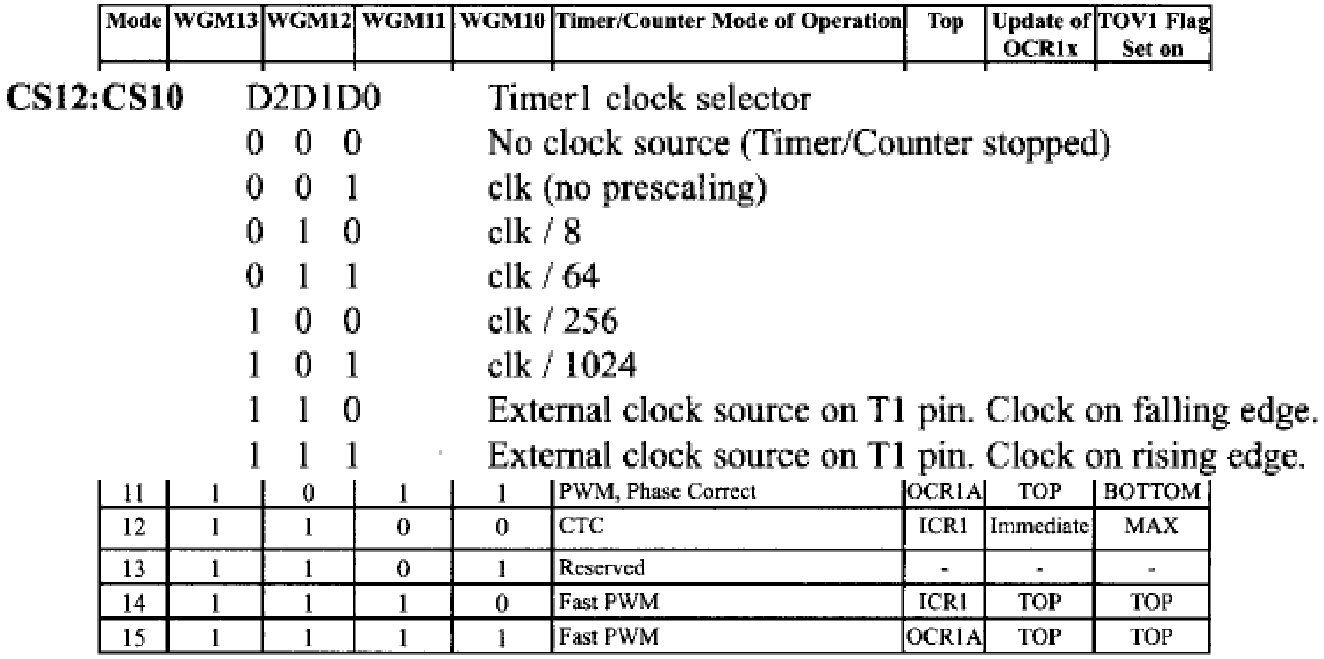
TCCR1A

Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
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

TCCR1B

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
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

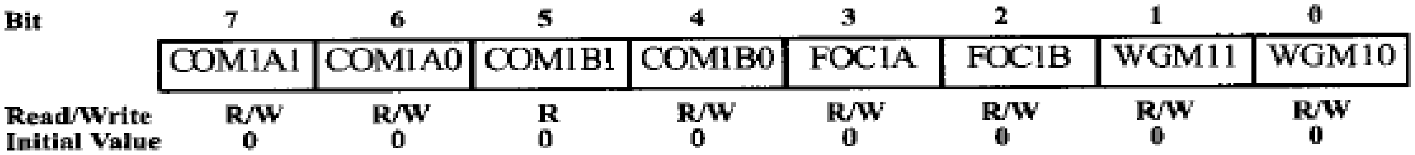
Timer 1 control registers



Mode	WGM13	WGM12	WGM11	WGM10	Timer/Counter Mode of Operation	Top	Update of OCR1x	TOV1 Flag Set on
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

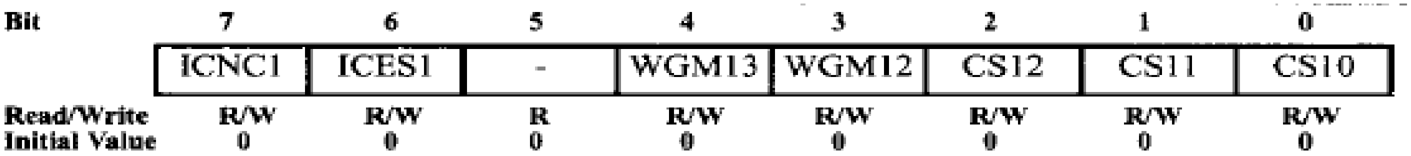
CS12:CS10	D2D1D0	Timer1 clock selector
	0 0 0	No clock source (Timer/Counter stopped)
	0 0 1	clk (no prescaling)
	0 1 0	clk / 8
	0 1 1	clk / 64
	1 0 0	clk / 256
	1 0 1	clk / 1024
	1 1 0	External clock source on T1 pin. Clock on falling edge.
	1 1 1	External clock source on T1 pin. Clock on rising edge.

TCCR1A



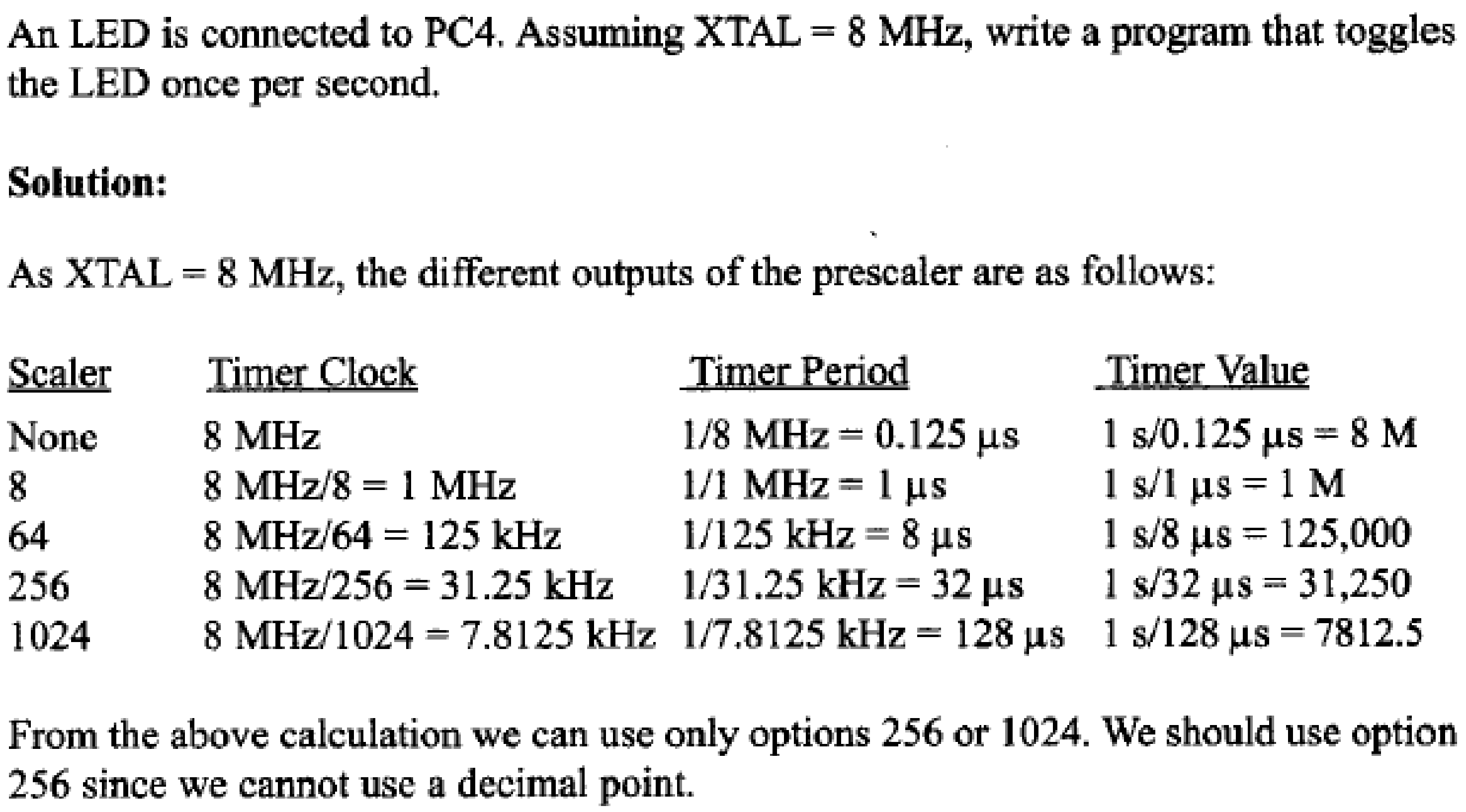
Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TCCR1B



Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
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 1 programming



An LED is connected to PC4. Assuming XTAL = 8 MHz, write a program that toggles the LED once per second.

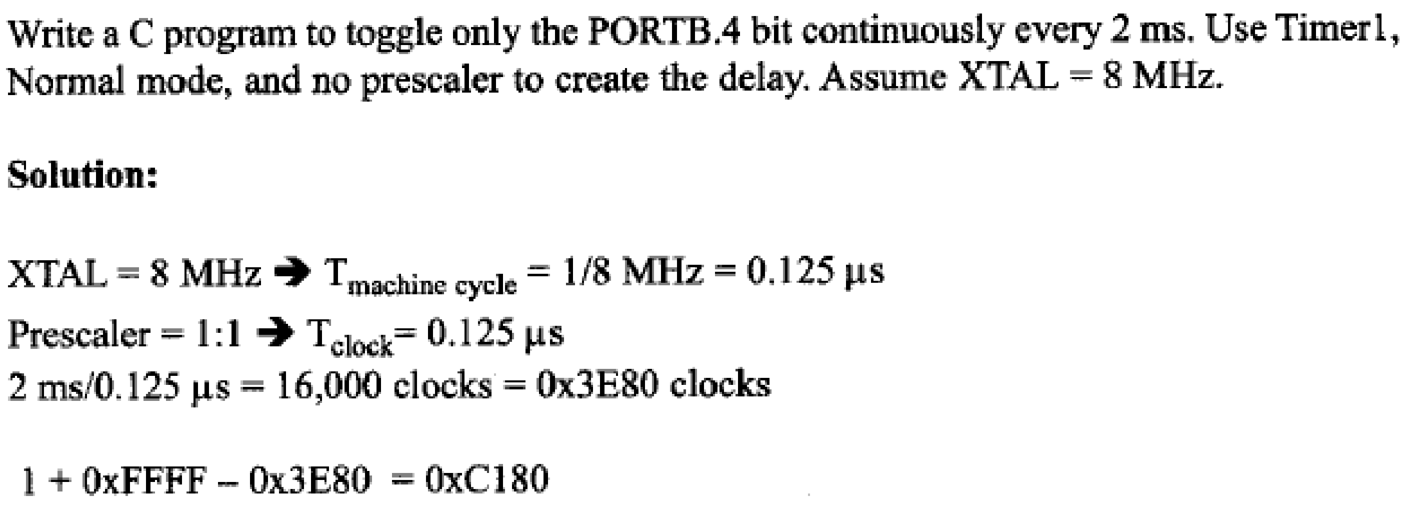
Solution:

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

Scaler	Timer Clock	Timer Period	Timer Value
None	8 MHz	1/8 MHz = 0.125 μs	1 s/0.125 μs = 8 M
8	8 MHz/8 = 1 MHz	1/1 MHz = 1 μs	1 s/1 μs = 1 M
64	8 MHz/64 = 125 kHz	1/125 kHz = 8 μs	1 s/8 μs = 125,000
256	8 MHz/256 = 31.25 kHz	1/31.25 kHz = 32 μs	1 s/32 μs = 31,250
1024	8 MHz/1024 = 7.8125 kHz	1/7.8125 kHz = 128 μs	1 s/128 μs = 7812.5

From the above calculation we can use only options 256 or 1024. We should use option 256 since we cannot use a decimal point.

Timer 1 programming



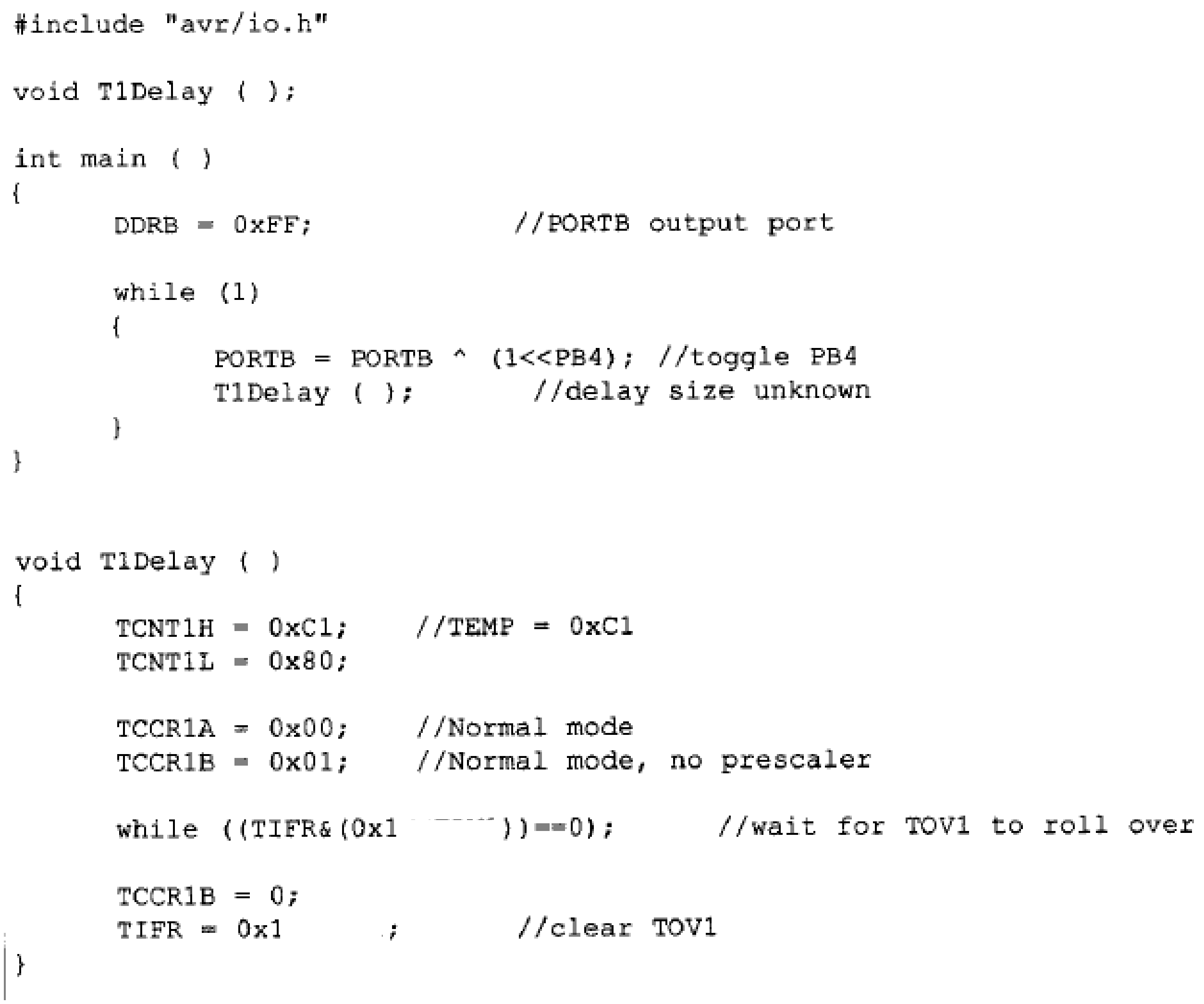
Write a C program to toggle only the PORTB.4 bit continuously every 2 ms. Use Timer1, Normal mode, and no prescaler to create the delay. Assume XTAL = 8 MHz.

Solution:

XTAL = 8 MHz ➔ $T_{\text{machine cycle}}$ = 1/8 MHz = 0.125 µs
Prescaler = 1:1 ➔ T_{clock}= 0.125 µs
2 ms/0.125 µs = 16,000 clocks = 0x3E80 clocks

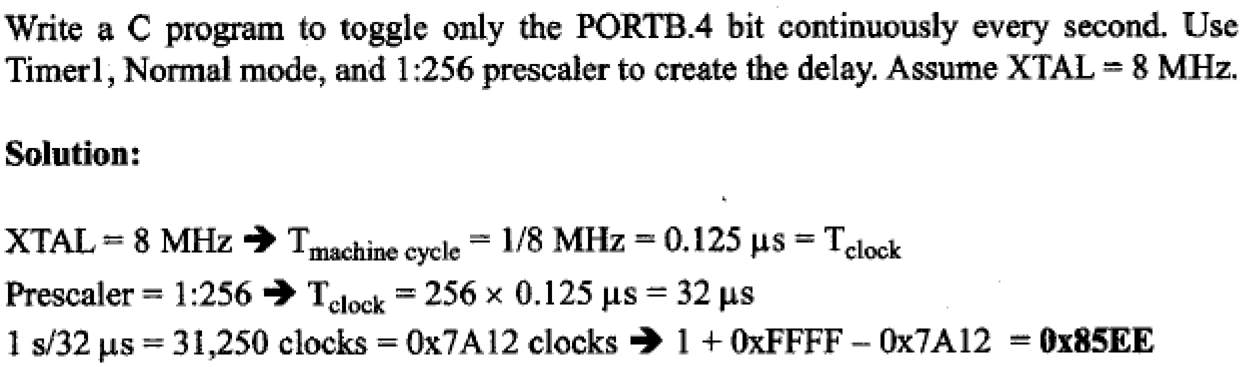
1 + 0xFFFF – 0x3E80 = 0xC180

Timer 1 programming



```
#include "avr/io.h"

void T1Delay ( );

int main ( )
{
        DDRB = 0xFF;                //PORTB output port

        while (1)
        {
                PORTB = PORTB ^ (1<<PB4); //toggle PB4
                T1Delay ( );          //delay size unknown
        }
}


void T1Delay ( )
{
        TCNT1H = 0xC1;     //TEMP = 0xC1
        TCNT1L = 0x80;

        TCCR1A = 0x00;     //Normal mode
        TCCR1B = 0x01;     //Normal mode, no prescaler

        while ((TIFR&(0x1        ))==0);      //wait for TOV1 to roll over

        TCCR1B = 0;
        TIFR = 0x1       ;        //clear TOV1
}
```

Timer 1 programming



Write a C program to toggle only the PORTB.4 bit continuously every second. Use Timer1, Normal mode, and 1:256 prescaler to create the delay. Assume XTAL = 8 MHz.

Solution:

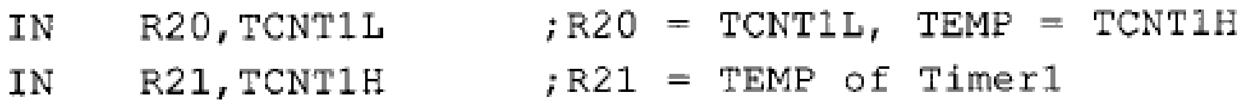
XTAL = 8 MHz ➔ $T_{\text{machine cycle}}$ = 1/8 MHz = 0.125 μs = T_{clock}

Prescaler = 1:256 ➔ T_{clock} = 256 × 0.125 μs = 32 μs

1 s/32 μs = 31,250 clocks = 0x7A12 clocks ➔ 1 + 0xFFFF – 0x7A12 = **0x85EE**

Accessing 16-bit registers in AVR

- **TCNT1=0x05ff, we want to save the content of TCNT1 in R20 and R21**
- **Cannot read TCNT in one cycle**
  - AVR is a 8-bit machine



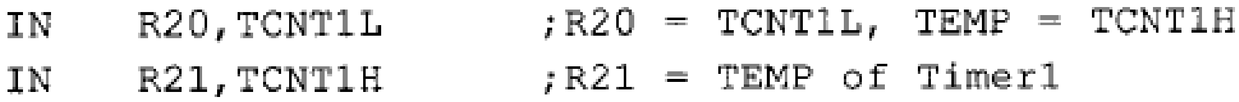
```
IN    R20,TCNT1L       ;R20 = TCNT1L, TEMP = TCNT1H
IN    R21,TCNT1H       ;R21 = TEMP of Timer1
```

- **Read TCNT1L (0xff) at t0, at the same cycle occurs TCNT=0x0600**
- **Read TCNT1H (0x06)**
- **The content is detected as 0x06ff instead of the correct value 0x05ff**

Accessing 16-bit registers in AVR

- **Solution:**
  - **AVR buffers the high byte when the lower byte is read**
  - **When the higher byte is read, the buffered value is used**

  → first read the lowest byte and then the higher byte

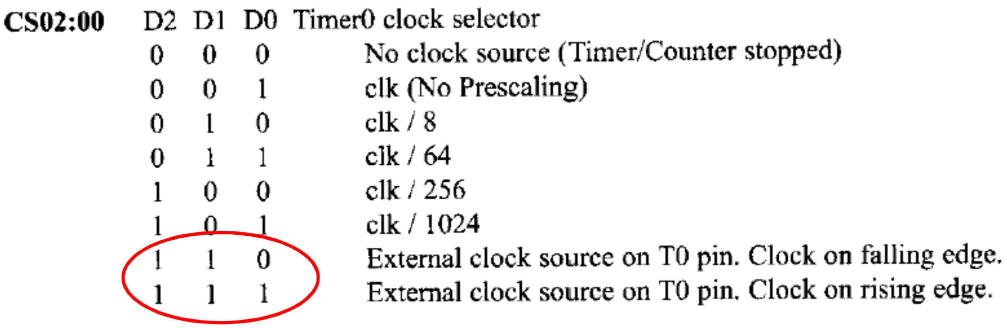


```
IN    R20,TCNT1L        ;R20 = TCNT1L, TEMP = TCNT1H
IN    R21,TCNT1H        ;R21 = TEMP of Timer1
```

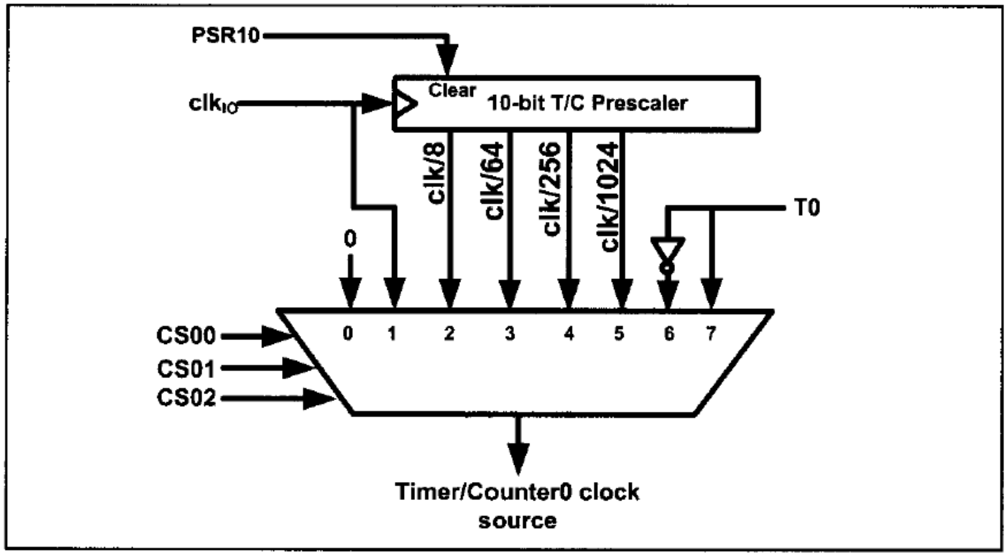
Counters in AVR

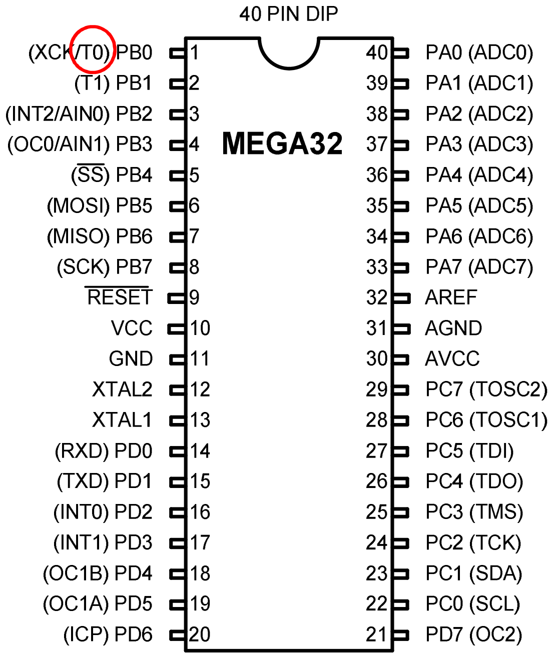
- To count external events

Counter programming in AVR



CS02:00	D2	D1	D0	Timer0 clock selector
	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.





Counter programming in AVR

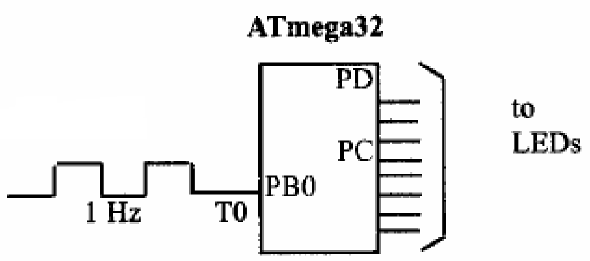
- Configure T0 (PB0) or T1 (PB1) as input
- Set the other registers as in timers

Counter

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

Solution:

```
#include "avr/io.h"

int main ( )
{
        PORTB = 0x01;           //activate pull-up of PB0
        DDRC = 0xFF;            //PORTC as output
        DDRD = 0xFF;            //PORTD as output

        TCCR0 = 0x06;           //output clock source
        TCNT0 = 0x00;

        while (1)
        {
                do
                {
                        PORTC = TCNT0;
                }while((TIFR&(0x1<<TOV0))==0);//wait for TOV0 to roll over

                TIFR = 0x1<<TOV0;       //clear TOV0
                PORTD ++;               //increment PORTD
        }
}
```


ATmega32
PD
PC
PB0
1 Hz
T0
to
LEDs