مقدمه:

امروزه در صنعت و در بسیاری از وسایل خانگی کنترل دور موتور مورد استفاده میگرد. از جمله مـی تـوان به کاربردهای کنترل گرهای دور موتور، به موارد زیر اشاره کرد:

1) وسایل خانگی:

کنترل گر های دور موتور در وسایل شخصی خانگی، در کار برد های کوچک و بزرگ مورد اسـتفاده قـرار میگیرند. به عنوان مثال، پنکه های دیواری یا پنکه تهویه حمام که توسط کلیدی کنترل می شوند.

2) در وسایل اداری و درمانی:

در این دسته دستگاه های بسیاری را می توان مثال زد. مداد تراش های برقی در ادارات، دستگاه های فکس، کامپیوتر ها یا دستگاه های کپی و.. . سیستم کاری این کنترل گر ها بسیار پیچیـده بـوده و حتـی در مورد وسایل درمانی پیچیده تر نیز میشود. مثلاً کنترل دور موتور داخـل هـارد دیسـک کـامپیوتر را در نظـر بگیرید.

3) در کار برد های تجاری:

ساختمان های تجاری دارای سیستم تهویه بزرگتر و مجهز تری نسبت به موارد مشـابه در منـازل شخصـی دارند. همچنین می‌توان در این دسته موتور ها برای آسانسور ها، پله های برقی و موارد مشابه را نام برد.

4) کار برد های صنعتی:

بسیاری از صنایع وابسته به موتور ها و کنترل دور موتور آن ها می باشـند. موتـور هـای کوچـک DC تـا موتور ای بزرگ صنعتی، یا موتور های استفاده شده در خطوط مترو. همچنین در صنعت ممکن است یک کنترل گر عمل کنترل بیش از یک موتور را به طور همزمان بر عهده داشته باشد.

5) در وسایل نقلیه:

تمام وسایل نقلیه از جمله، خودرو ها، هواپیما ها، دستگاه آلات کشاورزی، همه و همه ممکن اسـت دارای موتور برای انجام کار های گوناگونی باشند.

6) ابزار قدرت:

وسایل قدرتی همانند دریل ها، اره ها، چرخ سمباده ها که توسط کاربر خـانگی اسـتفاده مـی شـوند. تمـام وسایل قدرتی قایل حمل یا ثابت دارای معمولاً همراه با کنترل گر های سرعت این موتور ها نیز می باشند.

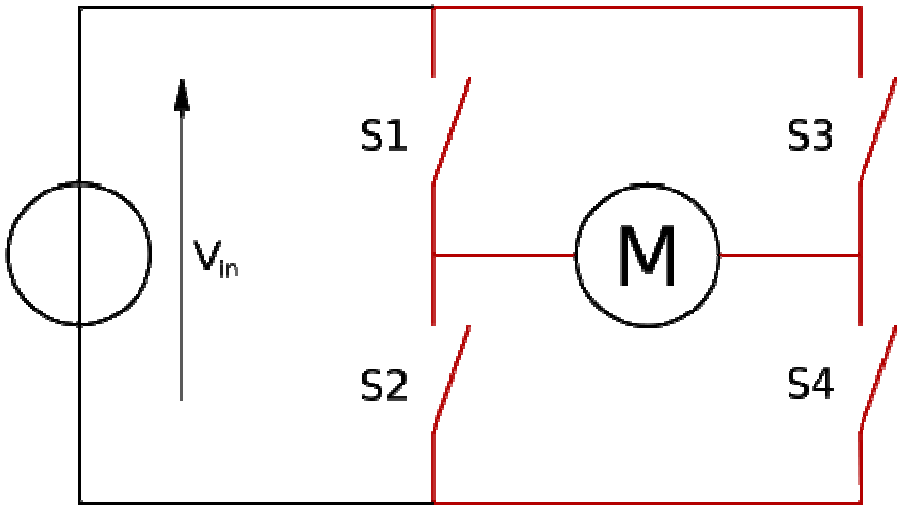
انواع کنترل گرها:

کنترل گر های موجود را می توان بر اساس موتوری که باید کنترل کنند، دسته بندی کرد. SERVO موتـور ها، مغناطیسی ثابت، سری، تحریک مجزا و جریان متناوب و مستقیم.

توضیح برخی از کنترل گرهای شناخته شده:

1) H-Bridge:

موتور های Dc معمولاً توسط ترانزیستور ها با مداری مشهور به مدار H-Bridge کنترل میشوند. این روش شامل حداقل 4 سویچ مکانیکی یا الکترونیکی (ساخته شده از نیمه هادی ها) می باشد.



در مدار فوق که نمونه ای از مدار H-Bridge است، با بستن کلید های S2 و S3 ولتاژ V_{in} بطور معکوس بر روی موتور قرار میگرید که موجب گردش موتور در خلاف جهت اولیه (زمـانی کـه کلیـد هـای S1 و S4 وصل بودند) می شود. این مدار توسط نیمه هادی با استفاده از دو عنصر با پلاریته های متفاوت ساخته می شوند. برای مثال: با ترانزیستور هـای BJT، PNP یـا MOSFET، P کانـال متصـل بـه سـطح ولتـاژ بـالا و ترانزیستور های BJT، NPN یا MOSFET، N کانال.

2) کنترل گرهای Servo:

بسیاری از موتور های SERVO توسط روشی مشهور به PWM کنترل میشوند که در ادامـه در ایـن مـورد بیشتر بحث خواهیم کرد.

3) کنترل گر های STEP:

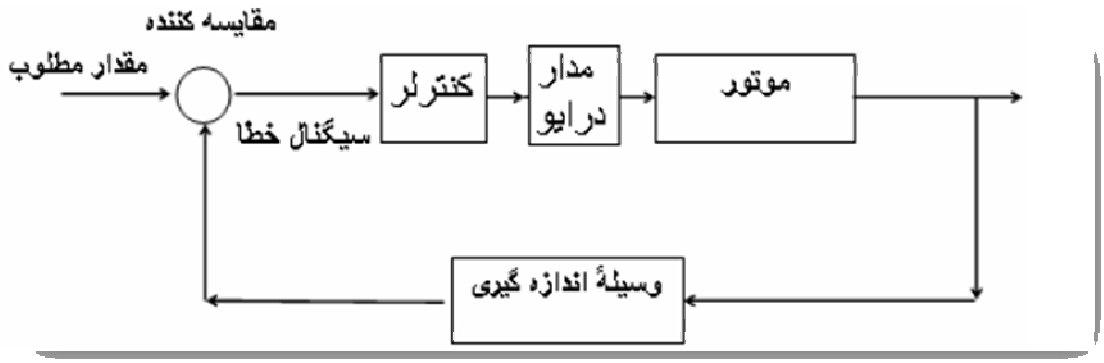
این موتور های توسط مدارات زمان بندی شده کنترل میشوند. این امر موجب میشود تا کنتـرل سـرعت و چرخش موتور با دقت زیادی صورت گیرد. از این کنترل گر ها در چاپگر ها استفاده میشوند.

از میان روش های گفته شده، در این مقاله کنترل دور موتور با استفاده از PWM توضیح داده خواهد شد.

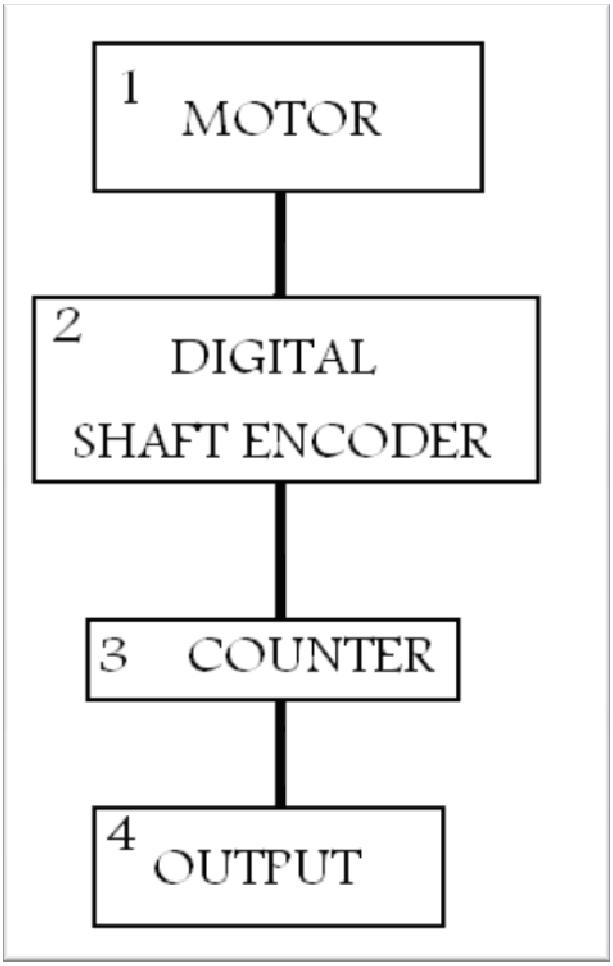
استفاده از PWM برای کنترل دور موتور:

ایده کلی استفاده از PWM برای کنترل دور موتور DC بسیار ساده است. اما ابتدا توضیحی در مورد PWM و روش های گوناگون تولید آن خواهیم داد.

در حالت کلی برای بسیاری از موارد گفته شده و یا راه کار های مشابه می توان بلوک دیاگرام کنترلی زیر را در نظر گرفت:



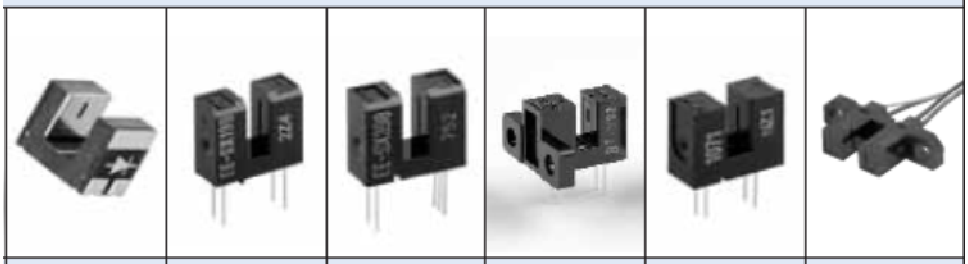
در این مقاله از میکروپروسسورها به عنوان کنترلر و از IC هایی که در ادامه تشریح خواهد بزای مدار درایور استفاده خواهد شد. قبل از اینکه وارد بحث کنترل دور موتور شویم، باید دور موتور را در ولتاژ های معین شده ای در معین کرده باشیم. در این پروژه یک موتور DC 12 ولت را بررسی و کنترل خواهیم کرد. برای بدست آوردن دور موتور در ولتاژ های مختلف یکی از روش های ساده استفاده از یک Shaft Encoder دیجیتال و شمردن تعداد پالس های این Shaft Encoder در مدت یک ثانیه یا یک دقیقه است. برای اینکه زمان و دقت بیشتری داشته باشیم، در مدت زمان یک ثانیه تعداد پالس های ورودی را خواهیم شمرد. از آنجایی که در ادامه پروژه نیز از میکروکنترلر AVR ATMEGA 16 استفاده خواهیم، پس در این قسمت از شمارنده داخلی همین IC برای شمارش تعداد پالس های Shaft Encoder استفاده خواهیم کرد.
پس تا کنون اشاره شد که باید با شمارش تعداد پالس های حاصل از Shaft Encoder، دور مورتور را در ولتاژ های معینی بدست آوریم. اصولاً به چنین سیستم دور سنج نیز گفته می شود.

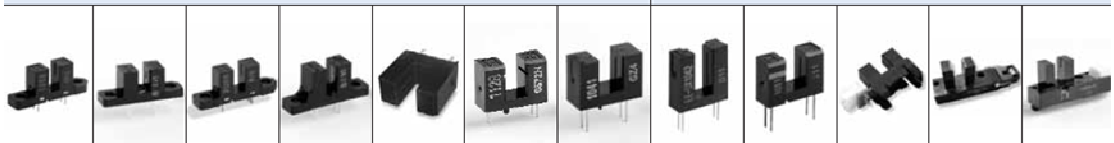


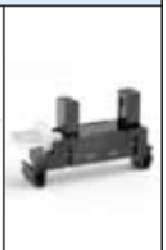
در ادامه به توضیح هر یک از بلوک های شکل فوق خواهیم پرداخت.

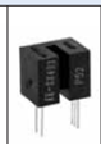
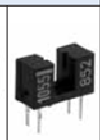
ـ) **بلوک شماره 1:** که مروبوط به موتور DC 12 ولت مـی باشـد، ورودی هـای موتـور بـه ولتـاژ متغیـر (Adaptor) متصل هستند تا بتوانیم با تغییر ولتاژ ورودی، دور موتور خروجـی را معـین کنـیم. بـرای وارد نمودن مقدار مطلوب، کلید هایی در نظر گرفته شده اند تا کاربر با استفاده از این کلیـدها دور موتـور را بـر روی مقدار مورد نظر تنظیم کند. فرض کنیم بطور ساده یک موتور 12 ولت سـاده را مـدل خـواهیم کـرد. پس حداکثر ولتاژی که بر روی موتور می توان قرار داد، برابر 12 ولت در نظر میگیریم.

ـ) **بلوک شماره 2:** مربوط به Digital Shaft Encoder می باشد. می توان این انکودر را بطـور حاضـری خریداری کرد و یا با هزینه ای کمتر، آن را سـاخت. در آیـن قسـمت بـه توضـیح نحـوه سـاخت انکـودر مکانیکی خواهیم پرداخت یعنی سیستم داخلی انکودر الکترومکانیکی است.
سیستم داخلی این نوع انکورد متشکل از یک سنسور نوری می باشد که مـوج (سـیگنال نـور بـا فرکـانس معین) از فرستنده این سنسور صادر شده و در طرف گیرنده دریافت می شود. این سنسور انواع گونـاگونی دارد. چند نمونه از انواع آن در شکل های زیر نشان داده شده اند.

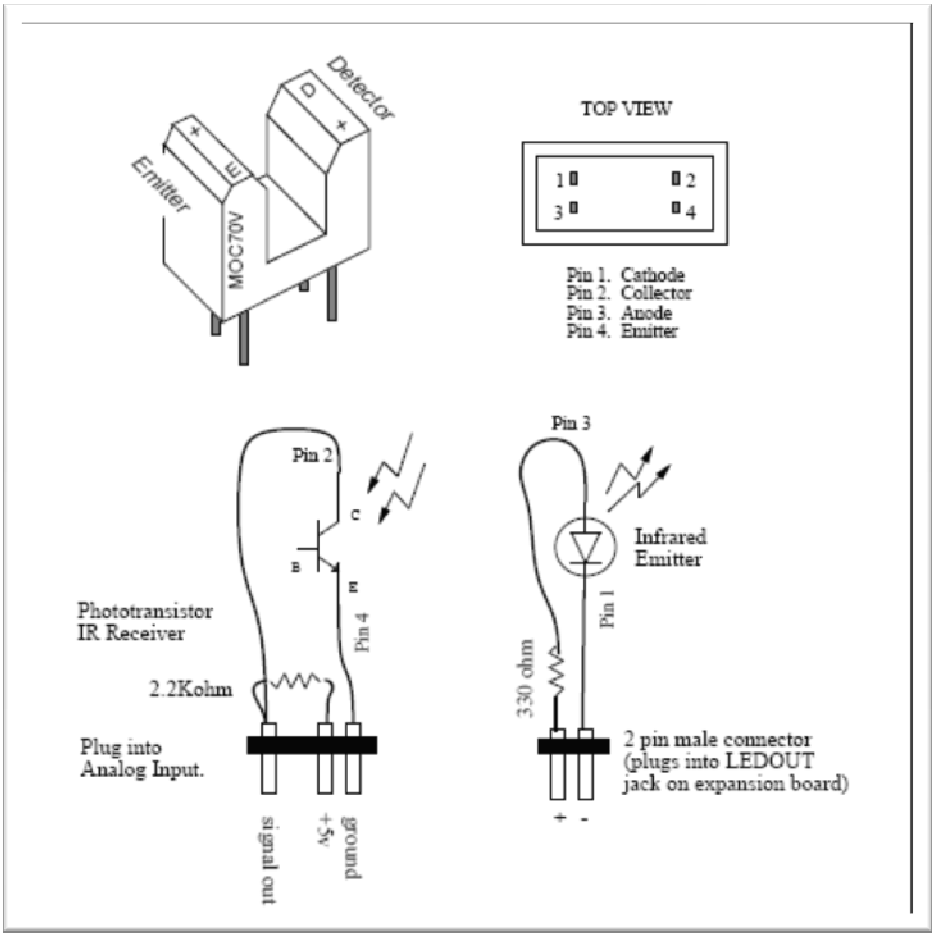
EE-SX1109	EE-SX199	EE-SX398/498	EE-SV3	EE-SX1071	EE-SX1096
Transmissive slot width 3mm - < 5mm					
					
6 x 4 x 5	12.2 x 5 x 10	12.2 x 5 x 10	19 x 15.1 x 10.2	13.6 x 6.2 x 10.2	25 x 6 x 10
Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive
3mm	3mm	3mm	3.4mm	3.4mm	3.4mm
0.5mm	0.5mm	0.5mm	0.2/0.5/1.0mm	0.5mm	0.5mm
940nm	940nm	940nm	940nm	940nm	940nm
Surface Mount	Through-hole	Through-hole	Through-hole	Through-hole	Lead Wires

E-SX1088	EE-SH3	EE-SX3088/4088	EE-SG3/SG3B	EE-SX1057	EE-SX1128	EE-SX1041	EE-SX1042	EE-SX1081	EE-SX1235A-P2	EE-SX3009-P1 /4009-P1	EE-SX4019-P
Transmissive slot width 3mm - < 5mm							Transmissive slot width 5				
											
25 x 6 x 10	25.4 x 6.2 x 10.2	25 x 6 x 10	25.4 x 6.3 x 11.5	13 x 6.3 x 8.6	13.5 x 5.2 x 9.3	14 x 6 x 10	14 x 5 x 14.5	13.7 x 5 x 10	27 x 8 x 15.9	34 x 11 x 21	38 x 11 x 21
Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive
3.4mm	3.4mm	3.4mm	3.6mm	3.6mm	4.2mm	5mm	5mm	5mm	5mm	5mm	5mm
0.5mm	0.2/0.5/1.0mm	0.5mm	2.0mm	2.0mm	0.5mm	0.5mm	0.5mm	0.5mm	0.5mm	0.5mm	0.5mm
940nm	940nm	940nm	940nm	940nm	940nm	940nm	940nm	940nm	940nm	940nm	940nm
Through-hole	Through-hole	Through-hole	Through-hole	Through-hole	Through-hole	Through-hole	Through-hole	Through-hole	Snap-In	Screw Mounting	Screw Mountin

EE-SX3081 /4081	EE-SX4235A-P2	EE-SX1070	EE-SX3070 /4070	EE-SX1140	EE-SX461-P11
m - 8mm				Transmissive slot width over 12mm	
					
13.7 x 5 x 10	27 x 8 x 15.9	17.7 x 6 x 10	17.7 x 6 x 10	23 x 5 x 16.3	32.5 x 12 x 23.6
Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive
5mm	5mm	8mm	8mm	14mm	15mm
0.5mm	0.5mm	0.5mm	0.5mm	1.5mm	2.0mm
940nm	940nm	940nm	940nm	940nm	940nm
Through-hole	Snap-In	Through-hole	Through-hole	Through-hole	Snap-In

E-SX1107	EE-SX1018	EE-SX1103	EE-SX1105	EE-SX1109	EE-SX1131	EE-SX4134	EE-SX493	EE-SX1055	EE-SX1046	EE-SX1082	EE-SX1106
Transmissive slot width up to 3mm											
											
3.4 x 3 x 3	8 x 4 x 6	5 x 4.2 x 5.2	4.9 x 2.6 x 3.3	5 x 4 x 4	5 x 4 x 4	5 x 4 x 4	11 x 8 x 9.5	8.9 x 4 x 5.4	10 x 6.5 X 5	10 x 6.5 x 5.2	6.4 x 4.2 x 5.4
Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive	Transmissive
1mm	2mm	2mm	2mm	2mm	2mm	2mm	2mm	2.8mm	3mm	3mm	3mm
0.15mm	0.5mm	0.4mm	0.4mm	0.3mm	0.3mm	0.3mm	0.2mm	0.5mm	0.5mm	0.2mm	0.4mm
940nm	940nm	950nm	950nm	940nm	940nm	940nm	940nm	940nm	920nm	920nm	950nm
urface Mount	Through-hole	Through-hole	Through-hole	Surface Mount	Surface Mount	Surface Mount	Through-hole	Through-hole	Through-hole	Through-hole	Through-hole

مدار داخلی این سنسورها تقریباً یکسان و بفرم زیر است:



همان طور که مشاهده می شود، سنسور ها متشکل از دو قسمت فرستنده و گیرنده هستند. در صورتی که مانعی در بین فرستنده و گیرنده موجود نباشد، سیگنال فرستاده شده توسط گیرنده دریافت شده و فوتو ترانزیستور مربوطه شروع به هدایت می کند. در این صورت ولتاژ 5 ولت ورودی که به پایه این سنسور وصل شده است، در خروجی ظاهر خواهد شد. سنسوری که در این پروژه استفاده خواهیم کرد، E907-E2-01 خواهد بود.

در صورتی که به برگه اطلاعاتی این سنسور مراجعه کنید، اطلاعات بیشتری در مورد جزئیات این IC خواهید یافت. در اینجا به بخشی از این جزئیات شاره می شود:

■ **Electrical and Optical Characteristics (Ta = 25°C)**

Item		Symbol	Value	Condition
Emitter	Forward voltage	V_F	1.2 V typ., 1.5 V max.	I_F = 30 mA
	Reverse current	I_R	0.01 µA typ., 10 µA max.	V_R = 4 V
	Peak emission wavelength	λ_P	940 nm typ.	I_F = 20 mA
Detector	Light current	I_L	0.5 mA min., 14 mA max.	I_F = 20 mA, V_{CE} = 10 V
	Dark current	I_D	2 nA typ., 200 nA max.	V_{CE} = 10 V, 0 lx
	Leakage current	I_{LEAK}	---	---
	Collector–Emitter saturated voltage	V_{CE} (sat)	0.1 V typ., 0.4 V max.	I_F = 20 mA, I_L = 0.1 mA
	Peak spectral sensitivity wavelength	λ_P	850 nm typ.	V_{CE} = 10 V
Rising time		tr	4 µs typ.	V_{CC} = 5 V, R_L = 100 Ω, I_L = 5 mA
Falling time		tf	4 µs typ.	V_{CC} = 5 V, R_L = 100 Ω, I_L = 5 mA

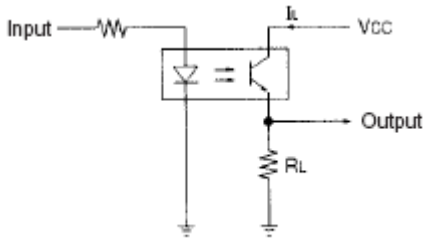
■ Absolute Maximum Ratings (Ta = 25°C)

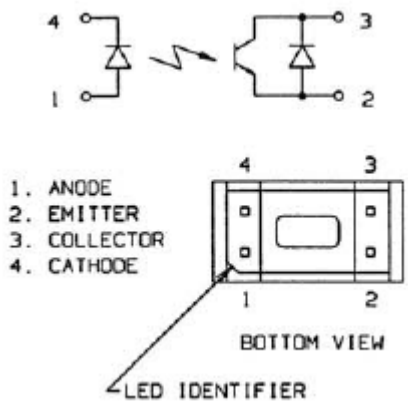
Item		Symbol	Rated value
Emitter	Forward current	I_F	50 mA (see note 1)
	Pulse forward current	I_{FP}	1 A (see note 2)
	Reverse voltage	V_R	4 V
Detector	Collector–Emitter voltage	V_{CEO}	30 V
	Emitter–Collector voltage	V_{ECO}	---
	Collector current	I_C	20 mA
	Collector dissipation	P_C	100 mW (see note 1)
Ambient temperature	Operating	Topr	–25°C to 85°C
	Storage	Tstg	–30°C to 100°C
Soldering temperature		Tsol	260°C (see note 3)

Note: 1. Refer to the temperature rating chart if the ambient temperature exceeds 25°C.
2. The pulse width is 10 μs maximum with a frequency of 100 Hz.
3. Complete soldering within 10 seconds.

آنچه که باید توجه کنید، مقادیر ماکزیمم این IC در دو طرف فرستنده (Emitter) و گیرنـــده (Detector) می باشد.

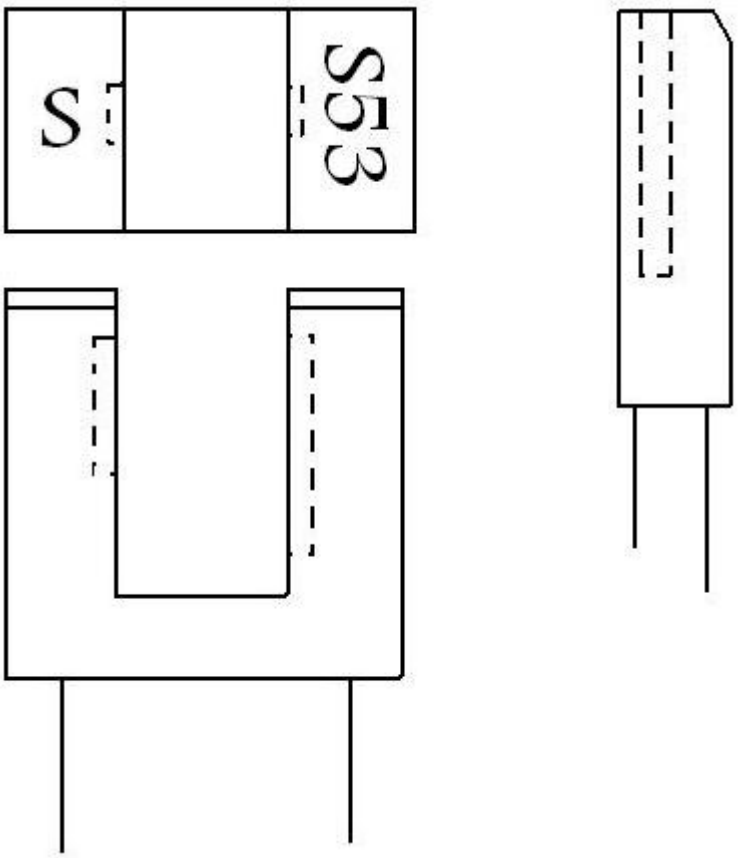
مدار معادلی که می توان برای IC در نظر گرفت طبق شکل زیر است:





با توجه به اطلاعات داده شده در قسمت های بالا، مقدار جریان حداکثر در سمت فرستنده (Emitter) 20mA می باشد. در نتیجه باید جریان گذرنده از فرستنده محدود گردد که برای این کار مطابق مدار معادل، مقاومتی در مسیر فرستنده باید قرار گیرد.

در این پروژه از نوع مشابهی از همین IC ها استفاده شده است. در شکل زیر دیأهای مختلف IC نشان داده شده اند:



این IC هم در قسمت فرستنده و هم در قسمت گیرنده دارای دو پایه بلند و کوتاه میباشد. جدول زیر نحوه اتصالات مربوطه را مشابه ا توضیح می دهد.

S	پایه بلند وصل به GND
	پایه کوتاه از طریق مقاومت 330Ω وصل به Vcc
S53	پایه بلند وصل به GND
	پایه کوتاه از طریق مقاومت 10KΩ وصل به Vcc
خروجی از انتهای مقاومت 10KΩ متصل به IC می باشد.	

حال در صورتی که مانعی بین فرستندهو گکیرنده قرار گیرد ولتاژ خروجی برابر 5 ولت (سطح منطق 1) و در صورتی که مانعی بین فرستنده و گیرنده موجود نباشد، ولتاژ خروجی برابر صفر ولت (منطق صفر) خواهد بود. حال کافی است تا با اتصال یک چرخ دنده یا شیئ مشابه که با چرخش موتور در ارتباط بین گیرنده و فرستنده؛ قطع و وصل ایجاد کند، سیگنال Digital مورد نظر را بدست آورد.

برای اینکه مشکل Slow rate نیز را حل کرده باشیم از یک Inverter استفاده خواهیم کرد. یعنی خروجی این OptoSwitch را بهیک Inverter وصل خواهیم کرد.

IC مورد استفاده در این پروژه، 4116 می باشد. اطلاعات بیشتر در مورد این IC را می توانید از سایت

www.Alldatasheet.com

کسب نمائید. تنها توضیح مختصر اینکه این IC یک معکوس گر با تکنولوژی CMOS می باشد.

از آنجایی که لبه های این پالس ایجاد شده توسط OptoSwitch برای ما مهم است _ زیرا توسط میکرو تعداد این لبه های بالا رونده یا پائیین رونده را خواهیم شمرد، یا به عبارتی دیگر از تنظیمات میکرو، آن را به لبه ها حساس خواهیم کرد _ معکوس کردن خروجی OptoSwitch تغییری در نتیجه ایجاد نخواهد کرد. تنها موجب خواهد شد تا ضریب کیفی مدارمان افزایش یابد.

نکته دیگر که باید به آن توجه کنید اینکه طبق آنچه که در برگه اطلاعاتی OptoSwitch نوشته شده است، حداکثر فرکانس کاری این IC (OptoSwitch) برابر 100Hz می باشد. یعنی در هر ثانیه حداکثر 100 بار میتوانید بین فرستنده و گیرنده تولید کنید!! توضیح بیشتر اینکه فرض کنید به موتور DC، یک پروانه متصل کرده اید که دارای دو پره می باشد که در نتیجه در هر دور موتور، دو پالس ایجاد خواهد شد. یعنی در این صورت حداکثر دور موتور مجاز موتور شما می تواند 3000rpm باشد. اما اگر پروانه تنها یه پره داشت، در این صورت در هر دور موتور، یک پالس ایجاد میشد و در نتیجه حداکثر دور موتور مجازی که می توانستید استفاده کنید 6000rpm می بود.

3) بلوک Counter:

در این قسمت از یک میکرو کنترلر AVR، مدل ATMEGA16 برای شمردن تعداد پالس های بلوک قبلی استفاده خواهیم کرد. این میکرو کنترلر دارای 3 تایمر داخلی می باشد که برای استفاده از آنها کافی است تا در برنامه CODEVISION با استفاده از Wizard به سادگی این Timer ها فعال کنیم.

درادامه نیز از این تایمر ها برای تولید PWM استفاده خواهد شد که در آنجا به بحث کامل تری در مورد انواع حالت های تایمر های این میکرو کنترلر خواهیم پرداخت. برنامه ای که برای شمارش پالس های Digital Shaft Encoder استفاده شده است، عبارت است از:

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.8d Standard
Automatic Program Generator
© Copyright 1998-2006 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : DC Motor Control
Version : 3.1
Date    : 2007/02/16
Author  : Yashar
Company : YBSH
Comments:


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 1.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x12 ;PORTD
#endasm
#include <lcd.h>
#include <stdlib.h>


int n=0;

char *s;

// Timer 2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
// Place your code here
```

```
int i=0;
n++;
if(n>=15)
{
TCCR0=0x00;
lcd_clear();
i=TCNT0;
lcd_putsf("Hi Yashar");
itoa(i,s);
lcd_gotoxy(0,1);
lcd_puts(s);
n=0;
TCNT0=0x00;
TCCR0=0x17;
}

}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTA=0x00;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=0 State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x08;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0xFF;

// Timer/Counter 0 initialization
// Clock source: T0 pin Rising Edge
// Mode: Normal top=FFh
// OC0 output: Disconnected
```

```
TCCR0=0x07;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 3.906 kHz
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x06;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x40;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);
```

```
// Global enable interrupts
#asm("sei")

while (1)
    {
    // Place your code here

    };
}
```

در این برنامه، از 2 تایمر این میکرو برای شمارش تعداد پالس ها استفاده شده است. یعنی یکی از تایمر ها، وظیفه ایجاد بازه های زمانی برابر 1 ثانیه را دارد و دیگری تعداد پالس های ورودی را می شمارد. به عبارتی دیگر، زمان نمونه برداری برابر 1 ثانیه می باشد که توسط یک تایمر ایجاد می شود. در هر ثانیه تعداد پالس ها شمرده شده و در LCD نمایش داده می شود. تایمر صفر با لبه بالارونده پایه T0، کلاک خورده و پالس های ورودی را میشمارد. این مقدار هر یک ثانیه خوانده شده (TCR0 مشخص کننده مقدار تایمر) و با ضرب در عدد 60 دور موتور در دقیقه بدست می آید. برای ایجاد یک ثانیه، متغیر Global مانند n تعریف میکنیم. میدانیم که فرکانس کاری AVR را بر روی 1 MHz تنظیم کرده ایم. (در ادامه نحوه این تنظیمات بطور کلی بیان خواهد شد.) فرکانس تایمر 2 نیز که برای ایجاد یک ثانیه از آن استفاده خواهد شد، برابر 3.906KHz در نظر گرفته شده است. در نتیجه تایمر 2 که 8 بیتی است، در مدت زمان

$$256 \times 256\mu sec = 65536\mu sec$$

یک دور کامل (یعنی تا عدد 256) خواهد شمرد و مجدداً از صفر آغاز خواهد کرد. قبل از شروع مجدد شمارش، سرویس وقفه را فعال کرده ایم. در سرویس وقفه، متغیر n را یک واحد افزایش می دهیم. بدیهی است برای اینکه مدت زمان 1 ثانیه را داشته باشیم باشد n برابر مقداری بین 15 تا 16 باشد. اما متغیر n، 8 بیتی و Integer تعریف شده است، پس امکان ذخیره اعشار در آن ممکن نیست. با قبول اندکی خطا؛ n را برابر با 15 در نظر میگیریم.

$$1sec = n \times 65536\mu sec \Longrightarrow n = \frac{1sec}{65536\mu sec} = 15.2578 \approx 15$$

نکته دیگر که باید دقت کنید تعداد پالس هایی است که از Encoder به ورودی میکرو می آید. همانطور که گفتیم هر بار که مانعی بین فرستنده و گیرنده IC قرار گیرد، یک پالس تولید خواهد شد. حال اگر در کل چرخ دنده ای را به موتور کوپل کنیم که دارای یک کمان خالی 90 درجه ای باشد یعنی در هر بار چرخش موتور تنها یک پالس تولید می شود، که Duty Cycle برابر 25% خواهد داشت. اما اگر از دو برش 90 درجه ای استفاده شود، در این صورت در هر دور چرخش موتور 2 پالس با Duty Cycle برابر

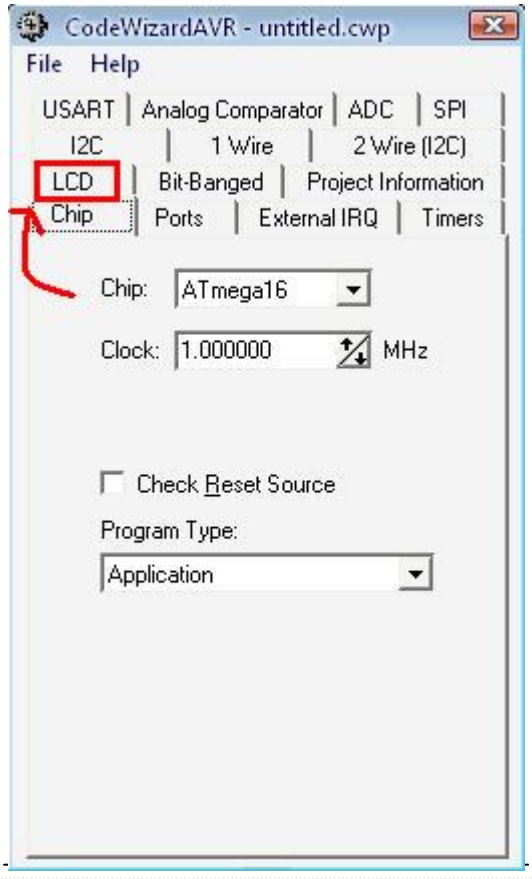
50% ایجاد می شود.. یعنی بسته به تعداد دندانه های چرخ دنده ای کـه در Shaft Encoder اسـتفاده مـی شود، تعداد پالس ها در هر دور تغییر می کند و در نتیجه باید در نهایت در میکرو کنترلر تعداد پالس هـای کل بدست آمده را بر تعداد برش های موجود روی چرخ دنده ی کوپل شده تقسـیم کنـیم تـا دور موتـور بدست آید.

4) بلوک Output:

در نهایت پس از انجام محاسبات مربوطه، عدد (دور مورتور) به ازای ولتاژ ورودی معین، در خروجـی بـر روی LCD نمایش داده می شود. بار دیگر یاد آور می شویم که برای نمایش نتایج بر روی LCD از طریـق AVR، در برنامه Code Vision تنظیمات مربوط به این قسمت را از طریق Wizard معین کنید. برای درک بهتر مطلب بطور خلاصه بیان شده است.

برنامه ریزی AVR ATMEGA 16 از طریق Code Vision، برای خروجـی LCD:

برای این کا کافی است تا پس از ایجاد یک پروژه جدید و استفاده از Wizard (که در ادامه نحوه انجام این کار در قسمت تولید PWM با جزئیات بیشتر بحث شده است) ایجـاد کنیـد. حـال در پنجـره Wizard بـه قسمت LCD بروید.



حال در این قسمت تنها کافی است پورت مورد نظر را که قصد دارید LCD بـه آن را وصـل کنیـد، معـین کنید.

یکی از راه های کنترل دور موتور همان طور که قبلاً نیز به آن اشاره شد استفاده از PWM می باشـد. طـرز کار با PWM برای کنترل دور موتور ساده است. با استفاده از آیسی L298، که توضیحات آن در ادامـه داده شده است ما سیگنال PWM را به موتور اعمال خواهیم کرد. مطمئناً آنچه که مد نظر ما است این است کـه در بازه های زمانی معین (که توسط Duty Cycle سیگنال PWM معین می شود،) موتور روشن و خاموش شود.
نتایج بدست آمده برای موتور DC، 12 ولت عبارت اند از:

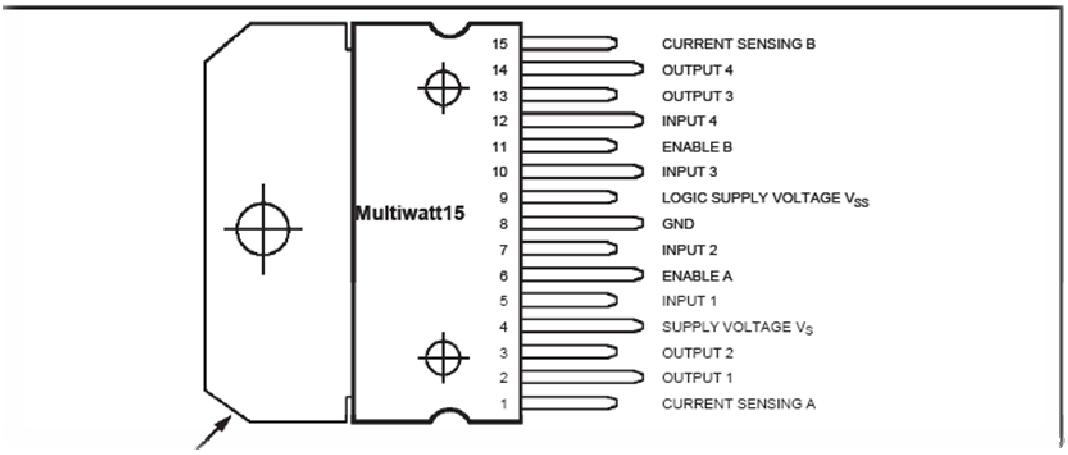
کد	ولتاژ	دور در ثانیه RPS	دور در دقیقه RPM
00	0v	0	0
01	4v	30	1800
10	6v	50	3000
11	8v	66	3960

DC Motor Driver:

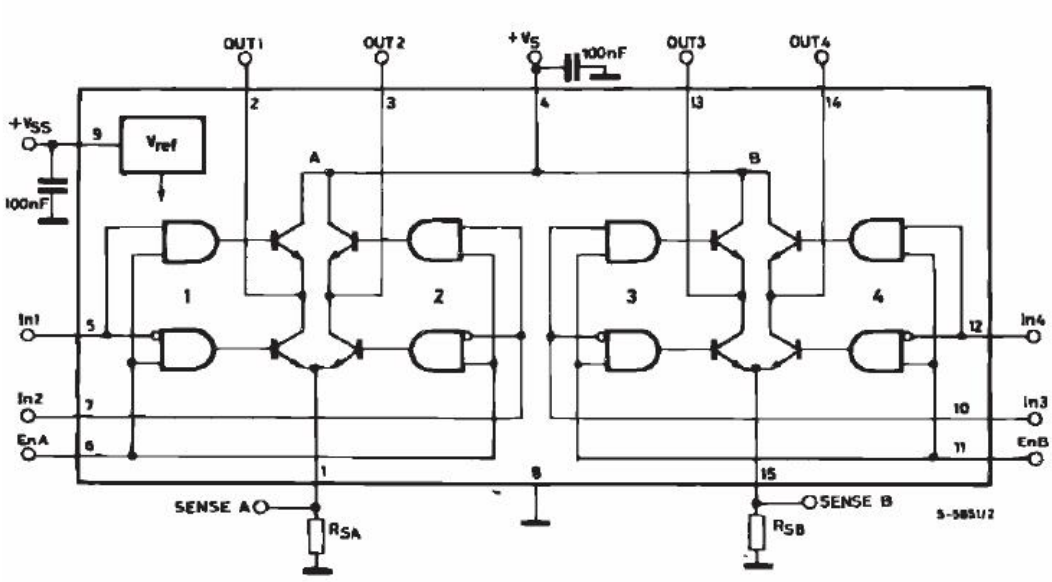
در این پروژه از L298 برای DRIVE موتور استفاده شده است. مشخصات این IC در ادامه نشان داده شـده است.

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

IN CONNECTIONS (top view)



این IC از دو پل H ساخته شده است. در نتیجه می توان با استفاده از این IC دو موتور را بطور همزمان کنترل نمود که ما تنها از یک خروجی آن جهت کنترل موتور خود استفاده خواهیم کرد. شماره پایه های ورودی و خروجی متناظر را از شکل زیر برای یک پل H بدست آورده و اتصالات مربوطه را برقرار خواهیم کرد. در ادامه مدار کامل مورد نظر ارائه شده است.

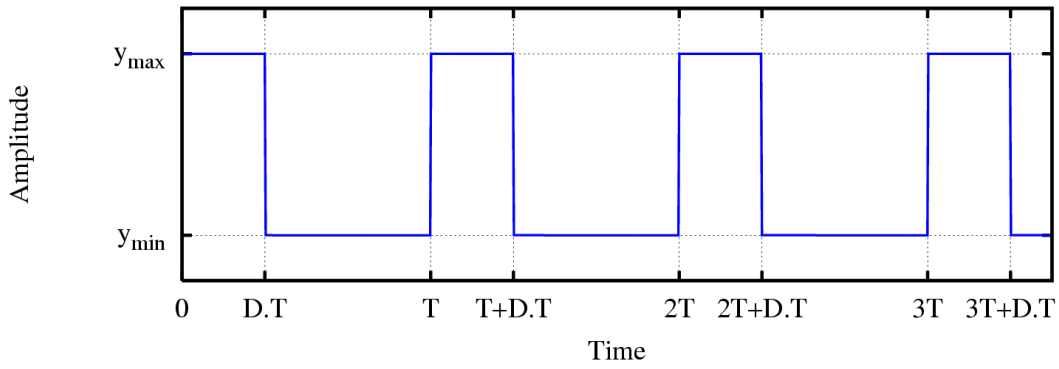


سیگنال PWM خروجی از میکرو را به پایه ENABLE این IC (L298) می دهیم. به این ترتیب این IC متناسب با پهنای پالس مدوله شده، فعال و غیر فعال شده و ولتاژ موتور قطع و وصل می شود و در نتیجه دور موتور بسته به Duty Cycle سیگنال PWM، کاهش یا افزایش می یابد.

:PWM

این کلمه مخفف Pulse Width Modulation به معنای مدولاسیون پهنای پالس می باشد. یا به عبارتی دیگر یعنی با تغییراتی در پهنای پالس، توان (قدرت) الکتریکی انتقالی به موتور را کاهش یا افزایش می دهیم. وقتی می گوییم موتور DC با ولتاژ DC دارای دور نامی مشخصی می باشد، یعنی اگر ولتاژی با مقدار معین را در سر موتور قرار دهیم، در این صورت قدرت انتقالی به موتور ثابت بوده و در نتیجه موتور با دور نامی خود، کار خواهد کرد. ولتاژ DC یعنی ولتاژ ثابت. میدانیم که یک ولتاژ ثابت همان پالس با پهنا پالس دلخواه است. یعنی به ازای این مقادیر زمانی ولتاژ دو سر بار مقداری مستقیم است. حال اگر ما به هر نحوی این ولتاژ مستقیم روی دو سر موتور را کاهش دهیم، نتیجتاً قدرت انتقالی به موتور و در نتیجه دور موتور کمتر خواهد بود. با ادامه بحث هر چه بیشتر متوجه منظورم خواهید شد.

اصل و مبنای PWM تغییر (مدوله کردن) پهنای پالس و در نتیجه تغییر مقدار متوسط ولتاژ موج است.در صورتی که یک موج مربعی را در نظر بگیریم، در این صورت خواهیم داشت:



می دانیم:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\,dt$$

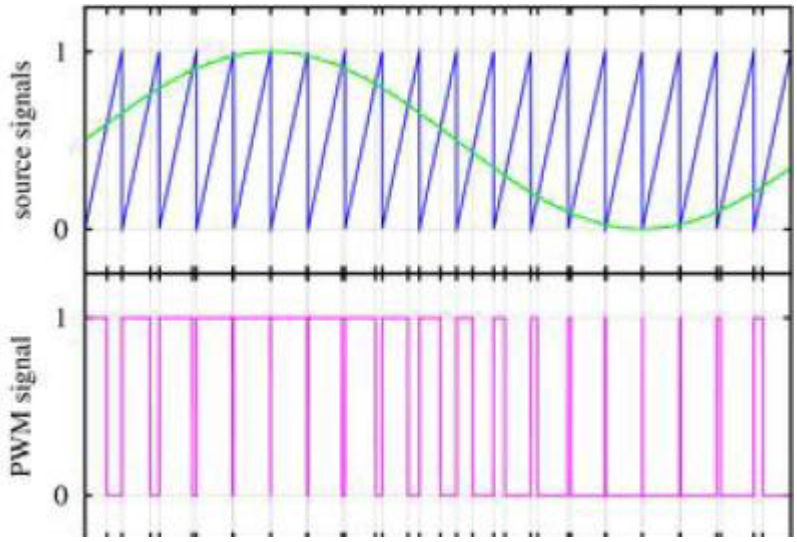
در شکل موج فوق،

$$\begin{aligned}\bar{y} &= \frac{1}{T}\left(\int_0^{DT} y_{max}\,dt + \int_{DT}^{T} y_{min}\,dt\right)\\ &= \frac{D\cdot T\cdot y_{max}+T(1-D)y_{min}}{T}\\ &= D\cdot y_{max} + (1-D)\,y_{min}\end{aligned}$$

که در آن y مقدار ولتاژ یکسو شده میباشد.

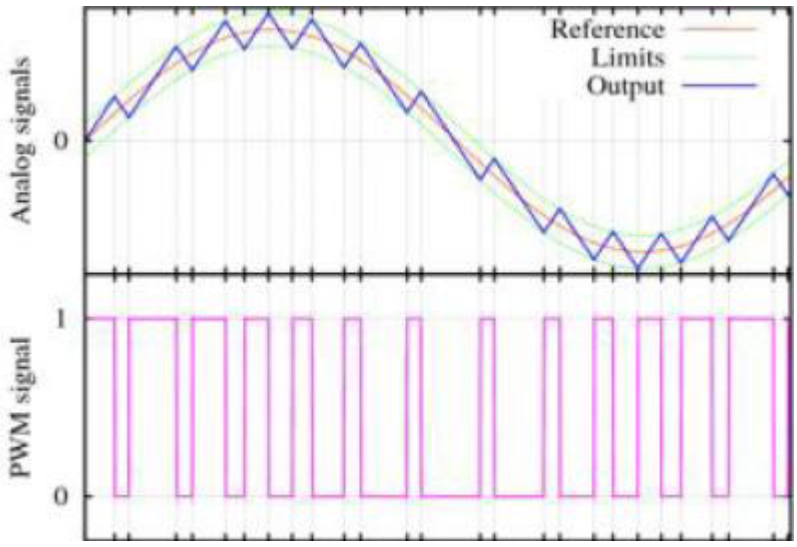
روش های تولید PWM:

1) ساده ترین راه ایجاد PWM استفاده از یک موج اره ای و یک موج سینوسی ایجاد ده توسط اسیلاتور می باشد. حال کافی است تا با استفاده از یک Op-Amp این دو ولتاژ را با یکدیگر مقایسه شوند. در صورتی که ولتاژ سبز رنگ در شکل زیر بیشتر از ولتاژ آبی رنگ باشد، سطح ولتاژ بالا و در صورتی که منحنی سبز رنگ از منحنی آبی رنگ پائین تر باشد، سطح ولتاژ پائین خواهد بود.



2) روش DELTA:

در این روش ولتاژ خروجی با دو سطح ولتاژ معین که یکی از آنها همان مقدار ولتاژ اول با مقداری Offset می باشد، مقایسه میشود. در صورتی که ولتاژ خروجی از یکی از این دو محدودیت افزایش یا کاهش یابد، در این صورت سطح ولتاژ پالس نیز تغییر خواهد کرد. شکل زیر بیان گر این موضوع می باشد.



روش 3) در این روش ولتاژ خروجی از یک ولتاژ مرجع کمتر میشود، در صورتی که مجموع این ولتاژ خطا (تفاضل ولتاژ خروجی از ولتاژ مرجع) از مقدار معینی بیشتر شود، سطح ولتاژ خروجی عوض میشود.

روش 4) بسیاری از مدارات دیجیتال می توانند PWM تولید کنند. برای مثال بسیاری از میکروپروسسور ها دارای خروجی PWM می باشند. معمولاً این میکروپروسسور ها درای شمارنده ای می باشند که پس از زمان معینی سطح ولتاژ خروجی را تغییر می دهند.

علاوه بر روش های گفته شده برای تولید PWM، سه حالت کلی برای PWM موجود می باشد:

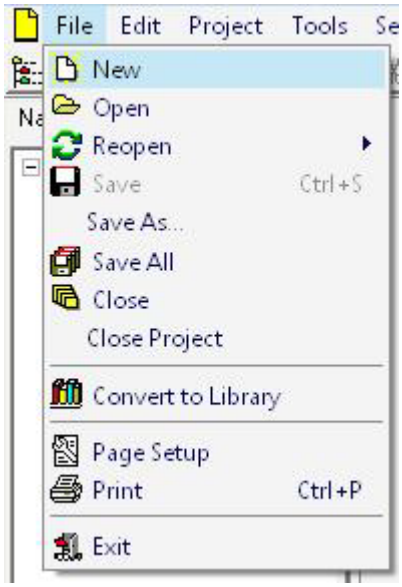
1) مرکز پالس در روی محور زمانی ثابت باشد و با افزایش یا کاهش کناره های (لبه های کناری) مدولاسیون پالس رو تغییر دهیم.
2) لبه بالایی را ثابت نگه داریم و لبه پایینی را تغییر دهیم
3) لبه پایینی را ثابت نگه داریم و لبه بالایی را تغییر دهیم.

از سایر موارد کار برد PWM علاوه بر کنترل دور موتور، می توان کار برد آن در مخابرات و تنظیم ولتاژ و پخش توان اشاره کرد.

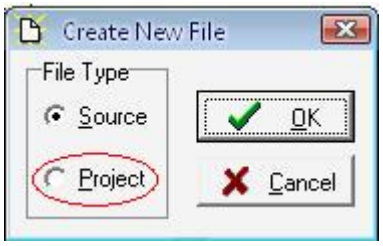
همان طور که اشاره شد یکی از راه های تولید PWM استفاده از میکرو ها یا همان روش های دیجیتالی بود. در اینجا به چگونگی ایجاد PWM با استفاده از AVR ATMEGA 16 می پردازیم.

در راحت ترین حالت برنامه CODE Vision را اجرا کنید:

حال از منوی File گزینه New را انتخاب کنید.

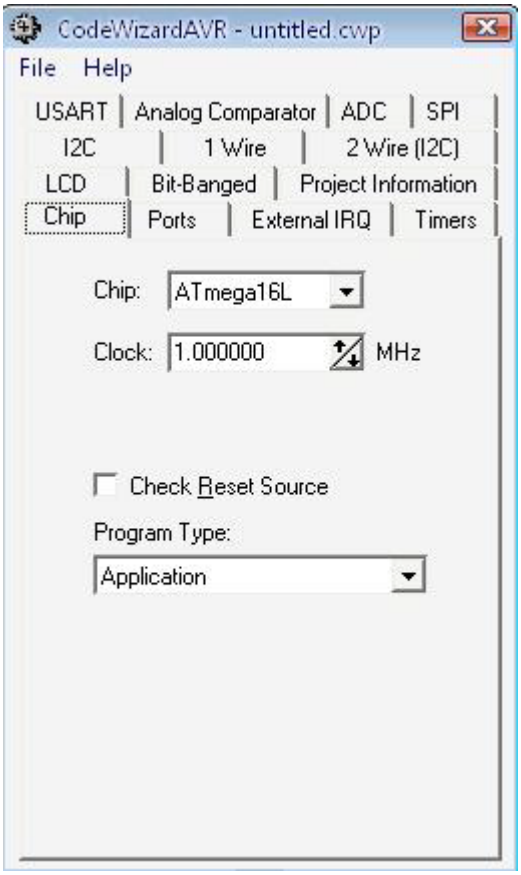


گزینه Project را مطابق شکل زیر انتخاب کنید:



حال گزینه Yes را انتخاب کنید.

صفحه زیر نمایان خواهد شد.

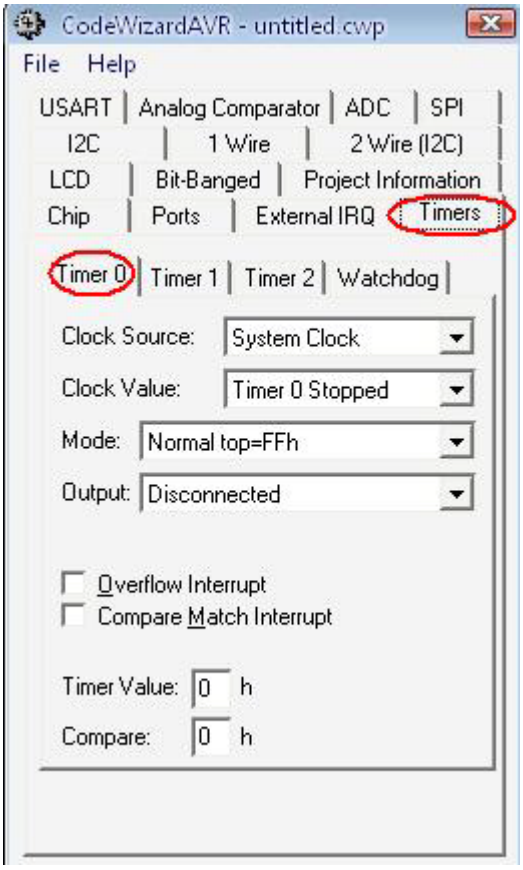


در همین صفحه (قمست Chip) نوع AVR ATMEGA16 مورد استفاده را معین کنید. در اینجا قصد توضیح تمامی قسمت های این پنجره را نداریم بلکه تنها در مورد تایمر ها و چگونگی ایجاد PWM بحث خواهیم کرد. نتیجتاً قسمت هایی مورد نیاز:

Chip
Timers
External IRQ

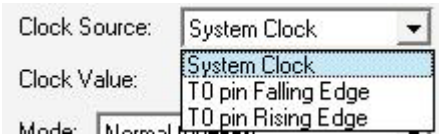
می باشد.

بعد از اینکه در پنجره فوق نوع میکرو و CLOCK آن را مشخص کردید، به منوی Timers بروید.



همان طور که مشاهده می کنید در این میکرو 3 تـایمر در اختیـار شماسـت. فرضـاً TIMER 0 را در نظـر بگیرید.

1) این تایمر 8 بیتی است.
2) Clock Source: در این قسمت منبع کلاک تایمر را انتخاب کنید. یعنی هر بار کـه کلاکـی از ایـن منبع به ورودی تایمر اعمال شود، تایمر یک بار خواهد شمرد. به عبـارتی دیگـر در ایـن قسـمت منظور از Timer، شمارنده است که البته می توان با آن بصـورت Timer نیـز اسـتفاده کـرد. بعـداً چگونگی این کار توضیح داده خواهد شد. در حال حاضر منبع کلاک را انتخاب کنید.



System Clock: با انتخاب این گزینه، فرکانس تایمر مطابق با فرکانس مشخص شده برای تایمر در قسمت Chip می باشد. (در این قسمت فرض می کنیم، نیازی به استفاده از منبع خارجی نباشد و از منبـع داخلـی تایمر استفاده کنیم)

T0 Falling Edge: در بین 40 پایه ی میکرو، پایه ای با نام T0 برای ورودی تایمر صفر در نظ گرفته شده است. انتخاب این گزینه موجب خواهد شد تا در صورت عبور پالس پائین گذر از پایه T0، تایمر (شمارنده) یک بار بشمارد.

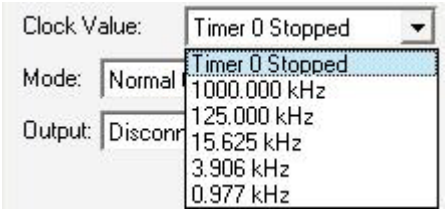
T0 Rising Edge: مطابق همان قسمت فوق ، با این تفاوت که تنها در هنگام عبور لبه بالا رونده، تایمر (شمارنده) خواهد شمرد.

در اینجا برای ایجاد PWM، نیاز به System Clock داریم.

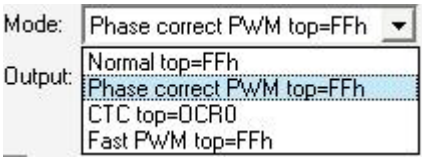
Clock Value:

خوب در این قسمت می توانید در صورت نیاز تنها قسمتی از فرکانس کاری میکرو را برای تایمر استفاده کنید.

فقط دقت کنید که مقداری برای آن انتخاب کنید! (برای روی Timer Stopped رها نکنید.)



Mode: حالت کاری میکرو را مشخص کنید. در این قسمت است که مشخص می کنید از این تایمر قصد استفاده بصورت مولد PWM را دارید.



Normal top=FFh: یعنی بصورت یک شمارنده معمولی با فرکانس تعیین شده در قسمت های قبلی تا عدد 255 خواهد شمرد. (یاد آور میشوم این تایمر 8 بیتی است)

CTC top=0CR0 مخفف (Clear Timer on Compare Match (CTC) Mode):

در این حالت زمانی که مقدار تایمر (مقدار رجیستر TCNT0) به مقدار مشخص شده برای رجیستر OCRO رسید، تایمر Reset میشود. در ضمن میتوانید با فعال نمودن گزینه Compare Match Interrupt، زمانی که تایمر به مقدار رجیستر OCR0 رسید، علاوه بر بازنشانی تایمر، سرویس وقفه مربوطه نیز فعال

میگردد که میتوانید برنامه مورد نظر را در آن سرویس وقفه وارد کنید. همچنین مقدار رجیستر OCR0 که بطور اولیه بر روی صفر تنظیم شده است را می توانید از قمست Compare مشخص کنید. (دقت کنید که عدد را به hex وارد کنید و بزرگتر از 255 نباشد.)

Fast PWM top=FFh:

امکان PWM با فرکانس های بالا را به شما میدهد. در این حالت تایمر تا مقدار رجیستر شمرده و سپس بسته به حالت قسمت Output:

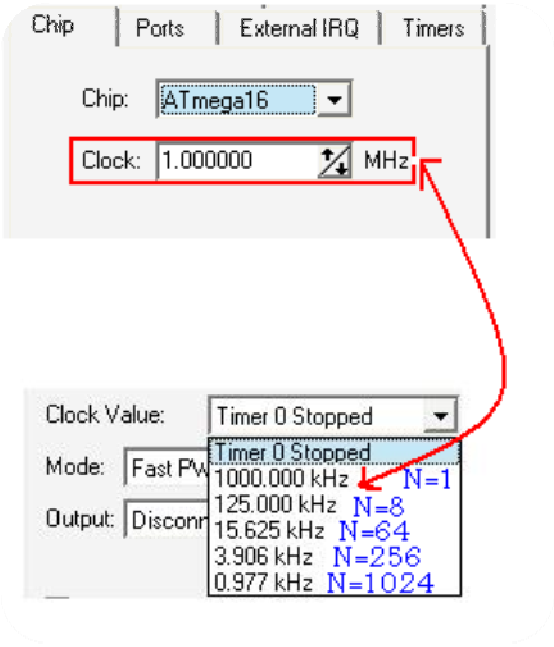
1) قسمت Output (

) بر روی Non_inverted تنظیم شده باشد. در این صورت به ازای مقادیر بیشتر از مقدار رجیستر OCR0 مقدار خروجی تایمر صفر و به ازای مقادیر کوچکتر از رجیستر OCR0 مقدار خروجی تایمر (پایه ی OC0 ـ 4) یک خواهد بود.
2) قسمت Output بر روی inverted تنظیم شده باشد. در این صورت به ازای مقادیر بیشتر از مقدار رجیستر OCR0 مقدار خروجی تایمر، یک و به ازای مقادیر کوچکتر از رجیستر OCR0 مقدار خروجی تایمر (پایه ی OC0 ـ 4)، صفر خواهد بود.

فرکانس سیگنال PWM از رابطه زیر قابل محاسبه است:

$$f_{OCnPWM} = \frac{f_{clk-I/O}}{N.256}$$

به متغیر N، یکی از مقادیر 1، 8، 64، 256، 1024 داده خواهد شد. (Pre-scale) در حقیقت فرکانس مشخص شده برای میکرو توسط متغیر N به فرکانس های پائیین تر تقسیم می شود و تایمر با این فرکانس جدید کار می کند.

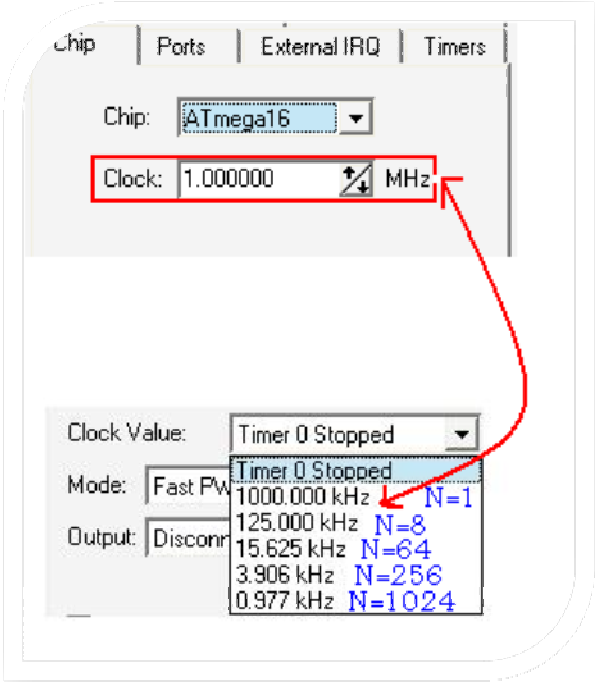


Phase correct PWM top=FFh:

در این حالت تایمر بطور دائم از مقدار صفر تا 255 خواهد شمرد و سپس از 255 بـه صـفر بـاز خواهـد گشت. (255، 254، 253، ...) اگر خروجی در حالت Non-Inverted باشد، زمانی که مقدار تایمر و مقدار رجیستر OCR0 برابر شدند، در حالت بالا شمار خروجی باز نشانی و در حالت پائین شمار خروجـی یـک میشود. فرکانس سیگنال PWM در این حالت برابر است با:

$$f_{OCnPCPWM} = \frac{f_{clk-I/O}}{N.510}$$

بـه متغیـر N، یکـی از مقـادیر 1، 8، 64، 256، 1024 داده خواهـد شـد. (Pre-scale) در حقیقـت فرکـانس مشخص شده برای میکرو توسط متغیر N به فرکانس های پائیین تر تقسیم می شود و تایمر با این فرکـانس جدید کار می کند.



پس بدین ترتیب تا اکنون به سادگی می توانید حدث بزنید که برای تولید PWM مورد نظر می توانید از یکی از حالت های Fast PWM یا Phase correct PWM استفاده کنید. روش سومی نیز استفاده از سرویی وقفه در زمان برابری مقدار تایمر با رجیستر OCR0 می باشد. به این ترتیب با تنظیم مقدار رجیستر OCR0 می توانید PWM خروجی با سیکل کاری مورد نظر را ایجاد کنید.(در این روش سوم، باید گزینه Compare Match Interrupt را نیز فعال کنید.)

حال با در نظر گرفتن توضیحات داده شده، سورس برنامه جهت کنترل دور موتور در AVR چنین خواهد بود:

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.8d Standard
Automatic Program Generator
© Copyright 1998-2006 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : DC Motor Control
Version : 3.2
Date    : 2007/02/21
Author  : Yashar
Company : YBSH
Comments:
```

```
Chip type           : ATmega16
Program type        : Application
Clock frequency     : 1.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x12 ;PORTD
#endasm
#include <lcd.h>
#include <stdlib.h>

int n=0;
int a=0;
int var1=0;
char *s;
int var4=0;

// Timer 2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
TCCR1B=0x07;
n=n+1;
if (n>=15) {
a=TCNT1L;
var4=1;
if(a>100){
var1++;
};
if (var1>=5) {
TCCR2=0x00;
TCNT2=0x00;
OCR0=0;
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("System restarted");
lcd_gotoxy(0,1);
lcd_putsf("Contact Provider");
var1=0;
};
n=0;
TCNT1H=0;
TCNT1L=0;
};
}
// Declare your global variables here


void main(void)
{
```

```
// Declare your local variables here
int b=0;
int IRPS=0;
int WRN=0;
int var2=4;
int var3=0;

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=0 State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x08;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 0.977 kHz
// Mode: Fast PWM top=FFh
// OC0 output: Non-Inverted PWM
TCCR0=0x6D;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: T1 pin Rising Edge
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
```

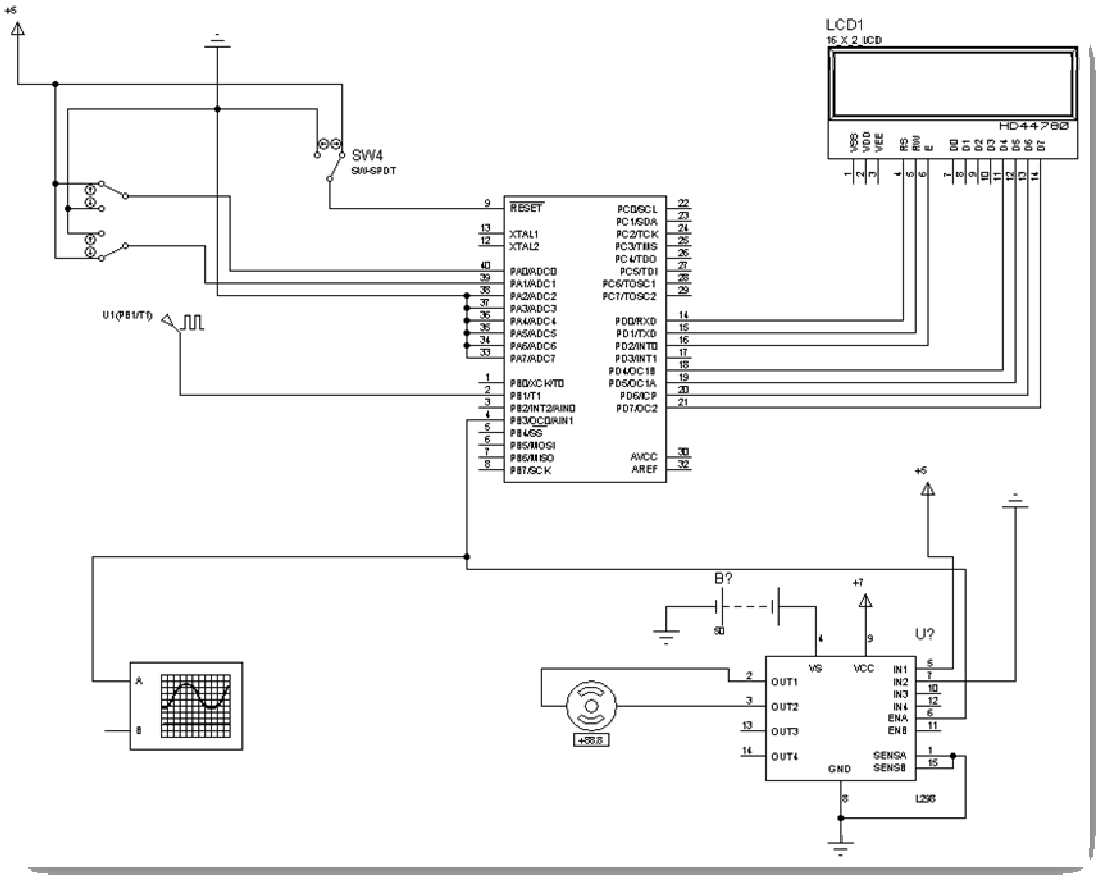
```
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 3.906 kHz
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x06;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x40;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")

while (a<100)
      {
      // Place your code here
if (var4==1) {
      b=PINA;
if (var2!=b) {
var2=b;
      if (b==0) {
      IRPS=5;
      OCR0=0;
};
if (b==1) {
IRPS=30;
OCR0=85;
};
```

```
if (b==2) {
IRPS=50;
OCR0=128;
};
if (b==3) {
IRPS=66;
OCR0=170;
};
};
var3=0;
WRN=0;
if (a<IRPS-5) {
if (OCR0>240) {
/*lcd_clear();
itoa(a,s);
lcd_gotoxy(0,0);
lcd_puts(s);
itoa(IRPS,s);
lcd_gotoxy(0,1);
lcd_puts(s);
lcd_putsf(" OCR0=");
itoa(OCR0,s);
lcd_puts(s);*/
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Contact Provider");
lcd_gotoxy(0,1);
lcd_putsf("Error NE_HL01");
WRN=1;
var3=1;
};
if (var3==0) {
OCR0=OCR0+15;
};
};
if (a>IRPS+4) {
if (OCR0<15) {
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Contact Provider");
lcd_gotoxy(0,1);
lcd_putsf("Error NE_HL02");
WRN=1;
var3=1;
};
if (var3==0) {
OCR0=OCR0-15;
};
};
if (WRN==0) {
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("    OCR0=");
itoa(OCR0,s);
lcd_puts(s);
```

```
lcd_gotoxy(0,1);
lcd_putsf("Status:Runing...");
};
};
    };
}
```

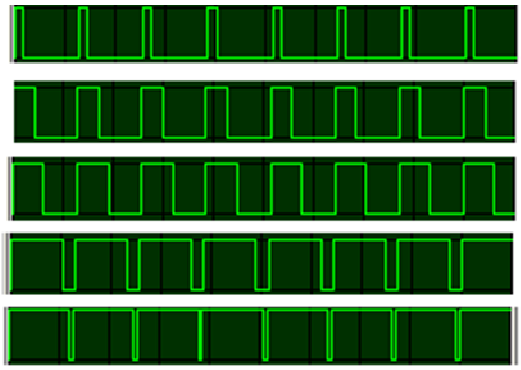
در برنامه فوق، پالس های ایجاد شده توسط Shaft Encoder دیجیتالی با استفاده از TIMER0 و TIMER2 شمارش شده و سپس با مقادیر مشخص شده در ورودی مقایسه شده و Duty Cycle برای سیگنال PWM تنظیم می شود. این سیگنال PWM به پایه ENABLE راه انداز موتور (L298) داده می شود و بدین ترتیب توان داده شده به موتور کنترل می شود.

مدار کنترل گر نیز به شکل زیر می باشد:



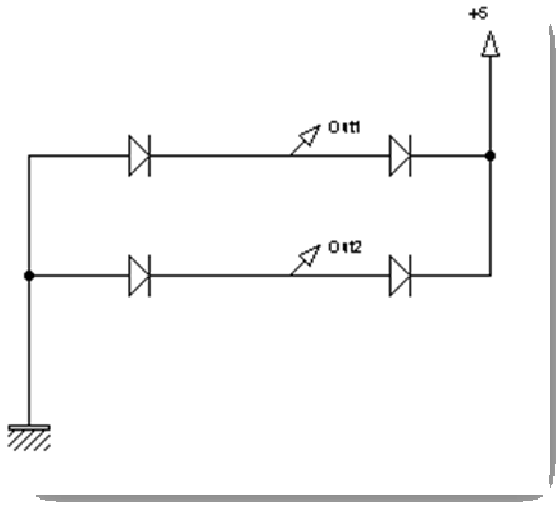
خروجی موتور توسط ENCODER به میکرو منتقل میشود تا دور موتور اندازه گیری شده و تنظیم PWM خروجی صورت گیرد با استفاده از 3 کلید می توان مقدار PWM را افزایش یا کاهش داد.

با تنظیم کلید ها از مقدار 1 تا 5 مقدار PWM مطابق شکل زیر افزایش خواهد یافت:



توجه:

در شبیه سازی رایانه ای، از جریان های برگشتی موتور صرف نظر می شد اما در عمل نمی تـوان از آن هـا صرف نظر کرد و در نتیجه از ترکیب دیودی زیر از عبور جریان هـای برگشـتی بـه مـدار کنترلـی و ایجـاد اختلال احتمالی جلوگیری کردیم.



حال باید بتوانیم برنامه ای نوشته شده برای میکرو را به میکرو انتقال دهیم. همان طور که میدانیم، برای انتقال برنامه به میکرو باید برنامه به زبان ماشین کامپایل شود. این کار توسط کامپایلر برنامه Code Vision صورت می پذیرد.

برای این کار کافی است تا پس از اتمام برنامه نویسی و رفع مشکلات احتمالی که توسط کامپایلر Code Vision اعلام میشوند، دستورات

1) Check Syntax for currently edited file
2) Compile the Project
3) Make the Project

را بترتیب اجرا کنید. آیکون این دستورات در شکل زیر نمایش داده شده است:

به ترتیب از سمت راست، گزینه های 3 تا 1.

حال فایل HEX، برای انتقال به میکرو در پوشه مبوطه آماده شده است. برنامه های متعددی برای Program کردن میکرو وجود دارد. با برنامه Code Vision نیز می توانید این کار را انجام دهید. برای این کار ابتدا باید تنظیمات مربوط به Programmer را برای Code Vision معین کنید.

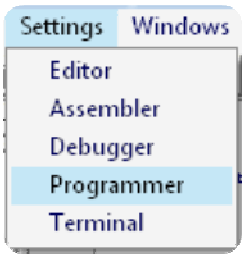
قبل از ادامه بحث در مورد نحوه برنامه ریزی میکرو به نحوه ساخت Programmer برای میکرو های AVR خواهیم پرداخت. از انواع Programmer های موجود، به علت سادگی از Programmer مدل STK200/300 استفاده خواهیم کرد. این Programmer از پورت موازی چاپگر برای اتصال به میکرو استفاده میکند.

چند نکته:

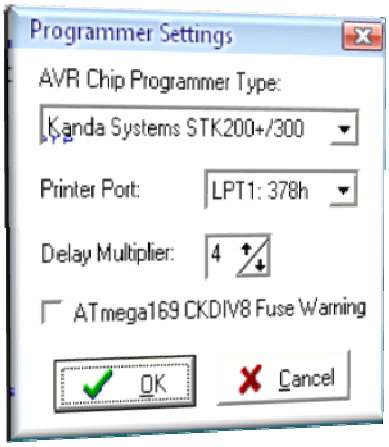
1) در هنگام استفاده از حویه، دقت کنید که مدت لحیم کاری بیش از 10 ثانیه نباشد.
2) در صورت امکان با استفاده از پایه های نظامی از قرار دادن IC بطور مستقیم بر روی برد خودداری کنید. تا در صورت زوم تعویض IC زمان کمتری را طلب خواهد کرد.
3) در صورتی که از پین نظامی استفاده کردید، از قرار داده IC در هنگام لحیم کاری بر روی مدار خودداری کنید.
4) از برقراری اتصال بطور صحیح اطمینان حاصل کنید.
5) در صورت امکان از کدار چاپی بجای برِد برد ها استفاده کنید.
6) از اتصال کوتاه کردن پایه های Vcc و GND در حین تست مدار خودداری کنید. (ممکن است موجی اسیب IC شود.)
7) در صورتی که الکتریسیته ساکن از طریق دست شما به IC منتقل شد، از سوختن IC خود مطمئن باشید!!!! قبل از تماس با پایه های IC دستان خود را زمین کنید.

و اما در مورد Code Vision و نحوه برنامه ریزی میکرو.

به منوی Setting رفته دستور Programmer را اجرا کنید.



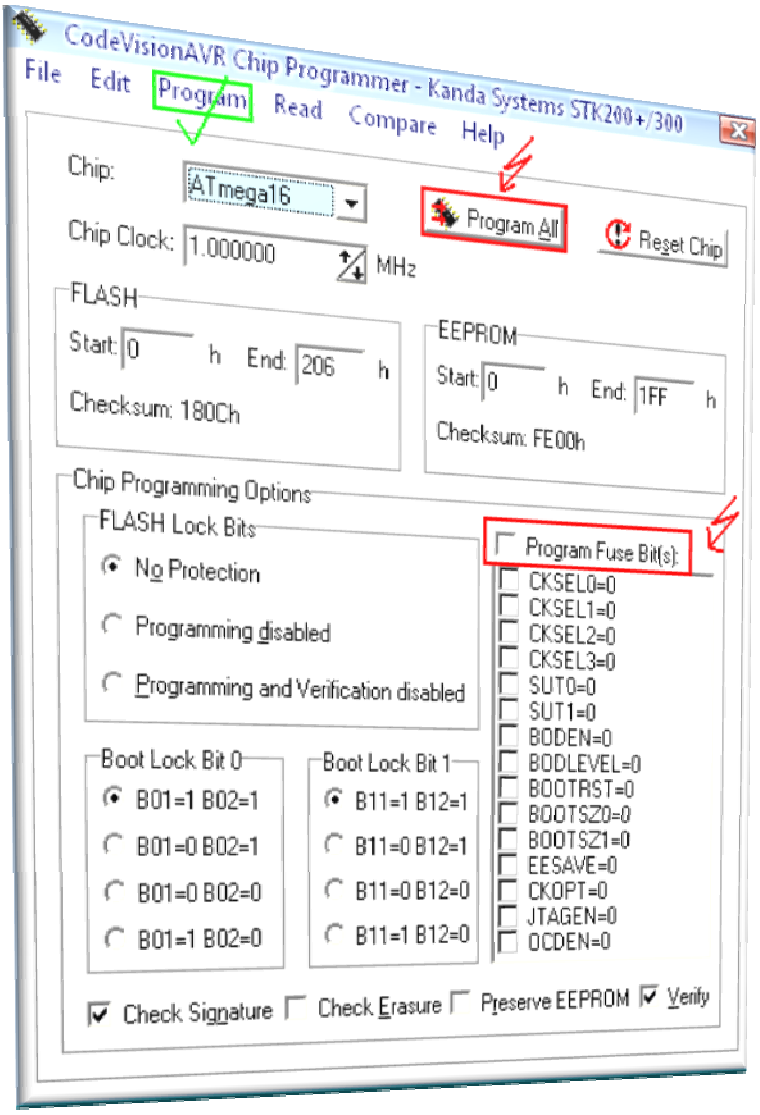
حال این قسمت را مطابق شکل زیر تنظیم کنید:



گزینه OK را بزنید و از منوی Tools دستور Chip Programmer را اجرا کنید.

در پنجره جدید باز شده به نکات زیر دقت کنید:

با این Programmer نمیتوانید فیوزبیت ها را مجدداً باز نویسی کنید، در نتیجه برای مثال، اگر فیوز بیت RSTDISBL برنامه نویسی شود، پایه ریست از حالت عادی خارج شده و میکرو فقط در حالت موازی برنامه ریزی خواهد شد. یعنی مجدداً با این Programmer برنامه ریزی نخواهد شد. اگر بخواهید دوباره میکرو را با STK200 استفاده کنید، باید ابتدا این فیوز بیت را به حالت اول بازنشانی کنید که برای این کار نیز باید از انواع دیگر Programmer استفاده کنید. برای اینکه با این مشکل مواجه نشوید، از غیر فعال بودن گزینه Program Fuse Bits اطمینان حاصل کنید. همچنین از گزینه Program All برای برنامه ریزی میکرو استفاده نکنید. بلکه از از منوی Program، دستور Flash را استفاده کنید.



حال می توانید میکرو را در مدار استفاده کنید.

چند نکته:

1) برای اینکه از صحت کار Programmer مطمئن شوید، ابتدا برنامه بسیار ساده ای را بنویسید و به میکرو منتقل کنید. مثلاً PORTD را برابر 7 یا مقداری دلخواه کنید. سپس بعد از برنامه ریزی میکرو این مقدار را در خروجی میکرو تست کنید.

---

2) در هنگام بستن مدارات، گام به گام پیش بروید. یعنی در هر مرحله تنها قسمتی از مدار را بسته و پس از اینکه از آن قسمت از مدار نتیجه گرفتید به مرحله بعدی بروید. این روش بهتـر از روشـی است که تمام مدار را بسته و سپس بدنبال خطاهای احتمالی باشید!!