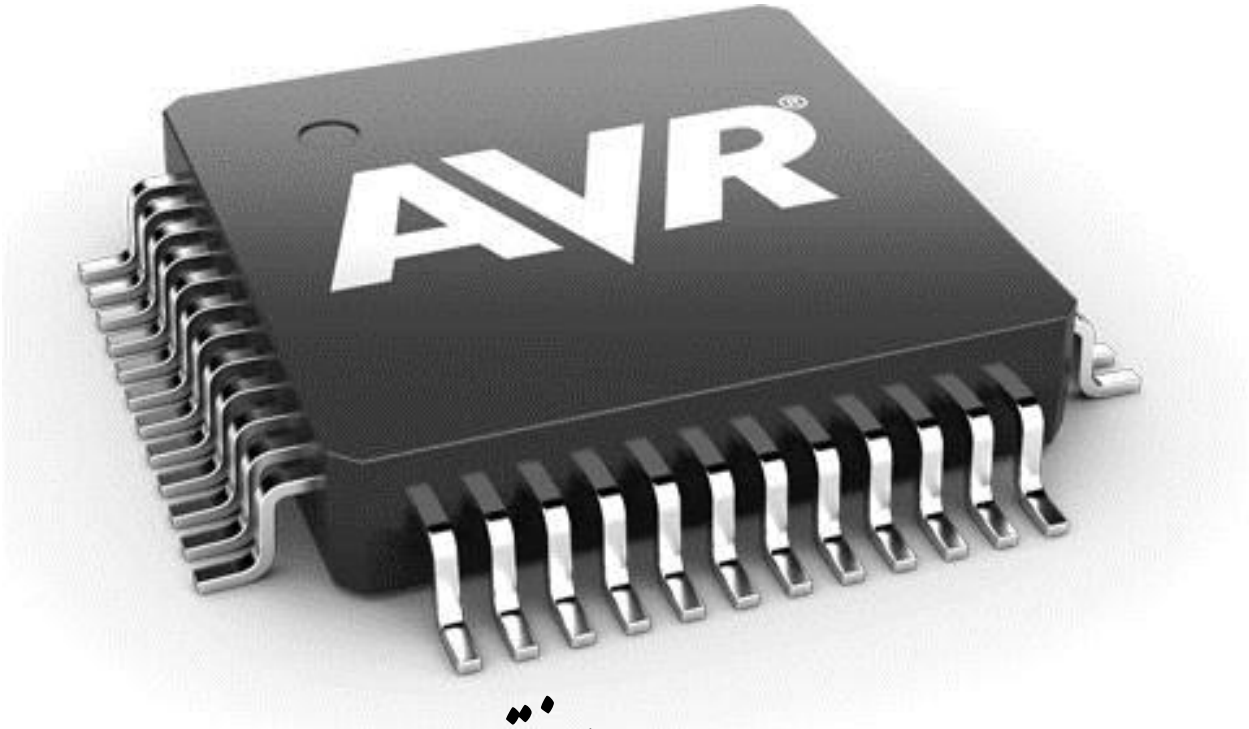
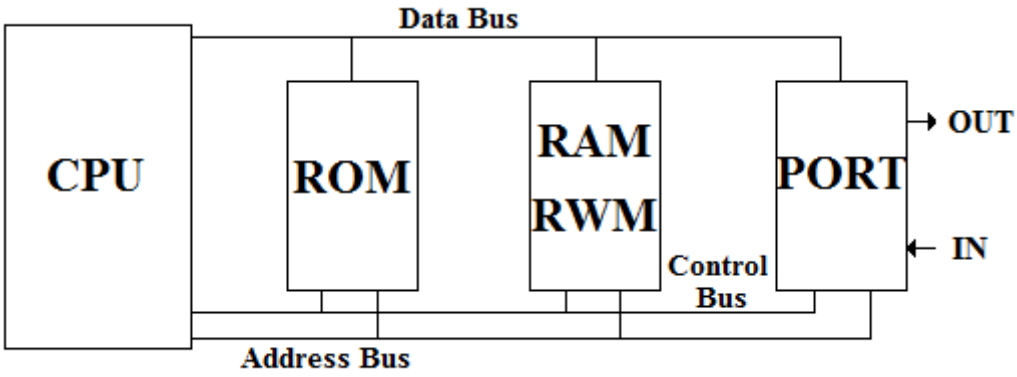
هوالفتاح العلیم

جزوه درس میکروکنترلرها



مدرس: رضا فتاحی

بلوک دیاگرام یک سیستم میکروپروسسوری µp:



CPU : Central Processing Unit یا Microprocessor

در هر سیستم یک CPU وجود دارد که کار کنترل و پردازش Data را انجام می دهد.

ROM : Read Only Memory

حافظه ای است فقط خواندنی که اطلاعات راه اندازی سیستم که به آن سیستم عامل OS نیز گفته می شود درون آن قرار دارد. (Operating System)

RAM: Read Write Memory (RWM)

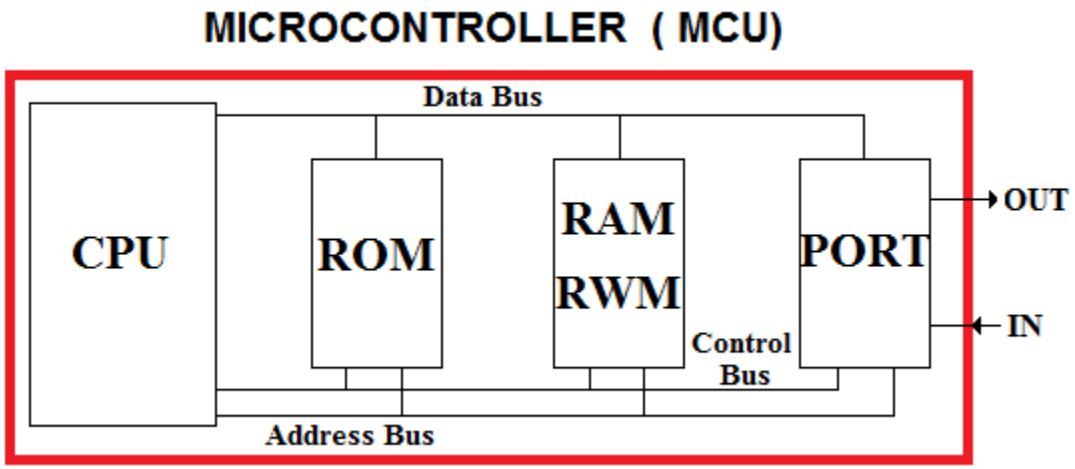
حافظه ای با قابلیت خواندن و نوشتن که اطلاعات میانی سیستم درون آن قرار می گیرد.

PORT : در هر سیستم تعدادی PORT وجود دارد که وظیفه آنها ورود و خروج اطلاعات است.

Bus : در هر سیستم سه نوع Bus وجود دارد: Data, Address, Control

- Address مشخص می کند با کدام حافظه یا پورت کار داریم .
- Control مشخص می کندچه کاری داریم.(خواندن یا نوشتن....)
- Data توسط این خطوط اطلاعات بین cpu و بقیه مدار منتقل می شود.

میکروکنترلر: اگر یک سیستم میکرو پروسسوری را داخل یک آی‌سی قرار دهیم و پایه های پورت‌ها ، در دسترس کاربر باشند به این قطعه میکروکنترلر گفته می شود.



توضیح و ترجمه بخش هایی از سه صفحه اول دیتا شیت AVR (ATMEGA 32)

نکته: هنگامی که گفته می شود یک میکرو کنترلر ۸ بیتی است؛ یعنی می تواند در یک مرحله ، عملی را روی دو عدد ۸ بیتی انجام دهد. حال اگر مثلا دو عدد ۱۶ بیتی داشته باشیم، باید آن را در دو مرحله انجام دهد.

به این پارامتر طول کلمه cpu نیز گفته می شود. (Word length)

این میکرو دارای ساختار RISC است:

RISC: Reduced Instruction Set Computer

CISC: Complex Instruction Set Computer

RISC: دستورهای ساده تر، برنامه نویسی مشکل، سرعت بالا (AVR)

CISC: دستورهای پیچیده، برنامه نویسی ساده تر، سرعت پایین (8051)

رجیسترها: در این میکرو ۳۲ رجیستر همه منظوره (General - purpose register) ۸ بیتی(R_{31} تا R_0) و ۶۴ رجیسترتک منظوره (special purpose register) IO وجود دارد؛ وظیفه رجیسترهای IO تنظیم قسمت های مختلف میکرو است.

برای مثال:

DDRA : Data Direction Register رجیستر تنظیم پورت A

PORTA : رجیستر خروجی پورت A

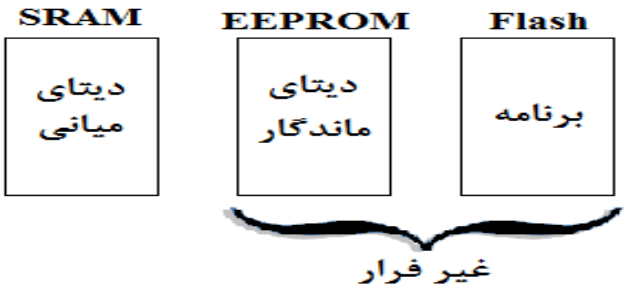
PINA : رجیستر ورودی پورت A

TCCR0: Timer Counter Control Register 0 تنظیم کننده تایمر کانتر صفر

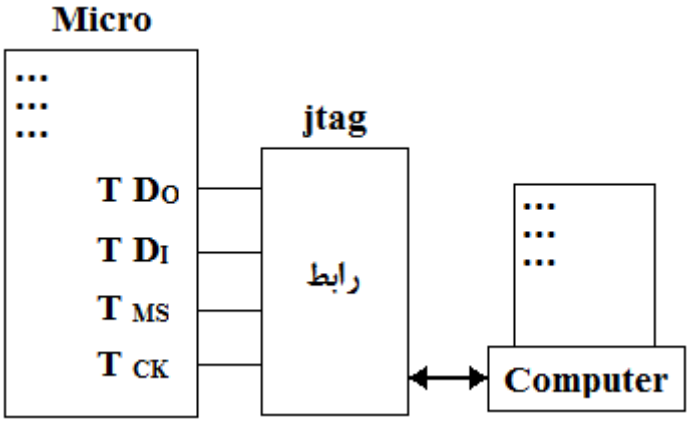
سرعت: این IC می تواند تحت فرکانس 16 MHZ با سرعت 16 MIPS عملیات را انجام دهد؛ یعنی در هر ثانیه ۱۶ میلیون دستور را اجرا نماید.

MIPS = Million Instruction Per Second

حافظه: در این میکرو سه نوع حافظه به شرح زیر وجود دارد:



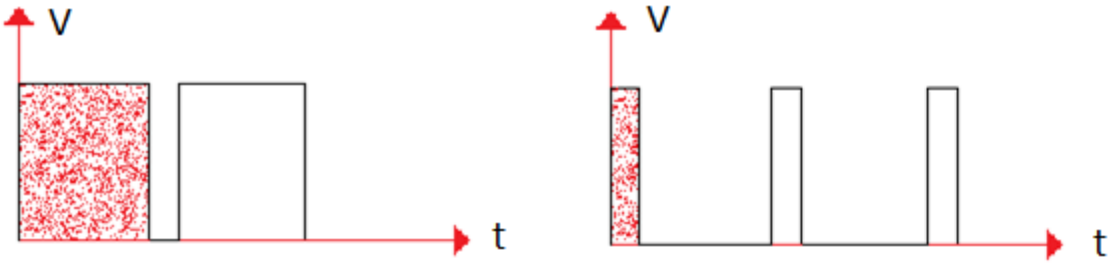
JTAG: یک واسطه بین میکرو و کامپیوتر است که می توان از طریق آن علاوه بر برنامه ریزی میکرو، برنامه داخل میکرو را خط به خط اجرا و در صورت نیاز آن را تصحیح کرد.Joint Test Action Group



تایمر و کانتر: در این میکرو دو تایمر کانتر ۸ بیتی و یک ۱۶ بیتی وجود دارد.

RTC (Real Time Counter): اگر بخواهیم زمان سنجی واقعی داشته باشیم، لازم است که بتوانیم پالس یک ثانیه را تولید کنیم؛ در این میکرو واحدی به نام RTC وجود دارد که می توان توسط آن پالس یک ثانیه و به تبع آن دقیقه و ساعت را تولید کرد.

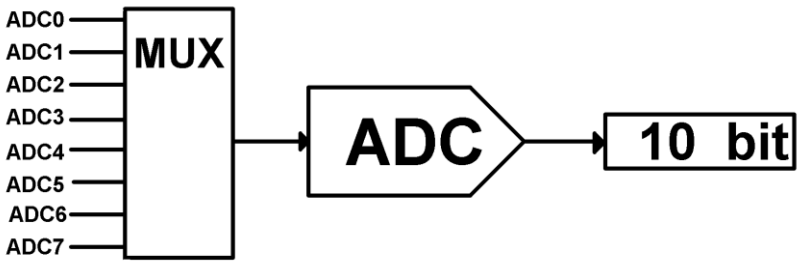
PWM (Pulse Width Modulator): یکی از راه های کنترل بعضی از دستگاه ها، کنترل انرژی است که به آن ها می رسد. توسط تغییر در عرض پالس، می توانیم انرژی و در نتیجه ،آن دستگاه را کنترل کنیم. برای مثال اگر دو موج هم دامنه و هم فرکانس زیر را به دو موتور مشابه بدهیم :



سرعت کمتر به خاطر انرژی کمتر سرعت بیشتر به خاطر انرژی بیشتر

در این میکرو چهار کانال PWM پیش بینی شده است، پایه های OC2, OC1B, OC1A, OC0.

ADC: Analog to Digital Converter تمام کمیت های اطراف ما مقادیری آنالوگ هستند ، مانند دما، فشار، رطوبت و... اگر بخواهیم آن ها را اندازه گیری و پردازش کنیم، لازم است که ابتدا به یک مقدار دیجیتال تبدیل شوند و سپس پردازش گردند. در این میکرو یک مبدل ۸ کاناله و ۱۰ بیتی در نظر گرفته شده است.



ارتباط: همواره یکی از مسائلی که با آن درگیر خواهید بود، ارتباط دو یا چند دستگاه است تا بتوان اطلاعات را از یک نقطه به نقطه دیگر منتقل کرد. در این میکرو سه نوع ارتباط پیش بینی شده است:

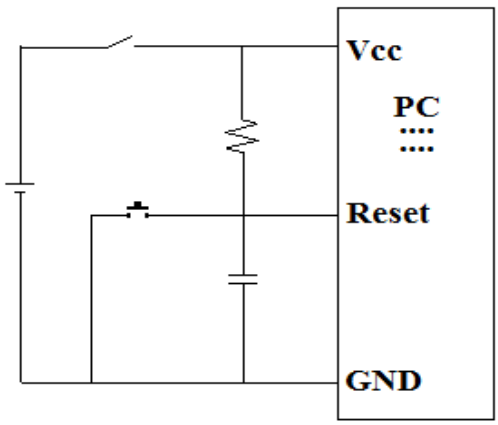
USART
- RXD دریافت دیتا
- TXD ارسال دیتا
- XCK پالس همزمانی

I2C (TWI)
- SDA ارسال و دریافت دیتا
- SCL پالس همزمانی

SPI
- MOSI: Master Out Slave In
- MISO: Master IN Slave Out
- SCK: Slave Clock
- SS: Slave Select

Watchdog Timer : هرگاه سیستمی هنگ کند برای این که دوباره به کار بیفتد لازم است که آن را ریست کنیم.(WD) وظیفه دارد که Cpu را تحت نظر داشته باشد، تا در صورتی که به هر علتی هنگ کرد آن را ریست نماید.این بخش یک تایمر است که حداکثر تا ۲ ثانیه تنظیم می شود اگر به هر علتی میکرو هنگ کند و نتواند WD را ریست کند بعد از زمان تنظیم شده WD میکرو را ریست می کند.

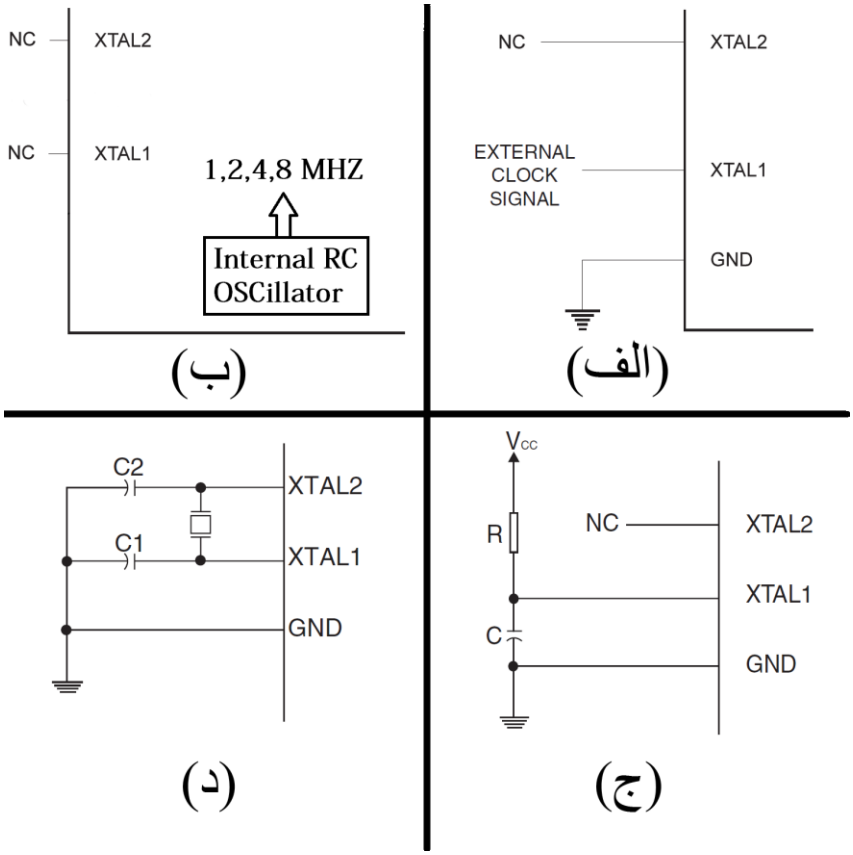
Reset : تمام میکروها لازم است که پس از روشن شدن ریست شوند، تا کار محول شده را از خط اول برنامه شروع کنند. این میکرو طوری طراحی شده که هر گاه آن را روشن کنید، میکرو را ریست می کند ، که به آن Power On Reset گفته می شود.



هرگاه ریست اتفاق می افتد،شمارنده برنامه (Program Counter) PC برابر صفر می شود؛ در نتیجه برنامه حتماً از خط اول شروع خواهد شد.

Brown-Out Detection : ساز و کاری است که ولتاژ تغذیه میکرو را پایش می کند، تا در صورتی که از مقدار مشخصی کمتر شد، آن را ریست کند.

Oscillator : می دانیم که پردازنده ها از تعداد زیادی مدارهای ترتیبی و ترکیبی ساخته شده اند. برای هماهنگی بین قسمت های مختلف نیاز به یک پالس ساعت داریم که همه قسمت ها را با هم هماهنگ کنند. برای این کار روش های زیر وجود دارد.



همانطورکه در شکل ها دیده می شود، برای تهیه این پالس چهار روش وجود دارد:

الف) اسیلاتور خارجی

ب) اسیلاتور RC داخلی (در میکروی نو روی فرکانس داخلی 1mhz تنظیم است)

ج) اسیلاتور RC خارجی $f=1/3RC$

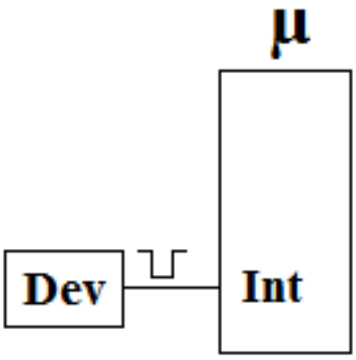
د) اسیلاتور داخلی با کریستال خارجی

نکته: برای تغیر منبع پالس لازم است که فیوز بیت های مربوط به این بخش را توسط پروگرامر تنظیم کنیم.

وقفه (Interrupt): برای سرویس دهی به هر دستگاه جانبی دو روش وجود دارد:

- سرکشی Polling
- وقفه Interrupt

در روش اول Cpu موظف است در فاصله زمانی های مشخص به دستگاه جانبی سرکشی کند، تا در صورت نیاز به آن دستگاه سرویس لازم را ارائه نماید. در این روش وقت زیادی از Cpu تلف می شود. اما در روش وقفه دستگاه هر موقع نیاز داشت با ارسال سیگنال وقفه درخواست سرویس می کند؛ میکرو کار خود را قطع و به آن دستگاه سرویس می دهد، در نتیجه وقتی تلف نمی شود. در این میکرو سه وقفه خارجی وجود دارد؛ پایه های Int2, Int1, Int0 .



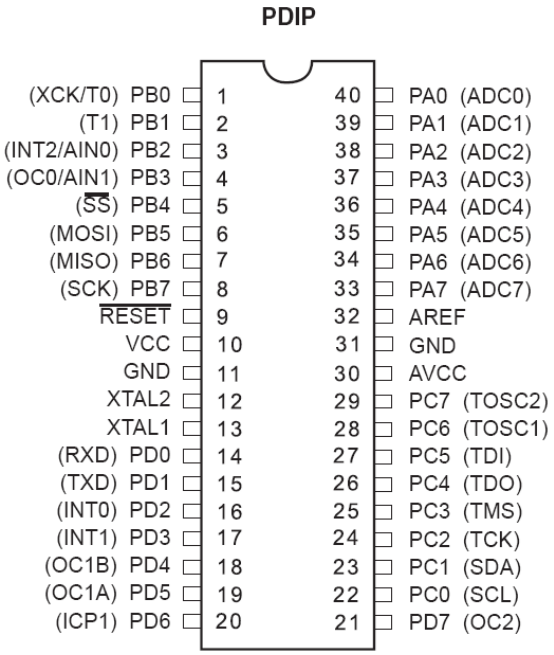
Sleep : برای ساخت دستگاه های قابل حمل و نقل (Portable) لازم است که تا حد امکان انرژی کمتری مصرف شود تا طول عمر باتری بیشتر گردد؛ لازم است که میکرو بتواند در هنگام بیکاری به حالت (Sleep) برود تا انرژی کمتری مصرف کند و طول عمر باتری افزایش یابد.دراین میکرو شش روش sleep پیش بینی شده است .

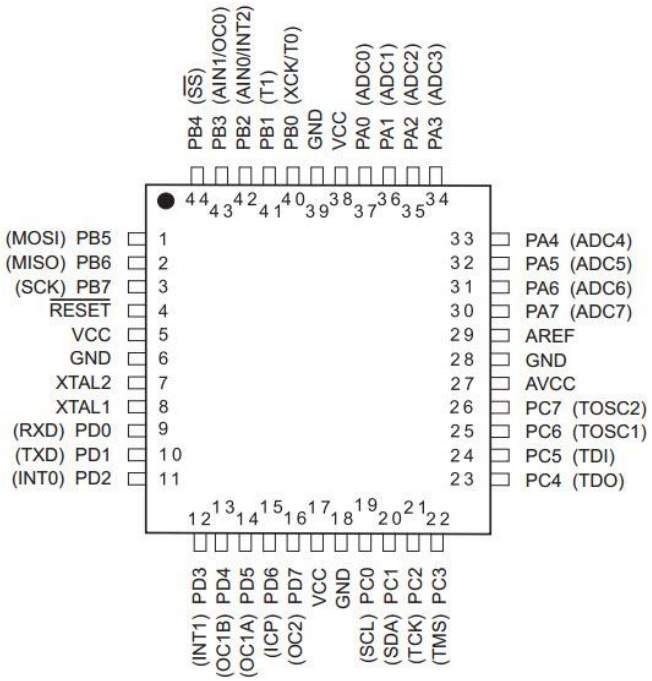
بسته بندی Packing:

این میکرو در دو بسته بندی PDIP و TQFP عرضه می شود.

PDIP: Plastic Dual Inline Package

TQFP: Thin Quad Flat Package





نام محصول : این میکرو در ابتدای تولید با دو نام ATmega32L و ATmega32 به بازار عرضه شد.

مزیت نوع L ولتاژ کار 2.7v تا 5.5v و اشکال آن حداکثرفرکانس کار 8Mhz بود.

مزیت نوع معمولی حداکثرفرکانس کار 16Mhz و اشکال آن ولتاژ کار 4.5v تا 5.5v بود.

کارخانه سازنده در ATmega32A هر دو مزیت را قرار داده است. هم اکنون ، این محصول در بازار وجود دارد.

پایه ها :

پایه های این میکرو تک وظیفه ای ، دو یا سه وظیفه ای می باشد.

PA0~PA7 PB0~PB7 PC0~PC7 PD0~PD7 : پایه های پورت ها

T0 و T1 : پالس ورودی به تایمر/کانتر صفر و یک.

AIN0~AIN1 : پایه های ورودی مقایسه کننده آنالوگ.

INT0,1,2 : پایه های ورودی اینتراپت خارجی .

OC0,OC1A,OC1B,OC2 : چهار کانال خروجی PWM .

SS,MOSI,MISO,SCK : چهار پایه مربوط به ارتباط SPI .

SS: slave select فعال کننده slave

MISO: master in slave out دیتای خارج شده از slave را به master وارد می کند.

MOSI: master out slave in دیتای خارج شده از master را به slave وارد می کند.

SCK: slave clock پالس همزمانی را از master به slave منتقل می کند.

RESET : هر گاه این پایه صفر شود میکرو باز نشانی و برنامه از ابتدا اجرا می شود.

VCC,GND : با اعمال ولتاژ بین 2.7v تا 5.5v به این دو پایه میکرو تغذیه و کار می کند.

XTAL1,2 : پایه های اتصال کریستال خارجی تا فرکانس 16mhz .

RXD,TXD,XCK: این سه پایه مربوط به واحد USART می باشند.

RXD: Receive data پایه دریافت دیتا به واحد usart .

TXD: Transmit data پایه ارسال دیتا از واحد usart .

XCK: Clock پایه ارسال و دریافت پالس همزمانی واحد usart .

ICP1: (Timer/Counter1 Input Capture Pin) با تحریک این پایه عدد داخل تایمر۱ در رجیستر ICR1 ذخیره می شود.

SCL,SDA : پایه های ارتباط I2C یا TWI (Two Wire Interface /Inter-Integrated Circuit)

Serial Data Line (SDA) : خط ارسال و دریافت دیتا

Serial Clock Line (SCL) : خط انتقال پالس همزمانی

TDI,TDO,TMS,TCK : چهار پایه مربوط به ارتباط JTAG

نکته: در میکرو نو JTAG فعال است در نتیجه چهار پایه از PORTC در اختیار پورت نیست.

TOSC1,2 : برای استفاده از RTC داخلی میکرو ، باید یک کریستال با فرکانس 32768hz به این دو پایه وصل شود.

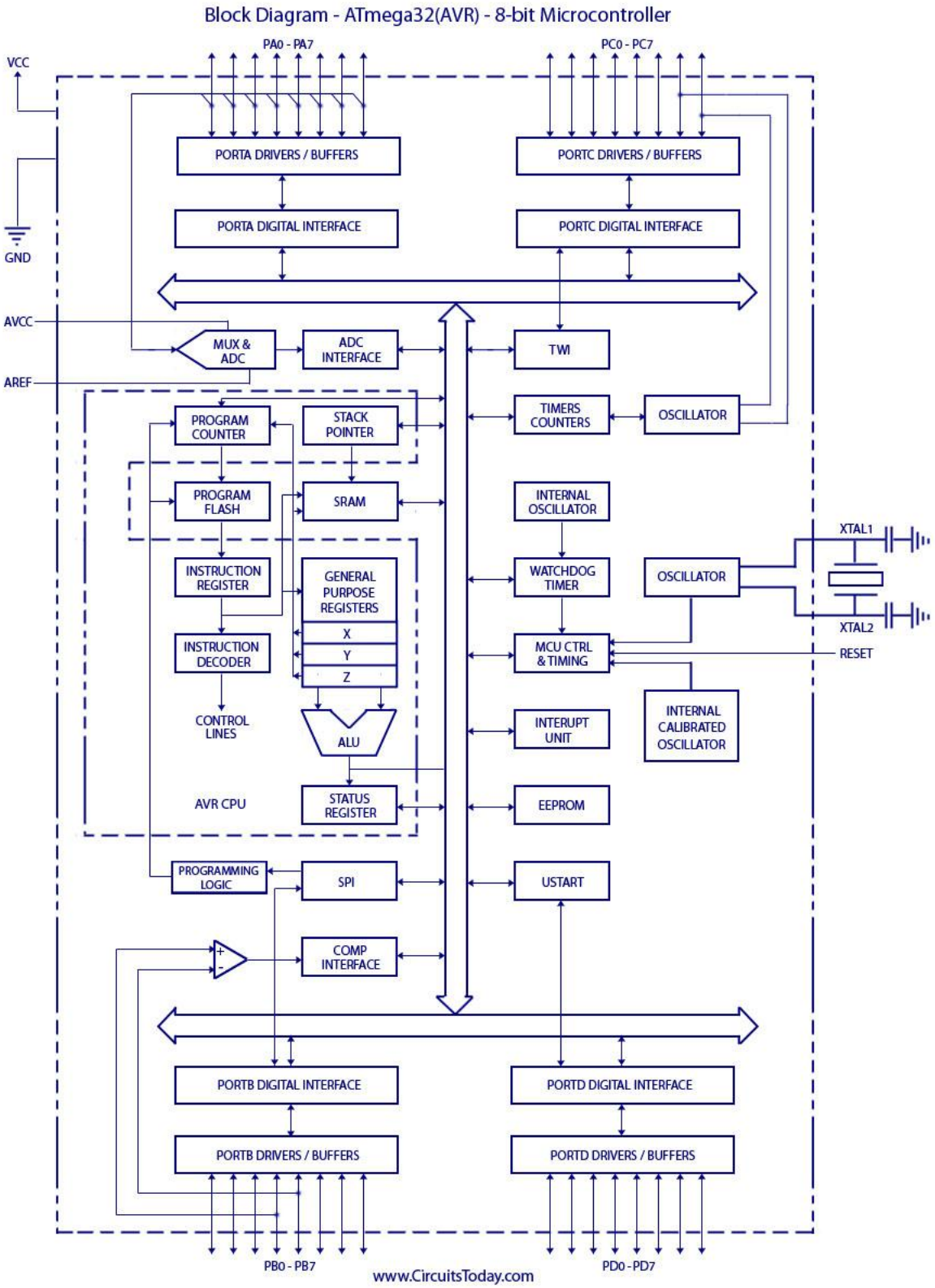
AVCC,GND,AREF : پایه های مربوط به مبدل آنالوگ به دیجیتال ADC

AVCC,GND : تغذیه واحد ADC را تامین می کنند.

AREF : ولتاژ مرجع خارجی بخش ADC که بین 0 تا 5v است به این پایه وصل می شود.

ADC0~ADC7 : هشت کانال ورودی آنالوگ به بخش ADC .

بلوک دیاگرام داخلی AVR :



نرم افزار: پس از طراحی و ساخت سخت افزار لازم است برنامه ای نوشته شود تا آن را کنترل کند. تنها زبان قابل فهم برای پردازنده ها زبان ماشین می باشد، ولی نوشتن و رفع اشکال برنامه به زبان ماشین سخت و مشکل است؛ لذا از زبان های برنامه نویسی سطح بالا **(High-level programming language)** مانند C استفاده می کنیم.

ساختار زبان C : همانطور که قبلا گفته شد، لازم است که برنامه ای برای میکرو نوشته شود و زبان C مناسب و ساختار آن به شکل زیر است:

```
#include   (هدر فایل)
ماکروها
متغیرهای عمومی
توابع
void  main (void)
{
….
….
….          برنامه
….
….
}
```

میکروی مورد نظر ما دارای ۴ پورت و هر پورت دارای سه رجیستر به شرح زیر می باشد:

DDRX Data Direction Register	مشخص می کند پورت ورودی باشد یا خروجی **0 برای ورودی 1 برای خروجی**
PINX	رجیستر برای ورود دیتا
PORTX	رجیستر برای خروج دیتا

نوشتن اعداد در مبناهای مختلف در زبان C :

در کد ویژن می توانید اعداد را در مبنای ۲،۸،۱۶ و ۱۰ بنویسید.

در جدول زیر اعداد 0 تا 15 در مبناهای ۸،۱۶،۲،۱۰ جهت یادآوری نوشته شده است.

مبنای ۱۰	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
مبنای ۲	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
مبنای۱۶	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
مبنای ۸	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17

فرض کنید می خواهیم عدد 12 را در مبنا های مختلف به PORTA ارسال کنیم .

$$(12)_{10} = (1100)_2 = (14)_8 = (c)_{16}$$

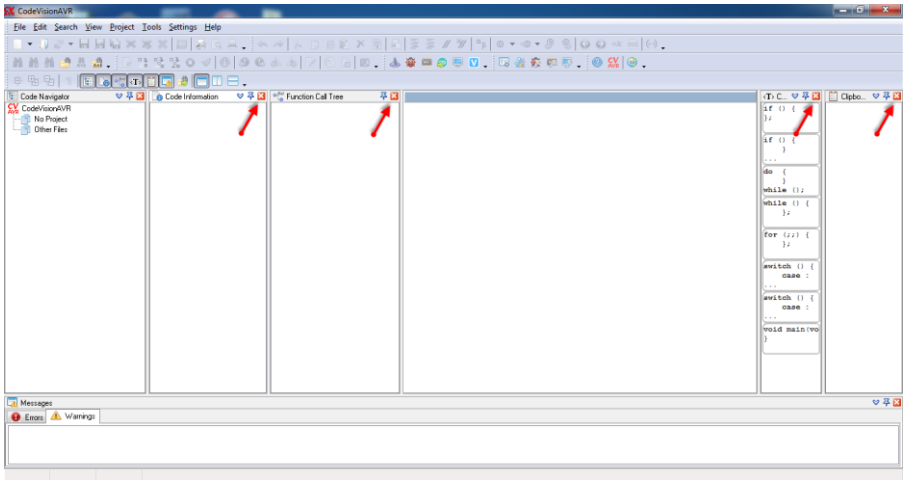
PORTA=12; مبنای ۱۰

PORTA=0b1100; مبنای ۲

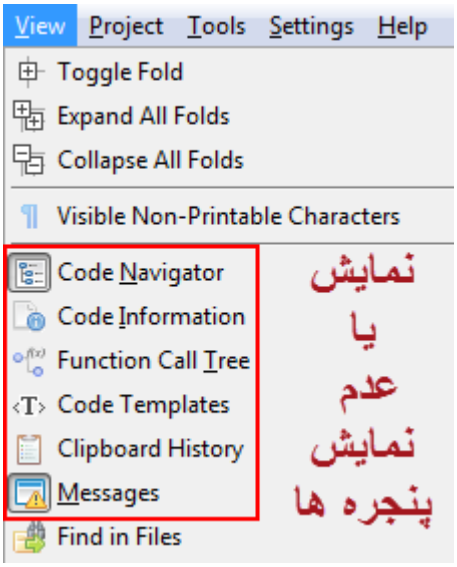
PORTA=014; مبنای ۸

PORTA=0xC; مبنای ۱۶

مراحل نوشتن برنامه در Code Vision نسخه 3.12 :

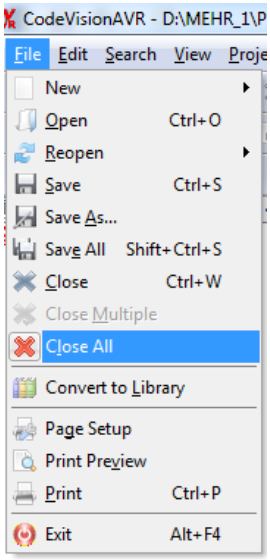


۱- اگر برای اولین بار است که برنامه کدویژن را اجرا می‌کنید ، تعداد زیادی پنجره باز را خواهید دید. که ممکن است برای افراد مبتدی باعث سردرگمی شود، بهتر است به غیر از پنجره Code Navigator بقیه را ببندید.

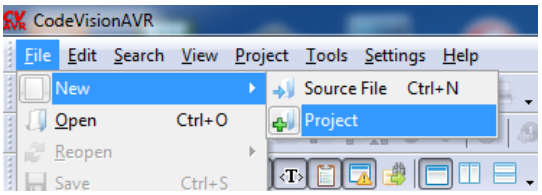


در صورت نیاز می توانید پنجره های مورد نیاز را از مسیر زیر باز کنید.

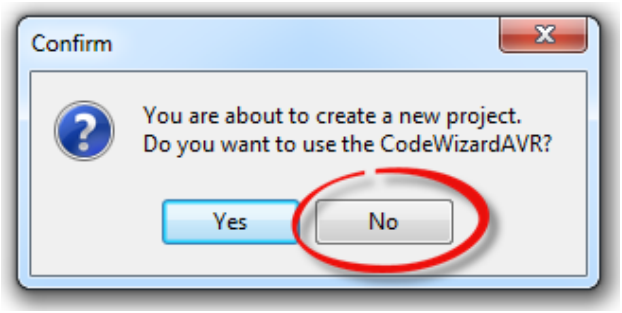
۲- معمولا پس از باز کردن برنامه آخرین پروژه ای که کار کرده اید باز می شود؛ از مسیر زیر آن را ببندید. File\Close All



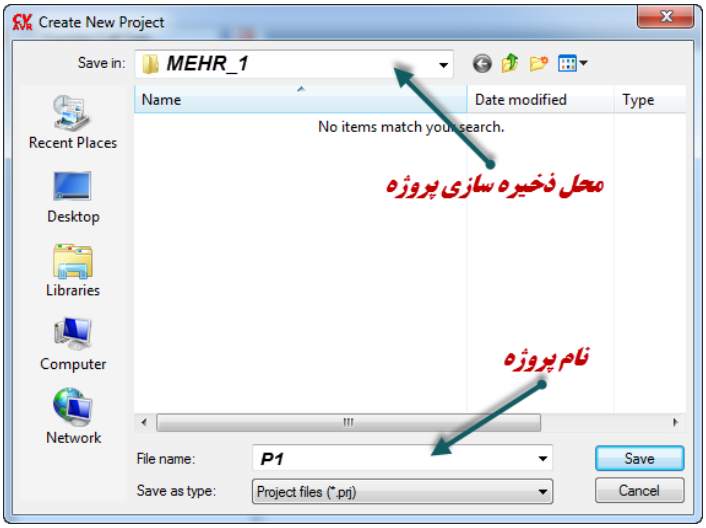
۳- از مسیر زیر یک پروژه برای نوشتن برنامه باز و آن را با نامی مناسب و در محلی مشخص ذخیره نمایید. File\New\Project

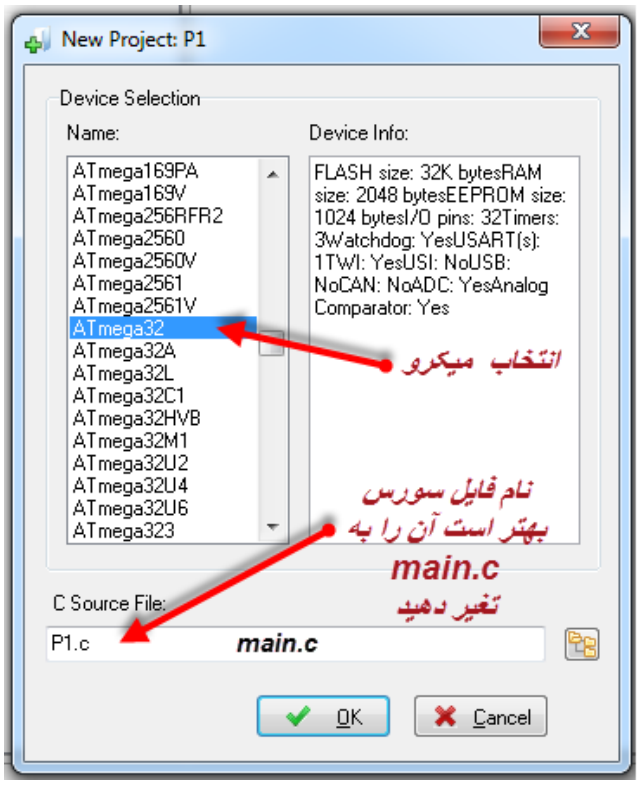


پنجره مقابل باز می شود و سوال می کند که آیا مایل هستید از CodeWizard استفاده کنید .در این مرحله نمی خواهیم از ویزارد استفاده کنیم ، بنابراین گزینه No را انتخاب می‌کنیم .

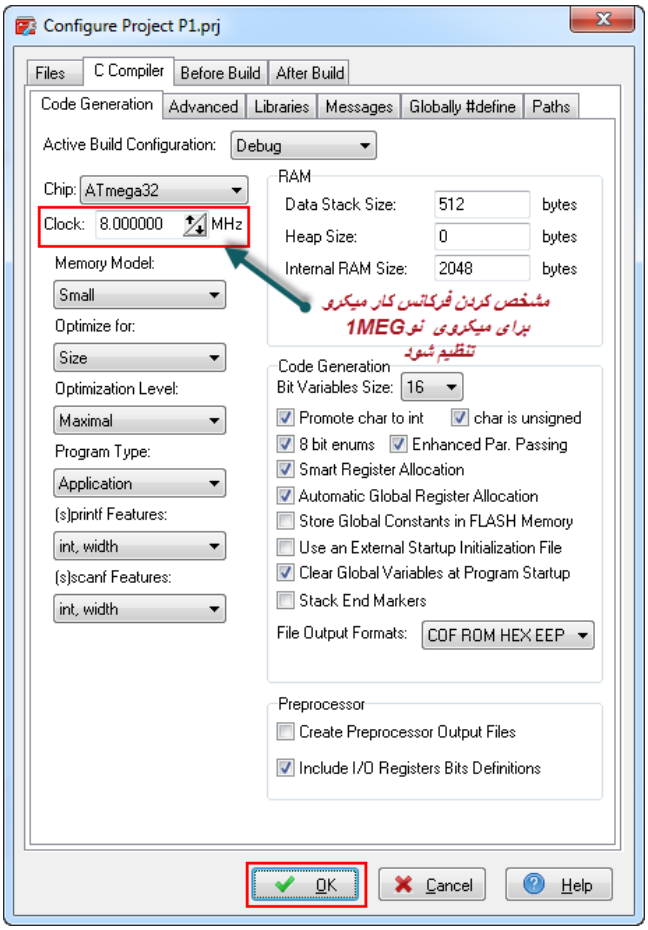


پنجره ذخیره سازی پروژه باز می شود یک پوشه باز کنید و نام پروژه را وارد کرده کلید Save را کلیک کنید.



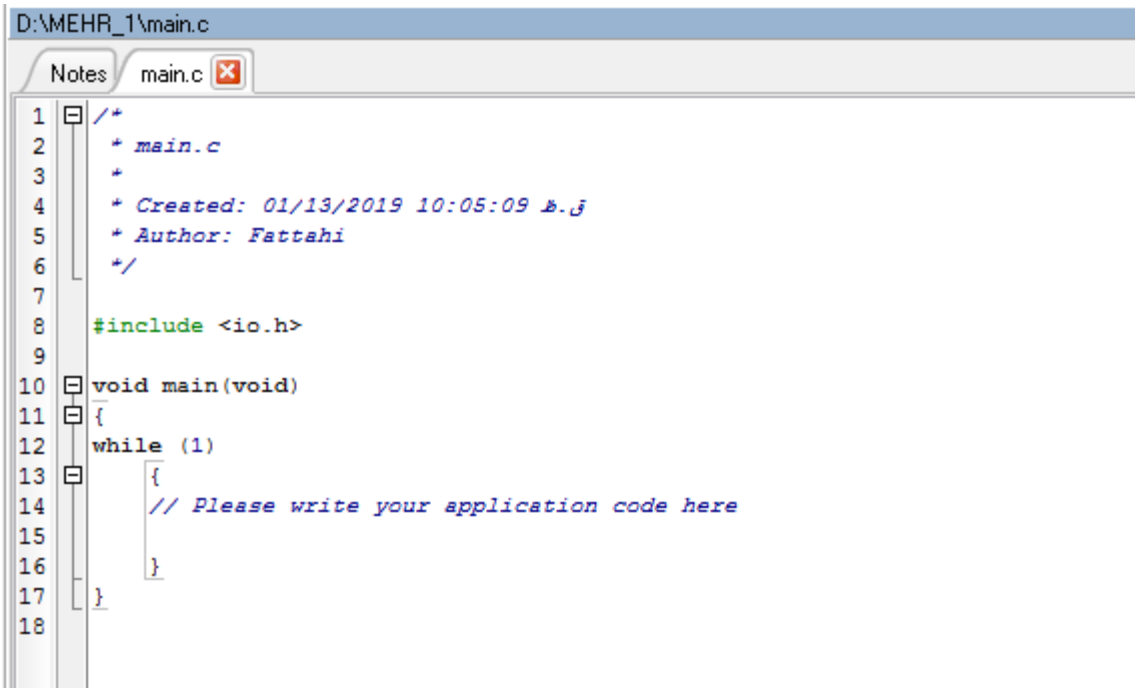


پنجره New Project باز می شود شماره میکرو و نام فایل Source را وارد کنید. البته برای سورس نامی پیشنهاد شده است ولی بهتر است آن را به main.c تغیر دهید. سپس Ok

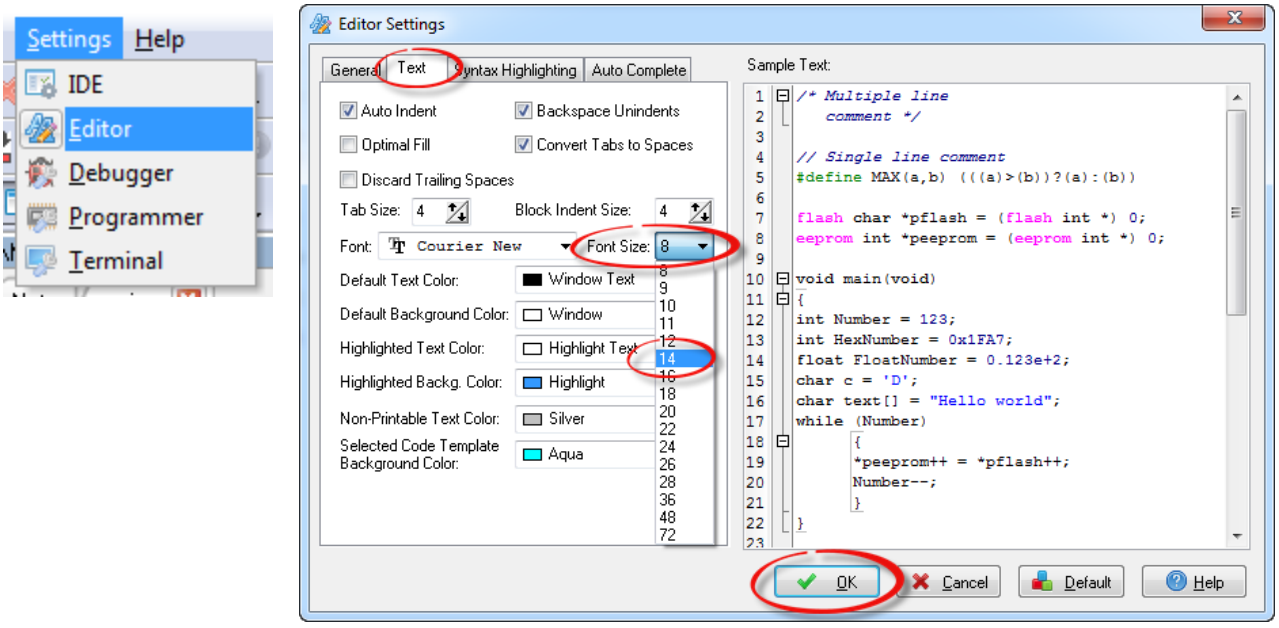


۴- پس از انتخاب میکرو ، و وارد کردن نام فایل سورس ، پنجره ی Configure Project باز می شود. در همین پنجره برگه C Compiler را باز کرده ، در بخش Clock فرکانس کار میکرو را مشخص کنید. و در آخر Ok را کلیک کنید.

۵- مراحل ساخت پروژه به پایان رسیده و فایل سورس به شکل زیر باز می شود .متن برنامه را در فایل main.c می نویسیم. توجه داشته باشید که فایل Notes مانند دفترچه یادداشت است و می توانید هر توضیح یا هر متن مورد نظری را داخل آن قرار دهید.



توجه: اندازه فونت بصورت پیش فرض ۸ می باشد بهتر است از مسیر زیر آن را بزرگتر کنید.



D:\MEHR_1\main.c

Notes main.c *

```
1  /*
2   * main.c
3   *
4   * Created: 01/13/2019 10:05:09 ق.ظ
5   * Author: Fattahi
6   */
7  #include <io.h>
8  void main(void)
9  {
10 while (1)
11      {
12      // Please write your application code here
13      }
14 }
15
```

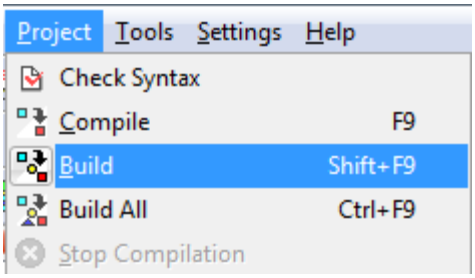
هدر فایل مربوط به معرفی میکرو
که البته می توانید آن را به نام میکرو تغییر دهید
#include <mega32.h>

۶- برنامه را در فایل main.c می نویسیم.

main.c Notes

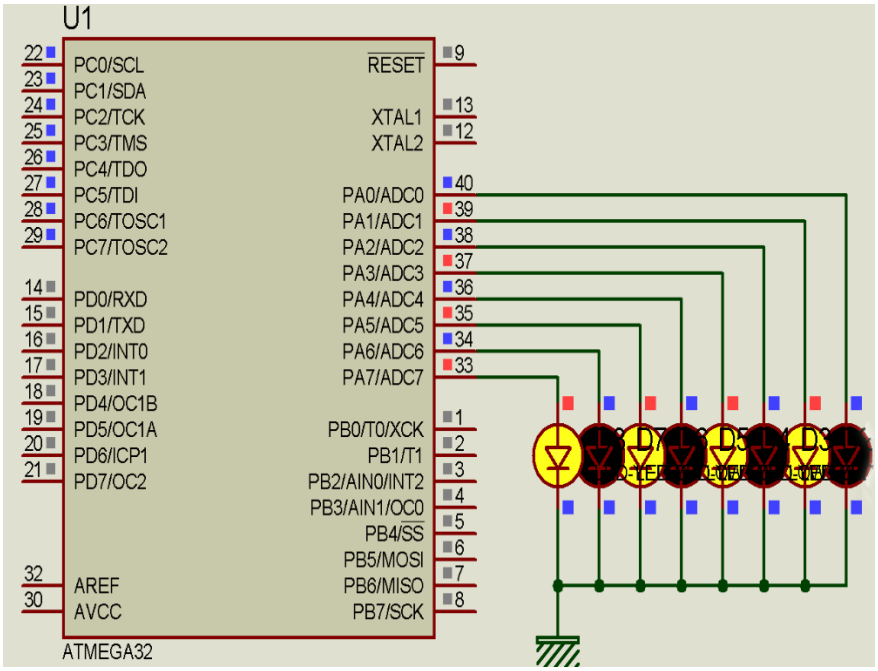
```
1  #include <mega32.h>
2  #include <delay.h>
3  void main(void)
4  {
5   DDRB=0xff;
6   s1:
7   PORTB=0b01010101;
8   delay_ms(500);
9   PORTB=0b10101010;
10  delay_ms(500);
11  goto s1;
12 }
```

۷- از منوی Project گزینه ی Build را انتخاب می کنیم تا برنامه به زبان ماشین تبدیل و فایل HEX ساخته شود.



مثال: هشت عدد LED را به پورت A متصل کنید برنامه ای بنویسید که آن ها را یکی در میان روشن کند و نیم ثانیه در این حالت نگه دارد. سپس LED های خاموش و روشن عوض شوند و نیم ثانیه نیز در این حالت بماند و در ادامه همین روند تکرار شود.

```
#include <mega32.h>
#include <delay.h>
void main (void)
{
DDRA=0xFF;
S1:
PORTA=0b01010101;
delay_ms(500);
PORTA=0b10101010;
delay_ms(500);
goto S1;
}
```

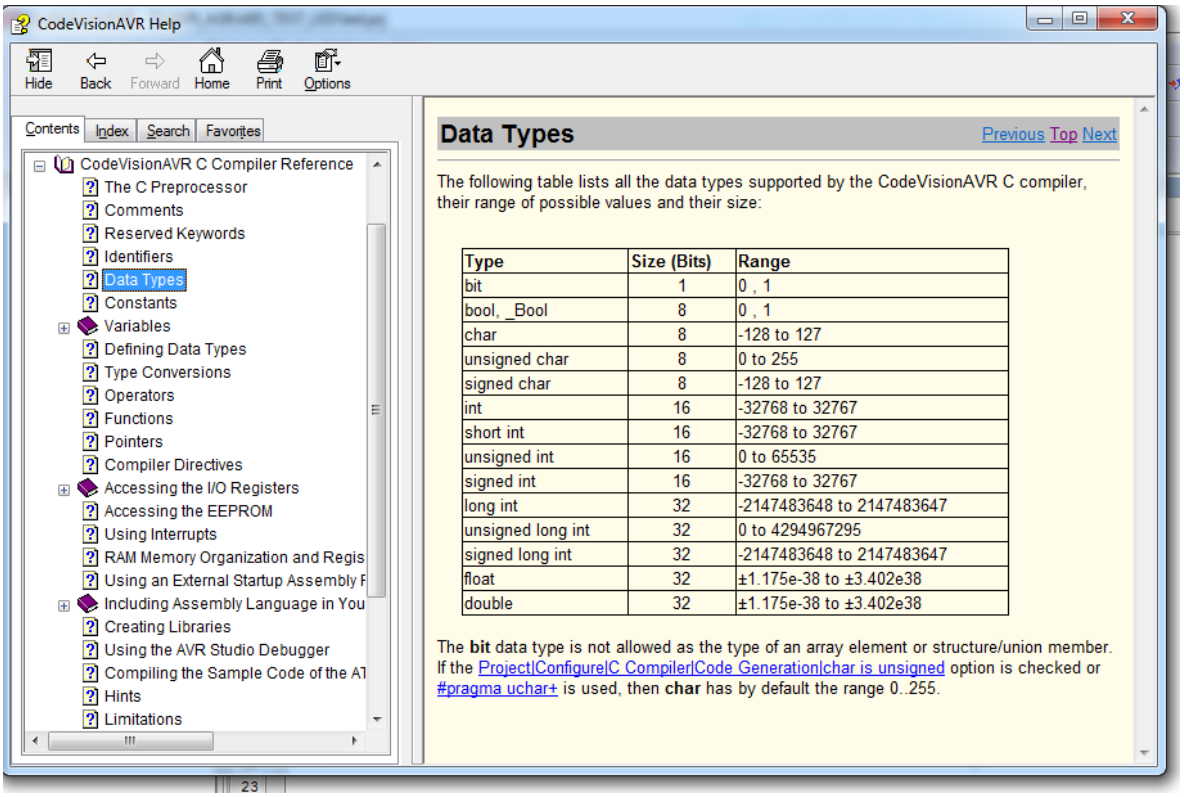


متغیر Variable : در هر زبان برنامه نویسی لازم است محل هایی از حافظه برای نگه داری اعداد،کاراکترها و رشته ها تعین گردد تا در هنگام اجرای برنامه بتوان آن ها را خواند یا نوشت. در زبان C برای تعریف یک متغیر از ساختار زیر استفاده می شود.

نوع متغیر نام متغیر =مقدار ;

```
unsigned char    a=5 ;
int    b,c,d=2000;
float  h=3.14;
char   t='R';
char    s[ ]="REZA";
```

انواع متغیرها و رنج اعداد قابل نمایش توسط آن ها در Help برنامه و در بخش Data Types در دسترس می باشد .



مثال : هشت عدد LED به PORTA متصل کنید و یک شمارنده بالا شمار بر روی آن ایجاد نمایید.

```
#include <mega32.h>
#include <delay.h>
unsigned char i=0;
```

```
void main(void)
{
DDRA=0xFF;
s1:
PORTA=i;
delay_ms(500);
i++;
goto  s1;
}
```

دستور شرطی if : هرگاه قرار باشد دستوراتی بنابر شرایط خاص انجام شود، از دستورهای شرطی استفاده می کنیم.

```
If (شرط)
{
 ....          دستورها
}
 else
{
  ....         دستورها
 }
```

```
If (شرط)  دستور;
```

```
If (شرط)
{
….
….
….          دستورها

}
```

مثال: بر روی LED های PORTA یک شمارنده بالا شمار 0 تا 9 ایجاد کنید.

```
#include <mega32.h>
#include <delay.h>
unsigned char i=0;
void main(void)
{
DDRA=0xFF;
s1:
PORTA=i;
```

```
delay_ms(500);
i++;
if (i==10)  i=0;
goto  s1;
}
```

مثال: بر روی LED های PORTA یک شمارنده پایین شمار 9 تا 0 ایجاد کنید.

```
#include <mega32.h>
#include <delay.h>
unsigned char i=9;
void main(void)
{
DDRA=0xFF;
s1:
PORTA=i;
delay_ms(500);
i--;
if (i==255)  i=9;
goto  s1;
}
```

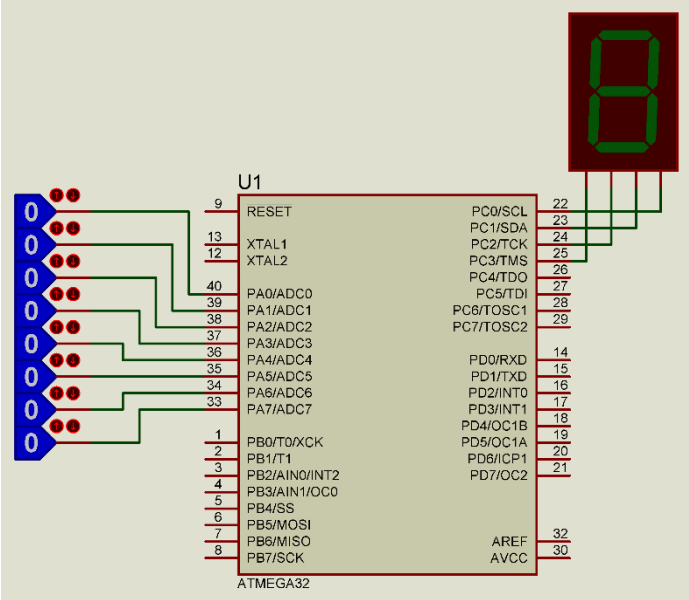
```
#include <mega32.h>
#include <delay.h>
signed char i=9;
void main(void)
{
DDRA=0xFF;
s1:
PORTA=i;
delay_ms(500);
i--;
if (i== -1)  i=9;
goto  s1;
}
```

دستور switch:

هر گاه لازم باشد در برنامه به ازای مقادیر مختلف یک متغیر دستورهای متفاوتی اجرا شود می توان از دستور switch با ساختار زیر استفاده کرد.

```
switch ( عبارتی که باید مورد بررسی قرار گیرد)
  {
    case مقدار ثابت ۱:
          مجموعه دستورات ۱
          break;
    case مقدار ثابت ۲:
          مجموعه دستورات ۲
          break;

    .
    .

    case مقدار ثابت n:
          مجموعه دستورات n
          break;
    default :
       مجموعه دستورات حالت پیش فرض
  }
```

مثال: عددی را از PORTA بخوانید ، اگر عدد خوانده شده ۱ بود روی PORTC عدد ۵ ، اگر ۲ بود عدد ۷ ، اگر ۳ بود عدد ۱۳ دیده شود و اگر غیر از ۱و۲و۳ بود عدد ۱۵ دیده شود.

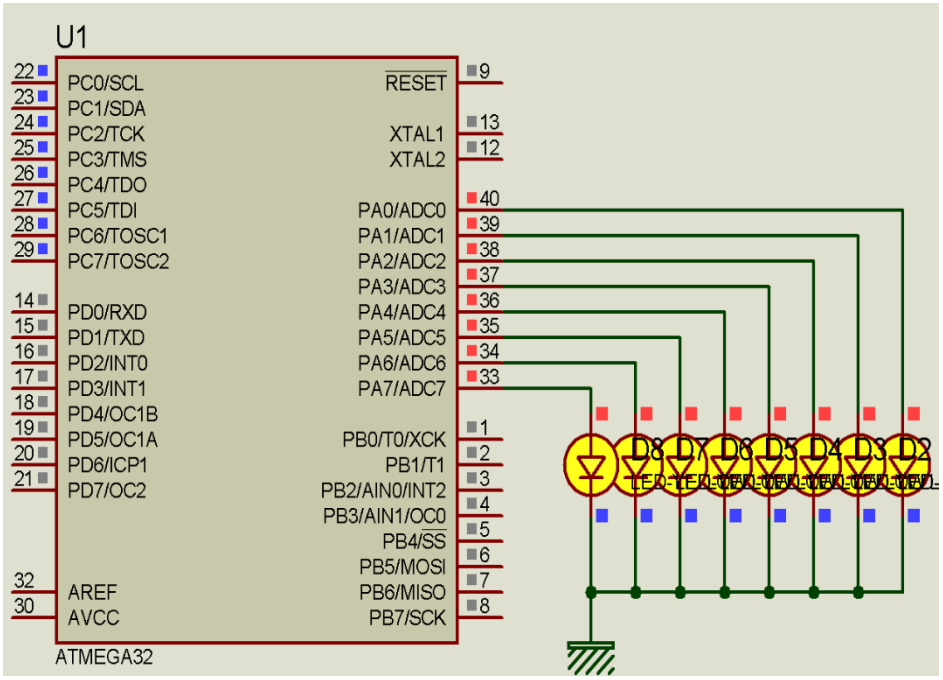



```
#include <mega32.h>
unsigned char  a;
void main(void)
{
DDRA=0x00;
DDRC=0xFF;
s1:
a=PINA;
      switch(a)
      {
       case 1:
       PORTC=5;
       break;
       case 2:
       PORTC=7;
       break;
       case 3:
       PORTC=13;
       break;
       default:
       PORTC=15;
       }
goto  s1;
}
```

حلقه Loop : در برنامه نویسی بسیار پیش می آید که لازم است دستور یا دستورهایی چندین بار اجرا گردد. راه مناسب این است که آن ها را درون یک حلقه قرار دهیم تا به تعداد دفعات لازم تکرار گردد. در هر حلقه یک کنتور وجود دارد؛ آن را با عدد حلقه پر می کنیم و با هر بار اجرا یک واحد از آن کم می کنیم؛ هرگاه صفر شد، از حلقه خارج می شویم.

مثال: هشت عدد LED به پورت A متصل کنید . برنامه ای بنویسید که آن ها را پنج بار خاموش و روشن کند.

```
#include <mega32.h>
#include <delay.h>
unsigned char i;
void main (void)
{
DDRA=0xFF;
i=5;
S1:
PORTA=0xFF;
delay_ms(500);
PORTA=0x00;
delay_ms(500);
i--;
if(i!=0) goto S1;
S2:goto S2;
}
```



دستور while : روش دیگر ایجاد حلقه دستور while با ساختار زیر می باشد.

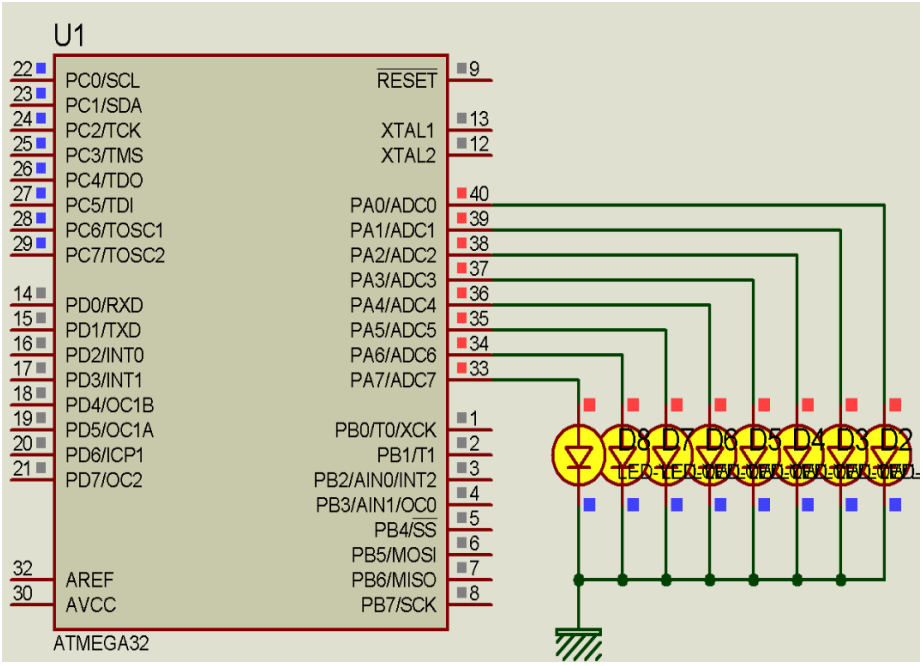
```
while (شرط اجرای حلقه)
{
....
....          دستورها
}
```

while (1)
{
……
……
}
	نکته: برای ایجاد یک حلقه بی نهایت می توان به شکل زیر عمل کرد.

while (1) ;	نکته: اگر می خواهید برنامه روی خطی متوقف شود دستور زیر را بنویسید.

مثال: مثال قبل را با دستور while اجرا نمایید.

```
#include <mega32.h>
#include <delay.h>
unsigned char i;
void main (void)
{
DDRA=0xFF;
i=5;
while(i>0)
{
PORTA=0xFF;
delay_ms(500);
PORTA=0x00;
delay_ms(500);
i--;
}
S2:goto S2;
}
```



حلقه for : هرگاه تعداد دفعات تکرار حلقه مشخص باشد، می توان از دستور for استفاده کرد.

for (متغیر حلقه = مقدار اولیه ; شرط اجرای حلقه ; گام حلقه)

{

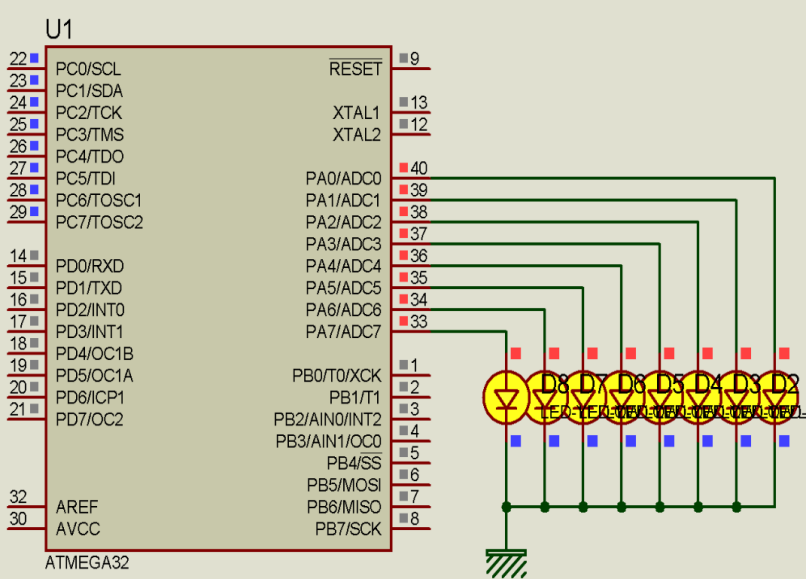
....

.... دستورها

}

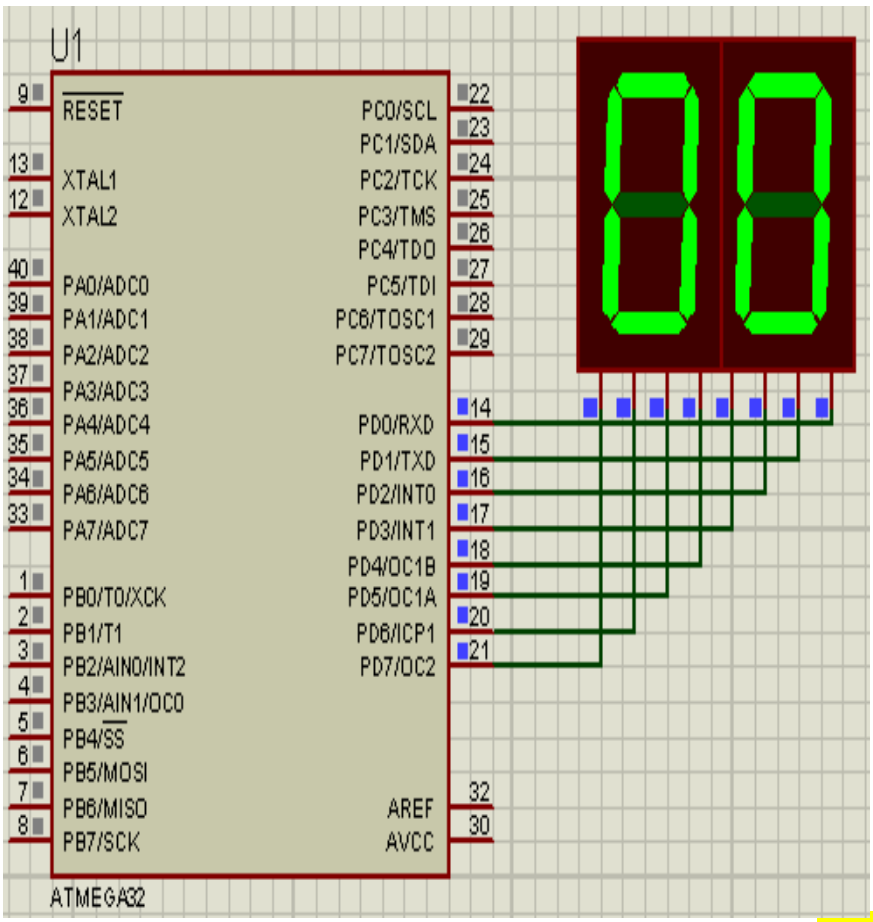
مثال: مثال قبل را با دستور for اجرا نمایید.

```
#include <mega32.h>
#include <delay.h>
unsigned char i;
void main (void)
{
DDRA=0xFF;
for(i=0;i<5;i++)
{
PORTA=0xFF;
delay_ms(500);
PORTA=0x00;
delay_ms(500);
}
while(1);
}
```



مثال: دو عدد سون سگمنت BCD به پورت D متصل کنید . برنامه ای بنویسید که یک شمارنده " ۰ " تا "۲۰" بر روی پورت D داشته باشیم.

```
#include <mega32.h>
#include <delay.h>
unsigned char i;
void main (void)
{
DDRD=0xFF;
while(1)
{
for(i=0;i<21;i++)
{
PORTD=i;
delay_ms(500);
}
}
}
```



مثال: مثال قبل را طوری تغییر دهید که یک شمارنده زوج شمار داشته باشیم.

برای این کار کافیست خط ۹ را به صورت زیر تغییر دهیم:

```
for(i=0;i<21;i=i+2)
```

مثال: مثال قبل را فرد شمار کنید.

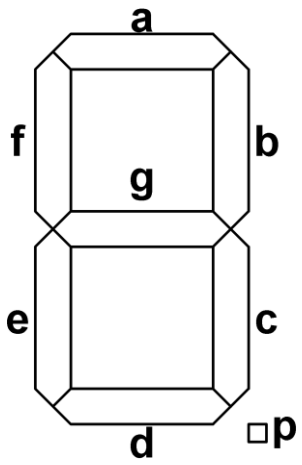
برای این کار کافیست خط ۹ را به صورت زیر تغییر دهیم:

```
for(i=1;i<21;i=i+2)
```

مثال: یک سون سگمنت کاتد مشترک را به پورتC متصل کنید و شمارنده ۰ تا ۹ بر روی آن بسازید.

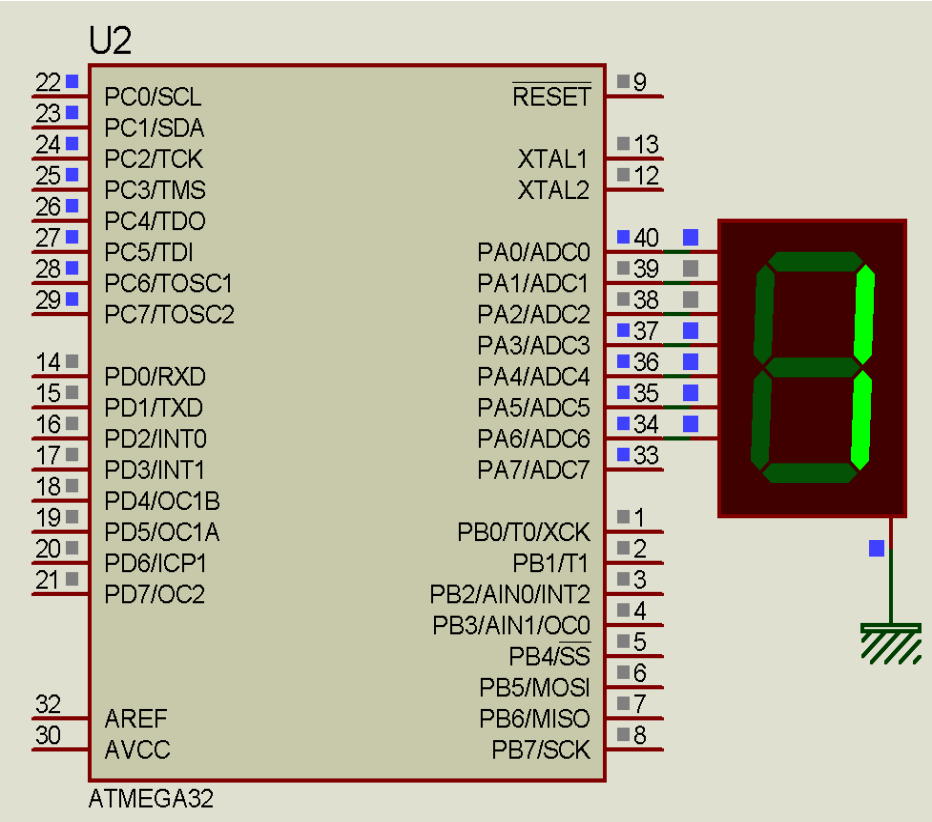
برای این کار باید از کد اعداد برای سون سگمنت استفاده نمود که به شرح زیر است.

عدد	p	g	f	e	d	c	b	a	کد
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F



Number	0	1	2	3	4	5	6	7	8	9
Code	0x3F	0x06	0x5B	0x4F	0x66	0x6D	0x7D	0x07	0x7F	0x6F

```
#include <mega32.h>
#include <delay.h>
void main (void)
{
DDRA=0xFF;
while(1)
{
PORTA=0x3F;
delay_ms(500);
PORTA=0x06;
delay_ms(500);
.
.
.
PORTA=0x6F;
delay_ms(500);
}
}
```



آرایه (ARRAY) : اگر به تعداد زیادی متغیر از یک نوع نیاز باشد بهتر است برای معرفی آن ها از آرایه یا متغیرهای اندیس دار استفاده نماییم.که به شکل زیر تعریف می شوند.

نوع متغیر نام آرایه [تعداد خانه ها]={ مقدار اول , مقدار دوم , ... }

```
int     d[5] = { 7 , 12 , 0 , 99 , 20};
```

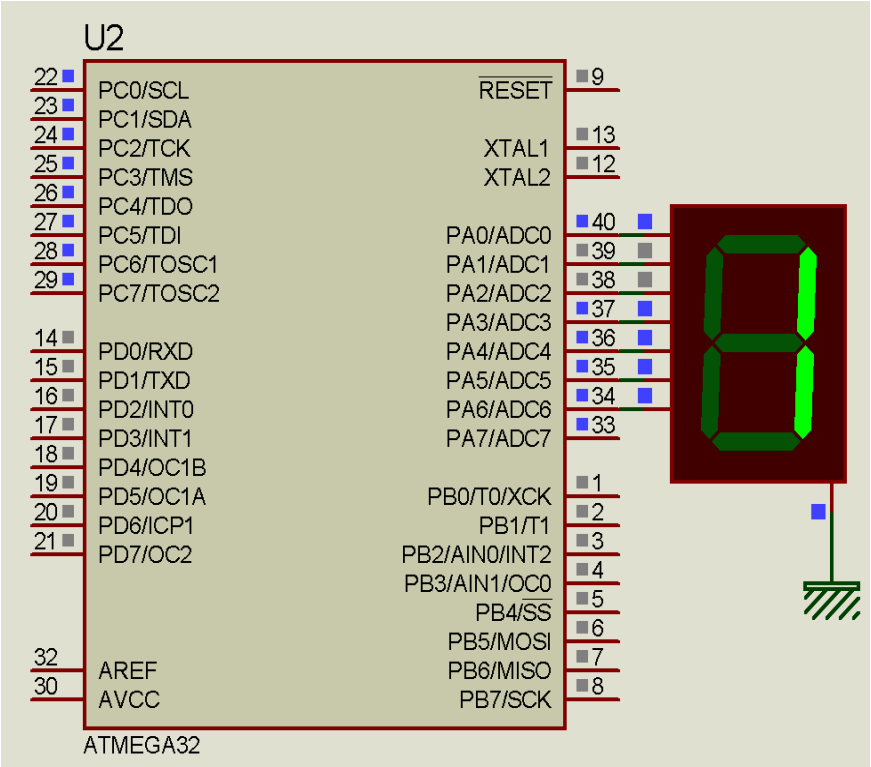
d[0]	d[1]	d[2]	d[3]	d[4]
7	12	0	99	20

آرایه	d[0]	d[1]	d[2]	d[3]	d[4]
مقدار	7	12	0	99	20
بعدازعملیات	8	11	60	100	12

```
d[4]= d[0]+5;

d[3]++;

d[2]= d[1]*5;

d[1]--;

d[0]=8;
```

مثال: مثال قبل را به کمک آرایه بنویسید.

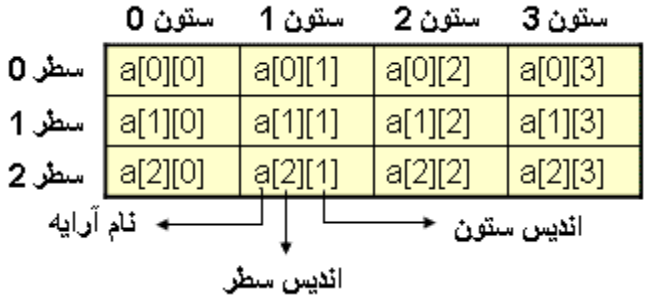
```
#include <mega32.h>
#include <delay.h>
unsigned char i;
unsigned char bts[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F ,0x6F};
void main (void)
{
DDRA=0xFF;
while(1)
{
for(i=0;i<10;i++)
{
PORTA=bts[i];
delay_ms(500);
}
}
}
```



آرایه های چند بُعدی: به شکل زیر تعریف می شوند.

نوع متغیر نام آرایه [سطر][ستون]={ {عناصر سطر اول} ,{عناصر سطر دوم} , ... };

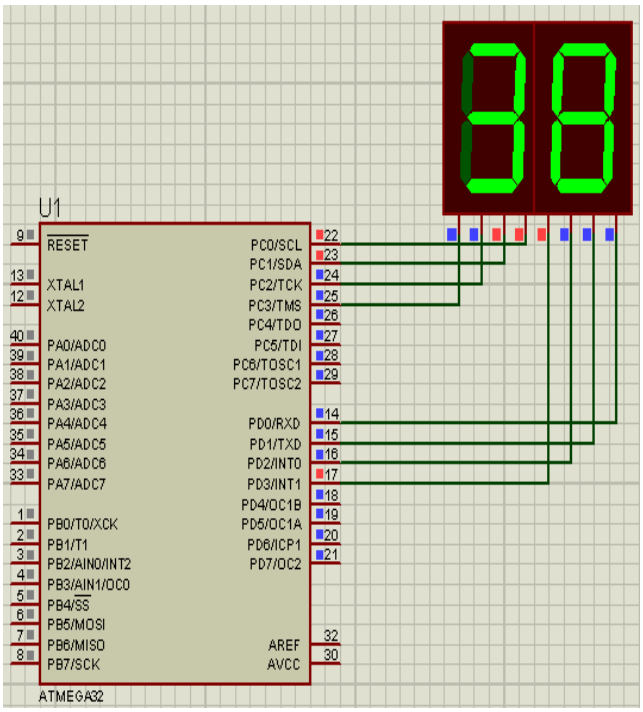
```
int a[3][4]={{1,5,3,4},{7,2,1,0},{1,2,3,4}};   یا   int a[3][4]={1,5,3,4,7,2,1,0,1,2,3,4};
```



مثال: دو عدد سون سگمنت BCD را به پورت A و B متصل کنید و یک شمارنده ۰ تا ۹۹ دسیمال بسازید.

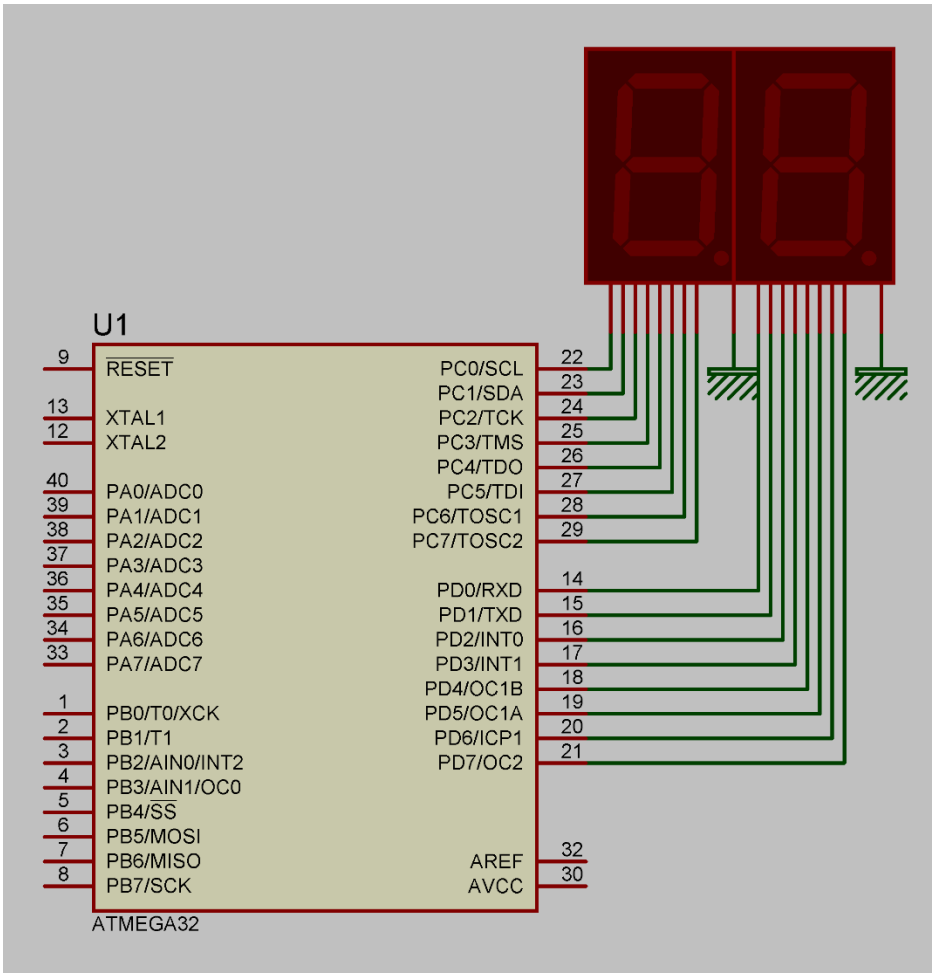
```
#include <mega32.h>
#include <delay.h>
unsigned char y=0,d=0;
void main (void)
{
DDRD=0xFF;
DDRC=0xFF;
while(1)
{
PORTC=d;
PORTD=y;
delay_ms(500);
y++;
if(y==10)
{
y=0;
d++;
if(d==10)  d=0;
}
}
}
```

```
#include <mega32.h>
#include <delay.h>
unsigned char i=0;
void main (void)
{
DDRD=0xFF;
DDRC=0xFF;
while(1)
{
For(i=0;i<100;i++)
{
PORTC=i/10;
PORTD=i%10;
delay_ms(500);
}
}
}
```

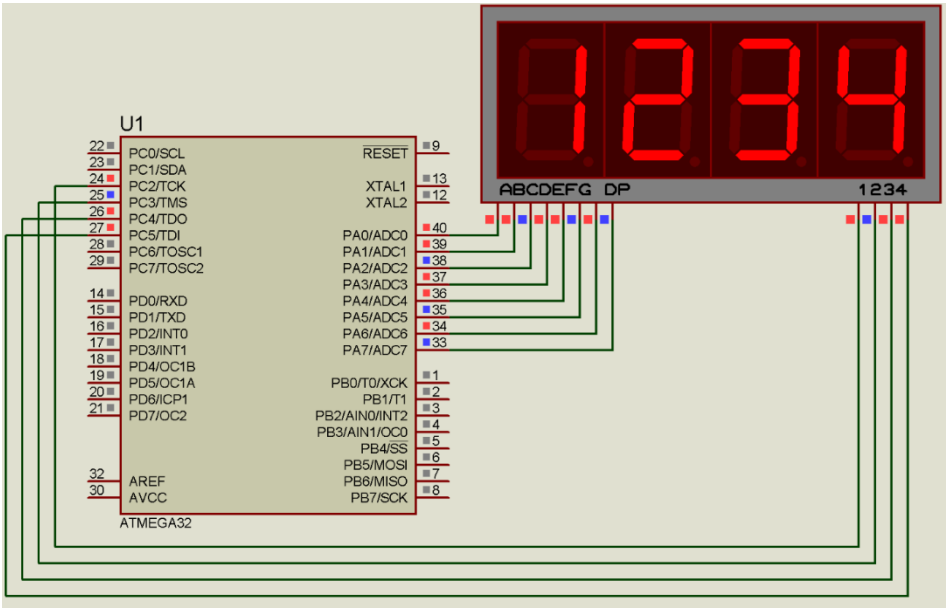


مثال: به جای سون سگمنت BCD سون سگمنت کاتد مشترک قرار دهید و مثال قبل را انجام دهید.

```
#include <mega32.h>
#include <delay.h>
unsigned char y=0,d=0;
unsigned char bts[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f ,0x6f};
void main (void)
{
DDRD=0xFF;
DDRC=0xFF;
while(1)
{
PORTD=bts[y];
PORTC=bts[d];
delay_ms(500);
y++;
if(y==10)
{
y=0;
d++;
if(d==10) d=0;
}
}
}
```



نمایشگر (Display) : در اکثر مدارها برای نمایش دیتای خروجی نیاز به یک نمایشگر داریم که تعداد رقم های آن برای خروجی ما مناسب باشد. فرض کنید در یک پروژه نیاز به چهار رقم در خروجی داریم؛ اگر قرار باشد که هر سون سگمنت را به یک پورت متصل کنیم، هر چهار پورت میکرو مشغول می شود و برای دستگاه های دیگر پورتی باقی نمی ماند. راه حل این مشکل این است که سون سگمنت را به شکل زیر ببندیم و آن ها را به روش مالتی پلکس روشن کنیم.

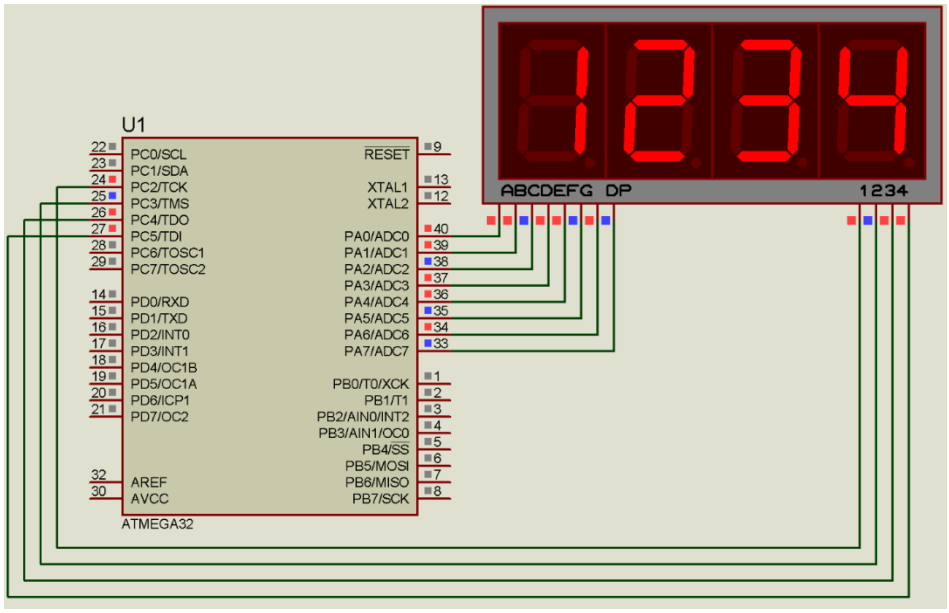


فرض کنید سون سگمنت ها کاتد مشترک است و می خواهیم عدد 1234 را بر روی آن نمایش دهیم. ابتدا با دادن 1 به کاتدها تمام رقم هارا خاموش می کنیم. سپس دیتای اولین رقم را می فرستیم. سپس کاتد اولین رقم را صفر می کنیم تا آن رقم روشن شود.برای مدتی آن را روشن نگه می داریم سپس خاموش می کنیم. حال رقم دوم را می فرستیم؛ کاتد آن را نیز صفر می کنیم. رقم دوم هم روشن می شود و پس از مدتی خاموش می کنیم. این کار را برای تمام ارقام انجام می دهیم. با توجه به سرعت چشم انسان که ۱۶ فریم بر ثانیه است و پسماند تصویر بر روی شبکیه ی چشم، اگر این کار را با سرعت مناسب انجام دهیم تمام ارقام را روشن خواهیم دید.

در مثال زیر عدد 1234 بر روی نمایشگر ، نمایش داده شده است.

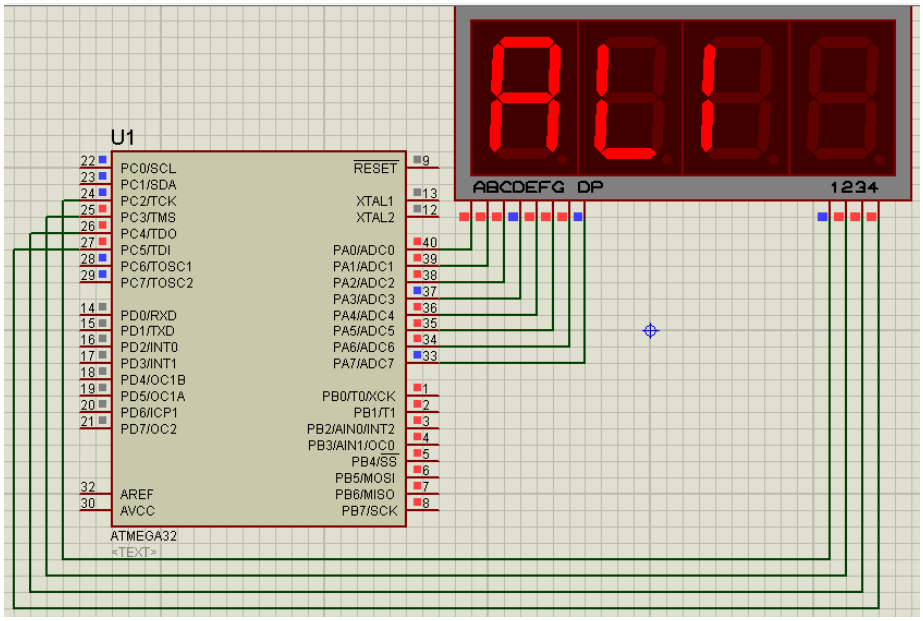
```
#include <mega32.h>
#include <delay.h>
unsigned char bts[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F ,0x6F};
void main (void)
{
DDRA=0xFF;
DDRC=0x3C; //0b00111100
while(1)
{
 PORTC=0x3C;   //    خاموش کردن تمام  رقم ها
 //------Digit1------
 PORTA=bts[1];   //    قرار دادن  کد عدد یک  روی پورت A
 PORTC.2=0;      //    روشن  کردن  رقم  اول
```

```
 delay_ms(5);      //    تاخیر برای دیدن رقم اول
 PORTC.2=1;        //    خاموش کردن رقم اول
//------Digit2------
 PORTA=bts[2];
 PORTC.3=0;
 delay_ms(5);
 PORTC.3=1;
//------Digit3------
 PORTA=bts[3];
 PORTC.4=0;
 delay_ms(5);
 PORTC.4=1;
//------Digit4------
 PORTA=bts[4];
 PORTC.5=0;
 delay_ms(5);
 PORTC.5=1;
}}
```



مثال: کلمه ALI را بر روی سون سگمنت ۴ تایی نمایش دهید.

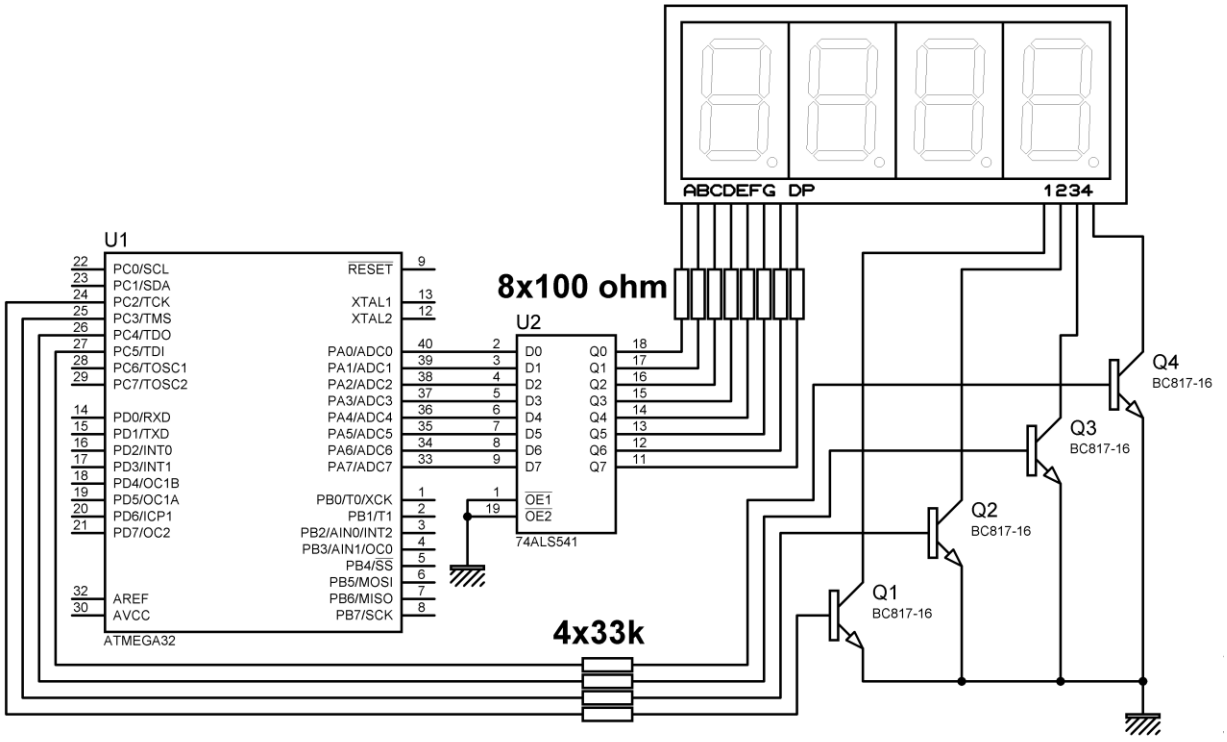
```
#include <mega32.h>
#include <delay.h>
void main(void)
{
 DDRC=0x3C;
 DDRA=0xff;
while(1)
 {
  PORTC=0x3C;
  //---DIGIT1----
  PORTA=0x77;
  PORTC.2=0;
  delay_ms(5);
  PORTC.2=1;
  //---DIGIT2----
  PORTA=0x38;
  PORTC.3=0;
  delay_ms(5);
  PORTC.3=1;
  //---DIGIT3----
  PORTA=0x30;
  PORTC.4=0;
  delay_ms(5);
  PORTC.4=1;
  //---DIGIT4----
```



```
PORTA=0;
PORTC.5=0;
delay_ms(5);
PORTC.5=1;
}
}
```

درایور نمایشگر :

با توجه به این که در مدار نمایشگر بالا ، کاتد هر رقم به یکی از بیت های PORTC متصل شده ، و جریان عبوری از کاتد در حالتی که تمام سگمنت ها و نقطه اعشار روشن باشد ، در حدود 80ma خواهد شد و با توجه به این که جریان قابل تحمل هر بیت از پورت حداکثر 20ma است پس در عمل نمی توان کاتد را مستقیم به پورت متصل کرد، و نیاز به یک درایور داریم مدار زیر که در برد آزمایش نیز استفاده شده است می تواند یک نمونه درایور برای نمایشگرهای سون سگمنتی باشد.



تمرین: برنامه ای بنویسید که یک شمارنده ۰ تا ۹۹۹۹ بر روی نمایشگر داشته باشیم.

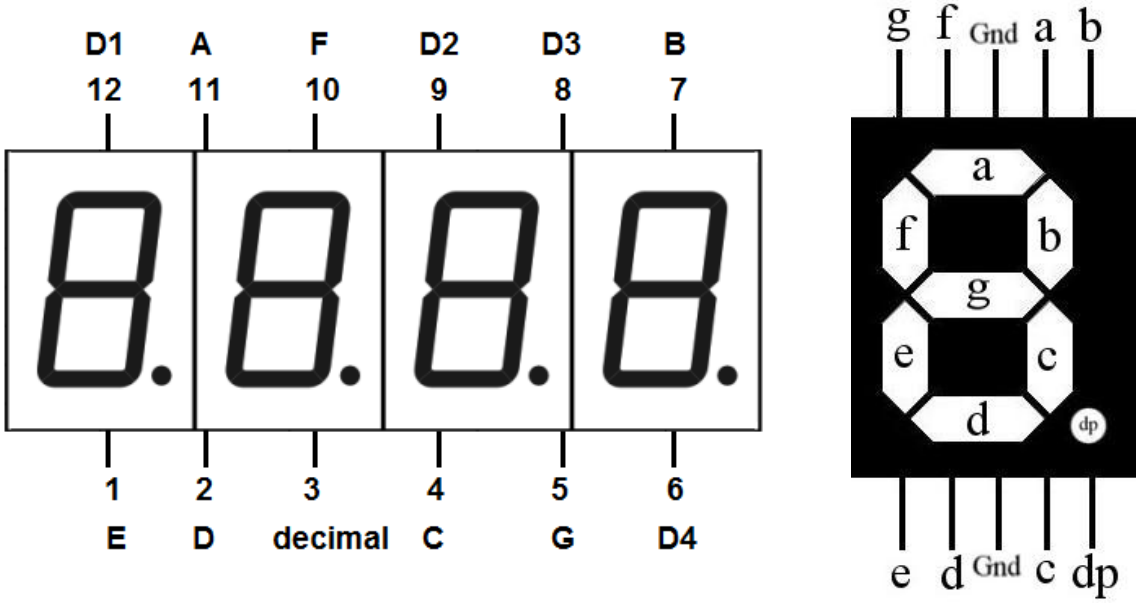
تمرین: برنامه ای بنویسید که به طور هم زمان دو شمارنده بالا شمار ۰ تا ۹۹ و شمارنده پایین شمار ۹۹ تا ۰ بر روی نمایشگر داشته باشیم .

تمرین: برنامه ای بنویسید که ثانیه و دقیقه شمار را بر روی نمایشگر داشته باشیم . در ضمن اعداد دقیقه و ثانیه را با روشن کردن نقطه اعشار یکان دقیقه از هم جدا کنید.

تمرین: برنامه ای بنویسید که کلمه ALI روی نمایشگر حرکت کند(تابلوی روان) .

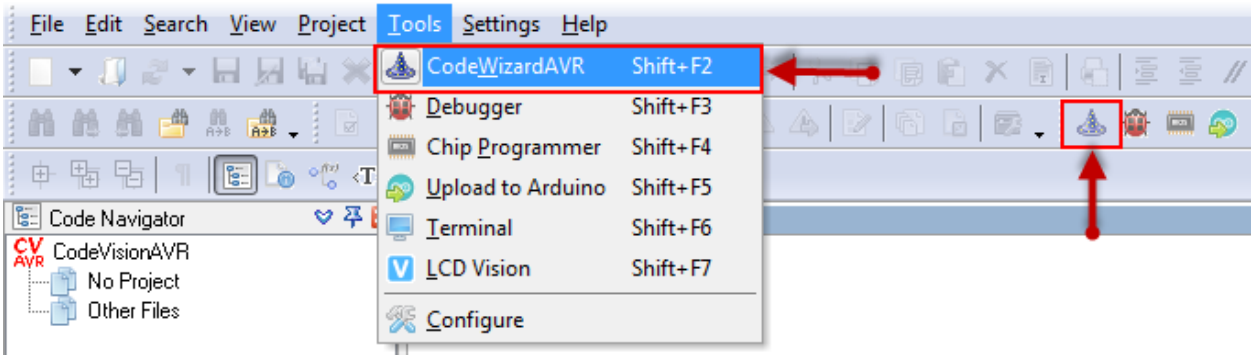
تمرین: در نرم افزار پروتئوس یک نمایشگر هشت رقمی را به میکرو متصل کنید و کلمه ALI را حرکت دهید . برنامه را طوری بنویسید که حرکت بصورت رفت و برگشت باشد.

❊ شماره پایه ها و نحوه قرارگیری آن ها در قطعات واقعی به شکل زیر است :

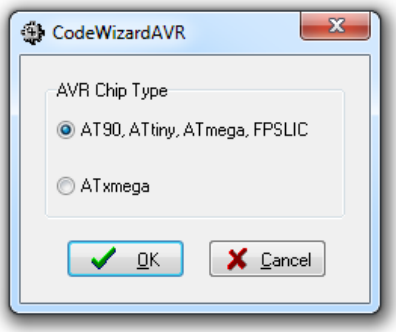


مراحل ایجاد پروژه با استفاده از ویزارد WIZARD :

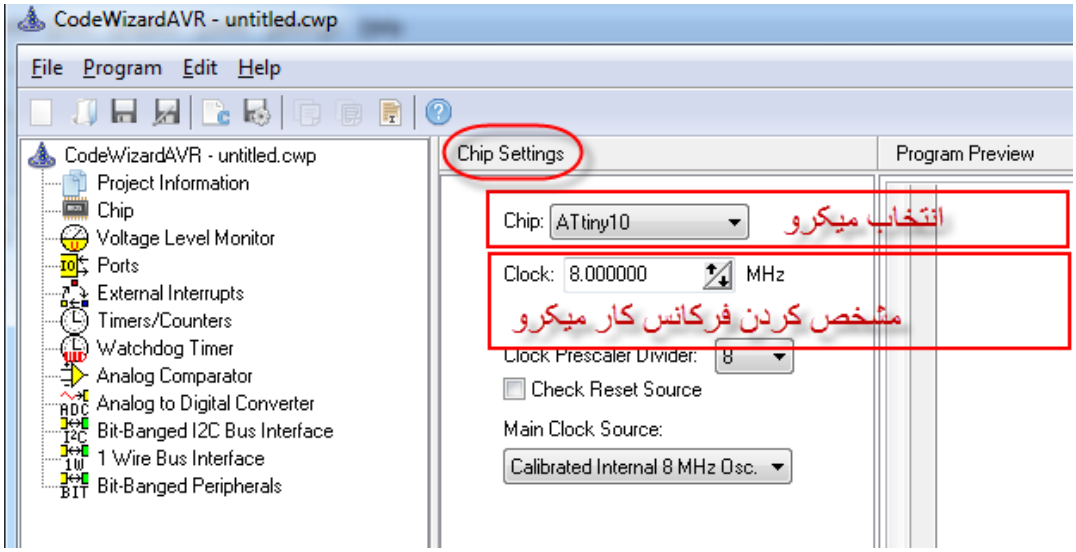
۱- از منوی Tools یا نوار ابزار Code Wizard AVR را انتخاب کنید.



۲- در پنجره CodeWizardAVR که باز می شود ATmega انتخاب شده است با کلیک روی OK به مرحله بعدی می رویم.

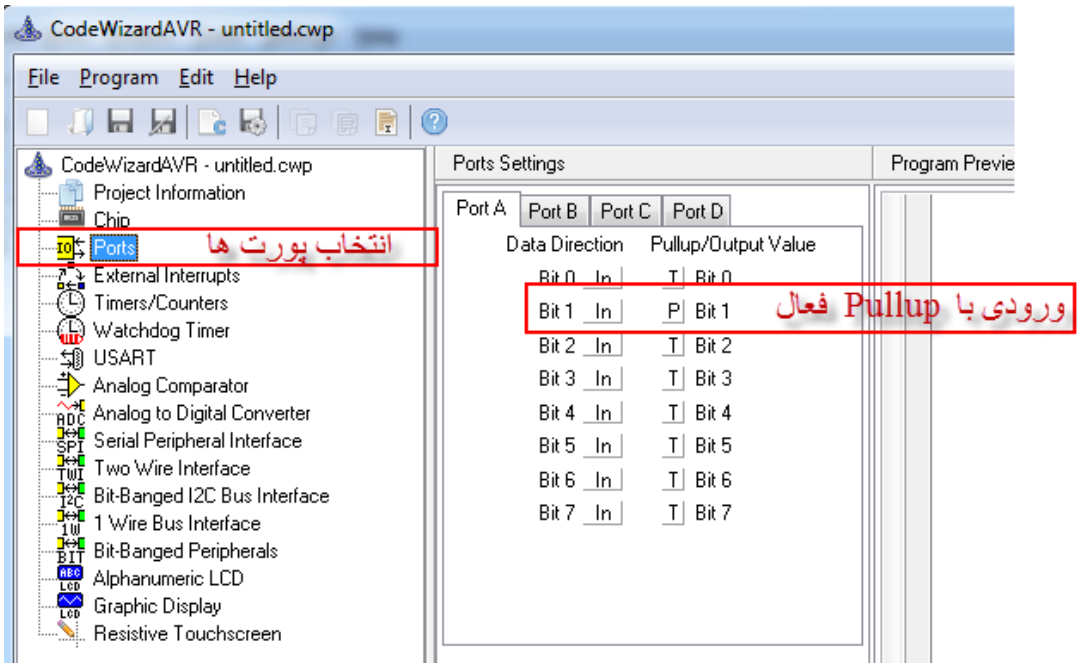


۳- پنجره Wizard در حالی که برگه Chip انتخاب شده باز می شود دراین برگه شماره میکرو و فرکانس کاری آن را مشخص می کنیم.

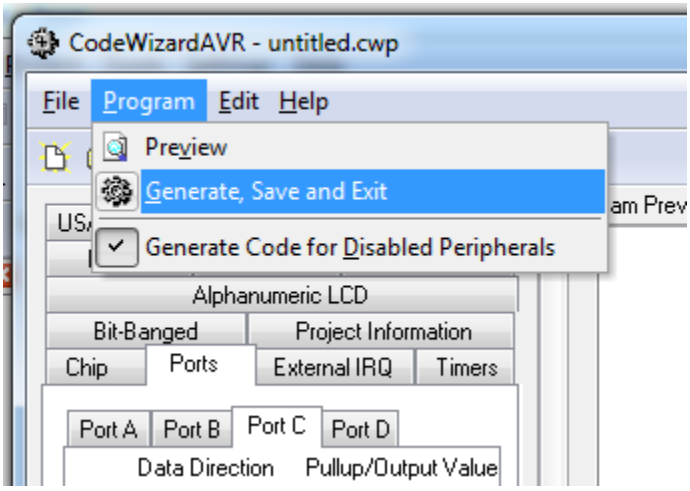


۴- با توجه به نیاز پروژه ، برگه های دیگر را باز و تنظیم های لازم را انجام می دهیم.

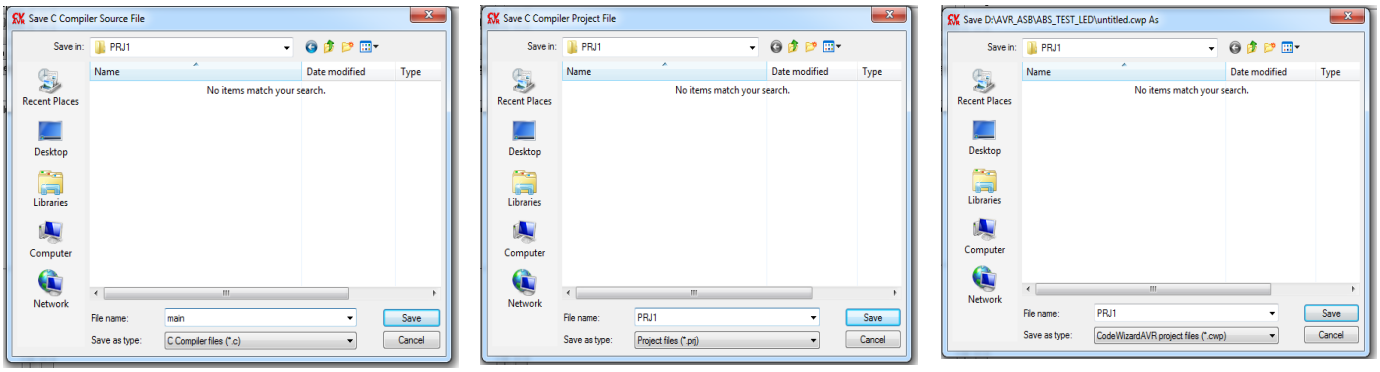
*نکته: در برگه پورت ها اگر پورتی ورودی تنظیم شود و در قسمت pull_up حرف T باشد یعنی این پایه HI_Z است و منطق لاجیکی آن بستگی به ورودی دارد . اگر حرف T را با کلیک به P تبدیل کنید یعنی این پایه از داخل میکرو pull_up شده و منطق یک دارد.



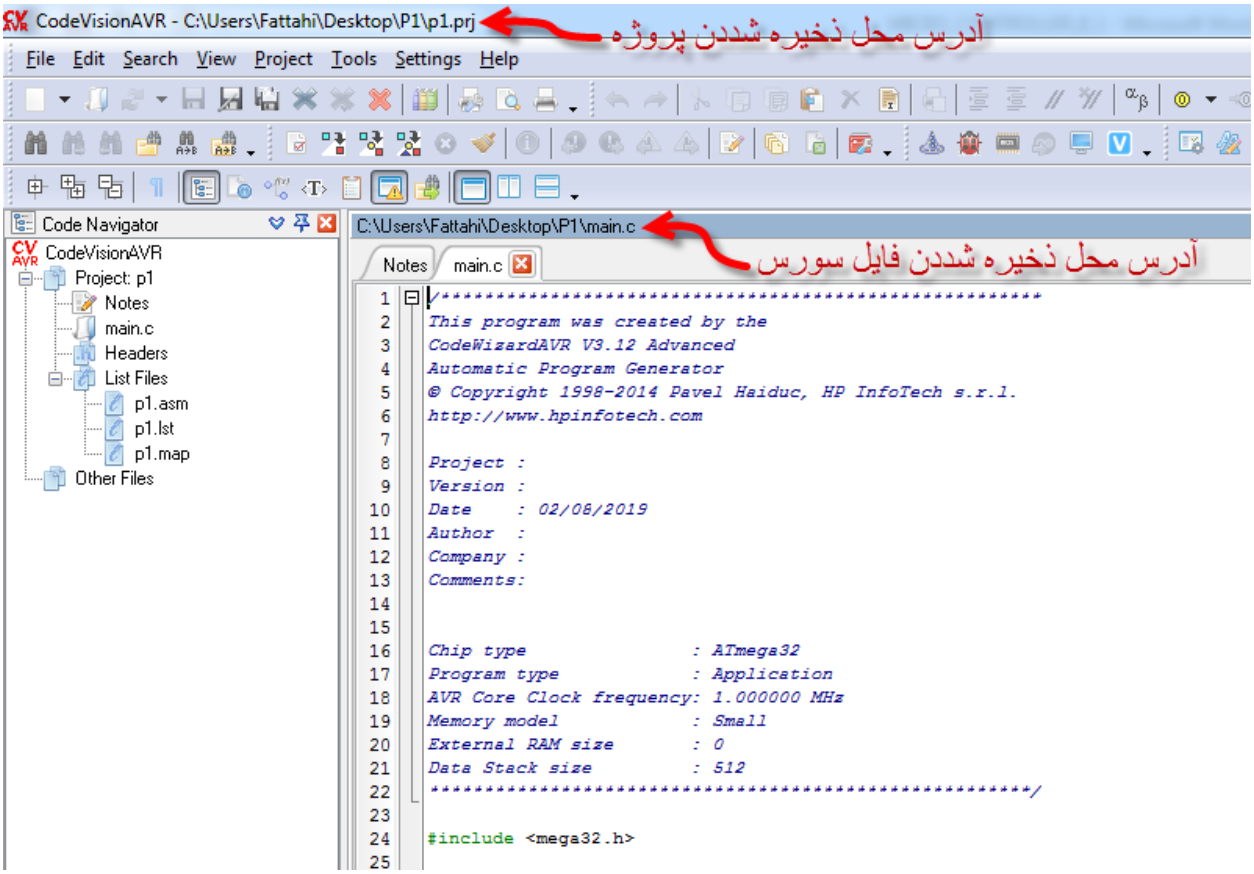
۵- پس از اتمام تنظیمات ، از مسیر زیر پروژه و فایل های مربوط به آن را با نام مناسب و در محلی مشخص ذخیره می کنیم. Program/Generate,Save and Exit



*نکته: در این مرحله لازم است سه بار فایل ذخیره کنیم ، Source --- Project --- CWP

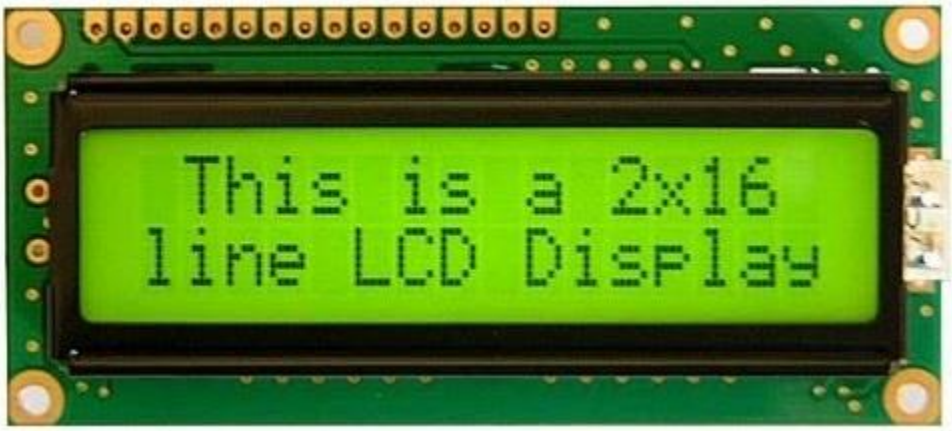


۶- پس از ذخیره فایل ها ، کد برنامه با توجه به تنظیمات ویزارد باز می شود حال می توان برنامه مورد نظر را به آن اضافه کرد.

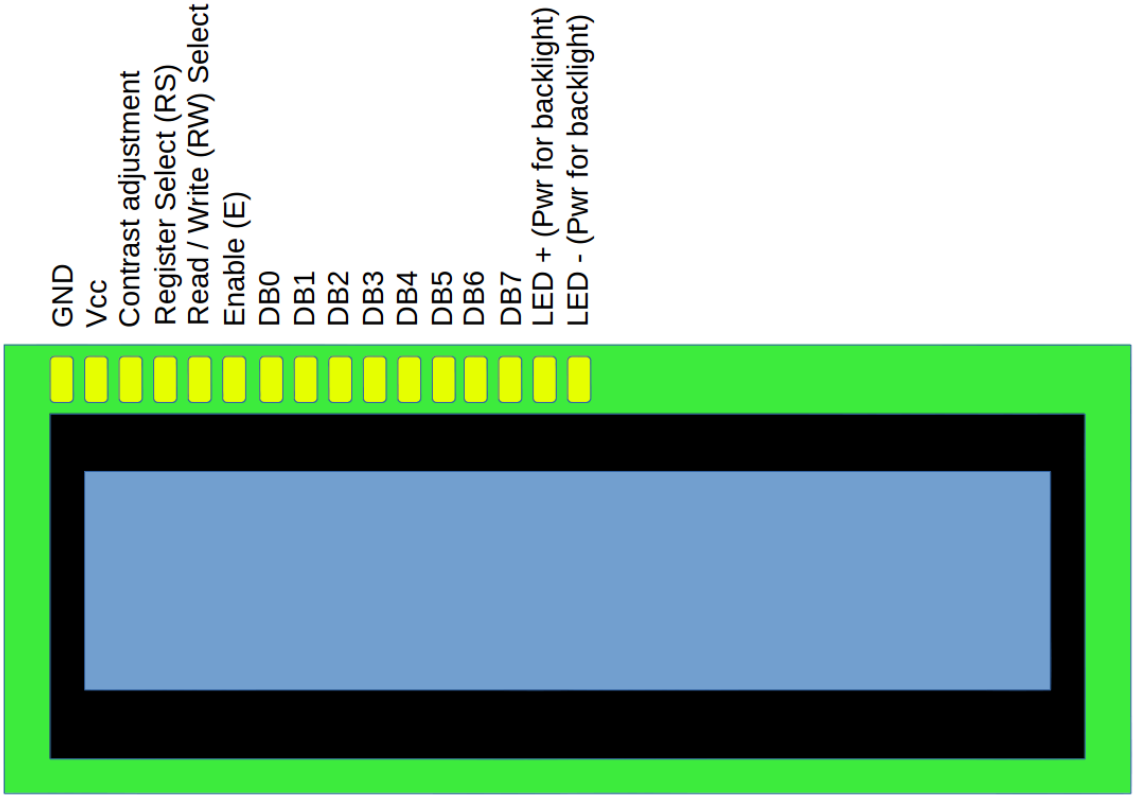


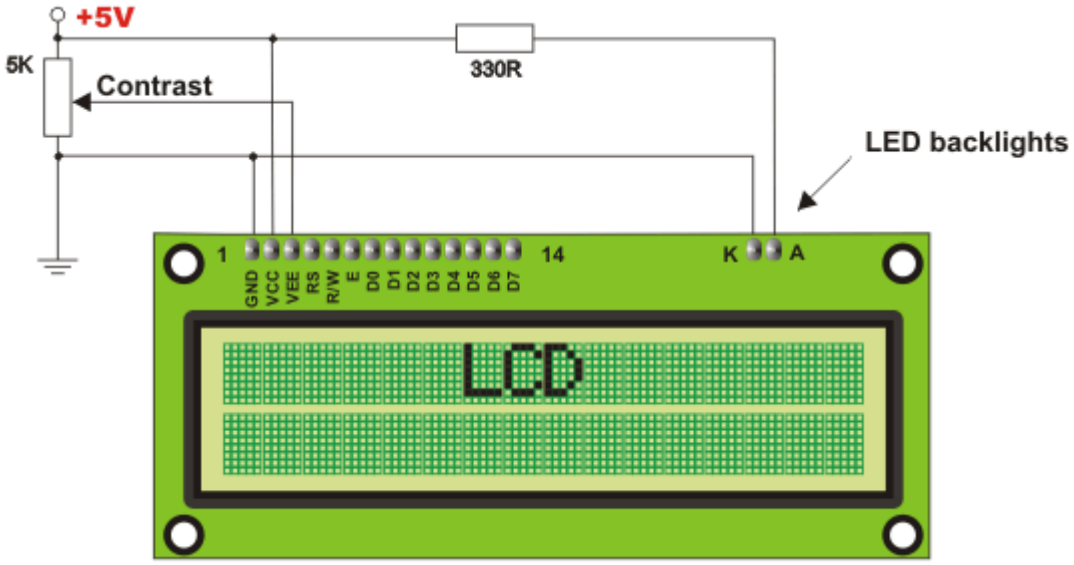
LCD (Liquid-Crystal Display): یکی دیگر از دستگاه های خروجی LCD کاراکتری می باشد که می توانید بر روی آن متن و اعداد را نمایش دهید. در LCD کاراکتری پارامتر های مهم تعداد خط و تعداد کاراکتر در هر خط است.

نوع ۱۶*۲ آن پر کاربردتر است.



پایه های LCD: معمولا ۱۶پایه به شرح زیر دارد.





A(16), K(15), D7(14),…,D0(7), E(6), RW(5), RS(4), VEE(3), VDD(2), VSS(1)

A , K: برای کنترل نور پس زمینه (Back Light) می باشد.

D7,…,D0: دستور و دیتا را دریافت می کند.

RS: اگر صفر باشد خطوط انتقال دیتا ،دستور و اگر یک باشد دیتا دریافت می کنند.

RW: اگر یک باشد یعنی Read و اگر صفر باشد یعنی Write.

E: با اعمال یک لبه ی پایین رونده می توان اجازه ورود دیتا یا دستور را صادر کرد.

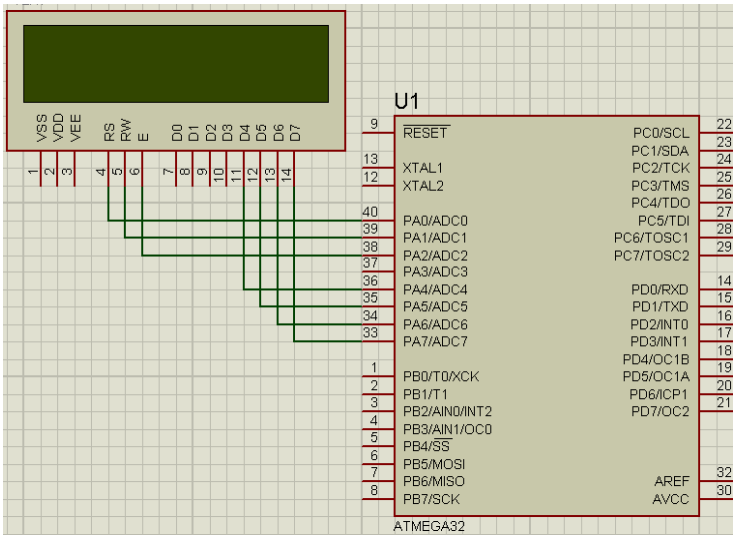
VEE: با متصل کردن یک پتانسیومتر 10k به این پایه می توان نور پشت صفحه Back Light را کنترل کرد.

VSS , VDD: تغذیه LCD از این پایه ها تامین می شود.

نکته: LCD های کاراکتری را می توان با ۴ خط دیتا را ه اندازی کرد (خطوط D4 تا D7). برای مثال در شکل زیر اتصال یک LCD به PORTA را مشاهده می کنید.

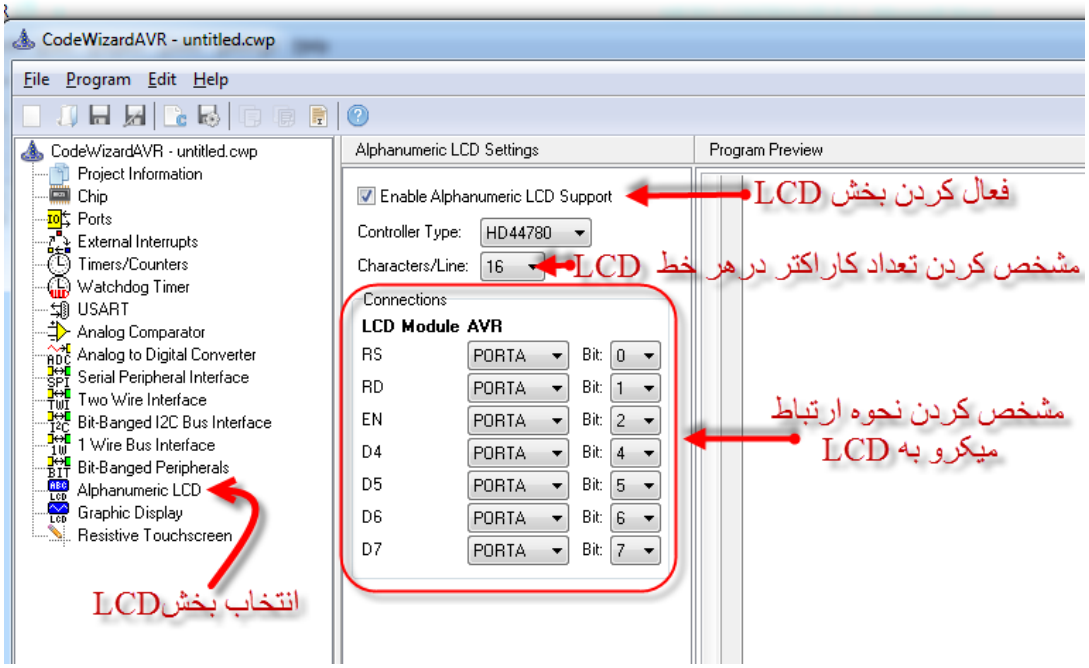
کد ویژن از این نوع اتصال

پشتیبانی می کند.



مثال: یک LCD را به پورت A متصل کنید. برنامه ای بنویسید که وسط خط اول کلمه IRAN و وسط خط دوم 1394 نوشته شود.

ابتدا در ویزارد برگه Alphanumeric LCD را باز و آن را فعال می کنیم. در قسمت Character/Line مشخص می کنیم که در هر خط چند کاراکتر وجود دارد. در قسمت Connections نحوه اتصال LCD به میکرو مشخص شده که قابل ویرایش است.



پس از تولید کد ،کتابخانه <alcd.h> به برنامه اضافه می شود که توابع کار با LCD در این کتابخانه قرار دارد.

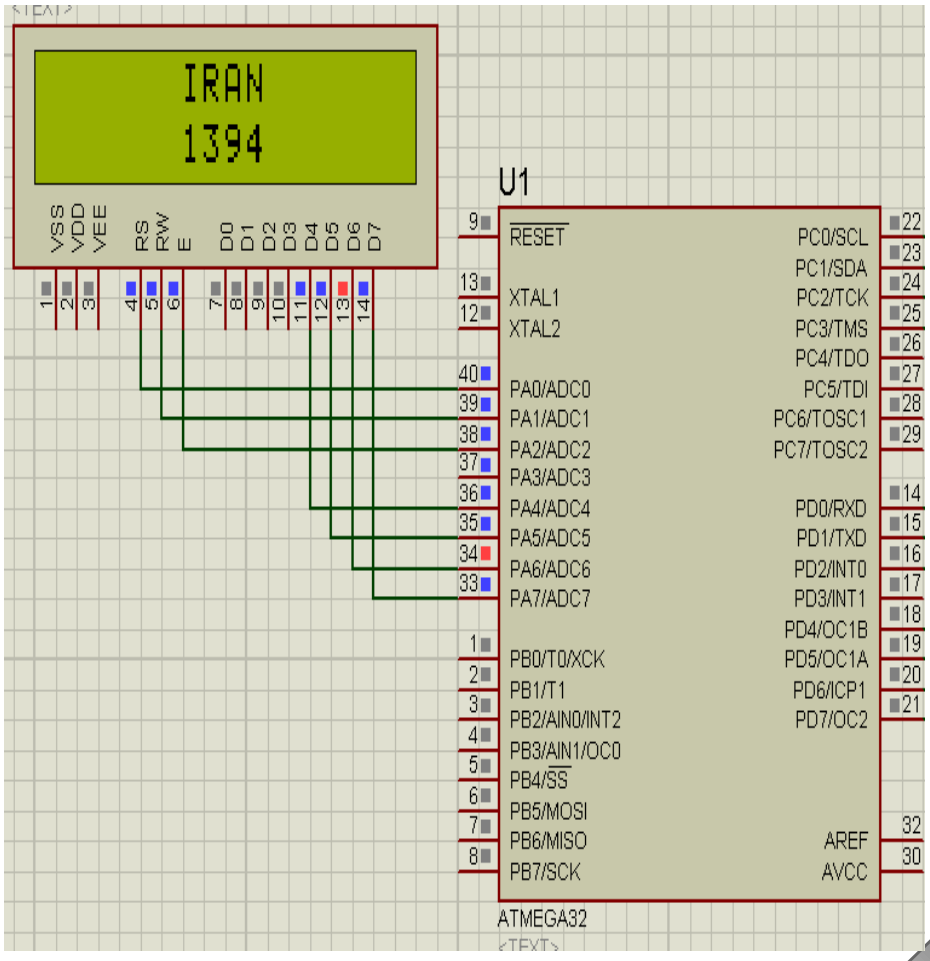
lcd_clear(); پاک کردن صفحه نمایش

lcd_gotoxy(x,y); مشخص کردن ستون و سطری که نوشتن از آنجا شروع می شود

Y=0 ⟶	X=0	X=1	…….	X=15
Y=1 ⟶	X=0	X=1		X=15

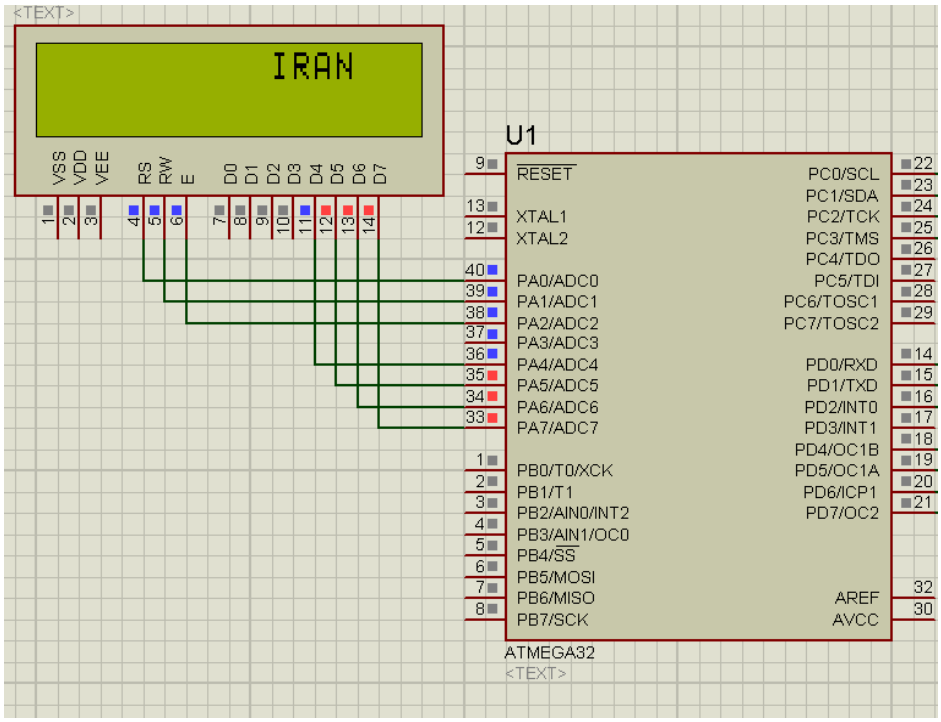
Lcd_putchar('کاراکتر'); Lcd_putsf("رشته"); Lcd_puts(متغیر رشته ای);

```
char s[ ]="IRAN";
void main(void)
{
lcd_init(16);
lcd_clear();
lcd_gotoxy(6,0);
lcd_puts(s);
lcd_gotoxy(6,1);
lcd_putsf("1394");
while (1)
{
}
}
```



مثال: کلمه IRAN را از ابتدای خط اول تا انتهای خط اول حرکت دهید.

```
#include <delay.h>
unsigned char x;
void main(void)
{
while (1)
{
for(x=0;x<13;x++)
{
lcd_clear();
lcd_gotoxy(x,0);
lcd_putsf("IRAN");
delay_ms(500);
}
}
}
```



کد اسکی (ASCII): American Standard Code Information Interchange

با توجه به اینکه کامپیوتراز بخش های مختلفی ساخته شده و لازم است که این بخش ها به یکدیگر اطلاعات ارسال و دریافت نمایند، باید کدهایی استاندارد و معرفی می شد تا تمام سازندگان کامپیوتر و دستگاه های جانبی از آن پیروی کنند. لذا انجمنی در آمریکا این کار را انجام داد و کد تولیدی را به نام ASCII معرفی کرد. در LCD های کاراکتری نیز از کد ASCII استفاده می شود.در زیر کدهای ASCII برخی کاراکترها و جدول کدهای ASCII آمده است .

A=65 a=97 0=48 Enter=13 Space=32 Esc=27

The ASCII code
American Standard Code for Information Interchange

www.theasciicode.com.ar

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

theASCIIcode.com.ar

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ö	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

اگر بخواهیم عددی را به LCD بفرستیم لازم است که ابتدا آن را توسط دستورهای موجود به کد ASCII تبدیل کنیم و سپس ارسال نماییم. برای این کار توابعی وجود دارد که یکی از آن ها تابع sprintf موجود در کتابخانه ی <stdio.h> می باشد. ساختار این تابع به شکل زیر است.

sprintf (متغیر رشته ای , "متن و نوع متغیر هاونحوه چاپ" , متغیر عددی)

برای تعریف متغیر رشته ای به شکل زیر عمل کنید.

مثال	تعریف متغیر رشته ای
char s[6];	char نام رشته [طول رشته];

برای مشخص کردن نوع و نحوه چاپ اعداد از علامت % به همراه کاراکترهای خاص استفاده می شود . برای مثال از علامت %d یا %i برای مشخص کردن اعداد صحیح و از %f برای اعداد اعشاری استفاده می‌شود

برای مشخص کردن نحوه چاپ علاوه بر نوع متغیر از اعداد نیز در کنار آ ن ها استفاده می شود برای مثال %02d یعنی این که عدد بصورت دو رقمی چاپ شود حال اگر عدد یک رقمی باشد با قرار دادن یک 0 قبل از عدد ، آن را در دو رقم چاپ می کند . برای مثال اگر مقدار متغیر 3 باشد روی lcd این عدد 03 دیده خواهد شد.

همچنین می‌توانید از دو تابع itoa و ftoa که در کتابخانه ی <stdlib.h> قرار دارند نیز استفاده کنید.

۱- itoa برای اعداد صحیح itoa (متغیر عددی , متغیر رشته ای)

۲- ftoa برای اعداد اعشاری ftoa (متغیر عددی , تعداد اعشار , متغیر رشته ای)

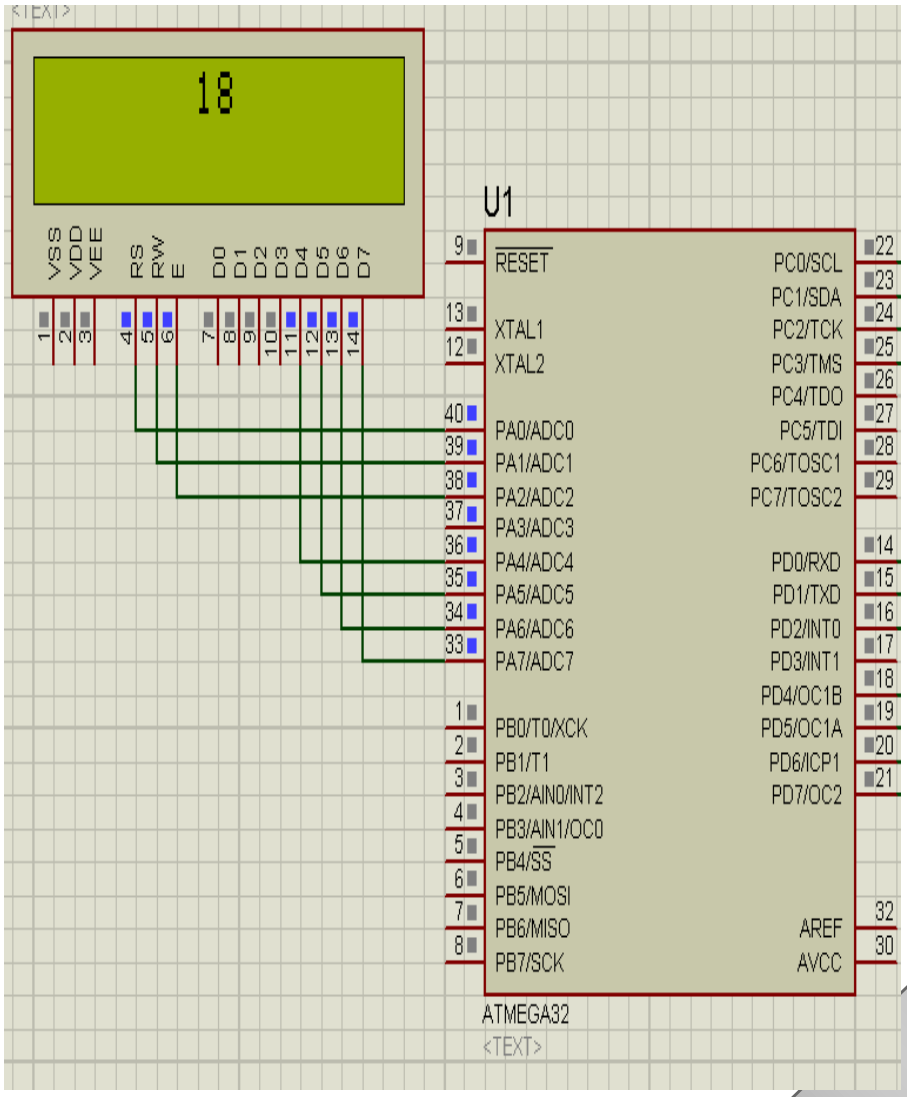
تعریف متغیر رشته ای char نام رشته [طول رشته];

نکته: بهتر است طول رشته از تعداد ارقام بزرگترین عددی که می خواهیم نمایش دهیم ۲ خانه بیشتر باشد . برای مثال اگر بزرگترین عدد 1023 (چهار رقم) است طول رشته را ۶ خانه وارد کنید.

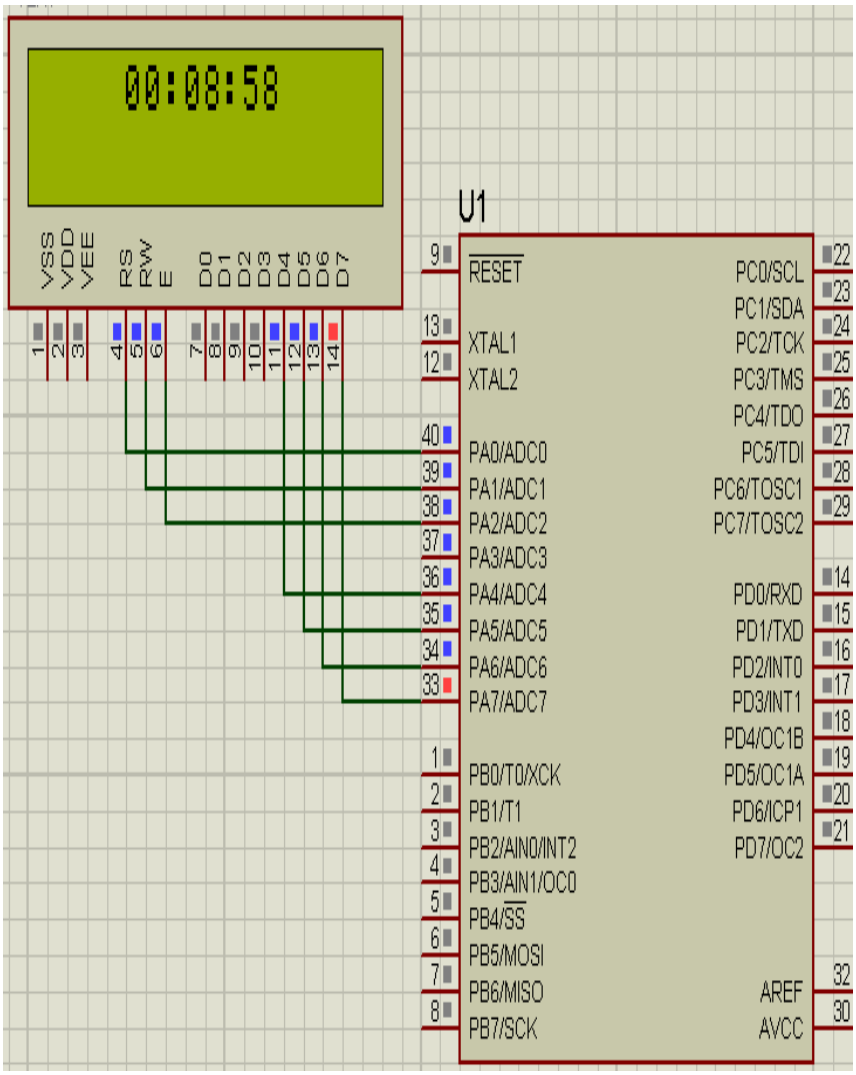
```
char  s[6];
```

مثال: بر روی LCD یک شمارنده ۰ تا ۱۹ بسازید.

```
#include <delay.h>
#include <stdlib.h>
unsigned char i;
char s[4];
void main(void)
{
while (1)
{
for(i=0;i<20;i++)
{
itoa(i,s);
lcd_gotoxy(7,0);
lcd_puts(s);
lcd_putsf(" ");
delay_ms(500);}}}
```

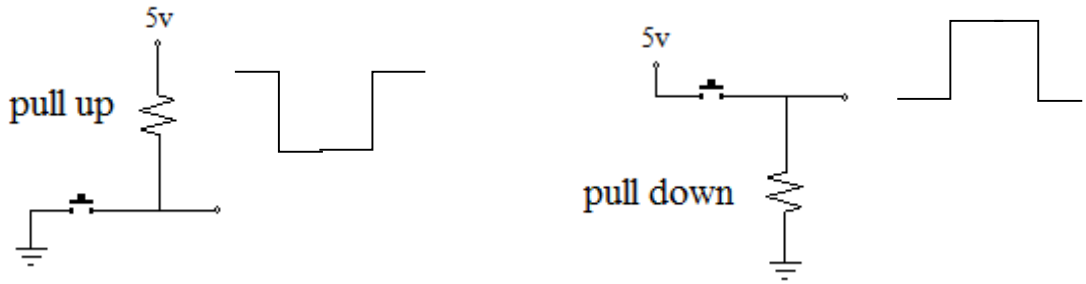


مثال: یک ساعت بر روی LCD طراحی کنید.

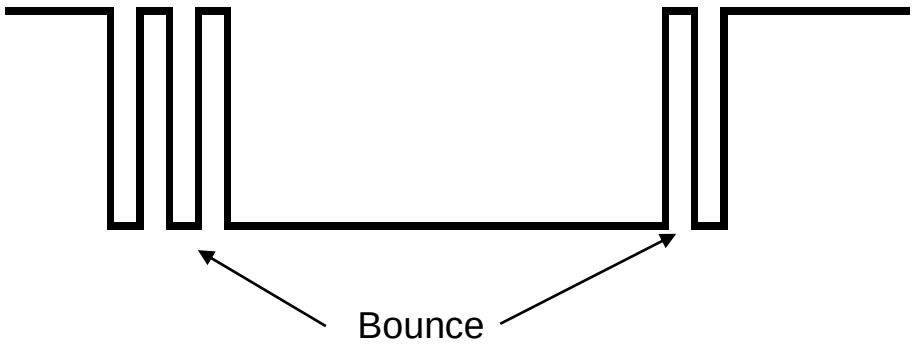



```
#include <stdio.h>
#include <delay.h>
unsigned char h,m,s;
char t[20]
void main(void)
{
lcd_clear();
while (1)
{
s++;
if(s==60)
{
s=0;
m++;
if(m==60)
{
m=0;
h++;
if(h==24) h=0;
}}
sprint(t,"%02d: %02d: %02d",h,m,s);
lcd_gotoxy(4,0);
lcd_puts(t);
delay_ms(1000);
}
```

کلید (key) : یکی از ساده ترین و پرکاربردترین دستگاه های ورودی کلید است. توسط کلید می توان صفر یا یک را تولید و به مدار اعمال کرد. معمولا دو مدار زیر برای کلیدها بسته می شوند.



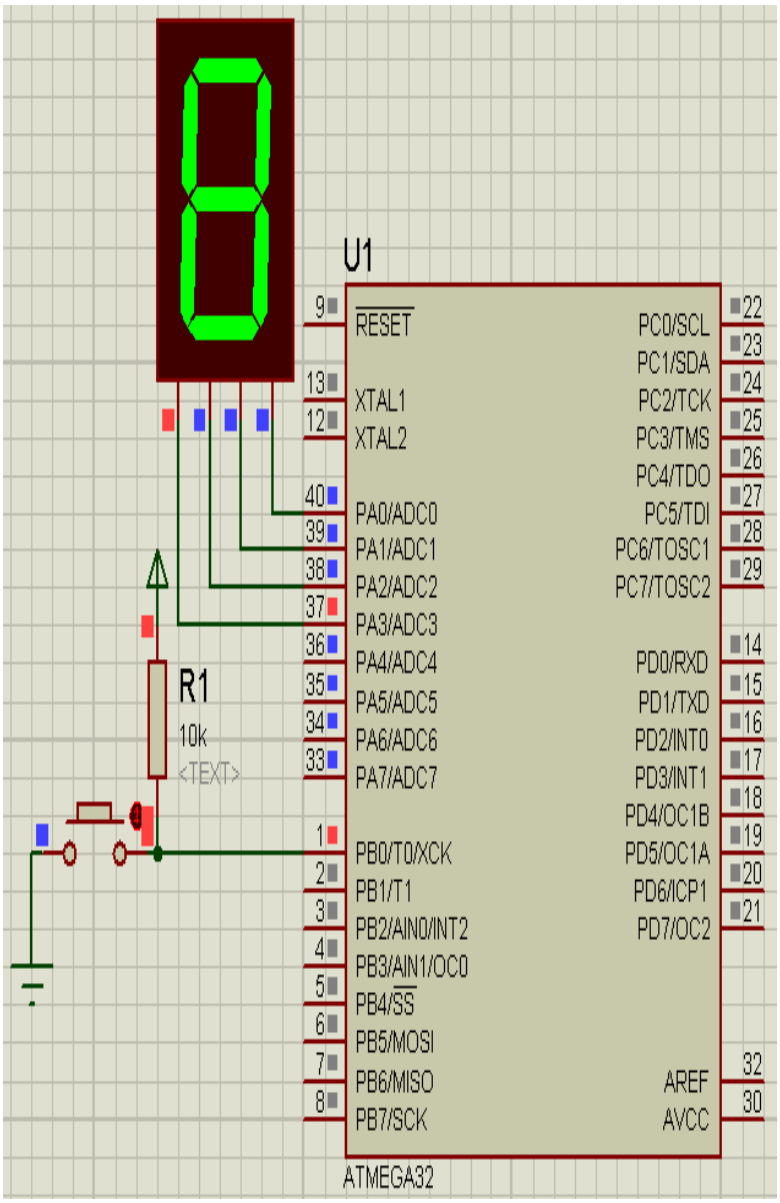
شکل های نشان داده شده بالا برای خروجی در حالت ایده آل است اما در عمل به علت لرزش یا bounce هنگام قطع و وصل کلید پالس های اضافی خواهیم داشت که کار مدار را دچار مشکل خواهد کرد.



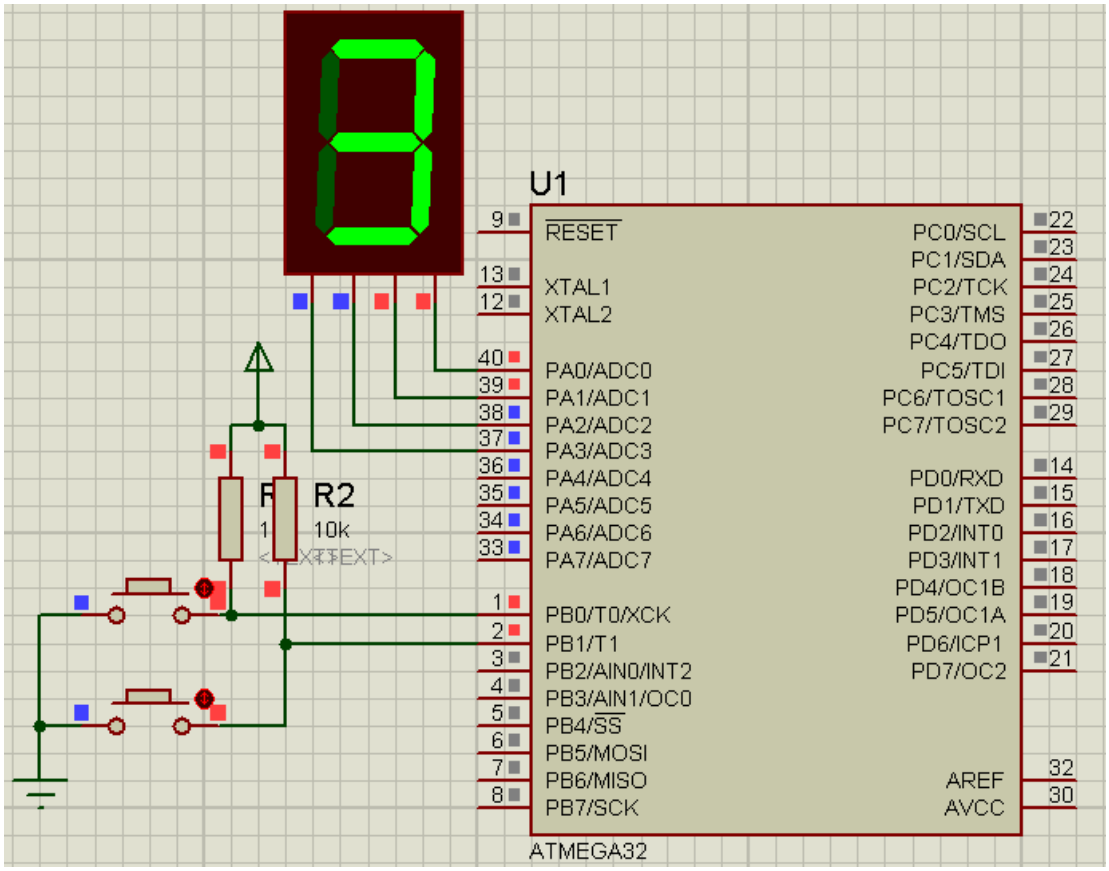
برای رفع این مشکل پس از هر تغییر در خروجی کلید حدود ۲۰ تا ۳۰ میلی ثانیه تاخیر ایجاد می کنیم تا لرزش ها تمام شود.

مثال: یک سون سگمنت BCD و یک کلید را به میکرو وصل می کنیم. برنامه ای بنویسید که با هربار زدن کلید یک واحد به عدد خروجی اضافه شود. در ضمن حلقه شمارش ۰ تا ۹ باشد و بعد از ۹ صفر شود.

```
#include <mega32a.h>
#include <delay.h>
unsigned char   i=0;
void main (void)
{
DDRA=0xFF;
PORTA=0x00;
DDRB=0x00;
while(1)
{
if(PINB.0==0)
{
delay_ms(25);
i++;
if( i==10) i=0 ;
PORTA=i;
while(PINB.0==0);
delay_ms(25);
}
}
}
```



مثال: یک کلید دیگر به مثال قبلی اضافه کنید و با آن عدد خروجی را کاهش دهید.



برای این کار کافی است برنامه زیر را به برنامه قبلی اضافه کنیم:

```
if(PINB.1==0)
{
delay_ms(25);
i--;
if(i==255) i=9;
PORTA=i;
while(PINB.1==0);
delay_ms(25);
}
```

مثال: مثال قبل را طوری تغییر دهید که هنگام زیاد شدن ، عدد به ۹ که رسید متوقف شود و هنگام کم شدن به صفر که رسید، متوقف شود.

برای این کار کافی است خط پنجم هر دو if را به صورت زیر بنویسیم.

```
if(i==10) i=9;

if(i==255) i=0;
```

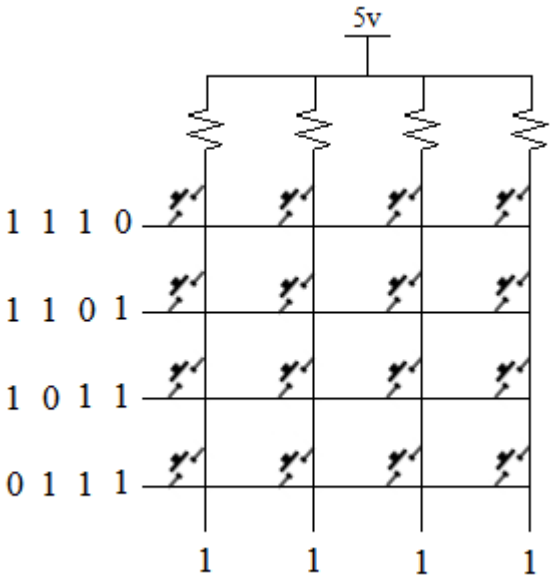
تمرین: یک کلید به PB.0 و یک Buzzer به PD.7 متصل است برنامه ای بنویسید که با نگه داشتن کلید ، Buzzer روشن و بوق بزند و با رها شدن Buzzer قطع شود.

تمرین: دو عدد کلید به PD.2 و PD.3 و یک Buzzer به PD.7 متصل است برنامه ای بنویسید که با زدن کلید PD.2، Buzzer روشن و بوق بزند و با کلید PD.3 قطع شود.

تمرین: هشت عدد led به PORTA و دو عدد کلید به PD.2 و PD.3 متصل کنید برنامه ای بنویسید که با نگه داشتن یکی از کلید ها روی PORTA شمارنده حلقوی و با کلید دیگر شمارنده جانسون ، نمایش داده شود .توجه داشته باشید با رها شدن کلید ، نمایش متوقف شود.

تمرین: یک lcd به PORTA و دو عدد کلید به PD.2 و PD.3 متصل کنید برنامه ای بنویسید که با نگه داشتن یکی از کلید ها عدد روی lcd با سرعت مناسب زیاد و با رسیدن به عدد 20 متوقف شود و دیگر زیاد نشود و با کلید دیگر کاهش یابد و با رسیدن به عدد 0 متوقف شود. در ضمن با هر بار افزایش یا کاهش عدد ، Buzzer متصل به PD.7 نیز بوق کوتاهی بزند.

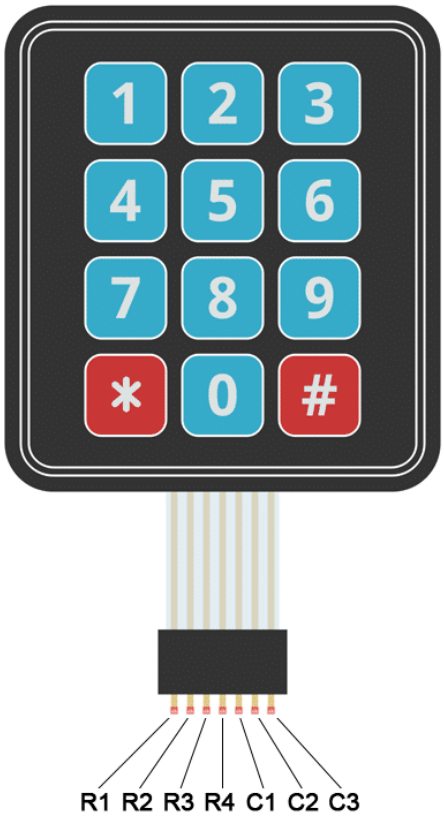
تمرین: یک lcd به PORTA و سه عدد کلید به PD.2 و PD.3 و PB.0 و یک Buzzer به PD.7 متصل کنید برنامه ای بنویسید که روی خط اول lcd ساعت و روی خط دوم lcd زمان آلارم را داشته باشیم . کاربر بتواند با کلید PB.0 مشخص کند کدام پارامتر باید تغییر کند و روی lcd آن عدد چشمک زن شود ، و با دو کلید PD2,3 عدد کم یا زیاد شود . و هر گاه زمان تنظیم شده آلارم با ساعت برابر شد Buzzer بطور مناسب بوق بزند.

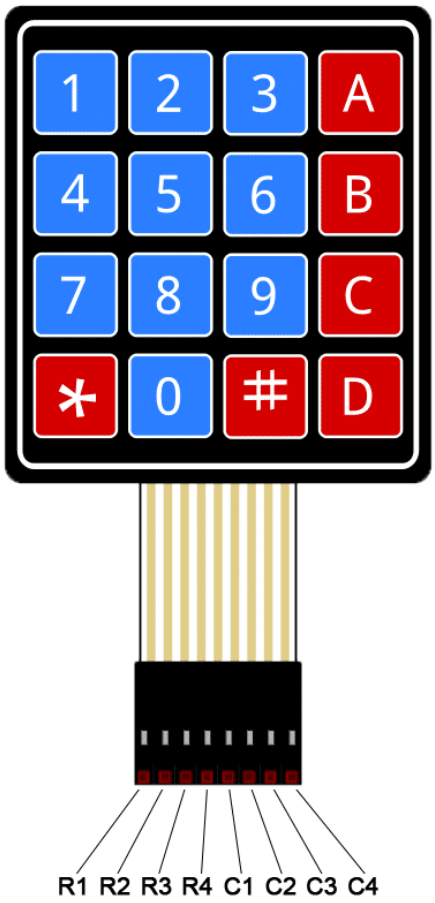


صفحه کلید Key Pad: یکی دیگر از وسایل ورودی صفحه کلید می باشد که در آن کلیدها به صورت ماتریسی چیده شده اند.

برای خواندن صفحه کلید به شکل زیر عمل شود:

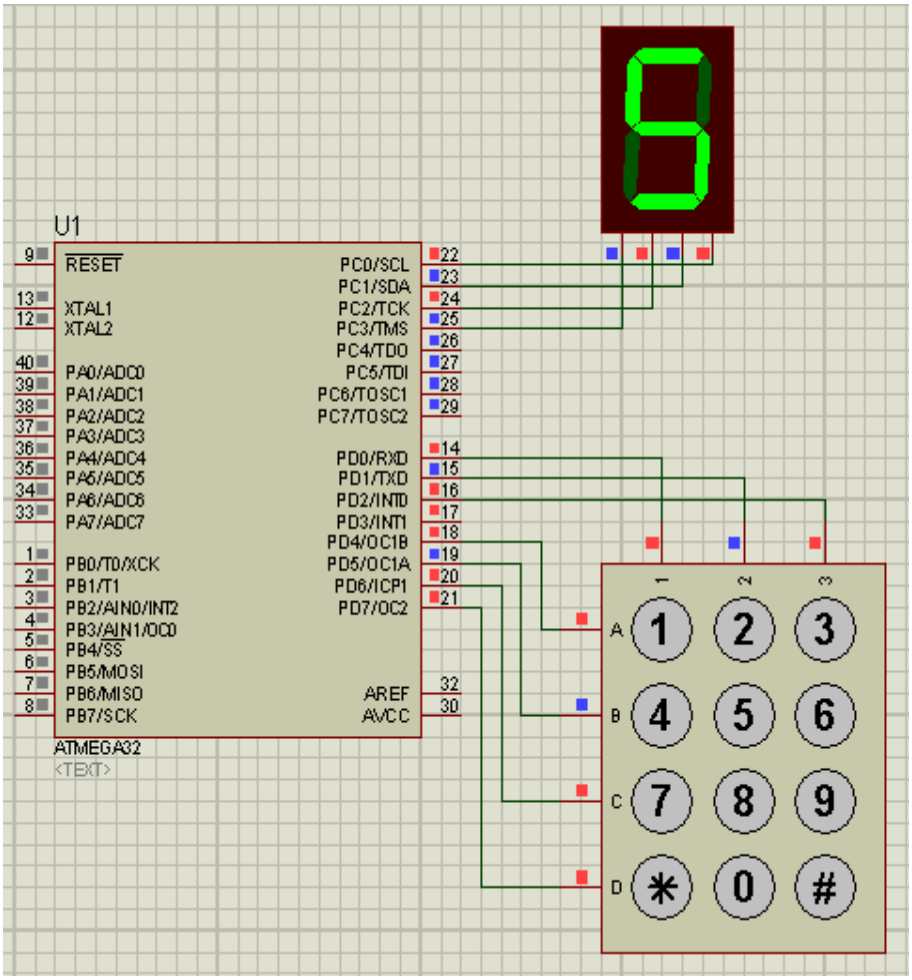
ابتدا یک سطر را صفر و بقیه را یک می کنیم، سپس ستون ها را می خوانیم. اگر همه ی آن ها یک باشند، یعنی در آن سطر کلیدی زده نشده است.آن سطر را یک و سطر بعدی را صفر می کنیم و مجددا ستون هارا می خوانیم. این کار را برای همه ی سطرها انجام می دهیم. اگر سطری را صفر کردیم و ستونی صفر شد با توجه به سطر و ستونی که صفر شده، کلید فعال مشخص می شود.





مثال: یک کیبورد تلفنی و یک سون سگمنت BCD را به میکرو متصل کنید. برنامه ای بنویسید که با زدن هر کلید عدد متناظر با آن در خروجی دیده شود. (برای * عدد ۱۰ و برای # عدد ۱۱ نمایش داده شود.)

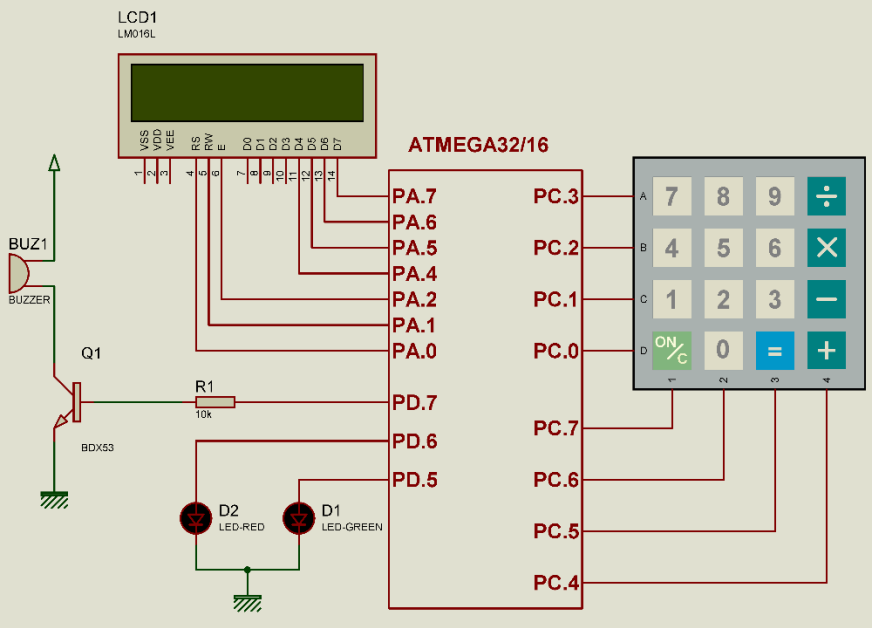
نکته: توجه داشته باشید که روی ستون ها از Pull-Up داخلی استفاده شده است.




```
#include <mega32a.h>
#include <delay.h>
unsigned char k;
void main (void)
{
DDRC=0xFF;
DDRD=0xF0;
while(1)
{
k=12;
PORTD=0xFF;
//---ROW1---//
PORTD.4=0;
delay_ms(3);
if(PIND.0==0){k=1;while(PIND.0==0);}
if(PIND.1==0){k=2;while(PIND.1==0);}
if(PIND.2==0){k=3;while(PIND.2==0);}
PORTD.4=1;
//---ROW2---//
```

```
PORTD.5=0;
delay_ms(3);
if(PIND.0==0){k=4;while(PIND.0==0);}
if(PIND.1==0){k=5;while(PIND.1==0);}
if(PIND.2==0){k=6;while(PIND.2==0);}
PORTD.5=1;
//---ROW3---//
PORTD.6=0;
delay_ms(3);
if(PIND.0==0){k=7;while(PIND.0==0);}
if(PIND.1==0){k=8;while(PIND.1==0);}
if(PIND.2==0){k=9;while(PIND.2==0);}
PORTD.6=1;
//---ROW4---//
PORTD.7=0;
delay_ms(3);
if(PIND.0==0){k=10;while(PIND.0==0);}
if(PIND.1==0){k=0;while(PIND.1==0);}
if(PIND.2==0){k=11;while(PIND.2==0);}
PORTD.7=1;
if(k!=12)PORTC=k;
}}
```

تمرین: یک صفحه کلید ۴*۴ و یک LCD کاراکتری را مطابق شکل زیر به میکرو متصل کرده برنامه ای بنویسید که با زدن هر کلید عدد یا علامت روی کلید بر روی LCD نمایش داده شود با زدن کلید ON/C نیز LCD پاک شود.



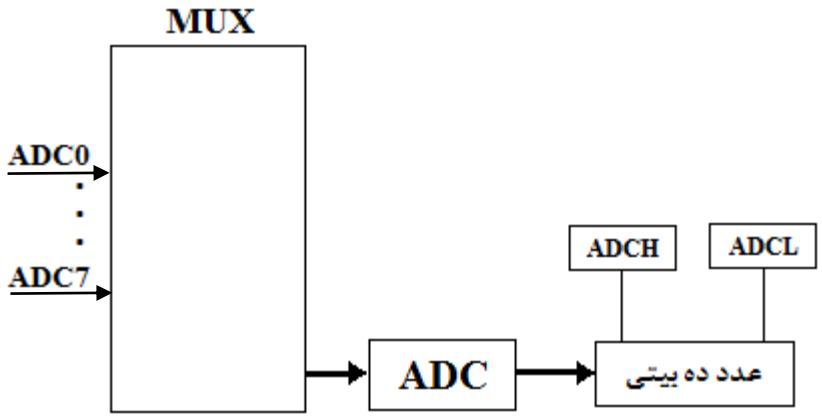
تمرین: برنامه صفحه کلید را طوری تکمیل کنید که با زدن هر کلید علاوه بر نمایش عدد و علامت Buzzer نیز بوق کوتاهی بزند.

تمرین: یک ماشین حساب بسازید. الف) یک رقمی ب) چند رقمی

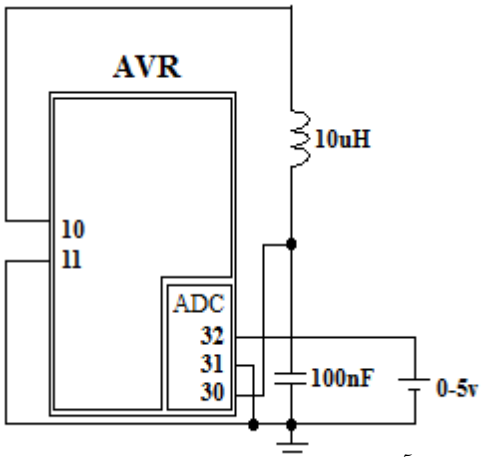
تمرین: یک قفل رمز بسازید.که کاربر یک رمز چهار رقمی را وارد کند اگر رمز صحیح بود led سبز روشن شود و buzzer یک بار بوق بزند،و اگر غلط بود led قرمز روشن شود و buzzer سه بار بوق بزند .

تمرین: تمرین قبل را طوری کامل کنید که مدیر بتواند رمز دستگاه را نیز عوض کند . در ضمن هنگامی که کاربران رمز را وارد می کنند اعداد زده شده بصورت * روی lcd دیده شوند. همچنین اگر کاربر رمز ورودی را سه بار اشتباه وارد کرد ، کیبورد قفل ، و هر دو led روشن شوند.

مبدل آنالوگ به دیجیتال (Analog to Digital Converter (ADC : یکی دیگر از قسمت های جانبی این میکرو واحد ADC می باشد. می دانیم که اکثر کمیت های اطراف ما مقادیری آنالوگ هستند؛ مانند دما، فشار و... که اگر بخواهیم آن ها را اندازه گیری و پردازش نماییم، لازم است که ابتدا به یک مقدار دیجیتال تبدیل و سپس پردازش گردند. در این میکرو یک ADC هشت کاناله و ۱۰ بیتی پیش بینی شده است.



تغذیه ADC : برای بالا بردن دقت ADC ولتاژ تغذیه این قسمت را از بقیه میکرو جدا کرده اند که می توانید ولتاژ جدا یا با قطعات زیر به ولتاژ اصلی وصل کنید.



- Vcc =10
- GND =11&31
- Aref =32
- AVcc =30

ولتاژ مرجع: در این نوع مبدل نیاز به یک ولتاژ مرجع داریم که می توان آن را از سه طریق تامین نمود:

۱. ولتاژ تغذیه: یعنی AVcc مرجع باشد. (5v)
۲. ولتاژ منبع خارجی: یعنی Aref .(5v تا 0)
۳. ولتاژ منبع داخلی: یعنی 2.56v (یک خازن در حد $1\mu F$ را روی پایه Aref قرار دهید)

ضریب تفکیک: پارامتری است که مشخص می کند حساسیت یا دقت ADC چقدر است و از رابطه زیر محاسبه می شود.

$$\text{ضریب تفکیک} = \frac{Vref}{2^{n}-1}$$

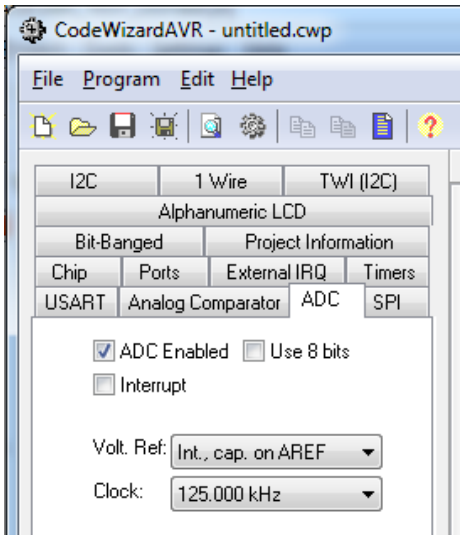
عدد خروجی: عدد خروجی مبدل را نیز می توان از رابطه زیر محاسبه کرد.

$$\text{عدد خروجی مبدل} = \frac{\text{Vin}}{\text{ضریب تفکیک}}$$

مثال: اگر ولتاژ ورودی به مبدل، یک ولت و ولتاژ مرجع ، داخلی و برابر 2.56v باشد، مطلوب است:

۱. ضریب تفکیک $\frac{2.56}{2^{10}-1} = 2.5mv$

۲. عدد خروجی مبدل $\frac{1}{2.5mv} = 400$



نکته: برای تنظیم ADC در ویزارد برگه ی ADC را باز و آن را فعال می کنیم. در قسمت volt. ref مشخص می کنیم ولتاژ مرجع از کدام منبع تامین شود.

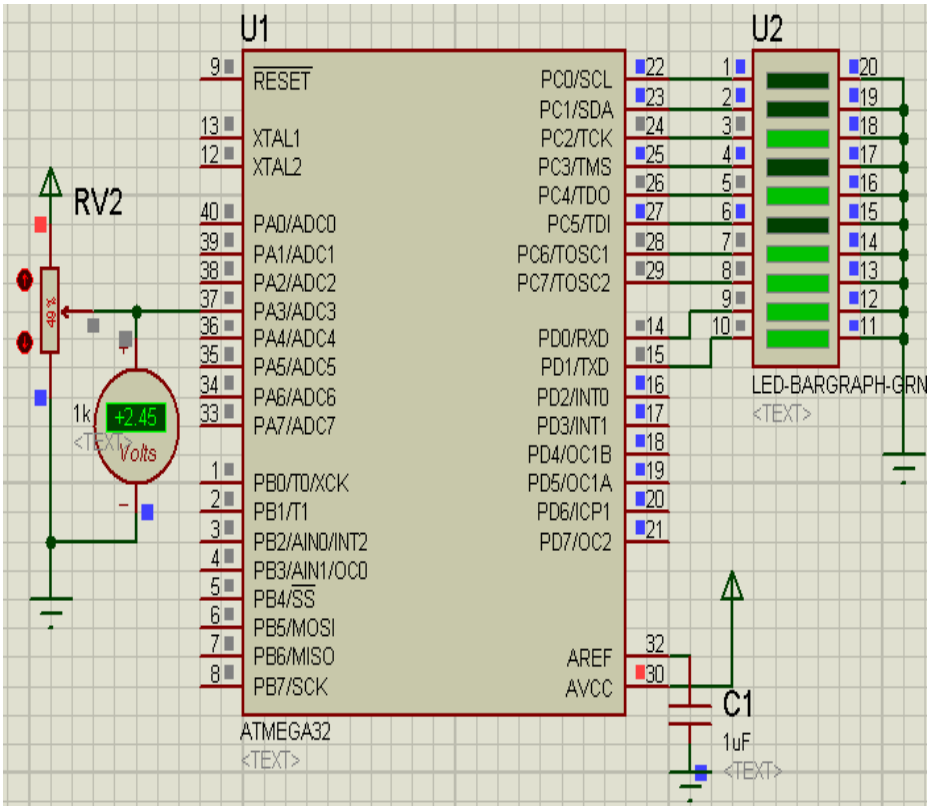
نکته: وقتی از مرجع داخلی استفاده می شود، باید یک خازن که طرف دیگر آن به زمین متصل است، در حد 1uF بر روی پایه Aref قرار گیرد.

در قسمت Clock فرکانس ورودی به قسمت ADC را مشخص می کنیم. لازم است که این فرکانس بین 50k تا 200k هرتز باشد. در صورت وجود ، قسمت Trigger را نیز روی ADC Stopped قرار دهید.

مثال: ۱۰ عدد LED را به پورت های C و D و یک ولتاژ متغیر را به کانال ADC(3) متصل کنید. برنامه ای بنویسید که ولتاژ کانال ۳ را به عدد دیجیتال تبدیل و روی LED ها نمایش دهد.

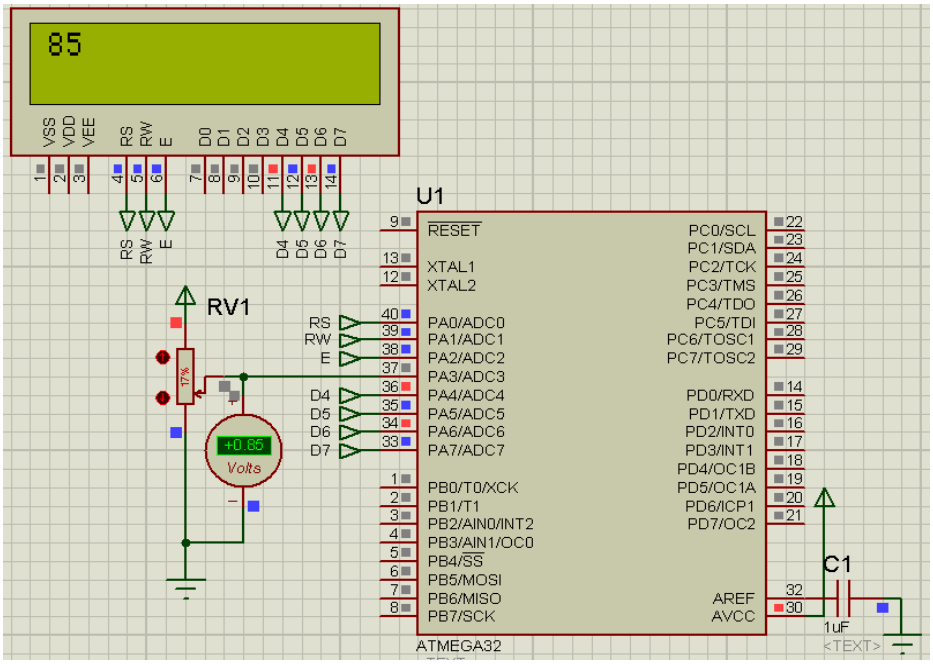
پس از تولید کد ملاحظه می شود تابع read_adc به برنامه اضافه شده است. این تابع مشخص می کند که از کدام کانال مقدار آنالوگ خوانده شود و خروجی آن عدد خروجی مبدل است.

```
void main(void)
{
while (1)
{
read_adc(3);
PORTC=ADCL;
PORTD=ADCH;
}
}
```

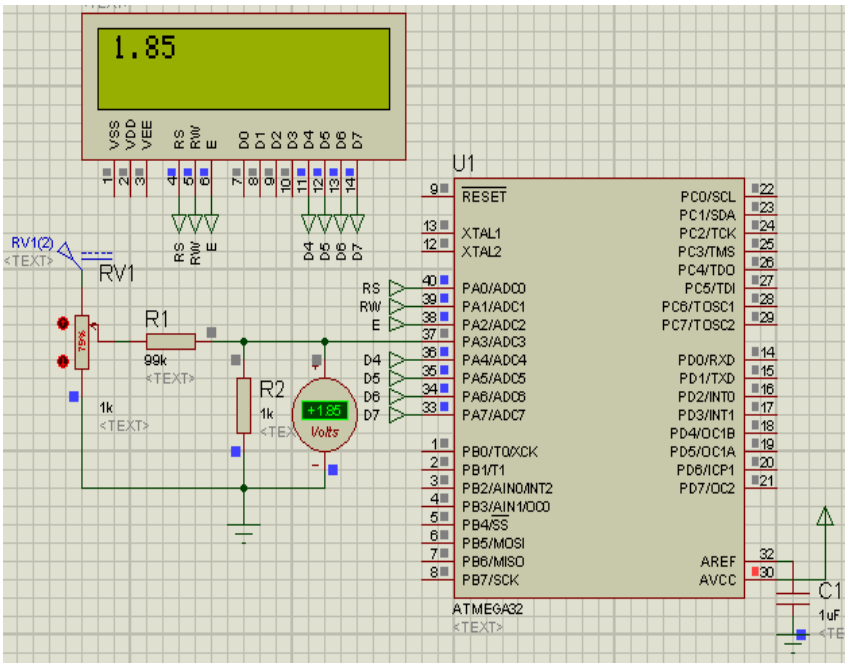


مثال: در مثال قبل یک LCD به پورت A متصل کنید و عدد خروجی مبدل را بر روی آن نمایش دهید.

```
#include <stdlib.h>
#include <delay.h>
unsigned int a;
char s[6];
void main(void)
{
while (1)
{
a=read_adc(3);
itoa(a,s);
lcd_gotoxy(0,0);
lcd_puts(s);
lcd_putsf("    ");
delay_ms(200);
}}
```

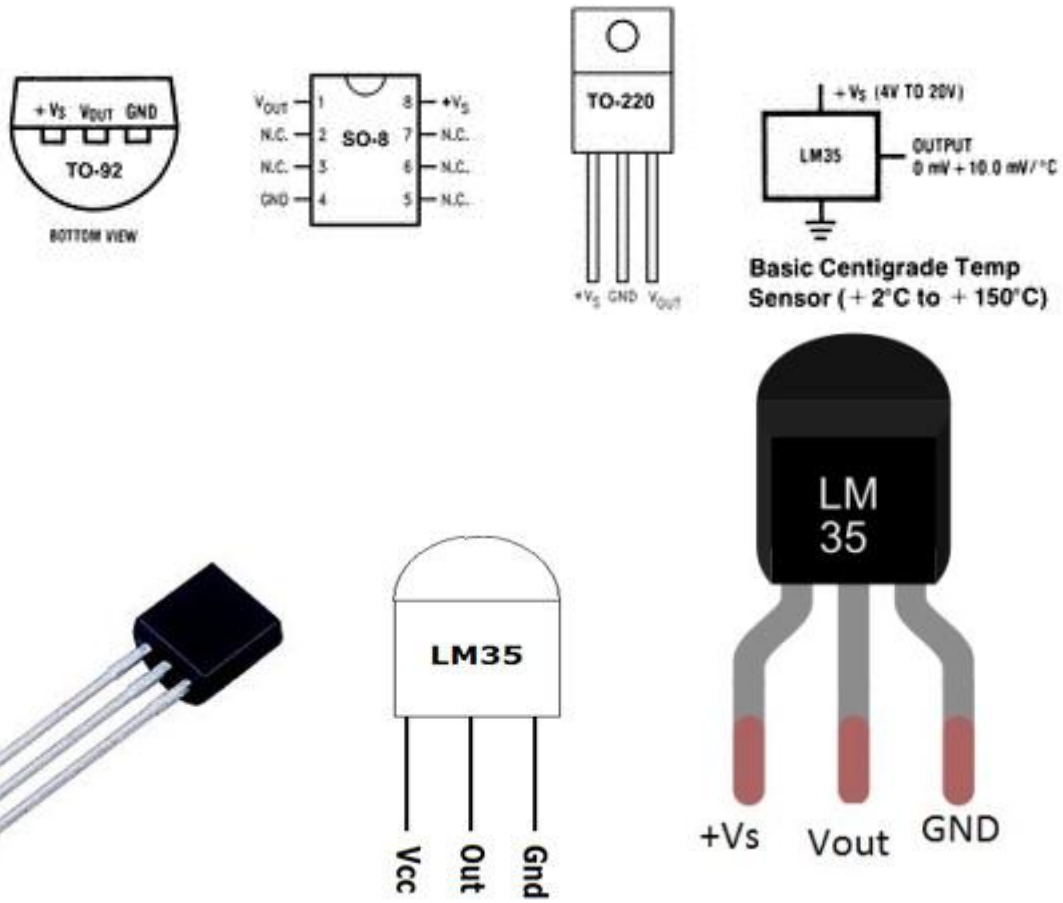


مثال: یک عدد LCD به پورت A متصل کرده و با استفاده از کانال ۳ یک ولتمتر بسازید که مقدار ولتاژ را بر روی LCD نمایش دهد.




```
#include <stdlib.h>
unsigned int a;
float b;
char s[5];
void main(void)
{
while (1)
{
a=read_adc(3);
b=(float)a/400;
ftoa(b,2,s);
lcd_gotoxy(0,0);
lcd_puts(s);
lcd_putsf("   ");
delay_ms(100);
}
}
```

سنسور LM35 : این سنسور مبدل دما به ولتاژ است و به ازای هر درجه دما 10mv خروجی دارد؛ یعنی خروجی آن دارای حساسیت 10mv بر درجه سانتیگراد است. برای مثال اگر دمای محیط ۱۵ درجه سانتیگراد باشد، خروجی سنسور 150mv خواهد بود.



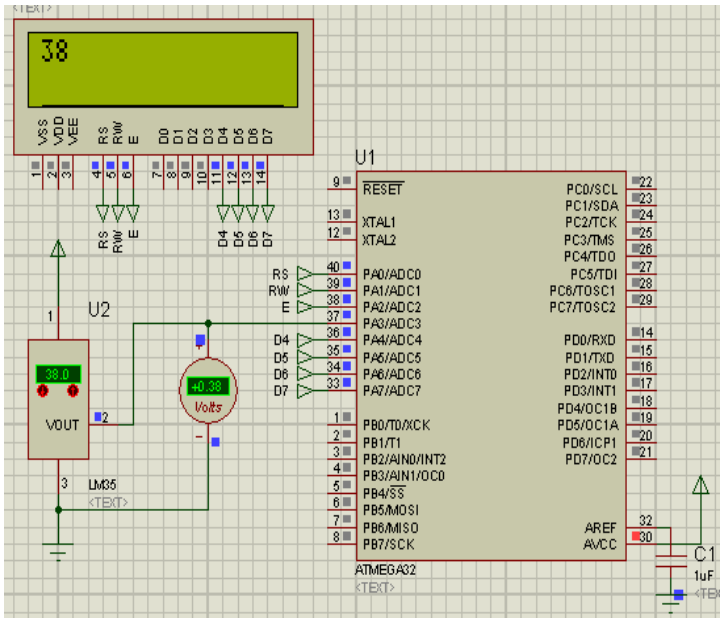
مثال: یک عدد LCD و سنسور LM35 به پورت A متصل کنید و برنامه ای بنویسید تا دمای محیط بر روی LCD نمایش داده شود.ولتاژ مرجع داخلی و برابر 2.56 V فرض شود.

پاسخ : فرض می کنیم دما یک درجه سانتیگراد باشد با توجه به ضریب تفکیک 2.5mv عدد خروجی مبدل 4 خواهد شد. عدد خروجی=10mv/2.5mv=4

با توجه به محاسبه بالا هر عددی که خوانده شد را بر ۴ تقسیم می کنیم تا دما بدست آید.

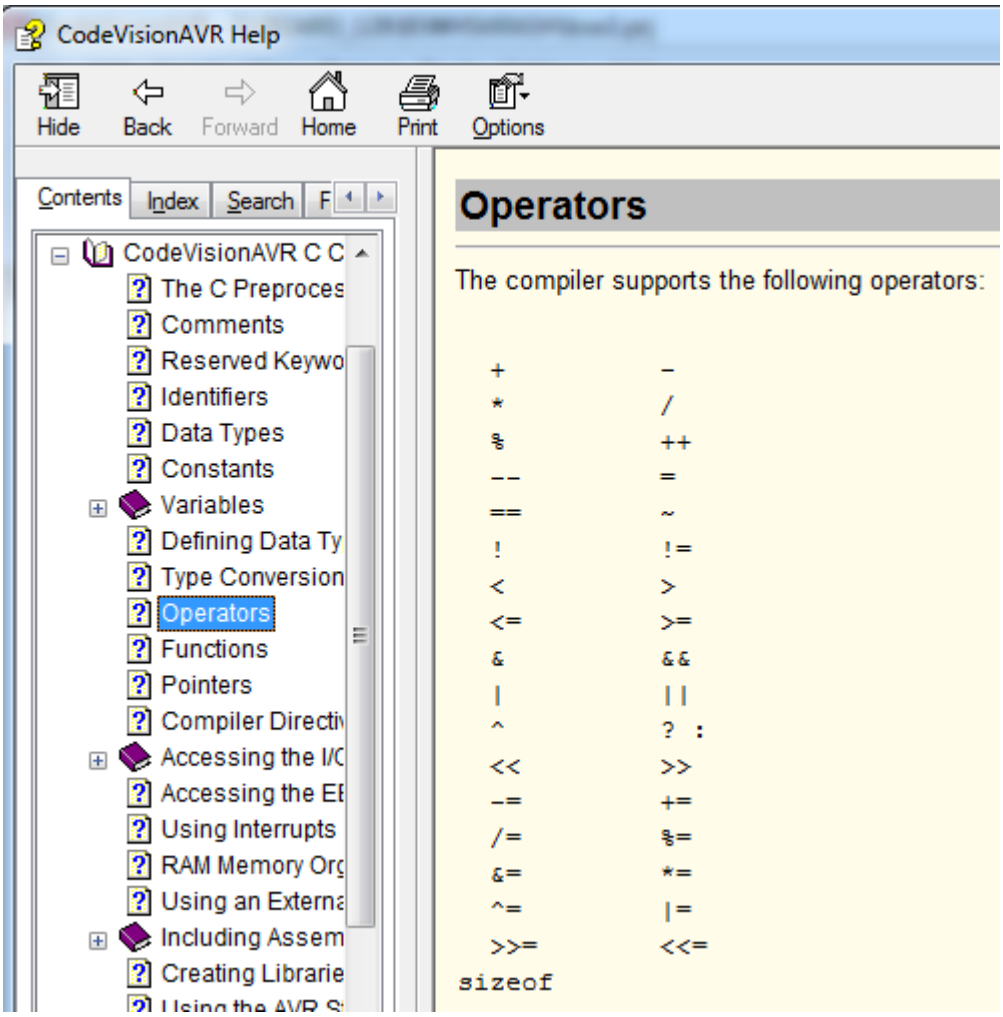
و برای برنامه کافی است در مثال بالا ولتمتر ، بجای 400 عدد خروجی مبدل را به 4 تقسیم کنیم.

```
b=(float) a/4;
```


38
VSS
VDD
VEE
RS
RW
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
C1
1uF
U2
39.0
VOUT
+0.38
Volts
LM35

عملگرها: OPERATORS

عملگرهایی که می توانید در زبان C استفاده کنید عبارتند از (جدول موجود در صفحه operators در help برنامه):



+ ، - ، * : عملگر های جمع،تفریق و ضرب نکته خاصی ندارد.

/ ، % : عملگر های تقسیم معمولی و باقی مانده.در مثال زیر تفاوت و نتیجه هر یک را مشاهده می کنید.

13 / 5=2
13 % 5=3

++ ، -- : عملگر های افزایش و کاهش به مقدار یک واحد.

a=5 a++; ⟶ a=6
a=5 a--; ⟶ a=4

= ، == ، =! : عملگر های انتساب ، شرط مساوی و شرط نامساوی :

تک مساوی مقداری را به یک متغیر نسبت می دهد، و جفت مساوی بررسی می کند که آیا دو مقدار با هم مساوی هستند یا خیر، و علامت تعجب ! و مساوی بررسی می کند که آیا دو مقدار با هم نامساوی هستند یا خیر .

در دستور مقابل مقدار 5 در متغیر K قرار می گیرد. K=5;

در دستور زیر مقدار K با عدد 5 مقایسه می شود اگر برابر باشند مقدار یک یا TRUE و در صورتی که برابر نباشند مقدار صفر یا FALSE نتیجه دستور می باشد. K==5;

در دستور زیر مقدار K با عدد 5 مقایسه می شود اگر برابر نباشند مقدار یک یا TRUE و در صورتی که برابر باشند مقدار صفر یا FALSE نتیجه دستور می باشد. K!=5;

! ، ~ ، & ، && ، | ، || ، ^ : عملگر های بیتی و منطقی :

منطقی	بیتی	عملگر
!	~	NOT
&&	&	AND
||	|	OR
ندارد	^	XOR

در عملگرهای بیتی ابتدا عملوندها را بصورت باینری نوشته و سپس بیت به بیت عمل مورد نظر را بر روی بیت ها انجام می شود.

مثال: اگر a=0x56 و b=0x9C باشد، مقادیرزیر را به دست آورید.

PORTC=a & b; PORTC=a | b; PORTC=a ^ b; PORTC=~a;

a=0x56 01010110

b=0x9C 10011100

a&b 00010100 Result PORTC=00010100=0x14

a=0x56 01010110

b=0x9C 10011100

a|b 11011110 Result PORTC=11011110=0xDE

a=0x56 01010110

b=0x9C 10011100

a^b 11001010 Result PORTC=00010100=0xCA

a=0x56 01010110

~a 10101001 Result PORTC=10101001=0xA9

مثال: اگر a=5 و b=8 باشد، پورت C چه عددی را نمایش می دهد؟

در این مثال به دلیل برقرار نبودن شرط اول else اجرا می شود و خروجی عدد 55 را نمایش می دهد.

```
if ((a>6)&&(b<10))
PORTC=99;
else
PORTC=55;
```

شیفت به چپ و راست:

مثال: اگر a=0xFE و b=0xCF باشد. حاصل عبارت زیر را بدست آورید.

PORTC=((a>>3)&(b<<2)) a=11001111 b=11111110

a>>3=00011001 & b<<2=11111000 Result PORTC=00011000=0x18

<u>>، <:</u> عملگر های شرط بزرگتر و شرط کوچکتر :

مثال: اگر a=5 و b=8 باشد، پورت C چه عددی را نمایش می دهد؟

در این مثال به دلیل برقرار بودن شرط دستور PORTC=99 اجرا می شود.

```
if(a<b)
PORTC=99;
else
PORTC=55;
```

<u>: ؟ : عملگر شرطی:</u>

این عملگر مانند دستور if else عمل می کند.

(دستور دوم) : (دستور اول) ؟ (عبارت شرطی)

اگر شرط برقرار باشد دستور اول و در غیر این صورت دستور دوم اجرا می شود.

(a>b) ? (PORTC=99) : (PORTC=55)

<u>عملگرهای تخصیص مرکب:</u>

توسط این روش می توان عبارات محاسبه ای را بصورت خلاصه نوشت.

a=a+5 a+=5
b=b*7 b*=7
c=c&8 c&=8
d=d<<4 d<<=4

sizeof :

خروجی این عملگر مقدار حافظه ای است که یک متغیر بر حسب بایت اشغال می کند.

char a; x=sizeof (a) نتیجه ← x=1

int b; x=sizeof (b) نتیجه ← x=2

float c; x=sizeof (c) نتیجه ← x=4

تمرین: برنامه بنویسید که عددی را از PORTA دریافت کند ، اگر فرد بود روی PORTC عدد 1 و اگر عدد زوج بود عدد 2 را نمایش دهد .

تمرین: برنامه بنویسید که یک عددی باینری را از PORTA دریافت کند ، و دهدهی آن را روی پورت های b,c,d نمایش دهد. به شکل مثال زیر:

PORTA	PORTB	PORTC	PORTD
10110110	1	8	2

تمرین: برنامه بنویسید که یک عددی باینری را از PORTA دریافت کند ، و تعداد صفرهای آن را روی PORTC و تعداد یک های آن را روی PORTD نمایش دهد. به شکل مثال زیر:

PORTA	PORTC	PORTD
10110110	3	5

تمرین: برنامه بنویسید که یک عددی باینری را از PORTA دریافت کند ، فرض کنید که خروجی سه سنسور به این پورت متصل شده عدد ارسالی هر سنسور را جدا کرد و به پورت های b,c,d ارسال و نمایش دهید. به شکل مثال زیر:

PORTA			PORTB	PORTC	PORTD
S1	S2	S3	S1	S2	S3
10	101	111	2	5	7

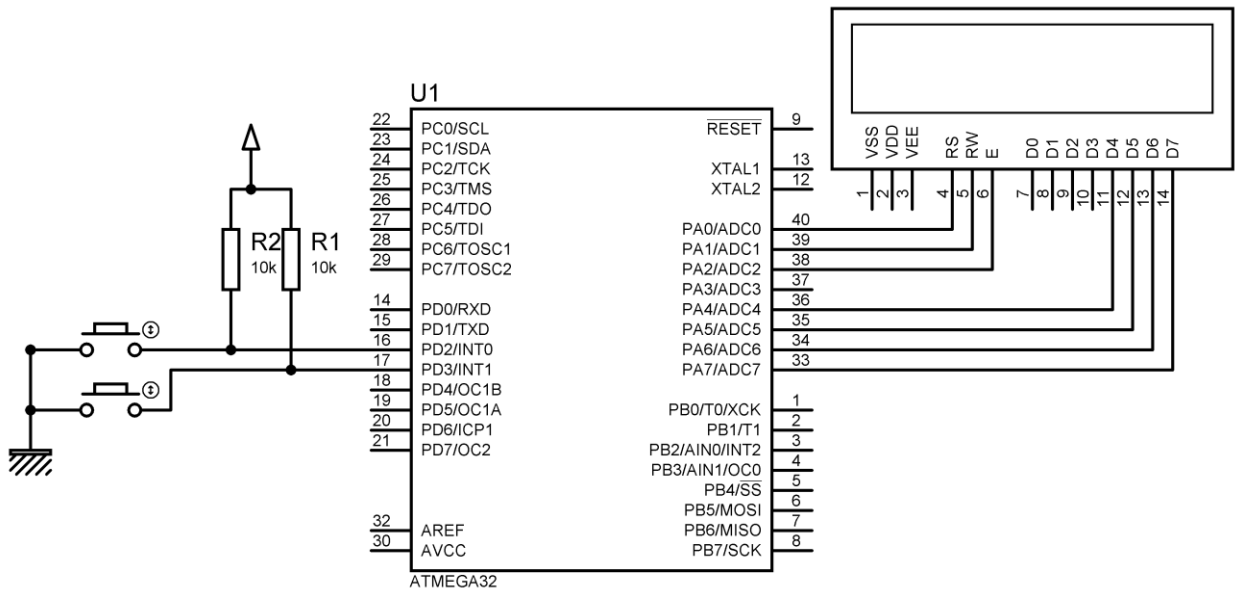
وقفهInterrupt :

همان طور که قبلا اشاره شد برای سرویس دهی به دستگاه های جانبی دو روش سرکشی Polling و وقفه را می توان استفاده کرد .

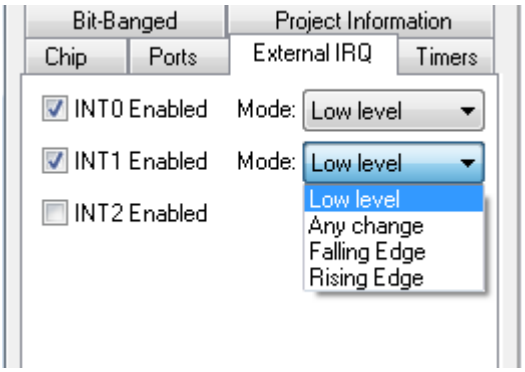
در میکرو مورد بحث ما سه وقفه خارجی INT0,INT1,INT2 وجود دارد . با یک مثال روش تنظیم،راه اندازی و استفاده از آن ها را فرامی گیریم.

مثال: یک LCD به PORTA و دو عدد کلید به وقفه های صفر و یک متصل کنید، برنامه ای بنویسید که وسط خط اول lcd یک شمارنده 0 تا 9 داشته باشیم (برنامه اصلی). حال اگر وقفه صفر حادث شد روی خط دوم lcd کلمه INT0 و اگر وقفه یک حادث شد روی خط دوم lcd کلمه INT1 به مدت دو ثانیه ظاهر و سپس پاک شود(روتین سرویس) و شمارش که به علت اجرای وقفه متوقف شده بود ادامه یابد.

(توجه داشته باشید سخت افزار پیشنهادی روی برد آزمایش فراهم می باشد)



برای تنظیم وقفه ها در wizard برگه External IRQ را باز و وقفه های مورد نظر را با زدن تیک فعال می کنیم.سپس در قسمت Mode باید مشخص کنیم که پاسخ به وقفه ، در چه وضعیتی از پالس وقفه داده و روتین سرویس آن وقفه اجرا شود.



❊اگر گزینه Any change را انتخاب کنید ، برنامه مربوط به آن وقفه دو بار اجرا می شود ، روی لبه پایین رونده و بالا رونده .

برای این مثال INT0 را روی لبه پایین رونده و INT1 را روی لبه بالا رونده قرار دهید.

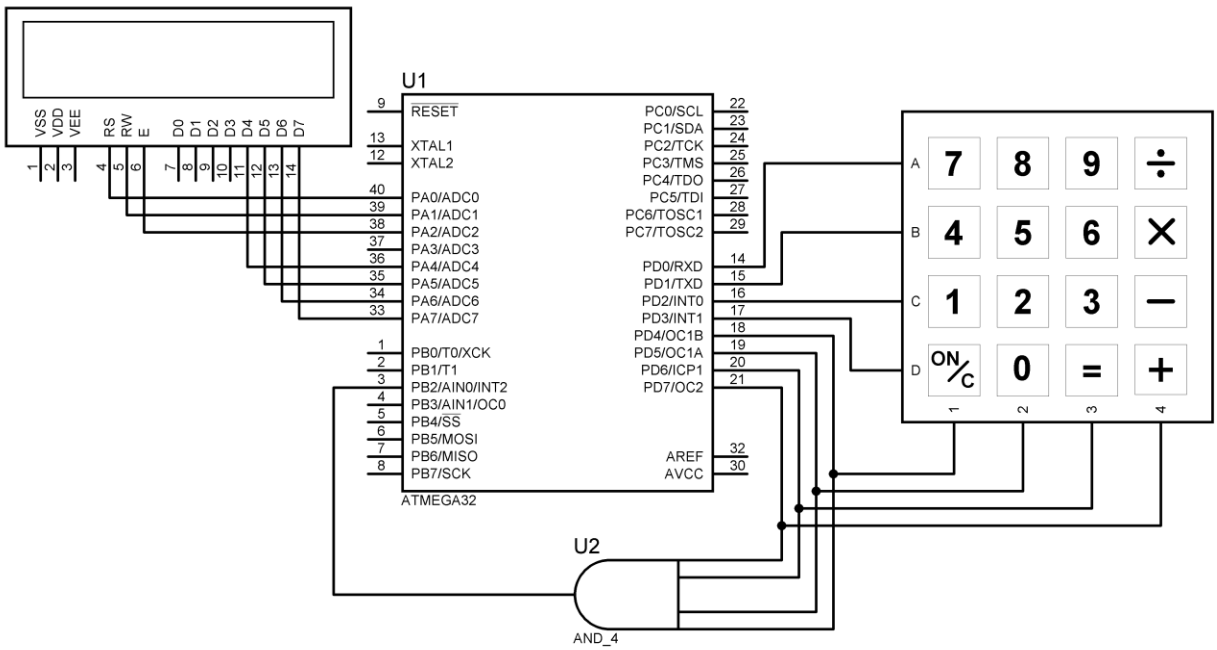
بعد از تولید کد ، مشاهده می کنید که برای هر یک از وقفه ها زیر برنامه ای آماده شده که می توانید روتین سرویس آن وقفه را آنجا بنویسید.

```
1   #include <mega32.h>
2
3   // External Interrupt 0 service routine
4   interrupt [EXT_INT0] void ext_int0_isr(void)
5   {
6   // Place your code here   محل نوشتن روتین سرویس INT0
7
8   }
9
10  // External Interrupt 1 service routine
11  interrupt [EXT_INT1] void ext_int1_isr(void)
12  {
13  // Place your code here   محل نوشتن روتین سرویس INT1
14
15  }
16
17  // Declare your global variables here
18
19  void main(void)
20  {
```

برنامه اصلی و وقفه ها بصورت زیر خواهد بود.

```
//External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
 lcd_gotoxy(0,1;(
 lcd_putsf("INT0;("
 delay_ms(2000;(
 lcd_clear;()
}
//External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
 lcd_gotoxy(0,1;(
 lcd_putsf("INT1;("
 delay_ms(2000;(
 lcd_clear;()
}
//Global enable interrupts
#asm("sei")فعال ساز وقفه سراسری
While(1)
{
     for(i=0;i<10;i++)
    {
      itoa(i,s);
      lcd_gotoxy(7,0);
      lcd_puts(s);
      delay_ms(500);}}
```

تمرین : یک LCD و یک صفحه کلید ۴*۴ را مانند شکل زیر به میکرو متصل کنید برنامه ای بنویسید که وسط خط اول lcd یک شمارنده 0 تا 9 داشته باشیم (برنامه اصلی). حال اگر کاربر هر کلیدی را زد عدد یا علامت آن کلید وسط خط دوم lcd نمایش داده شود (روتین سرویس).

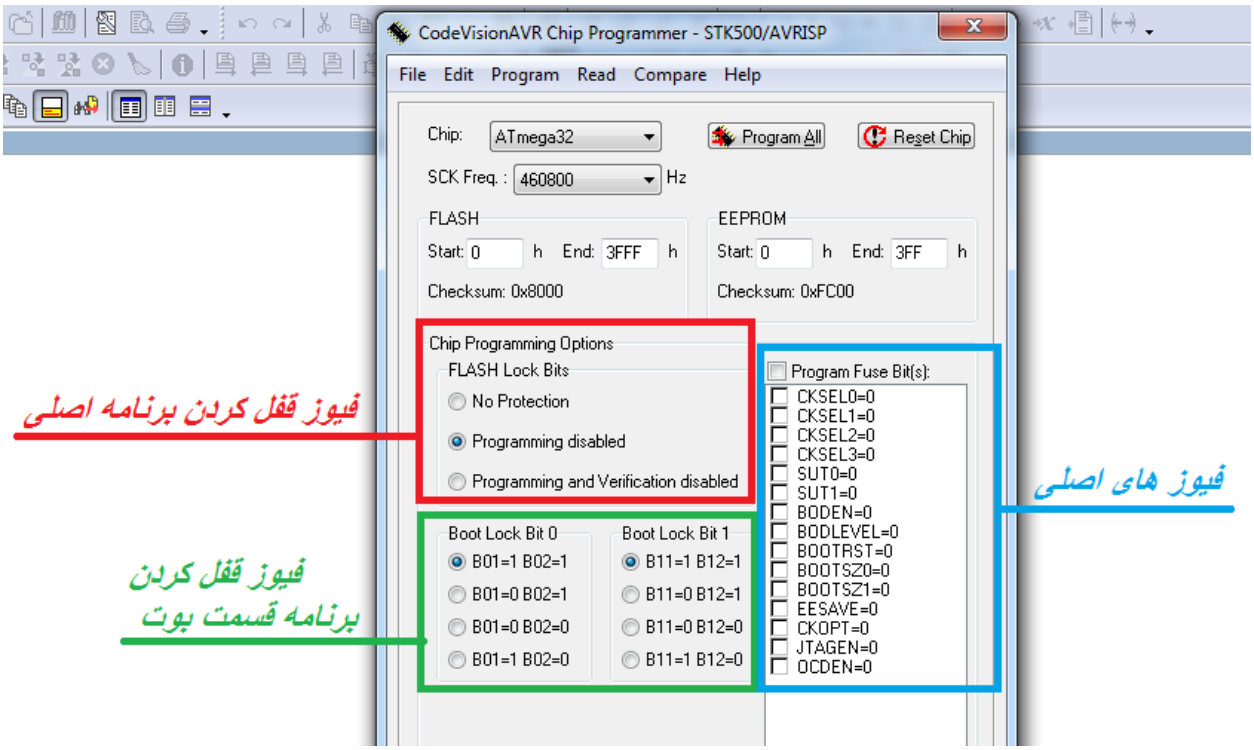


فیوزبیت Fuse bit :

تعدادی بیت در حافظه Flash که می توان آن ها را توسط پرگرامر خواند،ویرایش کرد و دوباره برنامه ریزی نمود.

هردسته یا هر یک از فیوز بیت ها می توانند بخشی از میکرو را در وضعیت مشخصی قرار دهند.

فیوز های ATMEGA32 عبارتند از:



در ATmega32 دو بایت برای فیوز بیت ها در نظر گرفته شده است که در جدول زیر آن ها را مشاهده می کنید:

شماره بیت	۰	۱	۲	۳	۴	۵	۶	۷
فیوز بیت های بالا	BOOTRST	BOOTSZ2	BOOTSZ1	EESAVE	CKOPT	SPIEN	JTAGEN	OCDEN
مقدار پیش فرض	1	0	0	1	1	0	0	1
شماره بیت	۰	۱	۲	۳	۴	۵	۶	۷
فیوز بیت های پایین	CKSEL0	CKSEL1	CKSEL2	CKSEL3	SUT0	SUT1	BODEN	BODLEVEL
مقدار پیش فرض	1	0	0	0	0	1	1	1

کاربرد این فیوز بیت ها به شرح زیر است:

CKSEL0_1_2_3: توسط این چهار فیوز می توان مشخص کرد که اولا منبع کلاک سیستم از کجا تامین شود و ثانیا فرکانس آن چقدر باشد.

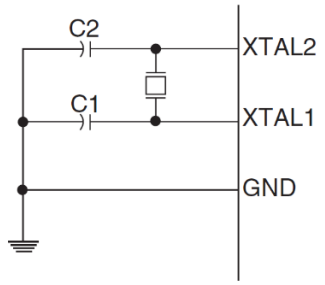
منبع کلاک پالس	CLKSEL3_2_1_0
کریستال خارجی/ تشدیدگر سرامیکی	1111-1010
کریستال فرکانس پایین خارجی	1001
نوسان ساز RC خارجی	1000-0101
نوسان ساز RC کالیبره شده داخلی	0100-0001
نوسان ساز خارجی	0000

تنظیم فیوزها برای نوسان ساز RC داخلی :

فرکانس بر حسب MHZ	CLKSEL3_2_1_0
1.00	0001
2.00	0010
4.00	0011
8.00	0100

کریستال خارجی :

در این حالت مطابق شکل زیر کریستال را متصل کنید. ظرفیت خازن ها بین 12PF تا 22PF باشد . و هر چهار فیوز بیت CLKSEL را روی مقدار 1111 قرار دهید .



فیوز بیت های SUT0,1 (Start-up Time) :

به وسیله این فیوز بیت ها زمان start-up time برای شروع به کار میکروکنترلر تعیین می شود. این زمان فاصله بین اتصال تغذیه تا شروع به کار میکروکنترلر است و برای پایدار شدن خروجی نوسان ساز لازم است، چرا که ممکن است نوسان ساز در زمان اتصال تغذیه دقیقا همان فرکانس مطلوب را تولید نکند وعملا با تنظیم یک زمان طولانی تر فرصت پایدار شدن به نوسان ساز داده می شود. برای کاربردهای معمولی توصیه می شود با مقادیر پیش فرض کار کنید و این فیوز بیت ها را تغییر ندهید.

فیوز بیت CKOPT (Oscillator options) :

این بیت بین دو حالت مختلف تقویت کننده اسیلاتور یکی را انتخاب می کند. وقتی که این بیت پروگرم یعنی صفر شود باعث ایجاد دامنه بیشتری برای نوسان ساز خواهد شد و در صورت غیرفعال شدن دامنه را کاهش می دهد. اگر از کریستال خارجی استفاده می کنید بهتر است که این فیوز بیت را فعال کنید تا نویزپذیری کمتر شود اما دقت کنید که با فعال کردن این بیت توان مصرفی بیشتر خواهد شد.

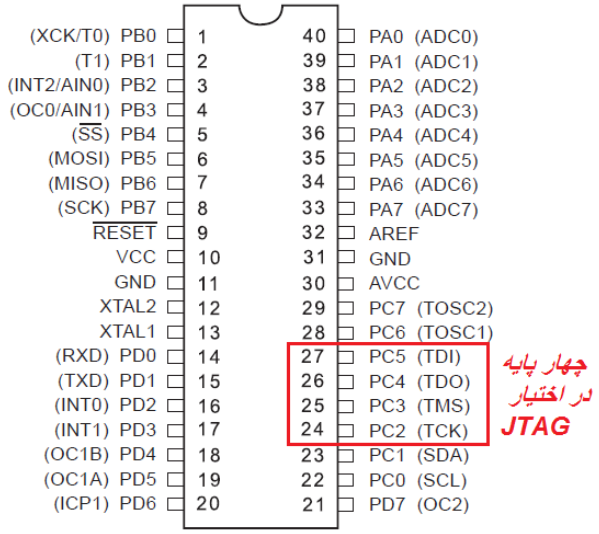
BODEN : برای فعال شدن Brown-Out Detection باید این فیوز را فعال کنید.

BODLEVEL : مشخص می کند Brown-Out Detection روی چه ولتاژی عمل کند.

0 = 4 volt 1 = 2.7 volt

EESAVE: اگر این فیوز بیت فعال باشد هنگام پاک کردن حافظه ، Flash پاک می شود ولی حافظه دیتای ماندگار EEPROM پاک نمی شود.

JTAGEN: برای فعال شدن jtag باید این فیوز فعال باشد.توجه داشته باشید در میکرو نو که می خرید این فیوز فعال است در نتیجه چهار پین از PORTC در اختیار jtag است و نمی توانید از تمام پایه های پورت C استفاده کنید.



(On-chip debug enabled) : OCDEN

اگر این فیوز ، همراه با فیوز JTAGEN فعال باشد آنگاه می توانید با استفاده از رابط jtag برنامه داخل میکرو را خط به خط اجرا و در صورت نیاز رفع اشکال نمایید.

(SPI Enable) : SPIEN

فعال بودن این فیوز ، امکان پروگرام کردن میکرو را بصورت سریال برقرار می کند توصیه می شود این فیوز همواره فعال باشد.

(Boot Reset): BOOTRST

اگر این فیوز بیت فعال باشد وقتی میکروکنترلر ریست شود به آدرس بلوک بوت لودر پرش خواهد کرد. می توان بخشی از حافظه Flash را به Boot اختصاص داد . بوت این امکان را فراهم می کند تا بتوان بدون استفاده از پروگرامر بخشی از حافظه Flash را برنامه ریزی کرد .

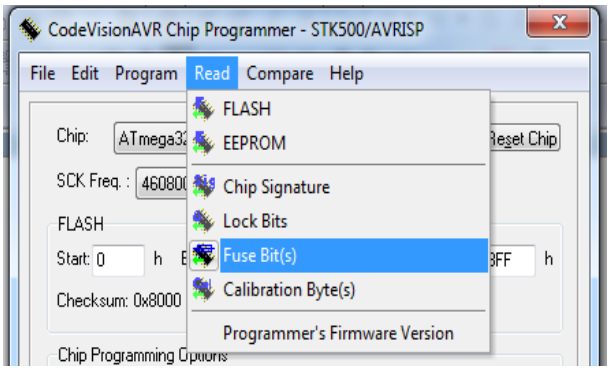
(Boot Size): BOOTSZ0,1

این فیوزها مشخص می کنند تا چه ظرفیتی از حافظه Flash به Boot اختصاص یابد.

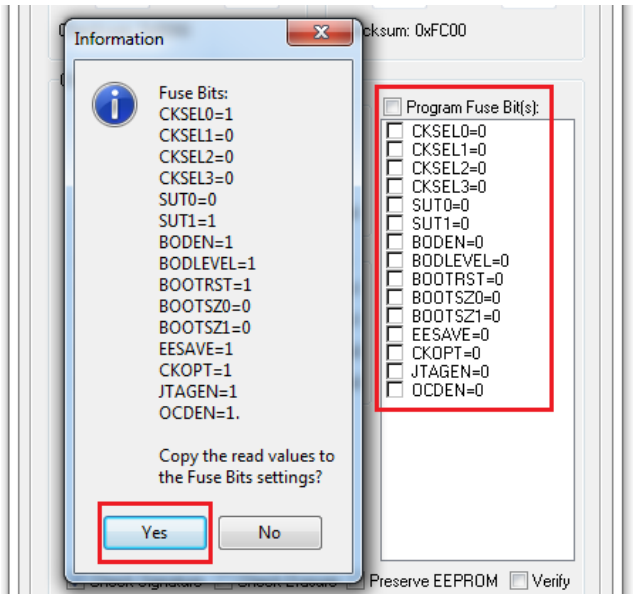
BOOTSZ1	BOOTSZ0	Boot Size
1	1	256 words
1	0	512 words
0	1	1024 words
0	0	2048 words

خواندن فیوز بیت ها:

برای این منظور در پنجره Chip Programmer مسیر زیر را طی می کنیم.Read→Fuse Bit(s)



بعد از این مرحله فیوزها خوانده شده ، و در یک پنجره نمایش داده می شود، اگر گزینه yes را انتخاب کنید وضعیت موجود فیوزها ، به قسمت Program Fuse Bit(s) منتقل خواهد شد و امکان ویرایش آن ها وجود دارد.



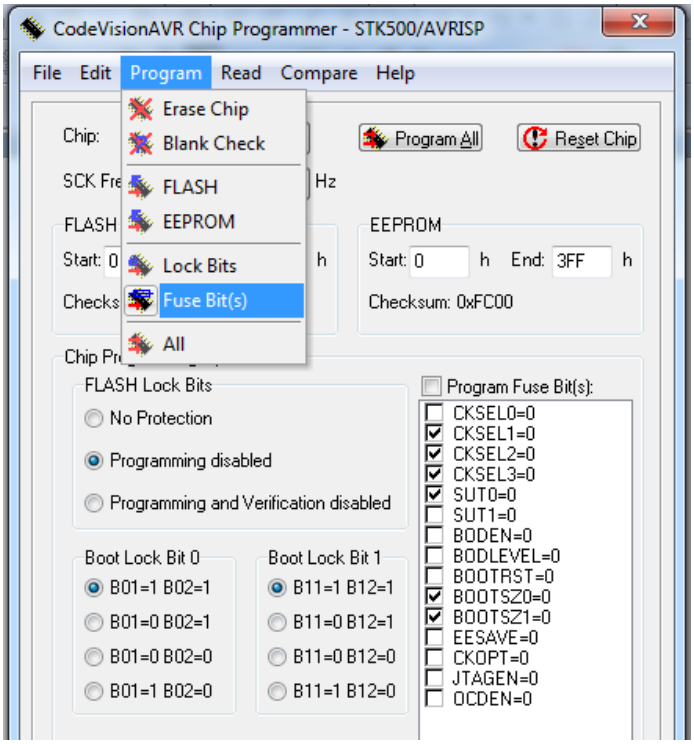
برنامه ریزی فیوز بیت ها:

برای این منظور در پنجره

Chip Programmer

مسیر زیر را طی می کنیم.

Program→Fuse Bit(s)



**توجه داشته باشید که ابتدا با مراجعه به Data sheet اطلاع کاملی از وضعیت Fuse Bit ها پیدا کنید و سپس مبادرت به تغییر آن ها نمایید در غیر این صورت ممکن است میکرو در حالتی قرار گیرد که دیگر در مدار شما کار نکند و نتوانید با پرگرامرهای معمولی آن را از این حالت خارج کنید.

تابع Function :

ساختار زبان C بر پایه توابع است . کاربر می تواند هر بخش از برنامه را بصورت یک تابع نوشته و هرگاه لازم بود آن را فراخوانی و اجرا نماید.نوشتن برنامه بصورت توابع باعث می شود :

۱- خوانایی برنامه بالا رود.
۲- رفع اشکال از برنامه ساده تر انجام می شود.
۳- اصلاح و ارتقاء برنامه ساده تر صورت گیرد.
۴- از توابع نوشته شده می توان در برنامه های دیگر استفاده کرد.

بسته به این که تابع ورودی یا خروجی داشته باشد چهار حالت زیر را خواهیم داشت:

۱- بدون ورودی – بدون خروجی `void f1(void)`
۲- با ورودی – بدون خروجی `void f2(unsigned char x)`
۳- بدون ورودی – با خروجی `char f3(void)`
۴- با ورودی – با خروجی `int f4(float y)`

نکته: اگر تابعی دارای چند ورودی از یک نوع باشد باید تک تک آن ها معرفی شوند مانند مثال زیر:

درست `void f1(int x,int y,int z)` نادرست `void f1 (int x,y,z)`

نکته: هر تابع فقط می تواند یک خروجی داشته باشد. و برای خروج دیتا از دستور return استفاده می شود.

`Return` مقدار ثابت ; یا `Return` نام متغیر ;

محل قرار گیری تابع می تواند یکی از اشکال زیر باشد:

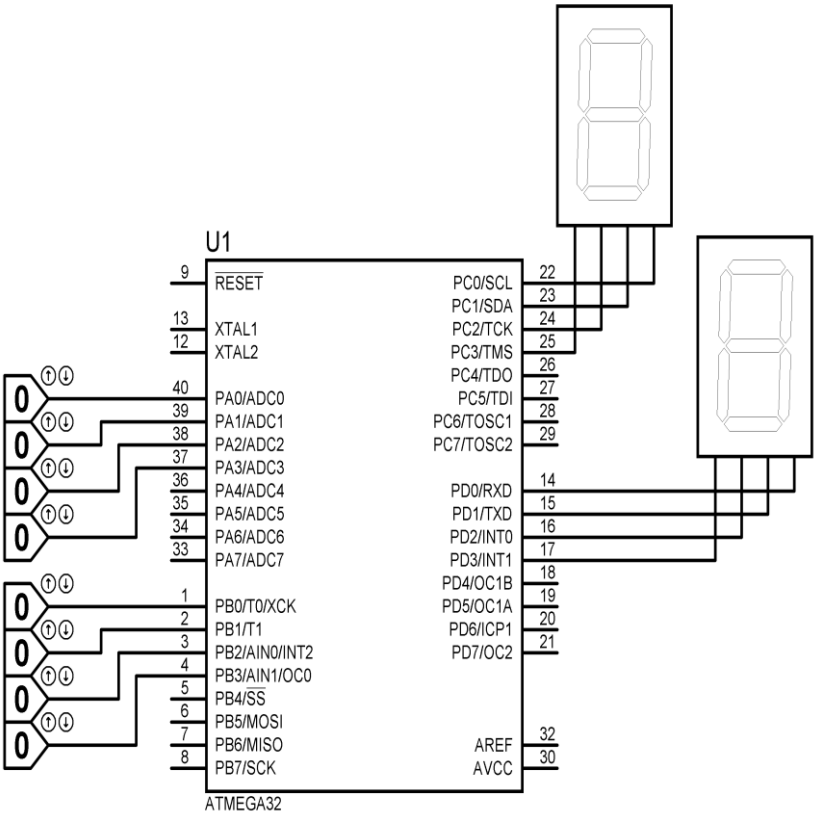
(الف) (ب)

(الف)	(ب)

معرفی تابع ;
 	
Void main (void)
{
…….
…….
فراخوانی تابع
…….
}
 	
معرفی و تعریف تابع
{
……
……
}
	معرفی و تعریف تابع
	{
	……
	……
	}
	
	Void main (void)
	{
	…….
	…….
	فراخوانی تابع
	…….
	}

اگر توابع مستقل بوده ، و از داخل یک تابع ، تابع دیگری فراخوانی نشود شکل الف یا ب تفاوتی نخواهد داشت در غیر این صورت باید روش الف را استفاده کنید.

مثال: برنامه ای بنویسید که دو عدد چهار بیتی را از PORTA و PORTB وارد کند سپس حاصل جمع آن ها را روی PORTC و حاصل تفریق آن ها را روی PORTD نمایش دهد. این برنامه را یک بار بدون استفاده از تابع و یک بار با استفاده از تابع بنویسید.




```
#include  <mega32.h>
unsigned char  a,b,c,d;
void main (void)
{
 DDRA=0x00;
DDRB=0x00;
DDRC=0xff;
DDRD=0xff;
While(1)
{
  a=PINA & 0x0f;
  b=PINB & 0x0f;
  c=a+b;
  d=a-b;
PORTC=c;
```

```
PORTD=d;
}
}
#include  <mega32.h>
void  init_port (void) ;
void  input (void) ;
unsigned char  sum (unsigned char  x, unsigned char  y) ;
unsigned char  sub  (unsigned char  x, unsigned char  y) ;
void  output (unsigned char  x, unsigned char  y) ;
unsigned char  a,b,c,d;
void main (void)
{
 Init_port();
While(1)
{
  input();
  c=sum(a,b);
  d=sub(a,b);
  output(c,d);
}
}
void  init_port (void)
{
```

```
 DDRA=0x00;
DDRB=0x00;
DDRC=0xff;
DDRD=0xff;
}

void  input (void)
{
  a=PINA & 0x0f;
  b=PINB & 0x0f;
}
unsigned char  sum (unsigned char  x, unsigned char  y)
{
 unsigned char   z;
 z=x+y;
return  z;
}
unsigned char  sub  (unsigned char  x, unsigned char  y)
{
 unsigned char   z;
 z=x-y;
return  z;
}
```

```
void  output (unsigned char  x, unsigned char  y)
{
 PORTC=x;
 PORTD=y
}
```

اشاره گر POINTER :

هرگاه لازم باشد به جای متغیر از آدرس آن متغیر استفاده کنیم می توان توسط اشاره گربا آدرس آن حافظه کار کرد.کاربرد اشاره گر در تخصیص حافظه پویا ، دسترسی آسان تر و کارکردن راحت تر با آرایه ها و ارسال متغیرها به توابع با ارجاع و..... می باشد.

در برنامه هایی که برای میکرو می نویسیم بیشترین کاربرد زمانی است که بخواهیم با فراخوانی یک تابع چند مقدار را بدست آوریم ، از آنجایی که هر تابع نمی تواند بیش از یک خروجی داشته باشد لازم است آرگومان های تابع ازطریق فراخوانی با ارجاع به تابع ارسال شود. برای مثال در خواندن زمان از آی سی DS1307 باید از تابع rtc_get_time استفاده شود ، پس باید سه مقدار، ساعت_دقیقه_ثانیه از تابع خارج شود که امکان پذیر نیست و باید از فراخوانی با ارجاع استفاده شود. پس به جای خود متغیر از آدرس آن ها استفاده می کنیم.

علامت & یعنی آدرس متغیر rtc_grt_time(&h,&m,&s);

تعریف متغیر از نوع اشاره گر:

برای این منظور مانند متغیر های معمولی عمل می شود با این تفاوت که قبل از نام متغیر اشاره گر یک علامت ستاره * قرار می گیرد.

```
Unsigned char   a,b,*p,*q;
```

در خط بالا a,b دو متغیر معمولی و p,q از نوع اشاره گر هستند.

نکته: در هنگام نوشتن برنامه هر جا آدرس یک متغیر مدنظر بود از علامت & و هرجا محتوا یا مقدار متغیر مدنظر بود از علامت * استفاده می شود.

با توجه به تعریف زیر فرض می کنیم متغیر a در خانه 60 حافظه ، متغیر b در خانه 62 حافظه و متغیر c در خانه 64 حافظه ذخیره شده باشند. unsigned char a=12 , b=3 , c , *p ,*q ,*r;

با توجه به تعریف متغیری که در بالا انجام شد می توان شکل زیر را تصور کرد.

متغیر	آدرس	محتوا
	59	
a	60	12
	61	
b	62	3
	63	
c	64	
	65	
	66	

حال برنامه زیر را در نظر بگیرید:

1) p=&a;
2) q=&b;
3) r=&c;
4) *r=*p+*q;
5) *p=*q;
6) PORTC=a+b;

در سه خط اول ، آدرس حافظه هایی که متغیر های a,b,c درآن جا ذخیره شده اند درون اشاره گر های p,q,r قرار می گیرند یعنی: p=60 q=62 r=64

درخط چهارم محتوای محل هایی که p وq به آن ها اشاره می کنندبا هم جمع شده در حافظه ای که اشاره گر r به آن اشاره می کند یعنی خانه 64 حافظه قرار می گیرد.

در خط پنجم محتوای متغیر مورد اشاره q یعنی مقدار متغیر b در متغیر مورد اشاره p یعنی متغیر a قرار می گیرد به عبارت دیگر مقدار a و b با هم برابر می شوند. a=b=3

در نتیجه خط ششم باعث خواهد شد روی PORTC عدد 6 را داشته باشیم.

متغیر از نوع ساختار structure :

هرگاه تعدادی متغیر غیر هم نوع داشته باشیم و بخواهیم آن ها را در قالب یک متغیر تعریف کنیم لازم است که از متغیرهای نوع ساختار استفاده شود.

برای مثال یک فرم استخدام را در نظر می گیریم .

ردیف	نام	نام خانوادگی	سن	معدل
۱	علی	بارانی	۲۳	۱۵/۷۵
۲	آرش	غیور	۲۴	۱۶/۲۵
۳	پروین	میرزایی	۲۲	۱۷/۳۳
۴	سایه	حیدری	۲۶	۱۴/۵

هر سطر یا Record در این فرم دارای پنج field می باشد ، ردیف و سن اعداد صحیح ، نام و نام خانوادگی از نوع رشته و معدل از نوع اعشاری می باشند.

حال می خواهیم متغیری تعریف کنیم که هر پنج قسمت را در خود داشته باشد.

برای این منظور از متغیر نوع struct و به شکل زیر استفاده می کنیم.

struct نام ساختار {

عناصر ساختار

}نام متغیرها ;

```
struct   form{
     int    id;
    char  name[20];
    char  family[20];
    int     age;
    float  average;
}f;
```

دسترسی به عناصر ساختار:

برای دسترسی به هر یک از عناصر ساختار باید به شکل زیر عمل کنیم:

مقدار=نام فیلد.نام متغیر

در مثال بالا نام متغیر f می باشد ، می خواهیم عناصر سطر اول را وارد کنیم.

```
f.id=1;
sprintf(f.name,”ali”);
sprintf(f.family,”barani”);
f.age=23;
f. average=15.75;
```

در برنامه کدویژن هنگامی که می خواهیم کنترل بیتی روی بیت های هر یک از پورت ها داشته باشیم از این نوع متغیر استفاده می شود.

```
DDRA.5=1;

If  (PIND.2==0)   PORTD.7=1;
```