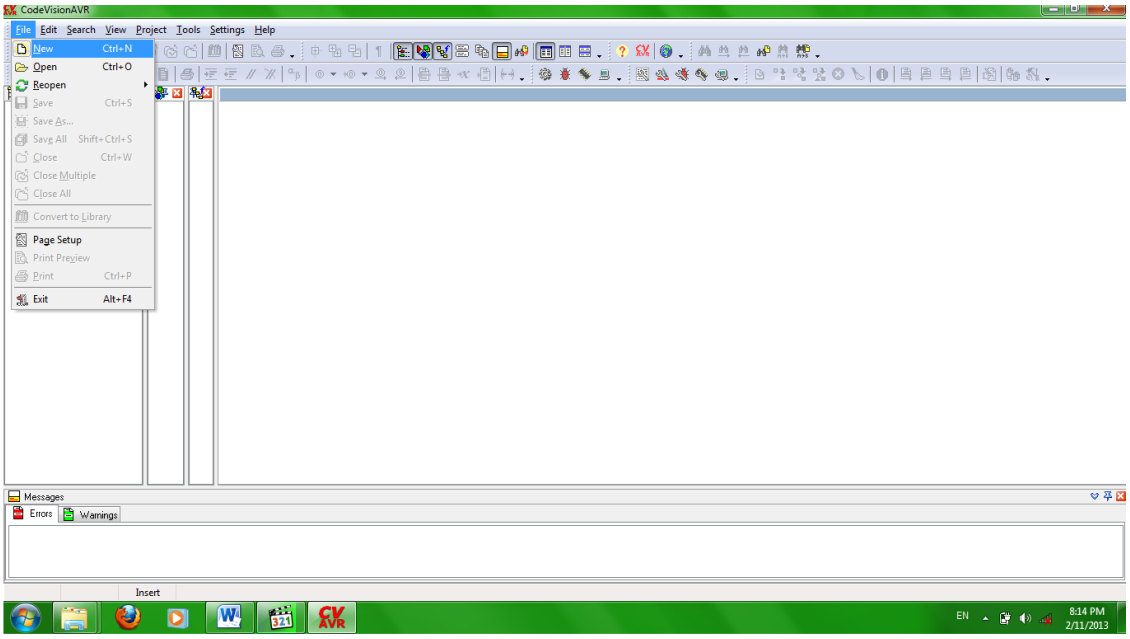
آزمایشگاه میکروکنترلر

جلسه اول:

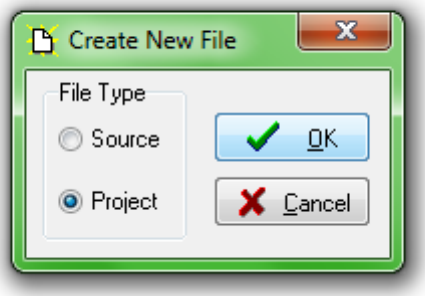
شروع کار با کامپایلر کدویژن

نمایی از محیط نرم افزار :

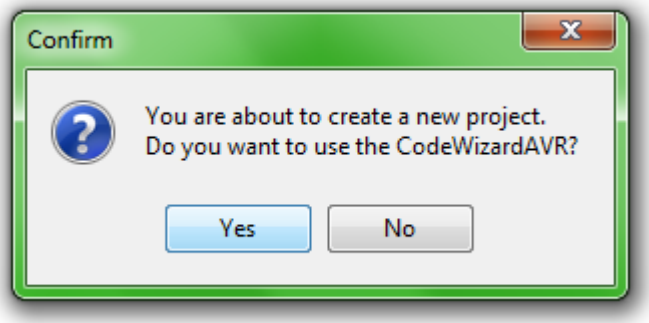


برای شروع کار با کامپایلر

ابتدا باید پروژه تعریف شود برای اینکار از منوی فایل NEW را انتخاب می کنیم.

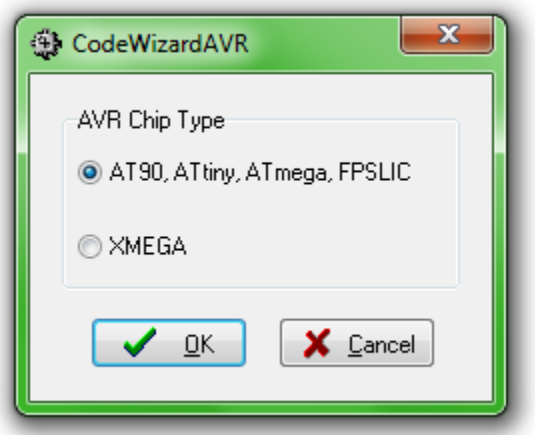


سپس در پنجره باز شده project را انتخاب و ok را می زنیم.

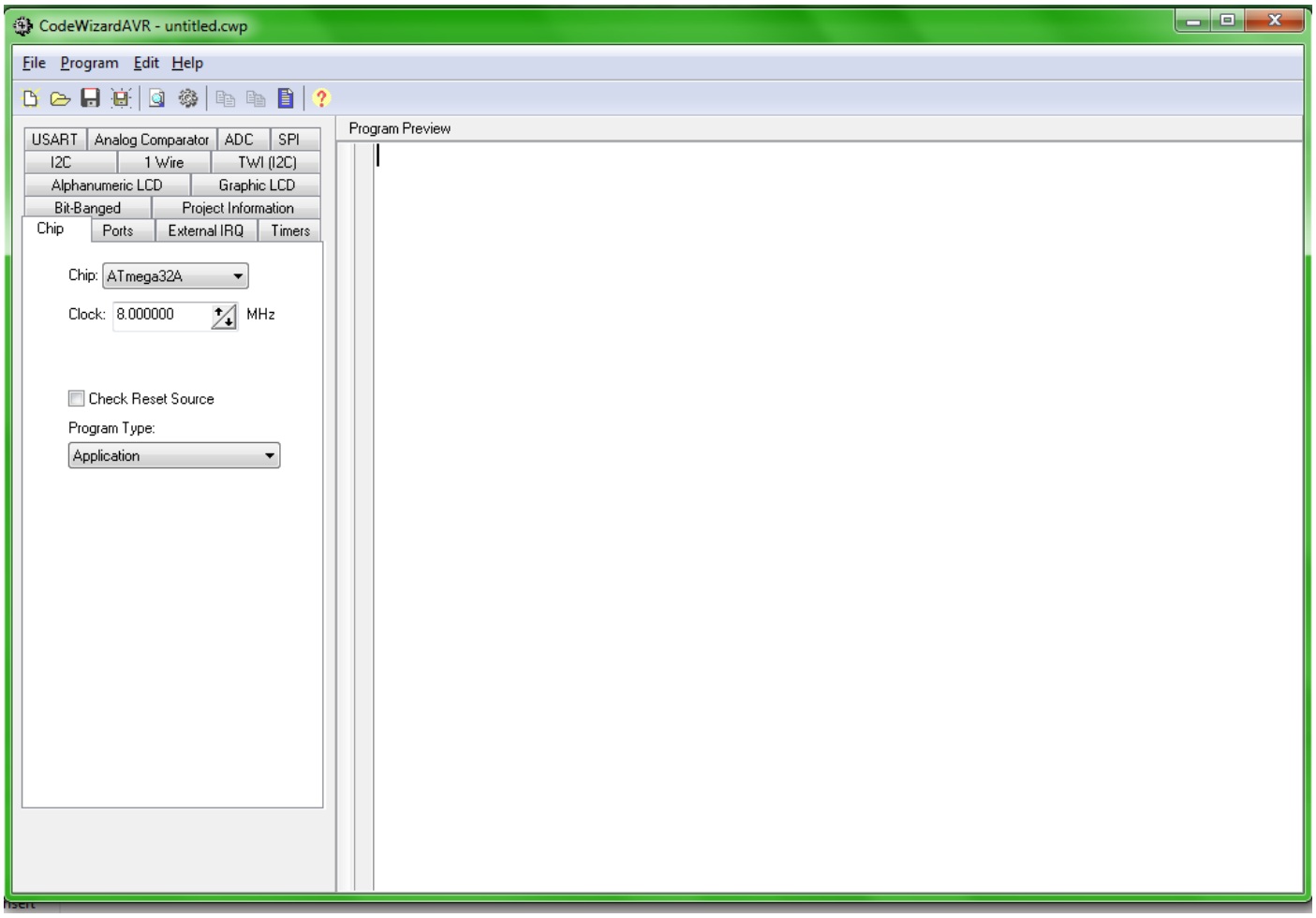


پنجره ای باز می شود که در آن از کاربر سوال می کند که آیا می خواهید از کدویزار برنامه استفاده یا نه ؟

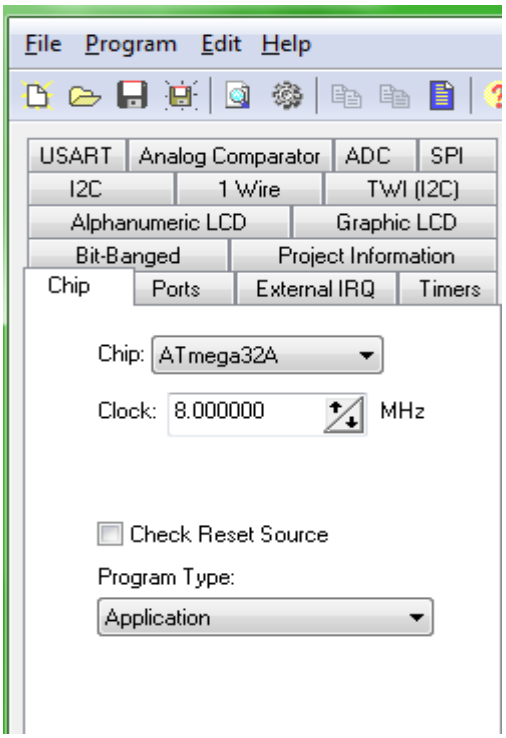
که ما از کدویزار استفاده می کنیم .



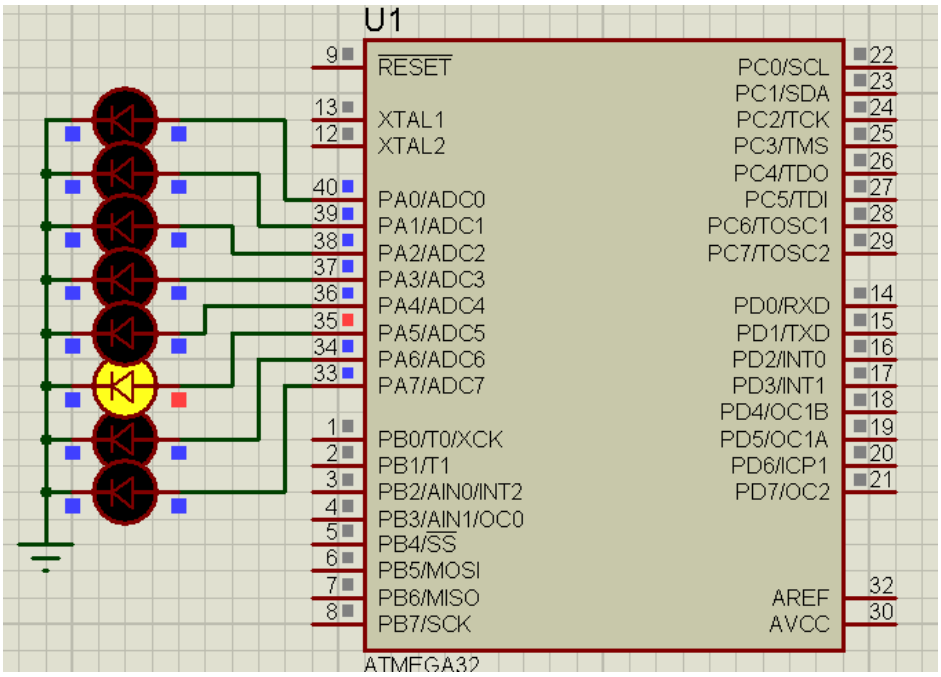
چون ورژن جدید کامپایلر از سری جدید avr یعنی xmega هم پشتیبانی می کند در مرحله بعد از شما می خواهد که نوع میکری خود را انتخاب کنید ما سری اتمگا را انتخاب می کنیم.



بعد اینکه ok کردیم برنامه کد ویزار باز می شود .از این پنجره می توانیم نوع میکرو ،پالس ساعت،رجیسترها و قسمت های مختلف میکرو را تنظیم کنیم.



از زبانه chip نوع میکرو و فرکانس کاری آن را انتخاب می کنیم.

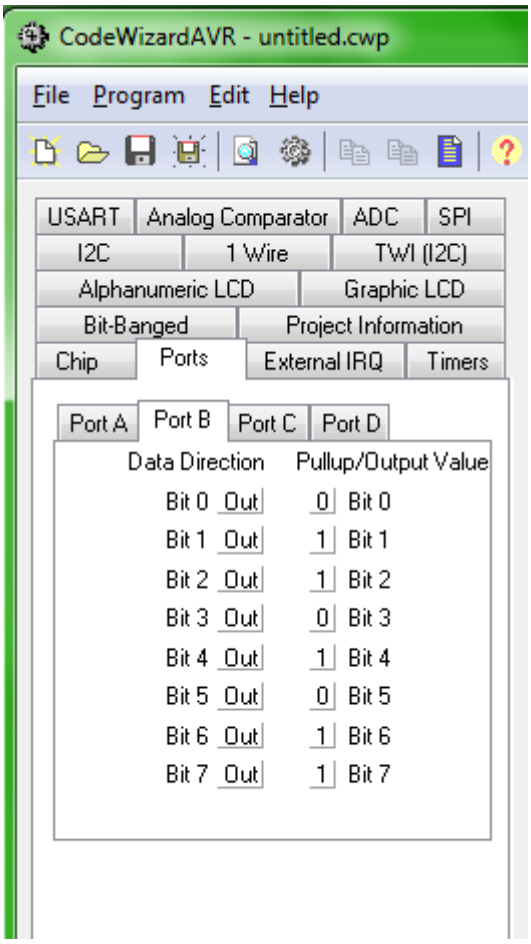


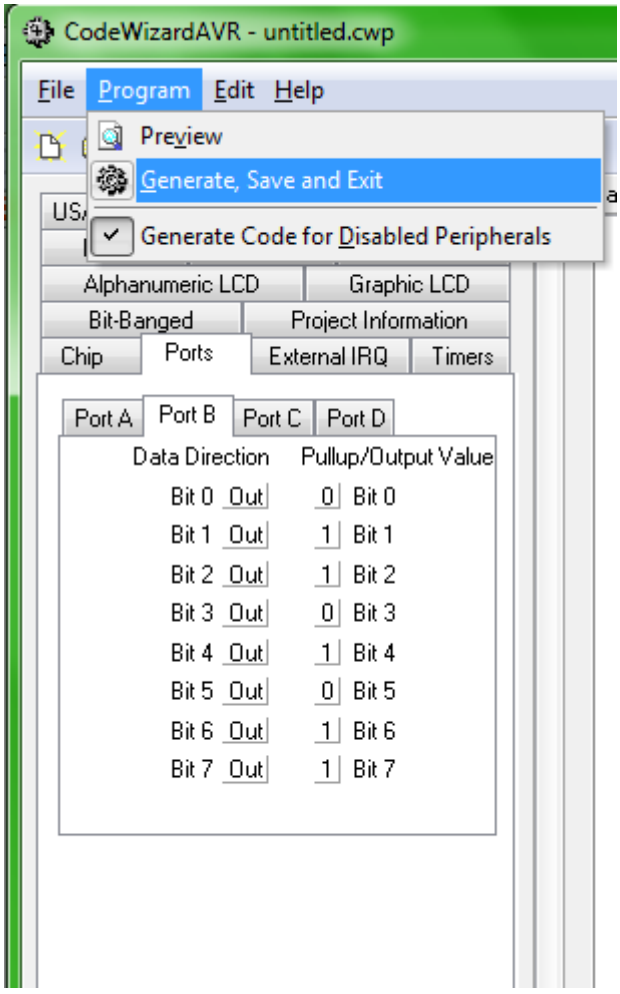
اولین پروژه : چشمک زن

در تمام پروژه ها مراحل بالا یکسان است .

در مدار چشمک زن یکسری LED به پایه های IO میکرو وصل می شود و بصورت دلخواه خاموش روشن می شوند .

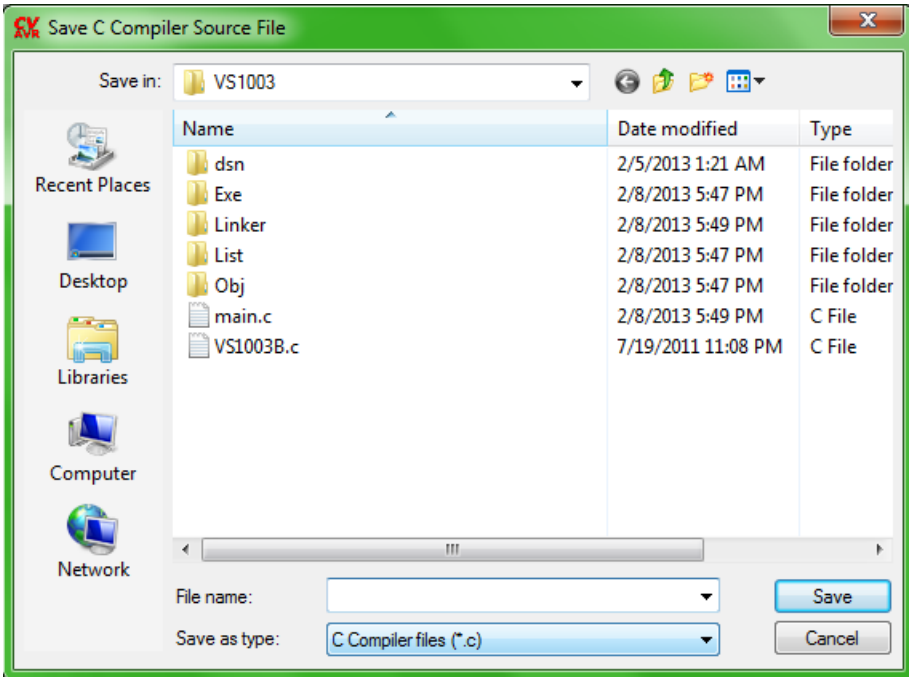
برای اینکار باید پایه های میکر را به عنوان خروجی تعریف کنیم .


CodeWizardAVR - untitled.cwp
File Program Edit Help
USART Analog Comparator ADC SPI
I2C 1 Wire TWI (I2C)
Alphanumeric LCD Graphic LCD
Bit-Banged Project Information
Chip Ports External IRQ Timers
Port A Port B Port C Port D
Data Direction Pullup/Output Value
Bit 0 Out 0 Bit 0
Bit 1 Out 1 Bit 1
Bit 2 Out 1 Bit 2
Bit 3 Out 0 Bit 3
Bit 4 Out 1 Bit 4
Bit 5 Out 0 Bit 5
Bit 6 Out 1 Bit 6
Bit 7 Out 1 Bit 7


CodeWizardAVR - untitled.cwp
File Program Edit Help
Preview
Generate, Save and Exit
Generate Code for Disabled Peripherals
Alphanumeric LCD Graphic LCD
Bit-Banged Project Information
Chip Ports External IRQ Timers
Port A Port B Port C Port D
Data Direction Pullup/Output Value
Bit 0 Out 0 Bit 0
Bit 1 Out 1 Bit 1
Bit 2 Out 1 Bit 2
Bit 3 Out 0 Bit 3
Bit 4 Out 1 Bit 4
Bit 5 Out 0 Bit 5
Bit 6 Out 1 Bit 6
Bit 7 Out 1 Bit 7

طبق شکل روبرو پایه ها را بصورت OUT تنظیم می کنیم و همچنین می توانیم به پایه ها مقدار اولیه نیز بدهیم .

بعد از تنظیم کلیه رجیسترهایی که در پروژه نیاز داریم طبق شکل زیر از منوی PROGRAM زیرمنوی save and exit ,Generate را انتخاب می کنیم.



سپس از شما می خواهد که پروژه را ذخیره کنید.

کد خود را در این قسمت می نویسیم

```
while (1)
      {
      // Place your code here

      }
}
```

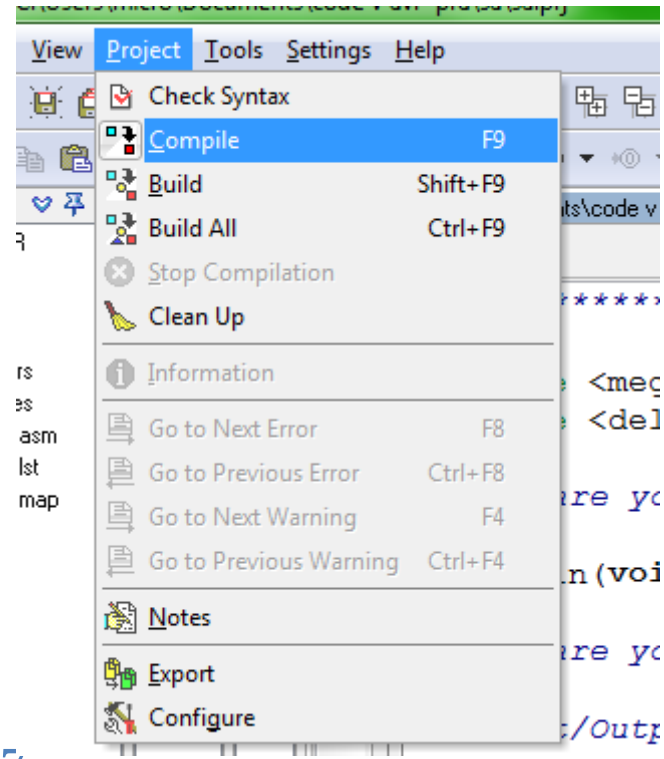
برنامه زیر هر 120میلی ثانیه یک واحد به پورت بی اضافه می کند .البته برای اینکه بتوانید از تابع delay_ms() استفاده کنید باید کتابخانه آنرا اضافه کنید .

```
while (1)
      {
      PORTB++;
      delay_ms(120);

      }
}
```

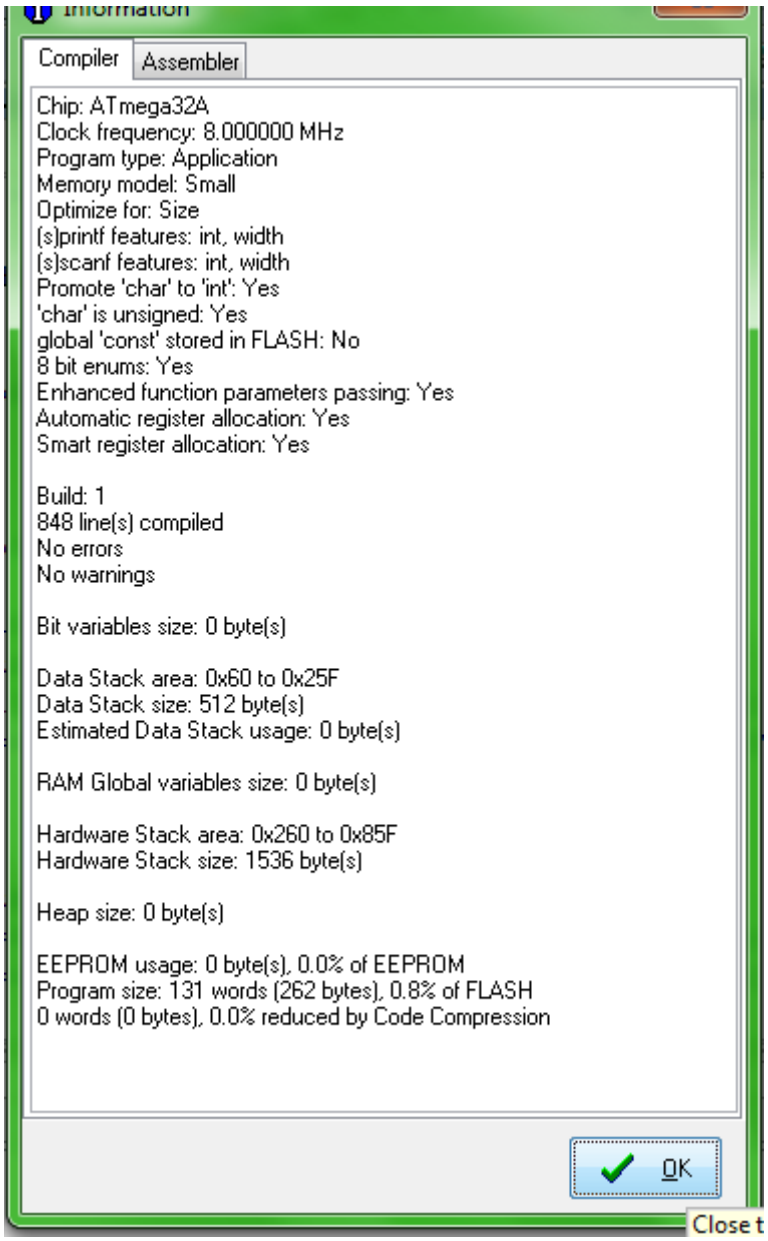
کتابخانه ها باید در اول برنامه فراخوانی شوند .شکل زیر نحوه نوشتن آنرا نشان می دهد.

```
*****************************************************/

#include <mega32a.h>
#include <delay.h>

// Declare your global variables here

void main(void)
{
```

سپس از منوی project پروژه را کامپایل می کنیم .برای اینکه فایل هگز تولید شود باید build all را بزنیم.



اگر در کدنویسی مشکلی وجود نباشد بعد از کامپایل با پنجره زیر روبرو می شوید .



جلسه دوم

وضعیت پایه های تغذیه قسمت آنالوگ به دیجیتال میکرو وقتی از این قسمت استفاده نمی شود . در این وضعیت **AVCC** به **VCC** میکرو و **AGND** به **GND** میکرو و پایه رفرنس به **VCC** وصل می شود.

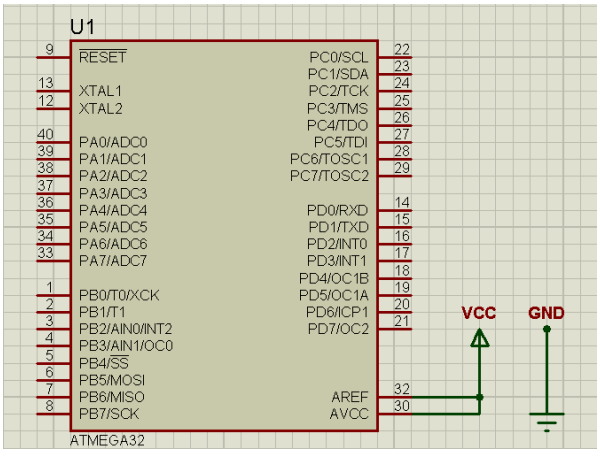
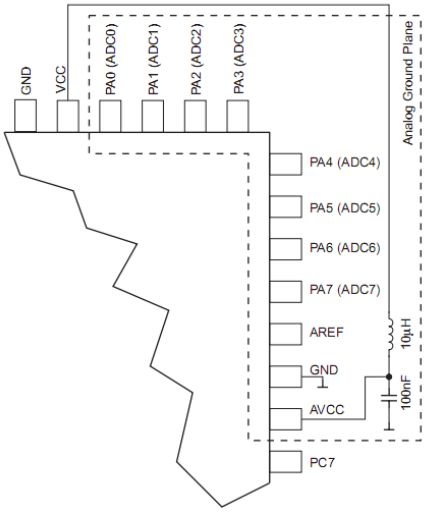
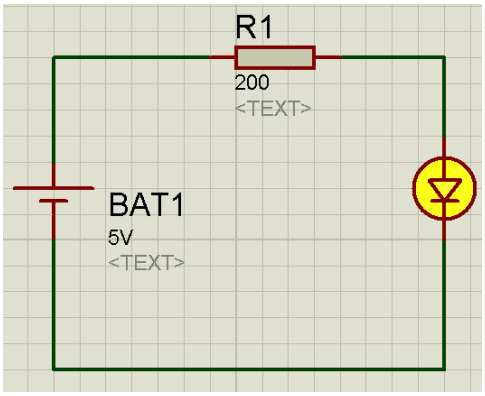


Figure 106. ADC Power Connections



اگه از قسمت آنالوگ به دیجیتال میکرو استفاده کنید باید GND و AVCC مطابق شکل روبرو بسته شوند .

که دلایل خودش را دارد که در دیتاشیت کاملا تشریح شده است.



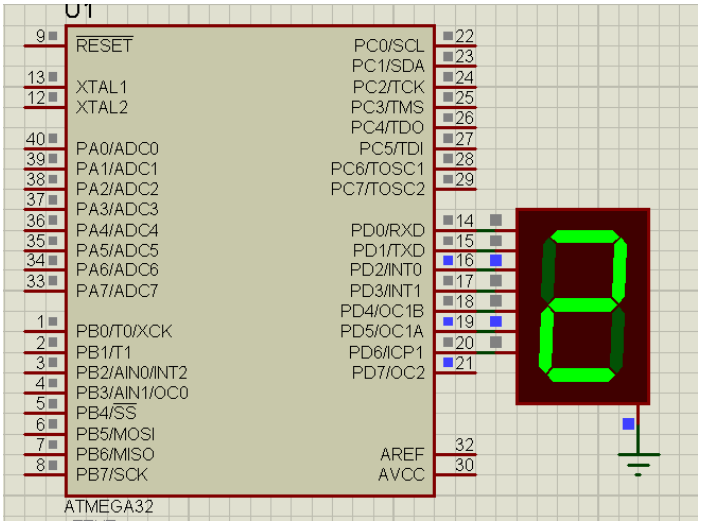
محاسبه مقاومت کنترل جریان LED :

چون میکرو با 5 ولت کار می کند و LED تقریبا با 2 ولت کار می کنند باید مقاومتی برای تقسیم ولتاژ و کنترل جریان با LED سری کنیم .

حداکثر جریان هر پین میکرو 20میلی آمپر است ولی باید جریانی حدود یک چهارم جریان نامی از میکرو جریان بکشیم تا فشای روی میکرو وارد نشود .البته در محیط های صنعتی و در بارهایی با جریان بالاتر از 20میلی آمپر از ترانزیستور استفاده می کنند.

پروژه هفته دوم:

راه اندازی 7سگمنت :شمارنده از صفر تا نه و برعکس.



اول از همه باید کدهای سون سگمنت را به دست بیاوریم . سون سگمنت من کاتد مشترک است و کدهای آن بصورت زیر می باشد :

```
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f}
```

Notes 7seg.c

```
#include <mega32.h>
#include <delay.h>
```

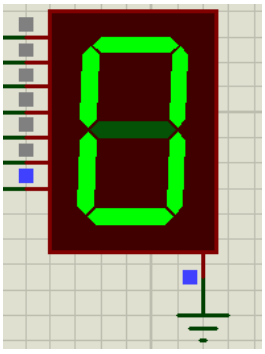
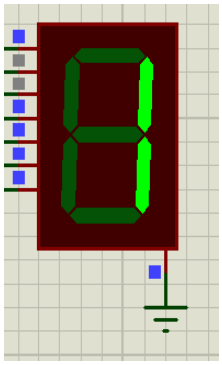
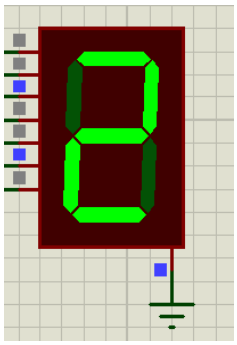
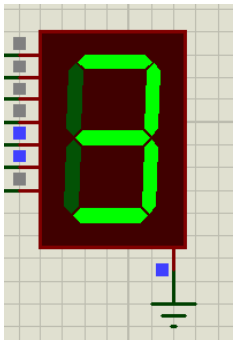
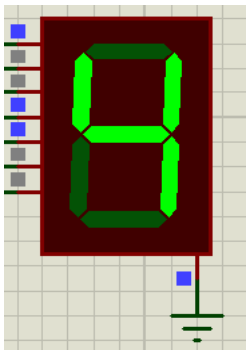
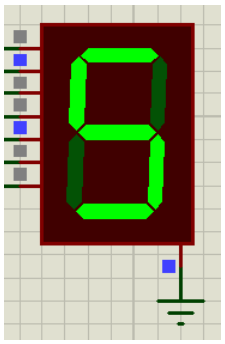
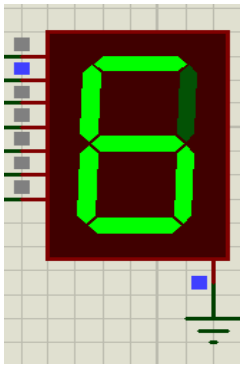
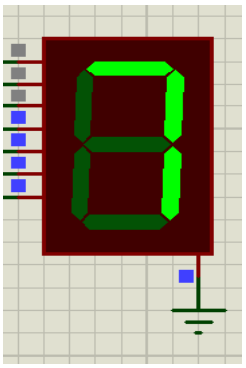
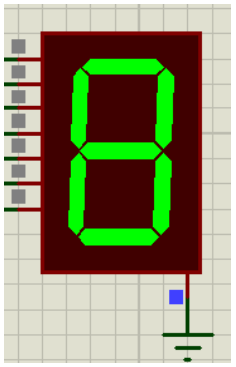
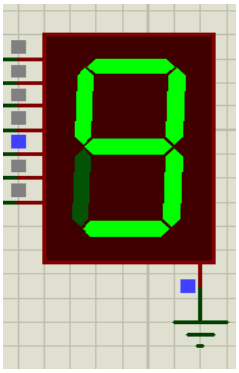
سپس همانند پروژه های قبل اول هدر ها را به برنامه اضافه می کنیم : در این پروژه فقط از دو کتابخونه تاخیر و مگا32 استفاده می کنیم.

```
void main()
{
    char i;
    DDRD=0xff;
    while(1)
    {
        for(i=0;i<10;i++)
        {
            PORTD=code[i];
            delay_ms(300);
        }
        delay_ms(50);
          for(i=9;i>1;i--)
        {
            PORTD=code[i];
            delay_ms(300);
        }
    }
}
```

بقیه برنامه :

پورت D را بصورت خروجی تعریف می کنیم و یک متغییر بنام i و از نوع char تعریف می کنیم .

داخل تابع اصلی یک حلقه بی نهایت تعریف می کنیم و داخل آن حلقه فور تعریف می کنیم که فقط 10بار داخل حلقه را اجرا می کند و هر بار یک واحد به مقدار متغییر اضافه می کند و در حلقه فور بعدی برعکس :

جلسه سوم

ساخت منبع تغذیه مناسب برای میکرو

تغذیه میکرو اهمیت بسزایی دارد چرا که تغییرات ناگهانی تغذیه باعث رست شدن میکرو می شود.چون داخل میکرو برای تشخیص سطح ولتاژ مداری پیش بینی شده و از فیوز بیت ها قابل تنظیم است .به هرحال تغذیه مناسب میکرو اهمیتش خیلی بیشتر از قسمت های دیگر مدار است.

ما در اینجا برای رفع این مشکل از یک مدار تغذیه سوئیچینگ استفاده می کنیم .چون طراحی مدار سوئیچینگ با قطعات مجزا برای ما مقرون به صرفه نیست از یک آی سی سوئیچینگ استفاده می کنیم.در زیر به بررسی این آی سی رگولاتور و مشخصاتش می پردازیم .

آی سی رگولاتور LM2576 :

این آی سی در انواع ولتاژهای مختلف 3.3 ،5 ،12 و یک نوع قابل تنظیم تولید می شود.

3 آمپر جریان گارانتی .

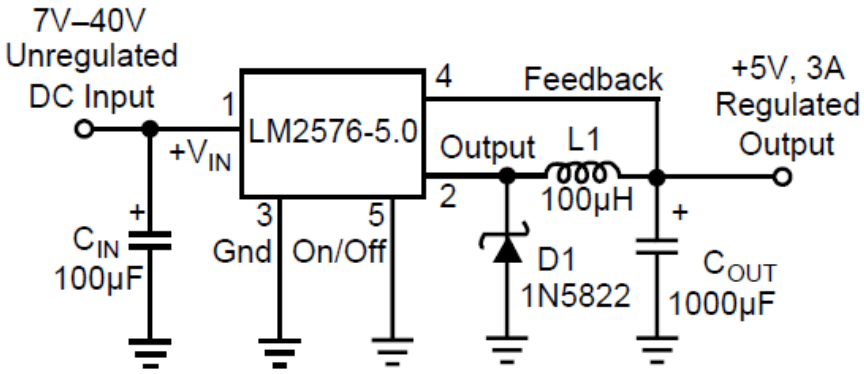
رنج ولتاژ ورودی از 4 تا 40 ولت.

رنج ولتاژ خروجی از 1.23 تا 37 ولت.

فقط به چهار قطعه خارجی نیاز دارد.

52کیلوهرتز فرکانس اسیلاتور داخلی آی سی.

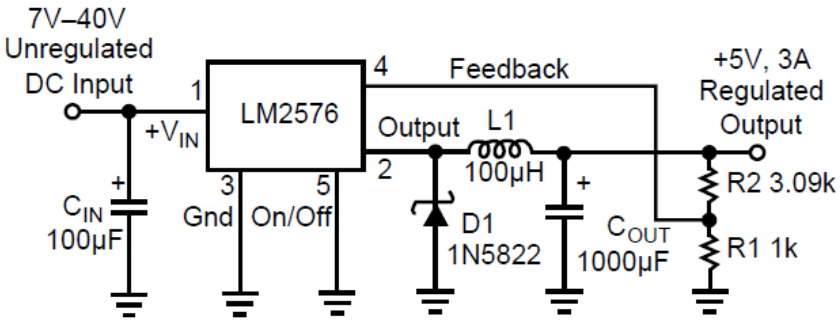
شماتیک رگولاتور ولتاژ برای ولتاژ ثابت 5 ولت:



Note: Pin numbers are for TO-220 Package

Fixed Regulator in Typical Application

شماتیک مدار رگولاتور متغییر مقدار مقاومت های R1 و R2 مقدار ولتاژ خروجی را تعیین می کند.

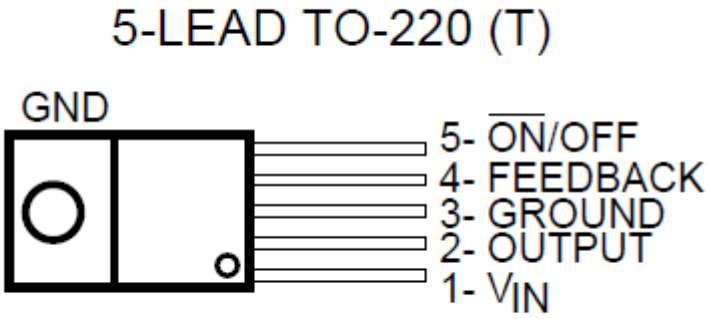


Note: Pin numbers are for TO-220 Package

$$V_{OUT} = 1.23\left(1 + \frac{R2}{R1}\right)$$

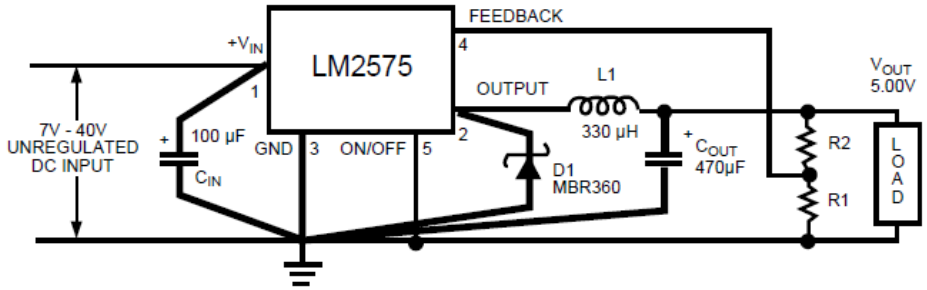
Adjustable Regulator in Fixed Output Application

شکل ظاهری رگولاتو و پایه های رگولاتور:

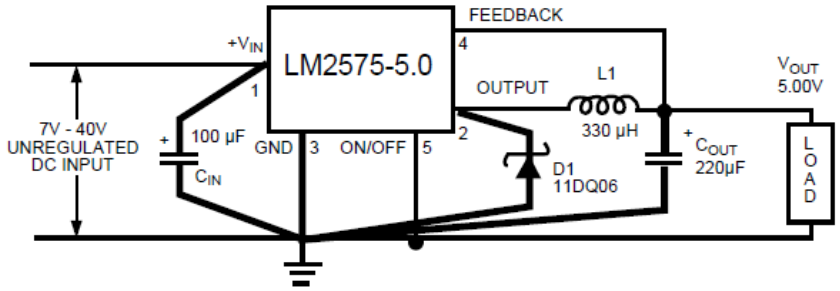


توصیه هایی برای رسم پی سی بی :

خازن و دیود ها باید در یک نقطه به زمین وصل بشوند.

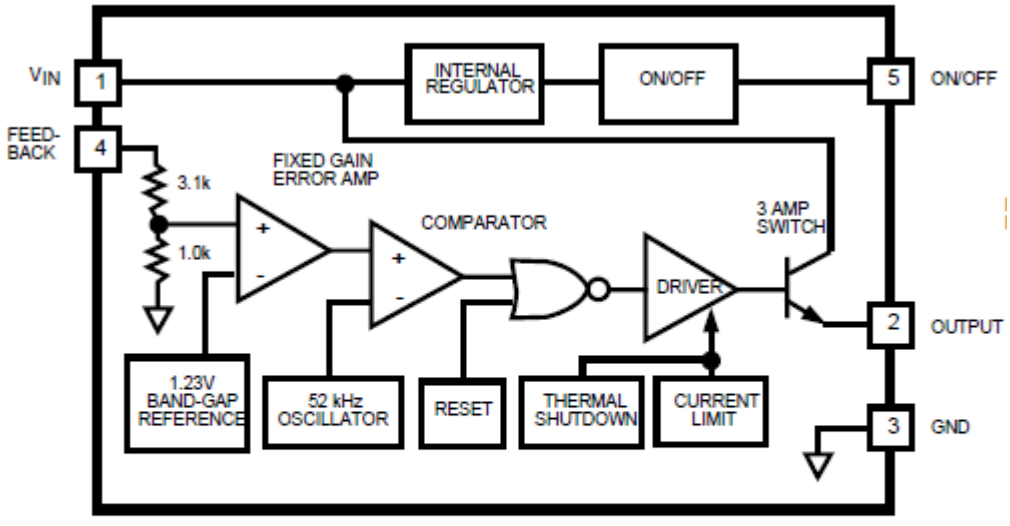


C_{IN} — 100µF, 75V Aluminum Electrolytic
C_{OUT} — 470µF, 15V Aluminum Electrolytic
D1 — Schottky, MBR360
L1 — 330µH, 415-0926 (AIE)
R1 — 1k, 0.01%
R2 — 3.065k, 0.01%
5-pin TO-220 socket—2936 (Loranger Mfg. Co.)
4-pin TO-3 socket—8112-AG7 (Augat Inc.)



C_{IN} — 100µF, 75V Aluminum Electrolytic
C_{OUT} — 330µF, 15V Aluminum Electrolytic
D1 — Schottky, 11DQ06
L1 — 100µH, 415-0926 (AIE)
5-pin TO-220 socket—2936 (Loranger Mfg. Co.)
4-pin TO-3 socket—8112-AG7 (Augat Inc.)

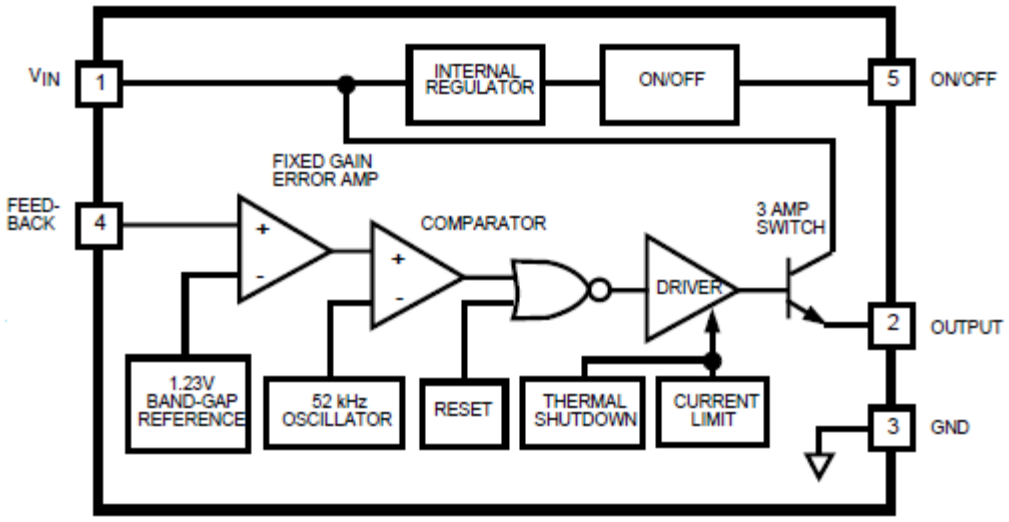
بلوک دیاگرام داخلی آی سی برای نوع ثابت:



Note: Pin numbers are for the TO-220 package

Fixed Regulator

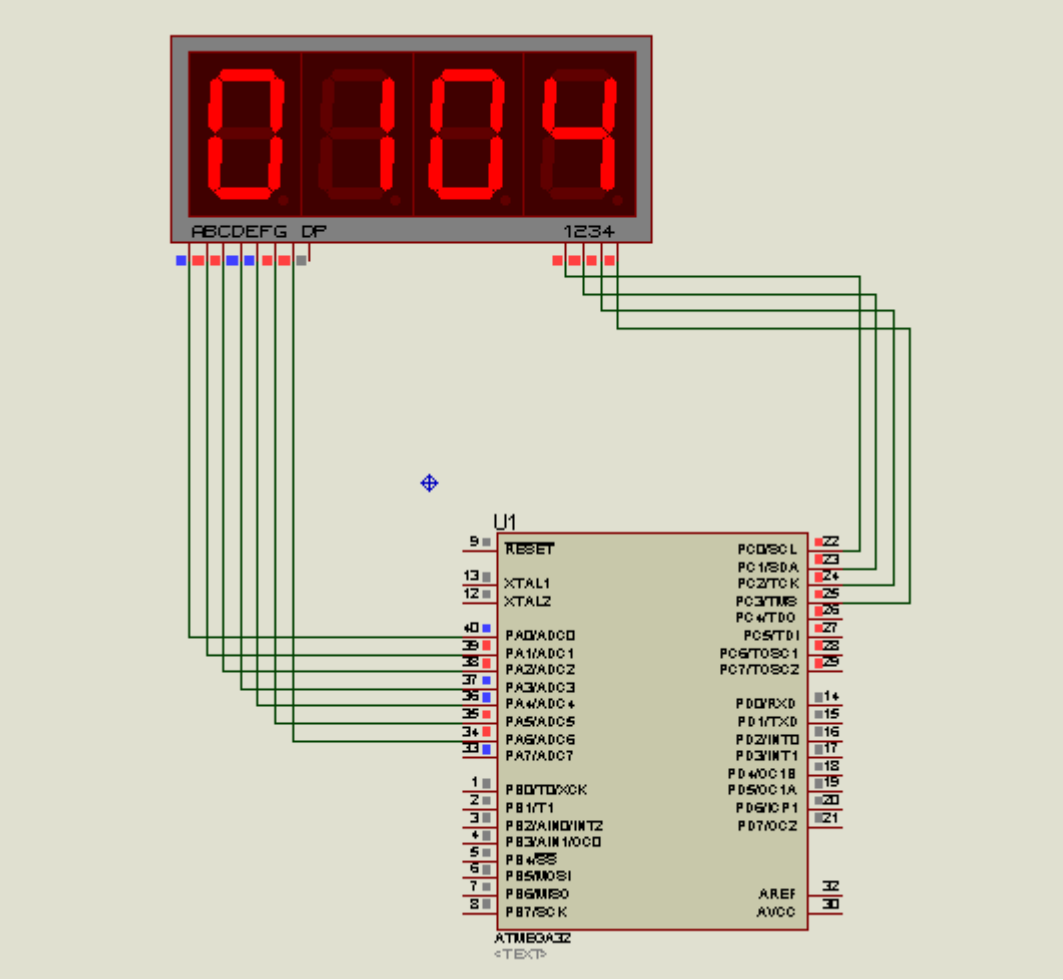
بلوک دیاگرام داخلی آی سی برای نوع متغییر



Adjustable Regulator

جلسه چهارم:

پروژه شمارنده از صفر تا 9999 و نمایش آن روی 7 سگمنت




```
#include <mega32.h>

#include <delay.h>

unsigned int i,b;

char code[11]={0x3F,0x6,0x5b,0x4f,0x66,0x6D,0x7D,0x7,0x7F,0x6f};

char a,c,e,d,f;

void main(void){

DDRA=0XFF;

DDRC=0XFF;

PORTC=0XFF;
```

اول هدر میکرو را اضافه

می کنیم.

بعد در آرایه ای کد سون

گمنت ها را ذخیره

می کنیم.

متغییر های سراسری را تعریف

می کنیم.

پرت A و C را خروجی تعریف می کنیم.

در حلقه اصلی اول به متغییر شمارنده بعد برای اینکه عدد چهار رقمی را جدا از هم روی سون سگمنت ها جای دهیم . مثلا اگر عدد 1234 باشد با تقسیم به هزار عدد 1 را می توان بدست آورد و باقیمانده این تقسیم 234 می باشد که به داخل متغییر از نوع اینتجر می ریزیم بعد با تقسیم این عدد به صد عدد 2 بدست می آید .باقیمانده این تقسیم 34 می باشد که با تقسیم این عدد به ده عدد 3 بدست و باقیمانده این نیز 4 می باشد.

بعد با روش قطع وصل سریع تر اعداد را در سون سگمنت ها نشان می دهیم.

```
while(1){
i++;
a=i/1000;
b=i%1000;
c=b/100;
d=b%100;
e=d/10;
if (i==1000){i=0;}

f=i%10;
PORTC.0=0;PORTC.1=1;PORTC.2=1;PORTC.3=1;
PORTA=code[a];
delay_us(500);
PORTC.0=1;PORTC.1=0;PORTC.2=1;PORTC.3=1;
PORTA=code[c];
delay_us(500);
PORTC.0=1;PORTC.1=1;PORTC.2=0;PORTC.3=1;
PORTA=code[e];
delay_us(500);
PORTC.0=1;PORTC.1=1;PORTC.2=1;PORTC.3=0;
PORTA=code[f];
delay_ms(1);
PORTC.3=1;
}
```

جلسه پنجم:

راه اندازی ال سی دی کارکتری

LCD کاراکتری

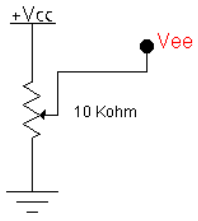
برای اتصال LCD به میکرو ابتدا باید با پایه ها و شیوه ی عملکرد آن آشنا شویم. در این مبحث با LCD 2×16 کار می کنیم بقیه ی نمایشگرهای کاراکتری نیز مشابه این نمایشگر می باشند.

در جدول زیر شماره پایه، نام پایه و عملکرد آن آمده است:

شماره پایه	نام پایه	عملکرد
1	Vss	زمین ، GND
2	Vcc	تغذیه مثبت ، 5v
3	Vee	تنظیم نور کاراکترها (کنتراست)
4	RS	اگر RS=0 باشد مقدار ورودی به عنوان یک دستور می باشد اما اگر RS=1 باشد مقدار ورودی یک داده برای چاپ شدن است
5	$R/\bar{W}$	اگر بخواهیم در LCD بنویسیم این پایه باید صفر باشد و اگر بخواهیم از LCD مقداری را بخوانیم باید آن را یک کنیم
6	E	پس از انجام هر عملیات ارسال یا دریافت باید پایه ی E را یکبار صفر و یکبار یک کنیم تا اطلاعات ثبت شوند
7 – 14	$DB_0 \rightarrow DB_7$	مسیر ورود و خروج اطلاعات LCD
15	Anod	تغذیه ی مثبت چراغ LCD
16	Katod	تغذیه ی منفی چراغ LCD

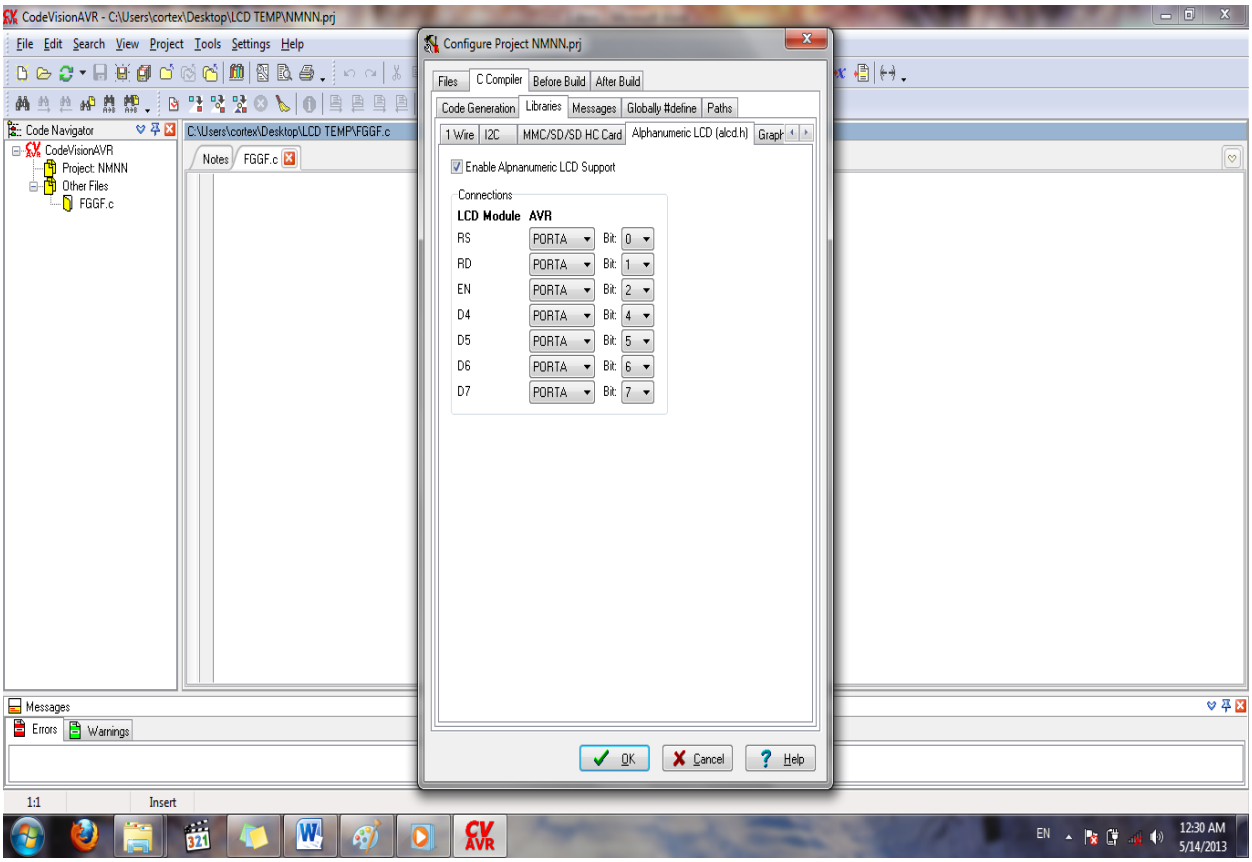
پایه ی Vee می تواند به صورت زیر برای کم و زیاد کردن نور نوشته ها به کار رود:

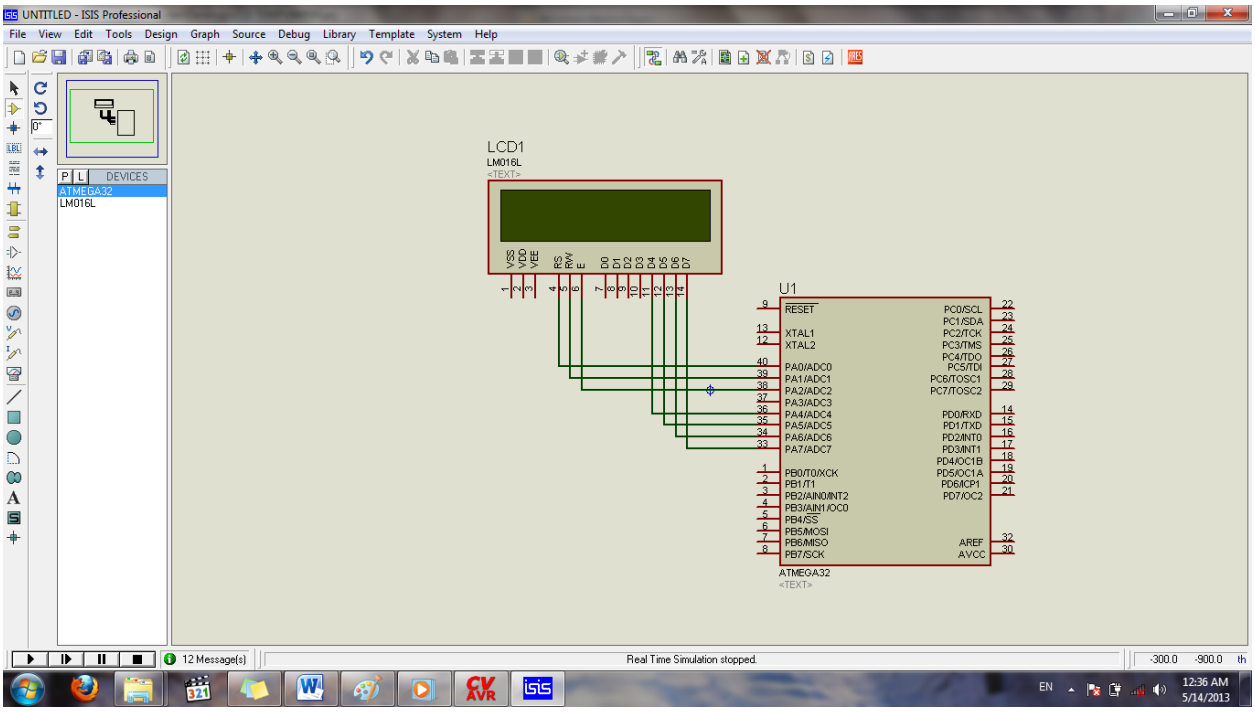
(اگر سر فلش به طرف منفی باشد، بیشترین نور، و اگر به طرف مثبت باشد، کمترین نور را خواهیم داشت.)

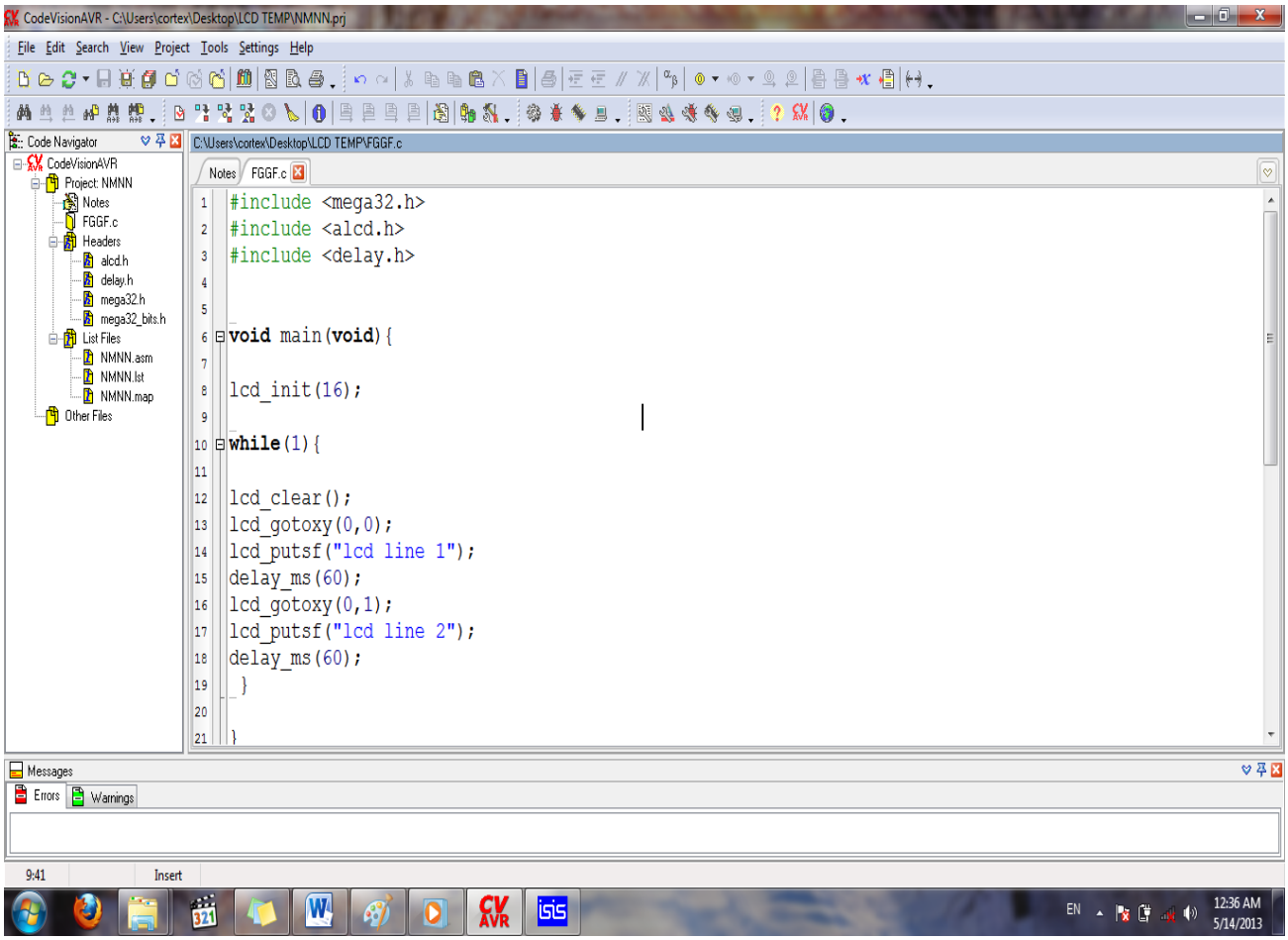


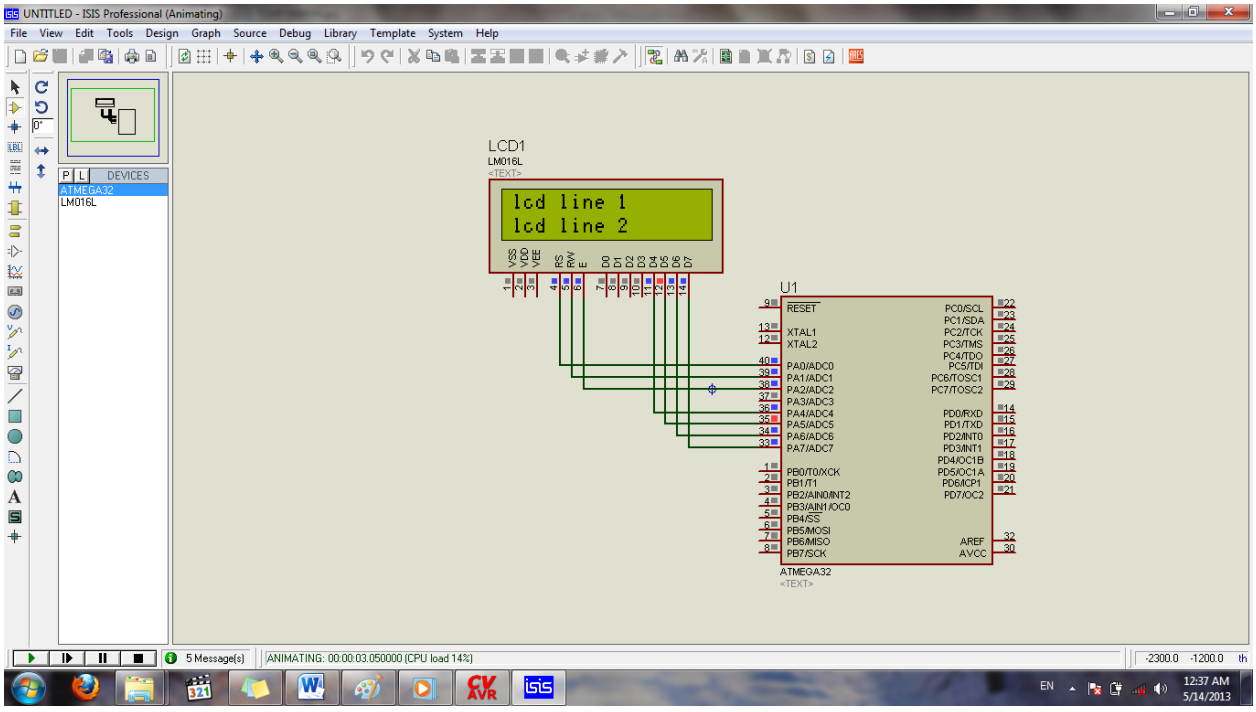
دستور	عملکرد
1	پاک کردن LCD
6	نوشتن در LCD از چپ به راست
E	LCD و مکان نما روشن شود
C	LCD روشن ولی مکان نما خاموش باشد
38	فرمت LCD
18	شیفت به چپ
1C	شیفت به راست

راه اندازی ال سی دی با استفاده از کتابخانه کدویژن


CodeVisionAVR - C:\Users\cortex\Desktop\LCD TEMP\NMNN.prj
File Edit Search View Project Tools Settings Help
Configure Project NMNN.prj
Files
C Compiler
Before Build
After Build
Code Generation
Libraries
Messages
Globally #define
Paths
1 Wire
I2C
MMC/SD/SD HC Card
Alphanumeric LCD (alcd.h)
Enable Alpnanumeric LCD Support
Connections
LCD Module
AVR
RS
RD
EN
D4
D5
D6
D7
PORTA
Bit:
OK
Cancel
Help
Code Navigator
CodeVisionAVR
Project: NMNN
Other Files
FGGF.c
Notes
Messages
Errors
Warnings
1:1
Insert
12:30 AM
5/14/2013


UNTITLED - ISIS Professional
File View Edit Tools Design Graph Source Debug Library Template System Help
DEVICES
ATMEGA32
LM016L
LCD1
LM016L
VSS VDD VEE RS R/W E D0 D1 D2 D3 D4 D5 D6 D7
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA32
12 Message(s)
Real Time Simulation stopped.
12:36 AM
5/14/2013


CodeVisionAVR - C:\Users\cortex\Desktop\LCD TEMP\NMNN.prj
File Edit Search View Project Tools Settings Help
Code Navigator
CodeVisionAVR
Project: NMNN
Notes
FGGF.c
Headers
alcd.h
delay.h
mega32.h
mega32_bits.h
List Files
NMNN.asm
NMNN.lst
NMNN.map
Other Files
C:\Users\cortex\Desktop\LCD TEMP\FGGF.c
Notes
FGGF.c
#include <mega32.h>
#include <alcd.h>
#include <delay.h>
void main(void){
lcd_init(16);
while(1){
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("lcd line 1");
delay_ms(60);
lcd_gotoxy(0,1);
lcd_putsf("lcd line 2");
delay_ms(60);
}
}
Messages
Errors
Warnings
9:41
Insert
EN
12:36 AM
5/14/2013


UNTITLED - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
DEVICES
ATMEGA32
LM016L
LCD1
LM016L
lcd line 1
lcd line 2
U1
ATMEGA32
5 Message(s)
ANIMATING: 00:00:03.050000 (CPU load 14%)
EN
12:37 AM
5/14/2013

توابع موجود در کتابخانه alcd کد ویژن :

Lcd_init(16);

برای راه اندازی اولیه ال سی دی 16 تعداد ستون هاست

Lcd_clear();

برای پاک کردن ال سی دی

Lcd_gotoxy(x,y);

با این تابع موقعیت مکان نمای ال سی دی تعیین می شود.

Lcd_puts(buffer);

با این تابع متغییر بافر از اس رم را روی ال سی
دی نشان می دهد.

Lcd_putsf(“jahandideh”);

این تابع کاراکتربین “ ” را روی ال سی دی قرار می دهد.

جلسه ششم:

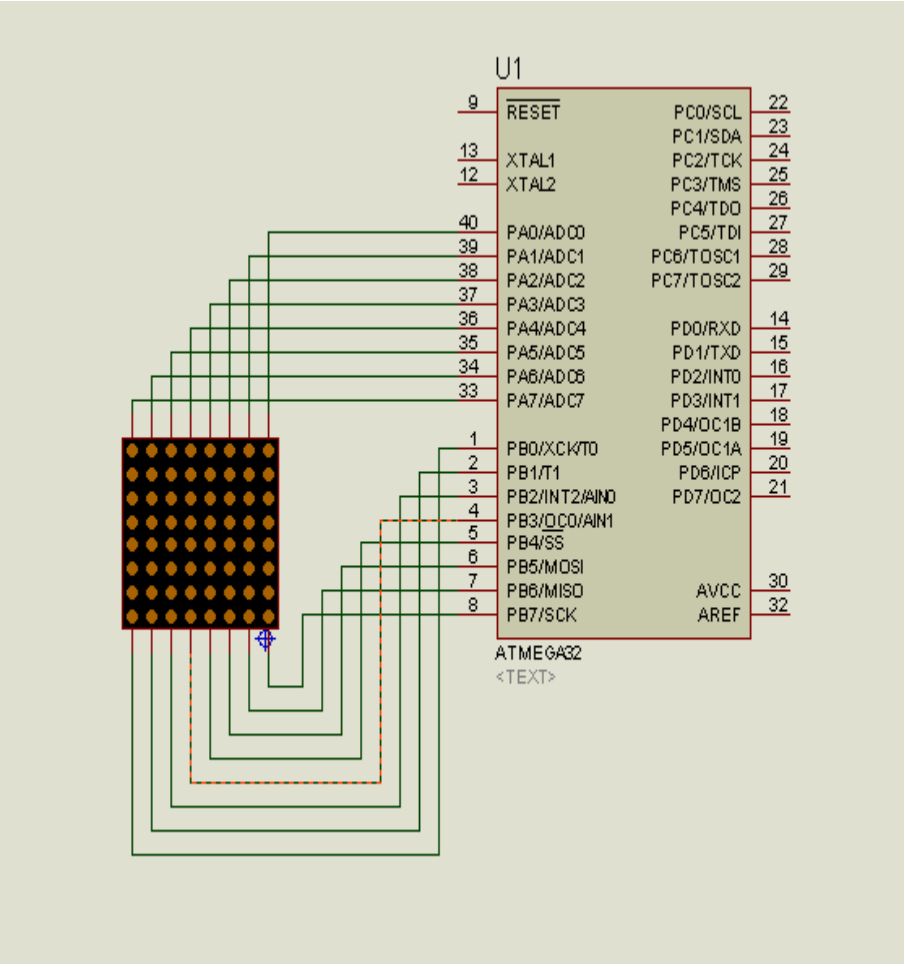
تبدیل کد اسکی به عدد معمولی:اگر از کد اسکی مقدار عدد 48 را کم کنیم،عدد مورد نظر به دست می آید.

برنامه چهار سون سگمنت به روش دیگر:

```
#include <mega32.h>
#include <delay.h>
Unsigned int count=5624;
Unsigned char font[10]{7segmeng codes};
Unsigned char ram[5];
Void main(void){
Unsigned char I,k;
DDRA=0XFF;
DDRB=0X0F;
While(1){
Sprint(ram,”%04d”,count);
K=8;
For(i=3;i>=0;i--){
PORTA=font[ram[i]-0x30];
PORTB=K;
Delay_ms(5);
PORTB=0;
K/=2;
```

جلسه هفتم:

پروژه نمایش حروف روی دات ماتریس



دات ماتریس ها از مجموعه ای از ال ای دی ها تشکیل شده اند که بصورت ماتریسی به هم وصل شده اند بهمین دلیل دات ماتریس اسم رفته اند.

برای روشن کردن هر ال ای دی باید سطر و ستون اون ال ای دی را پیدا کنی

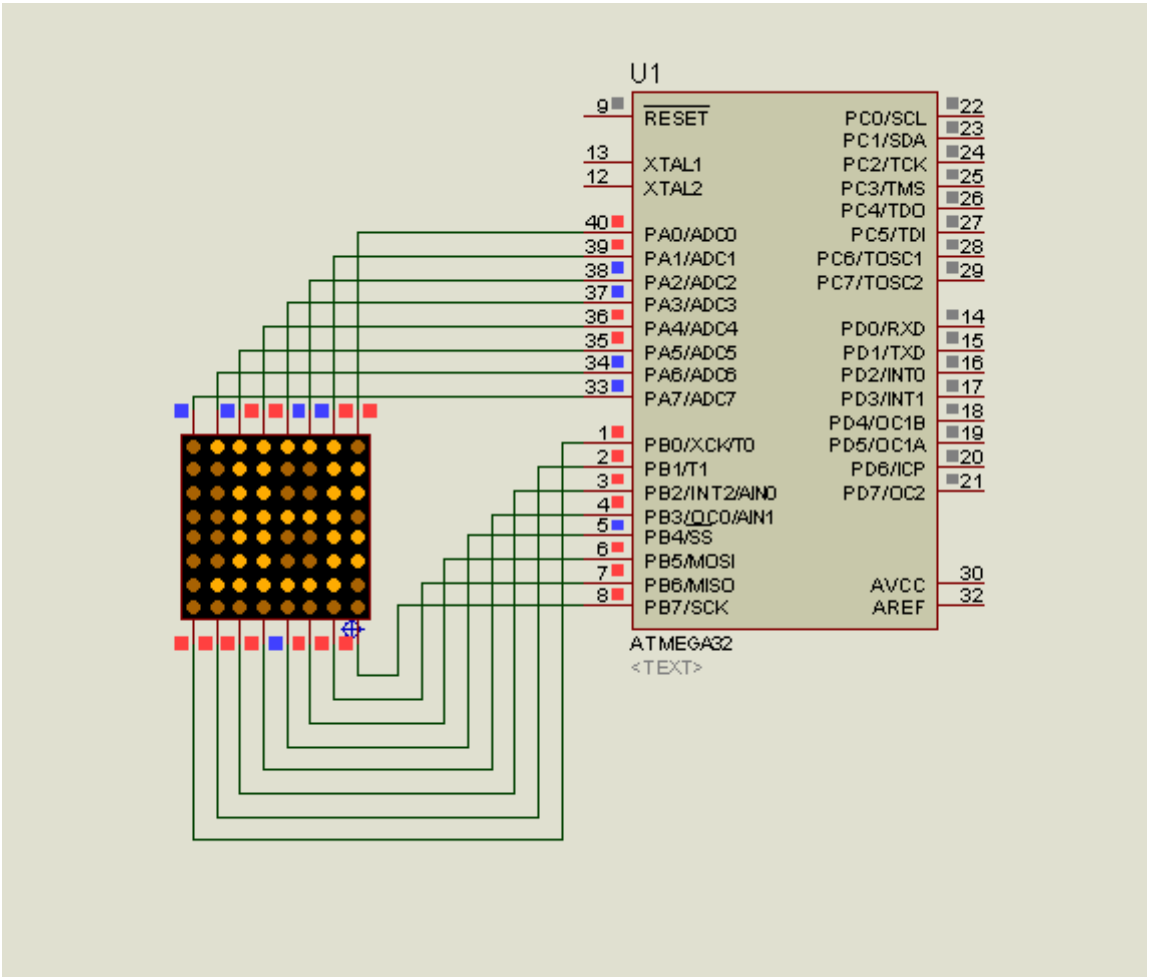
و همانند شکل زیر به سطر آن ولتاژ وی سی سی و ستون آن را به زمین وصل می کنیم.

طبق جدول گلایف بالا هر سطر با توجه به

شکل یا حروفی که می خواهیم نشان دهیم کد مخصوص به خودش را دارد و با جاروب کردن ستون ها

با سرعت زیاد می توان هر شکل یا حرفی را روی ال سی دی نشان داد .

کد سطر ها را داخل یک آرایه می ریزیم و با سرعت داخل یک حلقه ستون ها را عوض می کنیم در نتیجه یک حرف بصورت پایدار روی ال ای دی ها نشان داده می شود.



Hex
00
7E
81
A5
81
81
BD
99
81
81
7E
00

Hex
00
10
38
6C
C6
C6
FE
C6
C6
C6
C6
00

جدول گلایف نمونه برای کاراکتر A و صورتک خندان

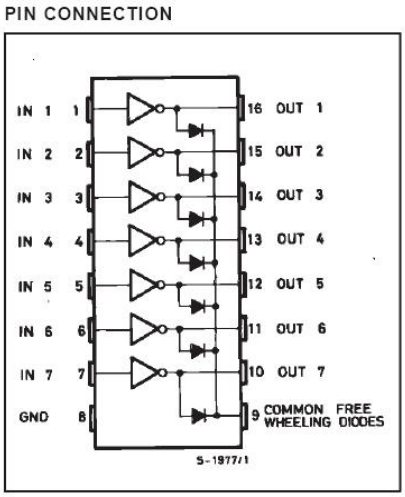
```
#include <mega32.h>

#include <delay.h>


unsigned char k;

unsigned char arr[8]={0x7E, 0x33, 0x33, 0x3E, 0x33, 0x33, 0x7E, 0x00};


void main(void)

{


PORTA=0xFF;

DDRA=0xFF;


PORTB=0xFF;

DDRB=0xFF;


while (1)

    {

        for(k=0;k<=7;k++)

        {

            PORTA=arr[k];

            PORTB&=~(1<<k);

            delay_us(100);

            PORTB=0xFF;

        }

    }

}
```

جلسه هشتم:

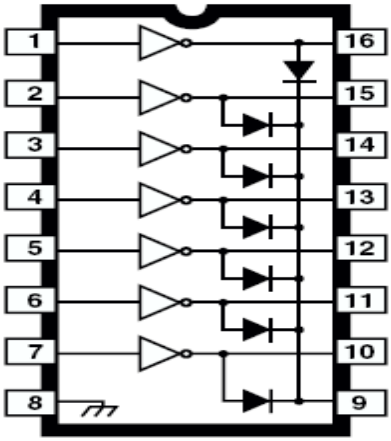
درایور مناسب برای جریان های نه چندان زیاد

ULN2003

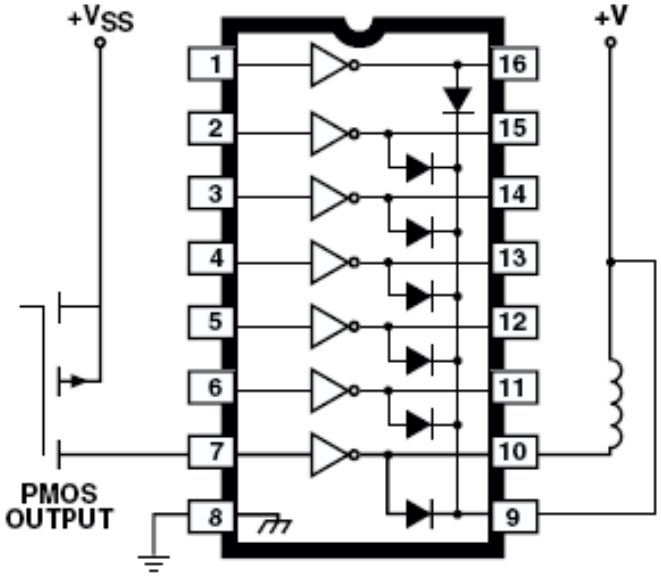
این آي سي يه درايور موتور با 7 كانال می باشد كه هر كانال ميتواند تا 600mA را عبور دهد. در تصوير زير Pin out اين آي سي پر كاربرد آورده شده است. يكي از كاربردهاي اين آي سي به عنوان Driver در مدارهاي كنترل موتور می باشد و بيشتر براي كنترل استپ موتورهاي كوچك مورد استفاده قرار مي گيرد.



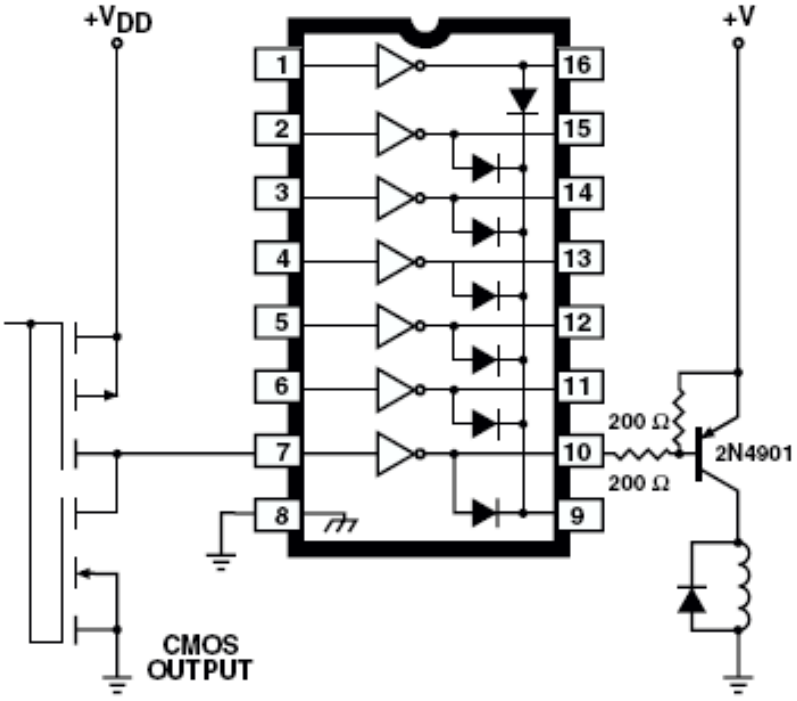
با توجه به اينكه جريان ورودي هر كانال در حالت يك منطقي حدود 25mA است، اگر اين آي سي رو مستقيماً به يك ميكروكنترلر با جريان خروجي كمتر از 25mA مثل AT89C51 وصل كنيم بعد از چند دقيقه ميكرو Reset مي شود. بهترين راه حل برای حل اين مشكل اين است كه از يك آي سي بافر مثل 74HC244 در مسير اتصال ميكرو به ULN2003 استفاده شود.



TYPICAL APPLICATIONS



Dwg. No. A-9652



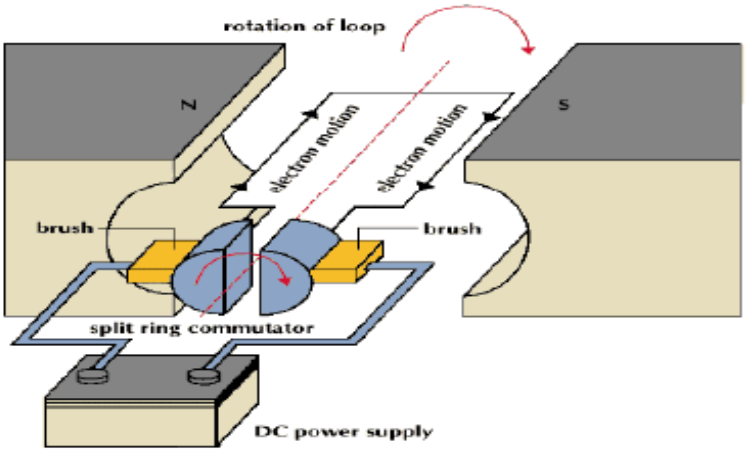
Dwg. No. A-9654A

جلسه دهم:

موتورهای الکتریکی و نحوه راه اندازی و کنترل آنها

● موتور:

یک موتور الکتریکی ، الکتریسیته را به حرکت مکانیکی تبدیل می‌کند. وقتی که یک ماده حامل جریان الکتریسیته تحت اثر یک میدان مغناطیسی قرار می‌گیرد، نیرویی بر روی آن ماده از سوی میدان اعمال می‌شود. در یک موتور استوانه‌ای ، **روتور** به علت گشتاوری که ناشی از نیرویی است که به فاصله‌ای معین از محور روتور به روتور اعمال می‌شود، می‌گردد.



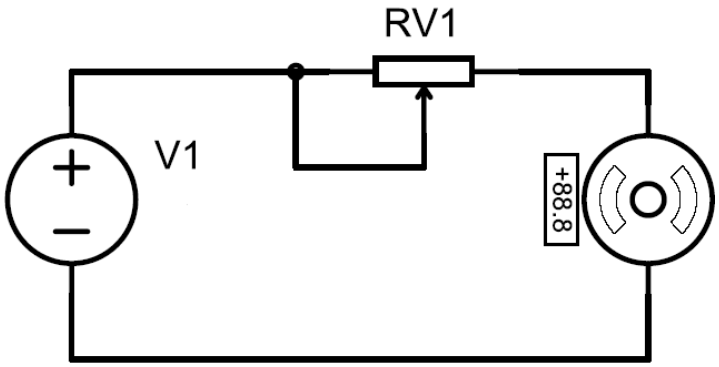
موتور کلاسیک DC دارای آرمیچری از آهنربای الکتریکی است. یک سوییچ گردشی به نام کموتاتور جهت جریان الکتریکی را در هر سیکل دو بار برعکس می کند تا در آرمیچر جریان یابد و آهنرباهای الکتریکی، آهنربای دائمی را در بیرون موتور جذب و دفع کنند. سرعت موتور DC به مجموعه ای از ولتاژ و جریان عبوری از سیم پیچهای موتور و بار موتور یا گشتاور ترمزی ، بستگی دارد.

سرعت موتور DC وابسته به ولتاژ و گشتاور آن وابسته به جریان است. معمولاً سرعت توسط ولتاژ متغیر یا عبور جریان و با استفاده از تپها (نوعی کلید تغییر دهنده وضعیت سیم پیچ) در سیم پیچی موتور یا با داشتن یک منبع ولتاژ متغیر ، کنترل می‌شود. بدلیل اینکه این نوع از موتور می‌تواند در سرعتهای پایین گشتاوری زیاد ایجاد کند، معمولاً از آن در کاربردهای ترکشن (کششی) نظیر لکوموتیوها استفاده می‌کنند .

اما به هرحال در طراحی کلاسیک محدودیتهای متعددی وجود دارد که بسیاری از این محدودیتها ناشی از نیاز به جاروبکهایی برای اتصال به کموتاتور است. سایش جاروبکها و کموتاتور ، ایجاد اصطکاک می‌کند و هر چه که سرعت موتور بالاتر باشد، جاروبکها می‌بایست محکمتر فشار داده شوند تا اتصال خوبی را برقرار کنند. نه تنها این اصطکاک منجر به سر و صدای موتور می‌شود بلکه این امر یک محدودیت بالاتری را روی سرعت ایجاد می‌کند و به این معنی است که جاروبکها نهایتاً از بین رفته نیاز به تعویض پیدا می‌کنند .

2-1 كنترل دور با يك رئوستا

ساده ترين روش براي كنترل يك موتور dc قرار دارن يك مقاومت متغير سر راه تغذيه (سري) جهت كنترل جريان آرميچر مي باشد(شكل1-1)، اين روش براي موتور هاي كوچك و جريان پايين عملي است ولي در جريان هاي بالا دو عيب اساسي آن بروز مي كند:



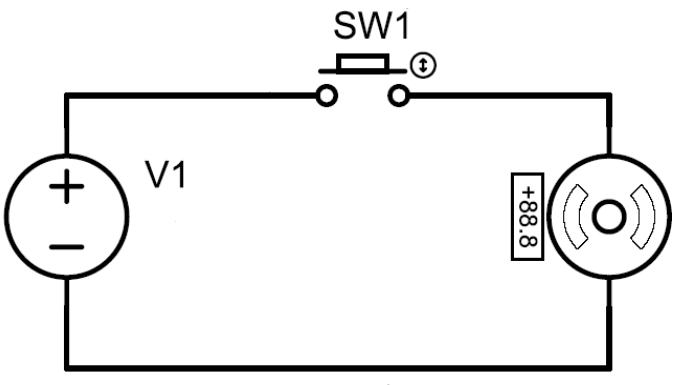
شكل1-1

1- جاري شدن جريان در رئوستا و توليد گرما به علت اتلاف توان در رئوستا.چون رئوستا هاي موجود از توان پاييني برخوردارند ودر صورت امكان ابعاد آن بزرگ مي شود،ازاين روش در جريان هاي بالا استفاده نمي شود.
2- چون موتور (بار سلفي) به عنوان بار متغير عمل مي كند، مدار ناپايدار است پس روش بالا نمي تواند براي موتور هاي بزرگ (جريان بالا)مفيد باشد.

كنترل دور با پالس PWM

يك روش مفيد براي كنترل دور موتورها قطع و وصل سريع تغذيه ي آن است كه در اين روش متناسب با زمان وصل تغذيه به زمان قطع آن ،تواني به موتور اعمال مي شودكه باعث چرخش موتور مي گردد.در واقع با اين ترفند يك پالس duty cycle توليد مي شود كه پس از اعمال آن به موتور؛ كه داراي يك مقاومت داخلي و يك خود القا است كه نقش يك فيلترپايين گذر رادارند،بعد يكسو سازي پالس متوسط آن را به موتور مي دهد،موتور نيز متناسب با آن باسرعتي ثابت مي چرخد.

براي عملي ساختن روش مذكور يك روش مفيد قراردادن يك كليد تك پل سرراه تغذيه وقطع و وصل سريع آن است(شكل2-1).در صورتي كه سرعت قطع و وصل كليد زياد باشد در چرخيدن موتور احساس نمي شود اما اين روش عملاً بعيد به نظر مي رسد و بايد به دنبال روشي الكترونيكي براي حل اين مسئله گشت.

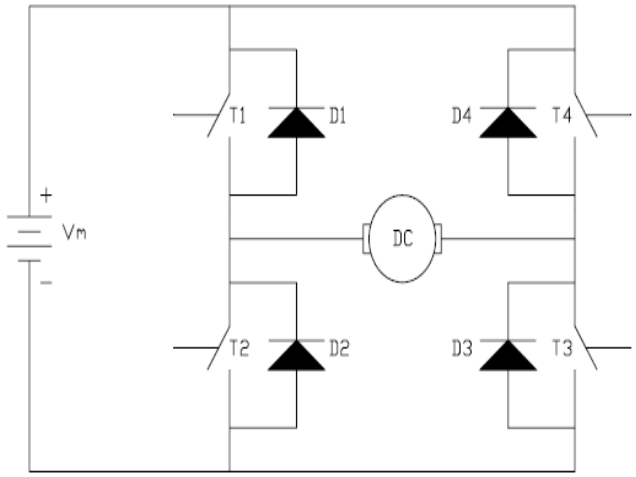


شكل2-1

4- 1 کنترل دور وجهت موتورتوسط پل H

کنترل جهت موتور DC :

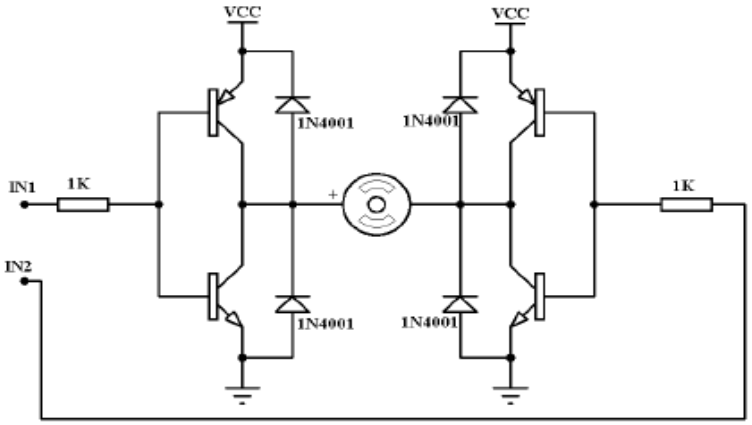
مهمترین ویژگی موتور هایDC این است که جهت چرخش آنها با تغییر جهت جریان عبوری از موتور تغییر می کند. به عبارتی اگر پلاریته ولتاژ اعمالی به موتور تغییر کند جهت چرخش موتور نیز تغییر پیدا می کند. شکل زیر این مطلب را به خوبی نشان می دهد



موتور	T4	T3	T2	T1
راستگرد	قطع	وصل	قطع	وصل
چپگرد	وصل	قطع	وصل	قطع
ترمز	قطع	وصل	وصل	قطع
ترمز	وصل	قطع	قطع	وصل

5-1 پل H با ترانزیستورهای BJT

مدار شکل زیر یک نمونه مدار معروف به پل H است که از چهار ترانزیستور برای راه اندازی،کنترل جهت و ترمز کردن موتور استفاده شده است.



شکل 4-1

عملکرد موتور	in2	in1
ترمز	0	0
مستقیم	1	0
معکوس	0	1
ترمز	1	1

ديود های استفاده شده در بر روی ترانزیستور ها دیود های هرزگرد می باشند که برای محافظت ترانزیستور در برابر خاصیت القایی موتور بکار می روند.

مي توان براي كنترل دور موتور ها در اين مدار به جاي اعمال 1منطقي (حداكثرسرعت)، به ورودي مورد نظر پالس pwm اعمال كردوسرعت موتور رانيز كنترل كرد.

در مدار فوق به دليل اينكه از ترانزيستور هاي BJT استفاده شده است كه اغلب از توان بالايي برخوردار نمي

باشند در جريان هاي بالا باعث داغ شدن ترانزيستورها ودر دراز مدت باعث سوختن ترانزيستورها مي شود. درصورت استفاده از Heat sinck براي انتقال حرارت به محيط يا استفاده از زوج دارلينگتون كه از توان بالاتري برخوردارند مي توان اين مشكل را تا حدودي حل كرد،امادر صنعت كه از موتور هاي با توان بالا استفاده مي شود مدارهاي فوق در اغلب موارد بامشكل برمي خورند.

1 استفاده از IGBT

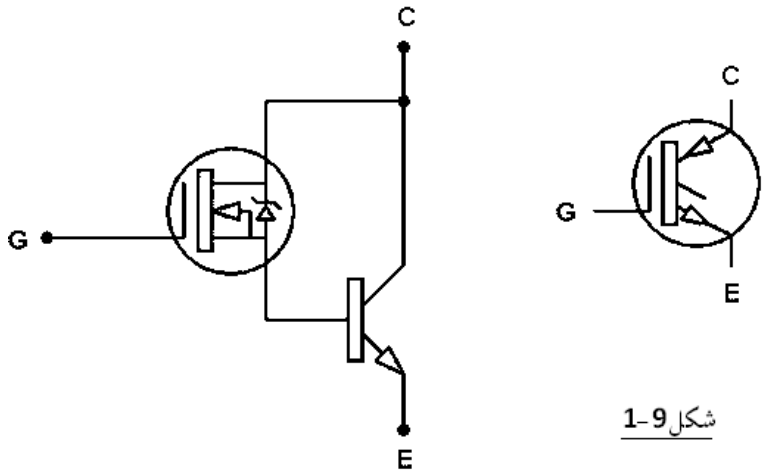
درترانزيستورهاي **bjt** جريان عبوري ازكلكتور- اميترتابع جريان بيس مي باشدوβبرابرجريان بيس دركلكتور جاري مي شود درنتيجه براي عبوريك جريان مثلا **10A** ازترانزيستورجريان بيس بايد حداقل**100ma** باشد.

توسط ترانزيستور هاي دارلينگتون تا حدودي مي توان مسئله ي فوق را حل كرد چون β دراين ترانزيستورها بزرگ مي باشد،مثلاً براي عبور **IC=10A** به **IB=1mA**نيازاست.(با فرض **10000=** β)

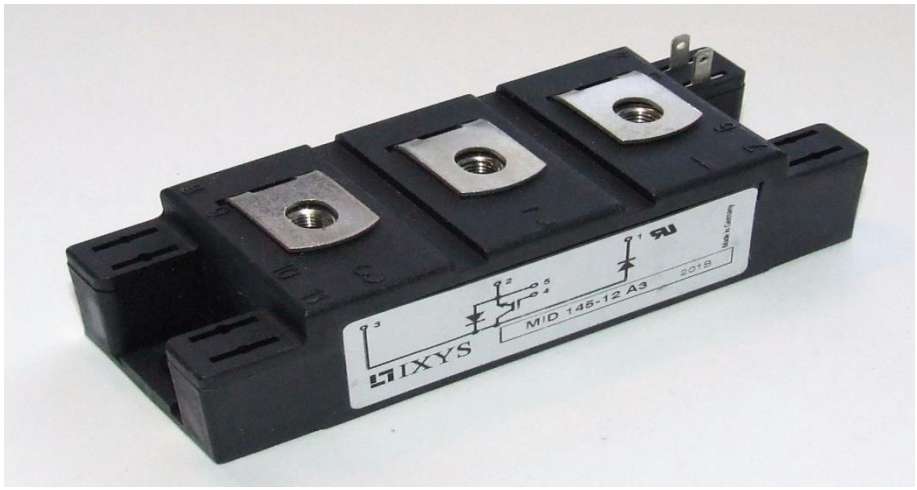
Mosfetها المان هايي هستند كه در آنها **Id** توسط **VGS** كنترل مي شود، اين ترانزيستور ها از نوع اثرميدان مي باشند ودر مدارهاي قدرت به وفور از آنها استفاده مي شود.

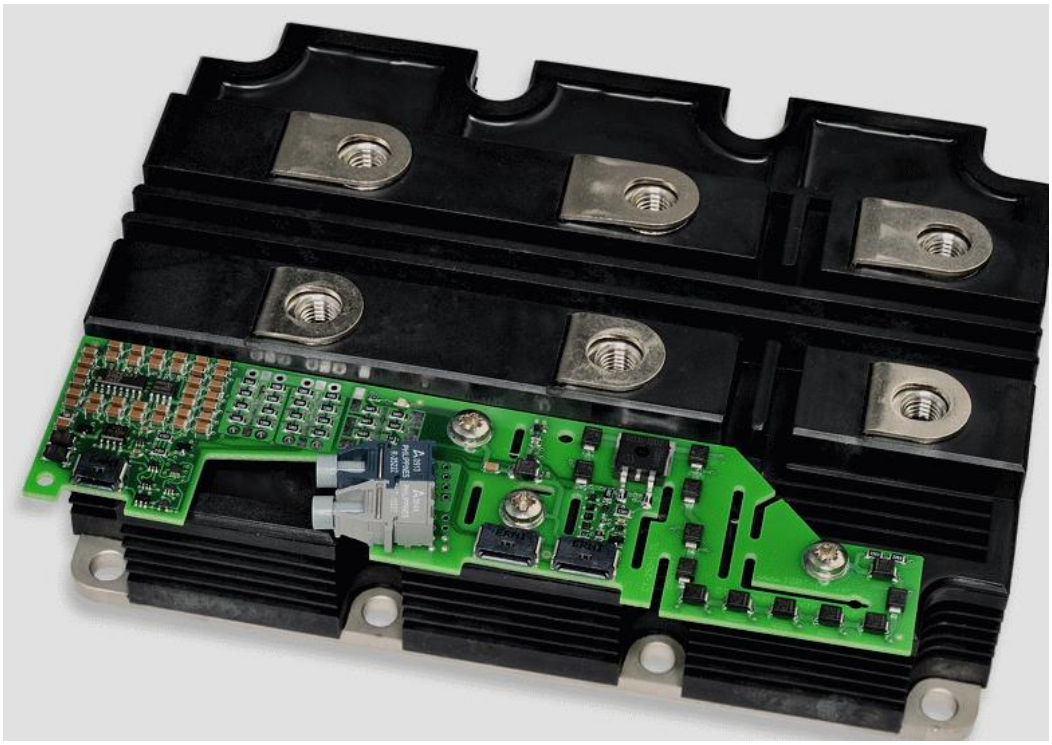
از تر كيب دو ترانزيستور فوق يك المان با ويژگي هاي منحصر به فردي به دست مي آيد كه مزاياي هر دو ترانزيستور را داراست كه به آن **IGBT** مي گويند ،از جمله مي توان موارد زير را نام برد:

- سرعت بالا
- توان بالا
- حساسيت كم به گرما
- كنترل جريان با ولتاژ
- مقاومت خروجي كم



شكل1-9


IXYS
MID 145-12 A3




SEMIKRON
SEMITRANS®M
SKM 145 GAL 123 D
04140

جلسه یازدهم:

پروژه تولید شکل موج سینوسی دو فاز با 120 درجه اختلاف فاز

برای تولید شکل موج سینوسی از روش PWM و فیلتر پایین گذر استفاده می کنیم.چون ما در این پروژه برای تولید PWM از تایمر یک استفاده می کنیم می توانیم دو شکل موج سینوسی را با این تایمر درست کنیم.در این پروژه از حالت fast pwm هشت بیتی استفاده می کنیم مقدار ماکزیمم تایمر 255 می باشد پس برای موج سینوسی جدولی را محاسبه می کنیم با فرمول زیر ساده زیر:

$$OCR1=256\sin(\alpha)$$

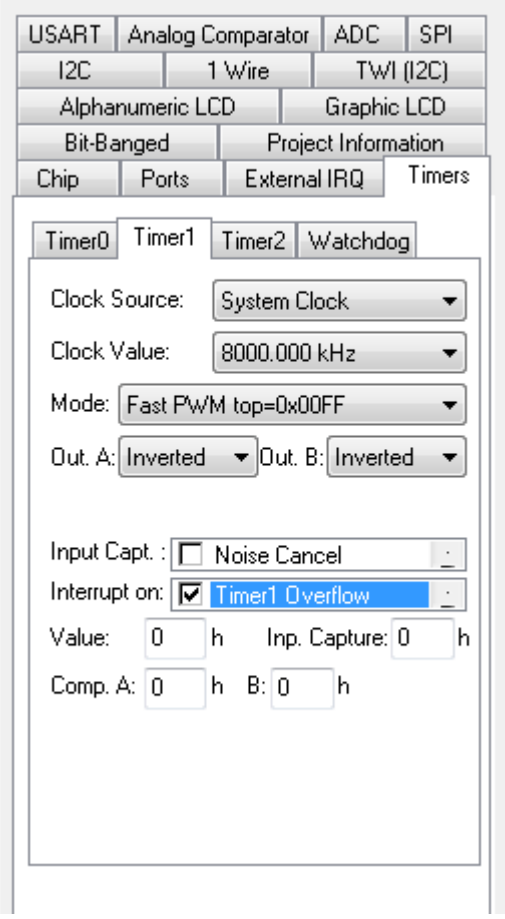
چون سرعت انجام عملیات سینوسی وقت زیادی از سی پی یو را می گیرد برای همین جدولی را بعنوان جدول سینوس داخل آرایه ای ایجاد می کنیم . چون باید شکل موج اولی با دومی 120درجه اختلاف داشته باشند باید مقدار ocrدومی 120 عدد نسبی (نسبت به این که جدول سینوس شامل چند عدد باشد) با توجه به نسبت زیر چون جدول پروژه ما 256 تا عدد دارد :

120	360
x	256

Xبرابر تقریبا 86 می شود.

```
flash char sinewave[]={
0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x02,0x03,0x03,0x04,0x05,0x06,0x07,0x08,
0x09,0x0a,0x0c,0x0d,0x0f,0x10,0x12,0x13,0x15,0x17,0x19,0x1b,0x1d,0x1f,0x21,0x23,
0x25,0x27,0x2a,0x2c,0x2e,0x31,0x33,0x36,0x38,0x3b,0x3e,0x40,0x43,0x46,0x49,0x4c,
0x4f,0x51,0x54,0x57,0x5a,0x5d,0x60,0x63,0x67,0x6a,0x6d,0x70,0x73,0x76,0x79,0x7c,
0x80,0x83,0x86,0x89,0x8c,0x8f,0x92,0x95,0x98,0x9c,0x9f,0xa2,0xa5,0xa8,0xab,0xae,
0xb0,0xb3,0xb6,0xb9,0xbc,0xbf,0xc1,0xc4,0xc7,0xc9,0xcc,0xce,0xd1,0xd3,0xd5,0xd8,
0xda,0xdc,0xde,0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xed,0xef,0xf0,0xf2,0xf3,0xf5,
0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfc,0xfc,0xfd,0xfe,0xfe,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xfe,0xfe,0xfd,0xfc,0xfc,0xfb,0xfa,0xf9,0xf8,0xf7,
0xf6,0xf5,0xf3,0xf2,0xf0,0xef,0xed,0xec,0xea,0xe8,0xe6,0xe4,0xe2,0xe0,0xde,0xdc,
0xda,0xd8,0xd5,0xd3,0xd1,0xce,0xcc,0xc9,0xc7,0xc4,0xc1,0xbf,0xbc,0xb9,0xb6,0xb3,
0xb0,0xae,0xab,0xa8,0xa5,0xa2,0x9f,0x9c,0x98,0x95,0x92,0x8f,0x8c,0x89,0x86,0x83,
0x80,0x7c,0x79,0x76,0x73,0x70,0x6d,0x6a,0x67,0x63,0x60,0x5d,0x5a,0x57,0x54,0x51,
0x4f,0x4c,0x49,0x46,0x43,0x40,0x3e,0x3b,0x38,0x36,0x33,0x31,0x2e,0x2c,0x2a,0x27,
0x25,0x23,0x21,0x1f,0x1d,0x1b,0x19,0x17,0x15,0x13,0x12,0x10,0x0f,0x0d,0x0c,0x0a,
0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x03,0x02,0x01,0x01,0x01,0x01,0x01,0x01,0x01
};
```

حال نوبت به تنظیم تایمر می رسد:




```
void main(void){

TCCR1A=0xF1;

TCCR1B=0x0A;

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x00;

OCR1BH=0x00;

OCR1BL=0x00;

DDRD=0xFF;

TIMSK=0x04;

#asm("sei")

while(1);

}
```

داخل وقفه سرریز چه اتفاقی می افتد:

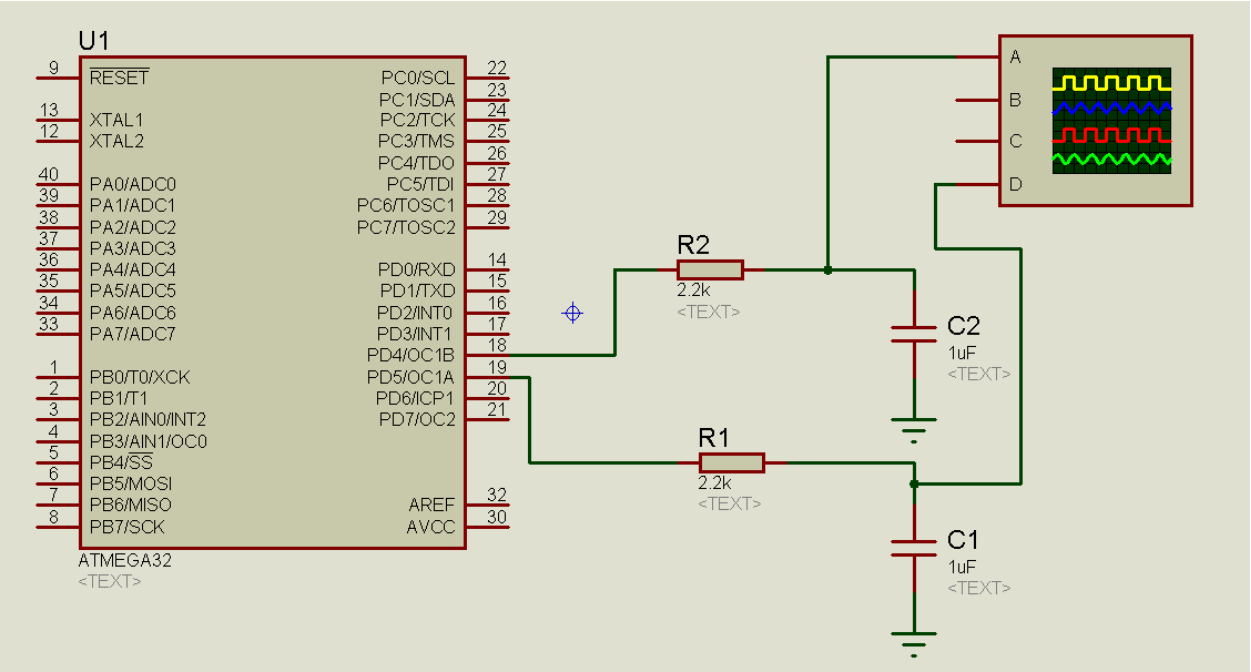
```
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
j=i+86;
i++;
OCR1B=sinewave[i];
OCR1A=sinewave[j];
}
```

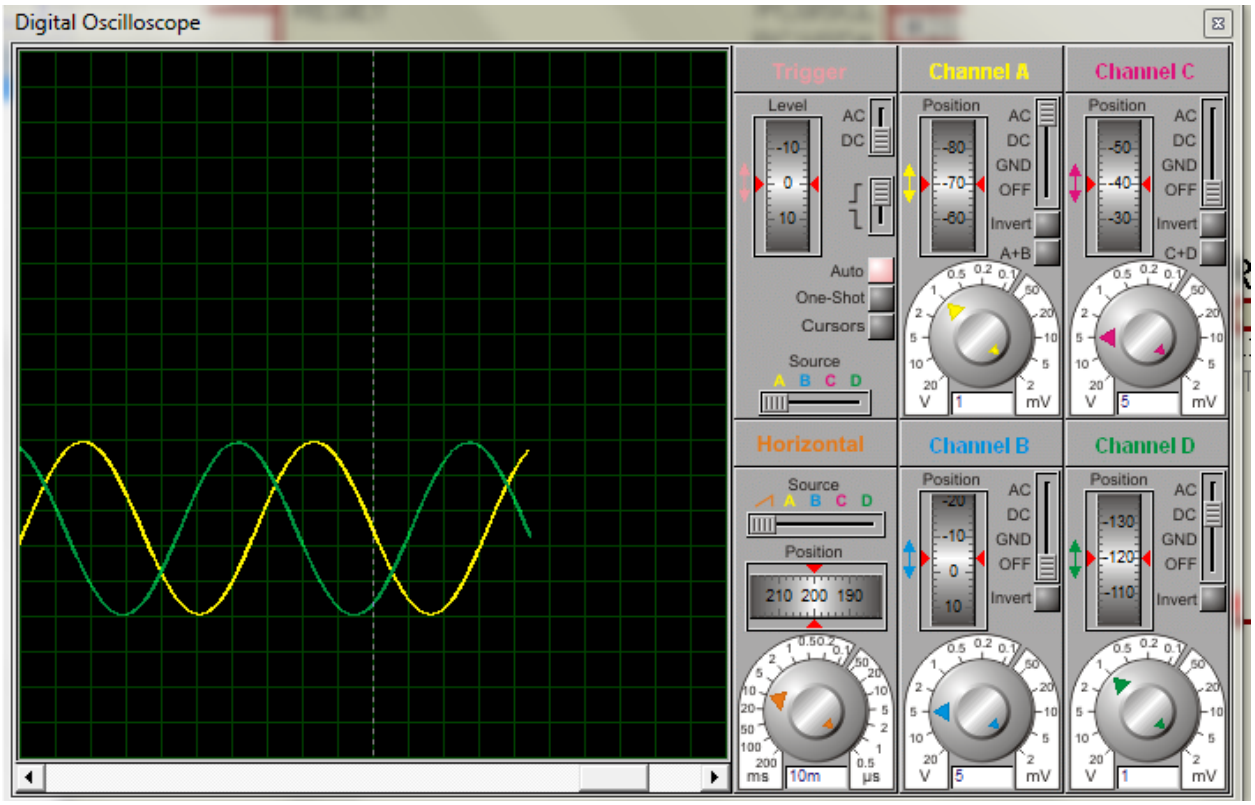
متغیرهای i و j از نوع char و سراسری می باشند.

چون رجیستر مقایسه pwmتایمر دارای بافر مضاعف می باشد.

بنابراین آپدیت مقدار ocr ها قبل از سرریز اهمیتی ندارد بنابراین باید صبر کنیم تا تایمر سرریز شود بعد مقدار جدید را در OCR قرار دهیم .

در یکی از Ocr ها مقدار خود i و در دیگری i+86 را قرار می دهیم چون خود رجیستر سرریز می شود لازم نیست شرطی برای متغییر ها بنویسیم.


U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA32
<TEXT>
R2
2.2k
<TEXT>
R1
2.2k
<TEXT>
C2
1uF
<TEXT>
C1
1uF
<TEXT>
A
B
C
D


Digital Oscilloscope
Trigger
Level
AC
DC
Auto
One-Shot
Cursors
Source
A B C D
Channel A
Position
AC
DC
GND
OFF
Invert
A+B
Channel C
C+D
Horizontal
210 200 190
Channel B
Channel D
10m
ms
µs
V
mV

پروژه تولید شکل موج سینوسی 3 فاز با 120 درجه اختلاف فاز

با اضافه کردن یک PWM دیگر به مدار دو فاز می توانیم شکل موج سینوسی سه فاز درست کنیم چون تایمر یک فقط دوتا خروجی pwm دارد بنابراین از تایمر صفر برای این کار کمک می گیریم و شکل موج سوم را با pwm تایمر صفر درست می کنیم .

اضافه شده های برنامه :

فعال کردن تایمر صفر در حالت fast pwm و با فرکانس یکسان با تایمر یک و خروجی بصورت اینورت و...

همچنین فعال سازی وقفه سرریز تایمر صفر

یک متغییر دیگر برای نگه داری درایه موج سوم

و برنامه داخل وقفه :

```
TCCR0=0x79;
TCNT0=0x00;
OCR0=0x00;
```

```
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
u=i+170;
OCR0=sinewave[u];

}
```

همچنین باید رجیستر TIMSK را برای فعال سازی هر دو وقفه برنامه ریزی کنیم.

```
TIMSK=0x05;
```

تعریف متغیر ها و ثابت ها :

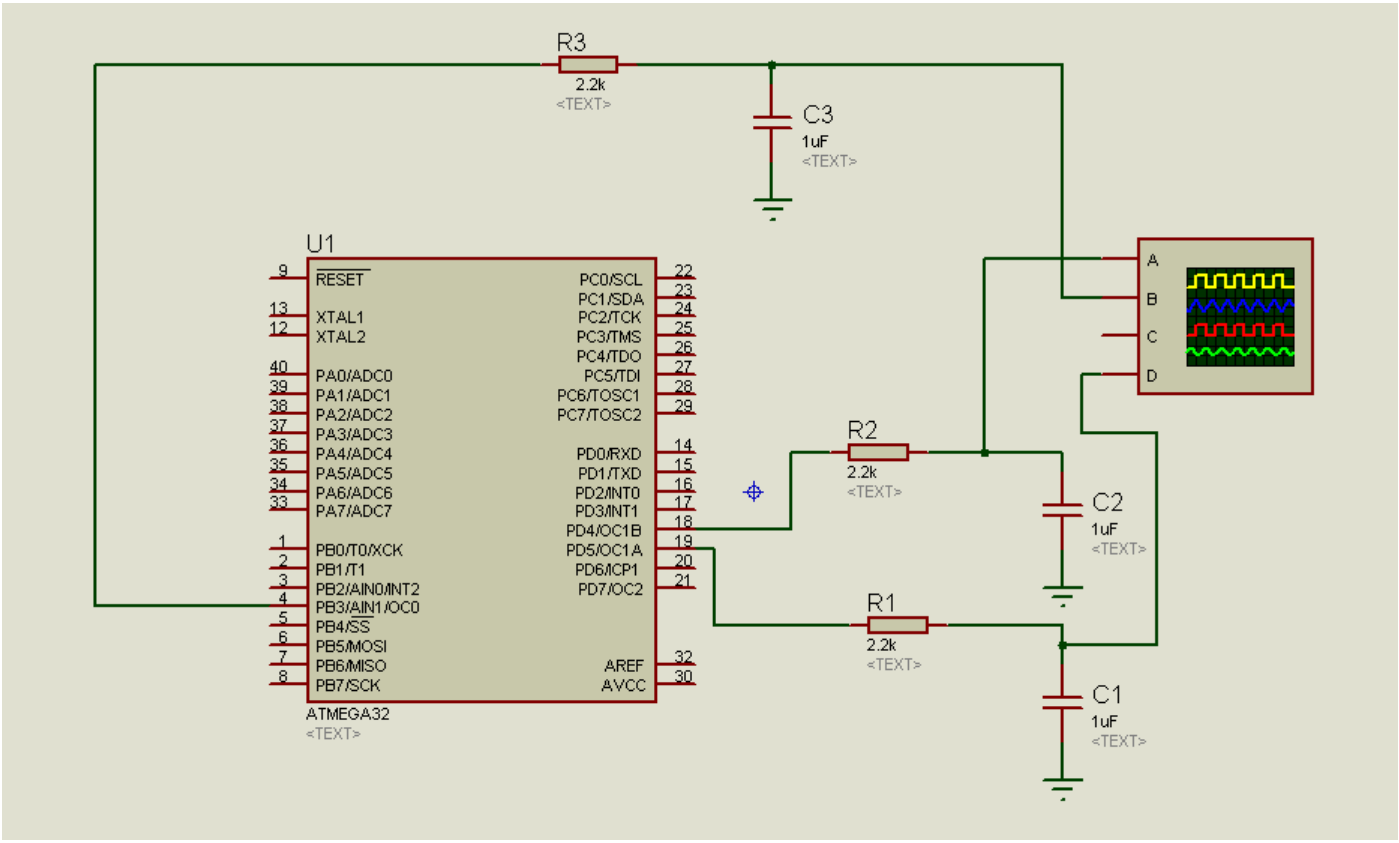
```
/*
barname:tolid moj sinoosi 3phase ba 120darage ekhtelaf faz.
nevisande:milad jahandideh
*/
#include <mega32.h>
#include <delay.h>
char i,j,u;
flash char sinewave[]={
0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x02,0x03,0x03,0x04,0x05,0x06,0x07,0x08,
0x09,0x0a,0x0c,0x0d,0x0f,0x10,0x12,0x13,0x15,0x17,0x19,0x1b,0x1d,0x1f,0x21,0x23,
0x25,0x27,0x2a,0x2c,0x2e,0x31,0x33,0x36,0x38,0x3b,0x3e,0x40,0x43,0x46,0x49,0x4c,
0x4f,0x51,0x54,0x57,0x5a,0x5d,0x60,0x63,0x67,0x6a,0x6d,0x70,0x73,0x76,0x79,0x7c,
0x80,0x83,0x86,0x89,0x8c,0x8f,0x92,0x95,0x98,0x9c,0x9f,0xa2,0xa5,0xa8,0xab,0xae,
0xb0,0xb3,0xb6,0xb9,0xbc,0xbf,0xc1,0xc4,0xc7,0xc9,0xcc,0xce,0xd1,0xd3,0xd5,0xd8,
0xda,0xdc,0xde,0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xed,0xef,0xf0,0xf2,0xf3,0xf5,
0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfc,0xfc,0xfd,0xfe,0xfe,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xfe,0xfe,0xfd,0xfc,0xfc,0xfb,0xfa,0xf9,0xf8,0xf7,
0xf6,0xf5,0xf3,0xf2,0xf0,0xef,0xed,0xec,0xea,0xe8,0xe6,0xe4,0xe2,0xe0,0xde,0xdc,
0xda,0xd8,0xd5,0xd3,0xd1,0xce,0xcc,0xc9,0xc7,0xc4,0xc1,0xbf,0xbc,0xb9,0xb6,0xb3,
0xb0,0xae,0xab,0xa8,0xa5,0xa2,0x9f,0x9c,0x98,0x95,0x92,0x8f,0x8c,0x89,0x86,0x83,
0x80,0x7c,0x79,0x76,0x73,0x70,0x6d,0x6a,0x67,0x63,0x60,0x5d,0x5a,0x57,0x54,0x51,
0x4f,0x4c,0x49,0x46,0x43,0x40,0x3e,0x3b,0x38,0x36,0x33,0x31,0x2e,0x2c,0x2a,0x27,
0x25,0x23,0x21,0x1f,0x1d,0x1b,0x19,0x17,0x15,0x13,0x12,0x10,0x0f,0x0d,0x0c,0x0a,
0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x03,0x02,0x01,0x01,0x01,0x01,0x01,0x01,0x01
};
```

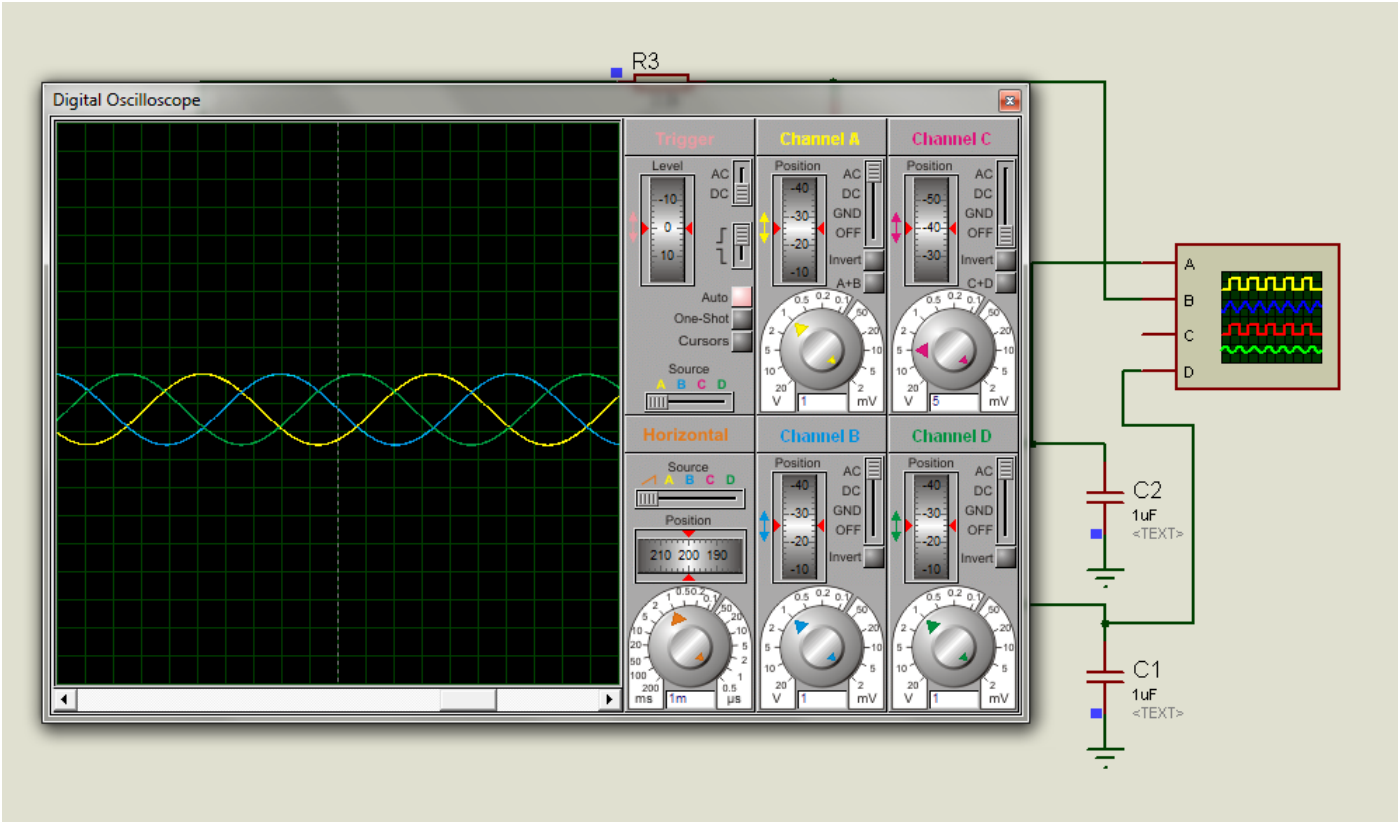
زیر برنامه داخل وقفه ها:

```
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
j=i+86;
i++;
OCR1B=sinewave[i];
OCR1A=sinewave[j];
PORTD.0=~PORTD.0;
}
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
u=i+170;
OCR0=sinewave[u];


}
```

مقدار دهی رجیسترها و فعال سازی وقفه سراسری:

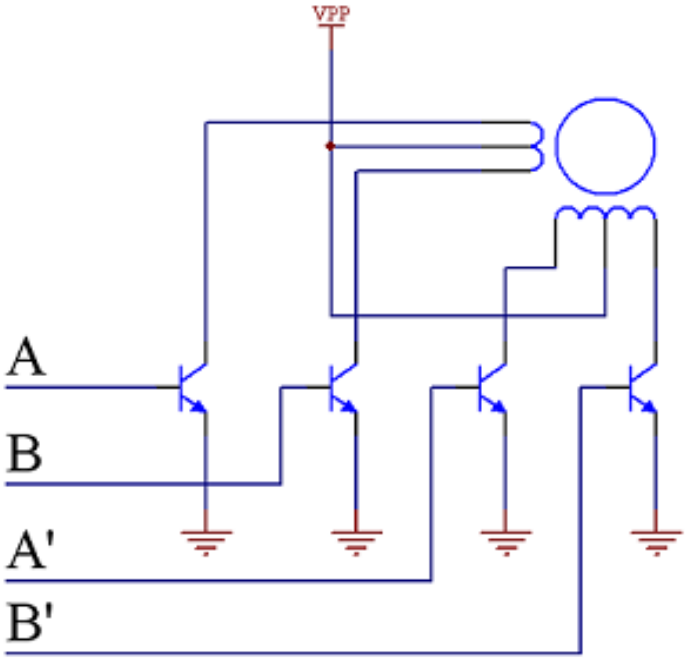
```
void main(void){



DDRB=0x08;

DDRD=0x30;


TCCR0=0x79;

TCNT0=0x00;

OCR0=0x00;

////////////////

TCCR1A=0xF1;

TCCR1B=0x09;

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x00;

OCR1BH=0x00;

OCR1BL=0x00;


TIMSK=0x05;


#asm("sei")

while(1);

}
```


R3
2.2k
<TEXT>
C3
1uF
<TEXT>
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA32
<TEXT>
R2
2.2k
<TEXT>
R1
2.2k
<TEXT>
C2
1uF
<TEXT>
C1
1uF
<TEXT>
A
B
C
D


R3
Digital Oscilloscope
Trigger
Channel A
Channel C
Horizontal
Channel B
Channel D
Level
Position
Source
Auto
One-Shot
Cursors
Invert
A+B
C+D
AC
DC
GND
OFF
C2
1uF
<TEXT>
C1
1uF
<TEXT>
A
B
C
D

پروژه راه اندازی استپ موتور با دورسنج

استپ موتور وسيله اي است كه پالس هاي الكتريكي را به حركت مكانيكي تبديل مي كند.هر استپ موتورداراي يك هسته ي متحرك مغناطيس دايمي است كه روتور يا شفت هم خوانده مي شود.استپ موتورها به صورت چهار،پنج وشش سيمه وجود دارند.سر وسط مشترك مركزي در صورتي كه زمين شود،جهت جريان را در هركدام از جفت سيم پيچ تغيير مي دهد،بنابراين تغيير قطبيت استاتور را موجب مي شود.شفت در استپ موتور به صورت پله اي حركت مي كند كه موجبات حركت آن به يك مكان دقيق رافراهم مي سازد.



A	B	A’	B’
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

اول کتابخانه های مورد نظر را اضافه می کنیم .

متغییر های لازم را تعریف می کنیم

تایمر صفر را برای ایجاد زمان فعال می کنیم.

```
#include <mega16.h>
#include <delay.h>
#include <alcd.h>
#include <stdio.h>

unsigned int count,d;
char buffer[16];
void main(void){

////////
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 7.813 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x05;
TCNT0=0x00;
OCR0=0x00;
```

```
// Timer/Counter 1 initialization
// Clock source: T1 pin Rising Edge
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x07;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
//////////
DDRA=0XFF;
TIMSK=0x01;
#asm("sei")
```

تایمر /کانتر یک را در حالت کانتر فعال می کنیم برای شمارش تعداد پالس های خروجی فتودیود.

پورت A را خروجی تعریف می کنیم.

اجازه وقفه تایمر صفر و وقفه سراسری را فعال می کنیم.

```
while(1){
   PORTA=0X01;
   delay_ms(100);
   PORTA=0X02;
   delay_ms(100);
   PORTA=0X04;
   delay_ms(100);
   PORTA=0X08;
   delay_ms(100);
   }}
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{

count++;
 if(count==445){
 d=TCNT1;
 d=d*4;
 delay_us(4);
 TCNT1=0X00;
 count=0;
  sprintf(buffer,"rpm=%u",d);
  lcd_clear();
  lcd_puts(buffer);
 }
```

داخل حلقه اصلی با ایجاد پالس های مناسب موتور را راه اندازی می کنیم.

در داخل وقفه زمان 15 ثانیه ایجاد می کنیم و تعداد شمارش ها در هر 15 ثانیه را در 4 ضربش می کنیم تا تعداد شمارش ها در یک دقیقه بدست آیند.