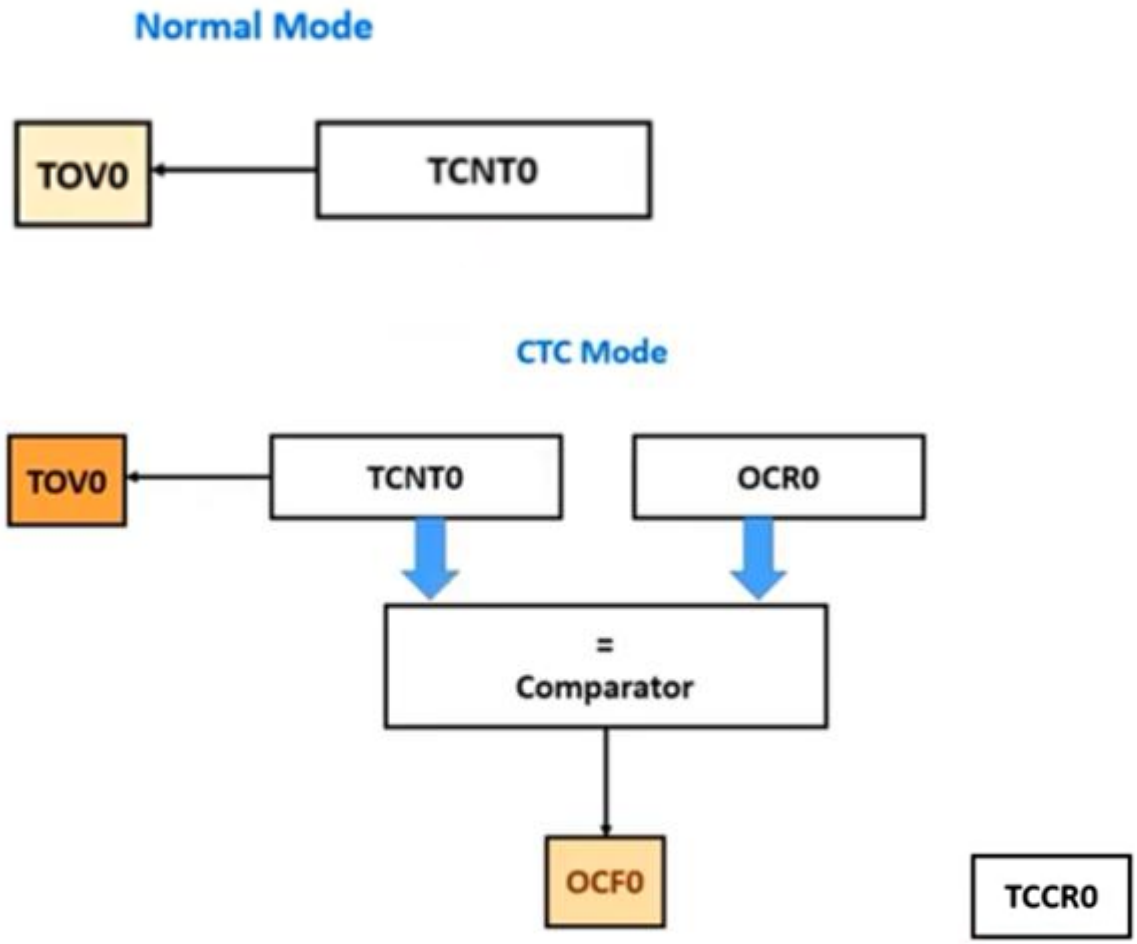
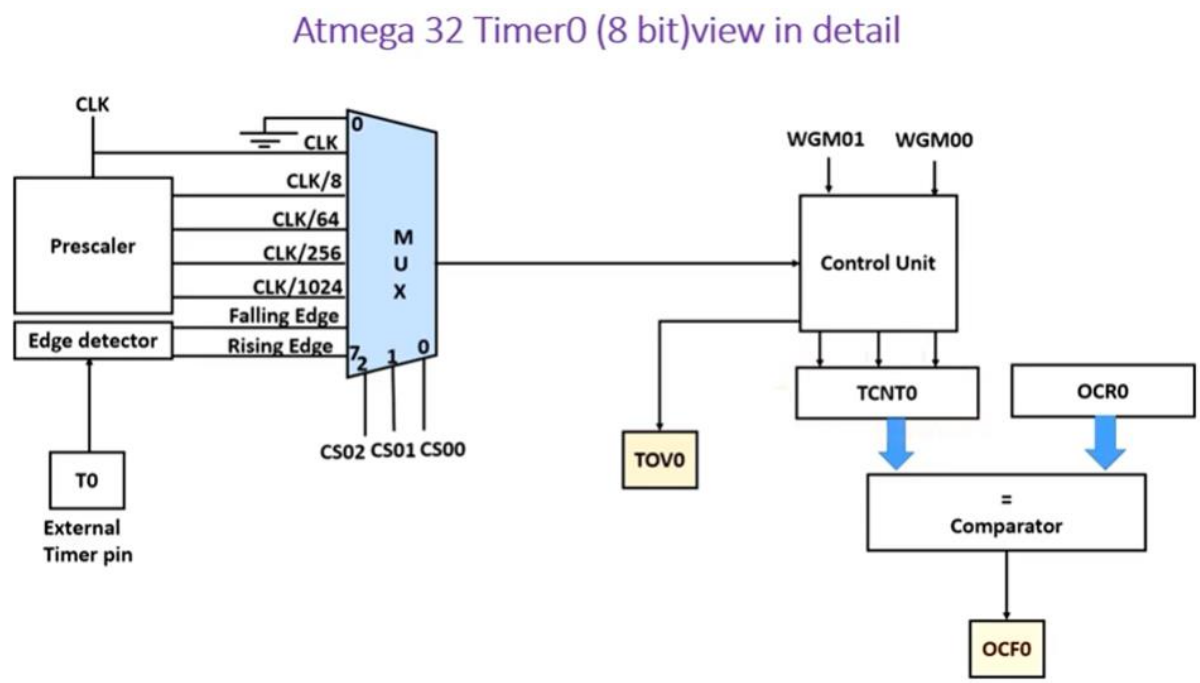
Programming the Timers

Timer 0 Programming

Atmega32 Timer 0 Internal View

Well, timer 0 and 2 are 8 bits meaning it could count from **0 to 255**.





Register Description

TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bits 7:6 – COM0A1:0: Compare Match Output A Mode

These bits control the output compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM02:0 bit setting. Table 14-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 14-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

Table 14-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Bits 3, 2 – Res: Reserved Bits

These bits are reserved bits in the Atmel® ATmega328P and will always read as zero.

Bits 1:0 – WGM01:0: Waveform Generation Mode

Combined with the WGM02 bit found in the TCCR0B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see

Table 14-8. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes.

Table 14-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – FOC0A: Force Output Compare A

The FOC0A bit is only active when the WGM bits specify a non-PWM mode.
However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written when operating
in PWM mode. When writing a logical one to the FOC0A bit, an immediate compare match is forced on the waveform generation unit. The OC0A output is changed according to its COM0A1:0 bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A1:0 bits that determines the effect of the forced compare.
A FOC0A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0A as TOP.

The FOC0A bit is always read as zero.

• Bit 6 – FOC0B: Force Output Compare B

The FOC0B bit is only active when the WGM bits specify a non-PWM mode.
However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0B bit, an immediate compare match is forced on the waveform generation unit. The OC0B output is changed according to its COM0B1:0 bits setting. Note that the FOC0B bit is implemented as a strobe. Therefore it is the value present in the COM0B1:0 bits that determines the effect of the forced compare.
A FOC0B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0B as TOP.

The FOC0B bit is always read as zero.

• Bits 5:4 – Res: Reserved Bits

These bits are reserved bits in the ATmega328P and will always read as zero.

• Bit 3 – WGM02: Waveform Generation Mode

See the description in the Section 14.9.1 “TCCR0A – Timer/Counter Control Register A” on page 84.

• **Bits 2:0 – CS02:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter.

Table 14-9. Clock Select Bit Description

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.

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
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	

The Timer/Counter register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0x registers.

OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	OCR0A[7:0]								OCR0A
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	

The output compare register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0A pin.

OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x28 (0x48)	OCR0B[7:0]								OCR0B
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	

The output compare register B contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0B pin.

TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bits 7…..3 – Res: Reserved Bits

These bits are reserved bits in the Atmel® ATmega328P and will always read as zero.

• Bit 2 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable

When the OCIE0B bit is written to one, and the I-bit in the status register is set, the Timer/Counter compare match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter interrupt flag register – TIFR0.

• Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable

When the OCIE0A bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

• Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable

When the TOIE0 bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bits 7..3 – Res: Reserved Bits

These bits are reserved bits in the Atmel ATmega328P and will always read as zero.

• Bit 2 – OCF0B: Timer/Counter 0 Output Compare B Match Flag

The OCF0B bit is set when a compare match occurs between the Timer/Counter and the data in OCR0B – output compare register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter compare B match interrupt enable), and OCF0B are set, the Timer/Counter compare match interrupt is executed.

• Bit 1 – OCF0A: Timer/Counter 0 Output Compare A Match Flag

The OCF0A bit is set when a compare match occurs between the Timer/Counter0 and the data in OCR0A – output compare register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 compare match interrupt enable), and OCF0A are set, the Timer/Counter0 compare match interrupt is executed.

• Bit 0 – TOV0: Timer/Counter0 Overflow Flag

The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared

by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 overflow interrupt enable), and TOV0 are set, the Timer/Counter0 overflow interrupt is executed. The setting of this flag is dependent of the WGM02:0 bit setting.

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
}
```

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 ➔ $T_{machine\ cycle}$ = 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
}
```

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-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 Assume that XTAL=8MHz. write a program to toggle at 25.6 ms. Use time 0 CTC mode with prescalar = 1024.

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.

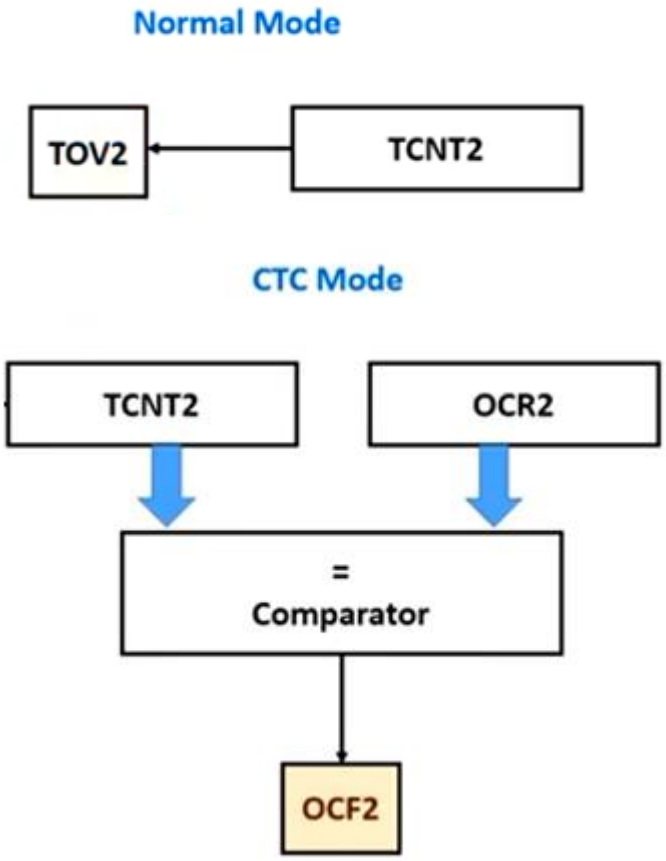
Notice that the comparator checks for equality; thus, if we load OCR0register with a value smaller than TCNT0's value, the counter will miss the compare match and will count up until it reaches the maximum value of 0xFF and rolls over. This causes a big delay and is not desirable in many cases.Example

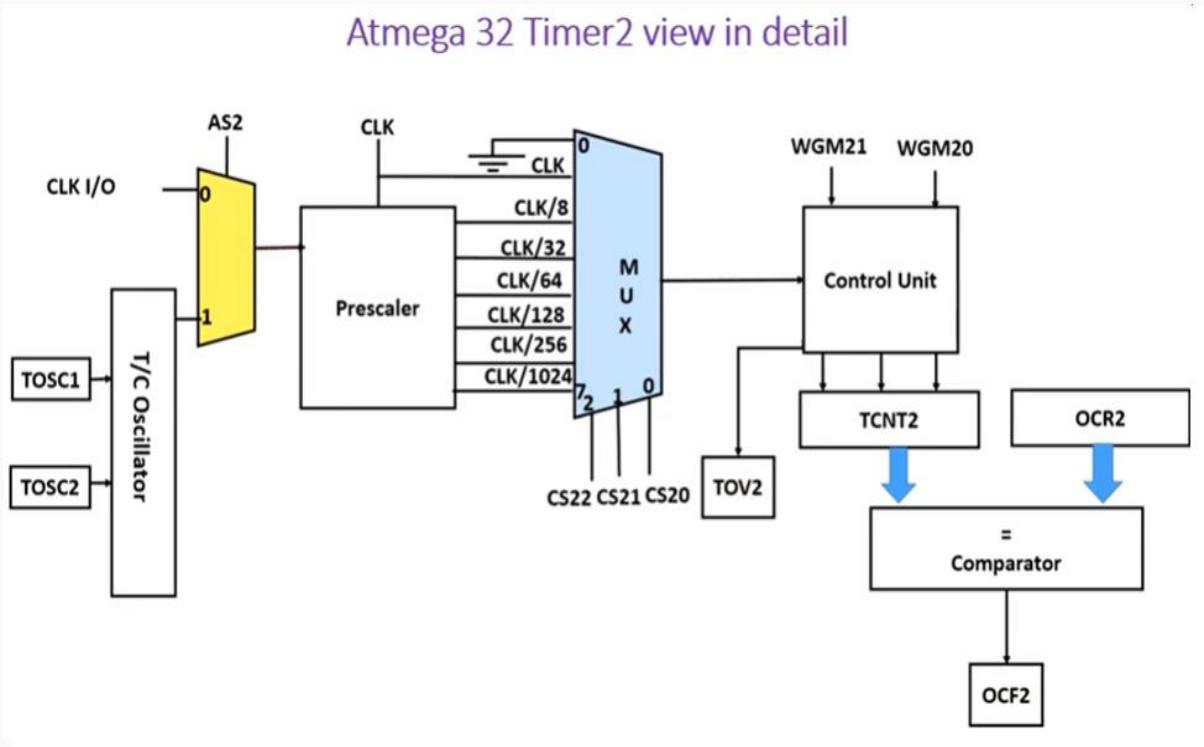
```
#include <avr/io.h>
void T0Delay ();
int main ()
{
DDRB = 0xFF;      // PORTB output port
while(1)
{              //repeat forever
PORTB = 0x55;
T0Delay();    //delay size unknown
```

```
PORTB = 0xAA;
T0Delay();
}
}
void T0Delay()
{
TCNT0 = 0;   // timer starts from 00
OCR0 = 199;  // Output Compare Register has value 199
TCCR0A= 0x02;
TCCR0B= 0x05;  /*Run Timer0,(CTC)Clear Timer on Compare Match,// 1:1024
Prescaler */

while((TIFR & 0x02)==0);    // wait for OCF0 to roll over
TCCRB = 0;// Stop Timer
TIFR = 0x02;//clear TF0
}
```

Timer 2 Programming

Atmega32 Timer 2 Internal View





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

- Timer/Counter2 is the preferred timer among programmers for short time delays because, its prescaler has the greatest number of options . It can run in Normal mode or CTC modes.
- In normal mode TOV2 can generate a Overflow interrupt. In order to activate the timer1 overflow interrupts you need to SET(1) the TOIE1 bit within the TIMSK2 register.

  In CTC mode OCIF2 can generate an interrupt when it detects a compare match. In order to activate the timer1 CTC interrupt SET(1) the OCF2 bit within the TIMSK register.

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20

Timer/Counter Control Register 2

MODE	WGM21	WGM20	DESCRIPTION	TOP
0	0	0	Normal	0xFF
1	0	1	PWM Phase Corrected	
2	1	0	CTC	OCR2
3	1	1	Fast PWM	

Waveform Generator Mode bits

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR2B	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20

Timer/Counter Control Register 2

CS22	CS21	CS20	DESCRIPTION
0	0	0	Timer/Counter2 Disabled
0	0	1	No Prescaling
0	1	0	Clock / 8
0	1	1	Clock / 32
1	0	0	Clock / 64
1	0	1	Clock / 128
1	1	0	Clock / 256
1	1	1	Clock / 1024

CS bits

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TIMSK2	-	-	-	-	-	OCIE2B	OCIE2A	TOIE2

Timer/Counter Interrupt Mask Register 2

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2

Timer/Counter Interrupt Flag Register 2

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCNT2								

Timer/Counter Register 2 (stores the counter value)

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
OCR2								

Output Compare Register 2 (stores the compare value)

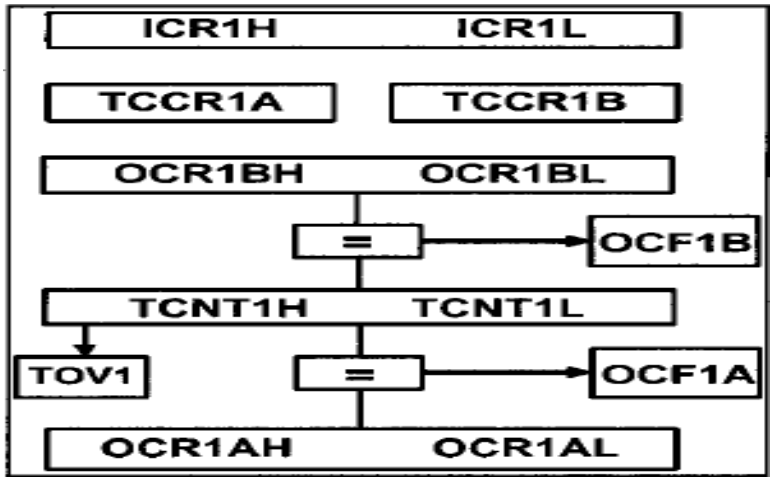
(*EXPLAIN the register same as timer 0*)

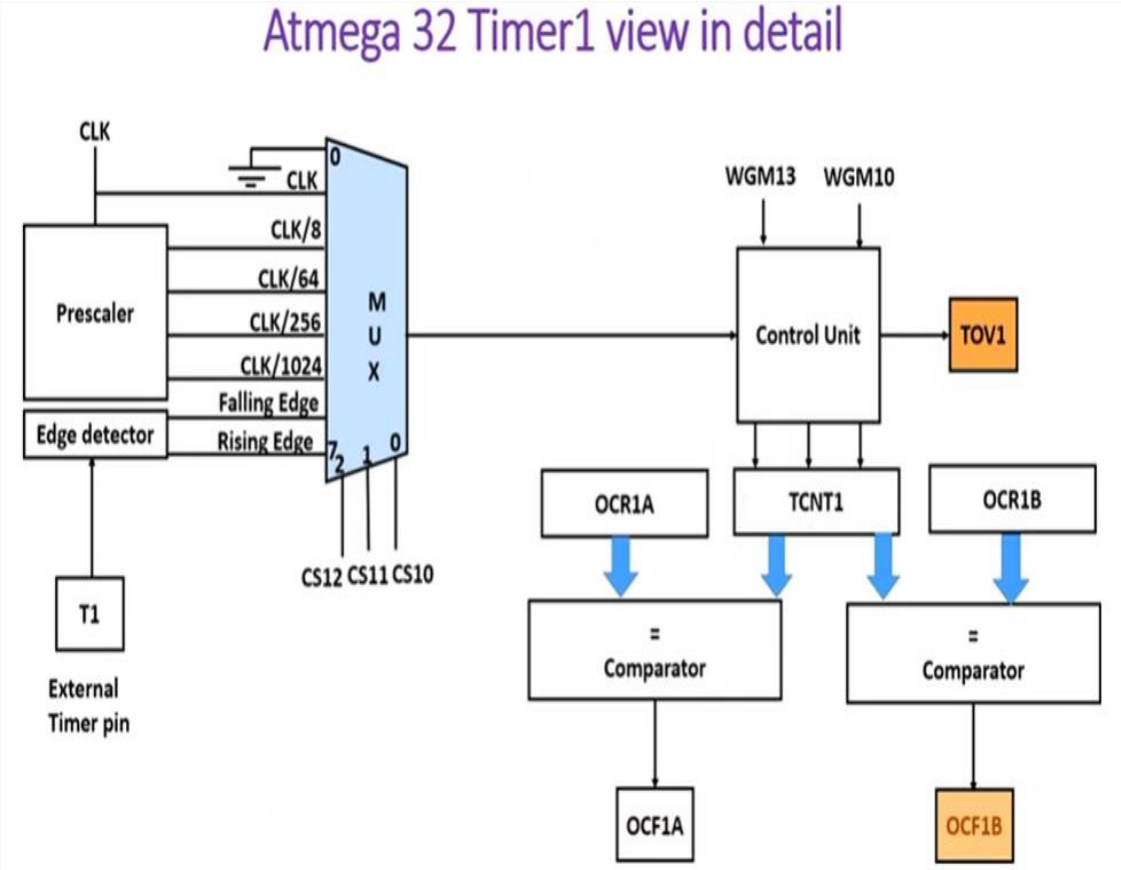
Program

```
// this code sets up timer2 for a 250us  @ 16Mhz Clock


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


int main(void)
{
    OCR2A = 62;

    TCCR2A |= (1 << WGM21);
    // Set to CTC Mode

    TIMSK2 |= (1 << OCIE2A);
    //Set interrupt on compare match

    TCCR2B |= (1 << CS21);
    // set prescaler to 64 and starts PWM

    sei();
    // enable interrupts


    while (1)
    {
        // Main loop
    }
}

ISR (TIMER2_COMPA_vect)
{
    // action to be done every 250 usec
}
```

Timer1 Programming

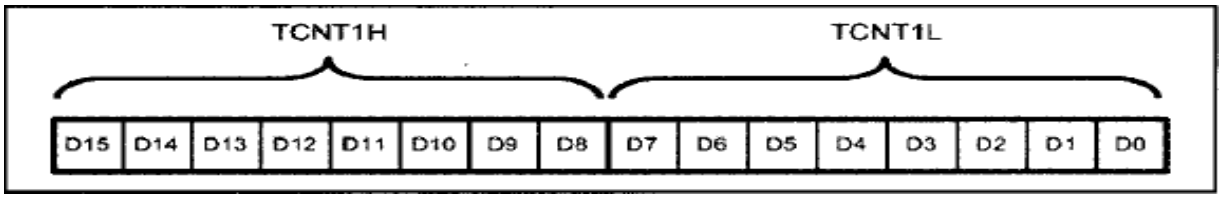
- **16-bit counter/timer**
- **TCNT1L and TCNT1H**
- **2 8-bit registers to control timer 1**
  - TCCR1L and TCCR1H
- **2 registers in compare mode**
  - OCR1A and OCR1B


ICR1H
ICR1L
TCCR1A
TCCR1B
OCR1BH
OCR1BL
=
OCF1B
TCNT1H
TCNT1L
TOV1
=
OCF1A
OCR1AH
OCR1AL


Atmega 32 Timer1 view in detail
CLK
CLK
CLK/8
CLK/64
CLK/256
CLK/1024
Falling Edge
Rising Edge
Prescaler
Edge detector
T1
External
Timer pin
MUX
0
7
2
1
0
CS12 CS11 CS10
WGM13
WGM10
Control Unit
TOV1
OCR1A
TCNT1
OCR1B
=
Comparator
=
Comparator
OCF1A
OCF1B

Timer 1 control registers

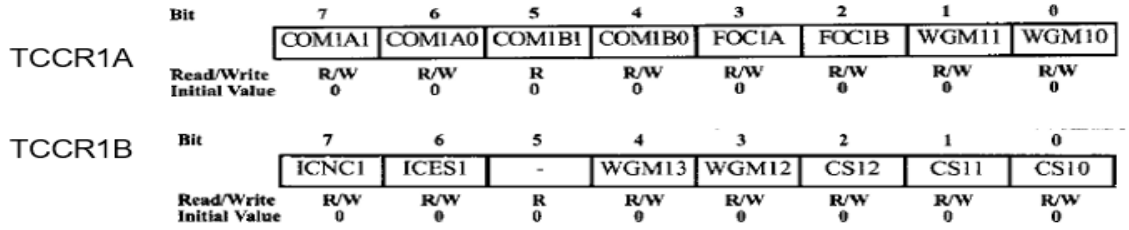
TCNT1



First register we need to change is the prescalar. To control the timers we have two main registers, the TCCR **A** and the TCCR **B**, each with the number of the timer. So for timer 1 we have TCCR 1A and TCCR 1B. TCCRA register is for controlling the PWM mode so for timer 1 we can control the OC1A which is related to the PWM signal on pins 9 and 10. For our interrupt examples we don't need this register but we do need to **set all its bits to 0** since Arduino by default has them set to 1. So, setting all these to 0 will disable the PWM signal on pin 9 and 10.

Now, we need to change the TCCRB register. Here, all we care are the **first 3 bits** which are used to define the prescaar value. As you can see in the table below, using the CS10 CS11 and CS12 bits, we can disable the prescalar or set it to 1, divided by 8, 64, 256, 1024 or even use an external clock source.

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



Bit 7 – ICNC1: Input Capture Noise Canceller

Setting this bit (to one) activates the Input Capture Noise Canceller after which the input from the Input Capture pin (ICP1) is filtered.

• Bit 6 – ICES1: Input Capture Edge Select

This bit selects which edge on the Input Capture pin (ICP1) is used to trigger a capture event.

Bit 5 – Reserved Bit

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCR1B is written.

• Bit 4:3 – WGM13:2: Waveform Generation Mode

See TCCR1A Register description.

• Bit 2:0 – CS12:0: Clock Select

The three Clock Select bits select the clock source to be used by the Timer/Counter

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

TIFR1 – Timer/Counter1 Interrupt Flag Register

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

• Bit 7, 6 – Res: Reserved Bits

These bits are unused bits in the Atmel® ATmega328P, and will always read as zero.

• Bit 5 – ICF1: Timer/Counter1, Input Capture Flag

This flag is set when a capture event occurs on the ICP1 pin. When the input capture register (ICR1) is set by the WGM13:0

to be used as the TOP value, the ICF1 flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the input capture interrupt vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

• Bit 4, 3 – Res: Reserved Bits

These bits are unused bits in the Atmel ATmega328P, and will always read as zero.

• Bit 2 – OCF1B: Timer/Counter1, Output Compare B Match Flag

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the output compare register B (OCR1B).

Note that a forced output compare (FOC1B) strobe will not set the OCF1B flag.

OCF1B is automatically cleared when the output compare match B interrupt vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

• Bit 1 – OCF1A: Timer/Counter1, Output Compare A Match Flag

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the output compare register A (OCR1A).

Note that a forced output compare (FOC1A) strobe will not set the OCF1A flag.

OCF1A is automatically cleared when the output compare match A interrupt vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

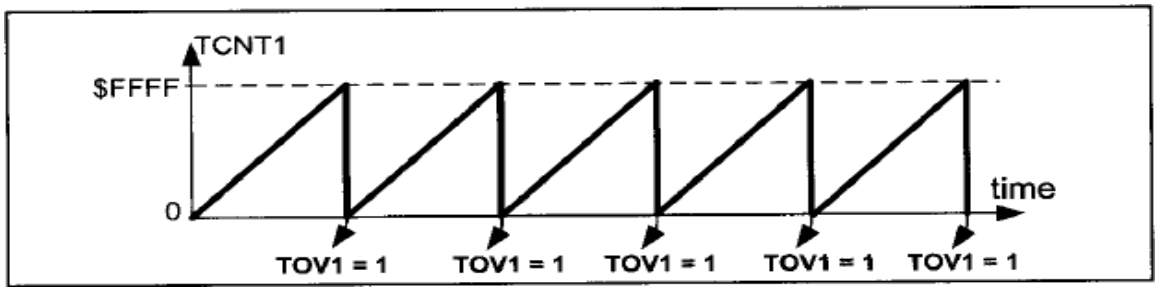
• Bit 0 – TOV1: Timer/Counter1, Overflow Flag

The setting of this flag is dependent of the WGM13:0 bits setting. In normal and CTC modes, the TOV1 flag is set when the timer overflows. he TOV1 flag behavior when using another WGM13:0 bit setting.

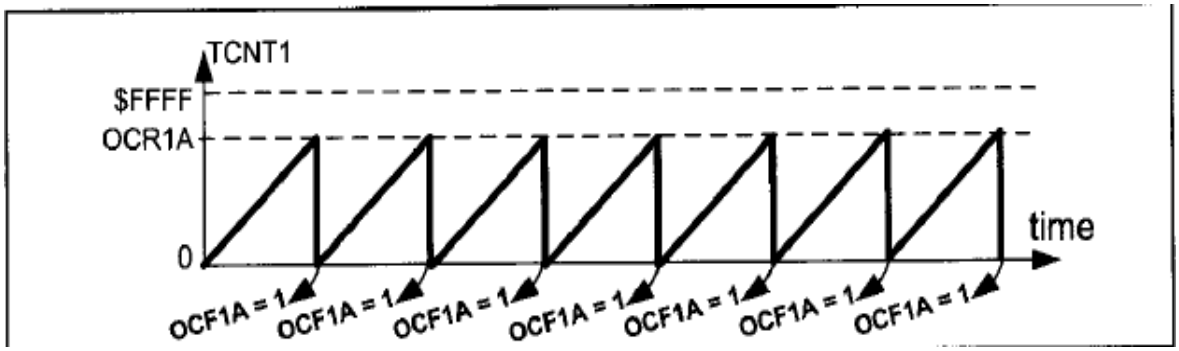
TOV1 is automatically cleared when the Timer/Counter1 overflow interrupt vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

Timer 1 modes

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



- **CTC mode**



(*Explain as given in Timer 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.

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 ➔ $T_{machine\ cycle}$ = 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

```
#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<<TOV1))==0);      //wait for TOV1 to roll over

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

ATmega168/328 Code:

```
// this code sets up timer1 for a 1s  @ 16Mhz Clock (mode 4)


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


int main(void)
{
    OCR1A = 0x3D08;

    TCCR1B |= (1 << WGM12);
    // Mode 4, CTC on OCR1A

    TIMSK1 |= (1 << OCIE1A);
    //Set interrupt on compare match

    TCCR1B |= (1 << CS12) | (1 << CS10);
    // set prescaler to 1024 and start the timer


    sei();
    // enable interrupts


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

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

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

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

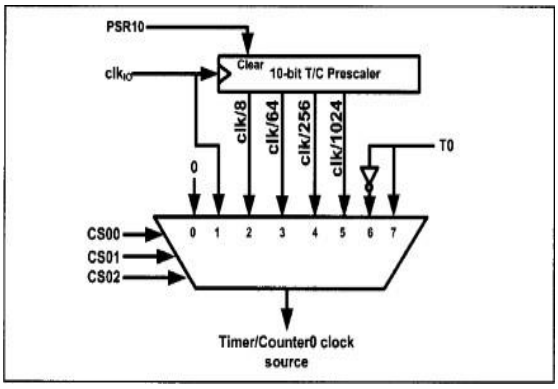
1 s/32 µs = 31,250 clocks = 0x7A12 clocks ➔ 1 + 0xFFFF – 0x7A12 = **0x85EE**

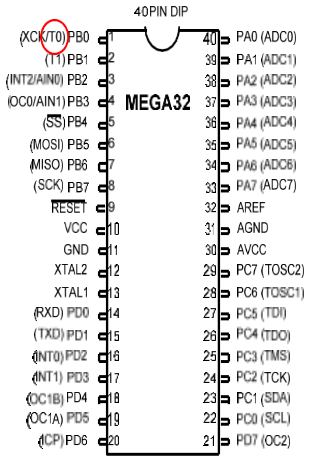
(*Write the code *)

Counter

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.





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

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
      }
}
```