AVR Microcontroller

Microprocessor Course

Chapter 16

PWM PROGRAMMING AND

DC MOTOR CONTROL IN AVR

Bahman 1397 (Version 1.2)

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.1 : DC MOTOR INTERFACING AND PWM

DC motors

A direct current (DC) motor is a widely used device that translates electrical pulses into mechanical movement. In the DC motor we have only + and - leads. Connecting them to a DC voltage source moves the motor in one direction. By reversing the polarity, the DC motor will move in the opposite direction. One can easily experiment with the DC motor.

For example, the small fans used in many motherboards to cool the CPU are run by DC motors. When the leads are connected to the + and - voltage source, the DC motor moves. While a stepper motor moves in steps of 1 to 15 degrees, the DC motor moves continuously.

In a stepper motor, if we know the starting position we can easily count the number of steps the motor has moved and calculate the final position of the motor. This is not possible with a DC motor.

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

DC motors

The maximum speed of a DC motor is indicated in rpm and is given in the data sheet. The DC motor has two rpms: no-load and loaded. The manufacturer's datasheet gives the no-load rpm. The no-load rpm can be from a few thousand to tens of thousands.

The rpm is reduced when moving a load and it decreases as the load is increased. For example, a drill turning a screw has a much lower rpm speed than when it is in the no-load situation. DC motors also have voltage and current ratings.

The nominal voltage is the voltage for that motor under normal conditions, and can be from 1 to 150 V, depending on the motor. As we increase the voltage, the rpm goes up. The current rating is the current consumption when the nominal voltage is applied with no load, and can be from 25 mA to a few amps.

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

DC motors

As the load increases, the rpm is decreased, unless the current or voltage provided to the motor is increased, which in turn increases the torque.

With a fixed voltage, as the load increases, the current (power) consumption of a DC motor is increased. If we overload the motor it will stall, and that can damage the motor due to the heat generated by high current consumption.

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Unidirectional control

Figure 16-1 shows the DC motor rotation for clockwise (CW) and counterclockwise (CCW) rotations.

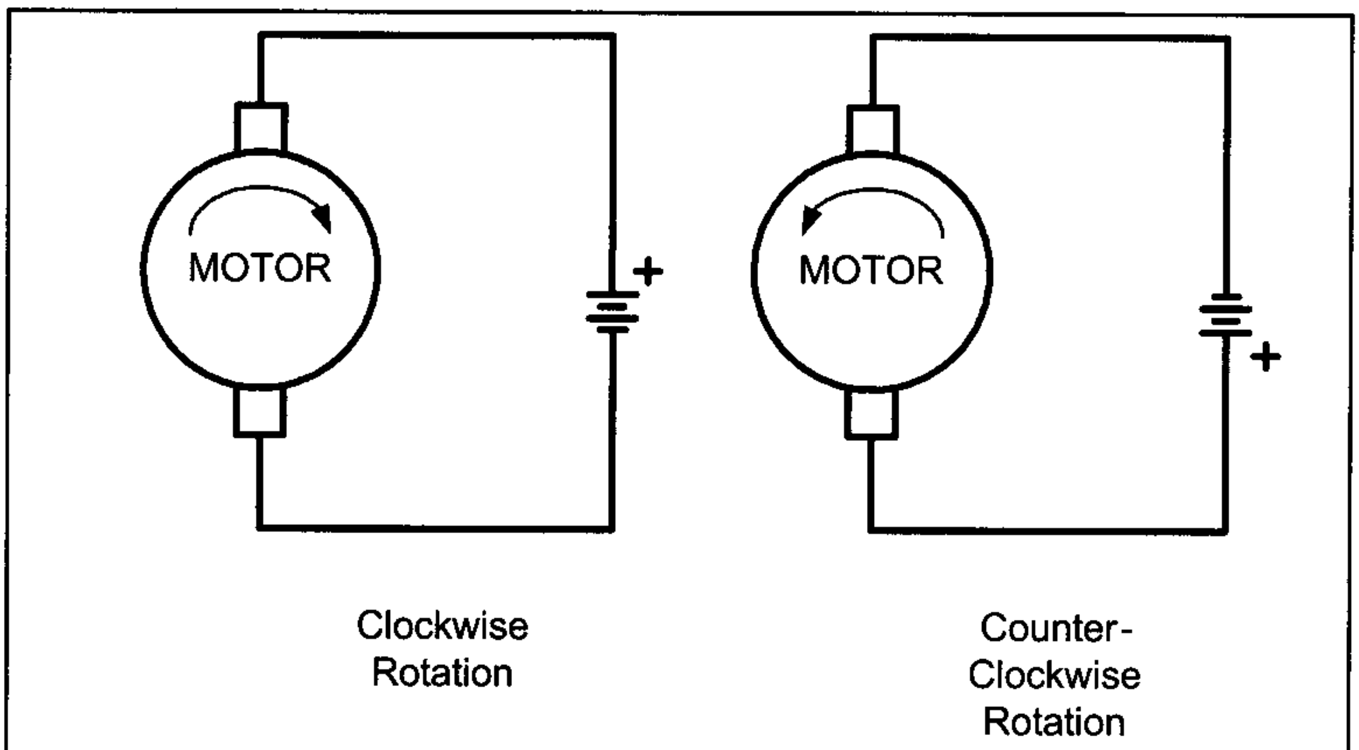


Figure 16-1. DC Motor Rotation (Permanent Magnet Field)

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.1 : DC MOTOR INTERFACING AND PWM

See Table 16-1 for selected DC motors

Table 16-1: Selected DC Motor Characteristics (www.Jameco.com)

Part No.	Nominal Volts	Volt Range	Current	RPM	Torque
154915CP	3 V	1.5–3 V	0.070 A	5,200	4.0 g-cm
154923CP	3 V	1.5–3 V	0.240 A	16,000	8.3 g-cm
177498CP	4.5 V	3–14 V	0.150 A	10,300	33.3 g-cm
181411CP	5 V	3–14 V	0.470 A	10,000	18.8 g-cm

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Bidirectional control

With the help of relays or some specially designed chips we can change the direction of the DC motor rotation. Figures 16-2 through 16-4 show the basic concepts of H-bridge control of DC motors.

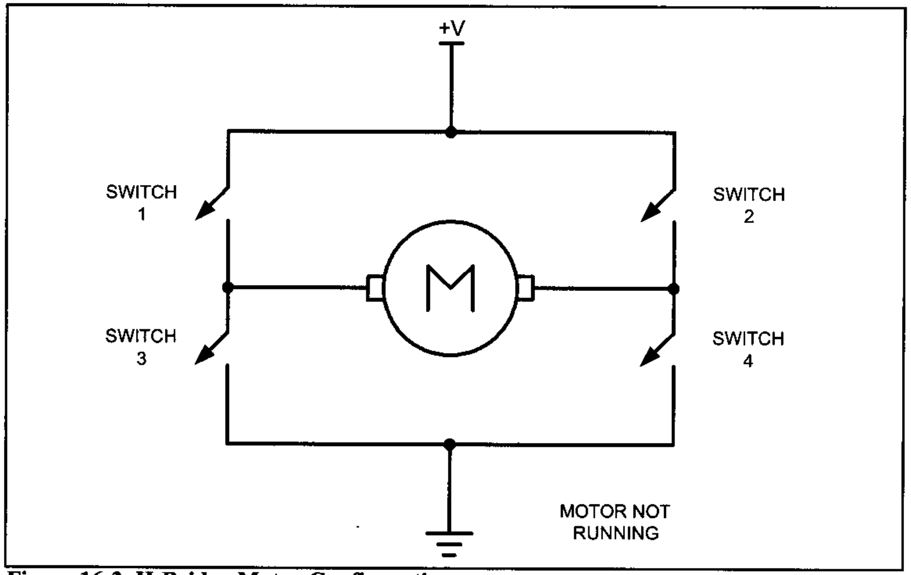


Figure 16-2. H-Bridge Motor Configuration

Figure 16-2 shows the connection of an H-bridge using simple switches. All the switches are open, which does not allow the motor to turn.

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Figure 16-3 shows the switch configuration for turning the motor in one direction. When switches 1 and 4 are closed, current is allowed to pass through the motor.

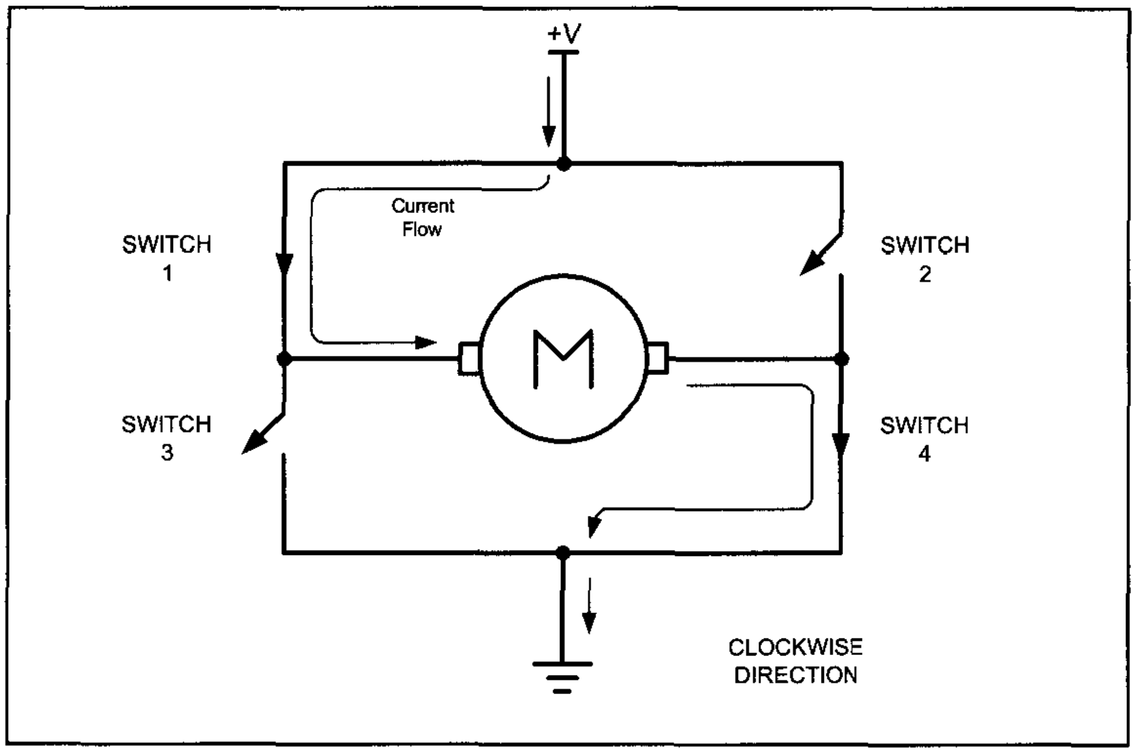


Figure 16-3. H-Bridge Motor Clockwise Configuration

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Figure 16-4 shows the switch configuration for turning the motor in the opposite direction from the configuration of Figure 16-3. When switches 2 and **3** are closed, current is allowed to pass through the motor.

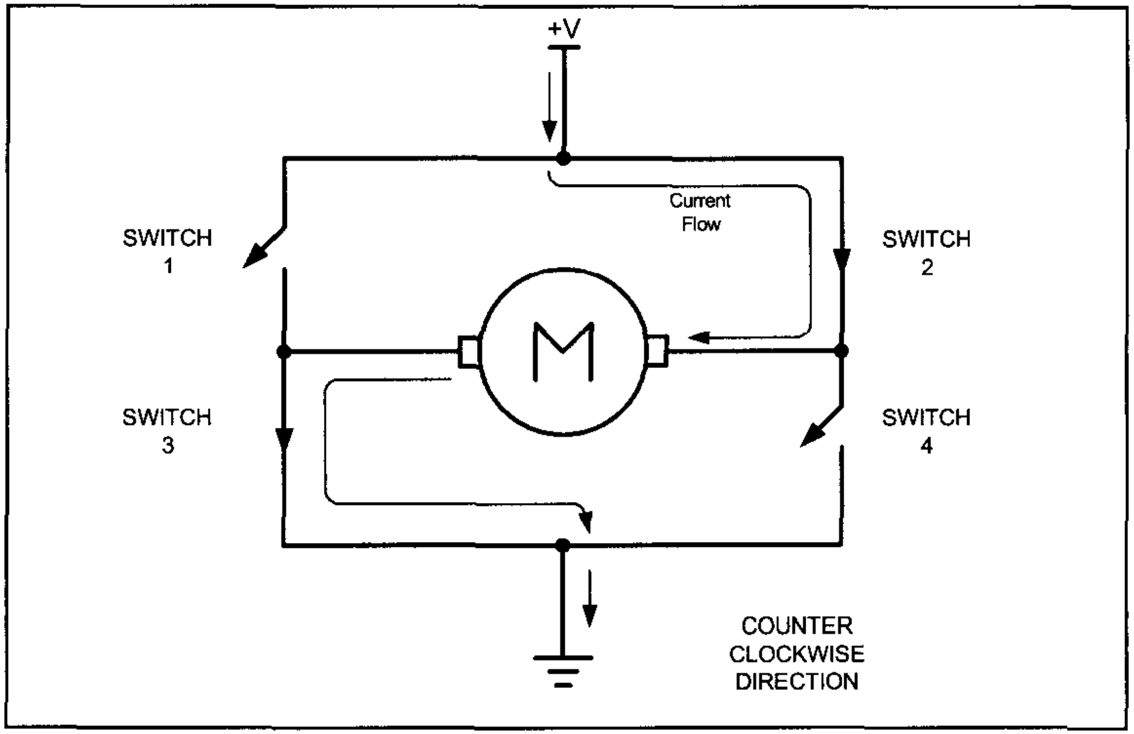


Figure 16-4. H-Bridge Motor Counterclockwise Configuration

mashhoun@iust.ac.ir Iran Univ of Science & Tech

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Figure 16-5 shows an invalid configuration. Current flows directly to ground, creating a short circuit. The same effect occurs when switches 1 and 3 are closed or switches 2 and 4 are closed.

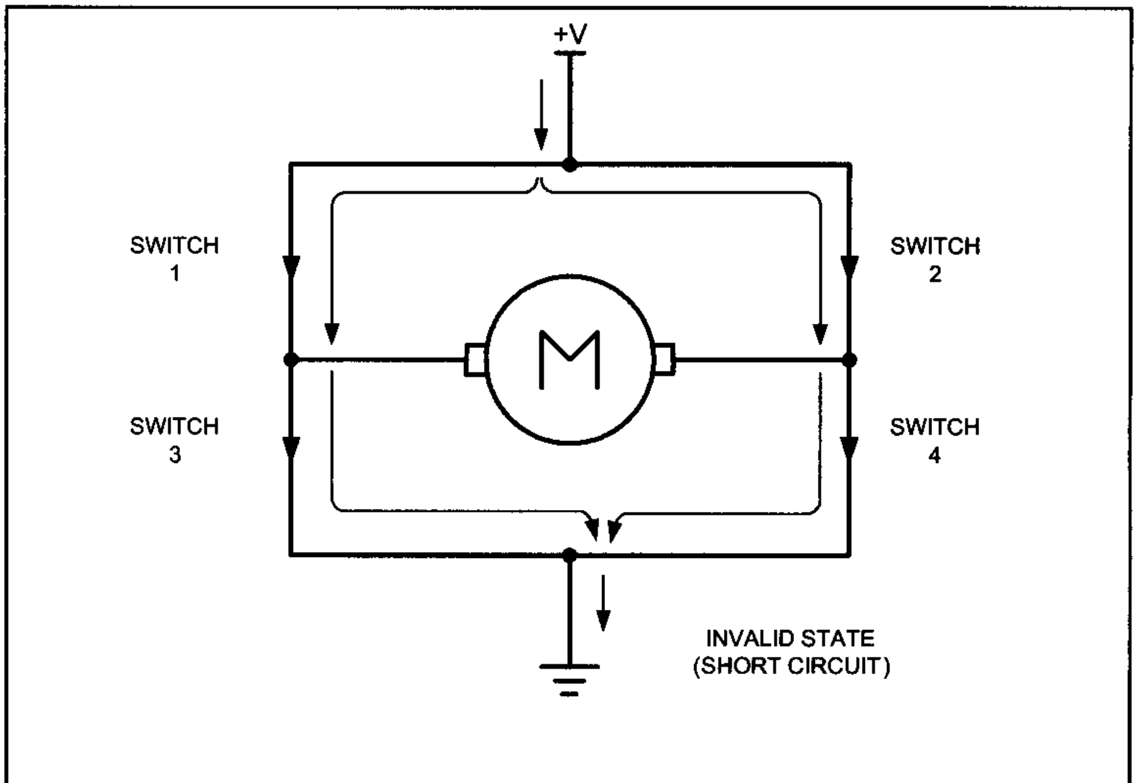


Figure 16-5. H-Bridge in an Invalid Configuration

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Table 16-2 shows some of the logic configurations for the H-bridge design. H-bridge control can be created using relays, transistors, or a single IC solution such as the L298. When using relays and transistors, you must ensure that invalid configurations do not occur.

Table 16-2: Some H-Bridge Logic Configurations for Figure 16-2

Motor Operation	SW1	SW2	SW3	SW4
Off	Open	Open	Open	Open
Clockwise	Closed	Open	Open	Closed
Counterclockwise	Open	Closed	Closed	Open
Invalid	Closed	Closed	Closed	Closed

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Example 16-1

A switch is connected to pin PA7 (PORTA.7). Using a simulator, write a program to simulate the H-bridge in Table 16-2. We must perform the following: (a) If PA7 = 0, the DC motor moves clockwise. (b) If PA7 = 1, the DC motor moves counterclockwise.

Solution:

```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,0          ;make PB0 an output (switchl)
        SBI     DDRB,1          ;make PB1 an output (switch2)
        SBI     DDRB,2          ;make PB2 an output (switch3)
        SBI     DDRB,3          ;make PB3 an output (switch4)
        CBI     DDRA,7          ;make PA7 an input
MONITOR:
        SBIS    PINA,7          ;skip next if PINA.7 is set
        RJMP    CLKWISE         ;if PA7 = 0 go to CLKWISE
        CBI     DDRB,1          ;switch2 = 0
        CBI     DDRB,2          ;switch3 = 0
        SBI     DDRB,0          ;switchl = 1
        SBI     DDRB,3          ;switch4 = 1
        JMP     MONITOR
CLKWISE:
        CBI     DDRB,0          ;switchl = 0
        CBI     DDRB,3          ;switch4 = 0
        SBI     DDRB,1          ;switch2 = 1
        SBI     DDRB,2          ;switch3 = 1
```

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Figure 16-6 shows the connection of the L298 to an AVR. Be aware that the L298 will generate heat during operation. For sustained operation of the motor, use a heat sink. Example 16-2 shows control of the L298.

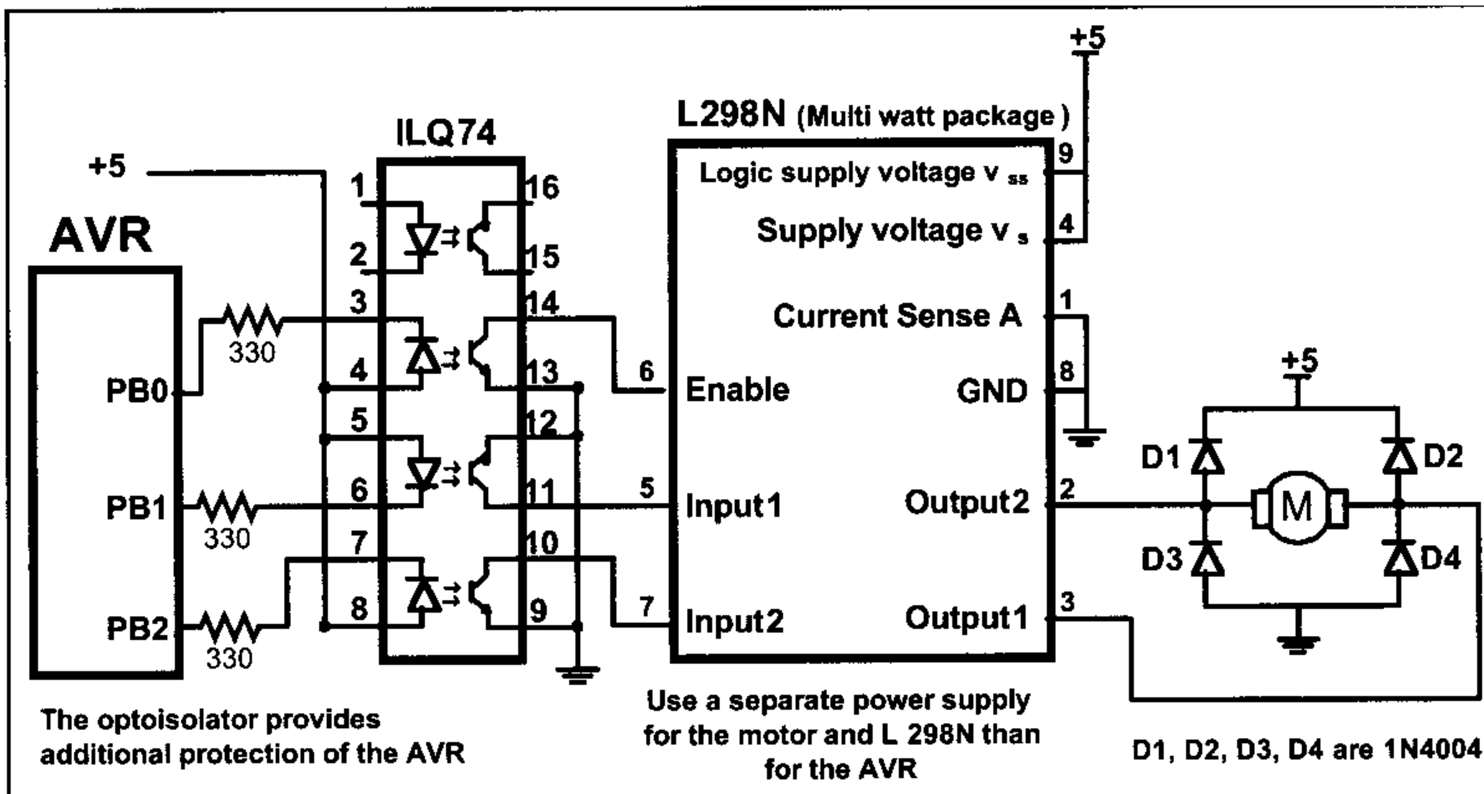


Figure 16-6. Bidirectional Motor Control Using an L298 Chip

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

DC motor control with optoisolator

As we discussed in Chapter 14, the optoisolator is indispensable in many motor control applications. Figures 16-6 through 16-8 show the connections to a simple DC motor using an optoisolator. Notice that the AVR is protected from EMI created by motor brushes by using an optoisolator and a separate power supply.

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Example 16-2

Figure 16-6 shows the connection of an L298. Add a switch to pin PA7 (PORTA.7). Write a program to monitor the status of SW and perform the following: (a) If SW = 0, the DC motor moves clockwise. (b) If SW = 1, the DC motor moves counterclockwise.

Solution:

```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,0          ;make PB0 an output (Enable)
        SBI     DDRB,1          ;make PB1 an output (clock)
        SBI     DDRB,2          ;make PB2 an output (counter)
        SBI     PORTB,0         ;Enable = 1
        CBI     DDRA,7          ;make PA7 an input
        SBI     PORTA,7
MONITOR:
        SBIS    PINA,7          ;skip next if PINA.7 is set
        RJMP    CLKWISE         ;if PA7 = 0 go to CLKWISE
        CBI     PORTB,1         ;switch1 = 0
        SBI     PORTB,2         ;switch2 = 1
        JMP     MONITOR
CLKWISE:
        SBI     PORTB,1         ;switch1 = 0
        CBI     PORTB,2         ;switch2 = 1
        JMP     MONITOR
```

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Figure 16-7 shows the connection of a bipolar transistor to a motor. Protection of the control circuit is provided by the optoisolator.

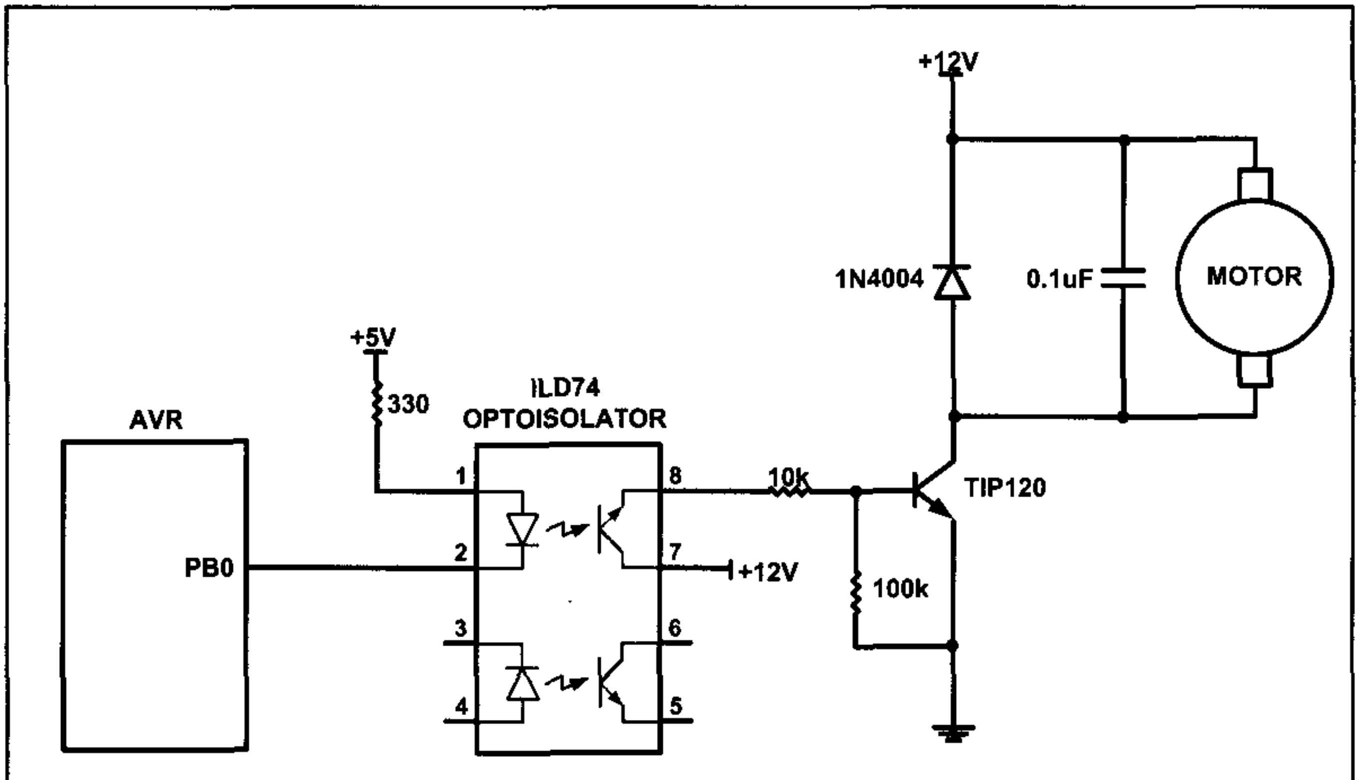


Figure 16-7. DC Motor Connection Using a Darlington Transistor

mashhoun@iust.ac.ir

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

The motor and AVR use separate power supplies. The separation of power supplies also allows the use of high-voltage motors. Notice that we use a decoupling capacitor across the motor; this helps reduce the EMI created by the motor. The motor is switched on by clearing bit PB0. The Zener diode is required for the transistor to reduce gate voltage below the rated maximum value.

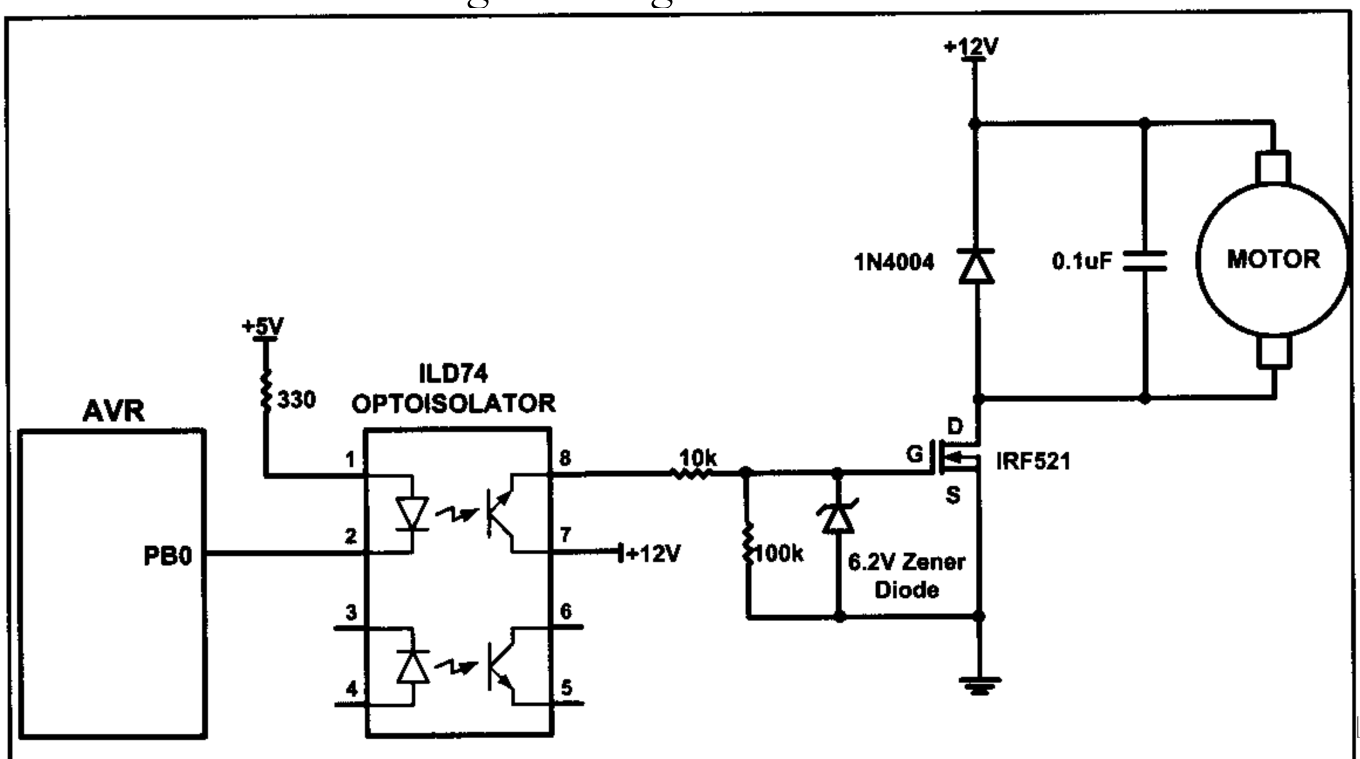


Figure 16-8. DC Motor Connection Using a MOSFET Transistor

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Pulse width modulation (PWM) page 564

The speed of the motor depends on three factors:

(a) **load**
(b) **Voltage**
(c) **Current**

For a given fixed load we can maintain a steady speed by using a method called ***pulse width modulation (PWM)*****.** By changing (modulating) the width of the pulse applied to the DC motor we can increase or decrease the amount of power provided to the motor, thereby increasing or decreasing the motor speed.

PWM is so widely used in DC motor control that some microcontrollers come with the PWM circuitry embedded in the chip. In such microcontrollers all we have to do is load the proper registers with the values of the high and low portions of the desired pulse, and the rest is taken care of by the microcontroller.

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

The ability to control the speed of the DC motor using PWM is one reason that DC motors are often preferred over AC motors. AC motor speed is dictated by the AC frequency of the voltage applied to the motor and the frequency is generally fixed. As a result, we cannot control the speed of the AC motor when the load is increased.

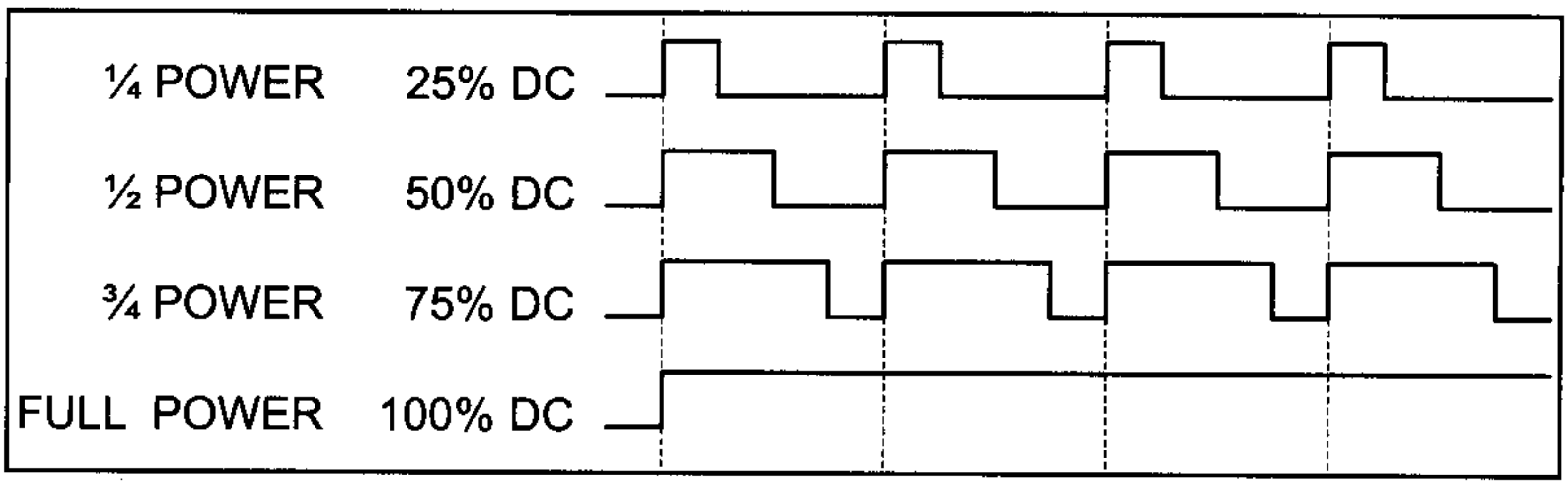
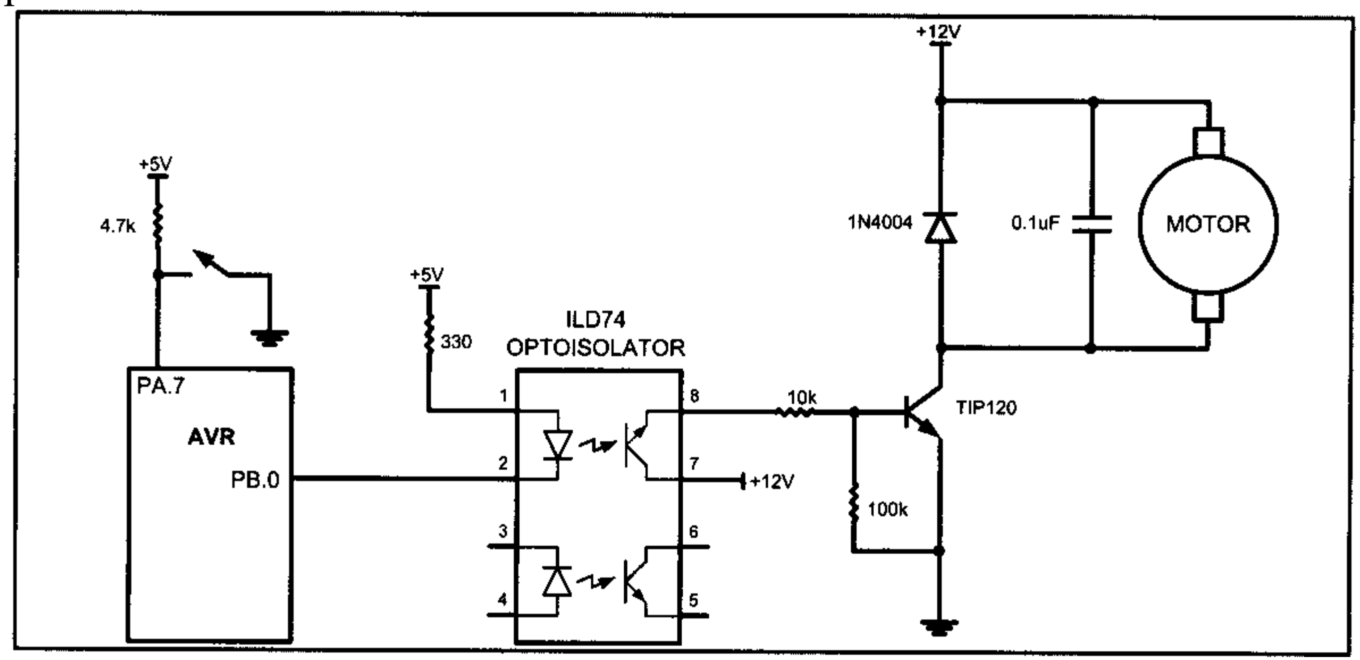


Figure 16-9. Pulse Width Modulation Comparisons

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Example 16-3

Refer to the figure in this example. Write a program to monitor the status of the switch and perform the following: (a) If PORTA.7 = 1, the DC motor moves with 25% duty cycle pulse. (11) If PORTA.7 = 0, the DC motor moves with 50% duty cycle pulse.



mashhoun@iust.ac.ir

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

```
.INCLUDE "M32DEF.INC"
        LDI     R16,HIGH(RAMEND)
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16                 ;initialize stack pointer
        SBI     DDRB,0                  ;PORTB.0 as output
        CBI     DDRA,7                  ;PORTA.7 as input
        SBI     PORTA,7                 ;enable pull-up
        CBI     PORTB,0                 ;turn off motor
CHK:    SBIC    PINA,7
        RJMP    P50
        SBI     PORTB,0                 ;high portion Of pulse
        RCALL   DELAY
        RCALL   DELAY
        RCALL   DELAY
        CBI     PORTB,0                 ;low portion of pulse
        RCALL   DELAY
        RJMP    CHK
P50:    SBI     PORTB,0                 ;high portion of pulse
        RCALL   DELAY
        RCALL   DELAY
        CBI     PORTB,0                 ;low portion of pulse
        RCALL DELAY
        RCALL DELAY
        RJMP  CHK
```

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Example 16-4 (C version of Example 16-2)

Refer to Figure 16-6 for connection of the motor. A switch is connected to pin PA7. Write a C program to monitor the status of SW and perform the following:

(a) If SW = 0, the DC motor moves clockwise.

(b) If SW = 1, the DC motor moves counterclockwise.

Solution:

```
#include    "avr/io.h"
#define     ENABLE   0
#define     MTR_1    1
#define     MTR_2    2
#define     SW       (PINA&0x80)

```

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.1 : DC MOTOR INTERFACING AND PWM

```c
int main()
{
        DDRA = 0x7F;                            //make PA7 input pin
        DDRB = 0xFF;                            //make PORTB output pin
        PORTB = PORTB & (~(1<<ENABLE));
        PORTB = PORTB & (~(1<<MTR_1));
        PORTB = PORTB & (~(1<<MTR_2));
        while (1)
        {
            PORTB = PORTB | (1<<ENABLE);
            if(SW == 1)
            {
                PORTB = PORTE | (1<<MTR_1);     //MTR 1 = 1
                PORTB = PORTB & (~(1<<MTR_2));  //MTR 2 = 0
            }
            else
            {
                PORTB = PORTB & (~(1<<MTR_1));  //MTR_1 = 0
                PORTE = PORTB | (1<<mTR_2) ;    //MTR_2 = 1
            }
        }
        return 0;
}
```

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

Example 16-5 (C version of Example 16-3)

Refer to the figure in this example. Write a C program to monitor the status of SW and perform the following:

(a) If SW = 0, the DC motor moves with 50% duty cycle pulse.

(b) If SW = 1, the DC motor moves with 25% duty cycle pulse.

Solution:

```
#define      F_CPU    8000000UL          //XTAL = 8 MHz
#define      SW       (PORTA&(1<<7))
#include "avr/io.h"
#include "util/delay.h"
```

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR

SECTION 16.1 : DC MOTOR INTERFACING AND PWM

```
void main()
{
        DDRA=0x7F;              //make PA7 input pin
        DDRB=0x01;              //make PB0 output pin
        while(1)
        {
            if (SW == 1)
            {
                PORTB = PORTB | (1<<0);
                _delay ms(75);
                PORTB = PORTB & (~(1<<0));
                _delay ms(25);
            }
            else
            {
                PORTB = PORTB |(1<<0);
                _delay ms(50);
                PORTB = PORTB & (~(1<<0));
                _delay ms(50);
            }
        }
}
```

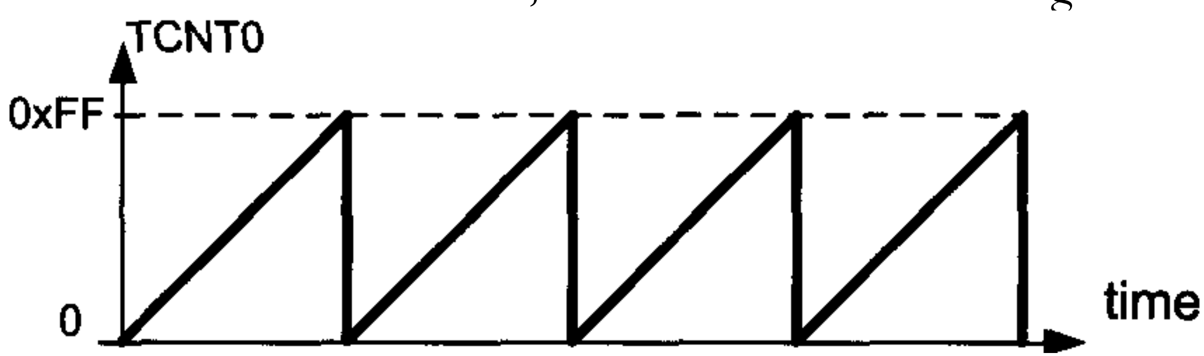
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

PWM MODES IN 8-BIT TIMERS

In this section we discuss the PWM feature of the AVR. The ATmega32 comes with three timers, which can be used as wave generators. The advantage of using the built-in PWM feature of the AVR is that it gives us the option of programming the period and duty cycle, therefore relieving the CPU to do other important things.

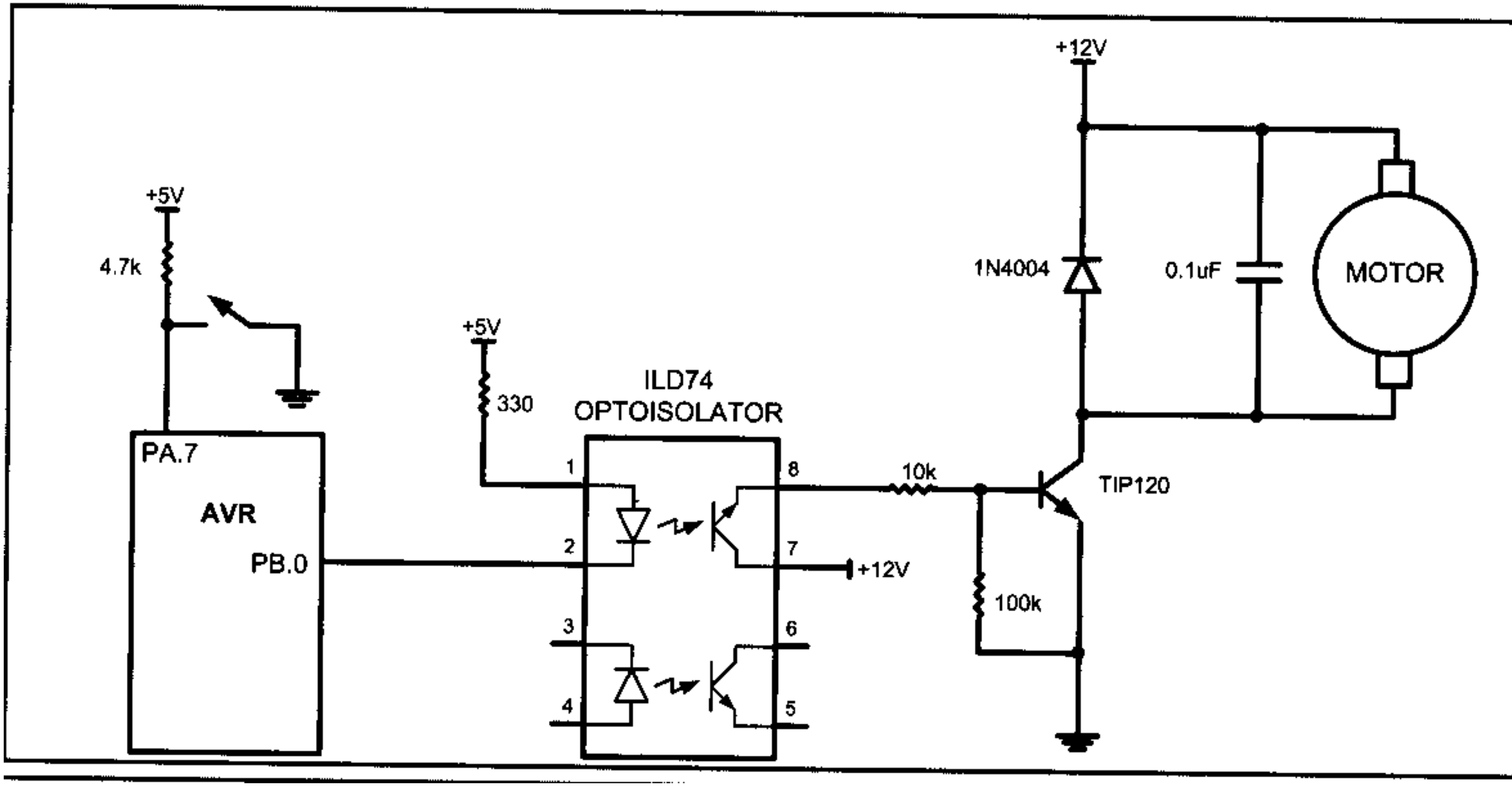
Fast PWM mode

In the Fast PWM, the counter counts like it does in the Normal mode. After the timer is started, it starts to count up. It counts up until it reaches its limit of 0xFF. When it rolls over from 0xFF to 00, it sets HIGH the TOV0 flag.



PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR

SECTION 16.1 : DC MOTOR INTERFACING AND PWM



PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

In Figure 16-12 you see the reaction of the waveform generator when compare match occurs while the timer is in Fast PWM mode.

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 compare match: it is a write-only bit, which can be used while generating a wave. Writing 1 to it causes the wave generator to act as if a compare match has occurred (see Chapter 15).
WGM01:00	D3D6	Timer0 mode selector bit
	0 0	Normal
	0 1	PWM, Phase correct
	1 0	CTC (Clear Timer on Compare match)
	1 1	Fast PWM

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

COM01:00 D5 D4 Compare Output Mode when Timer0 is in Fast PWM mode

COM01	COM00	Mode Name	Description
0	0	Disconnected	Normal port operation, OC0 disconnected
0	1	Reserved	Reserved
1	0	Non-inverted	Clear OC0 on compare match, set OC0 at TOP
1	1	Inverted PWM	Set OC0 on compare match, clear OC0 at TOP

CS02:00 D2D1D0 Timer0 clock selector

D2	D1	D0	
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 16-12. TCCR0 (Timer/Counter Control Register) Register

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

When COM01:00=00 the OC0 pin operates as an I/O port. *When* COM01:00=10, the waveform generator clears the OC0 pin whenever compare match occurs, and sets it at top. This mode is called **non-inverted PWM.** ***See Figures***16-13A through 16-13C. As you see from these figures, in the non-inverted PWM, the duty cycle of the generated wave increases when the value of OCR0 increase.

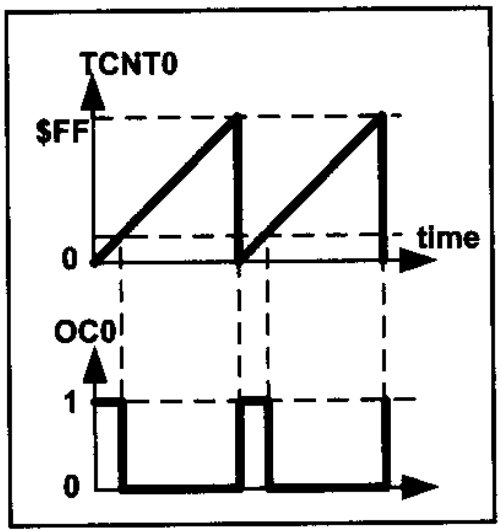


Figure 16-13A. Non-inverted

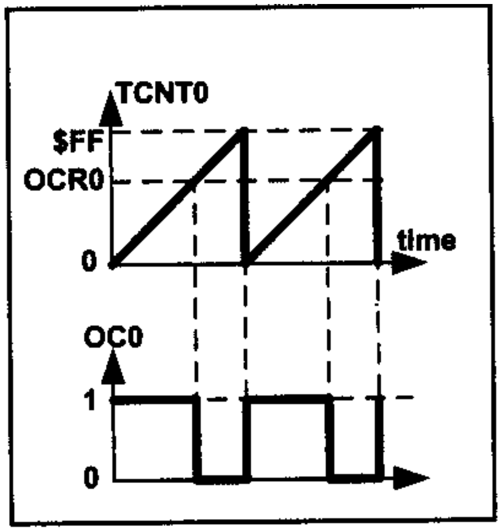


Figure 16-13B. Non-inverted

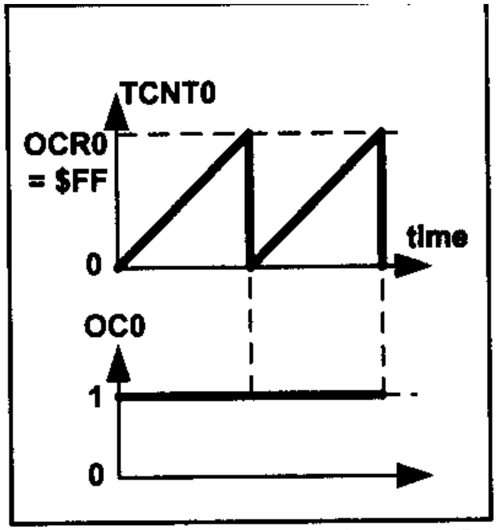


Figure 16-13C. Non-inverted

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

When COM01:00=11, the waveform generator sets the OC0 pin whenever compare match occurs, and clears it at top. This mode is referred as **inverted PWM mode**. See Figures 16-14A through 16-14C. As you see, in the inverted PWM mode when the value of OCR0 increases, the duty cycle of the generated wave decreases.

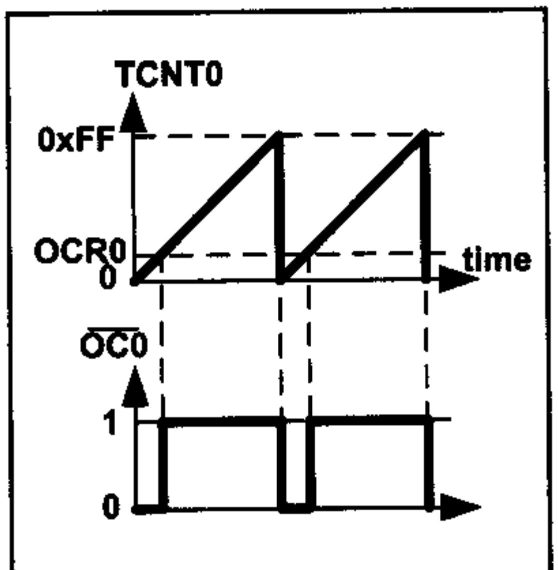


Figure 16-14A. Inverted

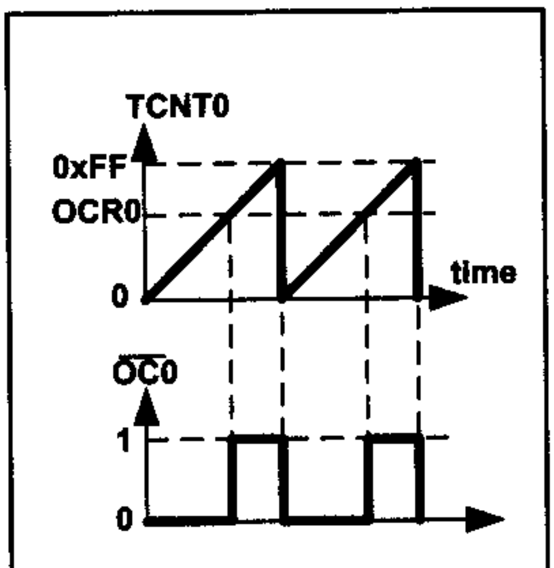


Figure 16-14B. Inverted

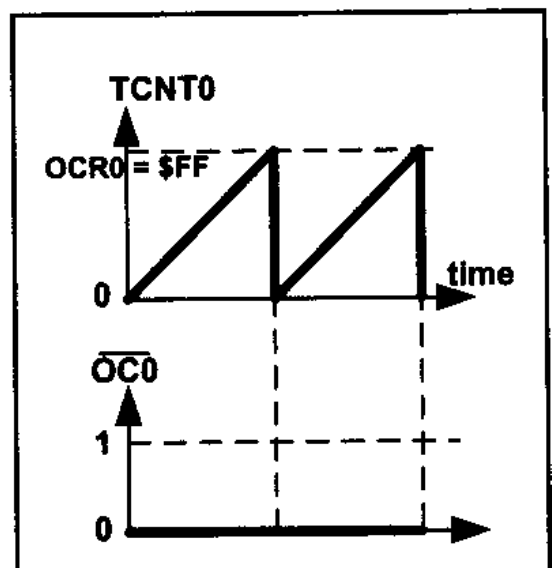


Figure 16-14C. Inverted

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Frequency of the generated wave in Fast PWM mode

In Fast PWM mode, the timer counts from 0 to top (0xFF in 8-bit counters) and then rolls over. So, the frequency of the generated wave is 1/256 of the frequency of timer clock. So, in 8-bit timers the frequency of the generated wave can be calculated as follows (N is determined by the prescaler):

$$\left.\begin{array}{l} F_{\text{generated wave}} = \dfrac{F_{\text{timer clock}}}{256} \\[2ex] F_{\text{timer clock}} = \dfrac{F_{\text{oscillator}}}{N} \end{array}\right\} \Rightarrow F_{\text{generated wave}} = \dfrac{F_{\text{oscillator}}}{256 \times N}$$

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

$$\text{Duty Cycle} = \frac{\text{OCR0} + 1}{256} \times 100$$

For non Inverted mode

$$\text{Duty Cycle} = \frac{255 - \text{OCR0}}{256} \times 100$$

For Inverted mode

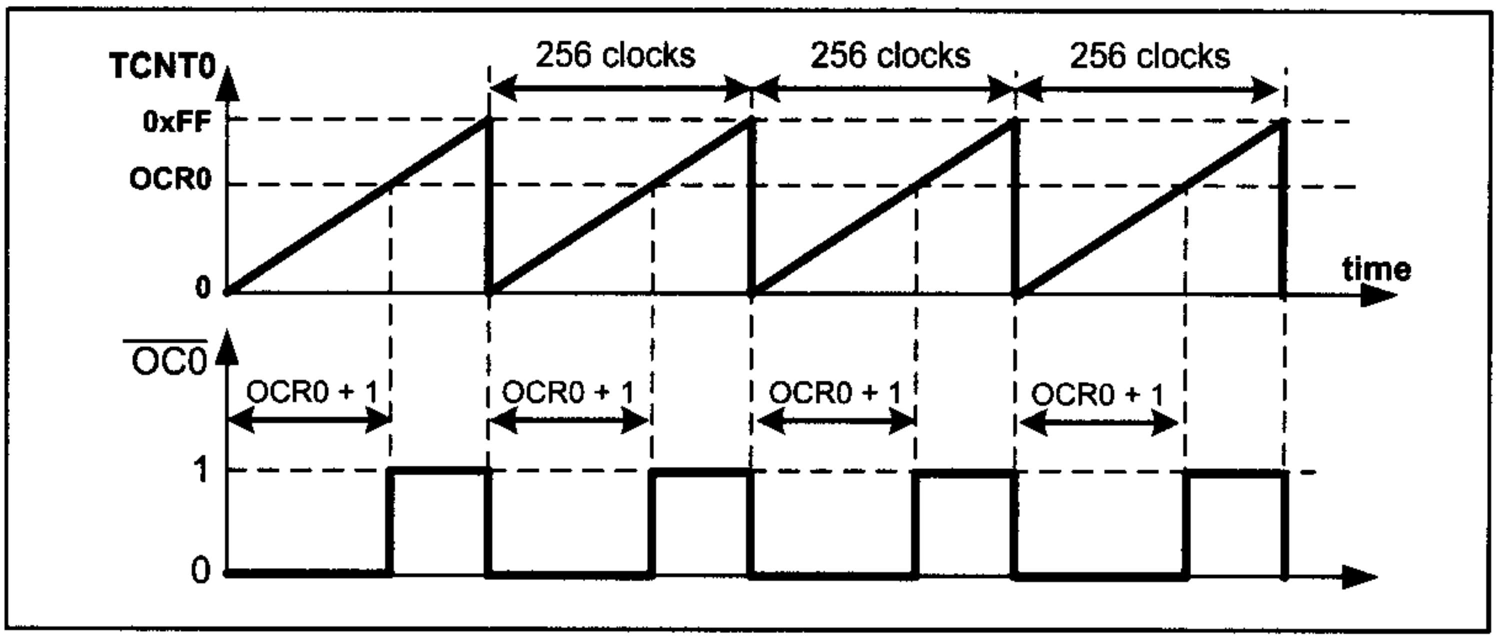


Figure 16-15. Timer/Counter 0 Fast PWM mode

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-6

To generate a wave with duty cycle of 75% in non-inverted mode, calculate the OCR0.

Solution:

$$75=(OCR0+1)\times 100/256 \Rightarrow OCR0+1=75\times 256/100=192 \Rightarrow OCR0 = 191$$

mashhoun@iust.ac.ir Iran Univ of Science & Tech

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-7

Find the value for TCCR0 to initialize Timer0 for Fast PWM mode, non-inverted PWM wave generator, and no prescaler.

Solution:

WGM01:00-11=Fast PWM mode

COM01:00=10=Non-inverted PWM

TCCR0=CS02:00=001=No prescaler

TCCR0 =

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

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-8

Assuming XTAL = 8 MHz, using non-inverted mode, write a program that generates a wave with frequency of 31,250 Hz and duty cycle of 75%.

Solution:

$31{,}250 = 8M/(256 \times N) \Rightarrow N = 8M/(31{,}250 \times 256) = 1 \Rightarrow N = 1 \Rightarrow$ No prescaler

```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,3
        LDI     R20,191                 ;from Example 16-6
        OUT     OCR0,R20                ;OCR0 - 191
        LDI     R20,0x69                ;from Example 16-7
        OUT     TCCR0,R20               ;Fast PWM, no prescaler, non-inverted
HERE:   RJMP    HERE                    ;infinite loop
```

Notice that instead of the infinite loop we can use the CPU to perform other things.

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-9

Assuming XTAL = 8 MHz, using non-inverted mode, write a program that generates a wave with frequency of 3906.25 Hz and duty cycle of 37.5%.

Solution:

$3906.25 = 8M/(256 \times N) \Rightarrow N = 8M/(3906.25 \times 256) = 8 \Rightarrow$ the prescaler value=8

$37.5 = 100 \times (OCR0+1)/256 \Rightarrow OCR0+1 = (256 \times 37.5)/100 = 96 \quad OCR0 = 95$

```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,3
        LDI     R20,95
        OUT     OCR0,R20        ;OCRO = 95
        LDI     R20,0x6A
        OUT     TCCR0,R20       ;Fast PWM, N = 8, non-inverted
HERE:   RJMP    HERE
```


mashhoun@iust.ac.ir

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-10

Rewrite Example 16-9 using inverted mode.

Solution:

$$37.5 = 100 \times (255 - OCR0)/256 \Rightarrow 255 - OCR0 = (256 \times 37.5)/100 = 96 \Rightarrow OCR0 = 159$$

```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,3
        LDI     R20,159
        OUT     OCR0,R20            ;OCR0 = 159
        LDI     R20,0x7A
        OUT     TCCR0,R20           ;Fast PWM, N = 8, inverted
HERE:   RUMP    HERE
```

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Loading values into the OCRx register in PWM modes

In the non-PWM modes (CTC mode and Normal), when we load a value into the OCR0 register, the value will be loaded instantly into the OCR0 register,

But in the PWM modes, there is a buffer between us and the OCR0 register. When we read/write a value from/into the OCR0 we are dealing with the buffer. The contents of the buffer will be loaded into the OCR0 register only when the TCNT0 reaches to its top most value.

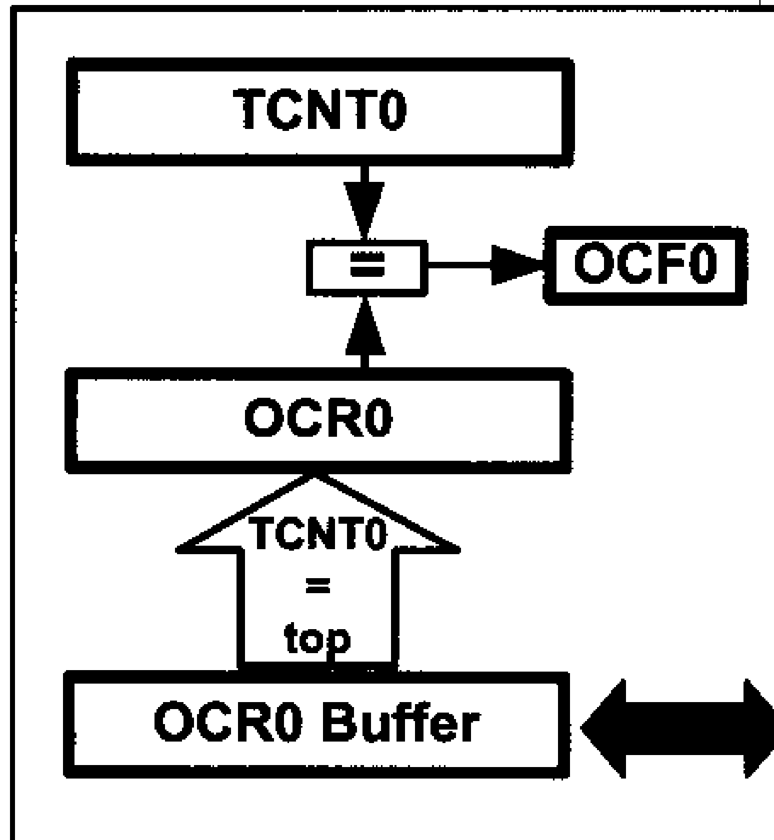


Figure 16-16. OCRn Buffer in PW

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-11

Draw the wave generated by the following program. Assume XTAL = 1 MHz.

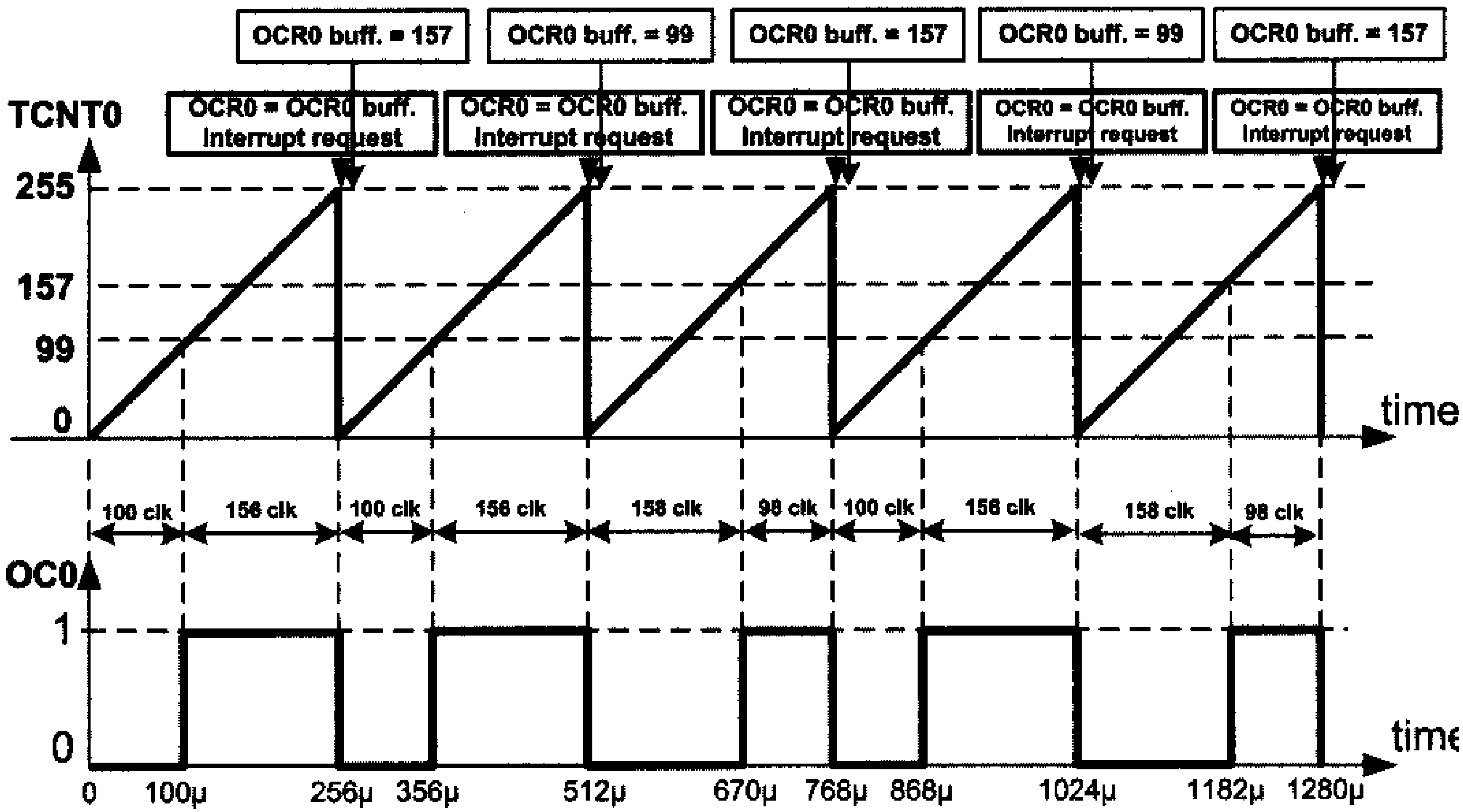
```
.INCLUDE "M32DEF.INC"
        RJMP    MAIN
.ORG 0x16                               ;Timer0 overflow interrupt
        NEG     R20                     ;Negative R20
        OUT     OCR0,R20                ;OCR0 = R20
        RETI                            ;return interrupt
MAIN:   LDI     R16,HIGH(RAMEND)
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16                 ;initialize stack
        SBI     DDRB,3                  ;OC0 as output
        LDI     R20,99                  ;R20 = 99
        OUT     OCR0,R20                ;OCR0 = 99
        LDI     R16,0x69                ;Fast PWM mode, non-invert
        OUT     TCCR0,R16
        OUT     OCR0,R20
        LDI     R16,(1<<TOIE0)
        OUT     TIMSK,R16
        SEI
HERE:   RJMP    HERE
```

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Solution:

The wave generator is in non-inverted Fast PWM mode, which means that on compare match the OC0 pin will be set high. The OCR0 register is loaded with 99; so compare match occurs when TCNT0 reaches 99. When the timer reaches the top value and over-flows, the interrupt request occurs, and the OCR0 buffer is loaded with 157 (the two's complement of 99). The next time that the timer reaches the top value, the contents of the OCR0 buffer (157) will be loaded into the OCR0 register. Then the second interrupt occurs and the OCR0 buffer will be loaded with 99 (the two's complement of 157).

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS



mashhoun@iust.ac.ir Iran Univ of Science & Tech

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Phase correct PWM mode programming of timer0

In the Phase correct PWM, the TCNT0 goes up and down like a yo-yo! First it counts up until it reaches the top value. Then it counts down until it reaches zero. The TOV0 flag is set whenever it reaches zero.

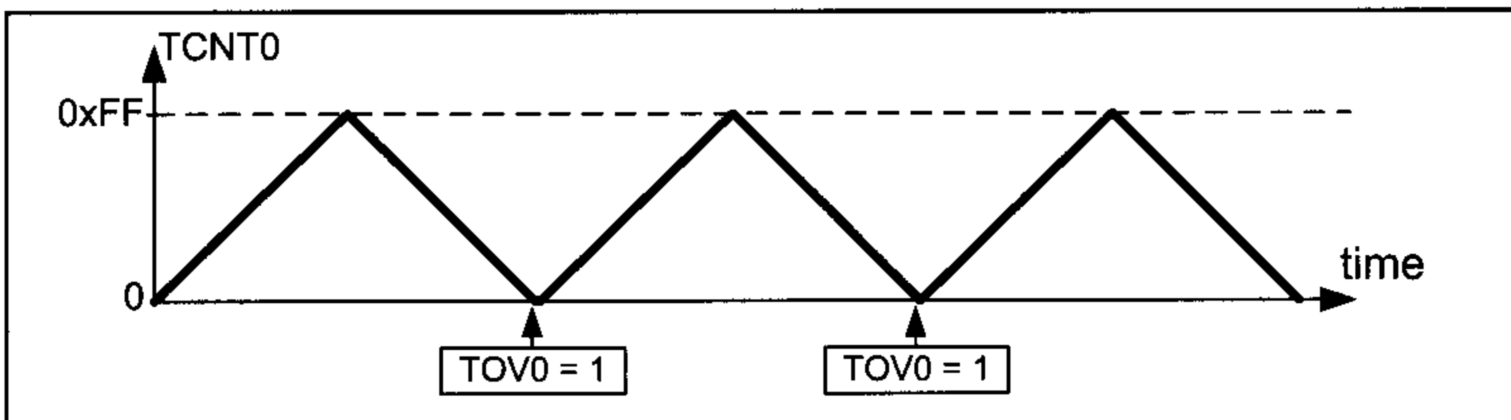


Figure 16-17. Timer/Counter 0 Phase Correct PWM Mode

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

When COM01:00 = 11, the waveform generator sets the OC0 pin on compare match when counting up, and clears it on compare match when counting down. This mode is referred as *inverted Phase correct PWM mode*. See Figures 16-20A through 16-20C.

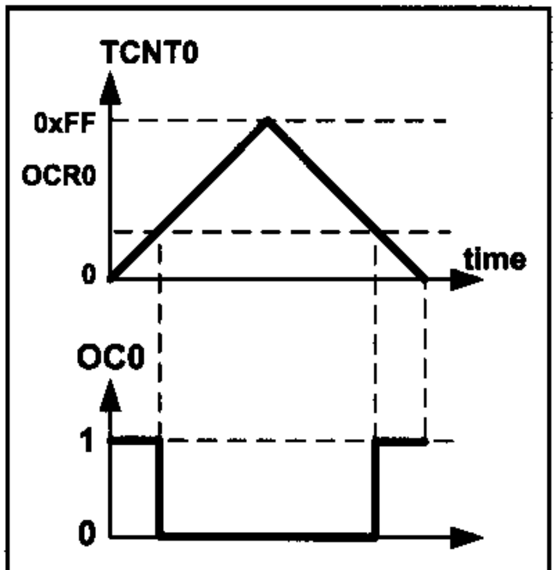


Figure 16-19A. Non-inverted

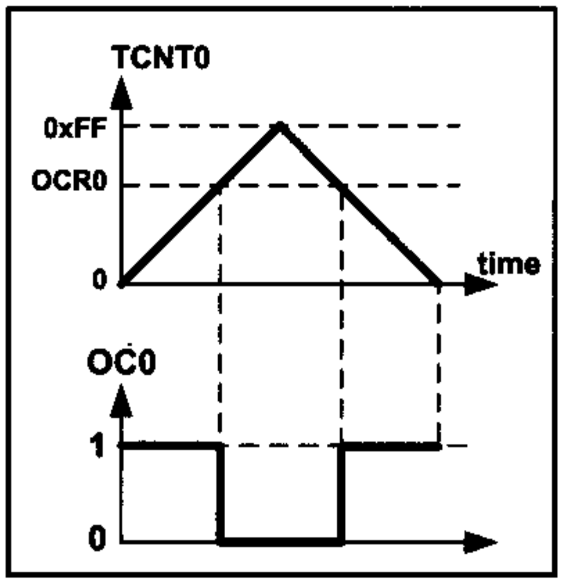


Figure 16-19B. Non-inverted

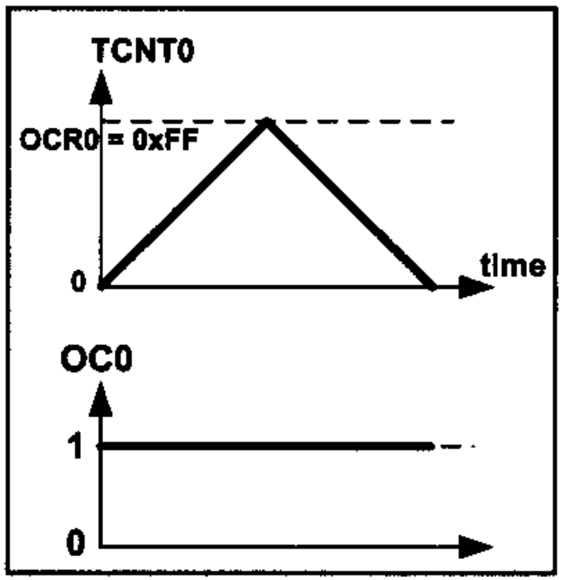


Figure 16-19C. Non-inverted

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

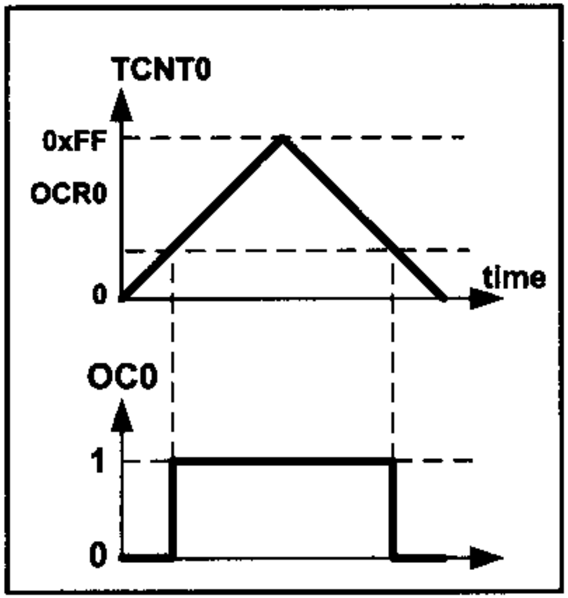


Figure 16-20A. Inverted

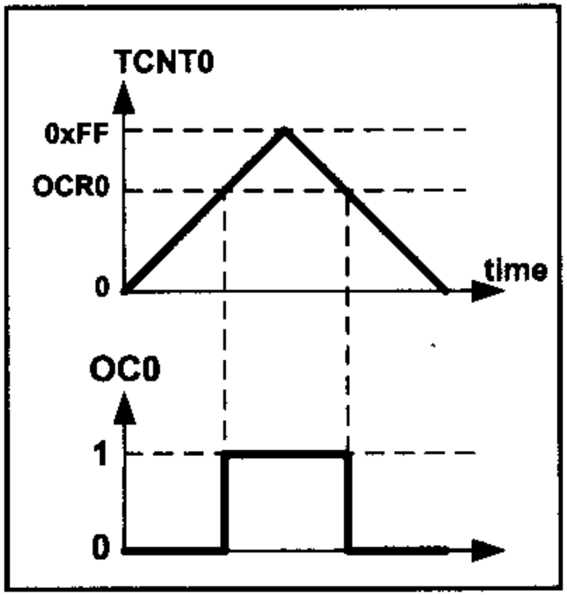


Figure 16-20B. Inverted

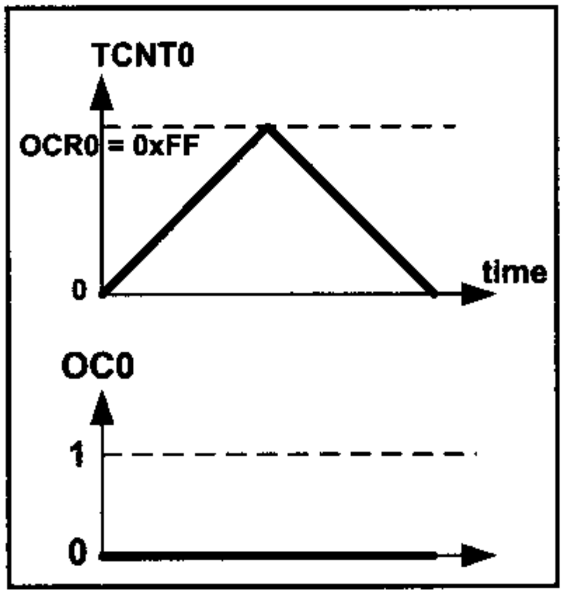


Figure 16-20C. Inverted

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Frequency of the generated wave in Phase correct PWM mode

As you see in Figure 16-21, the frequency of the generated wave is 1/510 of the frequency of timer clock. As you saw in Section 9-1, the frequency of timer clock can be selected using the prescaler. So, in 8-bit timers the frequency of the generated wave can be calculated as follows:

$$\left.\begin{aligned} F_{\text{generated wave}} &= \frac{F_{\text{timer clock}}}{510} \\ F_{\text{timer clock}} &= \frac{F_{\text{oscillator}}}{N} \end{aligned}\right\} \Rightarrow F_{\text{generated wave}} = \frac{F_{\text{oscillator}}}{510 \times N}$$

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Duty cycle of the generated wave in Phase correct PWM mode

The duty cycle of the generated mode can be determined using the OCR0 register. When COM01:00 = 10 (in non-inverted mode), the bigger OCR0 value results in a bigger duty cycle. When OCR0 = 255, the OC0 is high, 510 clocks out of 510 clocks, which means always (duty cycle = 100%). Generally speaking, the OC0 is high for a total of 2 × OCR0 clocks. See Figure 16-21. So, the duty cycle

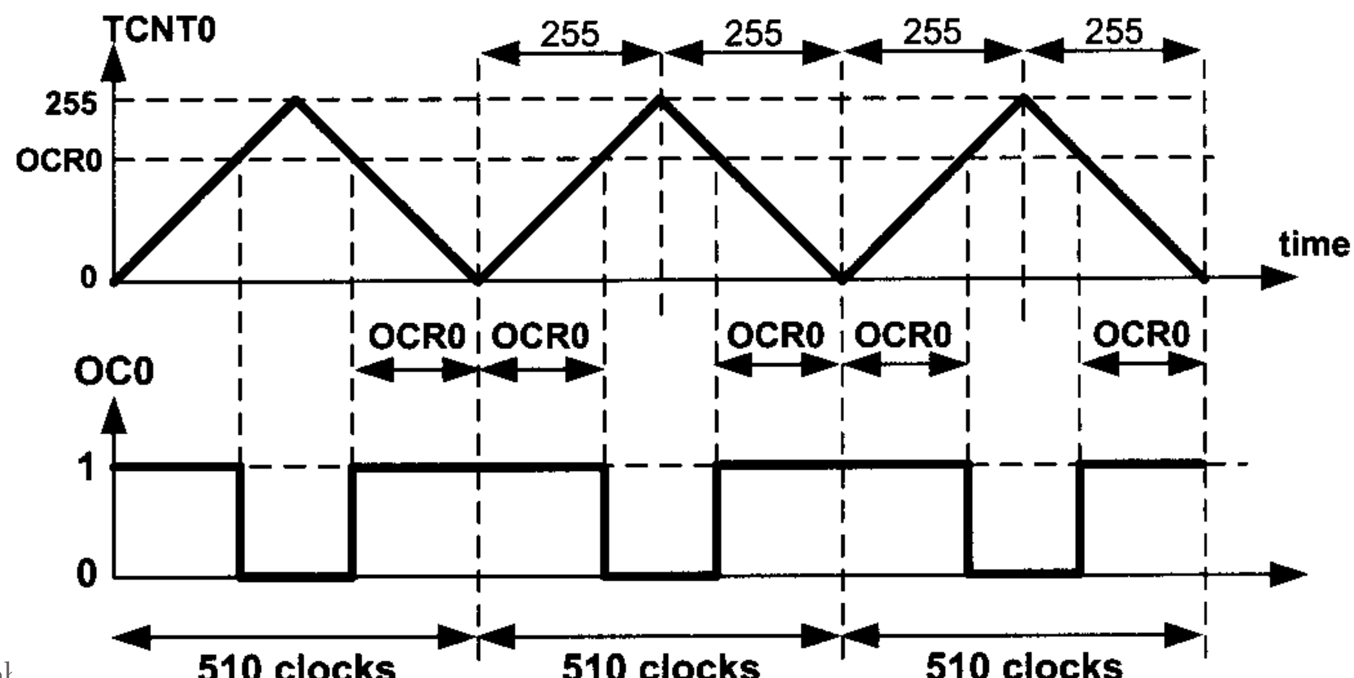


Figure 16-21. Phase Correct PWM

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

So the duty cycle can be calculated using formula in non-inverted mode:

$$\text{Duty Cycle} = \frac{2 \times OCR0}{510} \times 100 \Longrightarrow \text{Duty Cycle} = \frac{OCR0}{255} \times 100$$

Similarly, the duty cycle formula for inverted mode is as follows:

$$\text{Duty Cycle} = \frac{510 - 2 \times OCR0}{510} \times 100 \Longrightarrow \text{Duty Cycle} = \frac{255 - OCR0}{255} \times 100$$

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-12

Find the value for TCCR0 for Phase correct PWM, non-inverted PWM wave generator, and no prescaler.

Solution:

WGM01:00=01= Phase correct PWM mode

COM01:00=10= Non-inverted PWM

CS02:00=001=No prescaler

TCCR0 =

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

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-13

Assuming XTAL = 8 MHz, using non-inverted mode, write a program that generates a wave with frequency of 15,686 Hz and duty cycle of 75%.

Solution:

$15{,}686 = 8M/(510 \times N) \Rightarrow N = 8M/(15{,}626 \times 510) = 1 \Rightarrow$ Noprescaler

$75 = OCR0 \times 100/255 \Rightarrow OCR0 = 75 \times 255/100 = 191 \Rightarrow OCR0 = 191$

```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,3
        LDI     R20,191
        OUT     OCR0,R20        ;OCR0 = 191
        LDI     R20,0x61
        OUT     TCCR0,R20       ;Phase c. PWM, no prescaler, non-inverted
HERE:   RJMP    HERE
```

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-14

Find the value for TCCR0 for Phase correct PWM, inverted PWM wave generator, and prescaler = 256.

Solution:

WGM01:00 = 01 = Phase correct PWM mode

COM01:00 = 11 = Inverted PWM

CS02:00 = 100 = Scale 256

TCCR0 =

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

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-15

Assuming XTAL = 8 MHz, using inverted mode, write a program that generates a wave with frequency of 61 Hz and duty cycle of 87.5%.

Solution:

$61 = 8M/(510 \text{ x } N) \Rightarrow N = 8M/(61 \times 510) = 256$ $87.5 = 100 \text{ x } (255 - OCR0)/255$
$\Rightarrow 255 - OCR0 = (255 \times 87.5)/100 = 223 \Rightarrow OCR0 = 32$

```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,3          ;OC0 as output
        LDI     R20,32
        OUT     OCR0,R20        ;OCR0 = 32
        LDI     R20,0x74        ;from Example 16-14
        OUT     TCCR0,R20       ;Phase c. PWM, N = 256, inverted
HERE:   RJMP    HERE
```

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Difference between the wave generated by phase correct PWM and Fast PWM

In non-inverted Fast PWM, the duty cycle of the generated wave is (OCR0 +1)/256. Because the value of OCR0 is between 0 and 255, the duty cycle of the wave can be changed between 1/256 and 256/256. Therefore, in non-inverted Fast PWM the duty cycle of wave cannot be 0% (unless we turn off the wave-form generator).

Similarly, in inverted Fast PWM, the duty cycle changes between 0/256 and 255/256; thus, the duty cycle cannot be 100%. But in Phase correct PWM, the duty cycle changes between 0/255 and 255/255. Therefore, the wave can change between 0% (completely off) and 100% (completely on).

mashhoun@iust.ac.ir Iran Univ of Science & Tech

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

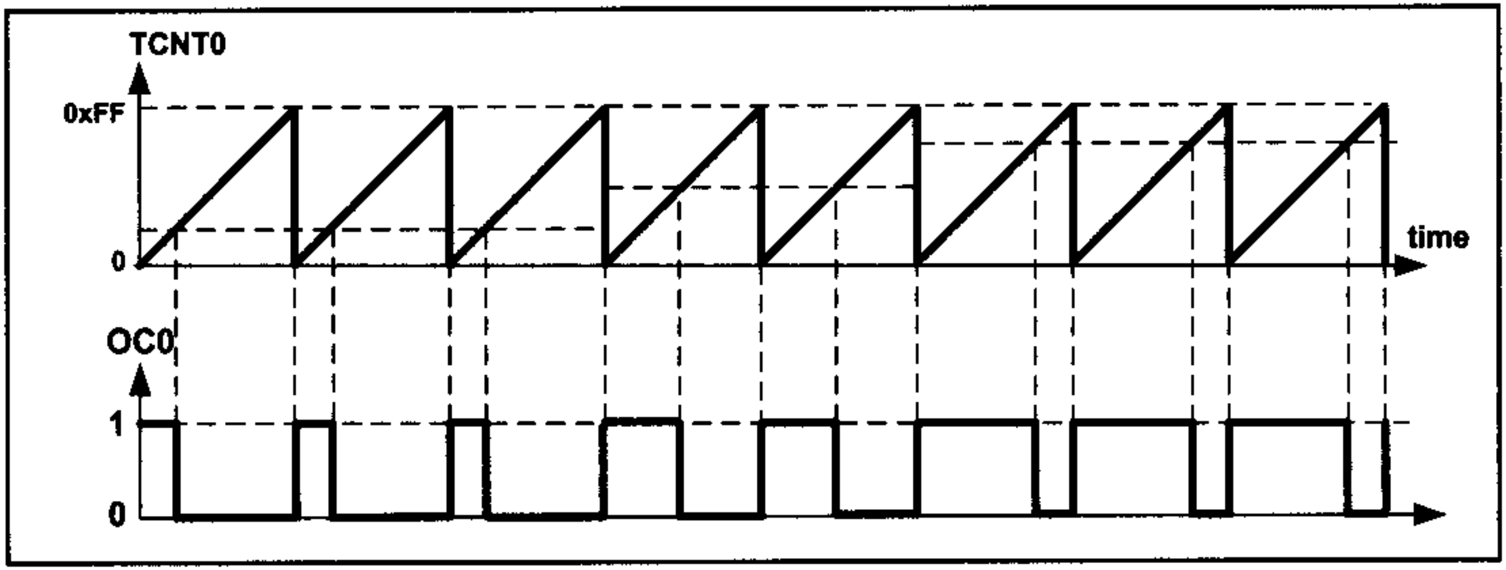
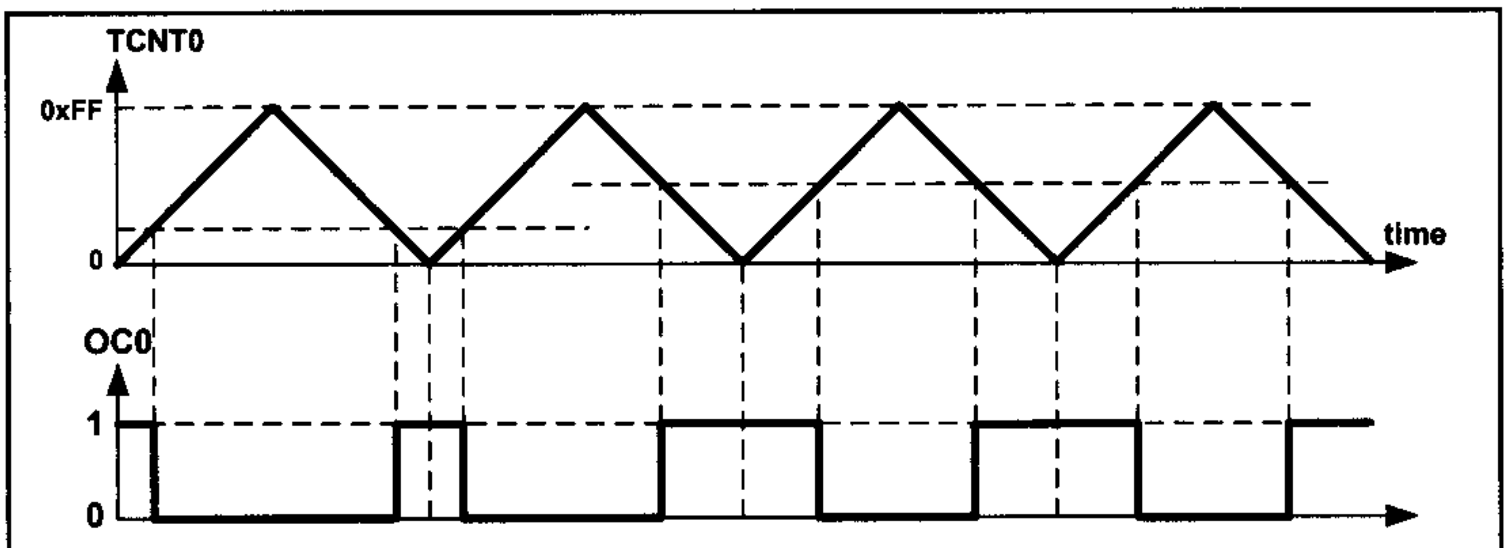


Figure 16-22. Fast PWM



PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Generating waves using Timer2

Timer2 is an 8-bit timer. Therefore, it works similar to Timer0. the differences are register names, output port, and the prescaler values of TCCRn register.

Example 16-15

Assuming XTAL = 8 MHz, using inverted mode, write a program that generates a wave with frequency of 61 Hz and duty cycle of 87.5%.

Solution:

$61 = 8M/(510 \times N) \Rightarrow N = 8M/(61 \times 510)$ 256 $87.5 = 100 \times (255 - OCR0)/255 \Rightarrow$ $255 - OCR0 = (255 \times 87.5)/100 = 223 \Rightarrow OCR0 = 32$

```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,3          ;OC0 as output
        LDI     R20,32
        OUT     OCR0,R20        ;OCRO = 32
        LDI     R20,0x74        ;from Example 16-14
        OUT     TCCR0,R20       ;Phase c. PWM, N = 256
HERE:   RJMP    HERE
```

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Generating waves using Timer2

Timer2 is an 8-bit timer. Therefore, it works similar to Timer0. the differences are register names, output port, and the prescaler values of TCCRn register

Example 16-16

Rewrite Example 16-15 using Timer2.

Solution:

According to Figure 9-11, the TCCR2 register should be loaded with:

TCCR2 =

0	1	1	1	0	1	1	0
FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20

```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,7              ;OC2 (PD7) as output
        LDI     R20,32
        OUT     OCR2,R20            ;OM = 32
        LDI     R20,0x76
        OUT     TCCR2,R20     ;Phase correct PWM, N = 256, inverted
HERE:   RJMP    HERE
```

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

8-bit programming in C

Example 16-17 (C version of Example 16-8)

Rewrite the program of Example 16-8 using C.

Solution:

```
#include "avrlio.h"
int main()
{
        DDRB (1 << 3);
        OCR0 = 191;
        TCCR0 = 0x69;   //Fast PWM, no prescaler, non-inverted
        while (1);
        return 0;
}
```


mashhoun@iust.ac.ir

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-18 (C version of Example 16-9)

Rewrite the program of Example 16-9 using C.

Solution:

```
#include "avrlio.h"
int main()
{
        DDRB (1 << 3);
        OCR0 = 95;
        TCCR0 = 0x6A;    //Fast PWM, no prescaler, non-inverted
        while (1);
        return 0;
}
```

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-19 (C version of Example 16-10)

Rewrite the program of Example 16-10 using C.

Solution:

```
#include "avrlio.h"
int main()
{
        DDRB (1 << 3);
        OCR0 = 159;
        TCCR0 = 0x7A;   //Fast PWM, no prescaler, inverted
        while (1);
        return 0;
}
```

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-20 (C version of Example 16-13)

Rewrite the program of Example 16-13 using C.

Solution:

```c
#include "avrlio.h"
int main()
{
        DDRB (1 << 3);
        OCR0 = 191;
        TCCR0 = 0x61;   //Phase C. PWM, no prescaler, non inverted
        while (1);
        return 0;
}
```


mashhoun@iust.ac.ir

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-21 (C version of Example 16-15)

Rewrite the program of Example 16-15 using C.

Solution:

```c
#include "avrlio.h"
int main()
{
        DDRB (1 << 3);
        OCR0 = 32;
        TCCR0 = 0x74;   //Phase C. PWM, N = 256, non inverted
        while (1);
        return 0;
}
```

SECTION 16.2: PWM MODES IN 8-BIT TIMERS

Example 16-22 (C version of Example 16-16)

Rewrite the program of Example 16-16 using C.

Solution:

```c
#include "avrlio.h"
int main()
{
        DDRB (1 << 7);
        OCR0 = 32;
        TCCR0 = 0x74;   //Phase C. PWM, N = 256, inverted
        while (1);
        return 0;
}
```


mashhoun@iust.ac.ir Iran Univ of Science & Tech

SECTION 16.3: PWM MODES IN TIMER1

Fast PWM Mode

In the Fast PWM, the counter counts like it does in the Normal mode. After the timer is started, it starts to count up. It counts up until it reaches its top limit.

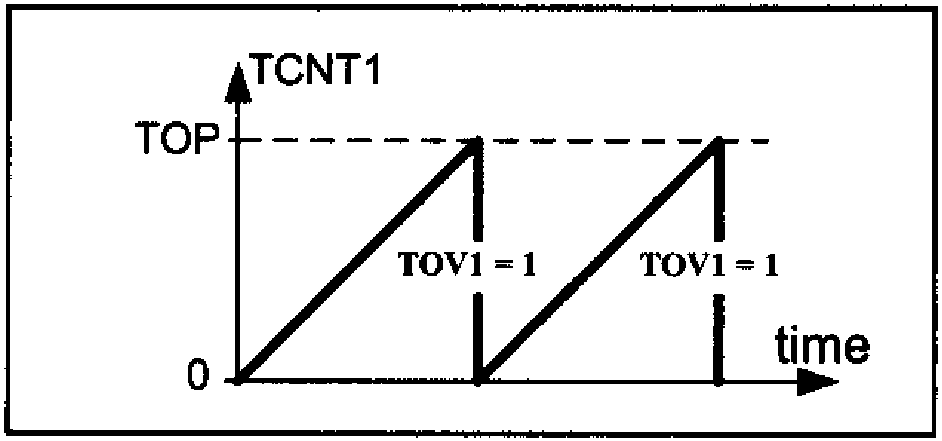


Figure 16-24. Fast PWM Mode

SECTION 16.3: PWM MODES IN TIMER1

Fast PWM Mode

We know that we have five Fast PWM modes in Timer1: modes 5, 6, 7, 14, and 15. In mode 5, 6, and 7 the top value is fixed at 0xFF, 0x1FF, and 0x3FF;

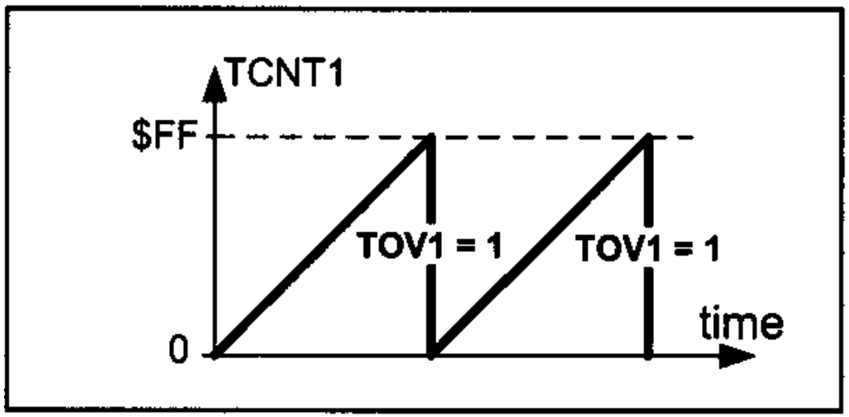


Figure 16-25. TOV in Mode 5

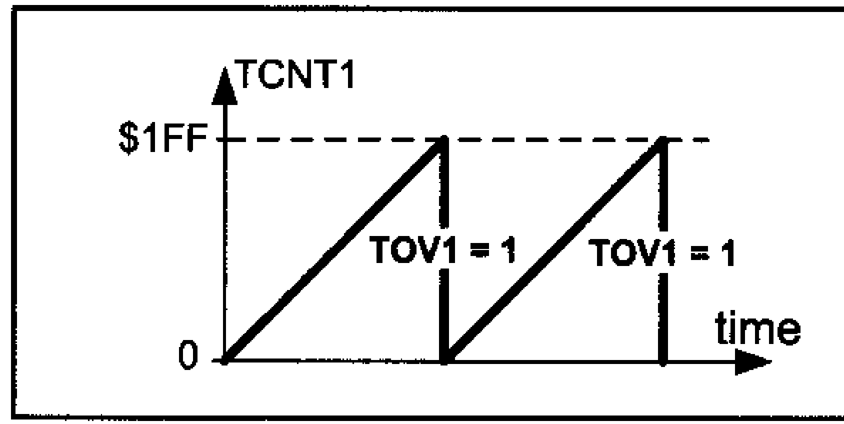


Figure 16-27. TOV in Mode 6

TOV1 flag in the timer rolls over

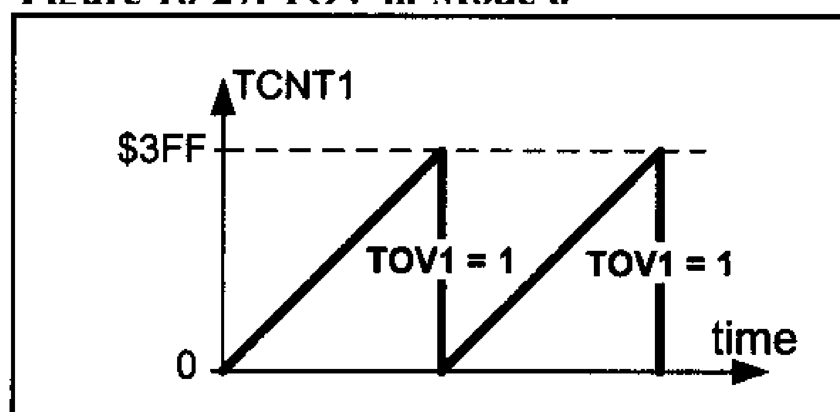


Figure 16-29. TOV in Mode 7

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

While in modes 14 and 15, the ICR1 and OCR1A registers represent the top value

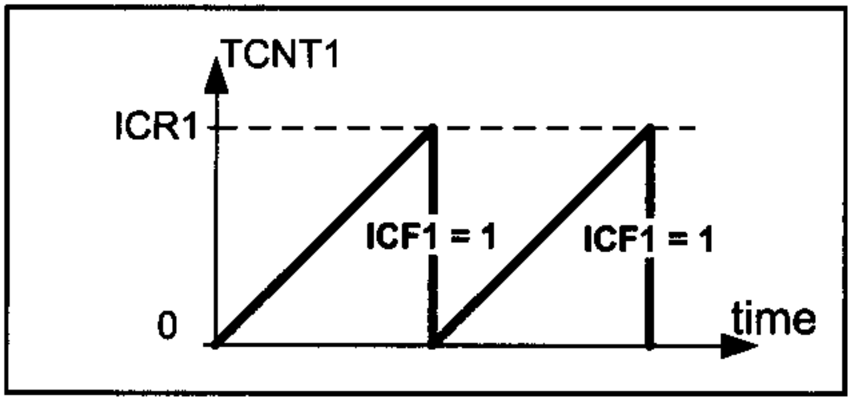


Figure 16-26. Mode 14

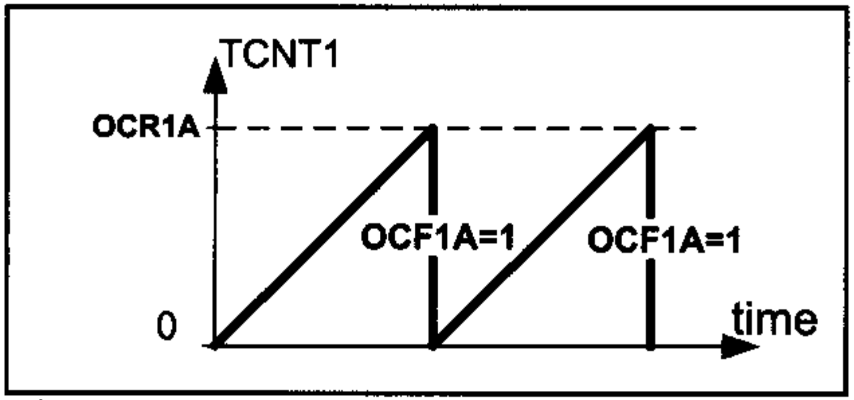


Figure 16-28. Mode 15

In Mode 14, whose top value is represented by ICR1, the ICF1 flag will be set when the timer rolls over, as shown in Figure 16-26.

In Mode 15, when the timer rolls over, the OCF1A flag will be set. See Figure 16-28.

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	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

COM1A1:COM1A0 D7 D6 Compare Output Mode for Channel A

COM1A1	COM1A0	Description
0	0	Normal port operation, OC1A disconnected
0	1	In mode 15, toggle OC1A on compare match. In other modes OC1A disconnected (Normal I/O port)
1	0	Clear OC1A on compare match. Set OC1A at Top.
1	1	Set OC1A on compare match. Clear OC1A at Top.

COM1B1:COM1B0 D5 D4 Compare Output Mode for Channel B

COM1B1	COM1B0	Description
0	0	Normal port operation, OC1B disconnected
0	1	Normal port operation, OC1B disconnected
1	0	Clear OC1B on compare match. Set OC1B at Top.
1	1	Set OC1B on compare match. Clear OC1B at Top.

FOC1A D3 Force Output Compare for Channel A

FOC1B D2 Force Output Compare for Channel B

WGM11:10 D1 D0 Timer1 mode (discussed in Figure 16-30)

Figure 16-31. TCCR1A (Timer1 Control) Register

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

In Figure 16-31 you see the reaction of the waveform generator when compare match occurs whie the timer is in Fast PWN mode.

When COM1A1:0 = 00 the OC1A pin operates as an I/O port.

When COM1A1:0 = 10 the waveform generator clears the OC1A pin whenever compare match occurs, and sets it at the top value.

This mode is called non-inverted PWM.

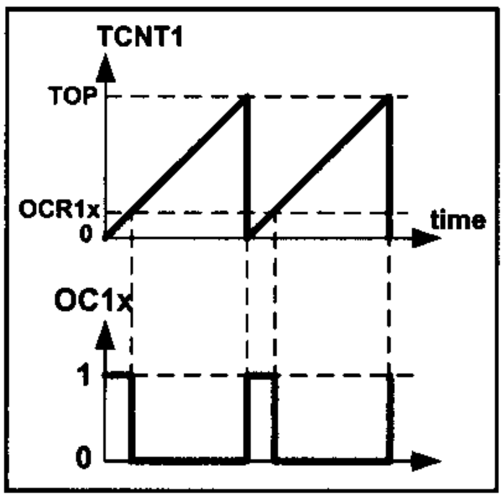


Figure 16-32A. Non-inverted

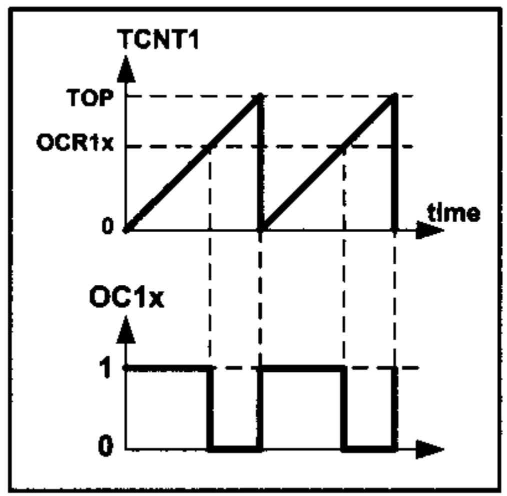


Figure 16-32B. Non-inverted

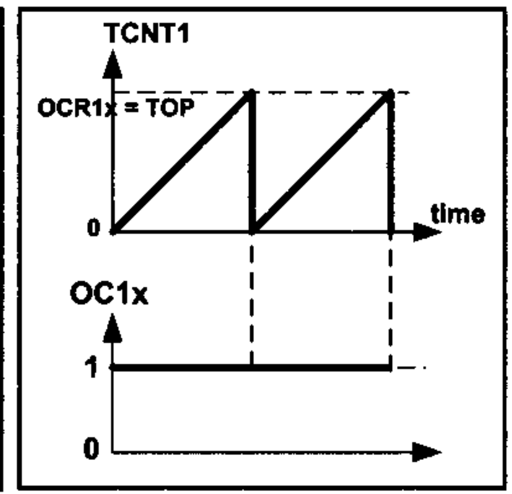


Figure 16-32C. Non-inverted

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

When COM1A1:0 = 11, the waveform generator sets the OC1A pin whenever compare match occurs, and clears it at the top value. This mose is refered to as inverted PWM mode. The duty cycle of the generated wave decreases when the value of OCR1A increases.

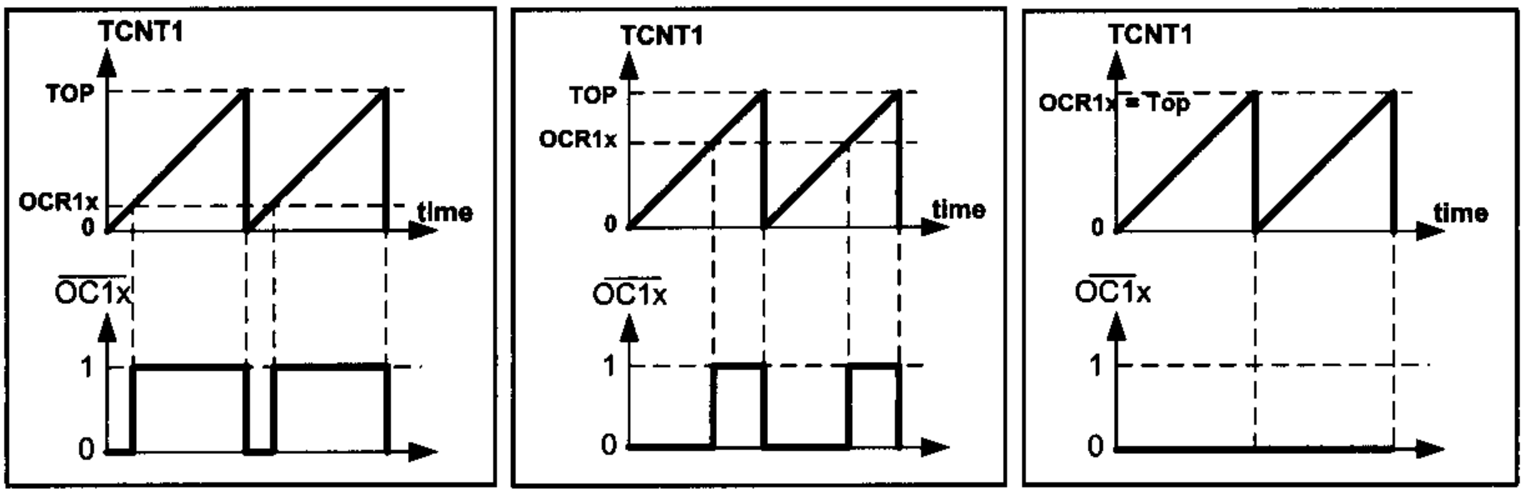


Figure 16-33A. Inverted **Figure 16-33B. Inverted** **Figure 16-33C. Inverted**

The same thing is true about the OCR1B register and COM1B1:0 bits.

SECTION 16.3: PWM MODES IN TIMER1

Frequency of the generated wave in Fast PWM mode

In Fast PWM mode, timer counts from 0 to top value and then rolls over. Thus, the frequency of generated wave is 1/(Top+1) of frequency of timer clock. (N is determined by the prescaler)

$$\left.\begin{aligned} F_{\text{generated wave}} &= \frac{F_{\text{timer clock}}}{\text{Top} + 1} \\ F_{\text{timer clock}} &= \frac{F_{\text{oscillator}}}{N} \end{aligned}\right\} \Longrightarrow F_{\text{generated wave}} = \frac{F_{\text{oscillator}}}{(\text{Top} + 1) \times N}$$

SECTION 16.3: PWM MODES IN TIMER1

Duty cycle of the generated wave in Fast PWM mode

Duty cycle of the generated mode can be determined using OCR1x register. When COM1x1:0 = 10 (in non-inverting mode), the bigger OCR1x value results in a bigger duty cycle. When OCR1x = Top, the OC1 is always high (duty cycle = 100%). Generally speaking the OC1x is high for a total of OCR1x + 1 clocks.

$$\text{Duty Cycle} = \frac{\text{OCR1x} + 1}{\text{Top} + 1} \times 100$$

In inverted mode, the duty cycle can be calculated using the following formula:

$$\text{Duty Cycle} = \frac{\text{Top} - \text{OCR1x}}{\text{Top} + 1} \times 100$$

SECTION 16.3: PWM MODES IN TIMER1

Example 16-23

Calculate the value for the OCRIB register to generate a wave with duty cycle of 75% for each of the following modes:

(a) Mode 5, non-inverted mode

(b) Mode 7, inverted mode

(c) Mode 6, non-inverted mode

(d) Mode 5, inverted mode

(e) Mode 7, non-inverted mode

SECTION 16.3: PWM MODES IN TIMER1

Example 16-23 (Cont.)

Solution:

(a) In mode 5, Top=0xFF=255. Thus, 75 = (OCRlx + 1)×100/(Top+1) ⇒ OCR1x + 1 = 75×256/100 =192 ⇒ OCR1B = 191

(b) In mode 7, Top=0x3FF =1023. Thus, 75 = (Top-OCR1x)×100/(Top+1) ⇒ 1023-OCRlx= 75×1024/100 = 768 ⇒ OCR1B = 255

(c) In mode 6, Top=0x1FF = 511. Thus, 75 = (OCRlx + 1×100/(Top+1) ⇒ OCRlx+1=75×512/100= 38 ⇒ OCR1A = 383

(d) In mode 5, Top=0xFF=255. Thus, 75=(Top - OCR1x)×100/(Top + 1) ⇒ 75=(255-OCR1x)×100/256 ⇒255-OCRlx=75×256/100=192 ⇒OCRIB = 255-192 = 63

(e) In mode 7, Top=0x3FF = 1023. Thus, 75 = (OCRlx+1)×100/(Top+1)⇒ OCRlx+1=75 ×1024/100=768 ⇒ OCR1B = 767

SECTION 16.3: PWM MODES IN TIMER1

Example 16-24

Find the values for TCCR1A and TCCR1B to initialize Timer1 for mode 5 (Fast PWM mode, top = 0xFF), non-inverted PWM wave generator, and no prescaler, using wave-form generator A.

Solution:

WGM13:10 = 0101 = Fast PWM mode

CS02:00 = 001 = No prescaler

COM01:00 = 10 = Non-inverted PWM

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

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

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

Example 16-25

Assuming XTAL = 8 MHz, using non-inverted mode, and mode 5, write a program that generates a wave with frequency of 31,250 Hz and duty cycle of 75%.

Solution:

$31{,}250 = 8M / (256 \times N) \Rightarrow N = 8M / (31{,}250 \times 256) = 1 \Rightarrow$ No prescaler

```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,5              ;PD5 = output
        LDI     R16,HIGH(191)       ;from Example 16-23
        OUT     OCR1AH,R16          ;Temp = 0x00
        LDI     R16,LOW(191)        ;R16 = 191
        OUT     OCR1AL,R16          ;OCR1A = 191
        LDI     R16,0x81            ;from Example 16-24
        OUT     TCCR1A,R16          ;COM1A = non-inverted
        LDI     R16,0x09
        OUT     TCCR1B,R16          ;WGM = mode 5, clock = no scaler
HERE:   RJMP    HERE
```

SECTION 16.3: PWM MODES IN TIMER1

Example 16-26

Assuming XTAL = 8 MHz, using non-inverted mode, and mode 7, write a program that generates a wave with frequency of 7812.5 Hz and duty cycle of 75%.

Solution:

$7812.5 = 8M / (1024 \times N) \Rightarrow N = 8M / (7812.5 \times 1024) = 1 \Rightarrow$ No prescaler

```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,5              ;PD5 = output
        LDI     R16,HIGH(767)       ;R16 = the high byte
        OUT     OCR1AH,R16
        LDI     R16,LOW(767)        ;R16 = the low byte
        OUT     OCR1AL,R16          ;OCR1A = 767 (from Example 16-23)
        LDI     R16,0x83
        OUT     TCCR1A,R16          ;COM1A = non-inverted
        LDI     R16,0x09
        OUT     TCCR1B,R16          ;WGM = mode 7, clock = no scaler
HERE:   RJMP    HERE                ;wait here forever
```

SECTION 16.3: PWM MODES IN TIMER1

Example 16-27

Assuming XTAL = 8 MHz, using non-inverted mode and mode 6, write a program that generates a wave with frequency of 1,953 Hz and duty cycle of 60%.

Solution:

In mode 6, Top=0x1FF=511. Thus,

$60=(OCRlx+1)\times100/(Top + 1) \Rightarrow OCRlx + 1 = 60\times512/100=307 \Rightarrow OCR1B=306$

$1953=8M/(512\times N) \Rightarrow N=8M/(1953\times512)=8 \Rightarrow prescaler=1:8 \Rightarrow CS12:0 = 010$

```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,5              ;PD5 = output
        LDI     R16,HIGH(306)       ;R16 = the high byte
        OUT     OCR1AH,R16          ;Temp = R16
        LDI     R16,LOW(306)        ;R16 = the low byte
        OUT     OCR1AL,R16          ;OCR1A = 306
        LDI     R16,0x82
        OUT     TCCR1A,R16          ;COM1A = non-inverted.
        LDI     R16,0x0A
        OUT     TCCR1B,R16          ;WGM = mode 6, clock = no scaler
HERE:   RJMP    HERE                ;wait here forever
```

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

Example 16-28

Rewrite Example 16-27 using inverted mode.

Solution:

60=(Top-OCR1x)×100/(Top + 1). ⇒ 511-OCRlx=60×512/100=307 ⇒ OCR1B = 511 -307 = 204

TCCR1A =

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

TCCR1B =

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

```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,5              ;PD5 = output
        LDI     R16,HIGH(204)       ;Temp = the high byte
        OUT     OCR1AH,R16
        LDI     R16,LOW(204)
        OUT     OCR1AL,R16          ;OCR1A = 204
        LDI     R16,0xB2
        OUT     TCCR1A,R16          ;COM1A = inverted
        LDI     R16,0x0A
        OUT     TCCR1B1,R16         ;WGM = mode 6, clock = no scaler
HERE:   RJMP    HERE                ;wait here forever
```

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

Loading values into the OCR1A and OCR1B registers in PWM modes

In the non-PWM modes (CTC mode and Normal mode), when we load a value into the OCR1x register, the value will be loaded instantly, but in the PWM modes, there is a buffer between us and OCR1A and OCR1B registers. When we read/write a value from/into the OCR1A or OCR1B register we are dealing with the buffer. The contents of the buffer will be loaded into the OCR1A/OCR1B registers only when the TCNT1 reaches its topmost value.

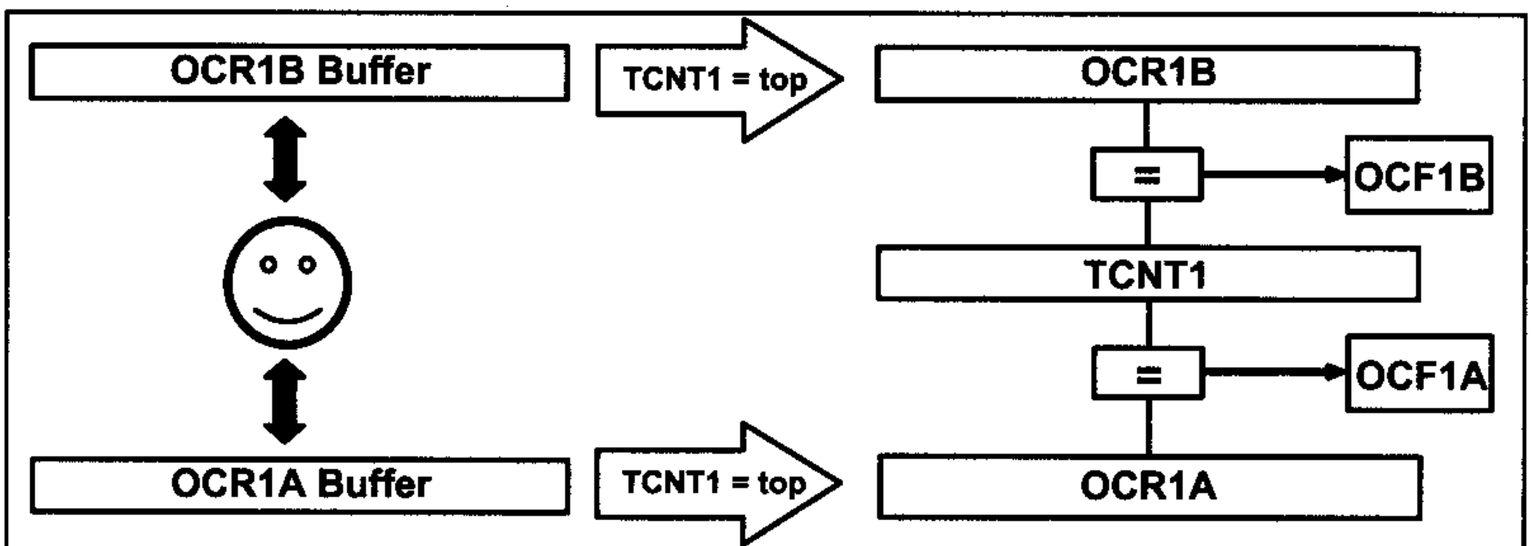


Figure 16-34. OCRnx Buffer in PWM Modes

SECTION 16.3: PWM MODES IN TIMER1

Example 16-29

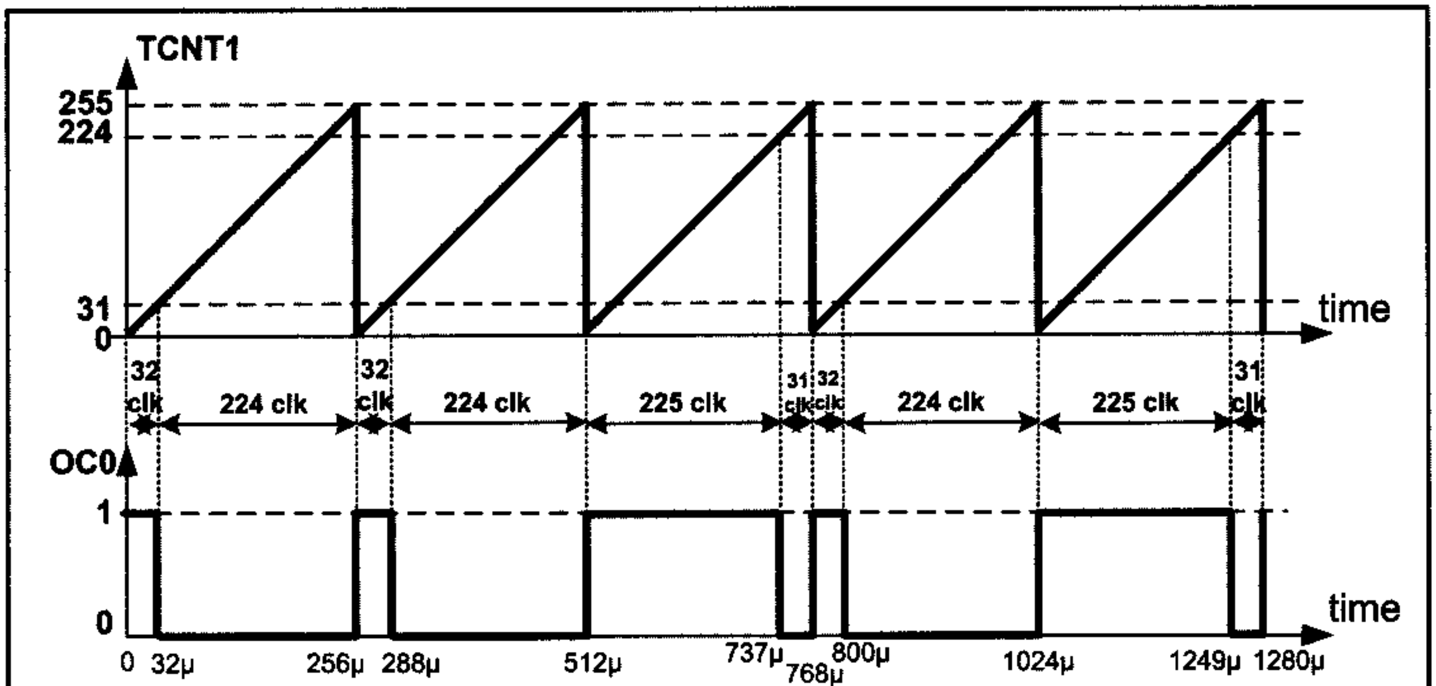
Draw the wave generated by the following program. Assume XTAL = 1 MHz.

```
 .INCLUDE "M32DEF.INC"
        RJMP    MAIN
.ORG 0x12                               ;Timer' overflow interrupt vector
        OUT     OCR1AH,R19              ;OCR1AH = R19 = 0
        NEG     R20
        OUT     OCR1AL,R20              ;OCR1A = R20
        RETI                            ;return from interrupt
MAIN:   LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,HIGH(RAMEND)
        OUT     SPH,R16                 ;initialize stack pointer
        SBI     DDRD,5                  ;PD5 = output
        LDI     R19,0
        OUT     OCR1AH,R19              ;Temp = 0x00
        LDI     R20,31
        OUT     OCR1AL,R20              ;OCR1A = 31
        LDI     R16,0x81
        OUT     TCCR1A,R16              ;COM1A = non-inverted.
        LDI     R16,0x0A
        OUT     TCCR1B,R16              ;WGM = mode 5, clock = no scaler
        LDI     R16,(1<<TOIE1)
        OUT     TIMSK,R16               ;enable timer interrupt
        SEI
HERE:   RJMP    HERE                    ;wait here forever
```

Example 16-29 (Cont.)

Solution:

The wave generator is in non-inverted Fast PWM mode, which means that on compare match the OC1A pin will be set high. The OCR1A register is loaded with 31, so compare match occurs when TCNT1 reaches 31. When the timer reaches top and overflows, the interrupt request occurs, and OCR1A buffer is loaded with 224 (the two's complement of 31). The next time that the timer reaches the top value the contents of the OCR1A buffer (224) will be loaded into the OCR1A register. Then the second interrupt occurs and OCR1A buffer will be loaded with 31 (the two's complement of 224).



SECTION 16.3: PWM MODES IN TIMER1

Generating waves with different frequencies (case study)

As we mentioned earlier, the frequency of the generated wave is equal to $F_{oscillator}$ /[N*(Top + 1)]. In the modes 5, 6, and 7, the Top value is fixed. Therefore, in these modes the only way to change the frequency of the generated wave is to change N. In following figure you see the different frequencies that can be generated using modes 5,6, and 7.

Prescaler	1	1:8	1:64	1:256	1:1024
Mode = 5	$\frac{F_{oscillator}}{1\times256}$	$\frac{F_{oscillator}}{8\times256}$	$\frac{F_{oscillator}}{64\times256}$	$\frac{F_{oscillator}}{256\times256}$	$\frac{F_{oscillator}}{1024\times256}$
Mode = 6	$\frac{F_{oscillator}}{1\times512}$	$\frac{F_{oscillator}}{8\times512}$	$\frac{F_{oscillator}}{64\times512}$	$\frac{F_{oscillator}}{256\times512}$	$\frac{F_{oscillator}}{1024\times512}$
Mode = 7	$\frac{F_{oscillator}}{1\times1024}$	$\frac{F_{oscillator}}{8\times1024}$	$\frac{F_{oscillator}}{64\times1024}$	$\frac{F_{oscillator}}{256\times1024}$	$\frac{F_{oscillator}}{1024\times1024}$

Figure 16-35. Different Frequencies Can Be Made Using Modes 5, 6, and 7

SECTION 16.3: PWM MODES IN TIMER1

Thus in these modes we can make a very limited number of frequencies. In modes 14 and 15 the Top value can be specified by ICR1 and OCR1A registers.

Example 16-30

Assuming XTAL = 8 MHz, find TCCR1A and TCCR1B to generate a wave with frequency of 80 kHz using mode 14.

Solution:

$80K=8M/[N\times(Top + 1)] \Rightarrow N\times(Top + 1)=8M/80K=100$

$\Rightarrow N\times (Top + 1) = 100 \Rightarrow N = 1; Top + 1 = 100$

$\Rightarrow Top = 99 \Rightarrow ICR1 = 99$

$\Rightarrow N= 1 \Rightarrow CS12:0 = 001$

TCCR1A =

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

TCCR1B =

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

SECTION 16.3: PWM MODES IN TIMER1

Example 16-31

Calculate the OCR1B to generate a wave with duty cycle of 20% in each of the following modes:

(a) mode 14, inverted mode, ICR1 = 45,

(b) mode 15, non-inverted mode, OCR1A = 124, and

(c) mode 14, non-inverted mode, ICR1 = 99.

Solution:

SECTION 16.3: PWM MODES IN TIMER1

Solution:

a) In mode 14, Top = ICR1 = 45. Thus,
$20 = (Top - OCR1x) \times 100 / (Top + 1) \Rightarrow 45 - OCRlx = 20 \times 46 / 100 = 9$
$\Rightarrow \boxed{OCR1A = 36}$

b) (b) In mode 15, Top = OCR = 124. Thus,
$20 = (OCR1x + 1) \times 100 / (124 + 1). \Rightarrow OCR1x + 1 = 20 \times 125 / 100 = 25$
$\Rightarrow \boxed{OCR1x = 24}$

c) In mode 14, Top = ICR1 = 99. Therefore,
$20 = (OCR1x + 1) \times 100 / (99 + 1) \Rightarrow OCR1x + 1 = 20 \Rightarrow \boxed{OCR1x = 19}$

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

Example 16-32

Assume XTAL = 8 MHz. Using mode 14 write a program that generates a wave with duty cycle of 20% and frequency of 80 kHz.

Solution:

```
.INCLUDE "M32DEF.INC"
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,HIGH(RAMEND)
        OUT     SPH,R16                 ;initialize stack pointer
        SBI     DDRD,5                  ;PD5 = output
        LDI     R16,HIGH(99)
        OUT     ICR1H,R16               ;Temp = 0
        LDI     R16,LOW(99)
        OUT     ICR1L,R16               ;ICR1 = 99
        LDI     R16,HIGH(19)
        OUT     OCR1AH,R16              ;Temp = 0
        LDI     R16,LOW(19)
        OUT     OCR1AL,R16              ;OCR1A = 19 (from Example 16-31)
        LDI     R16,0x82                ;from Example 16-30
        OUT     TCCR1A,R16              ;COM1A = non-inverted
        LDI     R16,0x19                ;from Example 16-30
        OUT     TCCR1B,R16              ;WGM = mode 14, clock = no scaler
HERE:   RJMP    HERE                    ;wait here forever
```

SECTION 16.3: PWM MODES IN TIMER1

Phase correct PWM Mode

In the phase correct PWM, the timer counts up until it reaches the top value then counts down until it reaches zero. The TOV1 flag will be set when the timer returns to zero,

There are five phase correct PWM modes: modes 1,2,3,10, and 11

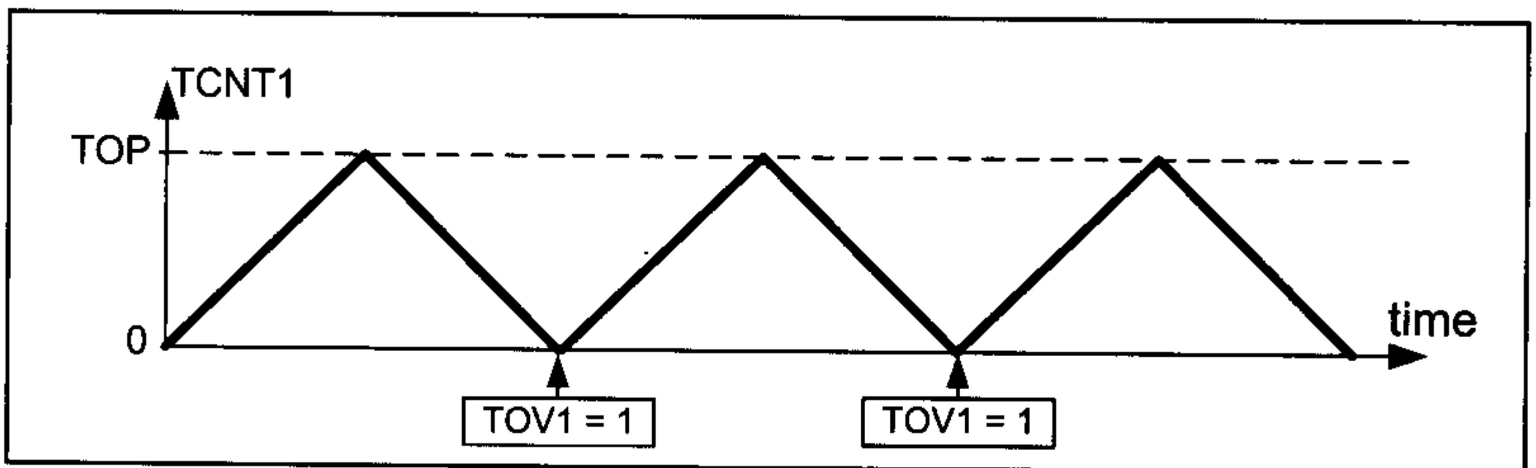


Figure 16-36. Timer/Counter 1 Phase Correct PWM Mode

SECTION 16.3: PWM MODES IN TIMER1

Phase correct PWM Mode

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

SECTION 16.3: PWM MODES IN TIMER1

Phase correct PWM Mode

In modes 1, 2, and 3 the top values are 0xFF, 0x1FF, and 0x3FF. In mode 10, the top value is defined by the ICR1 register; and in mode 11, the OCR1A register represents the top value.

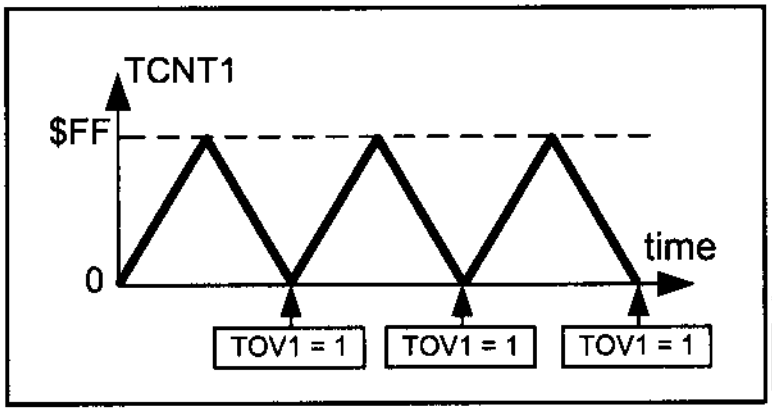


Figure 16-38. Mode 1

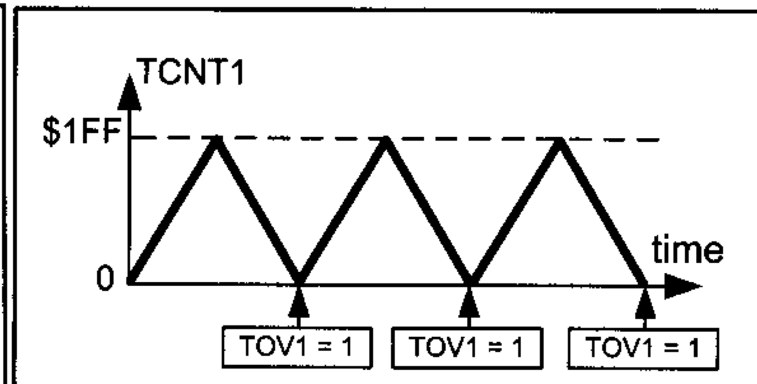
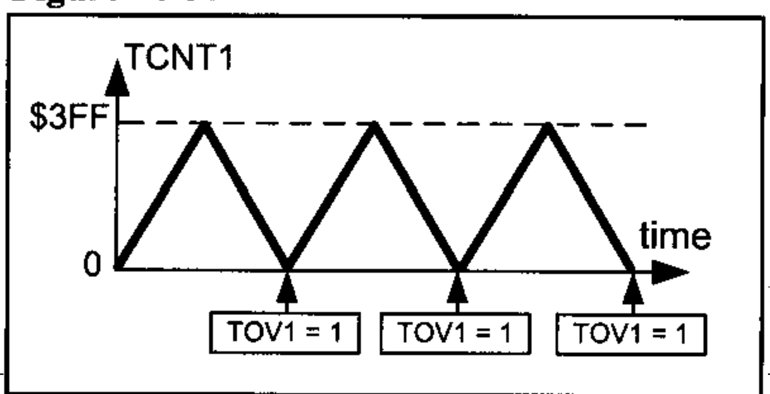
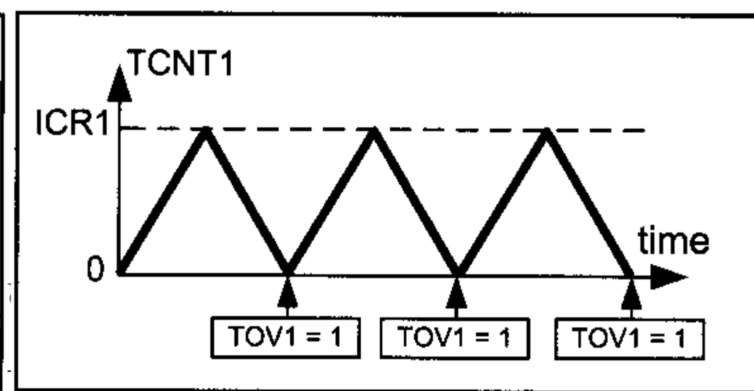


Figure 16-39. Mode 2





SECTION 16.3: PWM MODES IN TIMER1

Phase correct PWM Mode

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	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

COM1A1:COM1A0 D7 D6 Compare Output Mode for Channel A

COM1A1	COM1A0	Description
0	0	Normal port operation, OC1A disconnected
0	1	In mode 9 or 14 toggles on compare match. In other modes OC1A is disconnected (Normal I/O port).
1	0	Clear OC1A on compare match when up-counting. Set OC1A on compare match when down-counting.
1	1	Set OC1A on compare match when up-counting. Clear OC1A on compare match when down-counting.

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR

SECTION 16.3: PWM MODES IN TIMER1

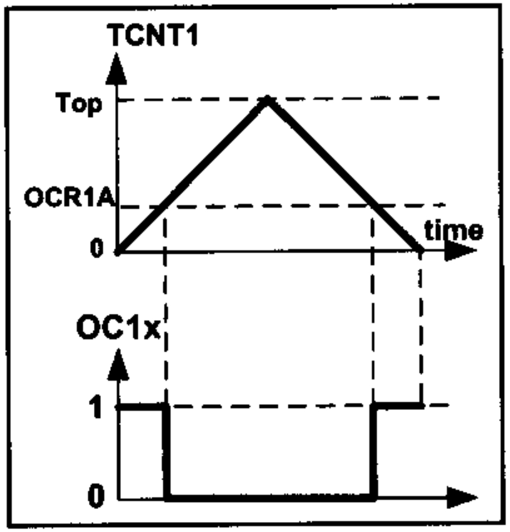


Figure 16-44A. Non-inverted

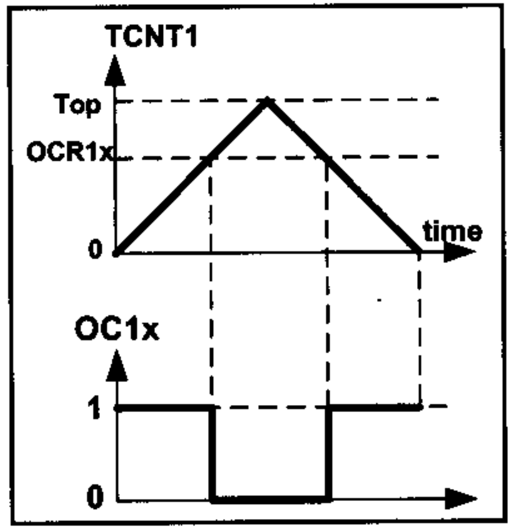


Figure 16-44B. Non-inverted

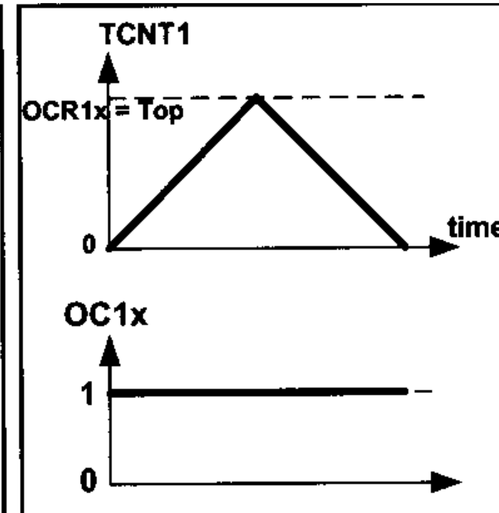


Figure 16-44C. Non-inverted

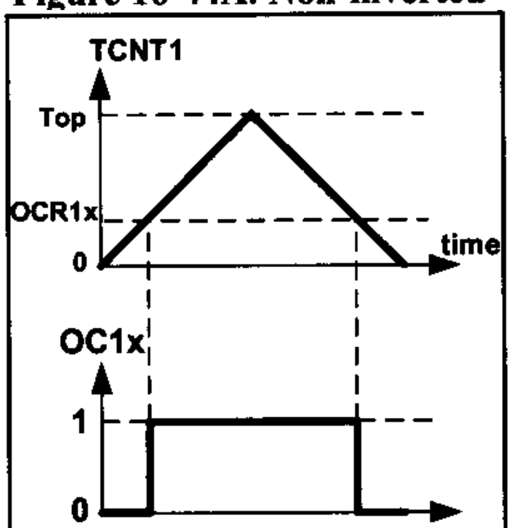


Figure 16-45A. Inverted

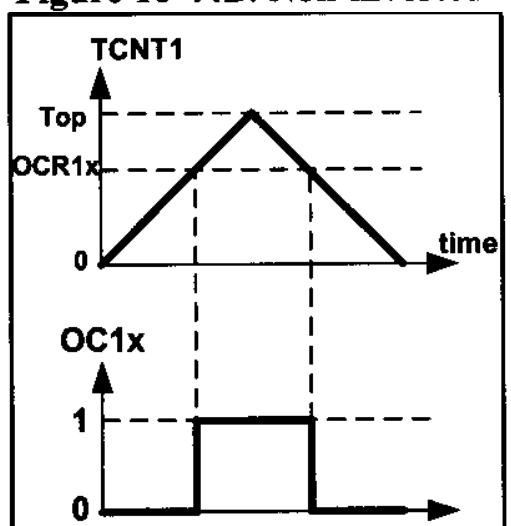


Figure 16-45B. Inverted

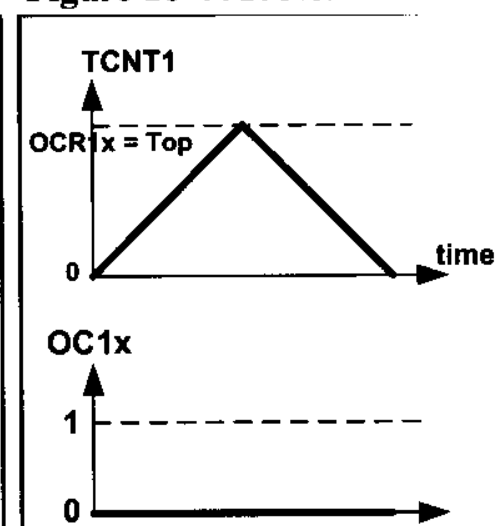


Figure 16-45C. Inverted

SECTION 16.3: PWM MODES IN TIMER1

Frequency of generated wave in phase correct PWM mode

The frequency of generated wave is 1/2TOP of the frequency of timer clock. The 8-bit timers the frequency of the generated wave can be calculated as follows:

$$\left.\begin{aligned} F_{\text{generated wave}} &= \frac{F_{\text{timer clock}}}{2 \times \text{Top}} \\ F_{\text{timer clock}} &= \frac{F_{\text{oscillator}}}{N} \end{aligned}\right\} \Rightarrow F_{\text{generated wave}} = \frac{F_{\text{oscillator}}}{2 \times N \times \text{Top}}$$

mashhoun@iust.ac.ir Iran Univ of Science & Tech

SECTION 16.3: PWM MODES IN TIMER1

Phase correct PWM Mode

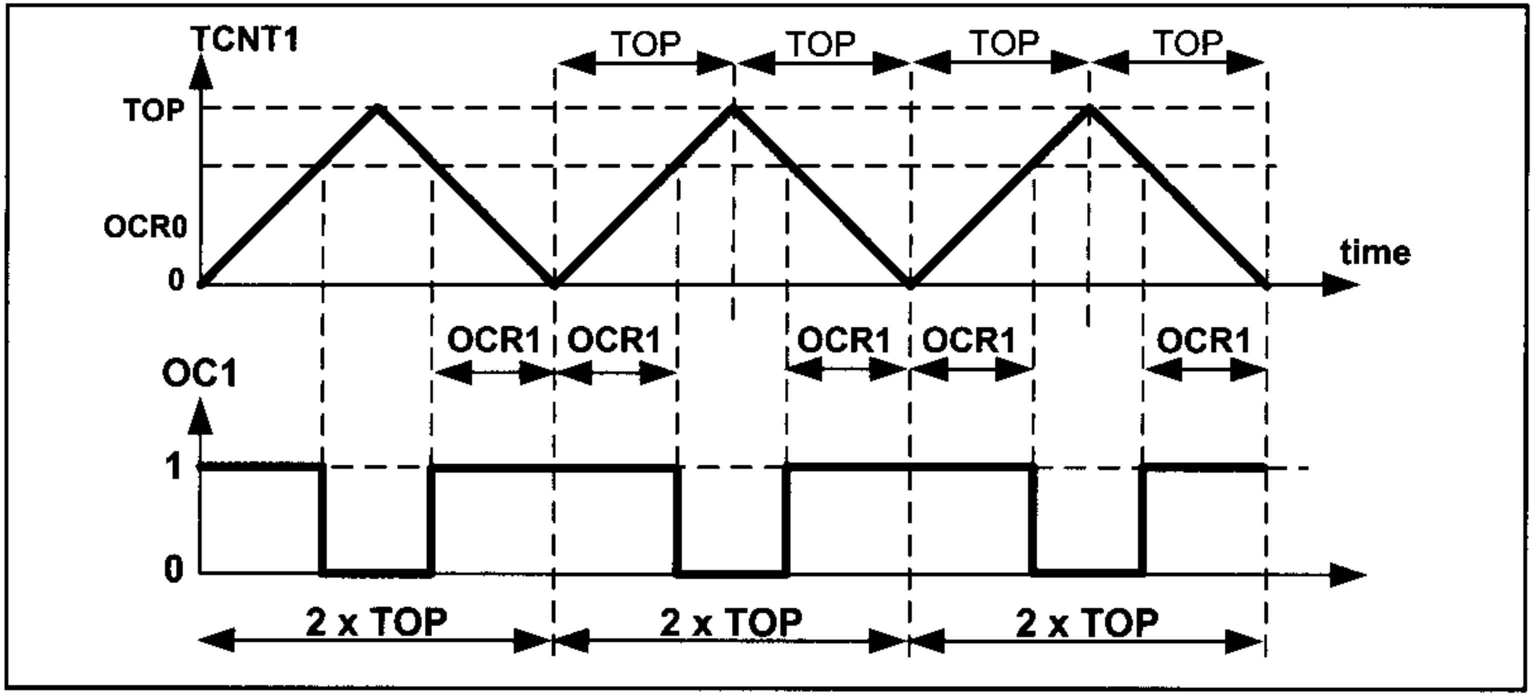


Figure 16-46. Timer/Counter 1 Phase Correct PWM Mode

SECTION 16.3: PWM MODES IN TIMER1

Duty cycle of the generated wave in Phase correct PWM mode

The duty cycle of the generated mode can be determined using OCR1x register. When COM1x1:0 = 10 (in non-inverted mode), the bigger OCR1x value results in a bigger duty cycle. When OCR1x = Top, the OC1x is always high (duty cycle = 100%). Generally speaking, OC1x is high for a total of OCR1x clocks.

So the duty cycle can be calculated using the following formula in non-inverted mod:

$$\text{Duty Cycle} = \frac{2 \times \text{OCR1A}}{2 \times \text{Top}} \times 100 \Longrightarrow \text{Duty Cycle} = \frac{\text{OCR1A}}{\text{Top}} \times 100$$

Similarly, the duty cycle formula for inverted mode is as follows:

$$\text{Duty Cycle} = \frac{2 \times \text{Top} - 2 \times \text{OCR1A}}{2 \times \text{Top}} \times 100 \Longrightarrow \text{Duty Cycle} = \frac{\text{Top} - \text{OCR1A}}{\text{Top}} \times 100$$

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

Example 16-35

Find the values for TCCR1A and TCCR1B to initialize Timer1 for mode 1 (Phase correct PWM mode, top = 0xFF), non-inverted PWM wave generator, and no prescaler.

Solution:

WGM13:10 = 0001 = Phase correct PWM mode
CS02:00 = 001 = No prescaler
COM01:00 = 10 = Non-inverted PWM

TCCR1A =

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

TCCR1B =

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

SECTION 16.3: PWM MODES IN TIMER1

Example 16-36

Calculate the OCR1B to generate a wave with duty cycle of 75% in each of the following modes:

(a) Mode 1, non-inverted mode

(b) Mode 3, inverted mode

(c) Mode 2, non-inverted mode

(d) Mode 2, inverted mode

(e) Mode 1, inverted mode

Solution:

a) In mode 1, Top = 0xFF= 255. So,

$75 = OCR1x \times 100 / Top \Rightarrow OCR1x = 75 \times 255 / 100 = 192$

$\Rightarrow \boxed{OCR1B = 191}$

mashhoun@iust.ac.ir Iran Univ of Science & Tech

SECTION 16.3: PWM MODES IN TIMER1

Solution:

b) In mode 3, Top=0x3FF=1023. So,
75 = (Top-OCR1x)×100/Top ⇒ 1023-OCRlx = 75 × 1023 / 100 = 767
⇒ $\boxed{\text{OCR1B = 255}}$

c) In mode 2, Top = 0x1FF = 511. So,
75 = OCR1x×100/Top+OCR1x=75×511/100=383
⇒ $\boxed{\text{OCR1B = 383}}$

d) In mode 2, Top=0x1FF=511. So,
75=(Top-OCR1x)×100/Top ⇒ 75=(511-OCR1x)×100/511 ⇒
511–OCR1x = 75×511/100-383 ⇒ $\boxed{\text{OCR1B = 511-383=128}}$

e) (e) In mode 1, Top=0xFF=255. So,
75-(Top-OCR1x)×100/Top ⇒ 75=(255-OCR1x)×100/255 ⇒
255–OCR1x= 75×255/100=191 ⇒ $\boxed{\text{OCR1B=255-191=64}}$

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

Example 16-37

Assuming XTAL = 8 MHz, using non-inverted mode and mode 1, write a program that generates a wave with frequency of 15,686 Hz and duty cycle of 75%.

Solution:

$15{,}686 = 8M/(510 \times N) \Rightarrow N = 8M/(15{,}686 \times 510) = 1 \Rightarrow$ No prescaler

```
.INCLUDE "M32DEF.INC"
        SBI     DDRD,5              ;PD5 = output
        LDI     R16,HIGH(191)       ;from Example 16-36
        OUT     OCR1AH,R16          ;Temp = 0x00
        LDI     R16,LOW(191)        ;R16 = 191
        OUT     OCR1AL,R16          ;OCR1A = 191
        LDI     R16,0x81            ;from Example 16-35
        OUT     TCCR1A,R16          ;COM1A = non-inverted
        LDI     R16,0x01
        OUT     TCCR1B,R16          ;WGM = mode 1, clock = no scaler
HERE:   RJMP    HERE
```

SECTION 16.3: PWM MODES IN TIMER1

Generating waves with different frequencies (case study)

As mentioned earlier, the frequency of generated wave is equal to $F_{oscillator}/(2N*Top)$. In modes 1, 2, and 3, the top value is fixed. Therefore, in these modes the only way to change the frequency of generated wave is to change N. Now you can see the different frequencies that can be generated using modes 1, 2, and 3.

Prescaler	1	1:8	1:64	1:256	1:1024
Mode = 1	$\frac{F_{oscillator}}{510}$	$\frac{F_{oscillator}}{8*510}$	$\frac{F_{oscillator}}{64*510}$	$\frac{F_{oscillator}}{256*510}$	$\frac{F_{oscillator}}{1024*510}$
Mode = 2	$\frac{F_{oscillator}}{1*1022}$	$\frac{F_{oscillator}}{8*1022}$	$\frac{F_{oscillator}}{64*1022}$	$\frac{F_{oscillator}}{256*1022}$	$\frac{F_{oscillator}}{1024*1022}$
Mode = 3	$\frac{F_{oscillator}}{1*2046}$	$\frac{F_{oscillator}}{8*2046}$	$\frac{F_{oscillator}}{64*2046}$	$\frac{F_{oscillator}}{256*2046}$	$\frac{F_{oscillator}}{1024*2046}$

Figure 16-47. Different Frequencies Can Be Made Using Modes 1, 2, and 3

SECTION 16.3: PWM MODES IN TIMER1

So, in these modes we can make a very limited number of frequencies. In modes 10, 11, the Top value can be specified by ICR1 and the OCR1A registers. Thus we can change the frequency by loading proper values to ICR1 and OCR1A.

Example 16-38

Assuming XTAL = 8 MHz, find TCCR1A and TCCR1B to generate two waves with frequency of 125 Hz on OC1A and 0C1B using mode 10, non-inverted mode, and prescaler = 1:256.

Solution:

$125 = 8M/(2N \times Top)$. $2N \times Top = 8M/125 = 64{,}000 \Rightarrow Top = 64{,}000 / 512 = 250$

$Top = 250 \Rightarrow ICR1 = 250$

$N=256 \Rightarrow CS12{:}0=100$

$Mode = 10 \Rightarrow WGMI2{:}10=1010$

OC1A in non-inverted mode $\Rightarrow$ COM1A1:COM1A0=10

OC1B in non-inverted mode $\Rightarrow$ COM1B1:COM1B0=10

TCCR1A =

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

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

Example 16-39

Calculate the OCR1x to generate the following waves in each of the following modes:

(a) Mode 11, inverted mode, OCR1A=50, duty cycle = 30%

(b) Mode 10, non-inverted mode, ICR1 = 250, duty cycle = 30%

(c) Mode 10, non-inverted mode, ICR1 = 250, duty cycle = 60%

Solution:

(a) In mode 11, Top=OCR1A=50. So,

$30 = (\text{Top OCR1B}) \times 100/\text{Top} \Rightarrow 50\text{-OCR1B} = 50 \times 30/100 = 15 \Rightarrow \boxed{\text{OCR1B} = 35}$

(b) In mode 10, Top = ICR1 = 250. So,

$30 = \text{OCR1x} \times 100/\text{Top} \Rightarrow \text{OCR1x} = 30 \times 250/100 = 75 \Rightarrow \boxed{\text{OCR1x} = 75}$

(c) In mode 10, Top = ICR1 = 250. So,

$60 = \text{OCR1x} \times 100/\text{Top} \Rightarrow \text{OCR1x} = 60 \times 250/100 \Rightarrow \boxed{\text{OCR1x} = 150}$

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

Example 16-40

Assume XTAL = 8 MHz. Using mode 10 write a program that generates waves with duty cycles of 30% and 60% on the OC1A and 0C1B pins, respectively. Frequency of the generated waves should be 125 Hz.

Solution:

```
.INCLUDE "M32DEF.INC"
        LDI     R16,LOW,(1RAMEND)
        OUT     SPL,R16
        LDI     R16,HIGH(RAMEND)
        OUT     SPH,R16                 ;initialize stack pointer
        SBI     DDRD,5                  ;PD5 (OC1A) = output
        SBI     DDRD,4                  ;PD4 (OC1B) output
        LDI     R16,0
        OUT     OCR1AH,R16              ;Temp = 0
        LDI     R16,75                  ;from Example 16-39
        OUT     OCR1AL,R16              ;OCR1AL = 75, OCR1AH = Temp = 0
        LDI     R16,150                 ;from Example 16-39
        OUT     OCR1BL,R16              ;OCR1BL = 150, OCR1BH = Temp = 0
        LDI     R16,250
        OUT     ICR1L,R16               ;ICR1L = 250, ICR1H = Temp = 0
        LDI     R16,0xA2                ;from Example 16-38
        OUT     TCCR1A,R16              ;COM1A = non-inverted, COM1B = non-inv
        LDI     R16,0x14                ;froM Example 16-38
        OUT     TCCR1B,R16              ;WGM mode 10, clock = no scaler
HERE:   RJMP    HERE                    ;wait here forever
```

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.3: PWM MODES IN TIMER1

16-bit PWM programming in C

Example 16-41 (C version of Example 16-25)

Assuming XTAL = 8 MHz, using non-inverted mode and mode 5, write a program that generates a wave with frequency of 31,250 Hz and duty cycle of 75%.

Solution:

```
"include "avr/io.h"
int main()
{
        DDRD |= (1<<5);          //PD5 = output
        OCR1AH = 0;              //Temp = 0
        OCR1AL = 191;            //OCR1A = 191
        TCCR1A = 0x81;           //COM1A = non-inverted
        TCCR1B = 0x09;           //WGM = mode 5, clock = no scaler
        while (1);
        return 0;
}
```

16-bit PWM programming in C

Example 16-42 (C version of Example 16-26)

Assuming XTAL=8 MHz, using non-inverted mode and mode 7, write a program that generates a wave with frequency of 7812.5 Hz and duty cycle of 75%.

Solution:

```c
#include "avr/io.h"
int main()
{
        DDRD |= (1<<5);            //PD5 = output
        OCR1AH = 767>>8;           //OCR1AH = HIGH (767)
        OCR1AL = 767;              //OCR1AL = LOW (767)
        TCCR1A = 0x83;             //COM1A = non-inverted
        TCCR1B = 0x09;             //WGM = mode 7, clock = no scaler
        while (1);
        return 0;
}
```

SECTION 16.4: DC MOTOR CONTROL USING PWM

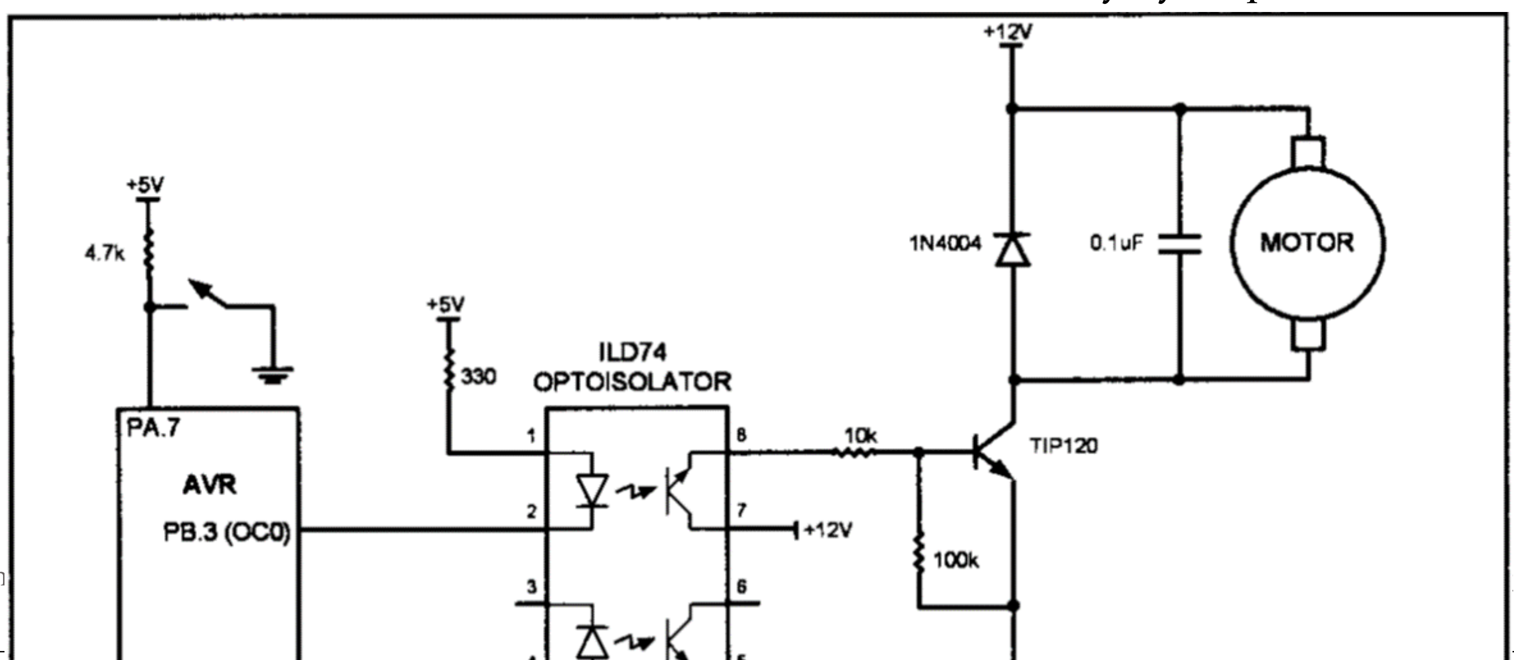
To generate the PWM waves for controlling the DC motor we can use the PWM feature of AVR.

Example 16-50 (Example 16-3 using AVR PWM features)

Refer to the figure in this example. Write a program to monitor the status of the switch and perform the following:

(a) If PORTA.7 = 1, the DC motor moves with 25% duty cycle pulse.

(b) If PORTA.7 = 0, the DC motor moves with 50% duty cycle pulse.



PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.4: DC MOTOR CONTROL USING PWM

Solution:

For driving motors it is preferable to use the Phase correct PWM mode.

$OCR0/255 = \text{duty cycle}/100 \Rightarrow OCR0 = 255 \times \text{duty cycle}/100$

For duty cycle = 25% $\Rightarrow OCR0 = 255 \times 25/100 = 64$

For duty cycle = 50% $\Rightarrow OCR0 = 255 \times 50/100 = 127$

In this example we generate waves with frequency of 245 Hz. To do so, $245 = 8M/(510 \times N) \Rightarrow N = 8M/(245 \times 510) = 64$

$\Rightarrow$ Prescaler = 64

TCCR0 =

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

PWM PROGRAMMING AND DC MOTOR CONTROL IN AVR
SECTION 16.4: DC MOTOR CONTROL USING PWM

```
.INCLUDE "M32DEF.INC"
        SBI     DDRB,3          ;make PB3 output
        SBI     PORTA,7         ;activate pull-up of PA7
        LDI     R16,0x73
        OUT     TCCR0,R16       ;N = 64, Phase correct PWM, inverted
L1:     SBIC    PINA,7          ;skip next instruct if PINA.7 is zero
        LDI     R16,64          ;if PINA.7 is one then R16 = 64
        SBIS    PINA,7          ;skip next instruct if PINA.7 is one
        LDI     R16,127         ;if PINA.7 is zero then R16 = 127
        OUT     OCR0,R16        ;OCR0 = R16
        RJMP    L1              ;jump L1
```

SECTION 16.4: DC MOTOR CONTROL USING PWM

Example 16-51

Write a program that gradually changes the speed of a DC motor from 50% to 100%. Use information given in Example 16-50.

Solution:

```
.INCLUDE "M32DEF.INC"
        LDI     R16,HIGH(RAMEND)
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16                 ;initialize stack pointer
        SBI     DDRB,3                  ;make PB3 output
        LDI     R16,0x73                ;from Example 16-50
        OUT     TCCR0,R1                ;N = 64, Phase correct PWM, inverted
        LDI     R20,127
L1:     OUT     OCR0,R20                ;OCR0 = R17
        RCALL   DELAY
        INC     R20                     ;increment R20
        BRNE    L1                      ;jump L1 if R20 is not zero
HERE:   RJMP    HERE
```

SECTION 16.4: DC MOTOR CONTROL USING PWM

DC motor control and PWM using C

Example 16-52 (C version of Example 16-50)

Write a program to monitor the status of the switch and perform the following:

(a) If PORTA.7 = 1, the DC motor moves with 25% duty cycle pulse.

(b) If PORTA.7 = 0, the DC motor moves with 50% duty cycle pulse.

Solution:

```c
 #include "avr/io.h"
int main()
{
        DDRB = 0x08;                    //PB3 as output
        PORTA = 0x80;                   //pull-up resistor
        TCCR0 = 0x73;                   //Phase correct PWM, inverted, N = 64
        while (1)
        {
            switch ((PINA&0x80))
            {
                case 0: OCR0 = 64; break;       //25%
                case 1: OCR0 = 127; break;      //50%
            }
        }
        return 0;
}
```

SECTION 16.4: DC MOTOR CONTROL USING PWM

Phase correct PWM Mode

Example 16-53 (C version of Example 16-51)

Write a program that gradually changes the speed of a DC motor from 50% to 100%.

Solution:

```
#define     F   CPU     8000000UL        //XTAL = 8 MHz
#include
#include "util/delay.h"
int main()
{
        unsigned char i;
        DDRB = 0x08;                     //PB3 as output
        i = 127;                         //duty cycle = 50%
        OCR0 = 127;              //Phase correct PWM, inverted, N = 64
        TCCR0 = 0x73;
        while (i != 0)
        {
            OCR0 = i;
            delay_ms(25);        //use AVR Studio library delay
            i++;
        }
        while (1);
        return 0;
}
```