AVR Microcontroller

Microprocessor Course

Chapter 15

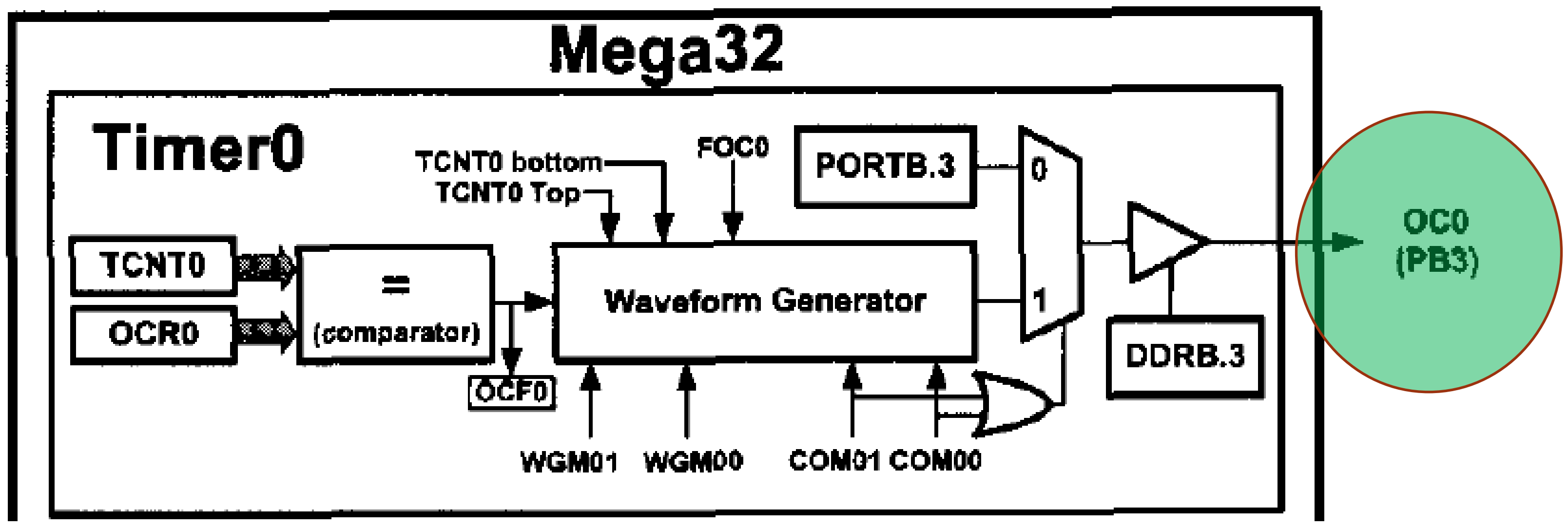
INPUT CAPTURE AND WAVE GENERATION IN AVR

Bahman 1397 (Version 1.2)

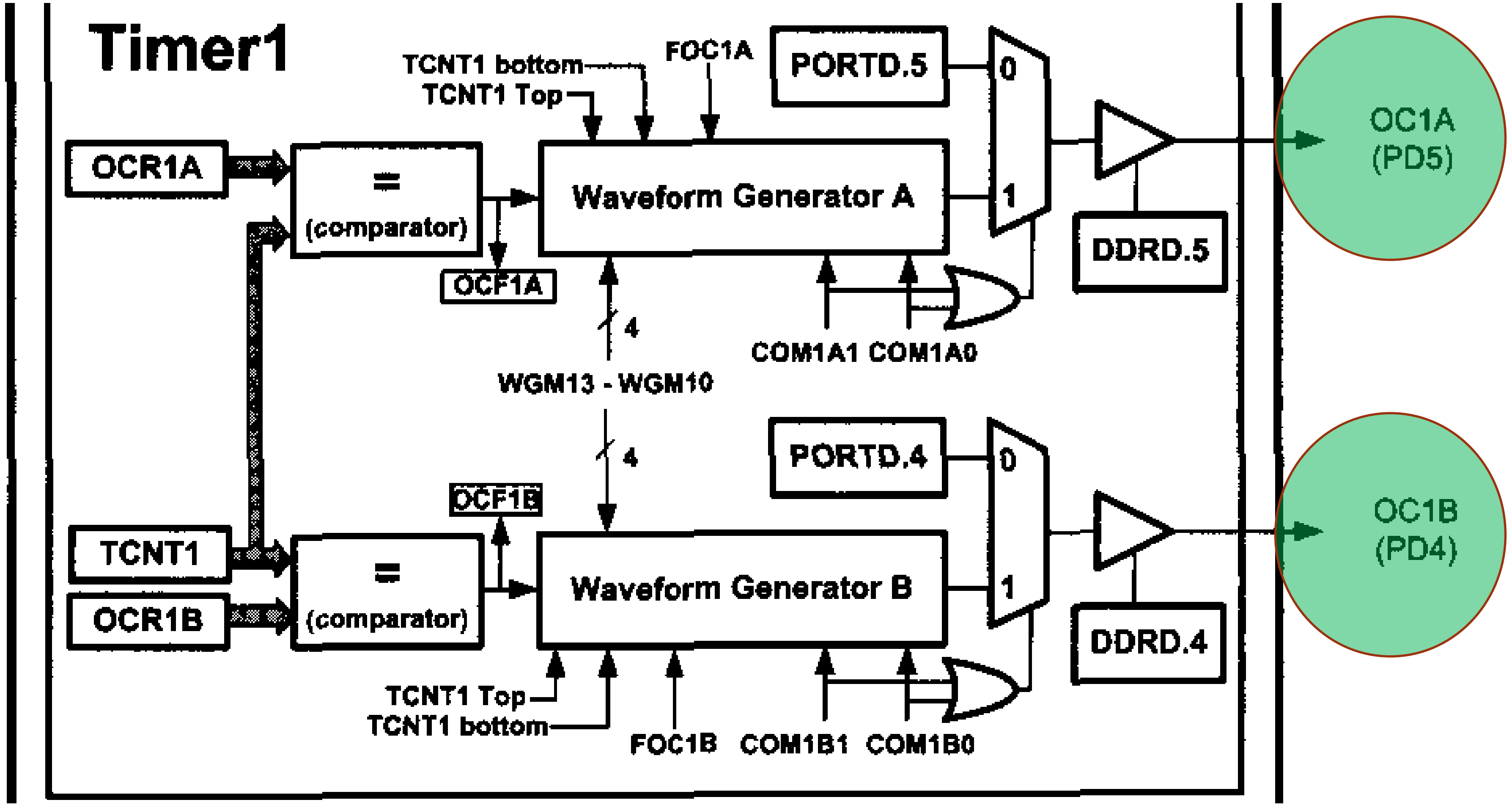
INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

For each timer there is, at least, an OCRn register (like OCR0 for Timer0). As shown in Figures 15-1 and 15-2, in each AVR timer there is a waveform generator.

The waveform generator can generate waves on the OCn pin. The WGMn and COMn bits of the TCCR register determine how the waveform generator works.



SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS



INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

When the TCNTn register reaches Top or Bottom or compare match occurs, the waveform generator is informed. Then the waveform generator changes the state of the OC0 pin according to the mode of the timer (WGM01:00 bits of the TCCR0 register) and the COM01 (Compare Output Mode) and COM00 bits. See Figure 15-4.

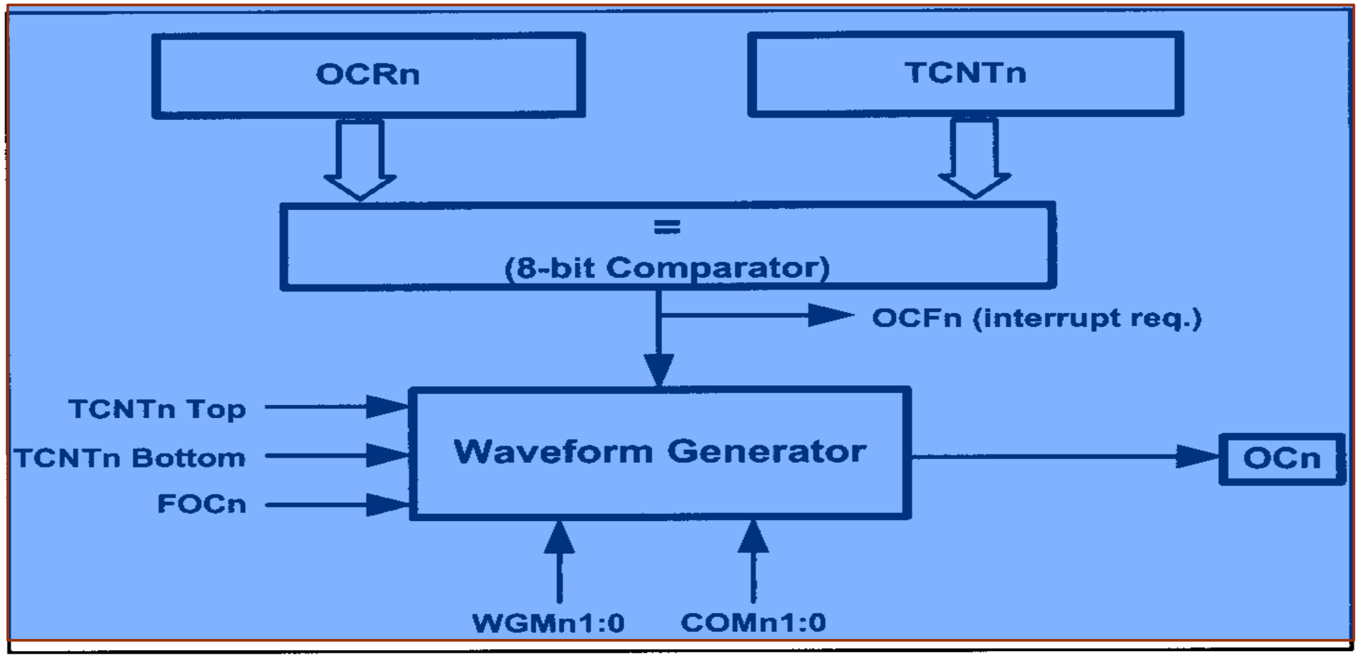


Figure 15-2. Waveform Generator

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

In ATmega32/ATmegal6, OC0 is the alternative function of PB3. In other words, the PB3 functions as an I/O port when both COM01 and COM00 are zero. Otherwise, the pin acts as a wave generator pin controlled by a waveform generator.

We should set the OC0 pin as an output pin when we want to use it for generating waves.

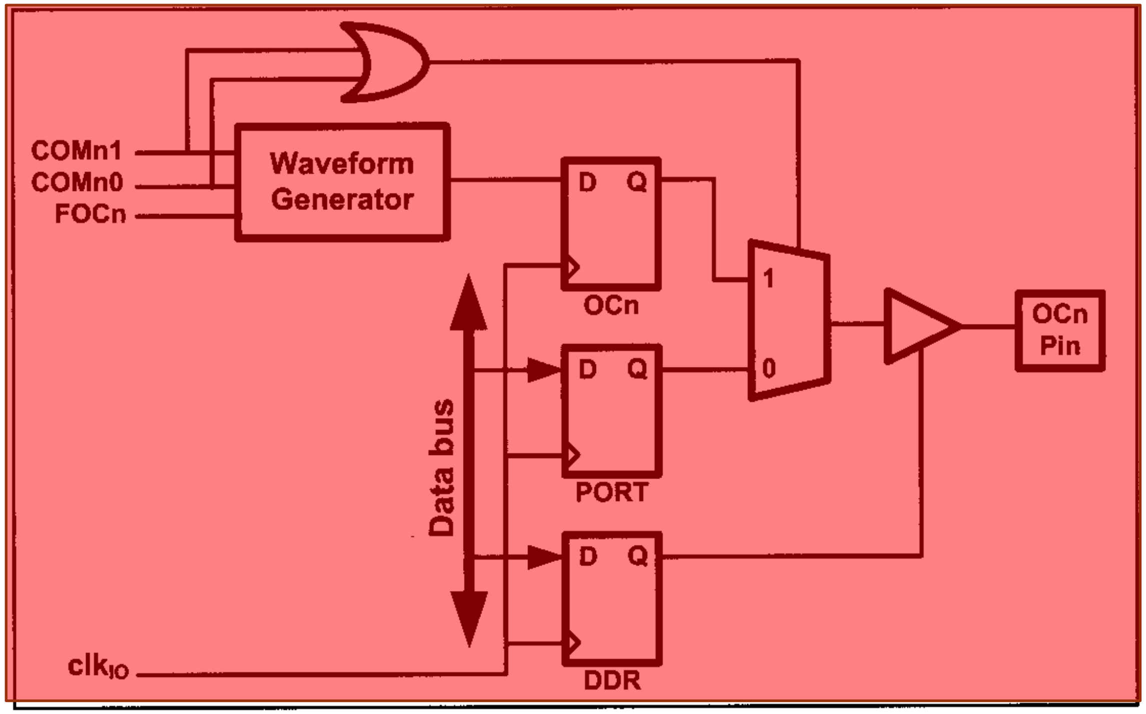
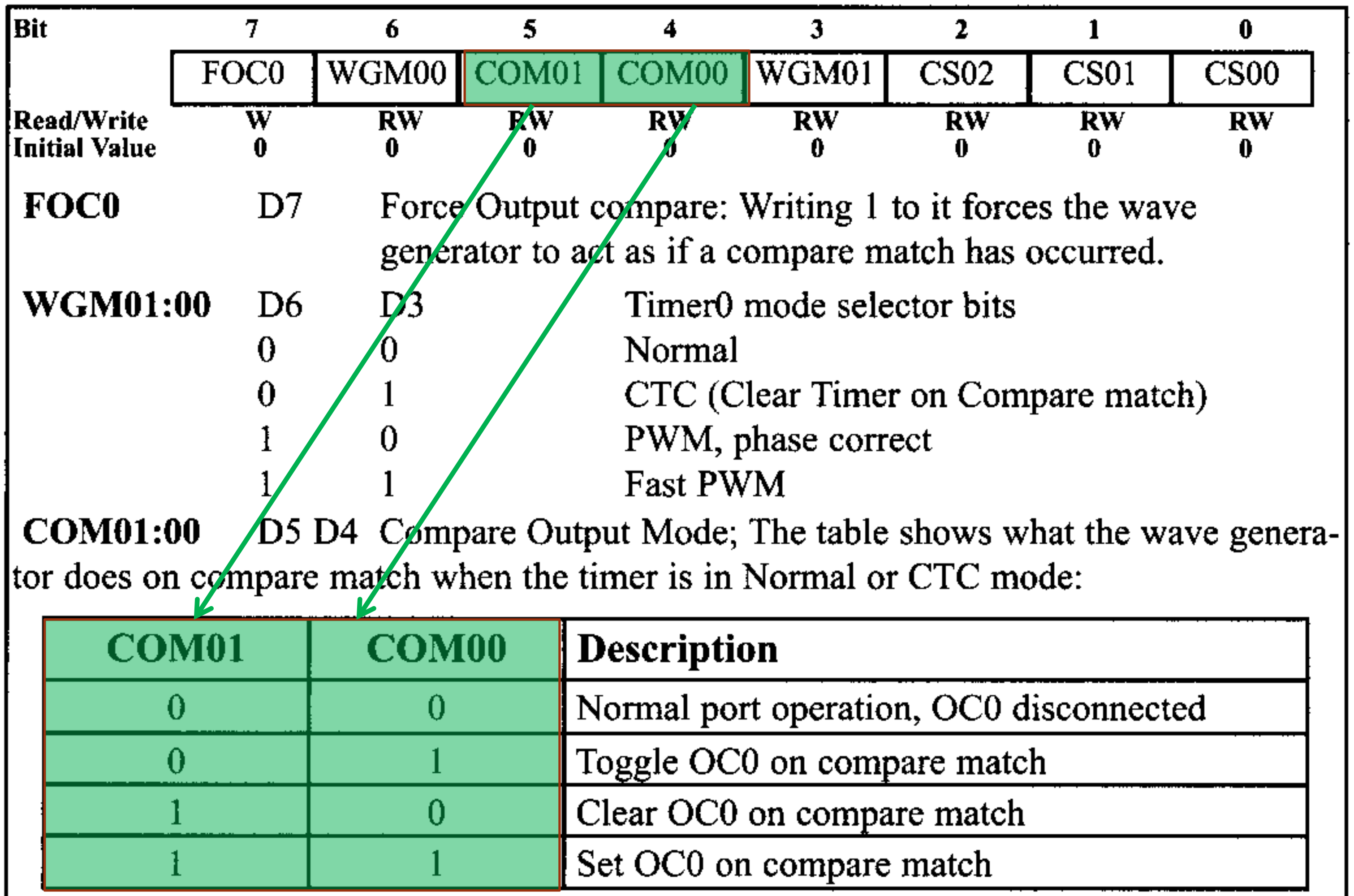


Figure 15-3. DDR Register and Waveform Generator

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Figure 15-4. TCCR0 (Timer/Counter Control Register) Register



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**

FOC0 D7 Force Output compare: Writing 1 to it forces the wave generator to act as if a compare match has occurred.

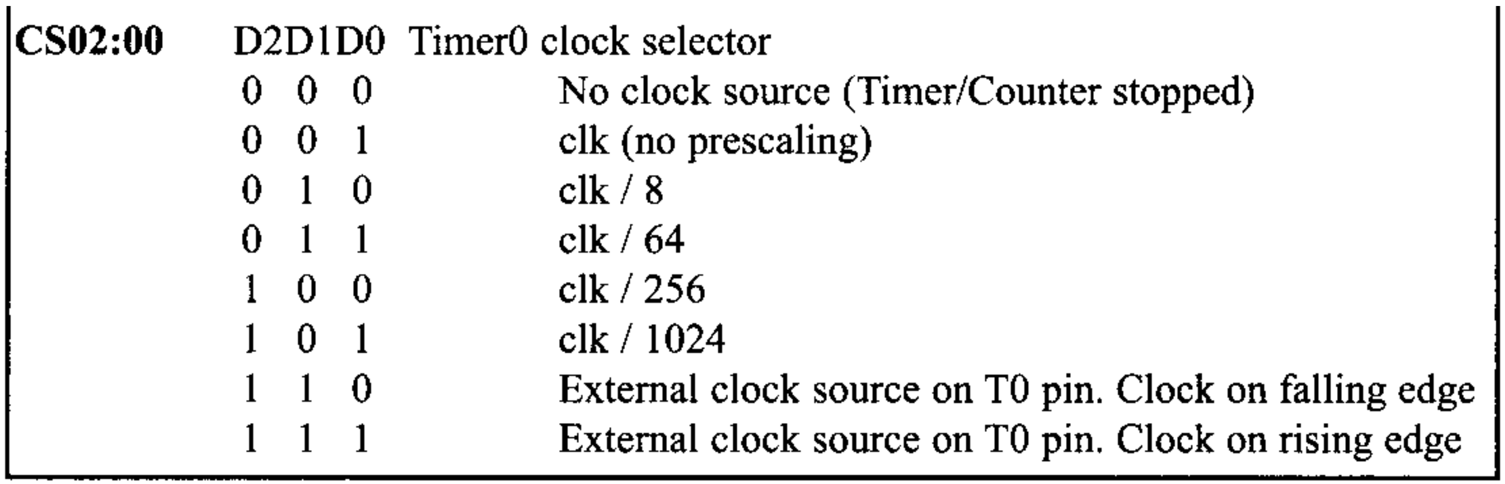
WGM01:00	D6	D3	Timer0 mode selector bits
	0	0	Normal
	0	1	CTC (Clear Timer on Compare match)
	1	0	PWM, phase correct
	1	1	Fast PWM

COM01:00 D5 D4 Compare Output Mode; The table shows what the wave generator does on compare match when the timer is in Normal or CTC mode:

COM01	**COM00**	**Description**
0	0	Normal port operation, OC0 disconnected
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS



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

Figure 15-4. TCCR0 (Timer/Counter Control Register) Register

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Wave generation Normal and CTC modes

When Timer0 is in CTC (WGM01:0 = 10) or Normal (WGM01:0 = 00) mode after a compare match occurs, the OC0 pin can perform one of the following actions, depending on the value of the COM01:0 bits:

(a) Remain unaffected

(b) Toggle the OC0 pin

(c) Clear (Drive low) the OC0 pin

(d) Set (Drive high) the OC0 pin

We use the COM01 and COM00 bits to select one of the above actions

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Example 15-1

Using Figure 15-4, find the TCCR0 register value to:

(a) Set high the OC0 pin upon match. Use external clock, falling edge, and Normal mode.

(b) Toggle the OC0 pin upon match. Use external clock, falling edge, and CTC mode.

Solution:

(a) TCCR0 =

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

(b) TCCR0 =

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

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

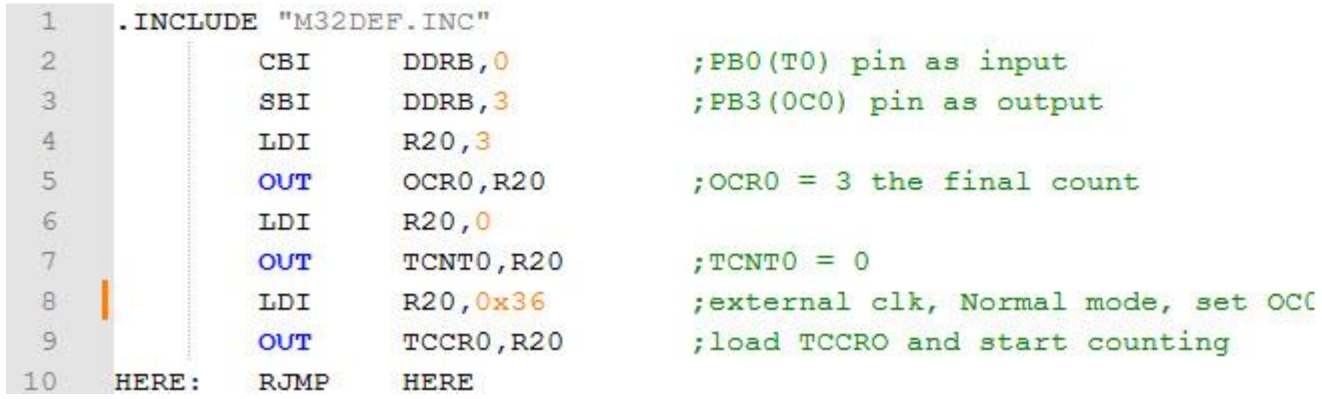
Example 15-2

Write a program that

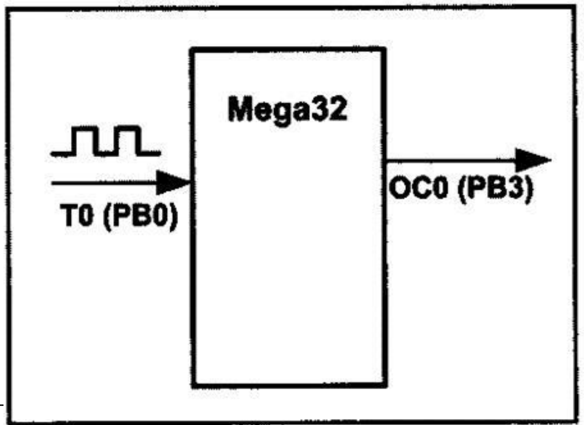
(a) after 4 external clocks turns on an LED connected to the OC0 pin,

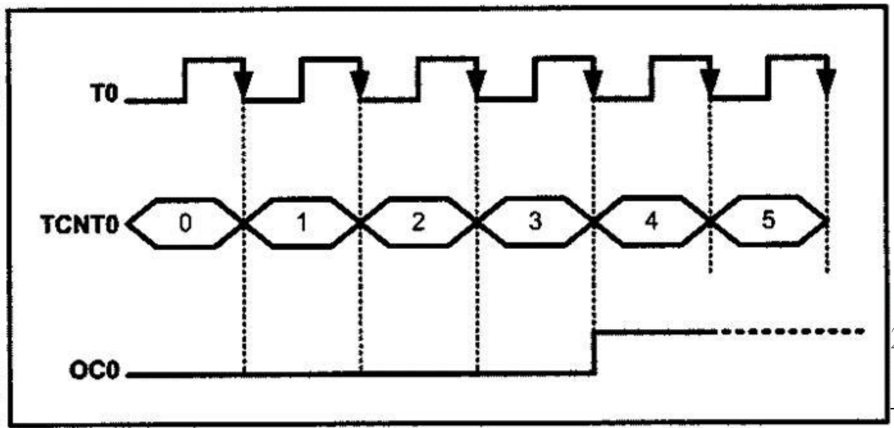
(b) toggles the OC0 pin every 4 pulses.

Solution: (a)



```
.INCLUDE "M32DEF.INC"
        CBI     DDRB,0          ;PB0(T0) pin as input
        SBI     DDRB,3          ;PB3(OC0) pin as output
        LDI     R20,3
        OUT     OCR0,R20        ;OCR0 = 3 the final count
        LDI     R20,0
        OUT     TCNT0,R20       ;TCNT0 = 0
        LDI     R20,0x36        ;external clk, Normal mode, set OC
        OUT     TCCR0,R20       ;load TCCRO and start counting
HERE:   RJMP    HERE
```

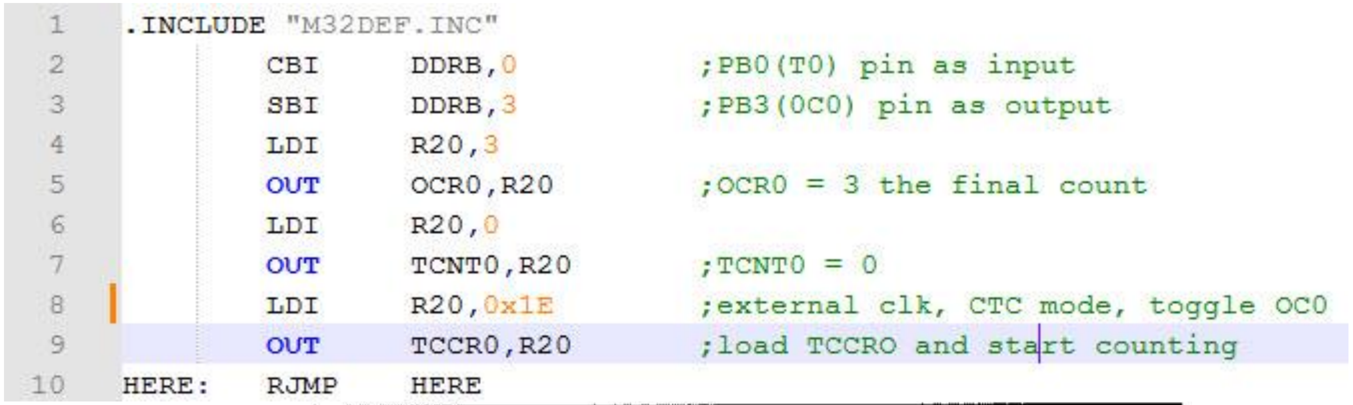




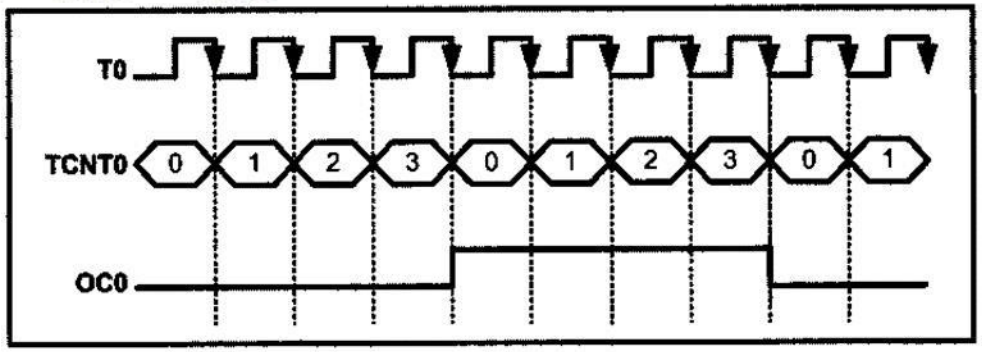
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

(b)



```
.INCLUDE "M32DEF.INC"
        CBI     DDRB,0          ;PB0(T0) pin as input
        SBI     DDRB,3          ;PB3(OC0) pin as output
        LDI     R20,3
        OUT     OCR0,R20        ;OCR0 = 3 the final count
        LDI     R20,0
        OUT     TCNT0,R20       ;TCNT0 = 0
        LDI     R20,0x1E        ;external clk, CTC mode, toggle OC0
        OUT     TCCR0,R20       ;load TCCRO and start counting
HERE:   RJMP    HERE
```



Notice that there is no need to monitor the OCF0 flag, which means the AVR can do other tasks.

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Generating square waves

To generate square waves we can set the timer to Normal mode or CTC mode and set the COM bits to the toggle mode (COM01:00 = 01). The OC0 pin will be toggled on each compare match and a square wave will be generated.

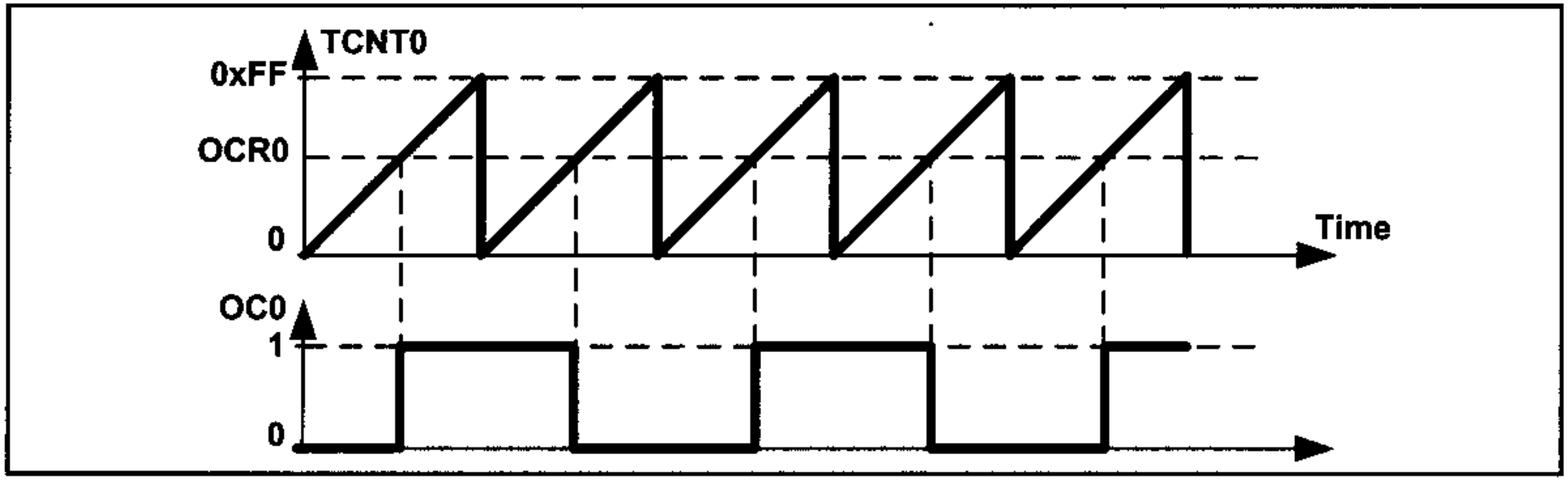


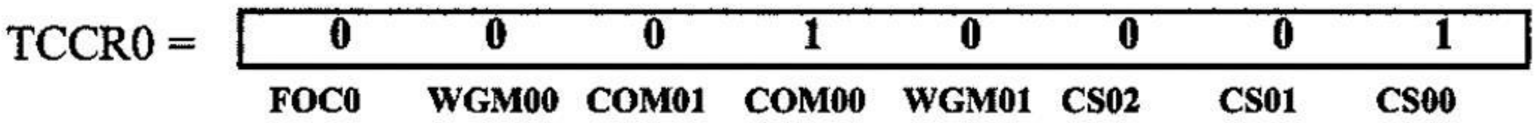
Figure 15-5. Generating Square Wave Using Normal

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Example 15-3

Find the value for TCCR0 if we want to program Timer0 as a Normal mode square wave generator and no prescaler.

Solution:

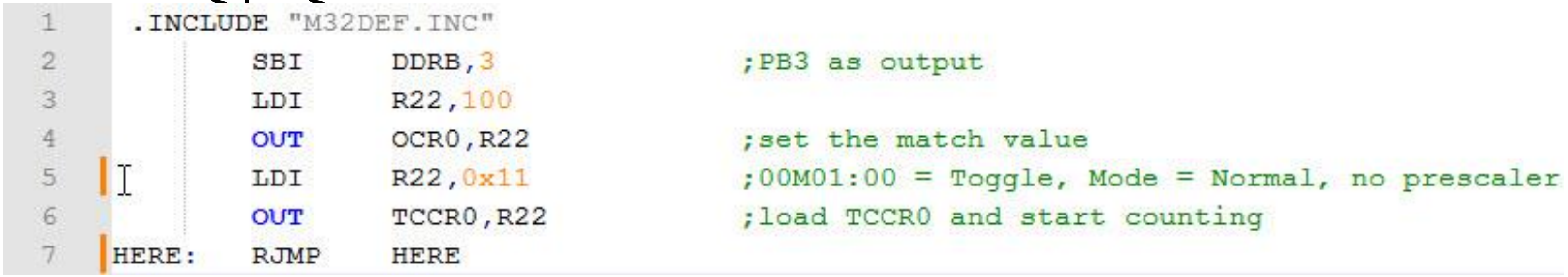


	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
TCCR0 =	0	0	0	1	0	0	0	1

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Example 15-4

Assuming XTAL = 8 MHz, calculate the frequency of the wave generated by the following program:



```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,3              ;PB3 as output
        LDI     R22,100
        OUT     OCR0,R22            ;set the match value
        LDI     R22,0x11            ;00M01:00 = Toggle, Mode = Normal, no prescaler
        OUT     TCCR0,R22           ;load TCCR0 and start counting
HERE:   RJMP    HERE
```

Solution:

There are 256 clocks between two consecutive matches. Therefore

$T_{\text{timer clock}} = 1/8 \text{ MHz} = 0.125\ \mu s$

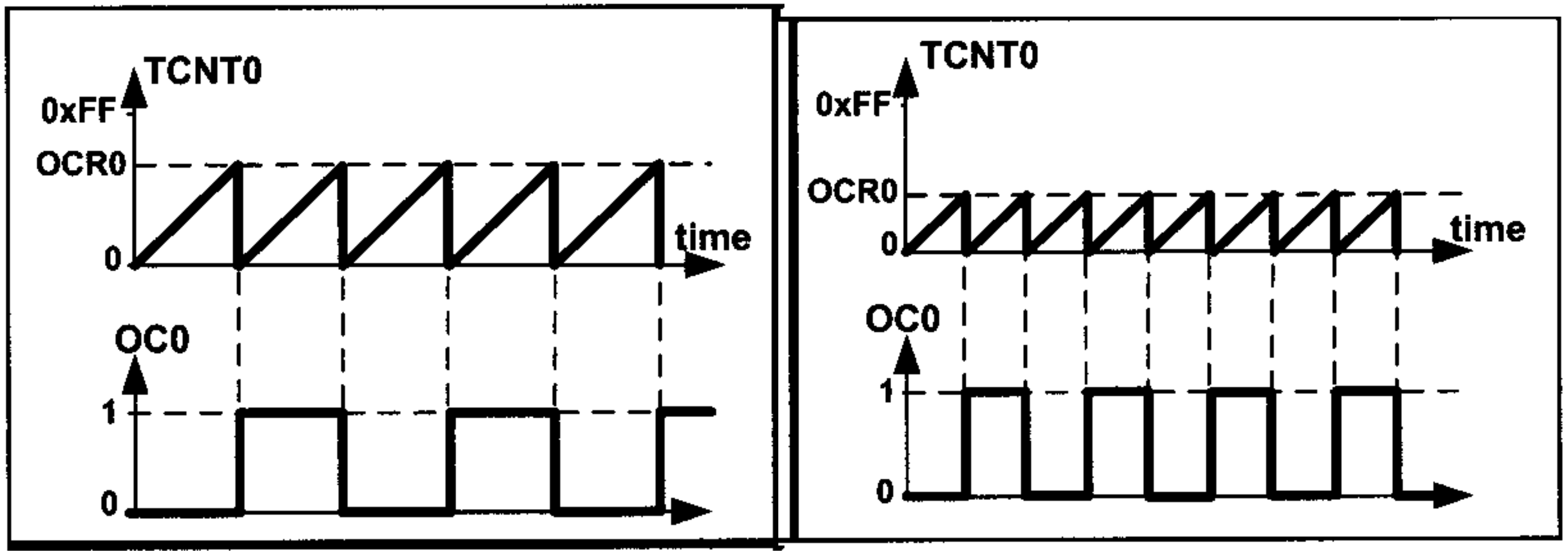
$T_{\text{wave}} = 2 \times 256 \times 0.125\ \mu s = 64\ \mu s$

$F_{\text{wave}} = 1/64\ \mu s = 15{,}625 \text{ Hz} = 15.625 \text{ kHz}$ Note: In Normal mode, when match occurs, the OC0 pin toggles and the timer continues to count up until it reaches the top value.

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Generating square waves using CTC

The CTC mode is better than Normal mode for generating square waves, since the frequency of the wave can be easily adjusted using the OCR0 register.



INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Example 15-5

Find the value for TCCR0 if we want to program Timer0 as a CTC mode square wave generator and no prescaler.

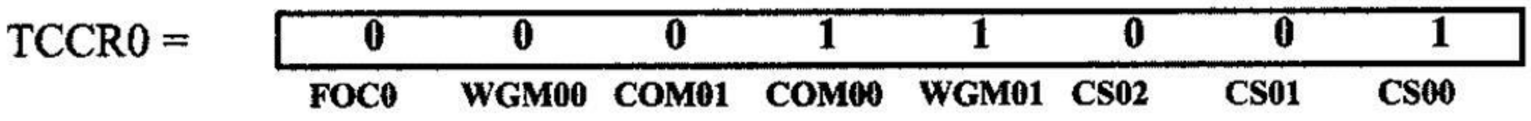
Solution:

WGM01:00 = 10 = CTC mode

COM01:00 = 01 = Toggle

CS02:00 = 001 = No prescaler

FOC0 = 0



TCCR0 =

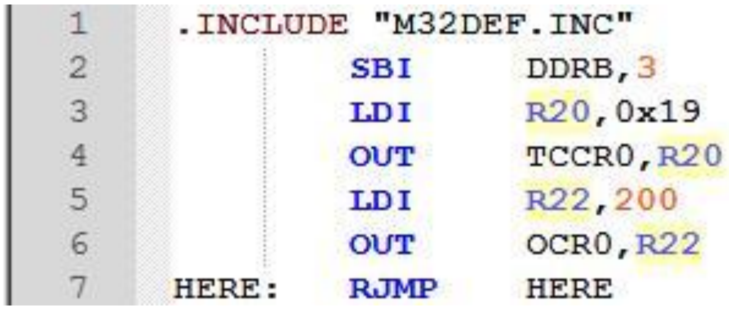
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
0	0	0	1	1	0	0	1

INPUT CAPTURE AND WAVE GENERATION IN AVR

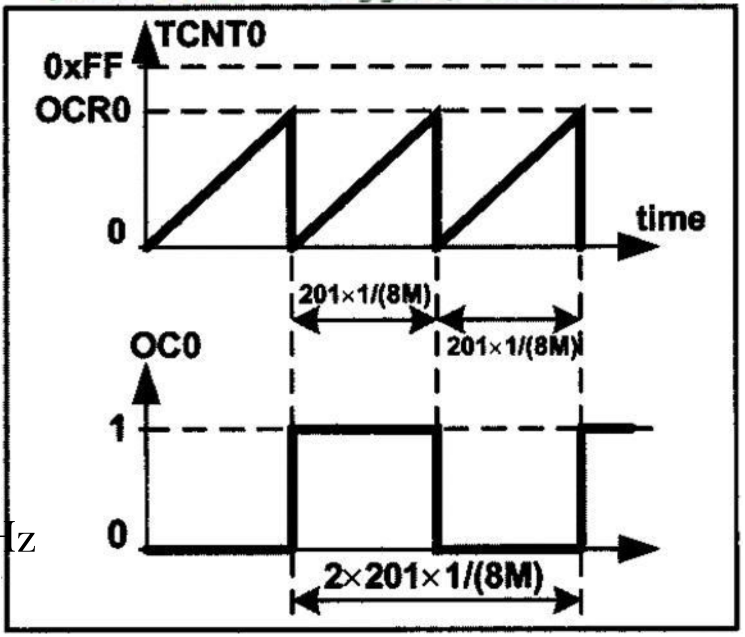
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Example 15-6

Assuming XTAL = 8 MHz, calculate the frequency of the wave generated by the following program:



```
.INCLUDE "M32DEF.INC"
            SBI     DDRB,3
            LDI     R20,0x19
            OUT     TCCR0,R20       ;COM01:00 = Toggle, Mode = CTC
            LDI     R22,200
            OUT     OCR0,R22
HERE:       RJMP    HERE
```



Solution:

Between two consecutive matches it takes 200+1=201 clocks and

$T_{\text{timer clock}}$ =1/8MHz=0.125 µs

T_{wave} = 2×201×0.125 µs=50.25 µs

F_{wave}=1/50.25 µs=19,900 Hz=19.900 kHz

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Example 15-7

In Example 15-6, calculate the frequency of the wave generated in each of the following cases:

(a) OCR0 is loaded with 50

(b) XTAL = 4 MHz and OCR0 is loaded with 95

(c) prescaler is 8, XTAL = 1 MHz, OCR0 = 150

(d) prescaler is N, XTAL = Fosc, OCR0 = X

Solution:

(a) 50 + 1 = 51 clocks and $t_{\text{timer clock}}=0.125\ \mu s \Rightarrow T_{\text{wave}}= 2\times51\times0.125\mu s=12.75\mu s$
$\Rightarrow F_{\text{wave}}=1/12.75\mu s=78{,}431$ Hz

(b) 95+1=96 clocks and $T_{\text{timer clock}}=1/4\ \text{MHz}=0.25\mu s \Rightarrow T_{\text{wave}}=2\times96\times0.25\mu s=48\mu s$
$\Rightarrow F_{\text{wave}}=1/48\mu s=20{,}833$ Hz = 20.833 kHz

(c) 150+1=151 clocks and $T_{\text{timer clock}}=8\times1/1\text{MHz}=8\mu s \Rightarrow T_{\text{wave}}=2\times151\times8\mu s=2416\mu s$
$\Rightarrow F_{\text{wave}}=1/2416\mu s=413.9$ Hz

(d) X+1clocks and $T_{\text{timer clock}}=N\times1/F_{\text{osc}}=N/Fosc \Rightarrow T_{\text{wave}}=2\times(X+1)\times N/Fosc$
$\Rightarrow F_{\text{wave}} = 1 / T_{\text{wave}} = F_{\text{osc}} / [2N(X + 1)]$

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Generating pulses using CTC mode

When a timer is in the CTC mode and COM is in the toggle mode, the value of the OCRn represents how many clocks it counts before it toggles the pin. This way, we can generate different pulses by loading different values into the OCRn register.

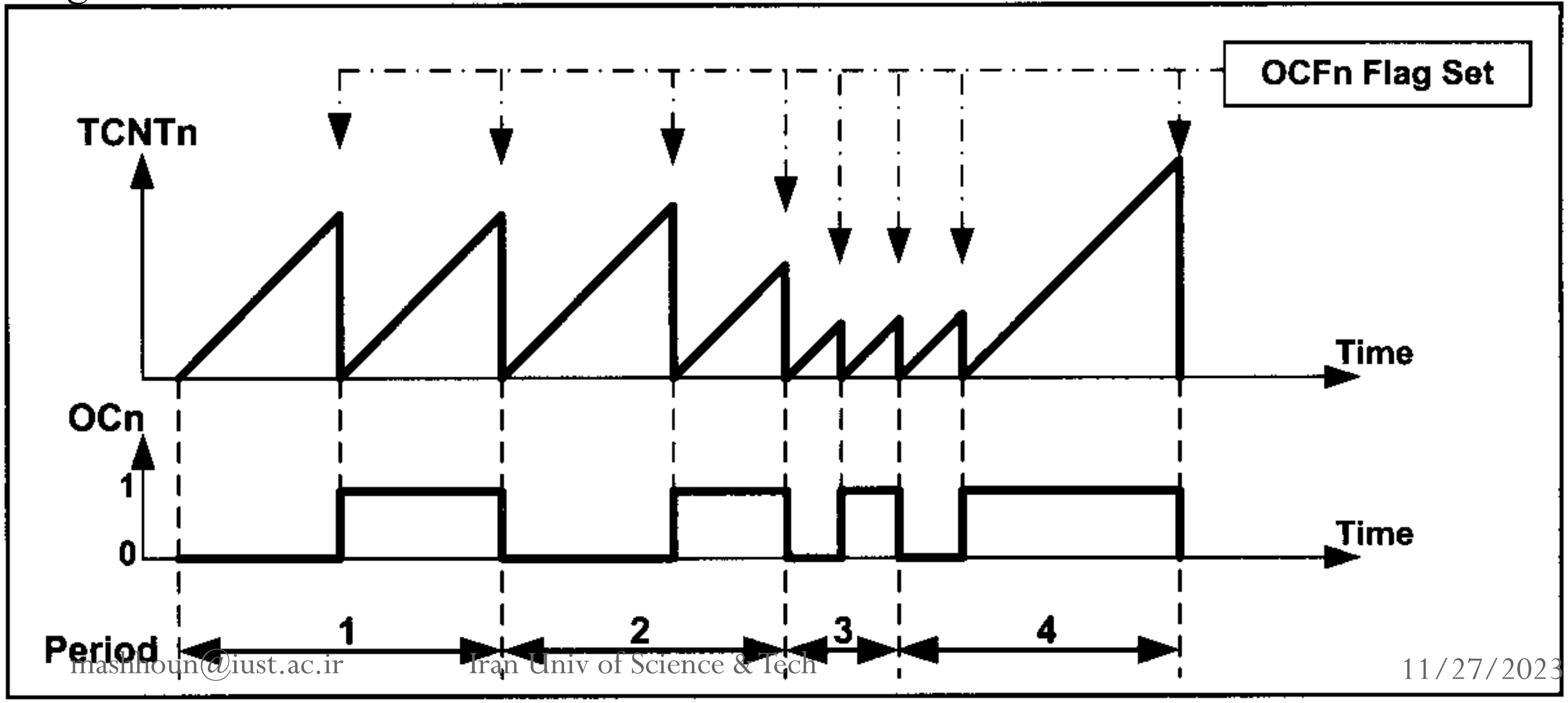
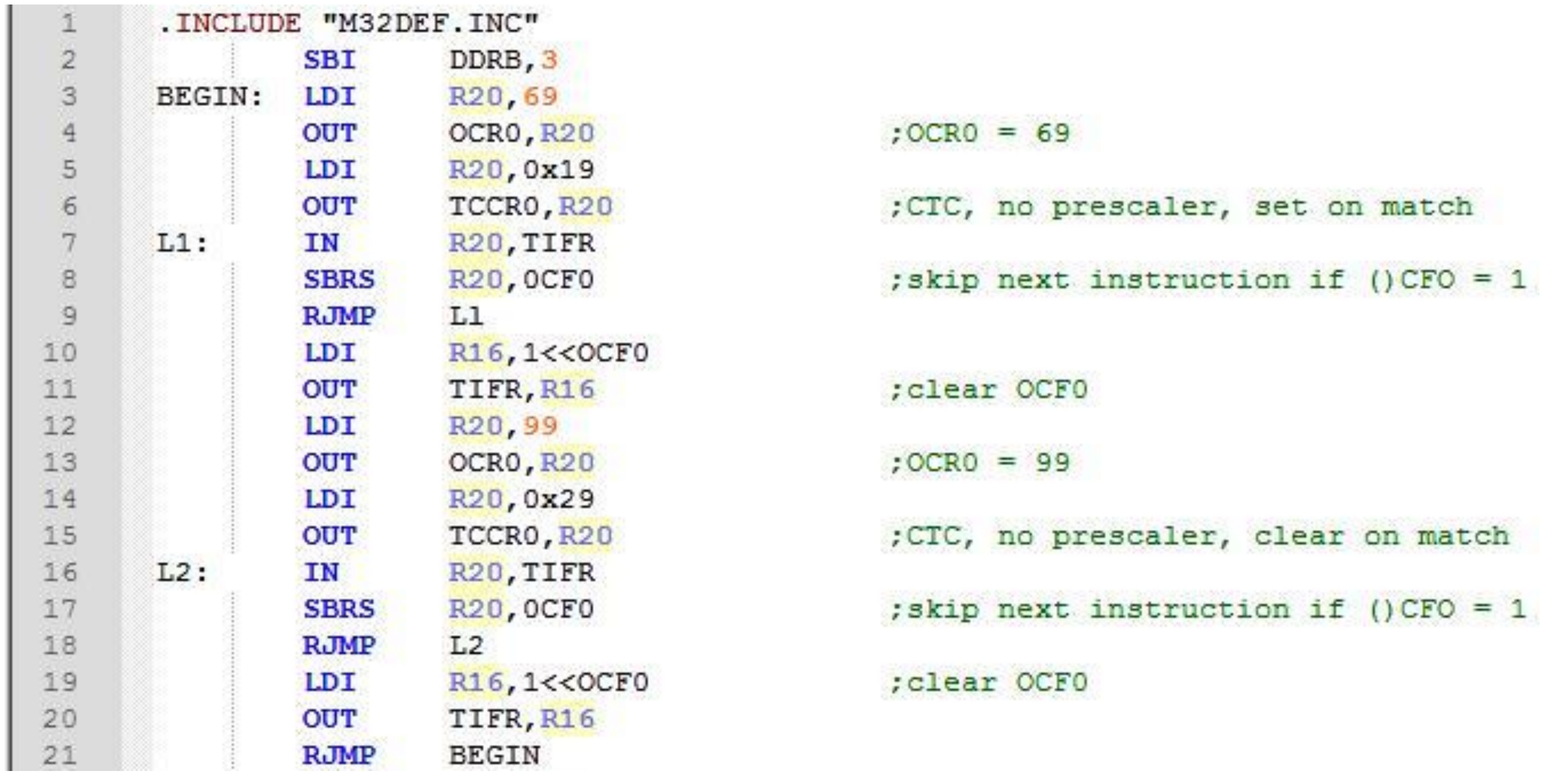


Figure 15-7. Generating Different Pulses Using CTC and Toggle Modes

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Example 15-8

Assuming XTAL = 1 MHz, draw the wave generated by the following program:



```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,3
BEGIN:  LDI     R20,69
        OUT     OCR0,R20                ;OCR0 = 69
        LDI     R20,0x19
        OUT     TCCR0,R20               ;CTC, no prescaler, set on match
L1:     IN      R20,TIFR
        SBRS    R20,OCF0                ;skip next instruction if ()CFO = 1
        RJMP    L1
        LDI     R16,1<<OCF0
        OUT     TIFR,R16                ;clear OCF0
        LDI     R20,99
        OUT     OCR0,R20                ;OCR0 = 99
        LDI     R20,0x29
        OUT     TCCR0,R20               ;CTC, no prescaler, clear on match
L2:     IN      R20,TIFR
        SBRS    R20,OCF0                ;skip next instruction if ()CFO = 1
        RJMP    L2
        LDI     R16,1<<OCF0             ;clear OCF0
        OUT     TIFR,R16
        RJMP    BEGIN
```

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Solution:

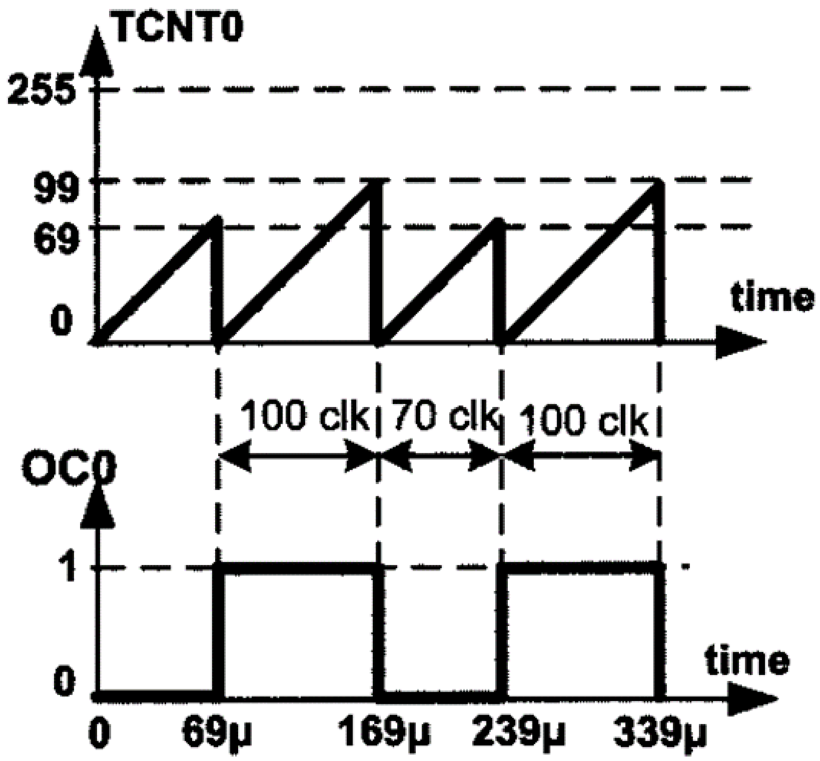
$T_{\text{timer clock}} = 1 / 1 \text{ MHz} = 1 \text{ µs}$

$T_0 = 70 \times 1 \text{ µs} = 70 \text{ µs}$

$T_1 = 100 \times 1 \text{ µs} = 100 \text{ µs}$

$T_{\text{wave}} = 70 \text{ µs} + 100 \text{ µs} = 170 \text{ µs}$

$F_{\text{wave}} = 1 / 170 \text{ µs} = 5882 \text{ Hz}$

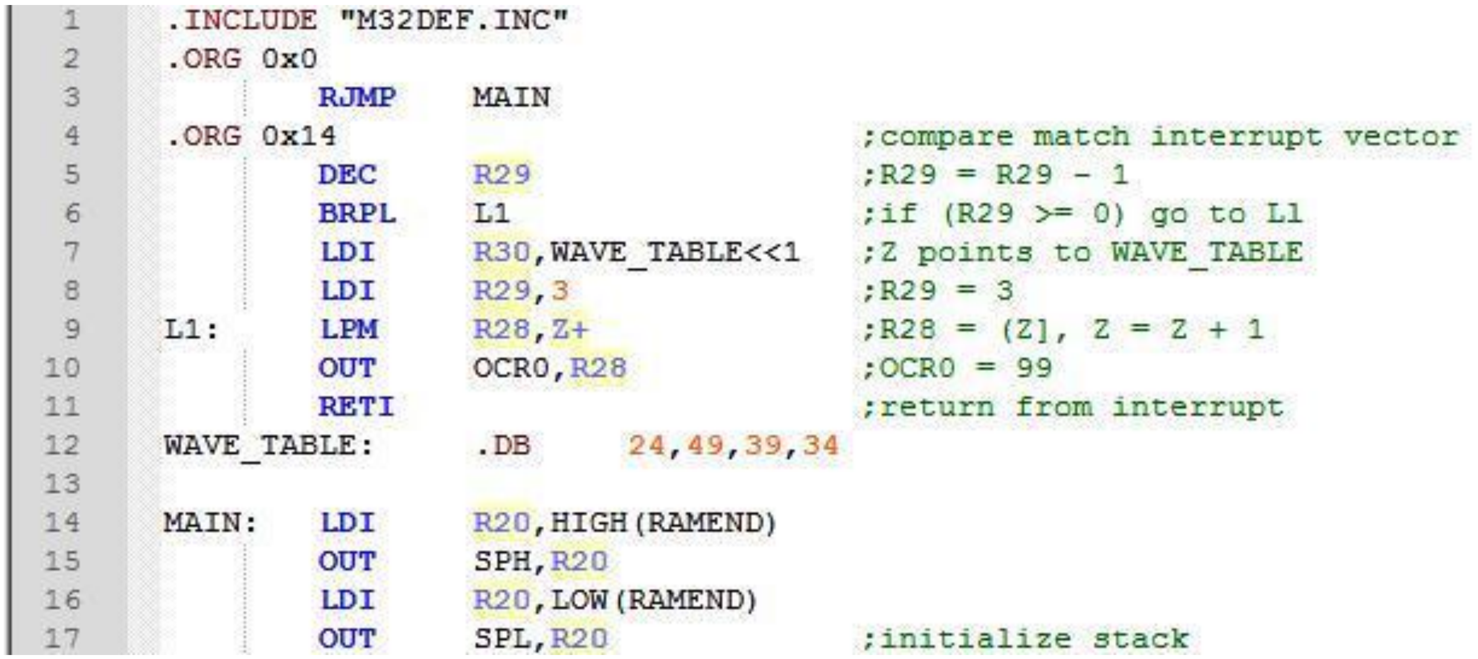


SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

To load values to the OCRn we can use the compare match interrupt as well. Upon a compare match, the pin will be toggled and an interrupt will be invoked.

Example 15-9

Assuming XTAL = 1 MHz, draw the wave generated by the following program:



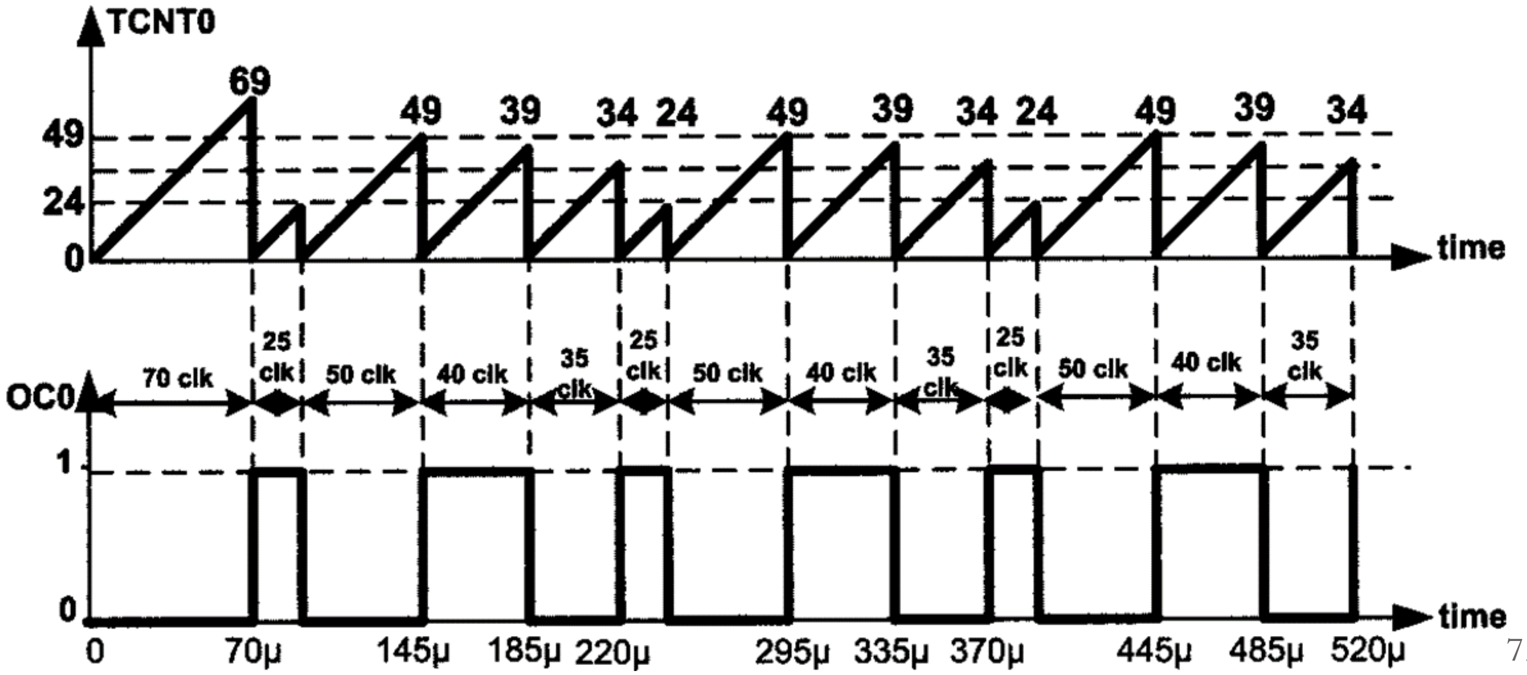
```
.INCLUDE "M32DEF.INC"
.ORG 0x0
        RJMP    MAIN
.ORG 0x14                                  ;compare match interrupt vector
        DEC     R29                        ;R29 = R29 - 1
        BRPL    L1                         ;if (R29 >= 0) go to L1
        LDI     R30,WAVE_TABLE<<1          ;Z points to WAVE_TABLE
        LDI     R29,3                      ;R29 = 3
L1:     LPM     R28,Z+                     ;R28 = (Z], Z = Z + 1
        OUT     OCR0,R28                   ;OCR0 = 99
        RETI                               ;return from interrupt
WAVE_TABLE:     .DB     24,49,39,34

MAIN:   LDI     R20,HIGH(RAMEND)
        OUT     SPH,R20
        LDI     R20,LOW(RAMEND)
        OUT     SPL,R20                    ;initialize stack
```

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

```
        SBI     DDRB,3              ;PB3 as output
        LDI     R20,69
        OUT     OCR0,R20            ;OCR0 = 69
BEGIN:  LDI     R20,0x19
        OUT     TCCR0,R20           ;CTC, no prescaler, toggle on match
        LDI     R20,1<<OCIE0
        OUT     TIMSK,R20           ;activate compare match interrupt
        SEI
HERE:   RJMP    HERE
```

Solution:



INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

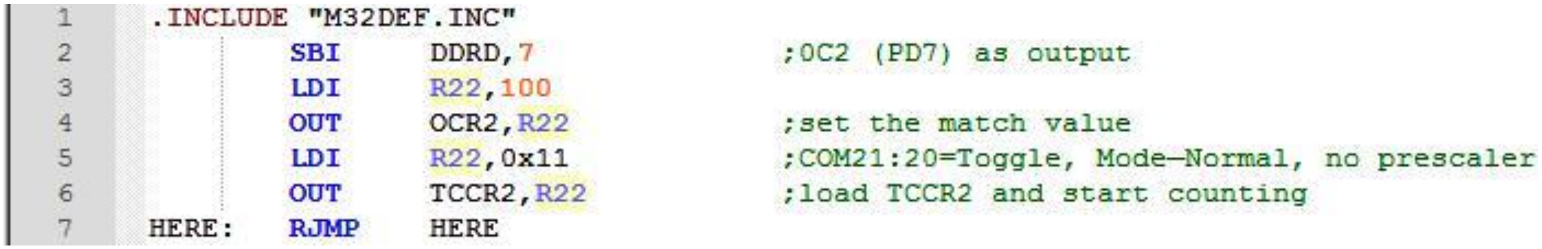
Generating waves using Timer2

We can generate waves using Timer2 or any other 8-bit timer the same way as we did using Timed. We should simply use the proper registers and monitor the associated flag.

Example 15-11

Rewrite the program of Example 15-4 using Timer2.

Solution:



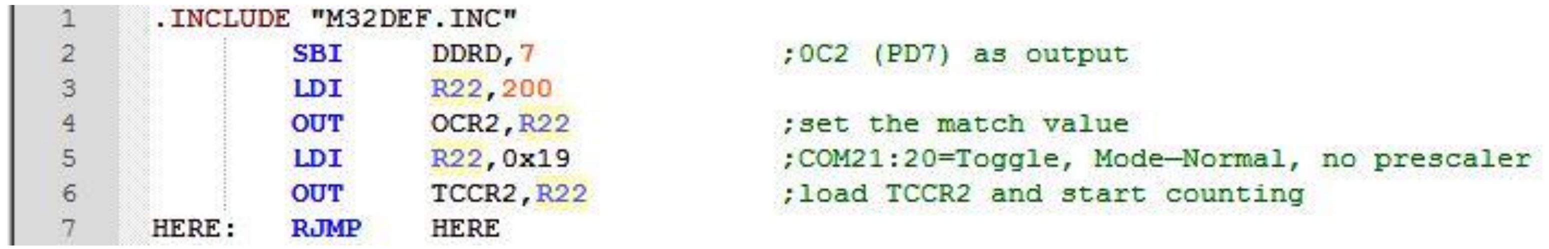
```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,7                  ;OC2 (PD7) as output
        LDI     R22,100
        OUT     OCR2,R22                ;set the match value
        LDI     R22,0x11                ;COM21:20=Toggle, Mode=Normal, no prescaler
        OUT     TCCR2,R22               ;load TCCR2 and start counting
HERE:   RJMP    HERE
```

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.1: WAVE GENERATION USING 8-BIT TIMERS

Example 15-12

Rewrite the program of Example 15-6 using Timer2.

Solution:



```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,7                  ;OC2 (PD7) as output
        LDI     R22,200
        OUT     OCR2,R22                ;set the match value
        LDI     R22,0x19                ;COM21:20=Toggle, Mode=Normal, no prescaler
        OUT     TCCR2,R22               ;load TCCR2 and start counting
HERE:   RJMP    HERE
```

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

The different modes of Timer1

The WGM13, WGM12, WGM11, and WGM10 bits define the mode of Timer1. Timer1 has 16 different modes. These modes can be categorized into five groups:

- Normal,
- CTC,
- Fast PWM,
- Phase Correct PWM, and
- Phase and Frequency Correct PWM

We learned about the operation of the first two categories in Chapter 9; the operation of the other categories will be discussed in this part.

Before discussing the operation of the different modes we should define the meaning of Top.

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

Top in Timer1

Top is the highest value that the TCNT register reaches while counting. In 8-bit timers (e.g., Timer0) the top value is 0xFF except for the CTC mode, whose top can be defined by OCRn. In 16-bit timers such as Timer1 the top values are as follows:

- In Normal mode (mode 0) the top value is 0xFFFF.
- In some modes the top value is fixed and is other than the maximum; the top value can be 0xFF, 0x1FF, or 0x3FF.
- In some other modes the top can be defined by either the OCR1A register or the ICR1 register.

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

- **Bit 7 – ICNC1: Input Capture Noise Canceler**
- **Bit 6 – ICES1: Input Capture Edge Select**

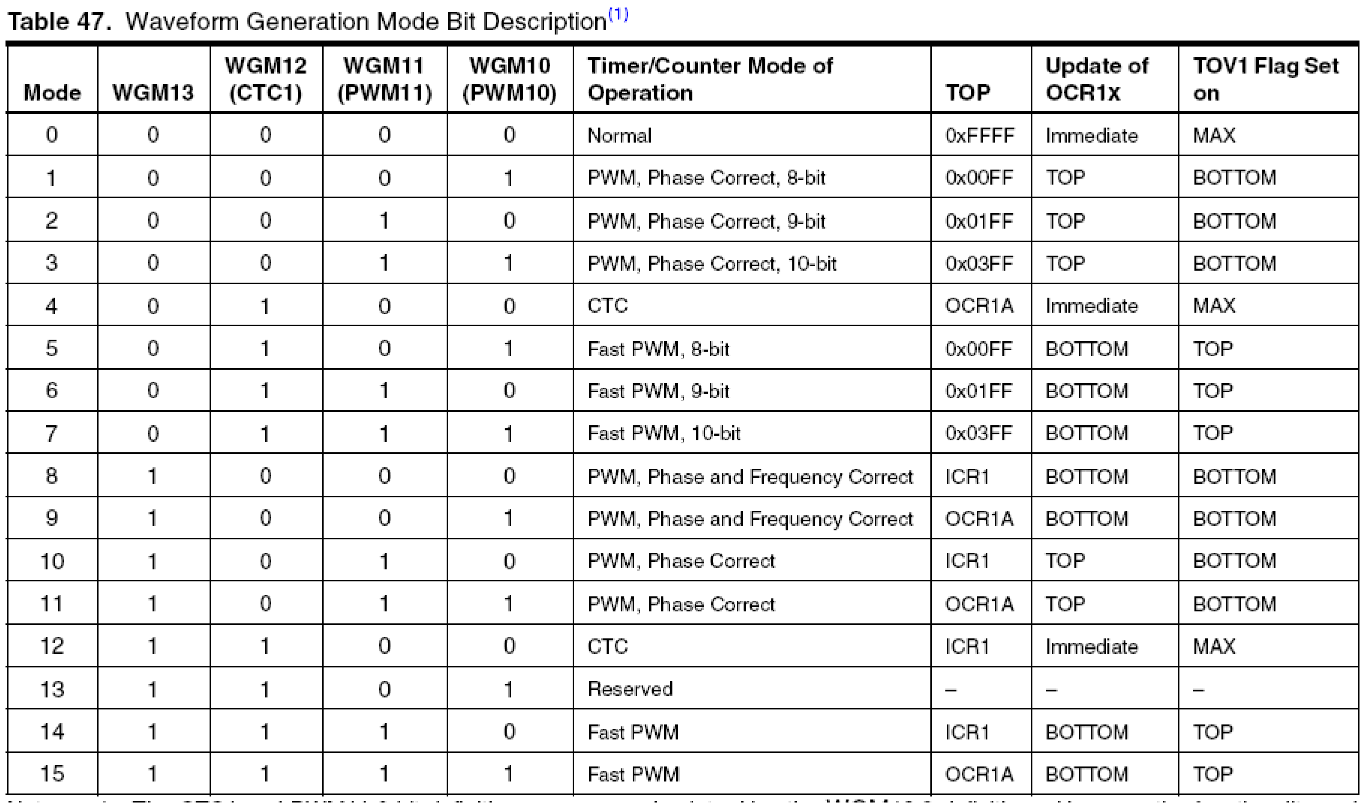


Table 47. Waveform Generation Mode Bit Description[1]

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	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	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	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Figure 15-8. TCCR1B (Timer 1 Control) Register

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING 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.

Figure 15-8. TCCR1B (Timer 1 Control) Register

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

CTC mode

As shown in Figure 15-8, modes 4 and 12 operate in the CTC mode. They are almost the same. The only difference between them is that in mode 4, the top value is defined by OCR1A, whereas in mode 12, ICR1 specifies the top.

As mentioned in Chapter 9, in mode 4, the timer counts up until it reaches OCR1A; then the timer will be cleared and the OCF1A flag will be set as a result of compare match.

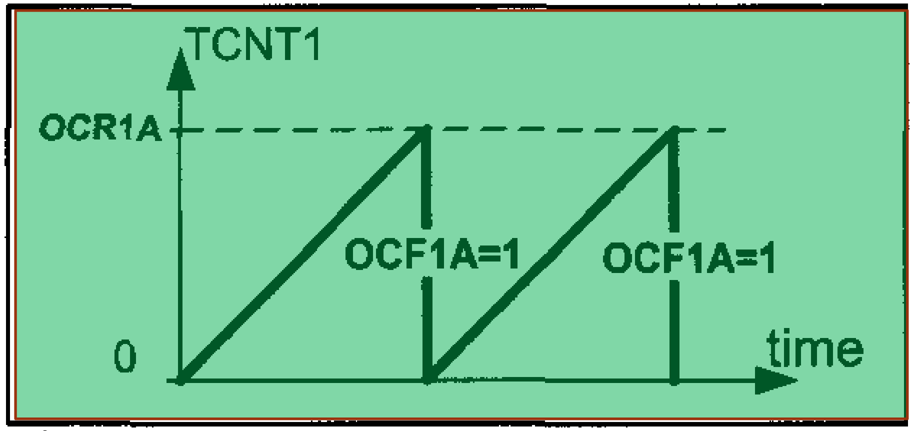


Figure 15-9. Modes 4 and 12

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

CTC mode

In mode 12, the timer counts up until it reaches ICR; then the timer will be cleared and the ICF1 flag will be set. So, in mode 12, the timer works almost the same way as mode 4. See Example 15-13 and compare it with Example 9-22.

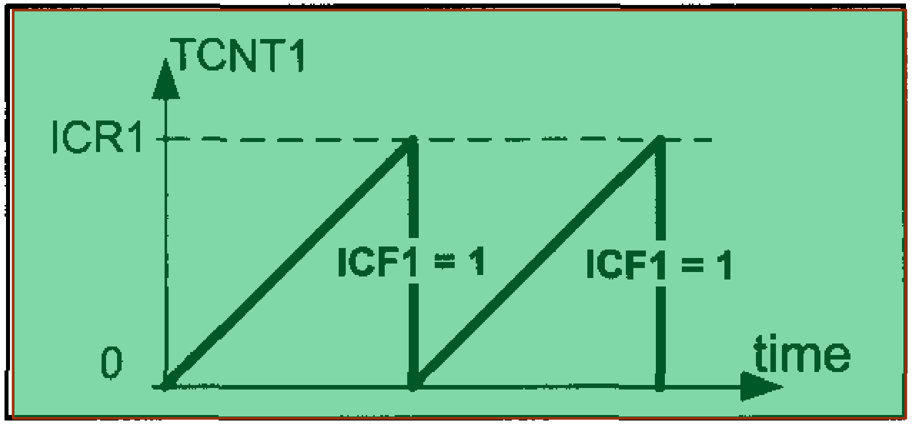


Figure 15-10. Modes 12 and 14

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

In other words, in Normal, CTC, and Fast PWM, the timer counts up until it reaches the top and then rolls over to zero. But the top value is different in the different modes and as a result, different flags are set when the timer rolls over.

- when the top value is a fixed value, the TOV1 flag is set;
- when the OCR1A register defines the top, the OCF1A flag will be set; and
- when the top is defined by the ICR1 register, the ICF1 flag will be set.

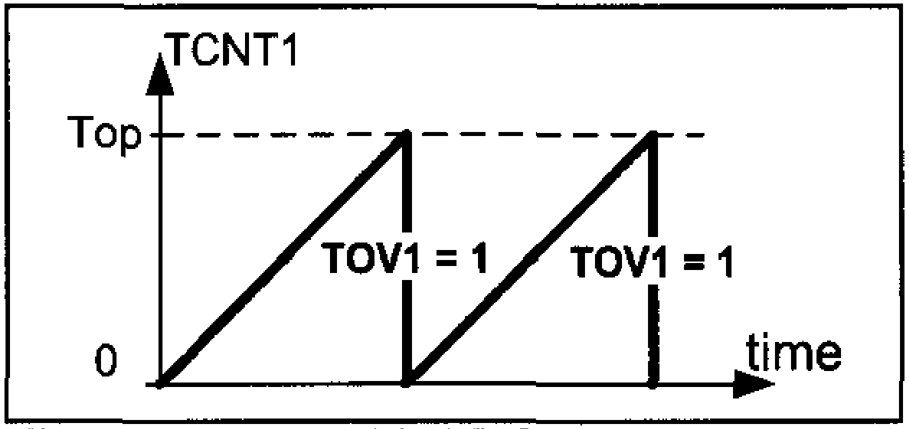


Figure 15-11. TOV1 in Modes 0, 5, 6, and 7

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.2: WAVE GENERATION USING TIMER1

See You might find the contents of these two pages confusing. There is no need to memorize the details. All you need to know is how the timer counts in each of the five categories of operations (Normal, CTC, etc.) and how to use the information mentioned in Figure 15-8. The following is a summary:

Counting:

In Normal, CTC, and Fast PWM modes the timer counts up until it reaches the top value. Then the timer rolls over to zero and a flag is set:

- ❑ If the top is a fixed value, TOV1 will be set.
- ❑ If the OCR1A register represents the top, the OCF1A will be set.
- ❑ If the ICR1 register defines the top, the ICF1 will be set.

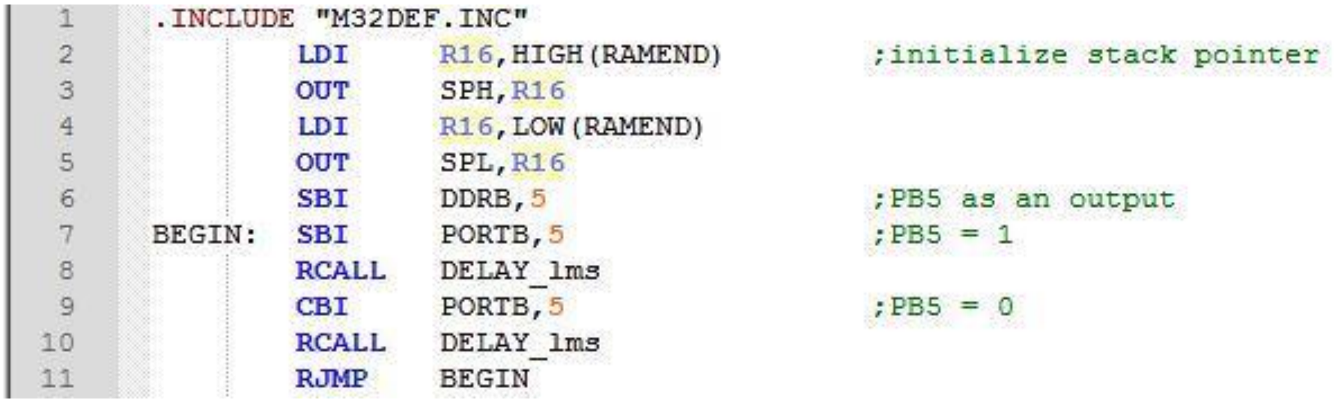
INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

Example 15-13

Rewrite Example9-27 using the ICR1 flag.

Solution:

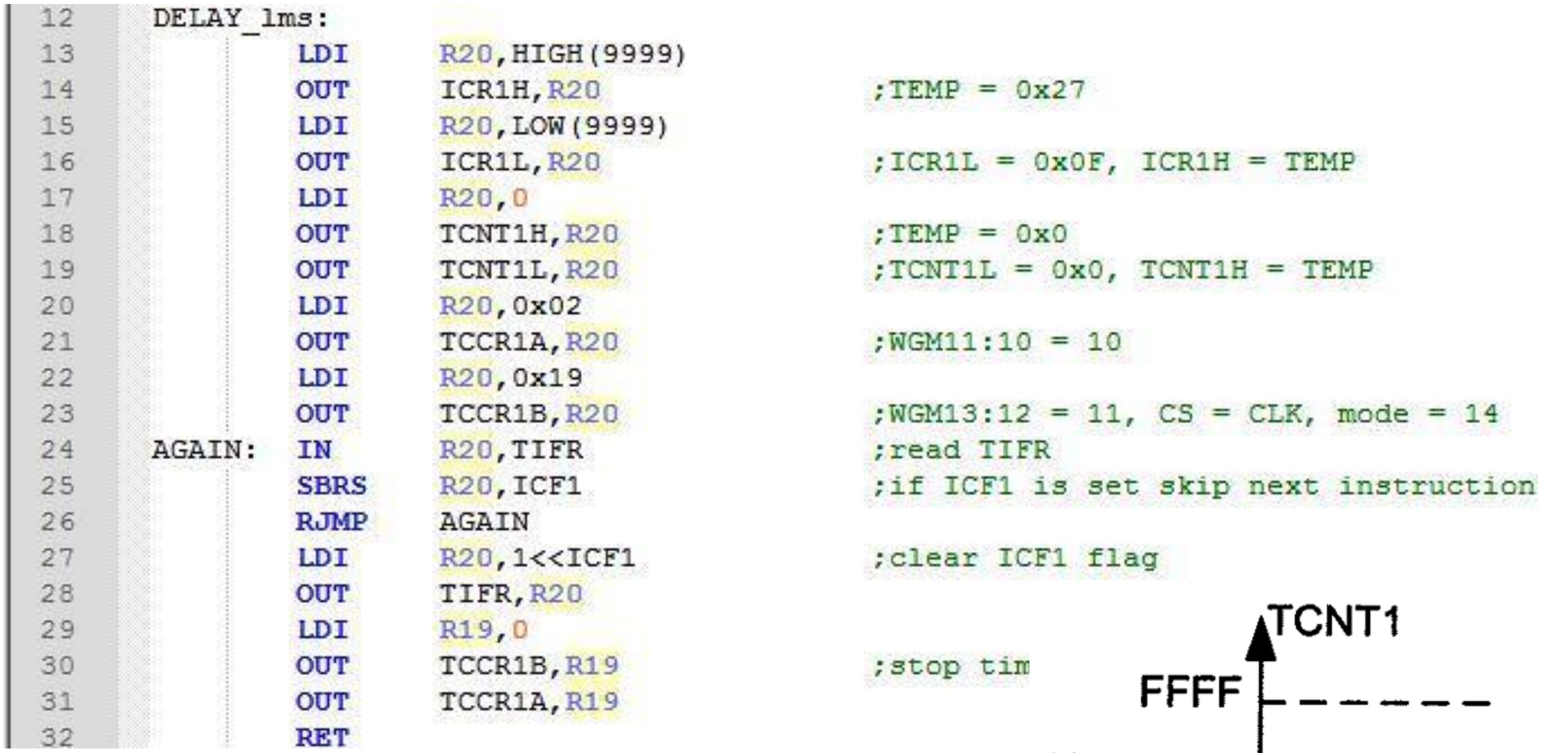
To wait 10,000 clocks we should load the ICR1 flag with 10,000- 1 = 9999 = 0x270F and use mode 14.



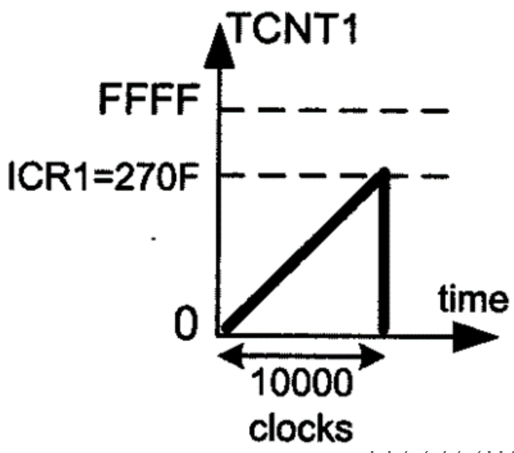
```
.INCLUDE "M32DEF.INC"
        LDI     R16,HIGH(RAMEND)        ;initialize stack pointer
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        SBI     DDRB,5                  ;PB5 as an output
BEGIN:  SBI     PORTB,5                 ;PB5 = 1
        RCALL   DELAY_1ms
        CBI     PORTB,5                 ;PB5 = 0
        RCALL   DELAY_1ms
        RJMP    BEGIN
```

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.2: WAVE GENERATION USING TIMER1



```
DELAY_1ms:
        LDI     R20,HIGH(9999)
        OUT     ICR1H,R20               ;TEMP = 0x27
        LDI     R20,LOW(9999)
        OUT     ICR1L,R20               ;ICR1L = 0x0F, ICR1H = TEMP
        LDI     R20,0
        OUT     TCNT1H,R20              ;TEMP = 0x0
        OUT     TCNT1L,R20              ;TCNT1L = 0x0, TCNT1H = TEMP
        LDI     R20,0x02
        OUT     TCCR1A,R20              ;WGM11:10 = 10
        LDI     R20,0x19
        OUT     TCCR1B,R20              ;WGM13:12 = 11, CS = CLK, mode = 14
AGAIN:  IN      R20,TIFR                ;read TIFR
        SBRS    R20,ICF1                ;if ICF1 is set skip next instruction
        RJMP    AGAIN
        LDI     R20,1<<ICF1             ;clear ICF1 flag
        OUT     TIFR,R20
        LDI     R19,0
        OUT     TCCR1B,R19              ;stop tim
        OUT     TCCR1A,R19
        RET
```



INPUT CAPTURE AND WAVE GENERATION IN AVR

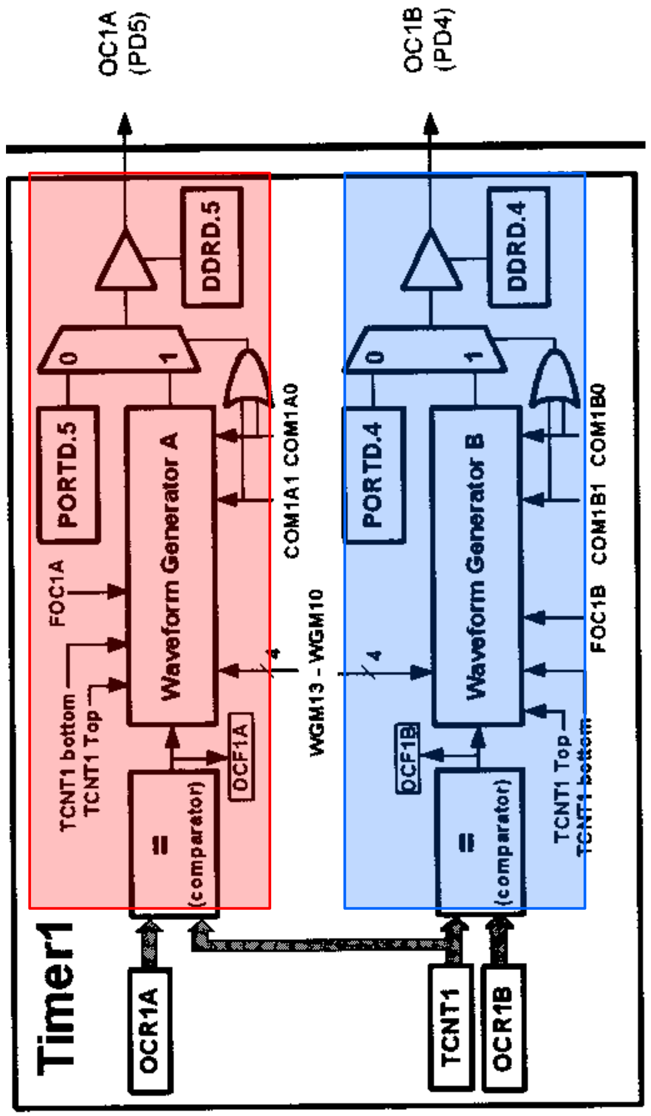
SECTION 15.2: WAVE GENERATION USING TIMER1

Waveform generators in Timer1

In examining Figures 15-12 and 15-13 we see that Timer1 has two independent waveform generators: Waveform Generator A and Waveform Generator B.

The compare match between OCR1A and TCNT1 affects Waveform Generator A, and the wave generated by Waveform Generator A shows up on the OC1A pin.

The compare match between OCR1B and TCNT1 affects Waveform Generator B, and the wave generated by Waveform Generator B shows up on the OC1B pin.



SECTION 15.2: WAVE GENERATION USING TIMER1

The COM1A1 and COM1A0 bits have control over Waveform Generator A; whereas COM1B1 and COM1B0 control Waveform Generator B. All of the COM bits are in the TCCR1A register.

The operation mode of Timer1 (WGM13, WGM12, WGM11, and WGM10 bits of TCCR1A and TCCR1B) affect both generators.

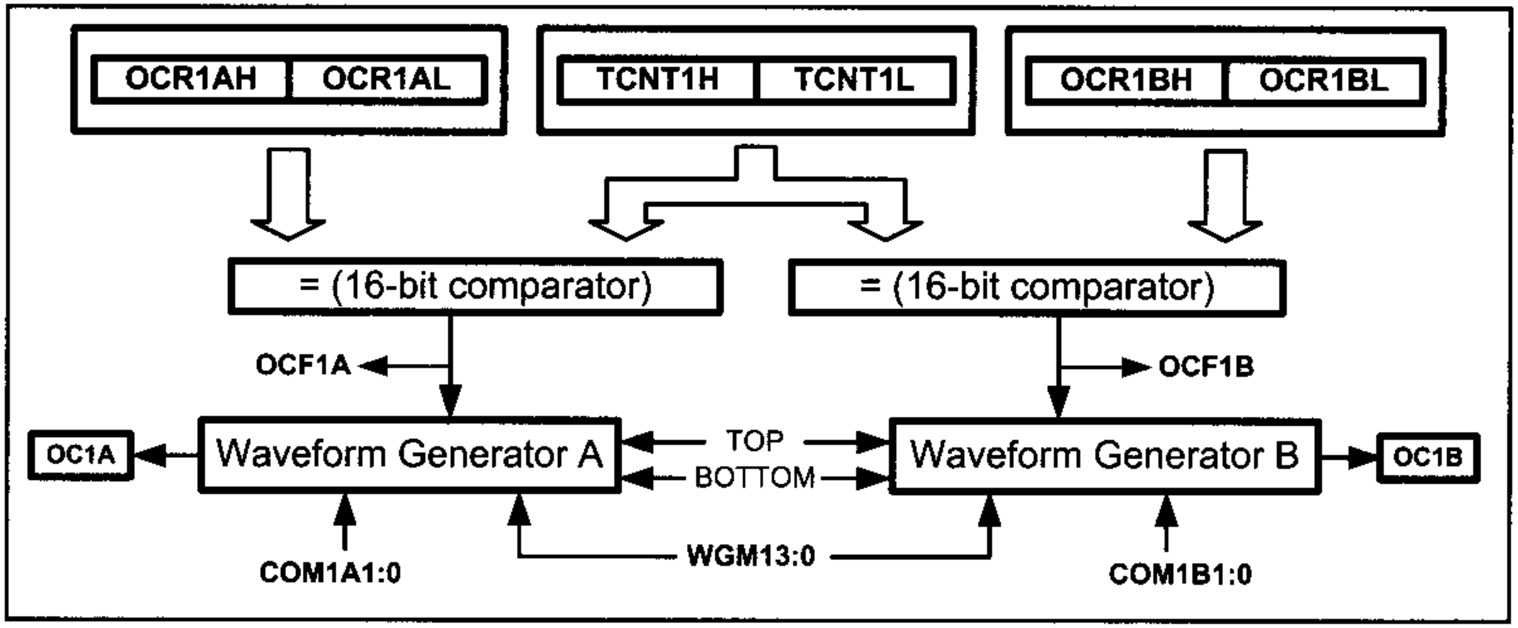


Figure 15-13. Simplified Waveform Generator Block Diagram

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.2: WAVE GENERATION USING TIMER1

Waveform generators in Timer1

In examining Figures 15-12 and 15-13 we see that Timer1 has two independent waveform generators: Waveform Generator A and Waveform Generator B.

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on compare match
1	0	Clear OC1A/OC1B on compare match (Set output to low level)
1	1	Set OC1A/OC1B on compare match (Set output to high level)

Figure 15-14. TCCR1A (Timer 1 Control) Register

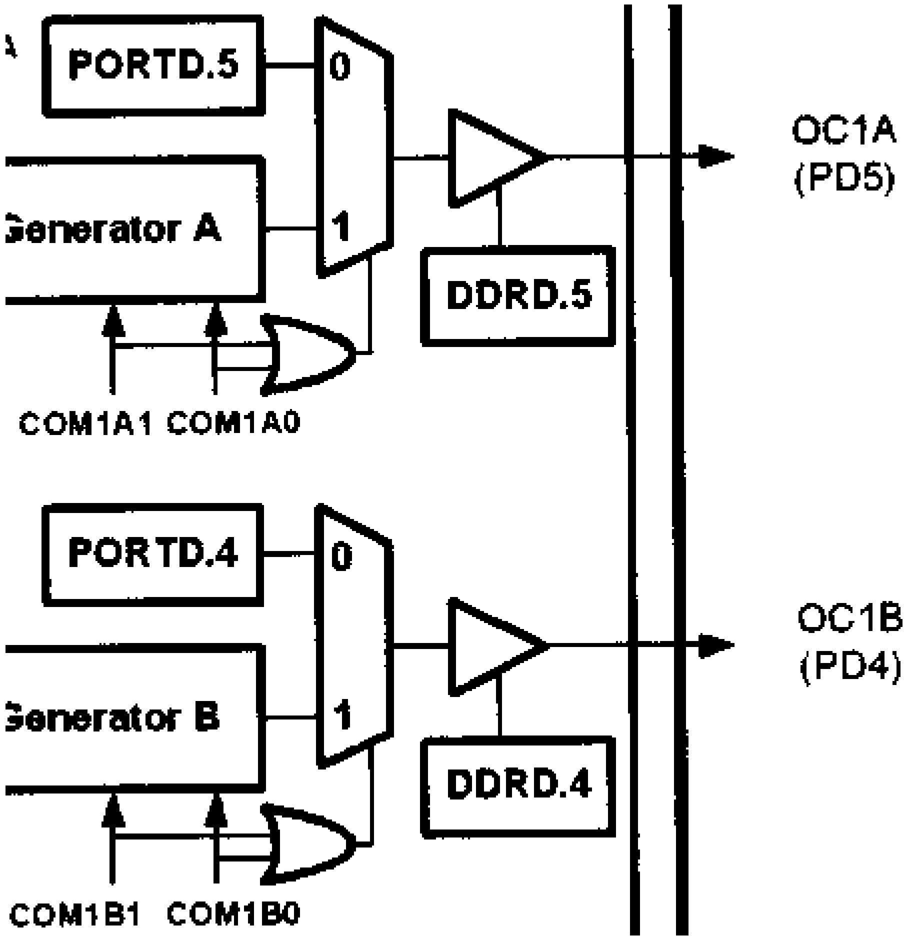
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.2: WAVE GENERATION USING TIMER1

In ATmega32, OC1A and OC1B are the alternative functions of PD5 and PD4, respectively.

The PD5 pin functions as an I/O port when both COM1A1 and COM1A0 are zero. Otherwise, the pin acts as a wave generator pin controlled by Waveform Generator A.

The PD4 functions as an I/O port when both COM1B1 and COM1B0 are zero. Otherwise, the pin acts as a wave generator pin controlled by Waveform Generator B.



INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.2: WAVE GENERATION USING TIMER1

Wave generation in Normal and CTC modes

When Timer1 is in CTC (WGM13:0 = 0100 or WGM13:0 = 1100) or Normal (WGM13:0 = 0000) mode after a compare match occurs, the waveform generators can perform one of the following actions, depending on the values of COM1A1:0 and COM1B1:0 bits, respectively:

- Remain unaffected
- Toggle the OC1x pin (OC1A or OC1B)
- Clear (drive low) the OC1x pin
- Set (drive high) the OC1x pin

The COM1A1 and COM1A0 bits select the operation of OC1A, while COM1B1 and COM1B0 select the operation of OC1B.

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

Example 15-14

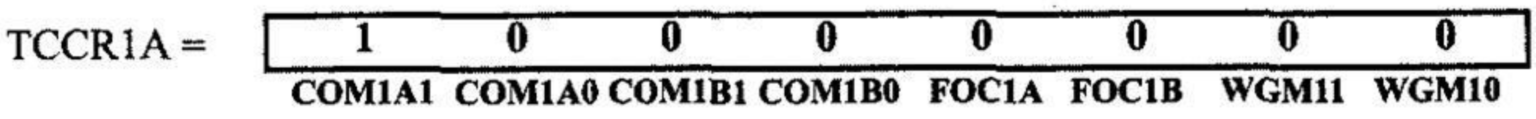
Using Figures 15-8 and 15-14, find the values of the TCCR1A and TCCR1B registers if we want to clear the OC1A pin upon match, with no prescaler, internal clock, and Normal mode.

Solution:

WGM13:10 = 0000 = Normal mode

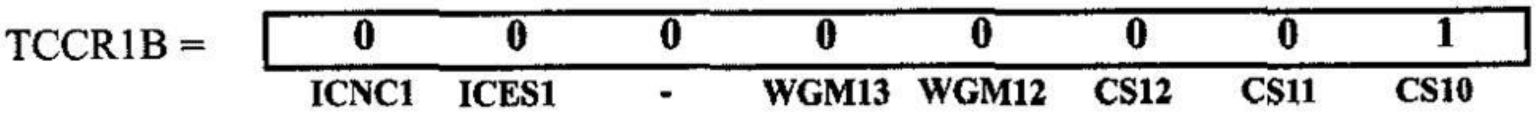
COM1A 1 :0 = 10 = Clear

CS 12:10 = 001 = No prescaler



TCCR1A =

1	0	0	0	0	0	0	0
COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10



TCCR1B =

0	0	0	0	0	0	0	1
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

Waveform generators in Timer1

In examining Figures 15-12 and 15-13 we see that Timer1 has two independent waveform generators: Waveform Generator A and Waveform Generator B.

Example 15-15

Find the value for TCCR1A and TCCR1B to program Timer1 as Normal mode and the OC1A generator as square wave generator and no prescaler.

Solution:

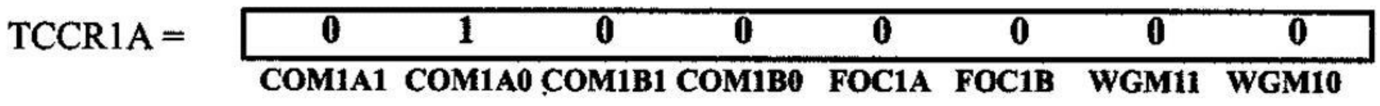
WGM13:10 = 0000 = Normal mode

COM1A1:0 = 01 = Toggle

CS12 :10 = 001 = No prescaler

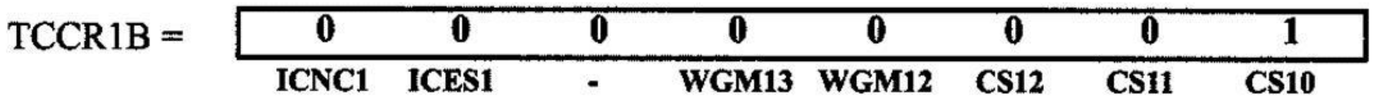
FOC1A = 1

FOC1B = 1



TCCR1A =

0	1	0	0	0	0	0	0
COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10



TCCR1B =

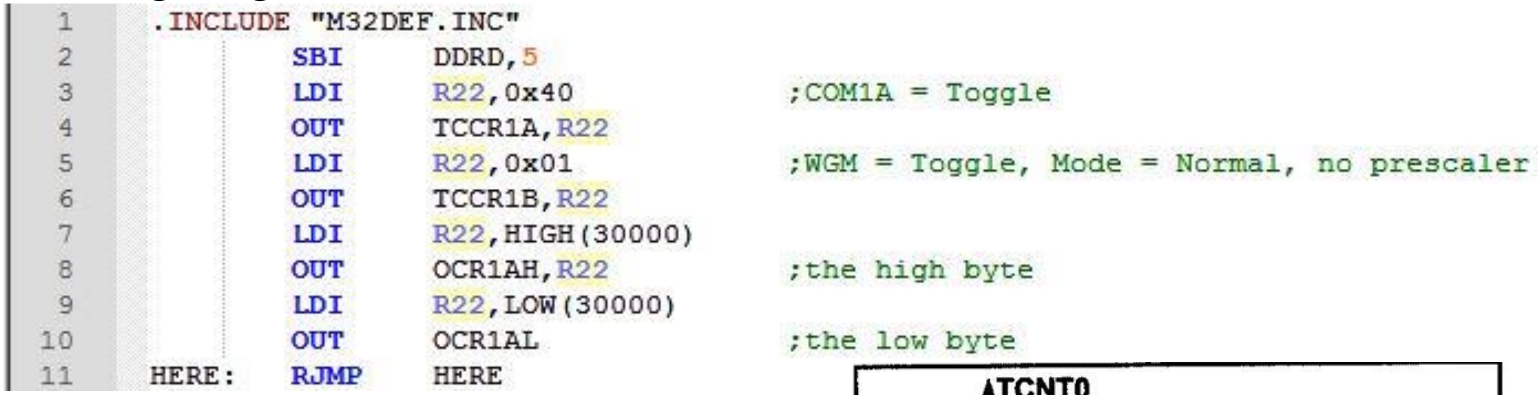
0	0	0	0	0	0	0	1
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.2: WAVE GENERATION USING TIMER1

Example 15-16

Assuming XTAL = 8 MHz, calculate the frequency of the wave generated by the following program:



```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,5
        LDI     R22,0x40              ;COM1A = Toggle
        OUT     TCCR1A,R22
        LDI     R22,0x01              ;WGM = Toggle, Mode = Normal, no prescaler
        OUT     TCCR1B,R22
        LDI     R22,HIGH(30000)
        OUT     OCR1AH,R22            ;the high byte
        LDI     R22,LOW(30000)
        OUT     OCR1AL                ;the low byte
HERE:   RJMP    HERE
```

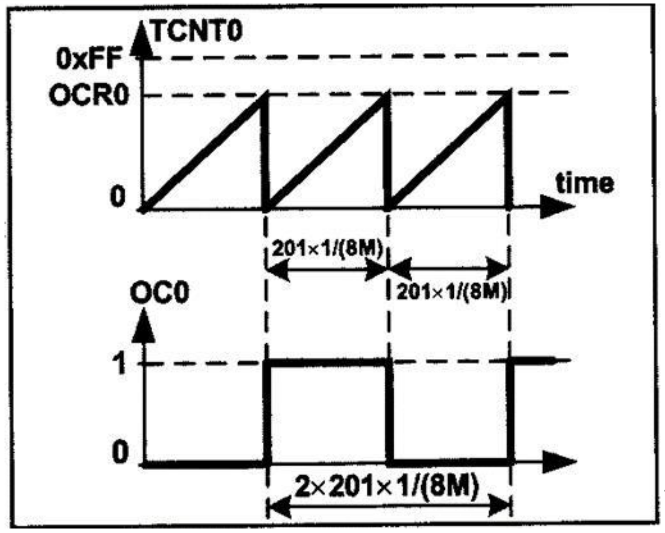
Solution:

From one compare match to the next one
 it takes 65,536 clocks and

$T_{\text{timer clock}} = 1/8\text{ MHz} = 0.125\ \mu s$

$T_{\text{wave}} = 2 \times 65{,}536 \times 0.125\ \mu s = 16{,}384\ \mu s$

$F_{\text{wave}} = 1/16{,}384\ \mu s = 61.035\text{ Hz}$



INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

CTC mode is better than Normal mode for generating square waves, as the frequency of the wave can be easily adjusted by changing the top value In CTC mode, when OCR1x has a lower value, compare match occurs earlier and the period of the generated wave is smaller (higher frequency).

Example 15-17

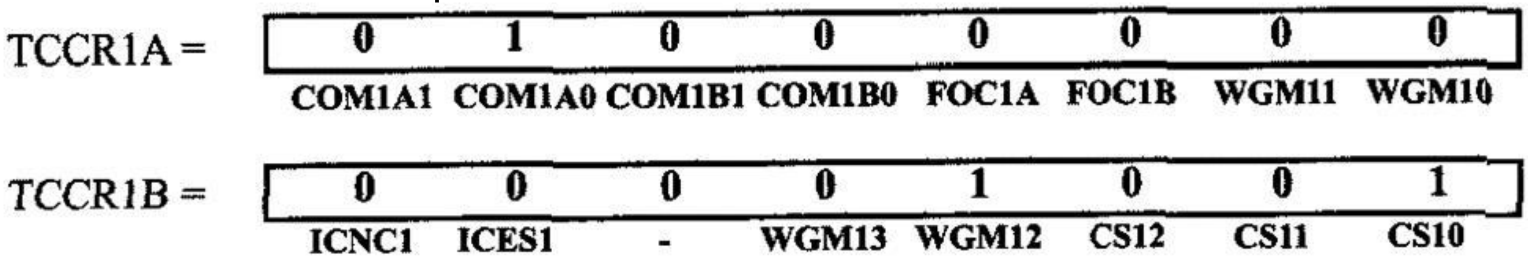
Find the value for TCCR1A and TCCR1B to program Timer1 as CTC mode and the OC1A generator as square wave generator and no prescaler.

Solution:

WGM13:10 = 0100 = CTC

COM1A1:0 = 01 = toggle

CS 12:10 = 001 = no prescaler



TCCR1A =	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
	0	1	0	0	0	0	0	0

TCCR1B =	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
	0	0	0	0	1	0	0	1

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.2: WAVE GENERATION USING TIMER1

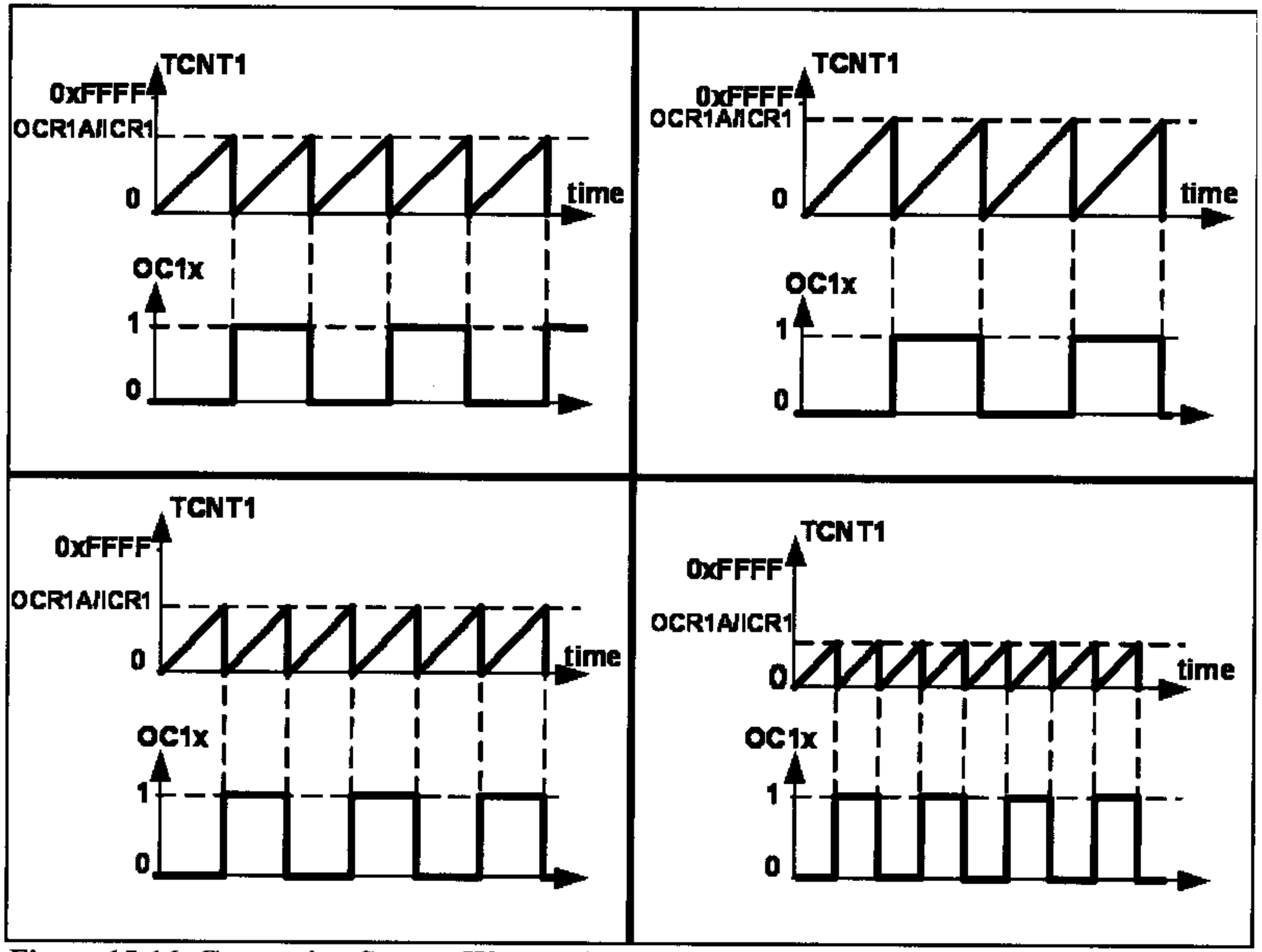


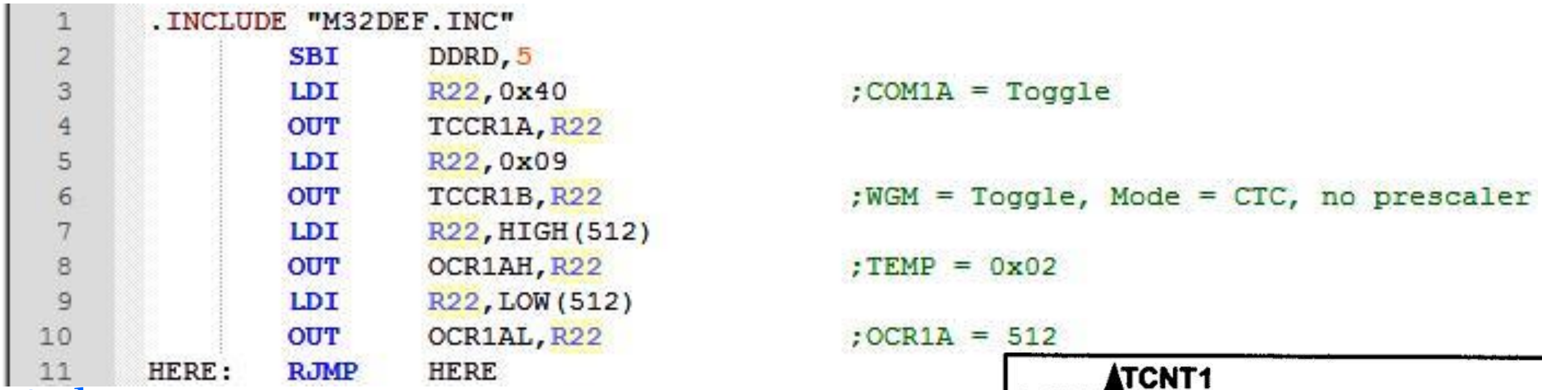
Figure 15-16. Generating Square Wave Using CTC Mode and Toggle Mode

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.2: WAVE GENERATION USING TIMER1

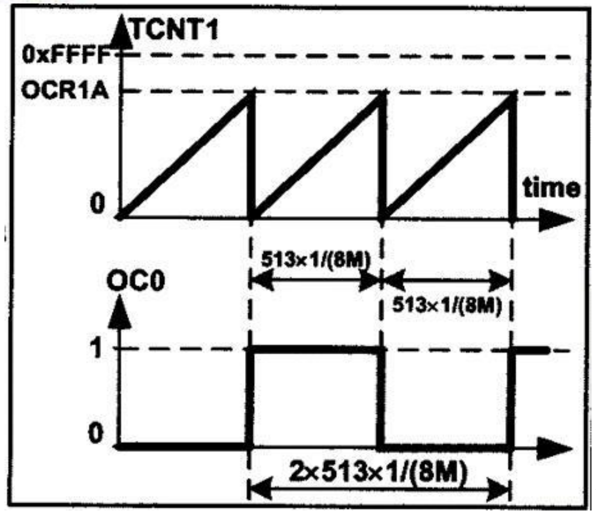
Example 15-18

Assuming XTAL = 8 MHz, calculate the frequency of the wave generated by the following program:



```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,5
        LDI     R22,0x40                ;COM1A = Toggle
        OUT     TCCR1A,R22
        LDI     R22,0x09
        OUT     TCCR1B,R22              ;WGM = Toggle, Mode = CTC, no prescaler
        LDI     R22,HIGH(512)
        OUT     OCR1AH,R22              ;TEMP = 0x02
        LDI     R22,LOW(512)
        OUT     OCR1AL,R22              ;OCR1A = 512
HERE:   RJMP    HERE
```

Solution:



From one compare match to the next one it takes 512 + 1 = 513 clocks and

$T_{\text{timer clock}} = 1 \ / \ 8 \text{ MHz} = 0.125 \ \mu s$

$T_{\text{wave}} = 2 \times 513 \times 0.125 \ \mu s = 128.25 \ \mu s$

$F_{\text{wave}} = 1 \ / \ 128.25 \ \mu s = 7797 \text{ Hz} = 7.797 \text{ kHz}$

SECTION 15.2: WAVE GENERATION USING TIMER1

Example 15-19

In Example 15-18, calculate the frequency of the Wave generated in each of the fallowing cases:

(a) OCR1A is loaded with 0x0500

(b) XTAL =1 MHz and OCR1A is loaded With 0x5

(c) a prescaler option of 8 is chosen, XTAL = 4 MHz, OCR = 0x150

(d) a prescaler option of N is chosen, XTAL = Fosc, OCR1A = X

Solution:

(a) 0x500+1=0x501=1281 clocks and $T_{\text{timer clock}}$=0.125 μs ⇒ T_{wave}=2×1281×0.125μs=320.25 μs ⇒ F_{wave}=1/320.25μs=3122.56 Hz

(b) 5+1=6 clocks and $T_{\text{timer clock}}$ =1/1 MHz= 1μs⇒ T_{wave}=2×6×1μs=12 μs ⇒ F_{wave}=1/12 μs= 83,333 Hz = 83.333 kHz

(c) 0x150+1=0x151=337 clocks and $T_{\text{Timer clock}}$=8×1/4 MHz= 2μs ⇒ T_{wave}=2×337×2μs=1348μs ⇒ F_{wave}=1/1348μs=741.8 Hz

(d) X+1clocks and $T_{\text{Timer clock}}$=N×1/F_{OSC}=N/Fosc ⇒ T_{wave}=2×(X+1)×N/F_{osc} ⇒F_{wave}=1/T_{waveFosc}/[2N(X+1)]

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

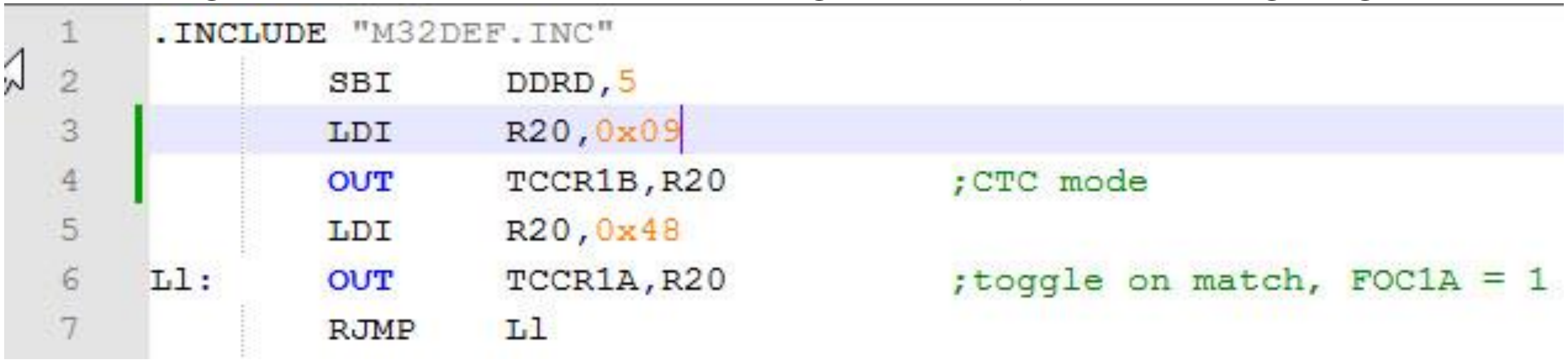
FOC1A (Force Output Compare) and FOC1B flags

Writing 1 to the FOC1A bit of the TCCR1A register forces the Waveform Generator A to act as if a compare match has occurred. Writing 1 to the FOC1B bit of the TCCR1B register forces Waveform Generator B to act as if a compare match has occurred.

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.2: WAVE GENERATION USING TIMER1

Example 15-20

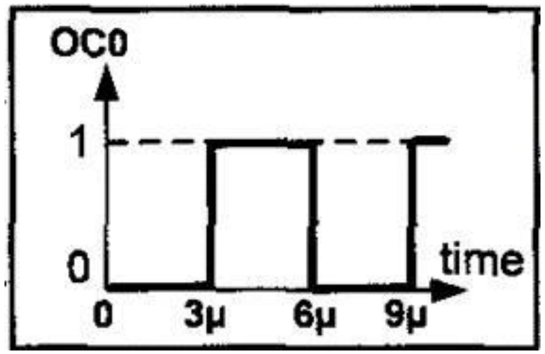
Assuming XTAL = 1 MHz, draw the wave generated by the following program:



```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,5
        LDI     R20,0x09
        OUT     TCCR1B,R20          ;CTC mode
        LDI     R20,0x48
L1:     OUT     TCCR1A,R20          ;toggle on match, FOC1A = 1
        RJMP    L1
```

Solution:

The wave generator is in toggle mode. So, it toggles on compare match. Setting the FOC1A bit causes the wave generator to act as if the compare match has occurred. So, the OC1A pin toggles. The execution of instructions "OUT TCCR1A, R20" and "RJMP L1" takes 1 and 2 clocks, respectively. So, toggle occurs after 1 + 2 = 3 clocks



INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING

INPUT CAPTURE PROGRAMMING

The Input Capture function is widely used for many applications. Among them are

(a) recording the arrival time of an event,
(b) pulse width measurement, and
(c) period measurement.

In ATmega32, Timer1 can be used as the Input Capture to detect and measure the events happening outside the chip. Upon detection of an event, the TCNT value is loaded into the ICR1 register, and the ICF1 flag is set.

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING

As shown in Figure 15-17, there are two event sources: (1) the ICP1 pin, which is PORTD.6 in ATmega32, and (2) the output of the analog comparator. We can use the ACIC flag to select the event source. ACIC is a bit of the ACSR register, as shown in Figure 15-18.

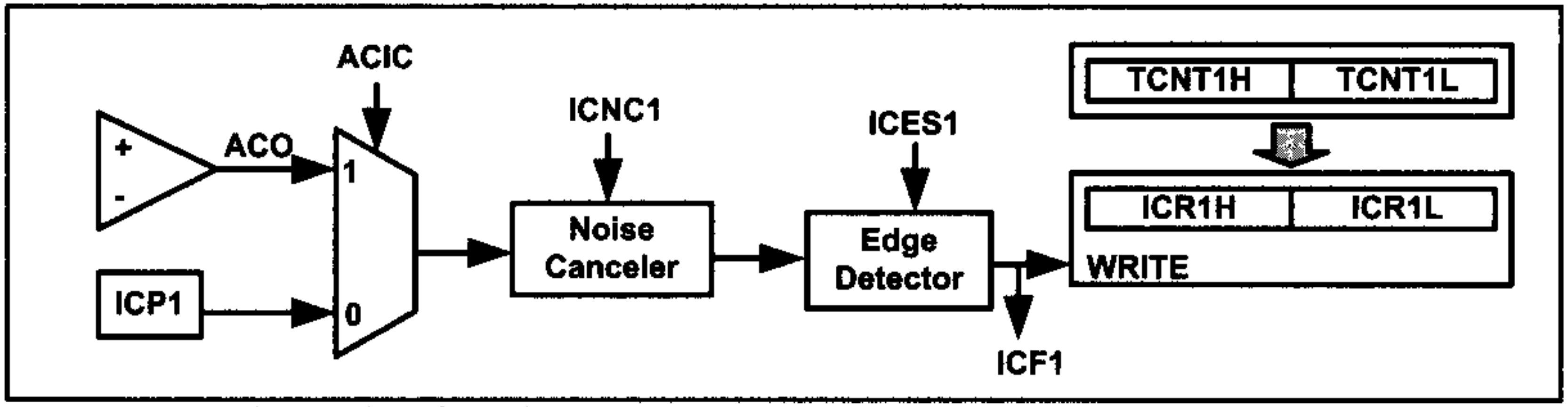
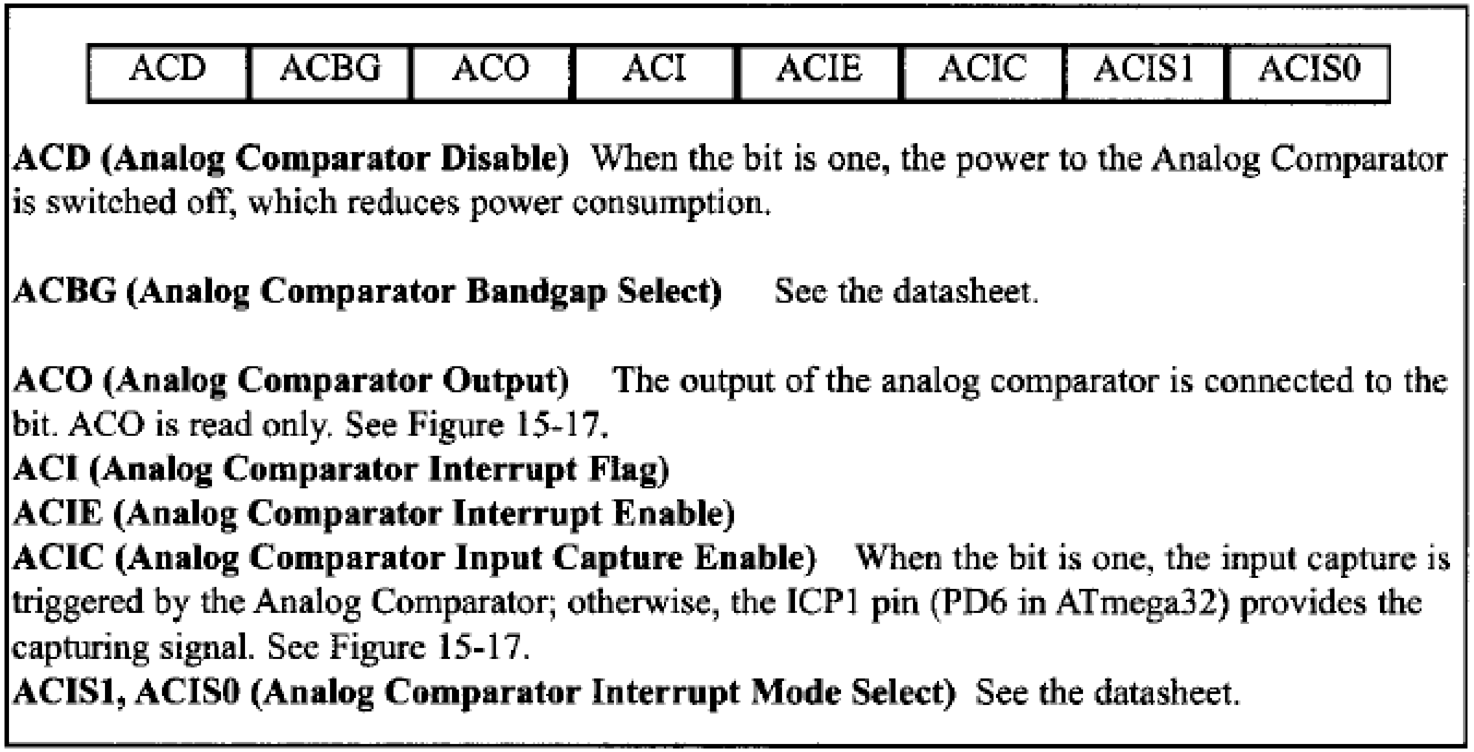


Figure 15-17. Capturing Circuit

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.3: INPUT CAPTURE PROGRAMMING



ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0

ACD (Analog Comparator Disable) When the bit is one, the power to the Analog Comparator is switched off, which reduces power consumption.

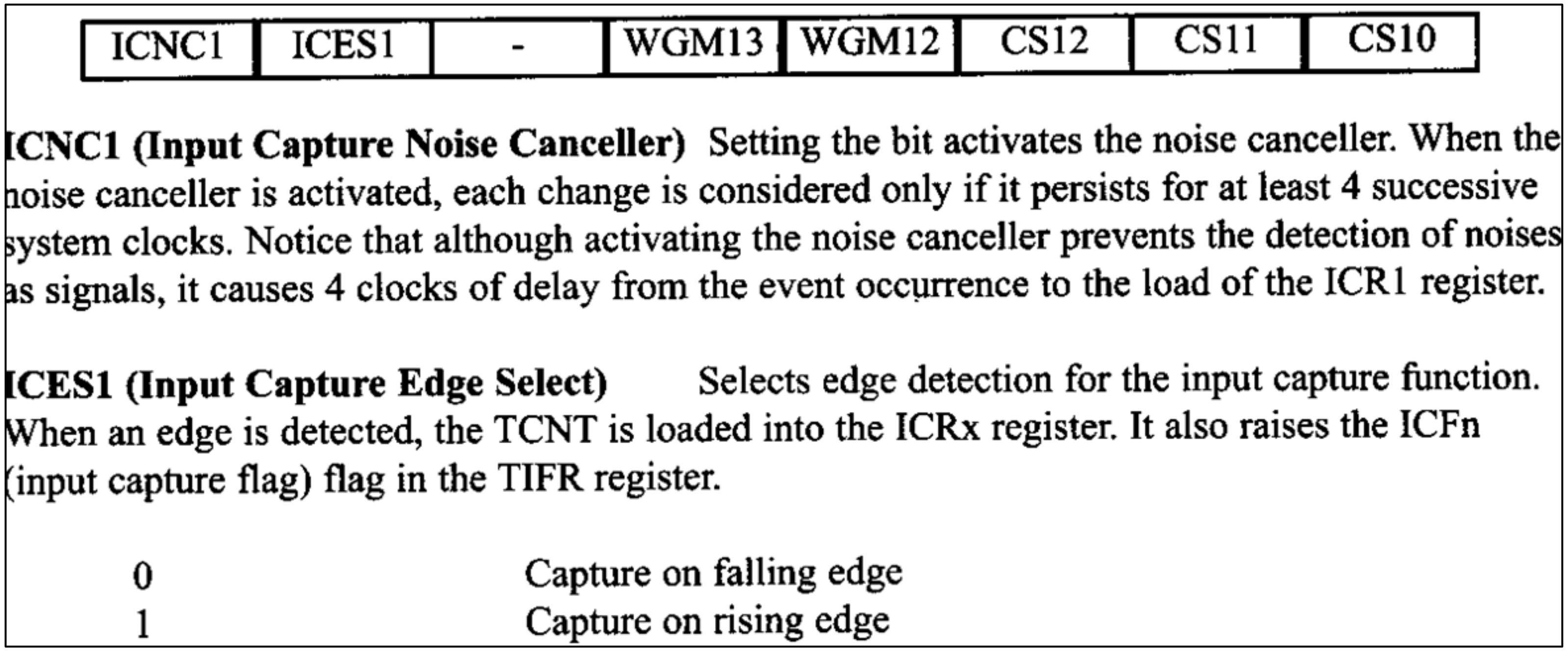
ACBG (Analog Comparator Bandgap Select) See the datasheet.

ACO (Analog Comparator Output) The output of the analog comparator is connected to the bit. ACO is read only. See Figure 15-17.
ACI (Analog Comparator Interrupt Flag)
ACIE (Analog Comparator Interrupt Enable)
ACIC (Analog Comparator Input Capture Enable) When the bit is one, the input capture is triggered by the Analog Comparator; otherwise, the ICP1 pin (PD6 in ATmega32) provides the capturing signal. See Figure 15-17.
ACIS1, ACIS0 (Analog Comparator Interrupt Mode Select) See the datasheet.

Fig 15-18 Analog Comparator Control and Status Register - ACSR

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING



ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

ICNC1 (Input Capture Noise Canceller) Setting the bit activates the noise canceller. When the noise canceller is activated, each change is considered only if it persists for at least 4 successive system clocks. Notice that although activating the noise canceller prevents the detection of noises as signals, it causes 4 clocks of delay from the event occurrence to the load of the ICR1 register.

ICES1 (Input Capture Edge Select) Selects edge detection for the input capture function. When an edge is detected, the TCNT is loaded into the ICRx register. It also raises the ICFn (input capture flag) flag in the TIFR register.

0	Capture on falling edge
1	Capture on rising edge

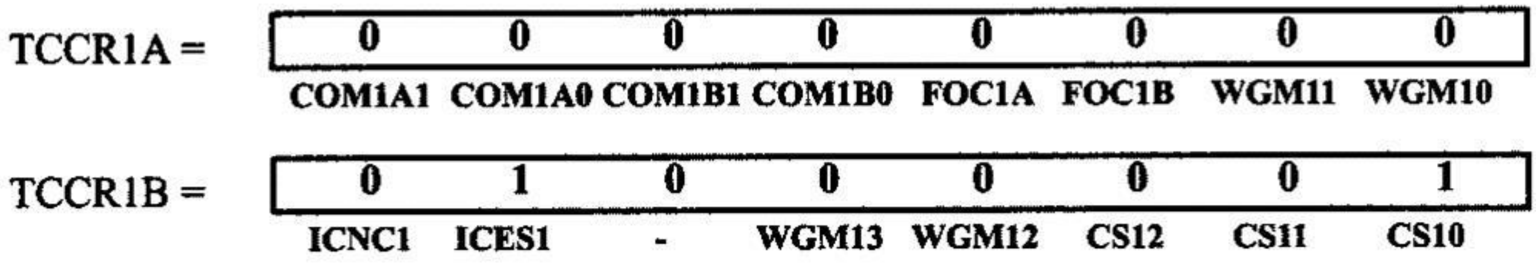
Fig 15-19 TCCR1B (Timer/Counter Control Register) Register ICNC1, ICES1

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.3: INPUT CAPTURE PROGRAMMING

Example 15-21

Using Figures 15-12 and 15-19, find TCCR1A and TCCRIB, for capturing on rising edge, no noise canceller, no prescaler, and timer mode = Normal.

Solution:



TCCR1A =

0	0	0	0	0	0	0	0
COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

TCCR1B =

0	1	0	0	0	0	0	1
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.3: INPUT CAPTURE PROGRAMMING

Steps to program the Input Capture function

We use the following steps to measure the edge arrival time for the Input Capture function.

1. Initialize the TCCR1A and TCCR1B for a proper timer mode (any mode other than modes 8, 10, 12, and 14), enable or disable the noise canceller, and select the edge (positive or negative) we want to measure the arrival time for.
2. Initialize the ACSR to select the desired event source.
3. Monitor the ICF1 flag in TIFR to see if the edge has arrived. Upon the arrival of the edge, the TCNT1 value is loaded into the ICR1 register automatically by the AVR.

The Input Capture function is widely used to measure the period or the pulse width of an incoming signal

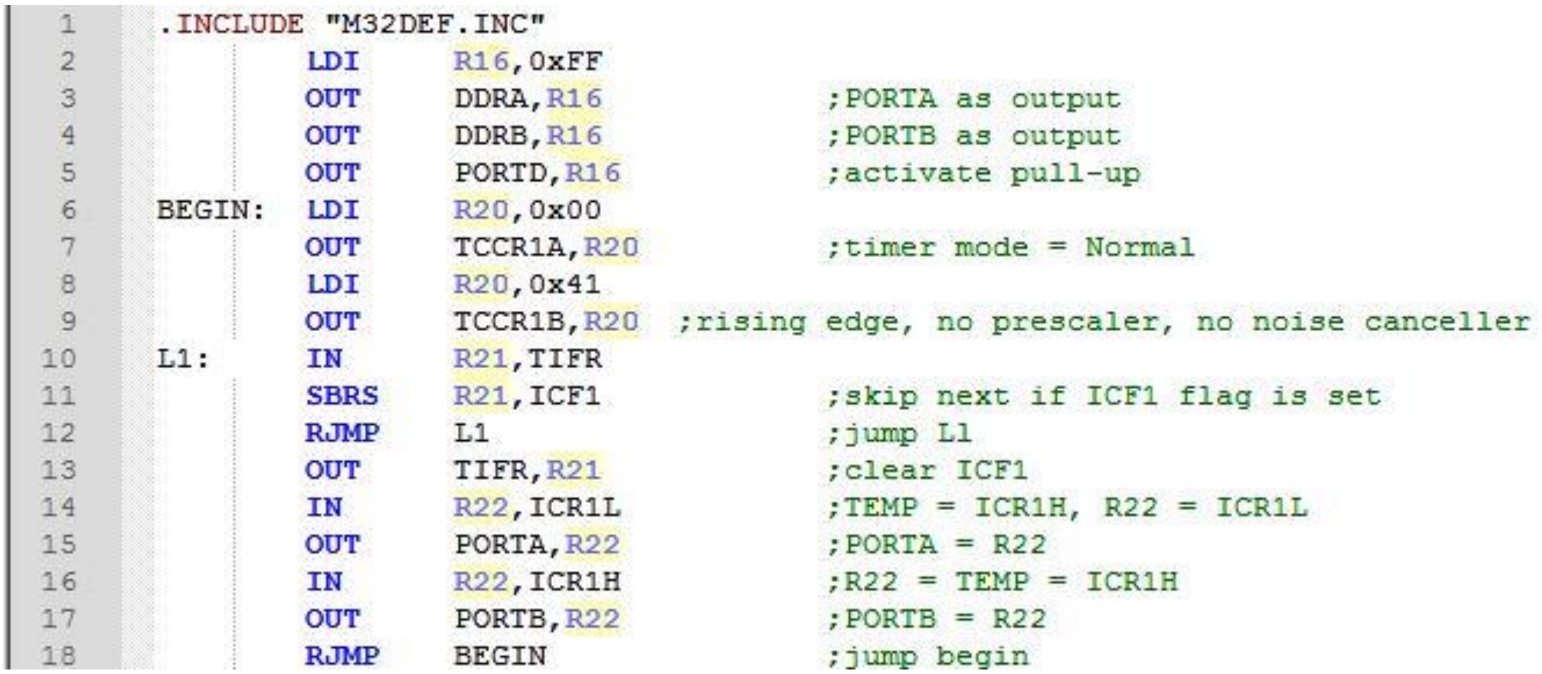
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING

Example 15-22

Assuming that clock pulses are fed into pin ICP1, write a program to read the TCNT1 value on every rising edge. Place the result on PORTA and PORTB.

Solution:



```
.INCLUDE "M32DEF.INC"
        LDI     R16,0xFF
        OUT     DDRA,R16            ;PORTA as output
        OUT     DDRB,R16            ;PORTB as output
        OUT     PORTD,R16           ;activate pull-up
BEGIN:  LDI     R20,0x00
        OUT     TCCR1A,R20          ;timer mode = Normal
        LDI     R20,0x41
        OUT     TCCR1B,R20   ;rising edge, no prescaler, no noise canceller
L1:     IN      R21,TIFR
        SBRS    R21,ICF1            ;skip next if ICF1 flag is set
        RJMP    L1                  ;jump L1
        OUT     TIFR,R21            ;clear ICF1
        IN      R22,ICR1L           ;TEMP = ICR1H, R22 = ICR1L
        OUT     PORTA,R22           ;PORTA = R22
        IN      R22,ICR1H           ;R22 = TEMP = ICR1H
        OUT     PORTB,R22           ;PORTB = R22
        RJMP    BEGIN               ;jump begin
```

Note: Upon the detection of each rising edge, the TCNT1 value is loaded into ICR1. Also notice that we clear the ICF1 flag bit.

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.3: INPUT CAPTURE PROGRAMMING

Measuring period

We can use the following steps to measure the period of a wave.

1. Initialize the TCCR1A and TCCR1B.
2. Initialize the ACSR to select the desired event source.
3. Monitor the ICF1 flag in TIFR to see if the edge has arrived. Upon the arrival of the edge, the TCNT1 is loaded into the ICR1 register automatically by the AVR.
4. Save the ICR1.
5. Monitor the ICF1 flag in TIFR to see if the second edge has arrived. Upon the arrival of the edge, the TCNT1 is loaded into the ICR1 register automatically by the AVR.
6. Save the ICR1 for the second edge. By subtracting the second edge value from the first edge value we get the time.

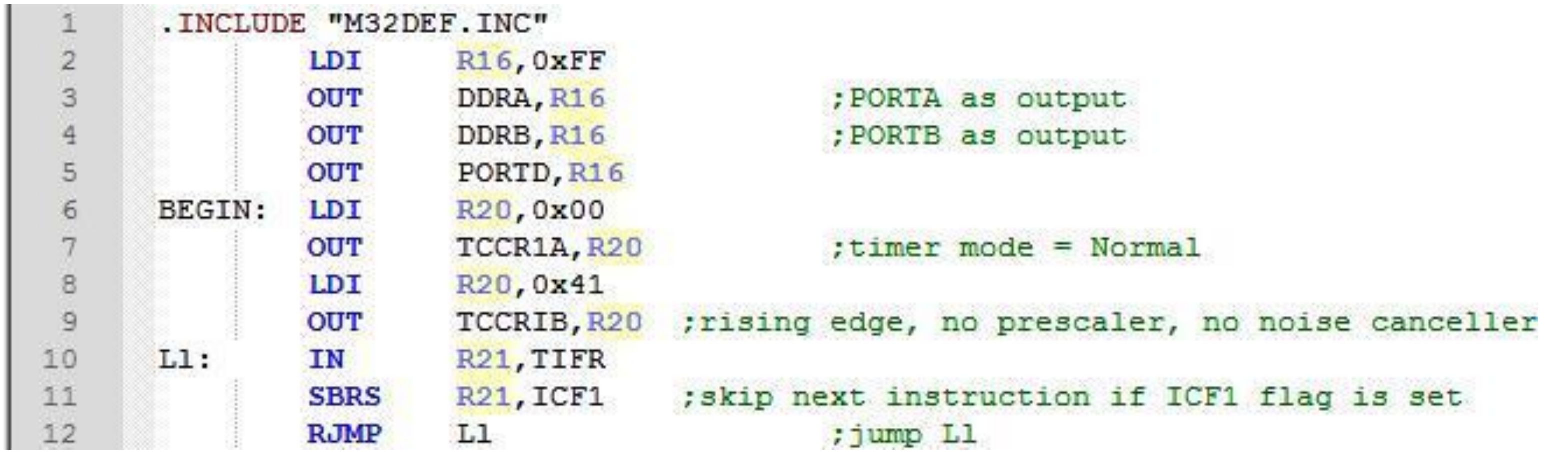
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING

Example 15-23

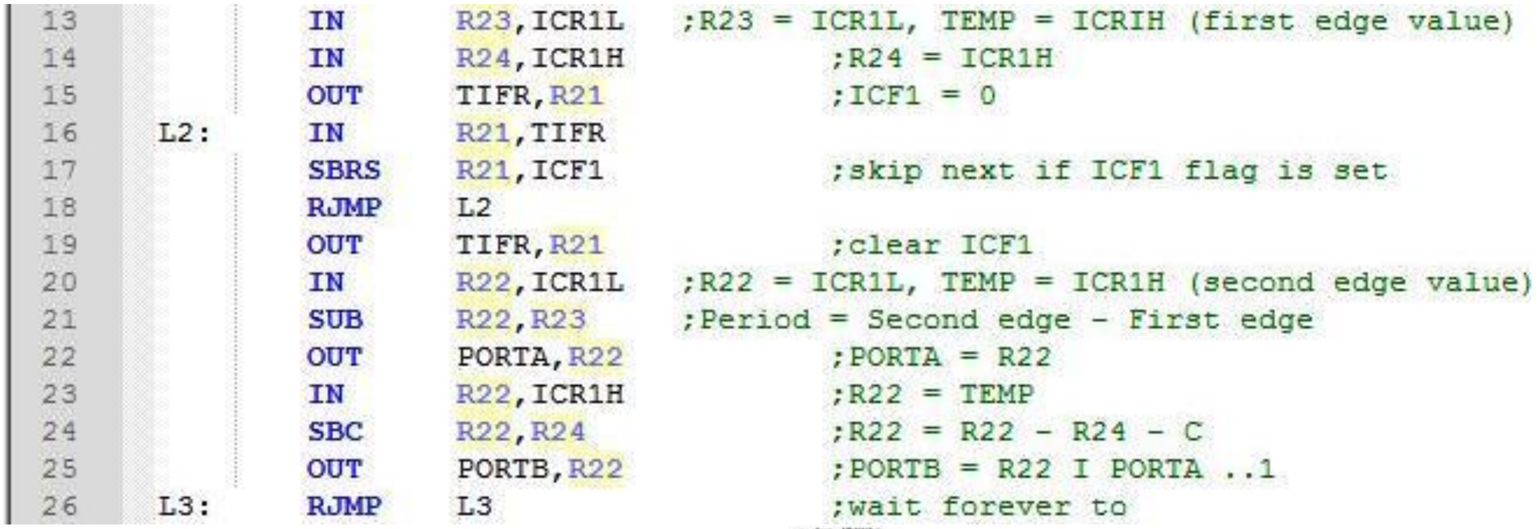
Assuming that clock pulses are fed into pin PORTD.6, write a program to measure the period of the pulses, Place the binary result on PORTA and PORTB.

Solution:

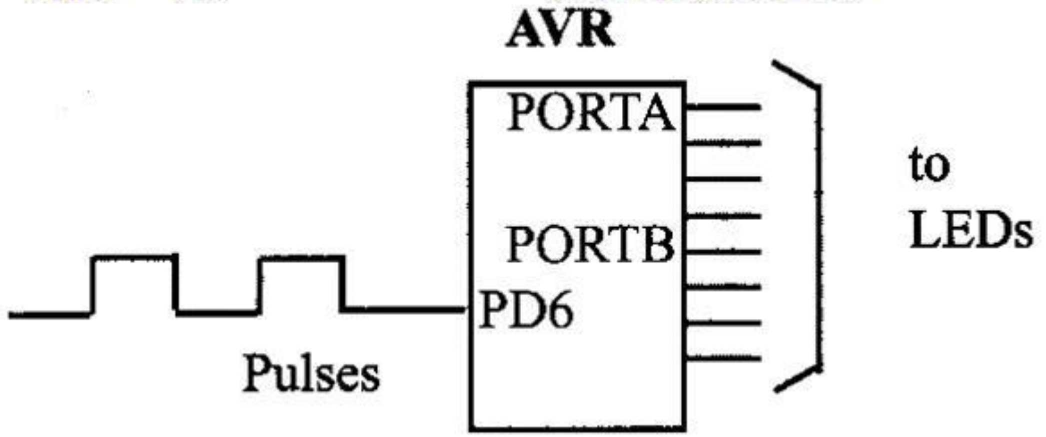


```
.INCLUDE "M32DEF.INC"
        LDI     R16,0xFF
        OUT     DDRA,R16            ;PORTA as output
        OUT     DDRB,R16            ;PORTB as output
        OUT     PORTD,R16
BEGIN:  LDI     R20,0x00
        OUT     TCCR1A,R20          ;timer mode = Normal
        LDI     R20,0x41
        OUT     TCCRIB,R20  ;rising edge, no prescaler, no noise canceller
L1:     IN      R21,TIFR
        SBRS    R21,ICF1    ;skip next instruction if ICF1 flag is set
        RJMP    L1                  ;jump L1
```

SECTION 15.3: INPUT CAPTURE PROGRAMMING



```
            IN      R23,ICR1L   ;R23 = ICR1L, TEMP = ICRIH (first edge value)
            IN      R24,ICR1H           ;R24 = ICR1H
            OUT     TIFR,R21            ;ICF1 = 0
    L2:     IN      R21,TIFR
            SBRS    R21,ICF1            ;skip next if ICF1 flag is set
            RJMP    L2
            OUT     TIFR,R21            ;clear ICF1
            IN      R22,ICR1L   ;R22 = ICR1L, TEMP = ICR1H (second edge value)
            SUB     R22,R23     ;Period = Second edge - First edge
            OUT     PORTA,R22           ;PORTA = R22
            IN      R22,ICR1H           ;R22 = TEMP
            SBC     R22,R24             ;R22 = R22 - R24 - C
            OUT     PORTB,R22           ;PORTB = R22 I PORTA ..1
    L3:     RJMP    L3                  ;wait forever to
```



INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING

Using Input Capture to Measure Period

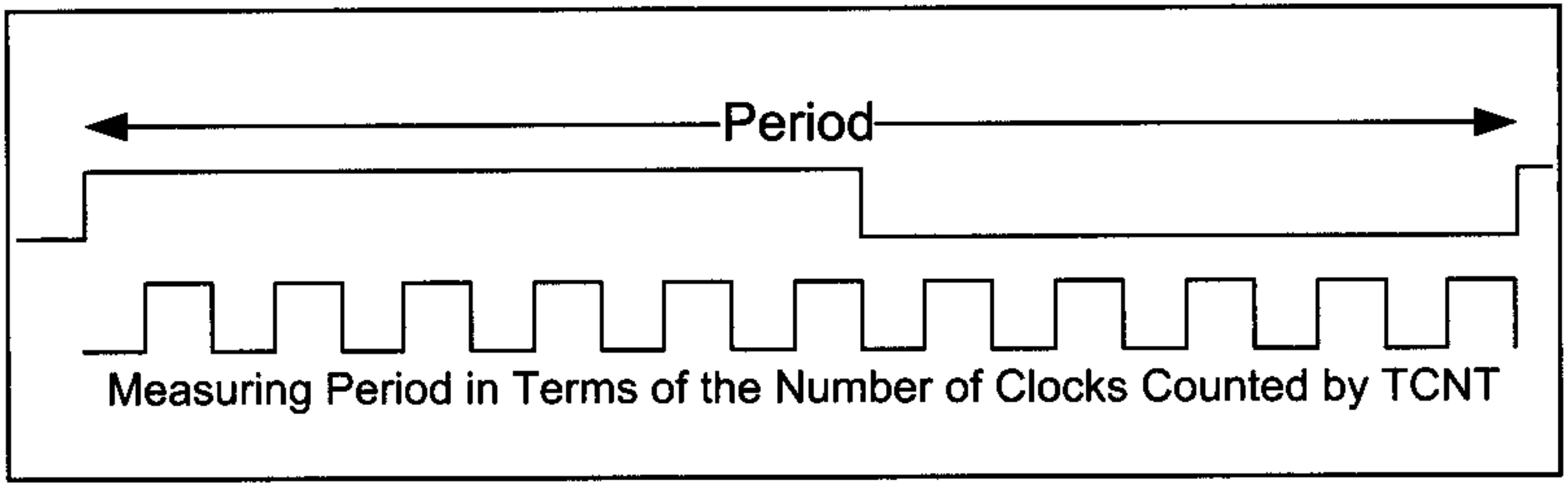

Period
Measuring Period in Terms of the Number of Clocks Counted by TCNT

Figure 15-20. Using Input Capture to Measure Period

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING

Example 15-24

The frequency of a pulse is between 50 Hz and 60 Hz. Assume that a pulse is connected to ICP1 (pin PD6). Write a program to measure its period and display it on PORTB. Use the prescaler value that gives the result in a single byte. Assume XTAL = 8 MHz.

Solution:

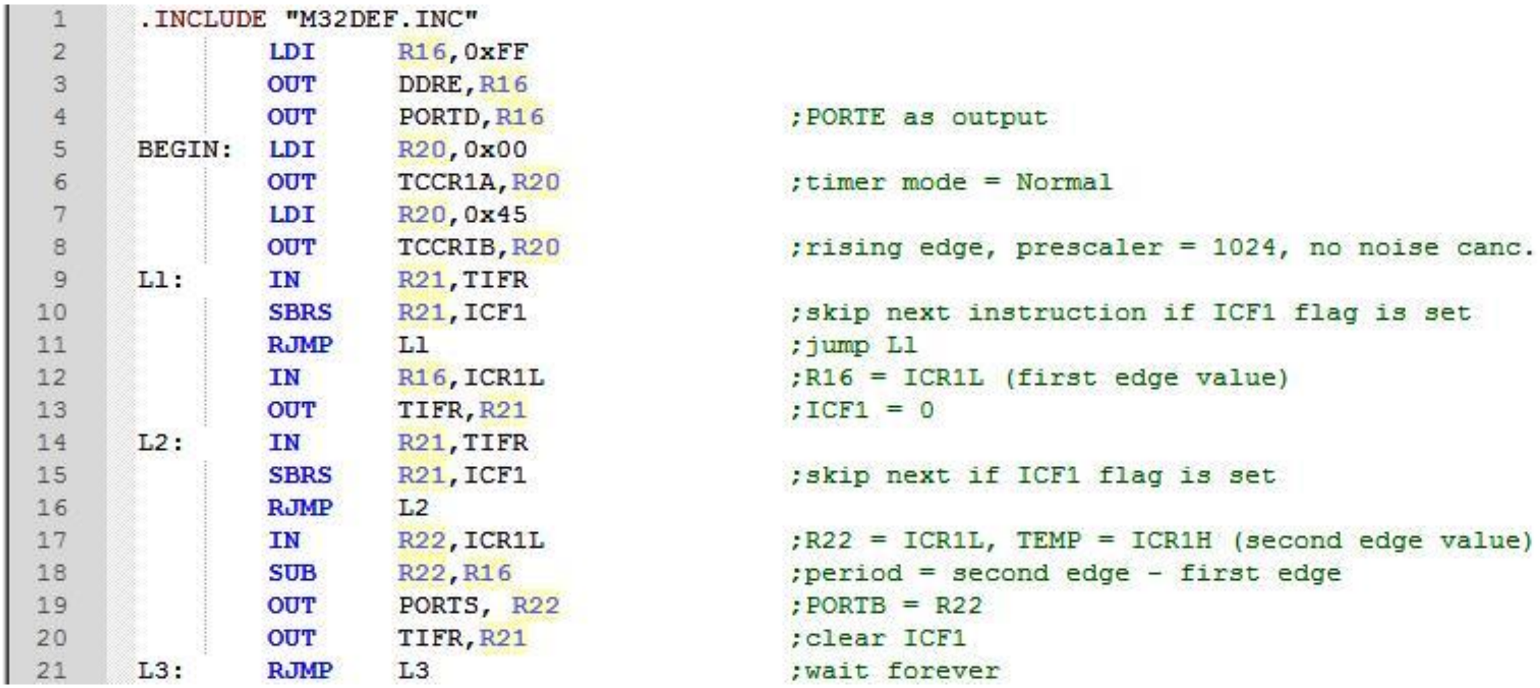
8MHz ×1/1024 = 7812.5 Hz due to prescaler and T = 1/7812.5 Hz = 128 µs.

The frequency of 50 Hz gives us the period of 1/50 Hz = 20 ms. So, the output is 20 ms/128 µs = 156.

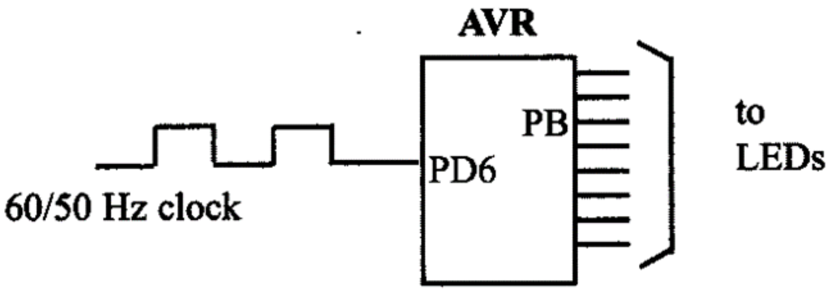
The frequency of 60 Hz gives us the period of 1/60 Hz = 16.6 ms. So, the output is 16.6 ms/128 µs = 130.

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING



```
.INCLUDE "M32DEF.INC"
        LDI     R16,0xFF
        OUT     DDRE,R16
        OUT     PORTD,R16                   ;PORTE as output
BEGIN:  LDI     R20,0x00
        OUT     TCCR1A,R20                  ;timer mode = Normal
        LDI     R20,0x45
        OUT     TCCRIB,R20                  ;rising edge, prescaler = 1024, no noise canc.
L1:     IN      R21,TIFR
        SBRS    R21,ICF1                    ;skip next instruction if ICF1 flag is set
        RJMP    L1                          ;jump L1
        IN      R16,ICR1L                   ;R16 = ICR1L (first edge value)
        OUT     TIFR,R21                    ;ICF1 = 0
L2:     IN      R21,TIFR
        SBRS    R21,ICF1                    ;skip next if ICF1 flag is set
        RJMP    L2
        IN      R22,ICR1L                   ;R22 = ICR1L, TEMP = ICR1H (second edge value)
        SUB     R22,R16                     ;period = second edge - first edge
        OUT     PORTS, R22                  ;PORTB = R22
        OUT     TIFR,R21                    ;clear ICF1
L3:     RJMP    L3                          ;wait forever
```



INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING

Measuring pulse width

We can use the following steps to measure the pulse width of a wave.

1. Initialize TCCR1A and TCCR1B, and select capturing on rising edge.
2. Initialize ACSR to select the desired event source.
3. Monitor the ICF1 flag in TIFR to see if the edge has arrived. Upon the arrival of the edge, the TCNT1 value is loaded into the ICR1 register automatically by the AVR.
4. Save the ICR1 and change the capturing edge to the falling edge.
5. Monitor the ICF1 flag in TIFR to see if the second edge has arrived. Upon the arrival of the edge, the TCNT1 value is loaded into the ICR1 register automatically by the AVR.
6. Save the ICRl for the second edge. Subtract the second edge value from the first edge value to get the time.

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING

Using Input Capture to Measure Period and Pulse Width

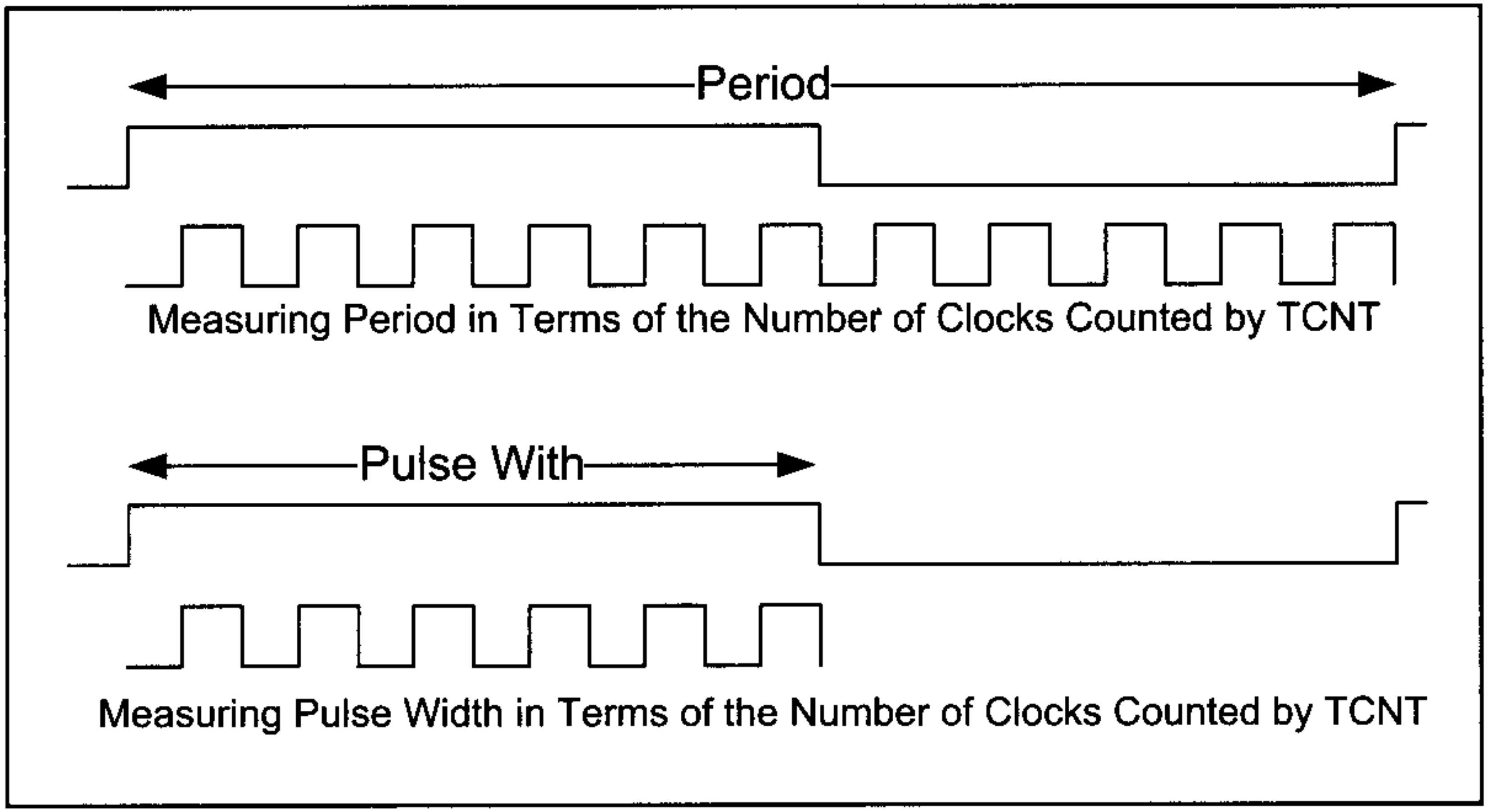

Period
Measuring Period in Terms of the Number of Clocks Counted by TCNT
Pulse With
Measuring Pulse Width in Terms of the Number of Clocks Counted by TCNT

Figure 15-21. Using Input Capture to Measure Period and Pulse Width

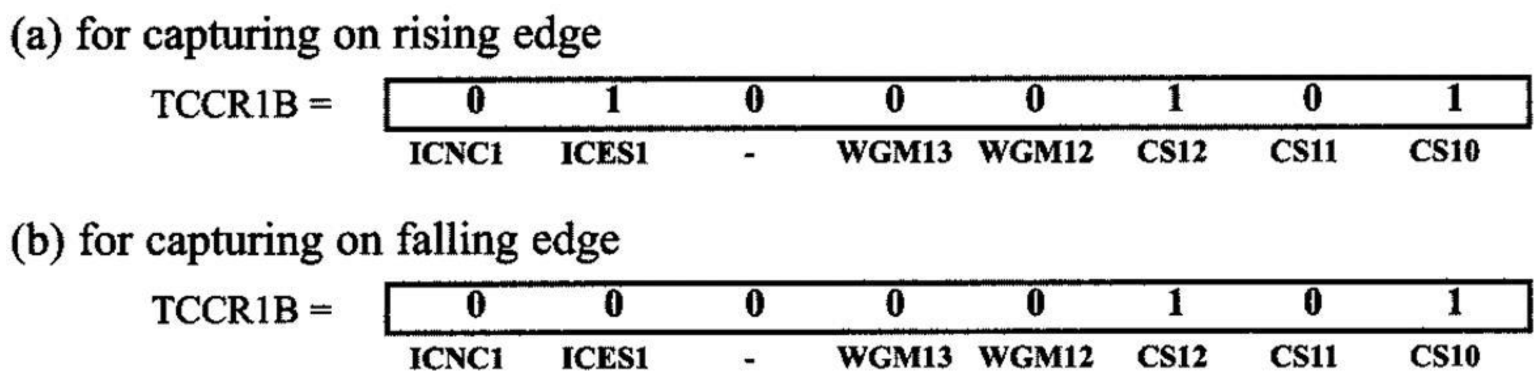
INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.3: INPUT CAPTURE PROGRAMMING

Waveform generators in Timer1

Example 15-25

Using Figure 15-19, find TCCR1 B for no noise canceller, prescaler = 1024, and timer in Normal mode: (a) for capturing on rising edge (b) for capturing on falling edge

Solution:



(a) for capturing on rising edge

TCCR1B =

0	1	0	0	0	1	0	1
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

(b) for capturing on falling edge

TCCR1B =

0	0	0	0	0	1	0	1
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.3: INPUT CAPTURE PROGRAMMING

Example 15-26

Assume that a 60-Hz frequency pulse is connected to ICP1 (pin PD6). Write a program to measure its pulse width. Use the prescaler value that gives the result in a single byte. Display the result on PORTB. Assume XTAL = 8 MHz.

Solution:

The frequency of 60 Hz gives us the period of 1/60 Hz = 16.6 ms.

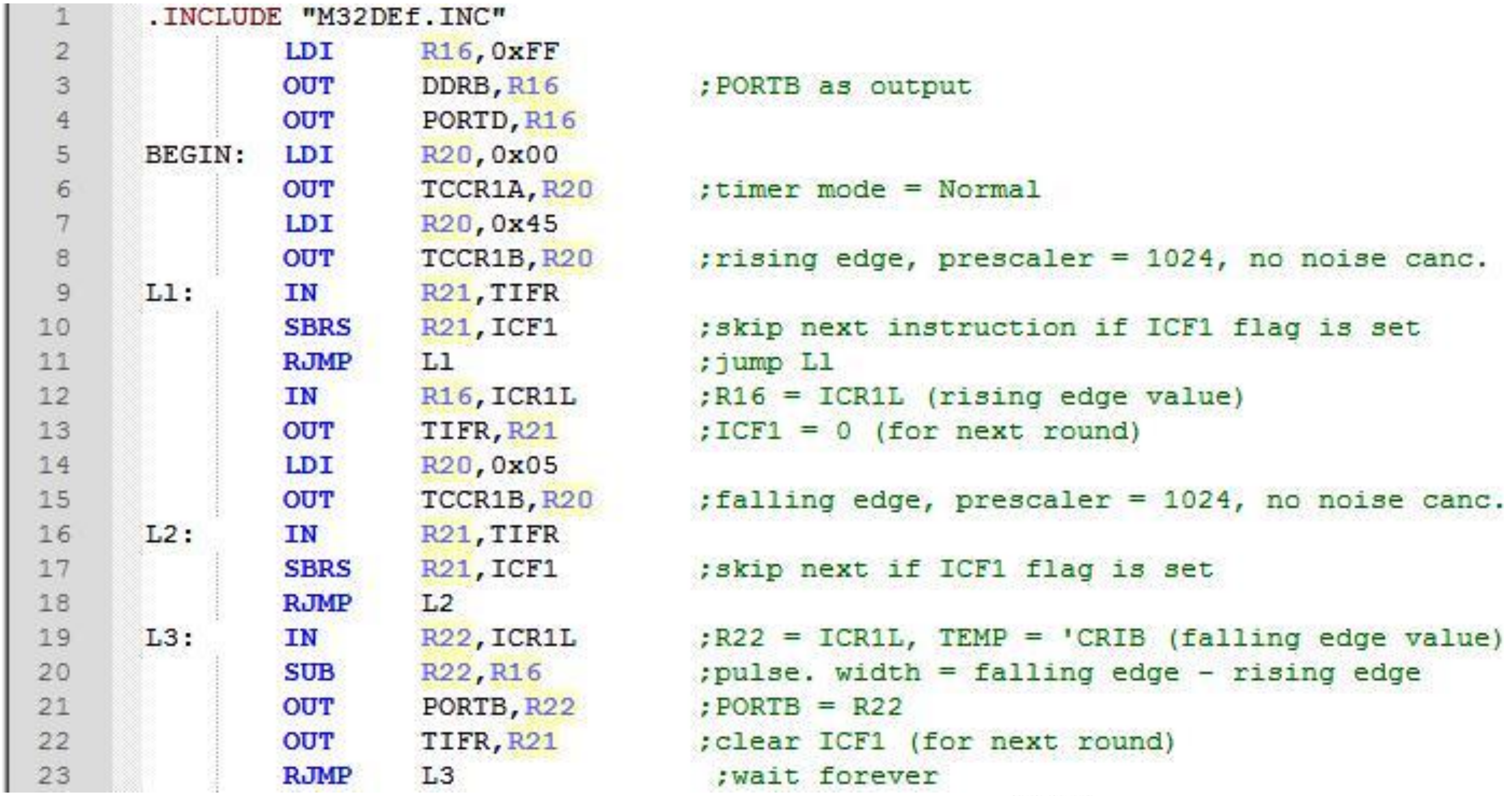
Now, 8 MHz × 1/1024 = 7812.5 Hz due to prescaler and

T = 1/7812.5 Hz = 128 μs for TCNT. That means we get the value of 130 (1000 0010 binary) for the period since 16.6 ms / 128 us = 130.

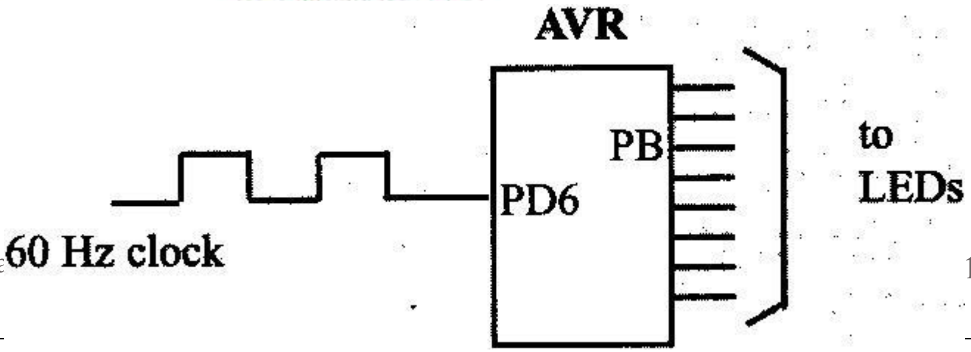
Now the pulse width can be anywhere between 1 to 129.

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING



```
.INCLUDE "M32DEf.INC"
        LDI     R16,0xFF
        OUT     DDRB,R16        ;PORTB as output
        OUT     PORTD,R16
BEGIN:  LDI     R20,0x00
        OUT     TCCR1A,R20      ;timer mode = Normal
        LDI     R20,0x45
        OUT     TCCR1B,R20      ;rising edge, prescaler = 1024, no noise canc.
L1:     IN      R21,TIFR
        SBRS    R21,ICF1        ;skip next instruction if ICF1 flag is set
        RJMP    L1              ;jump L1
        IN      R16,ICR1L       ;R16 = ICR1L (rising edge value)
        OUT     TIFR,R21        ;ICF1 = 0 (for next round)
        LDI     R20,0x05
        OUT     TCCR1B,R20      ;falling edge, prescaler = 1024, no noise canc.
L2:     IN      R21,TIFR
        SBRS    R21,ICF1        ;skip next if ICF1 flag is set
        RJMP    L2
L3:     IN      R22,ICR1L       ;R22 = ICR1L, TEMP = 'CRIB (falling edge value)
        SUB     R22,R16         ;pulse. width = falling edge - rising edge
        OUT     PORTB,R22       ;PORTB = R22
        OUT     TIFR,R21        ;clear ICF1 (for next round)
        RJMP    L3               ;wait forever
```



INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.3: INPUT CAPTURE PROGRAMMING

Example 15-27

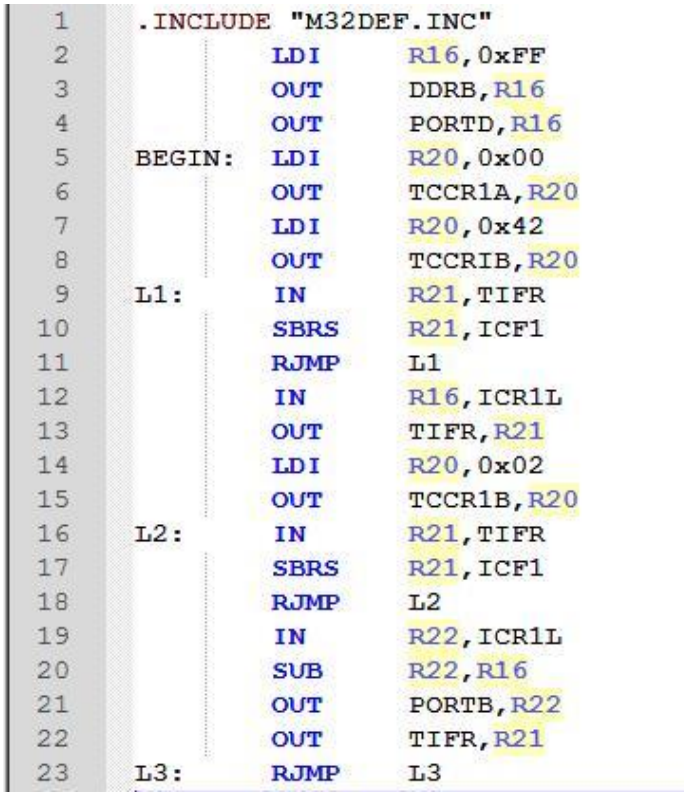
Assume that a temperature sensor is connected to pin PD6. The temperature provided by the sensor is proportional to pulse width and is in the range of 1µS to 250µs. Write a program to measure the temperature if 1 µs is equal to 1 degree. Use the prescaler value that gives the result in a single byte. Display the result on PORTB. Assume XTAL = 8 MHz.

Solution:

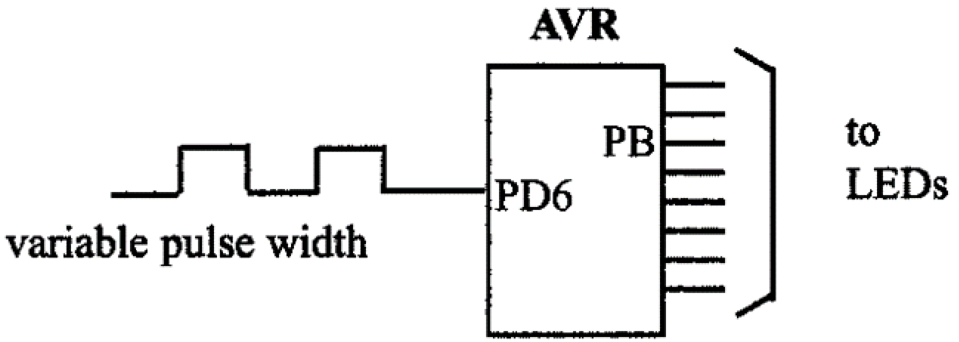
8 MHz ×1/8 = 1 MHz = 1,000,000 Hz due to prescaler and T = 1/1,000,000 Hz = 1 µs for TCNT. That means we get the values between 1 and 65,536 µs for the TCNT, but since the pulse width never goes beyond 250 µs we should be able to display the temperature value on PORTB.

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.3: INPUT CAPTURE PROGRAMMING



```
.INCLUDE "M32DEF.INC"
        LDI     R16,0xFF
        OUT     DDRB,R16            ;PORTB as output
        OUT     PORTD,R16
BEGIN:  LDI     R20,0x00
        OUT     TCCR1A,R20          ;timer mode = Normal
        LDI     R20,0x42
        OUT     TCCRIB,R20          ;rising edge, prescaler 8, no noise canceller
L1:     IN      R21,TIFR            ;stay here for ICP rising
        SBRS    R21,ICF1            ;skip next instruction if ICF1 flag is set
        RJMP    L1                  ;jump L1
        IN      R16,ICR1L           ;R16 = ICR1L
        OUT     TIFR,R21            ;ICF1 = 0
        LDI     R20,0x02
        OUT     TCCR1B,R20          ;falling edge, prescaler 8, no noise canceller
L2:     IN      R21,TIFR            ;stay here for ICP falling edge
        SBRS    R21,ICF1            ;skip next if ICF1 flag is set
        RJMP    L2
        IN      R22,ICR1L           ;R22 = ICR1L, TEMP = ICR1H
        SUB     R22,R16             ;period = falling edge - rising edge
        OUT     PORTB,R22
        OUT     TIFR,R21
L3:     RJMP    L3
```



INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.3: INPUT CAPTURE PROGRAMMING

Analog comparator

As shown in Figure 15-17, when the ACIC bit is set, the analog comparator provides the trigger signal for the input capture unit. The analog comparator is an op-amp that compares the voltage of AIN1 (PORTB.3 in ATmega32) with AIN0 (PORTB.2 in ATmega32).

If the voltage of AIN1 is higher than AIN0, the comparator's output is 1; otherwise, its output is 0. For more information, see the datasheet of the ATmega32.

INPUT CAPTURE AND WAVE GENERATION IN AVR

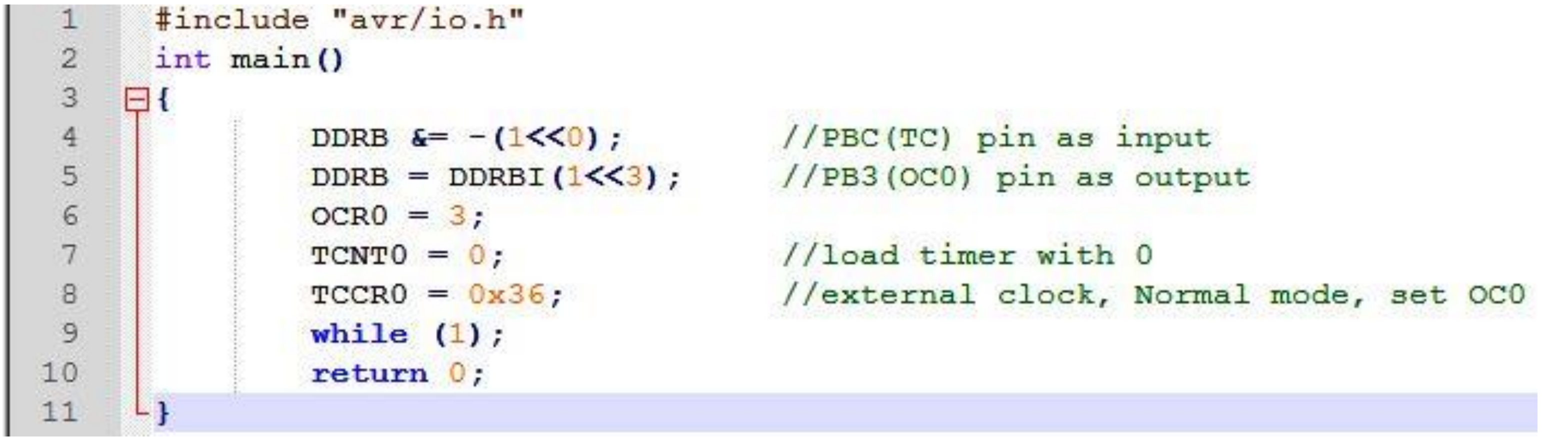
SECTION 15.4: C PROGRAMMING

Example 15-28 (C version of Example 15-2)

Write a program that (a) after 4 external clocks turns on an LED connected to the OC0 pin, and (b) toggles the OC0 pin every 4 pulses.

Solution:

(a)

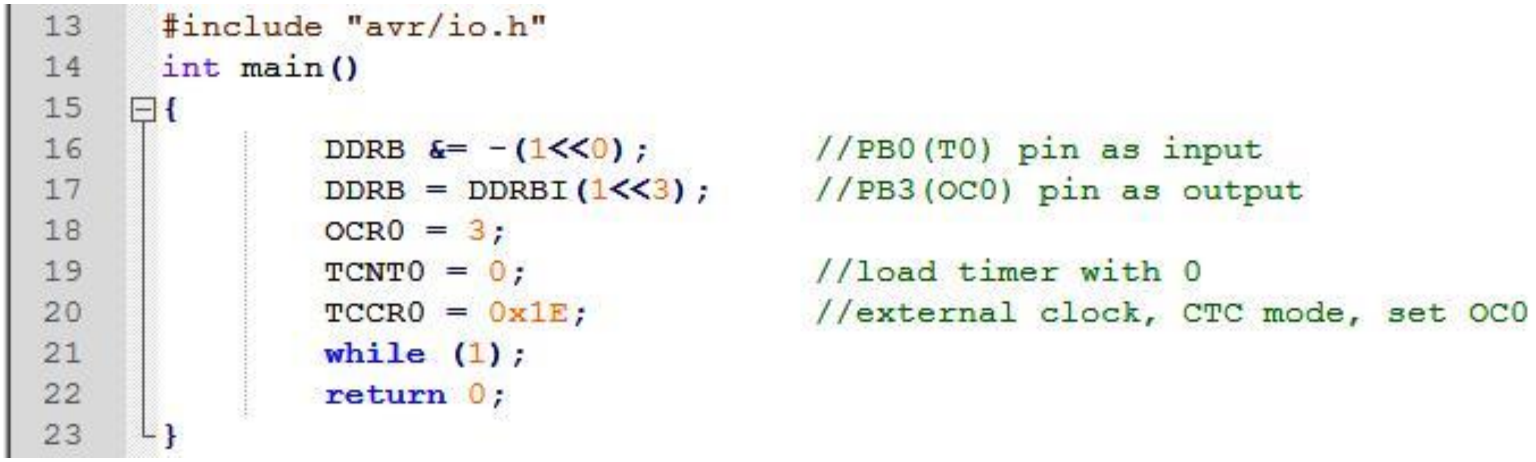


```
#include "avr/io.h"
int main()
{
        DDRB &= -(1<<0);        //PBC(TC) pin as input
        DDRB = DDRBI(1<<3);     //PB3(OC0) pin as output
        OCR0 = 3;
        TCNT0 = 0;              //load timer with 0
        TCCR0 = 0x36;           //external clock, Normal mode, set OC0
        while (1);
        return 0;
}
```

INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

(b)



```c
#include "avr/io.h"
int main()
{
        DDRB &= ~(1<<0);            //PB0(T0) pin as input
        DDRB = DDRB|(1<<3);         //PB3(OC0) pin as output
        OCR0 = 3;
        TCNT0 = 0;                  //load timer with 0
        TCCR0 = 0x1E;               //external clock, CTC mode, set OC0
        while (1);
        return 0;
}
```

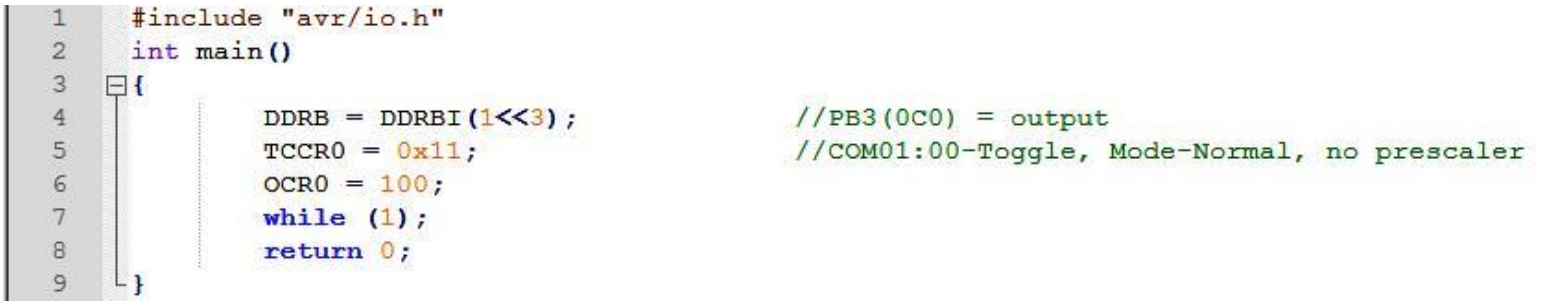
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

Example 15-29 (C version of Example 15-4)

Rewrite the program of Example 15-4 using C.

Solution:



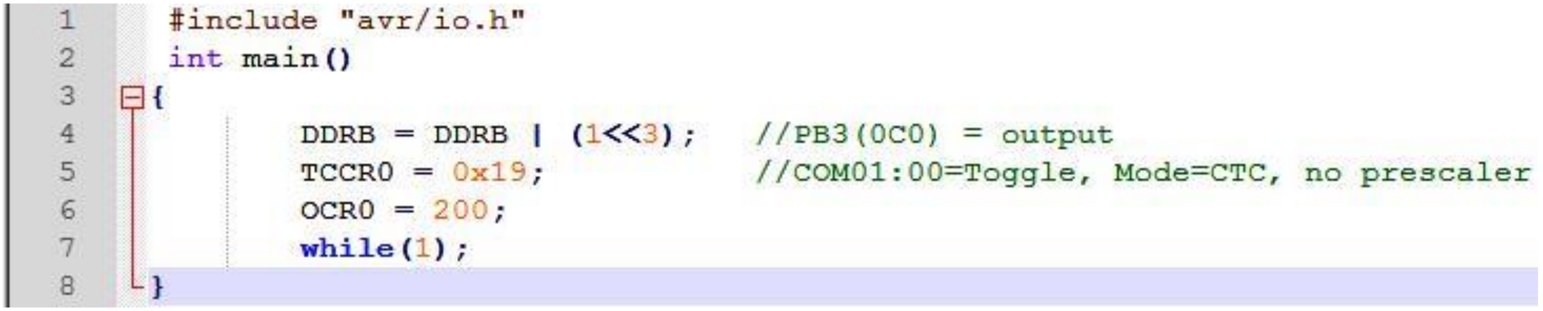
```
#include "avr/io.h"
int main()
{
        DDRB = DDRBI(1<<3);              //PB3(OC0) = output
        TCCR0 = 0x11;                    //COM01:00-Toggle, Mode-Normal, no prescaler
        OCR0 = 100;
        while (1);
        return 0;
}
```

SECTION 15.4: C PROGRAMMING

Example 15-30 , (C, version of Example 15-6)

Rewrite the Program of Example 15,6 using C.

Solution:



```
#include "avr/io.h"
int main()
{
        DDRB = DDRB | (1<<3);    //PB3(OC0) = output
        TCCR0 = 0x19;            //COM01:00=Toggle, Mode=CTC, no prescaler
        OCR0 = 200;
        while(1);
}
```

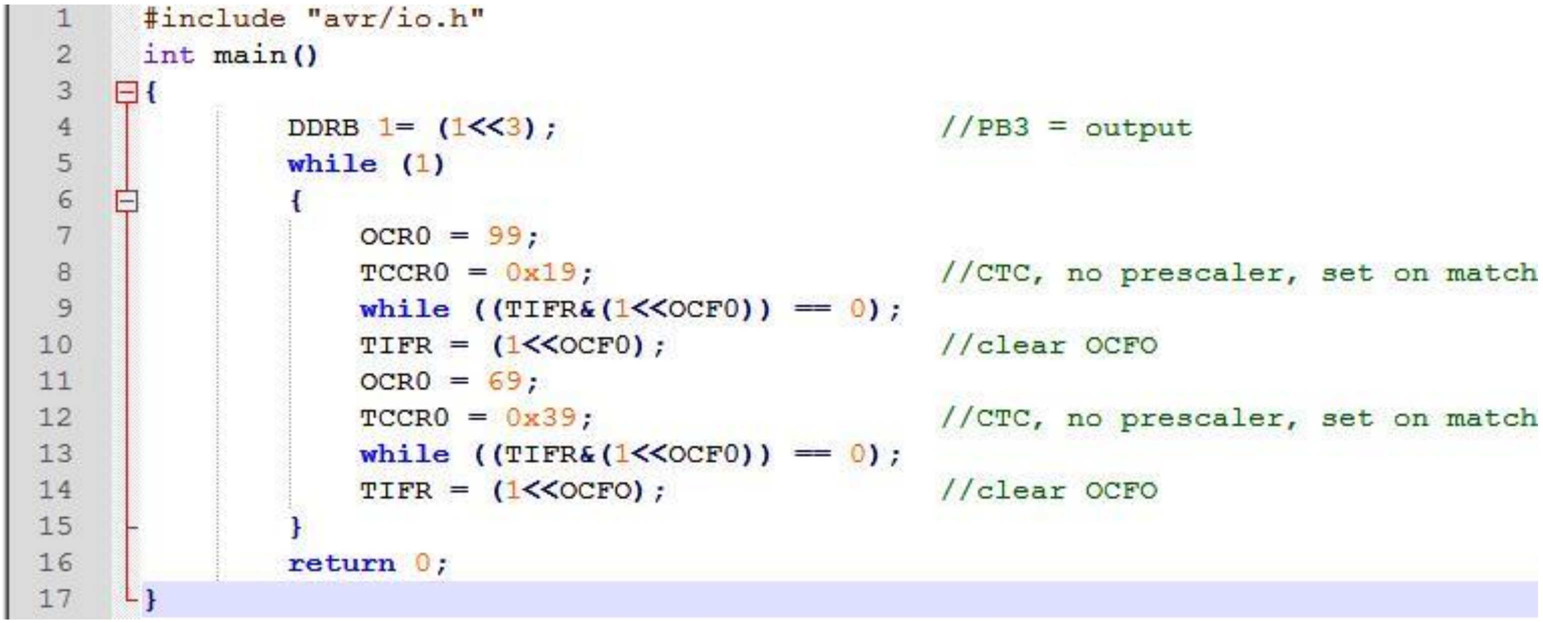
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

Example 15-31 (C,version of. Example 15-8)

Rewrite the program of Example 15-8 using C.

Solution:



```
#include "avr/io.h"
int main()
{
        DDRB |= (1<<3);                             //PB3 = output
        while (1)
        {
            OCR0 = 99;
            TCCR0 = 0x19;                           //CTC, no prescaler, set on match
            while ((TIFR&(1<<OCF0)) == 0);
            TIFR = (1<<OCF0);                       //clear OCFO
            OCR0 = 69;
            TCCR0 = 0x39;                           //CTC, no prescaler, set on match
            while ((TIFR&(1<<OCF0)) == 0);
            TIFR = (1<<OCFO);                       //clear OCFO
        }
        return 0;
}
```

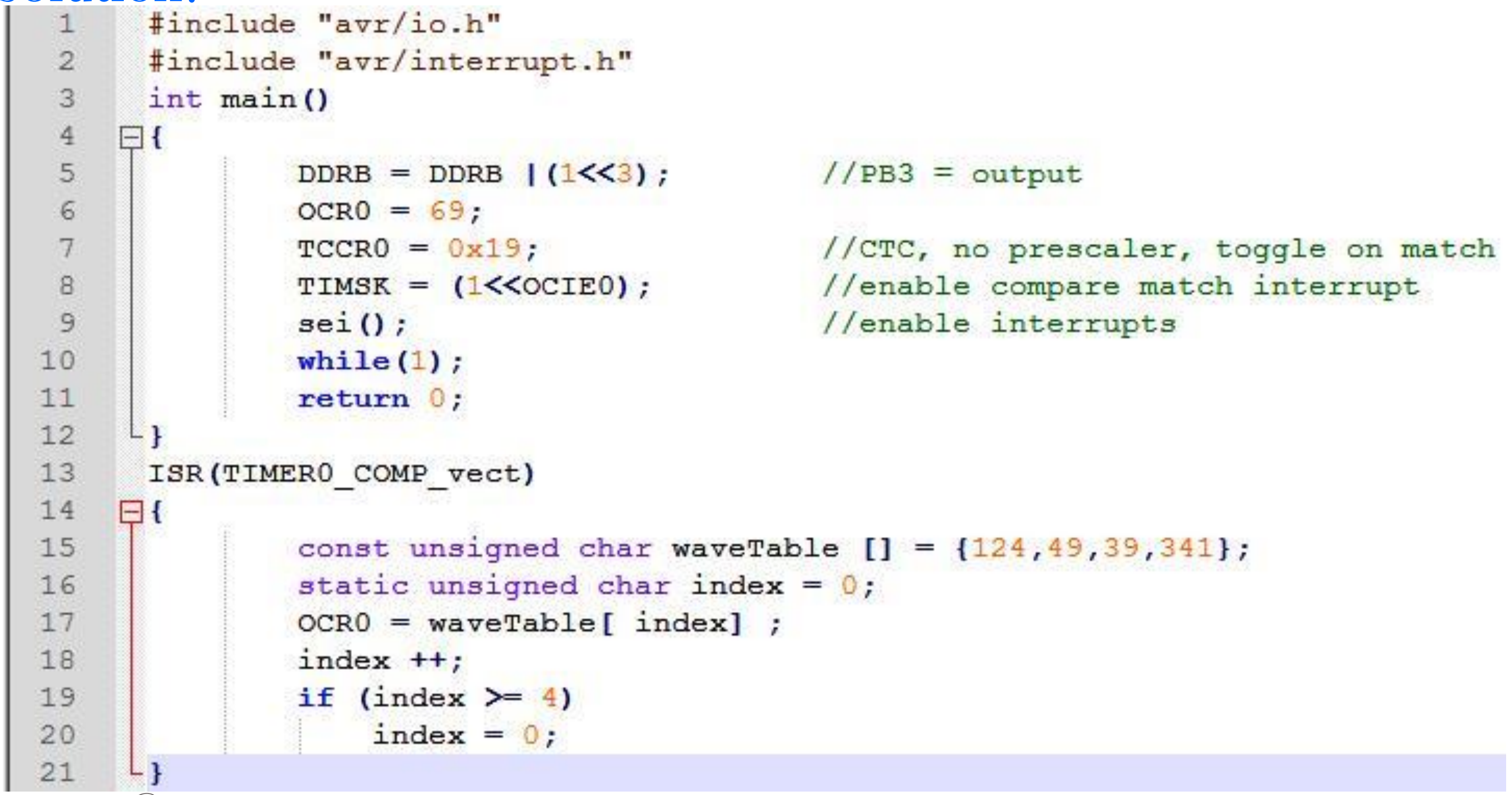
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

Example 15-32 (C version of Example 15-9)

Rewrite the program of Example 15-9 using C.

Solution:



```
#include "avr/io.h"
#include "avr/interrupt.h"
int main()
{
        DDRB = DDRB |(1<<3);          //PB3 = output
        OCR0 = 69;
        TCCR0 = 0x19;                 //CTC, no prescaler, toggle on match
        TIMSK = (1<<OCIE0);           //enable compare match interrupt
        sei();                        //enable interrupts
        while(1);
        return 0;
}
ISR(TIMER0_COMP_vect)
{
        const unsigned char waveTable [] = {124,49,39,341};
        static unsigned char index = 0;
        OCR0 = waveTable[ index] ;
        index ++;
        if (index >= 4)
            index = 0;
}
```

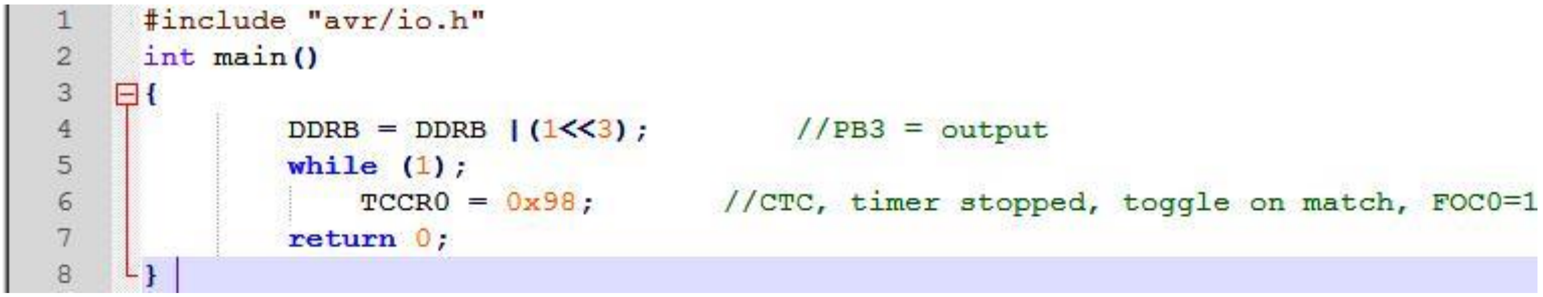
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

Example 15-33 (C version of Example 15-10)

Rewrite the program of Example 15-10 using C.

Solution:



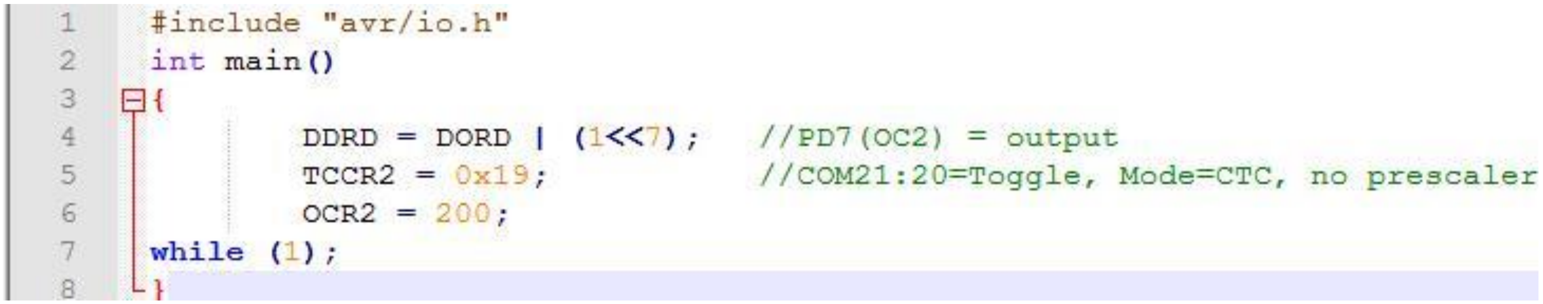
```
#include "avr/io.h"
int main()
{
        DDRB = DDRB |(1<<3);        //PB3 = output
        while (1);
            TCCR0 = 0x98;       //CTC, timer stopped, toggle on match, FOC0=1
        return 0;
}
```

SECTION 15.4: C PROGRAMMING

Example 15-34 (C version of Example 15-12)

Rewrite the program of Example 15-12 using C.

Solution:



```
#include "avr/io.h"
int main()
{
            DDRD = DORD | (1<<7);    //PD7(OC2) = output
            TCCR2 = 0x19;            //COM21:20=Toggle, Mode=CTC, no prescaler
            OCR2 = 200;
while (1);
}
```

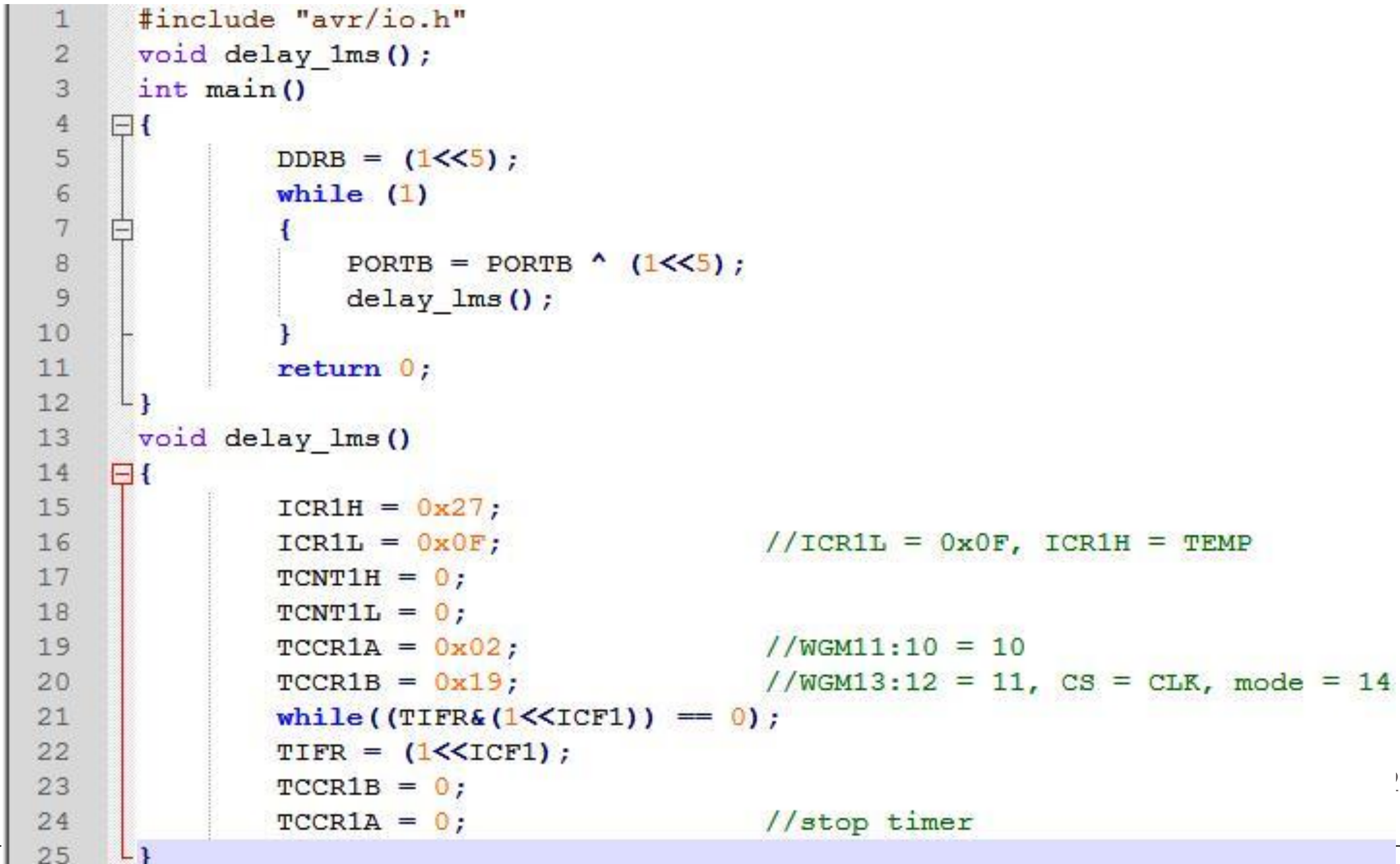
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

Example 15-35 (C version of Example 15-13)

Rewrite the program of Example 15-13 using C.

Solution:



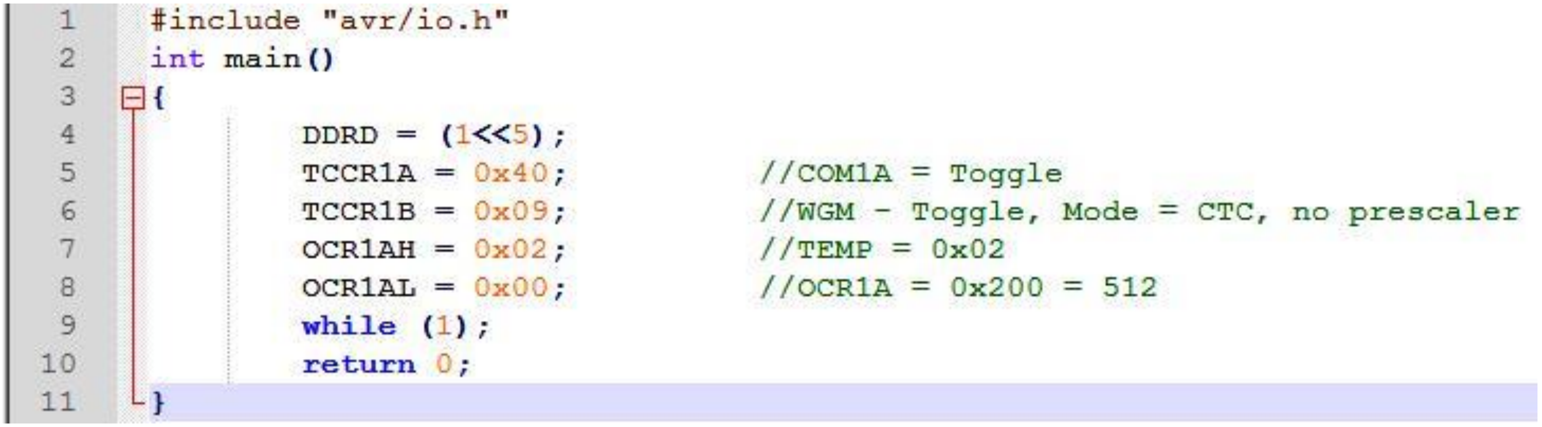
```
#include "avr/io.h"
void delay_1ms();
int main()
{
        DDRB = (1<<5);
        while (1)
        {
            PORTB = PORTB ^ (1<<5);
            delay_1ms();
        }
        return 0;
}
void delay_1ms()
{
        ICR1H = 0x27;
        ICR1L = 0x0F;                  //ICR1L = 0x0F, ICR1H = TEMP
        TCNT1H = 0;
        TCNT1L = 0;
        TCCR1A = 0x02;                 //WGM11:10 = 10
        TCCR1B = 0x19;                 //WGM13:12 = 11, CS = CLK, mode = 14
        while((TIFR&(1<<ICF1)) == 0);
        TIFR = (1<<ICF1);
        TCCR1B = 0;
        TCCR1A = 0;                    //stop timer
}
```

SECTION 15.4: C PROGRAMMING

Example 15-36 (C version of Example 15-18)

Rewrite the program of Example 15-18 using C.

Solution:



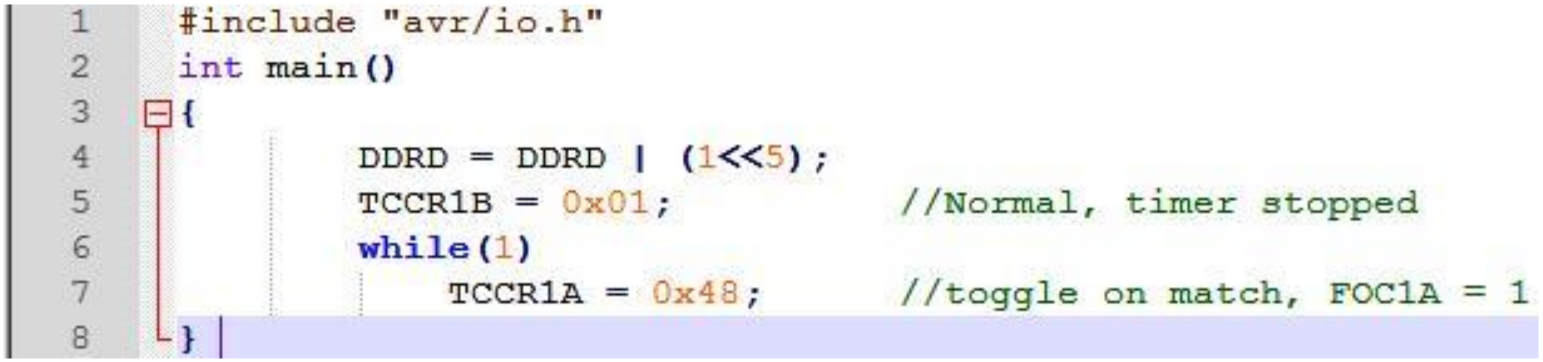
```c
#include "avr/io.h"
int main()
{
        DDRD = (1<<5);
        TCCR1A = 0x40;          //COM1A = Toggle
        TCCR1B = 0x09;          //WGM - Toggle, Mode = CTC, no prescaler
        OCR1AH = 0x02;          //TEMP = 0x02
        OCR1AL = 0x00;          //OCR1A = 0x200 = 512
        while (1);
        return 0;
}
```

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.4: C PROGRAMMING

Example 15-37 (C version of Example 15-20)

Rewrite the program of Example 15-20 using C.

Solution:



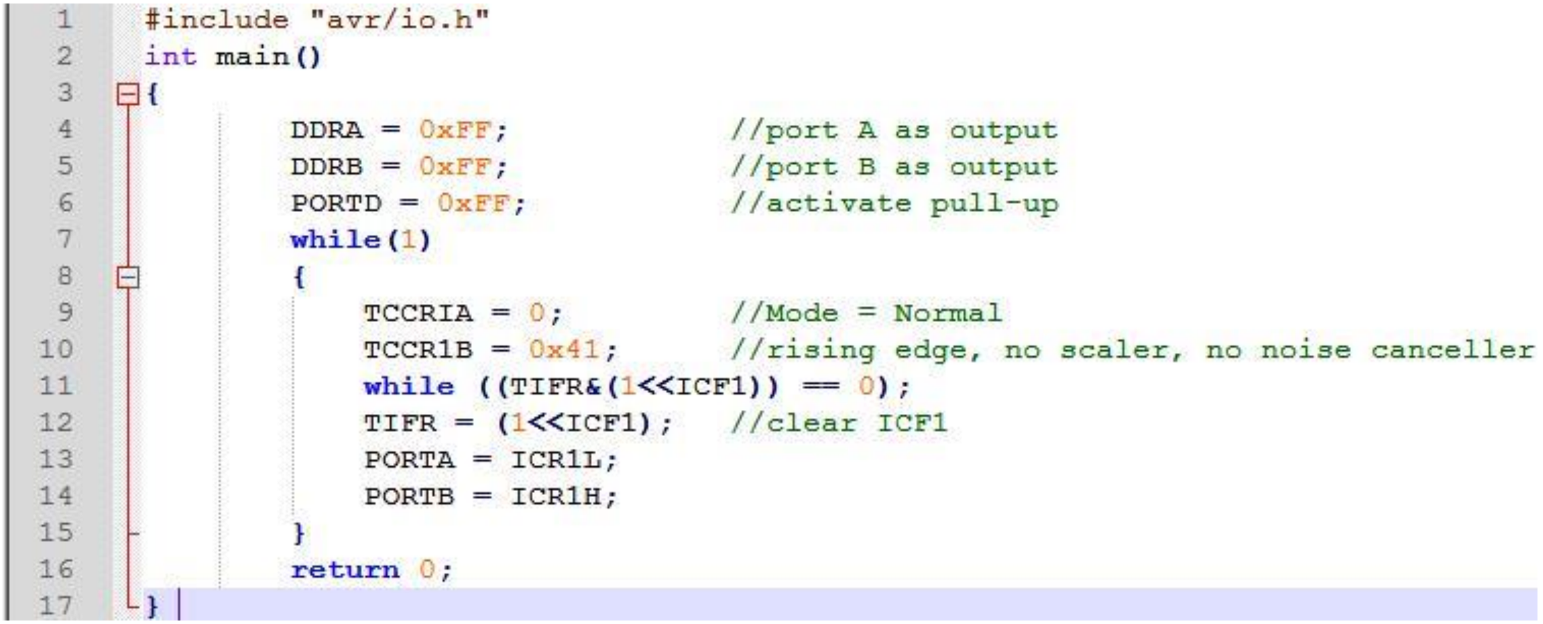
```c
#include "avr/io.h"
int main()
{
        DDRD = DDRD | (1<<5);
        TCCR1B = 0x01;          //Normal, timer stopped
        while(1)
            TCCR1A = 0x48;      //toggle on match, FOC1A = 1
}
```

INPUT CAPTURE AND WAVE GENERATION IN AVR
SECTION 15.4: C PROGRAMMING

Example 15-38 (C version of Example 15-22)

Assuming that clock pulses are fed into pin ICP1, write a program to read the TCNT1 value on every rising edge. Place the result on PORTA and PORTB.

Solution:



```
#include "avr/io.h"
int main()
{
        DDRA = 0xFF;                //port A as output
        DDRB = 0xFF;                //port B as output
        PORTD = 0xFF;               //activate pull-up
        while(1)
        {
            TCCRIA = 0;             //Mode = Normal
            TCCR1B = 0x41;          //rising edge, no scaler, no noise canceller
            while ((TIFR&(1<<ICF1)) == 0);
            TIFR = (1<<ICF1);       //clear ICF1
            PORTA = ICR1L;
            PORTB = ICR1H;
        }
        return 0;
}
```

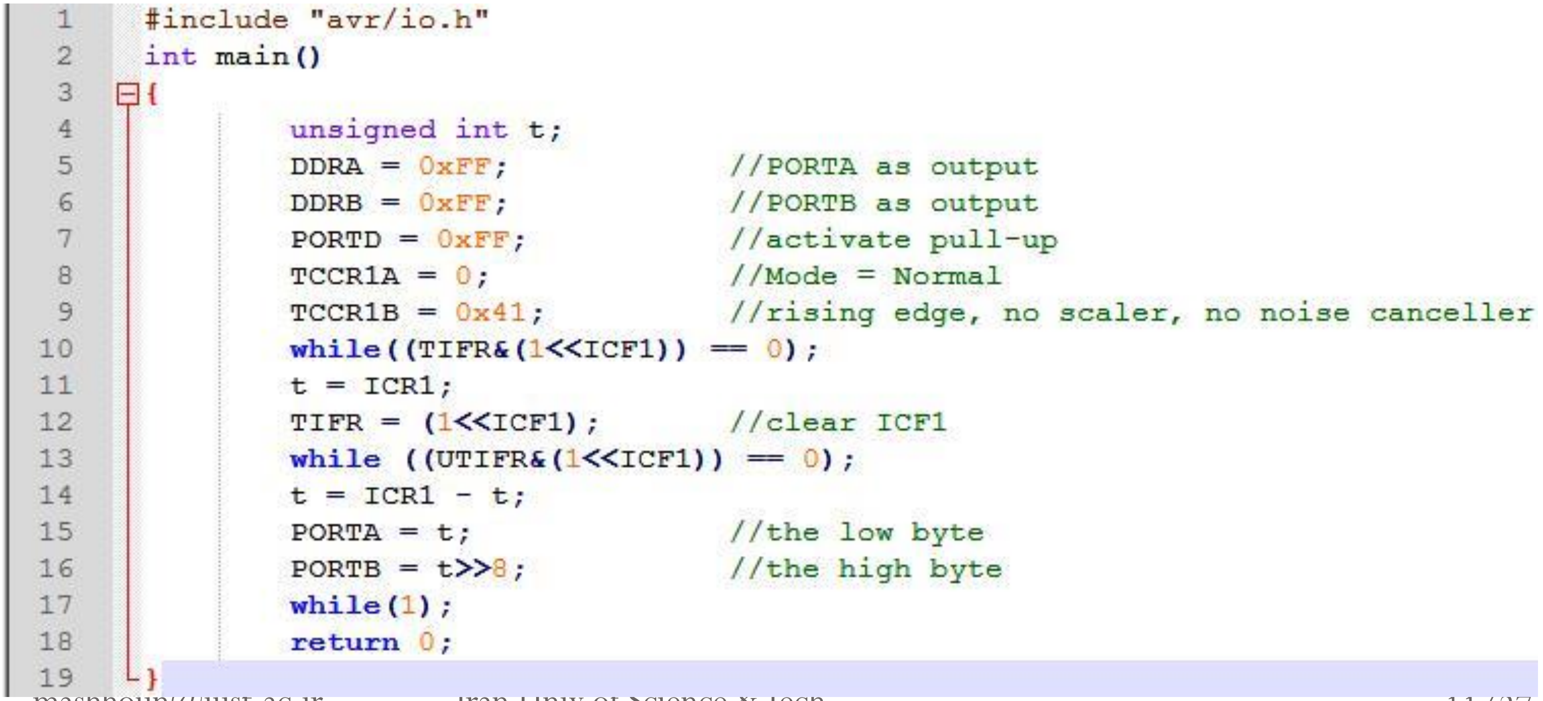
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

Example 15-39 (C version of Example 15-23)

Assuming that clock pulses are fed into pin PORTD.6, write a program to measure the period of the pulses. Place the binary result on PORTA and PORTB.

Solution:



```c
#include "avr/io.h"
int main()
{
        unsigned int t;
        DDRA = 0xFF;            //PORTA as output
        DDRB = 0xFF;            //PORTB as output
        PORTD = 0xFF;           //activate pull-up
        TCCR1A = 0;             //Mode = Normal
        TCCR1B = 0x41;          //rising edge, no scaler, no noise canceller
        while((TIFR&(1<<ICF1)) == 0);
        t = ICR1;
        TIFR = (1<<ICF1);       //clear ICF1
        while ((UTIFR&(1<<ICF1)) == 0);
        t = ICR1 - t;
        PORTA = t;              //the low byte
        PORTB = t>>8;           //the high byte
        while(1);
        return 0;
}
```

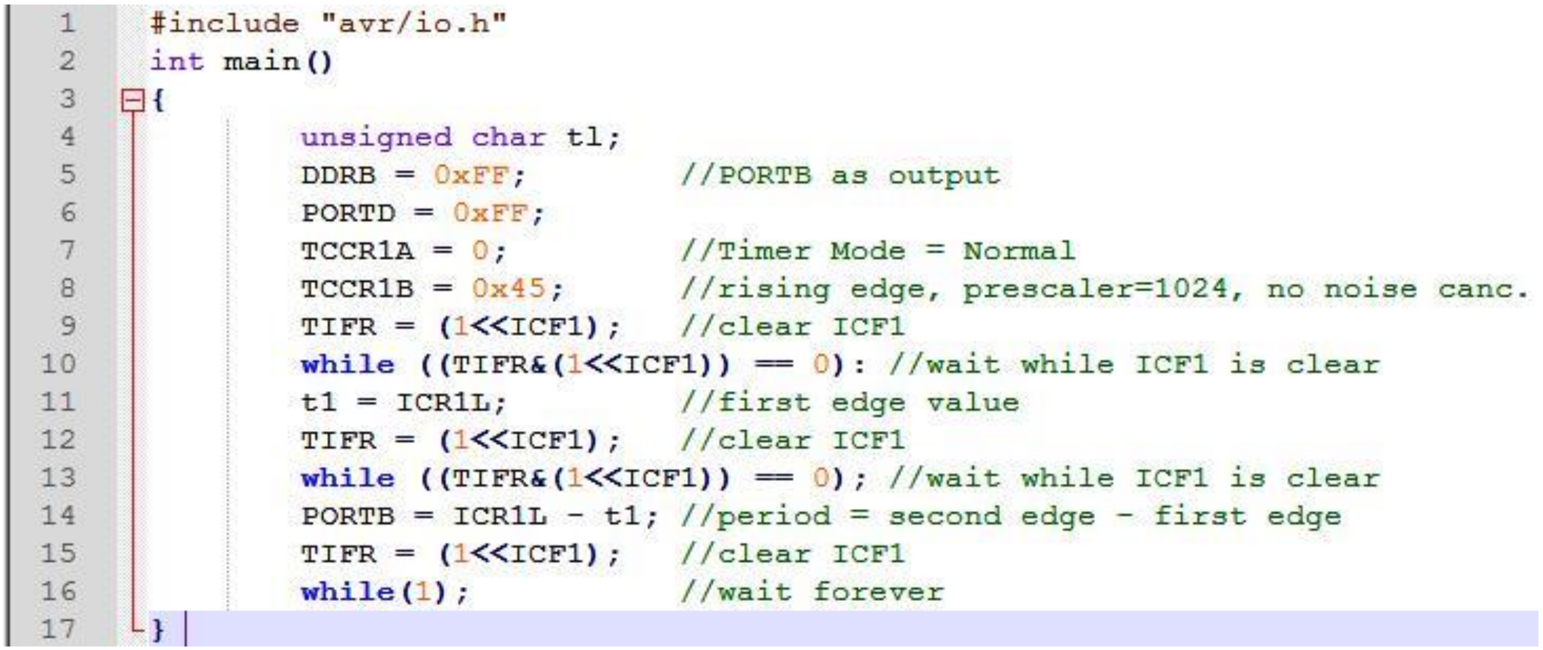
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

Example 15-40 (C version of Example 15-24)

The frequency of a pulse is either 50 Hz or 60 Hz. Assume that a the pulse is connected to ICP1 (pin PD6). Write a program to measure its period and display it on PORTB. Use the prescaler value that gives the result in a single byte. Assume XTAL = 8 MHz.

Solution:



```c
#include "avr/io.h"
int main()
{
        unsigned char t1;
        DDRB = 0xFF;            //PORTB as output
        PORTD = 0xFF;
        TCCR1A = 0;             //Timer Mode = Normal
        TCCR1B = 0x45;          //rising edge, prescaler=1024, no noise canc.
        TIFR = (1<<ICF1);       //clear ICF1
        while ((TIFR&(1<<ICF1)) == 0); //wait while ICF1 is clear
        t1 = ICR1L;             //first edge value
        TIFR = (1<<ICF1);       //clear ICF1
        while ((TIFR&(1<<ICF1)) == 0); //wait while ICF1 is clear
        PORTB = ICR1L - t1; //period = second edge - first edge
        TIFR = (1<<ICF1);       //clear ICF1
        while(1);               //wait forever
}
```

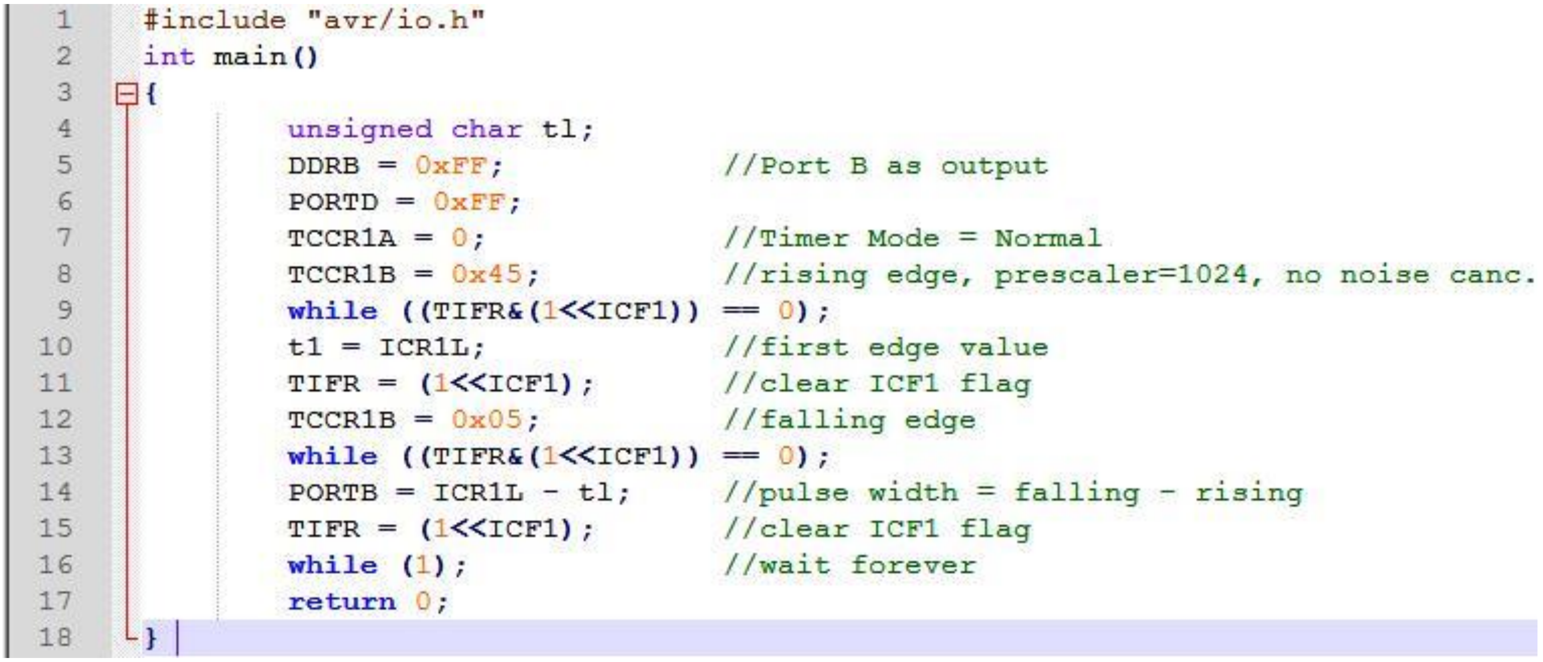
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

Example 15-41 (C version of Example 15-26)

Assume that a 60-Hz frequency pulse is connected to ICP1 (pin PD6). Write a program to measure its pulse width. Use the prescaler value that gives the result in a single byte. Display the result on PORTB. Assume XTAL = 8 MHz.

Solution:



```c
#include "avr/io.h"
int main()
{
        unsigned char t1;
        DDRB = 0xFF;                //Port B as output
        PORTD = 0xFF;
        TCCR1A = 0;                 //Timer Mode = Normal
        TCCR1B = 0x45;              //rising edge, prescaler=1024, no noise canc.
        while ((TIFR&(1<<ICF1)) == 0);
        t1 = ICR1L;                 //first edge value
        TIFR = (1<<ICF1);           //clear ICF1 flag
        TCCR1B = 0x05;              //falling edge
        while ((TIFR&(1<<ICF1)) == 0);
        PORTB = ICR1L - t1;         //pulse width = falling - rising
        TIFR = (1<<ICF1);           //clear ICF1 flag
        while (1);                  //wait forever
        return 0;
}
```

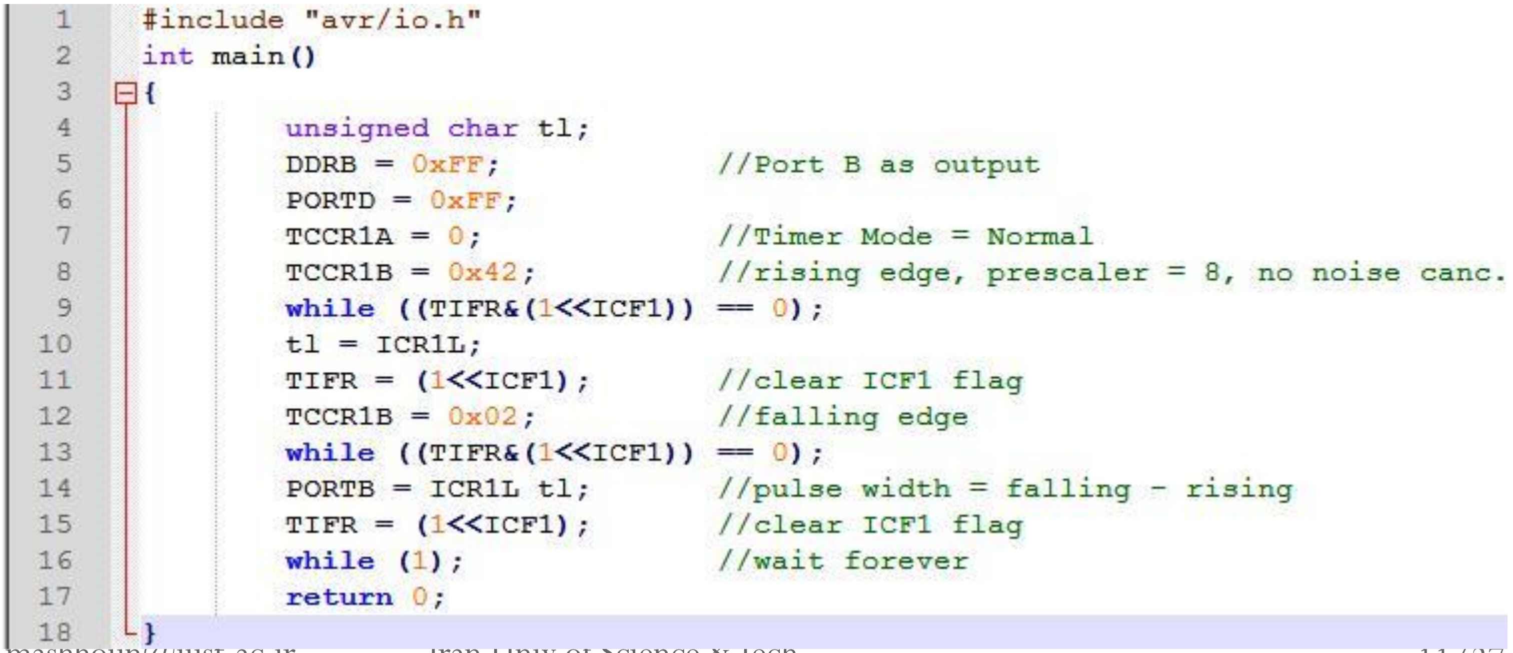
INPUT CAPTURE AND WAVE GENERATION IN AVR

SECTION 15.4: C PROGRAMMING

Example 15-42 (C version of Example 15-27)

Assume that a temperature sensor is connected to pin PD6. The temperature provided by the sensor is proportional to pulse width and is in the range of 1µs to 250 µs. Write a program to measure the temperature if 1 µs is equal to 1 degree. Use the prescaler value that gives the result on PORTS. Assume XTAL = 8 MHz.

Solution:



```
#include "avr/io.h"
int main()
{
        unsigned char tl;
        DDRB = 0xFF;                //Port B as output
        PORTD = 0xFF;
        TCCR1A = 0;                 //Timer Mode = Normal
        TCCR1B = 0x42;              //rising edge, prescaler = 8, no noise canc.
        while ((TIFR&(1<<ICF1)) == 0);
        tl = ICR1L;
        TIFR = (1<<ICF1);           //clear ICF1 flag
        TCCR1B = 0x02;              //falling edge
        while ((TIFR&(1<<ICF1)) == 0);
        PORTB = ICR1L tl;           //pulse width = falling - rising
        TIFR = (1<<ICF1);           //clear ICF1 flag
        while (1);                  //wait forever
        return 0;
}
```