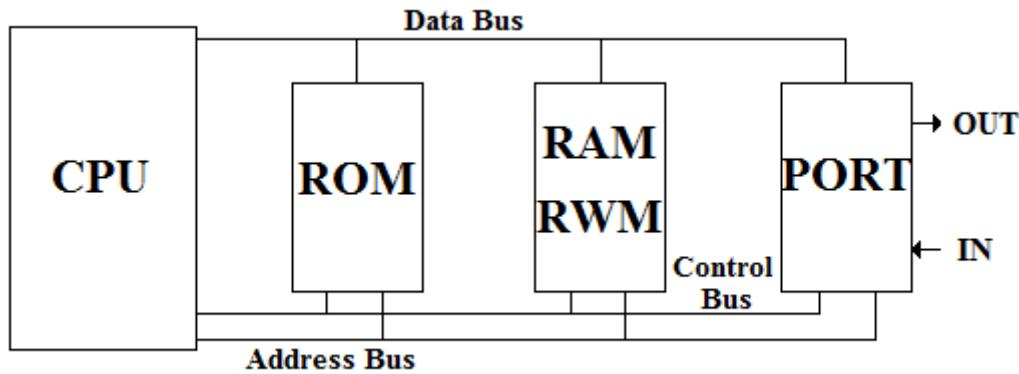


درس: رضا فتاحی

بلوک دیاگرام یک سیستم میکروپروسسوری μP



Microprocessor یا **Central Processing Unit : CPU**

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

Read Only Memory : ROM

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

Read Write Memory (RWM): RAM

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

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

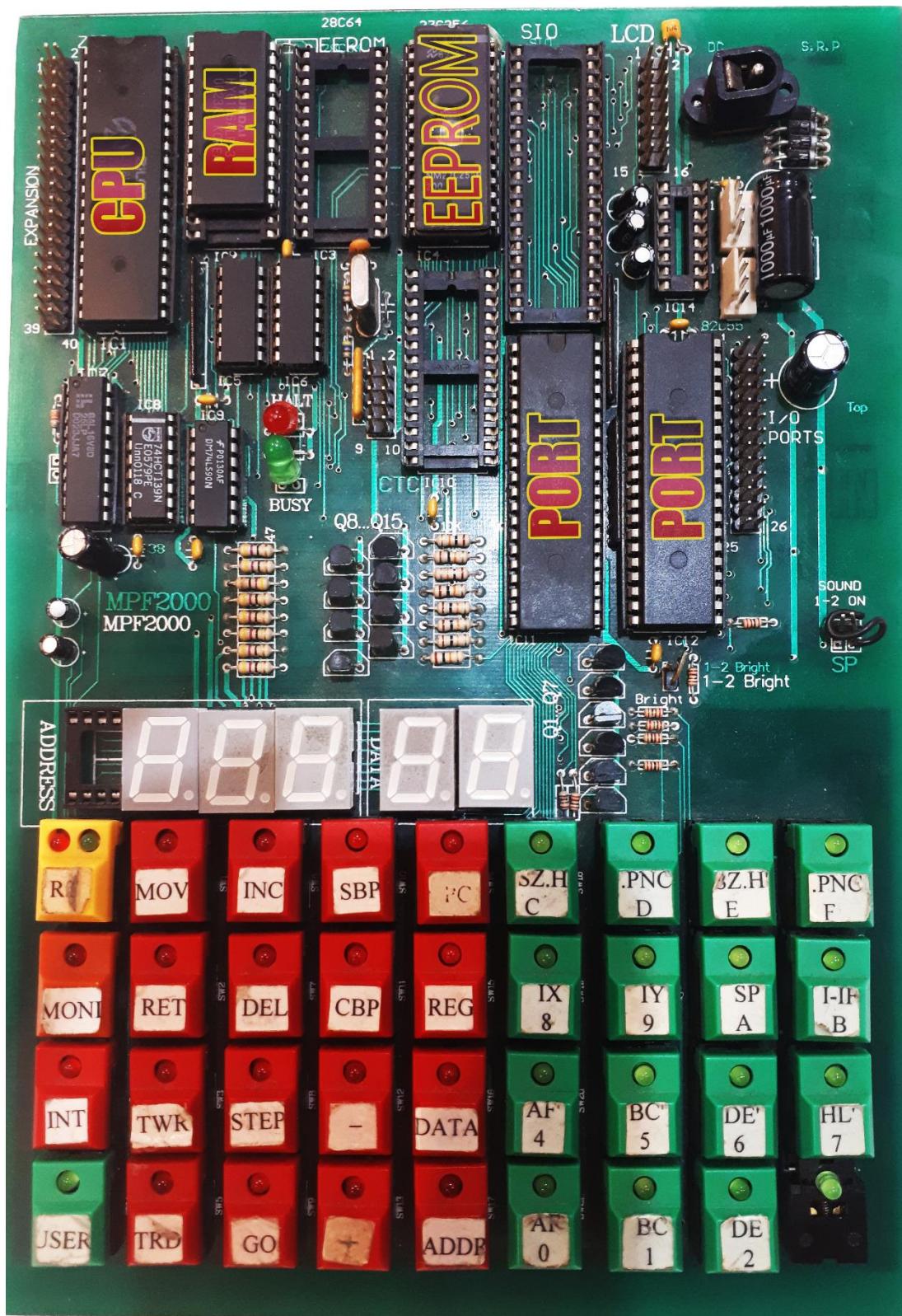
در هر سیستم سه نوع Bus وجود دارد: **Bus**

- مشخص می کند با کدام حافظه یا پورت کار داریم . Address

- مشخص می کند چه کاری داریم.(خواندن یا نوشتن....) Control

- توسط این خطوط اطلاعات بین CPU و بقیه مدار منتقل می شود. Data

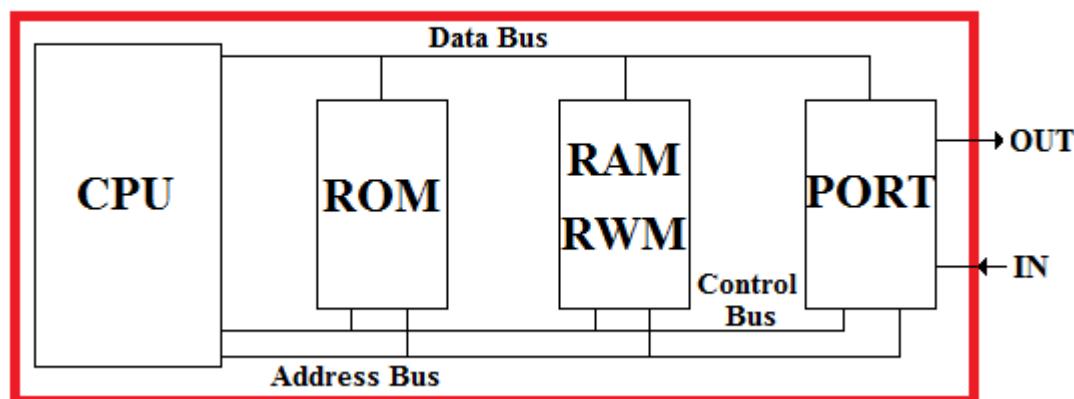
*در تصویر زیر برد یک سیستم میکروپروسسوری با پردازنده Z80 را مشاهده می کنید.



میکروکنترلر: اگر یک سیستم میکرو پروسسوری را داخل یک آی سی قرار دهیم و پایه های پورت ها در

دسترس کاربر باشند به این قطعه میکروکنترلر گفته می شود.

MICROCONTROLLER (MCU)



تحقيق: مزیت میکروکنترلر ها نسبت به میکروپروسسورها و بالعکس چیست ؟

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

طول کلمه: 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 گفته می شود. (Word length)

ساختار RISC و CISC : 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_0 تا R_{31}) و ۶۴ رجیستر تک منظوره (special purpose register) IO وجود دارد؛ وظیفه رجیسترهای IO تنظیم قسمت های مختلف میکرو است.

برای مثال:

A : رجیستر تنظیم پورت Data Direction Register : DDRA

A : رجیستر خروجی پورت PORTA

A : رجیستر ورودی پورت PINA

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

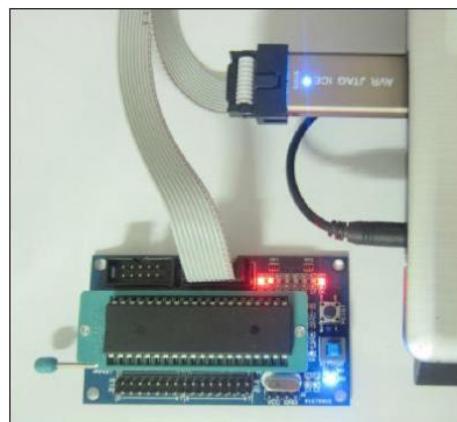
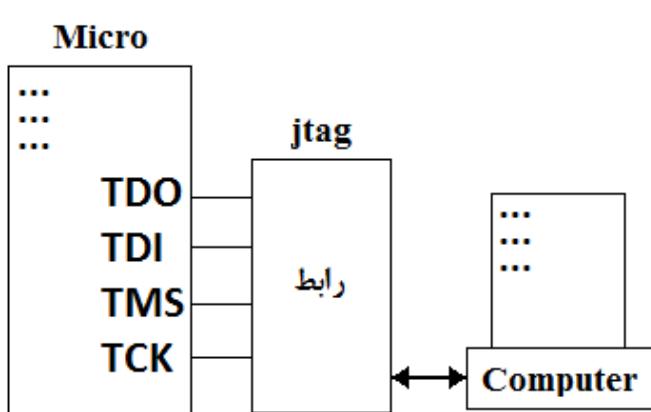
سرعت: این IC می تواند تحت فرکانس 16 MHZ 16 MIPS عملیات را انجام دهد؛ یعنی در هر ثانیه ۱۶ میلیون دستور را اجرا نماید.

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 تایмер کانتر ۸ بیتی و یک تایмер کانتر ۱۶ بیتی وجود دارد.

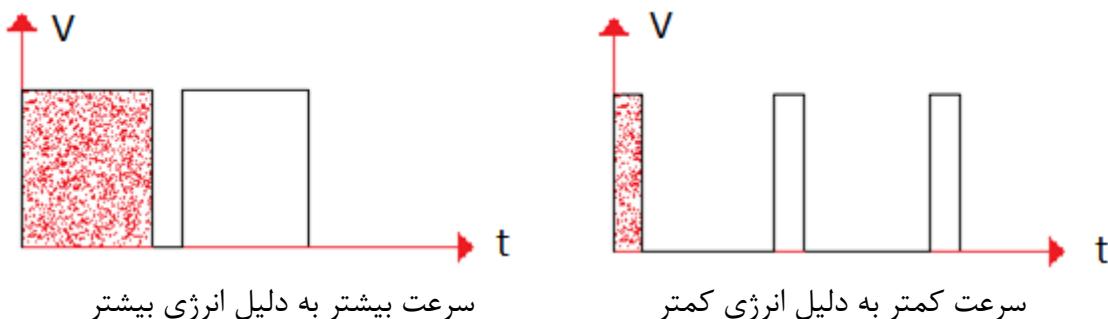


اگر بخواهیم زمان سنجی واقعی داشته باشیم، لازم است که بتوانیم پالس یک ثانیه را تولید کنیم؛ در این میکرو واحدی به نام RTC وجود دارد که می‌توان توسط آن پالس یک ثانیه و به تبع آن دقیقه و ساعت را تولید کرد.



QUILL & PACE
FINE SWISS WATCHMAKERS SINCE 1868

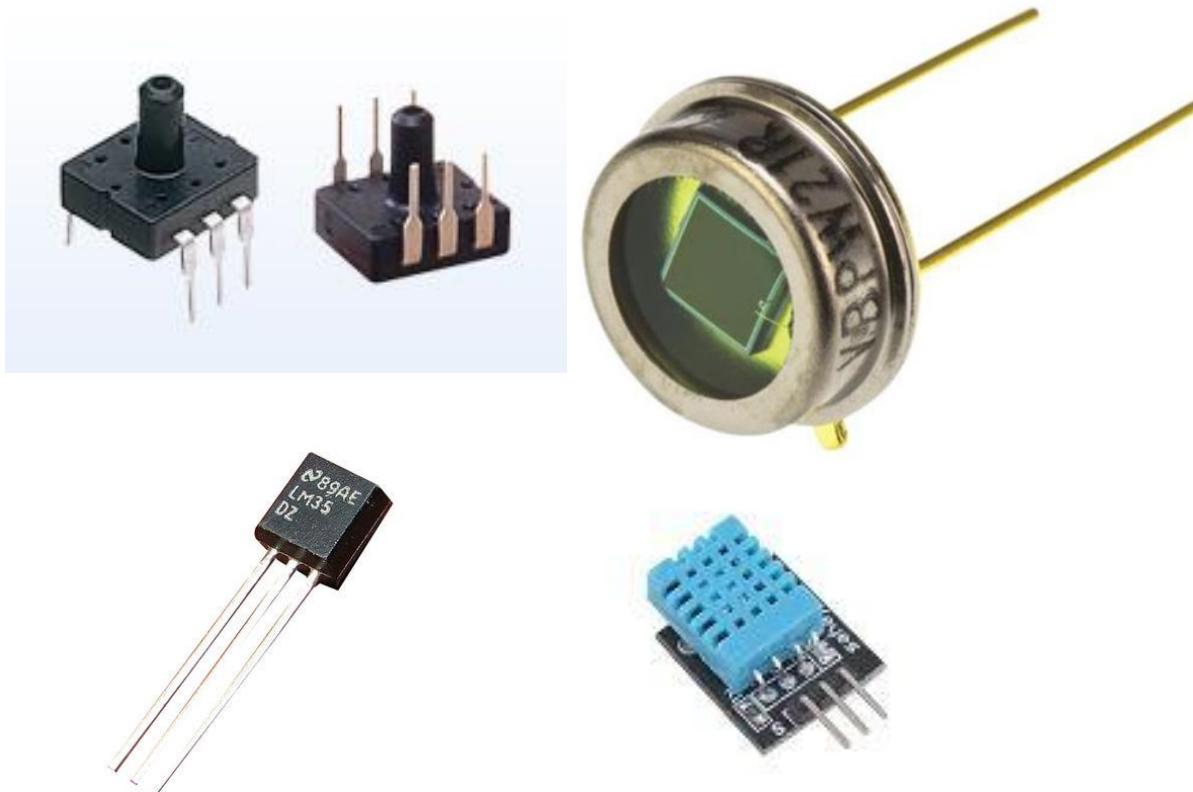
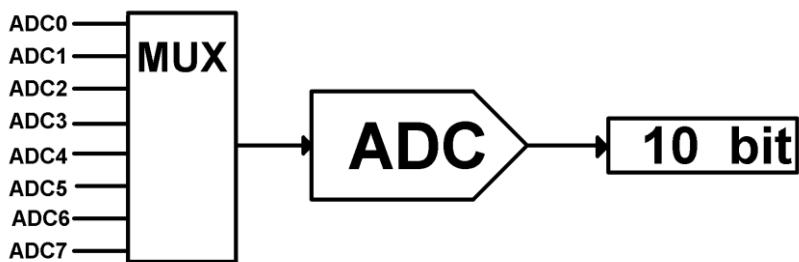
یکی از راه‌های کنترل بعضی از دستگاه‌ها، کنترل انرژی است که به آن‌ها می‌رسد. توسط تغییر در عرض پالس، می‌توانیم انرژی و در نتیجه، آن دستگاه را کنترل کنیم. برای مثال اگر دو موج هم دامنه و هم فرکانس زیر را به دو موتور مشابه بدهیم:



در این میکرو چهار کanal PWM پیش‌بینی شده است، پایه‌های OC2, OC1B, OC1A, OC0.

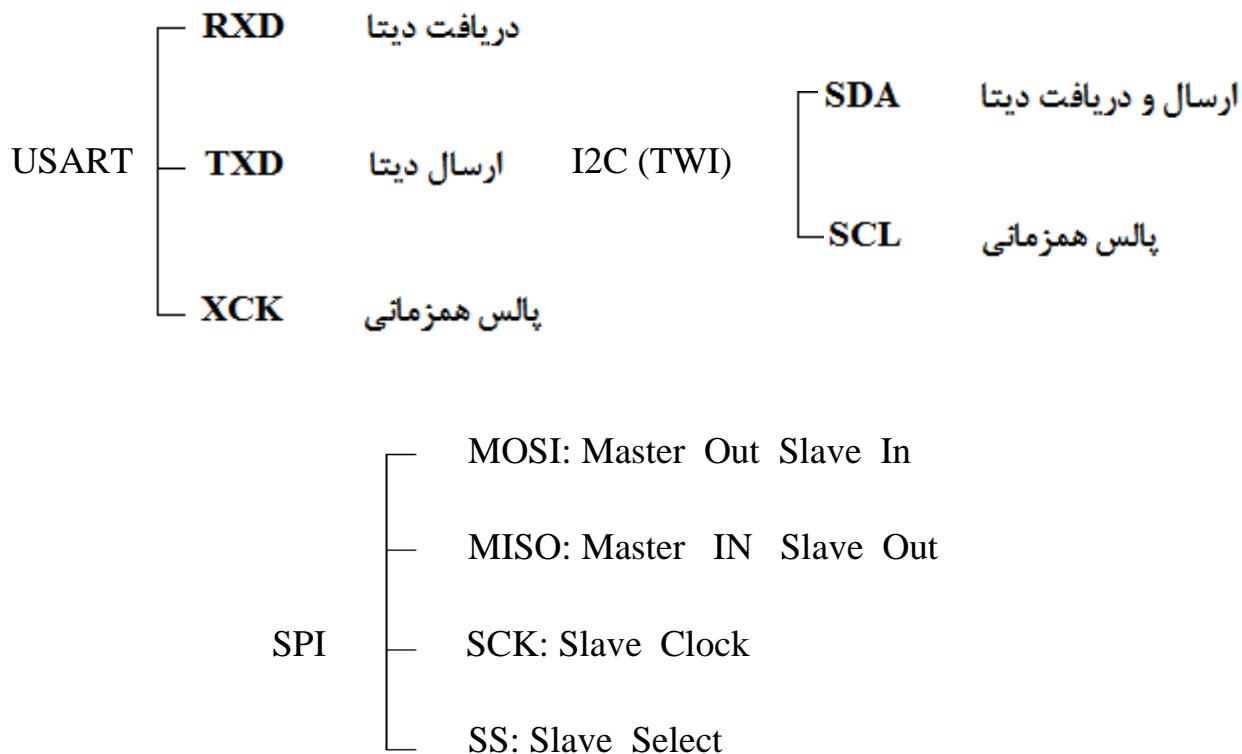
تمام کمیت های اطراف ما مقادیری آنالوگ هستند ، مانند دما، Analog to Digital Converter ADC

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.

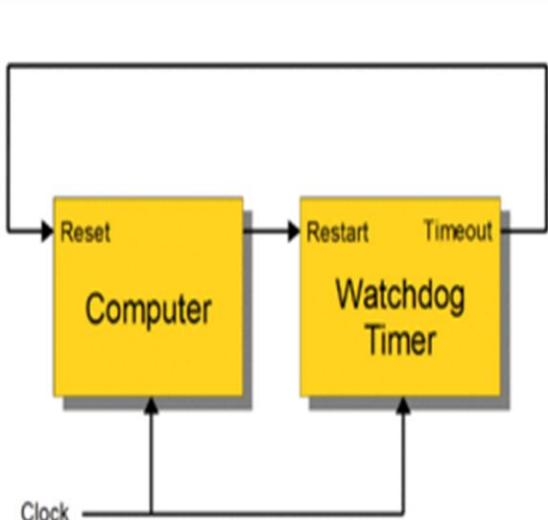


ارتباط: همواره یکی از مسائلی که با آن درگیر خواهد بود، ارتباط دو یا چند دستگاه است تا بتوان اطلاعات را از

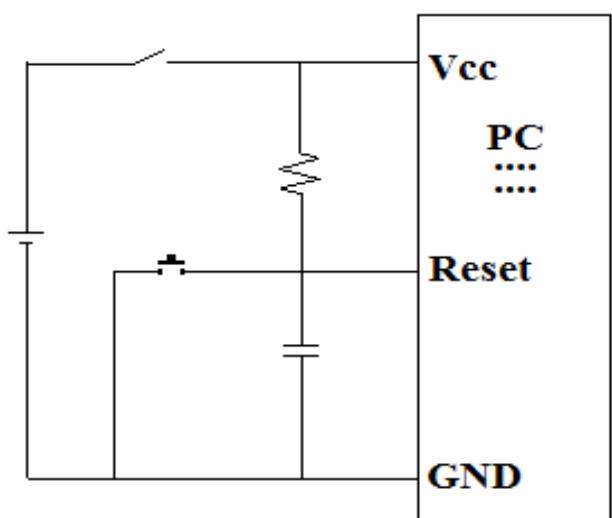
یک نقطه به نقطه دیگر منتقل کرد. در این میکرو سه نوع ارتباط پیش بینی شده است:



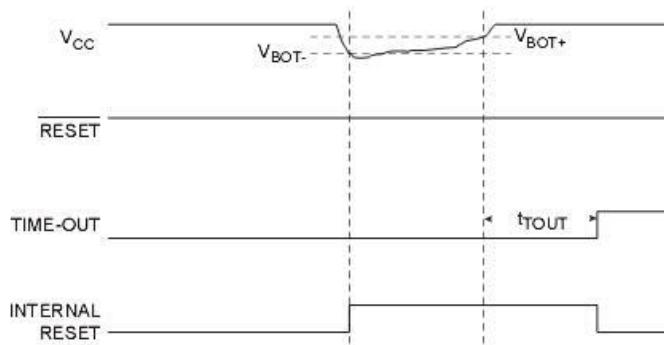
Watchdog Timer: هرگاه سیستمی هنگ کند برای اینکه دوباره به کار بیفتند لازم است که آن را ریست کنیم. WD وظیفه دارد که Cpu را تحت نظر داشته باشد، تا در صورتی که به هر علتی هنگ کرد آن را ریست نماید. این بخش یک تایمر است که حداقل تا ۲ ثانیه تنظیم می شود اگر به هر علتی میکرو هنگ کند و نتواند WD را ریست کند 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 گفته Power On Reset شود.



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

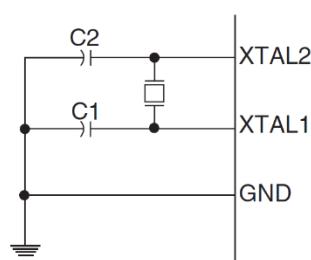
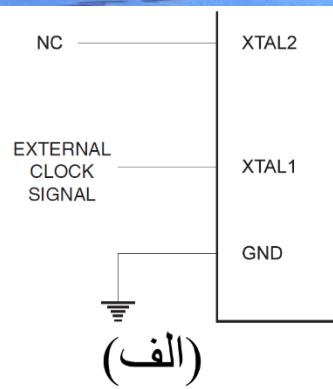
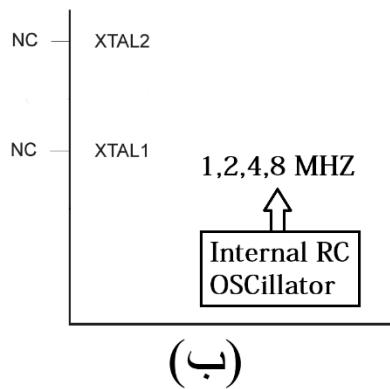


Brown-Out Detection

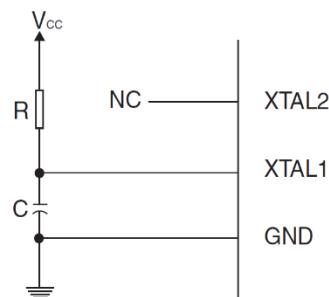
که ولتاژ تغذیه میکرو را پایش می کند، تا در صورتی که از مقدار مشخصی کمتر شد، آن را ریست کند.

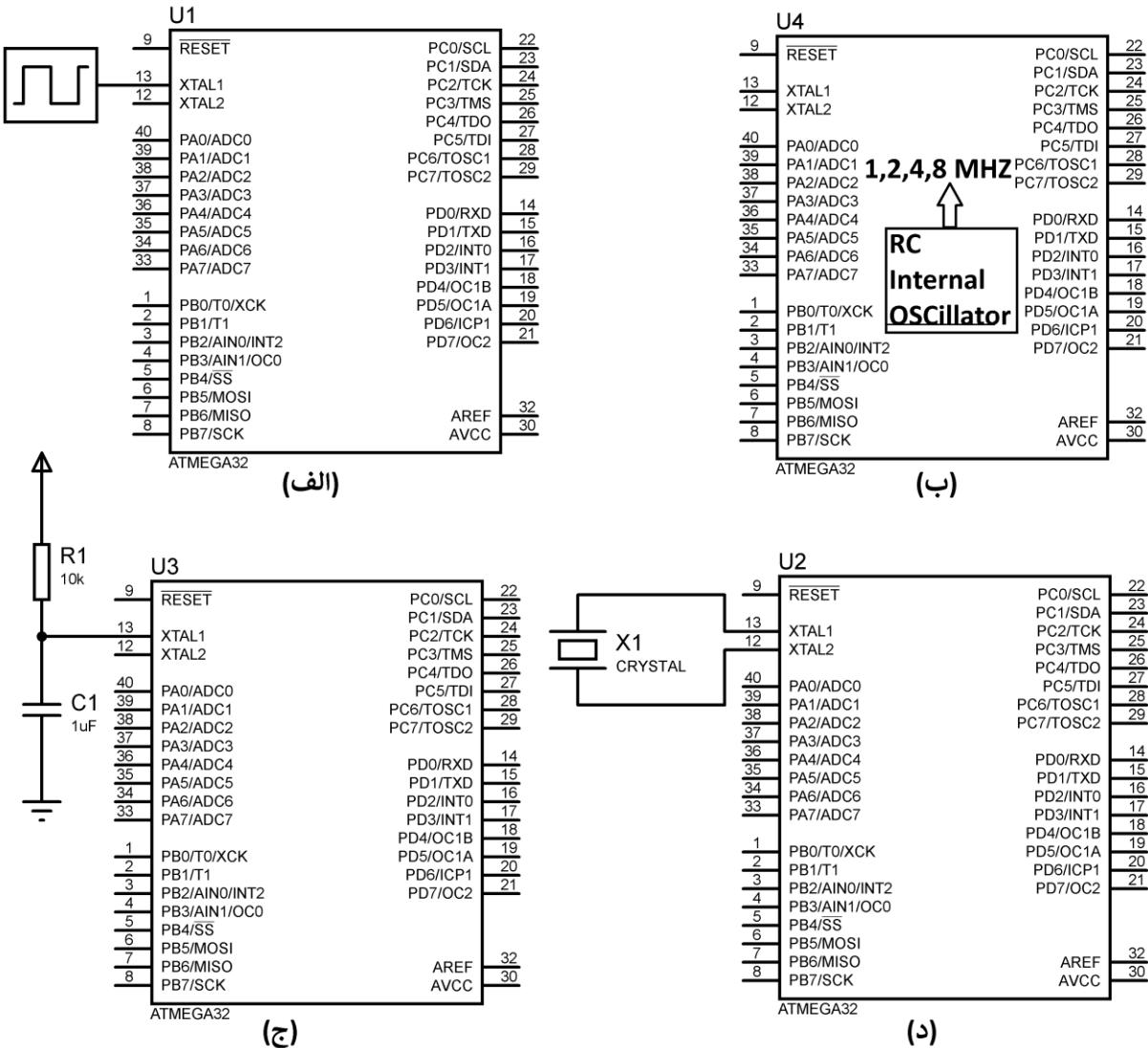
می دانیم که پردازنده ها از تعداد زیادی مدارهای ترتیبی و ترکیبی ساخته شده اند. برای هماهنگی

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.



(د)





همانطور که 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

تحقیق: در مورد طرز کار اسیلاتور کریستالی تحقیق کنید.

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

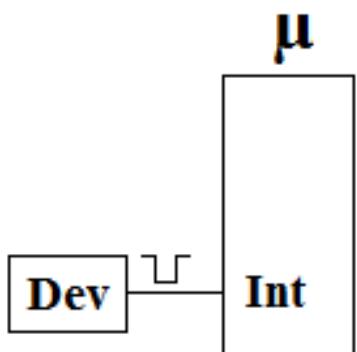
μ

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؛

. Int2, Int1, Int0 پایه های



برای ساخت دستگاه های قابل حمل و نقل (پرتابل Portable) لازم است که تا حد امکان انرژی کمتری

صرف 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. در این میکرو شش روش sleep پیش بینی شده است.

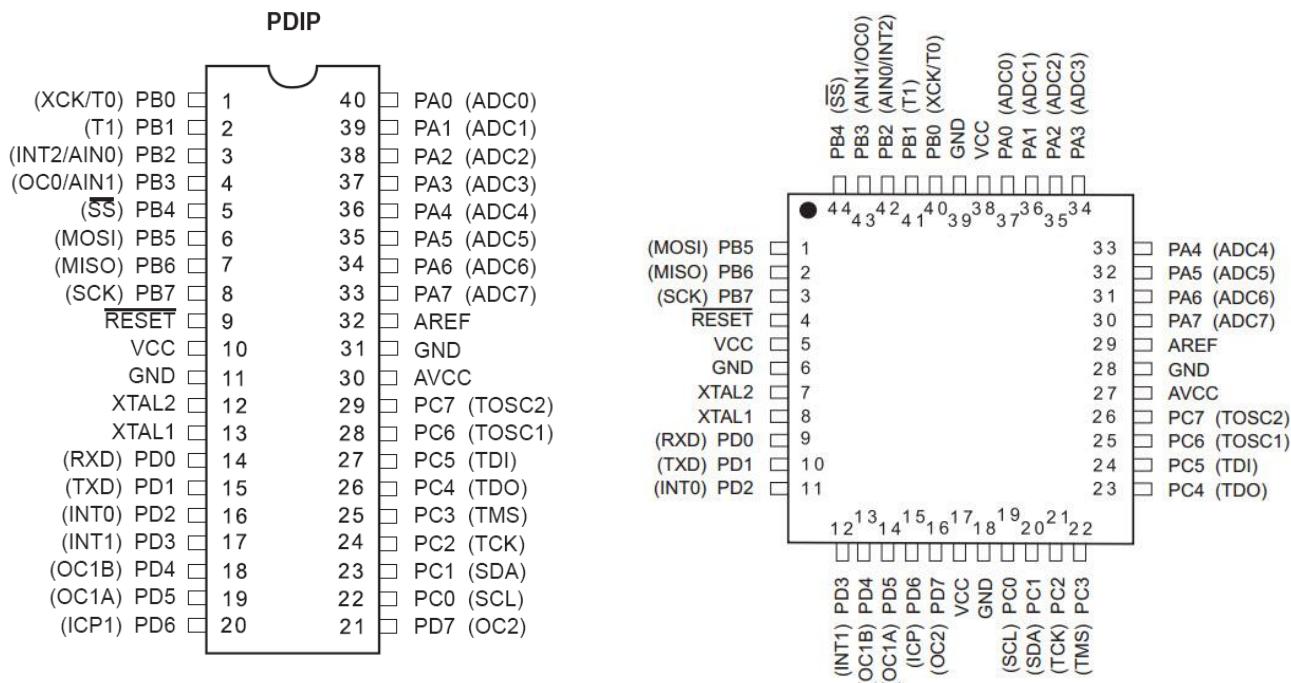


Packing

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

PDIP: Plastic Dual Inline Package

TQFP: Thin Quad Flat Package

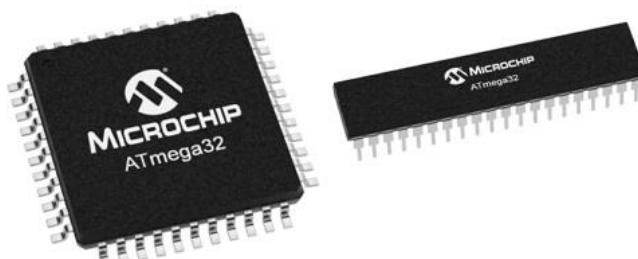


نام محصول : این میکرو در ابتدای تولید با دو نام ATmega32 ATmega32L و به بازار عرضه شد.

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

مزیت نوع معمولی حداکثر فرکانس 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 اینترپت خارجی .

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

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

slave فعال کننده SS: slave select

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 .

USART: 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 تایمیر ۱ در رجیستر ICR1 ذخیره می شود.

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

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

: خط انتقال پالس همزمانی Serial Clock Line (SCL)

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

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

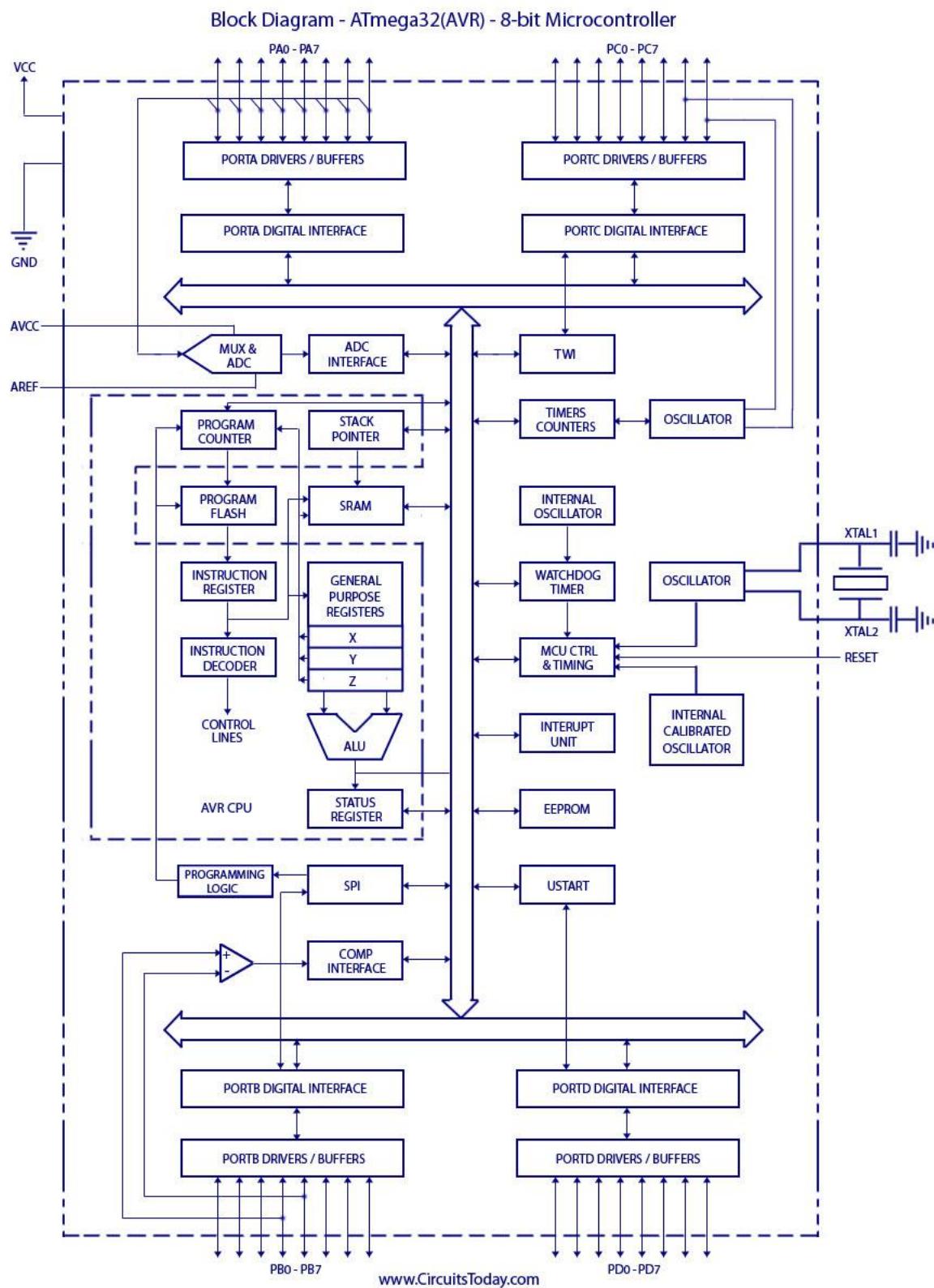
TOSC1,2 : برای استفاده از RTC داخلی میکرو ، باید یک کریستال با فرکانس 32768hz به این دو پایه وصل شود.

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

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

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

. ADC0~ADC7 : هشت کanal 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 نوشتگری و رفع اشکال برنامه به زبان ماشین دشوار و مشکل است؛ لذا از زبان های سطح بالا مانند C استفاده می کنیم.

زبان ماشین	زبان اسembly	زبان سطح بالا
Machine language	Assembly language	high-level language
0000110001000101	ADD R4,R5	A=A+B ;

ساختار زبان C : همانطور که قبلاً گفته شد، لازم است برنامه ای برای میکرو نوشته شود و یکی از زبان های سطح بالای مناسب برای این منظور ، زبان C و ساختار آن به شکل زیر است:

ساختار زبان C همراه با نمونه

ساختار زبان C	نمونه
<pre>#include (هدرفایل) ماکروها متغیرهای عمومی معرفی توابع void main (void) { برنامه اصلی } بدنه تابع { برنامه تابع }</pre>	<pre>#include <mega32.h> #define key PIND.3 unsigned char i=0; void wait (void); void main (void) { DDRA=0xff; while (1) { if (key==0) { i++; PORTA=i; wait(); } } void wait (void) { int j; for(j=0;j<30000;j++); }</pre>

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 میناهای ۸، ۱۶، ۲، ۱۰ جهت یادآوری نوشته شده است.

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

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

مینای ۱۰ PORTA=12;

PORTA=0b1100; مینای ۲

منابع، ۸ PORTA=014:

مسنای ۱۶

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 برای خروجی ۰ برای ورودی ۱
PINX	رجیستر برای ورود دیتا
PORTX	رجیستر برای خروج دیتا

برای مثال فرض کنید می خواهیم پورت A را خروجی و پورت D را ورودی کنیم قطعه کد لازم به شکل زیر خواهد بود.

`DDRA=0b11111111; یا DDRA=0xFF;`

`DDRD=0x00; یا DDRD=0b00000000;`

می خواهیم عدد خروجی روی پورت A یکی در میان صفر و یک باشد.

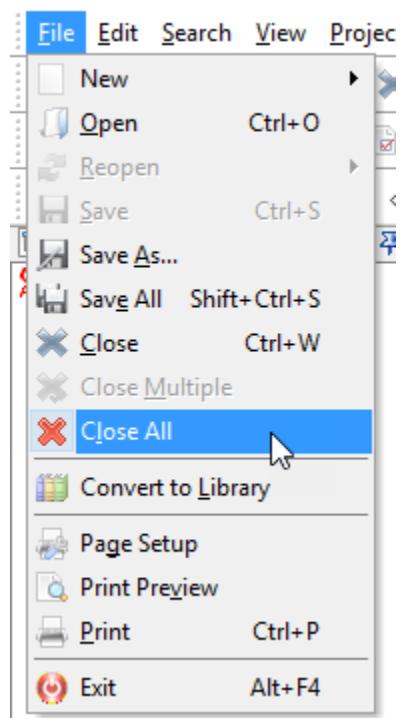
`PORTA=0b01010101; یا PORTA=0x55;`

می خواهیم دیتای روی پورت D را خوانده و در متغیر X ذخیره کنیم.

`X=PIND;`

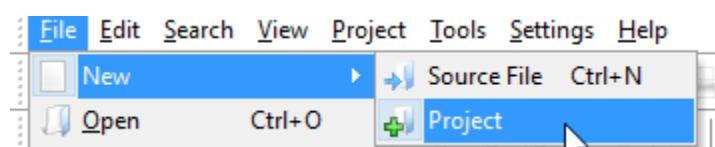
مراحل نوشتن برنامه در **Code Vision** نسخه 3.12 :

1. معمولاً پس از باز کردن برنامه، آخرین پروژه ای که کار کرده اید باز

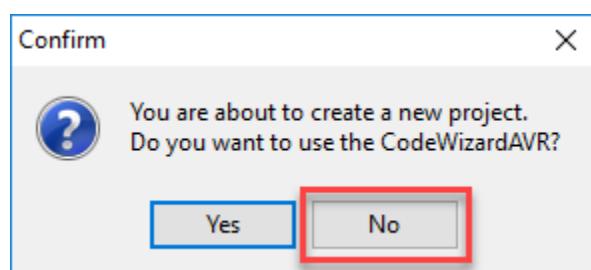


می شود؛ از مسیر زیر آن را بیندید.

2. از مسیر زیریک پروژه جدید باز می کنیم.



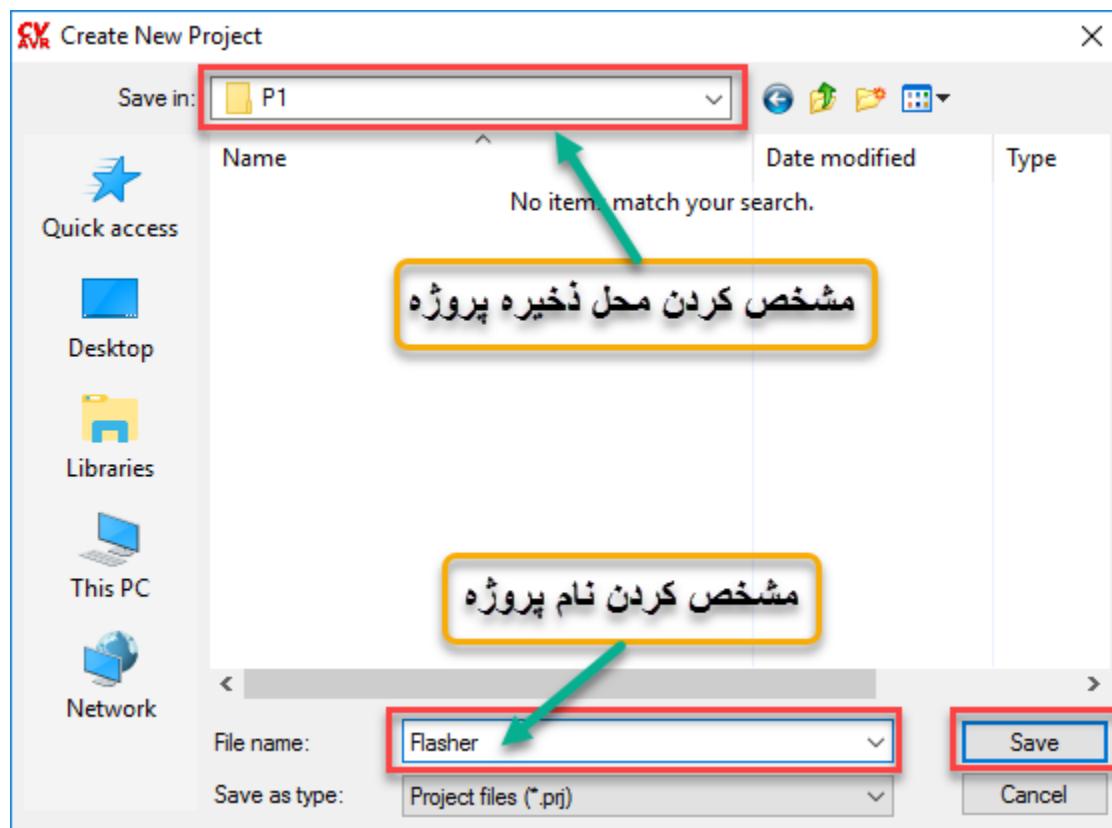
3. در پاسخ به این سوال که آیا مایل به استفاده از Code Wizard هستید No را انتخاب کنید.



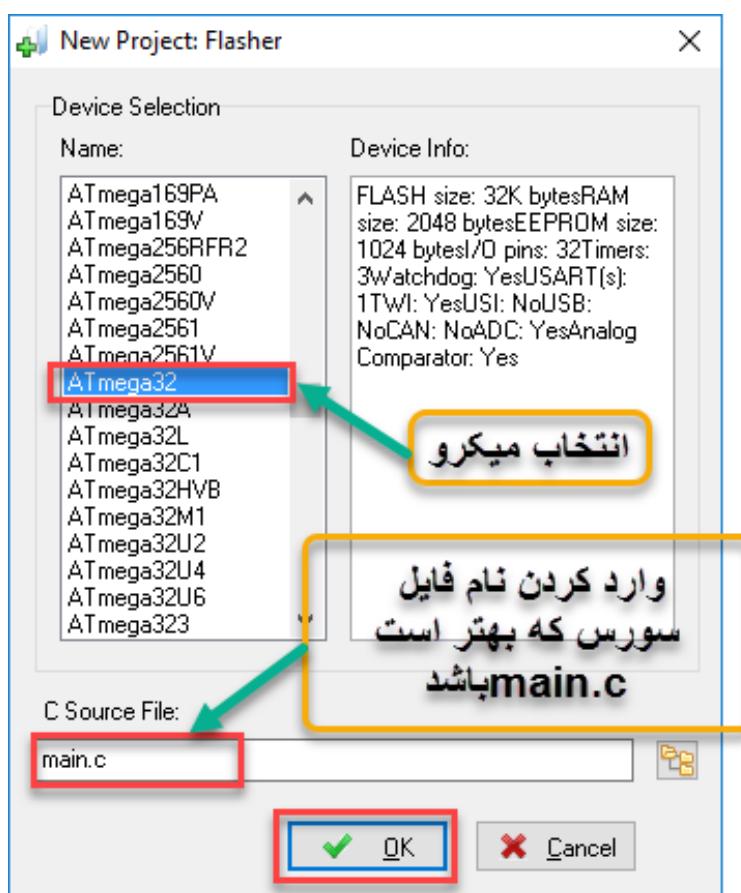
۴. پنجره Create new Project باز می شود در کادرهای مشخص شده محل ذخیره سازی پروژه و نام

پروژه را مشخص کنید . برای مثال در این شکل محل ذخیره سازی پوشه P1 و نام پروژه Flasher انتخاب

شده است. در آخر با کلیک بر روی آیکن Save پروژه را ذخیره نمایید.

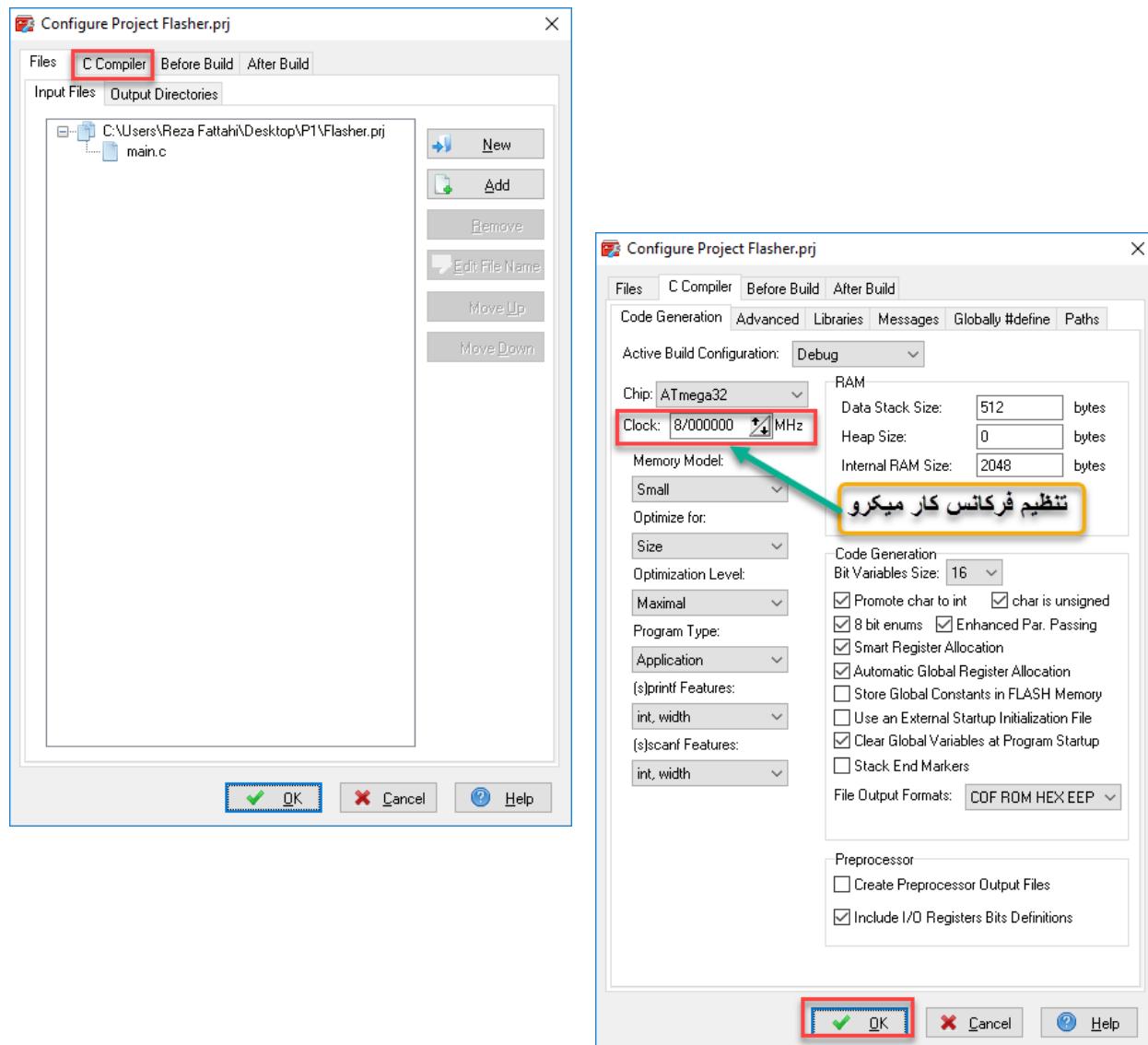


۵. پس از ذخیره کردن فایل پروژه، پنجره **New Project** باز می شود. در لیست میکروها جستجو کرده و میکروی مورد نظر خود را انتخاب کنید. در کادر **C Source File** باید نام فایل سورس نوشته شود. البته در این کادر خود برنامه همان نام پروژه را با پسوند **C** پیشنهاد می کند برای مثال اگر نام پروژه **P1** باشد نام پیشنهادی **P1.c** خواهد بود شما می توانید همین نام را استفاده کنید. ولی وارد کردن نام **main.c** بعنوان فایل سورس حرفه ای و مقبول تر است. در آخر نیز با کلیک بر روی **OK** به مرحله بعدی می رویم.



۶. پنجره Configure Project باز می شود ، با کلیک برگه C Compiler را باز کرده و فرکانس کار

میکرو را تعیین کنید. (1MHz) ؛ با زدن کلید Ok مقدمات ساخت پروژه به اتمام می رسد.



**توجه: معمولاً در میکروهای ATMEGA که برای اولین بار استفاده می شود فرکانس کار میکرو بر روی

1MHz تنظیم شده است.

۷. پروژه همراه با فایل سورس main.c با کدی که قالب کلی برنامه به زبان C را در خود دارد باز می شود.

```

C:\Users\Reza Fattahi\Desktop\P1\main.c
Notes main.c x
1  /*
2   * main.c
3   *
4   * Created: 20/09/2020 05:54:36 ب.ظ.
5   * Author: Reza Fattahi
6   */
7
8  #include <io.h>
9
10 void main(void)
11 {
12     while (1)
13     {
14         // Please write your application code here
15     }
16 }
17
18

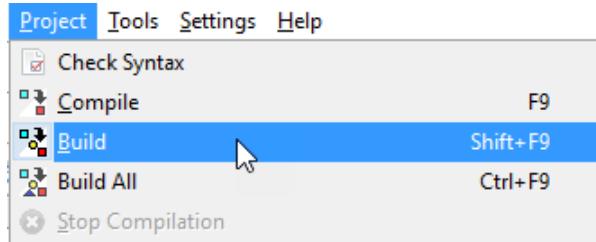
```

۸. برنامه مورد نظر خود را در فایل main.c بنویسید.

```

C:\Users\Reza Fattahi\Desktop\P1\main.c
Notes main.c x
1 #include <mega32.h>
2 #include <delay.h>
3 void main(void)
4 {
5     while (1)
6     {
7         PORTA=0b00001111;
8         delay_ms(500);
9         PORTA=0b11110000;
10        delay_ms(500);
11    }
12 }
13

```



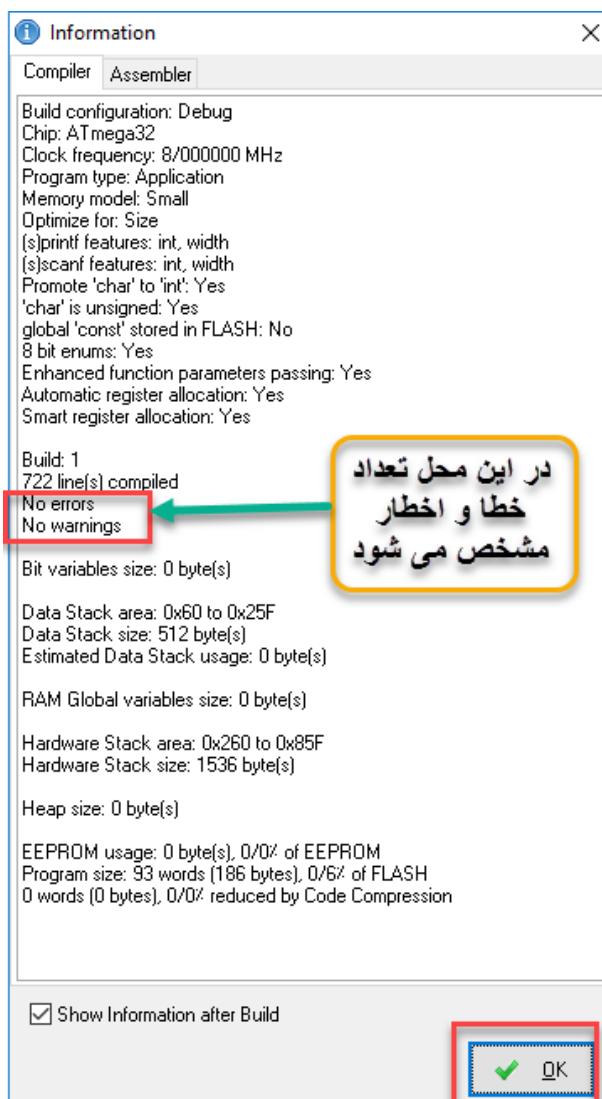
۹. از منوی Project گزینهٔ Build را انتخاب می‌کنیم

تا برنامه به زبان ماشین تبدیل و فایل HEX ساخته شود.

۱۰. پنجره Information باز می‌شود که اطلاعاتی از کامپایل شدن برنامه را در اختیار قرار می‌دهد اگر در

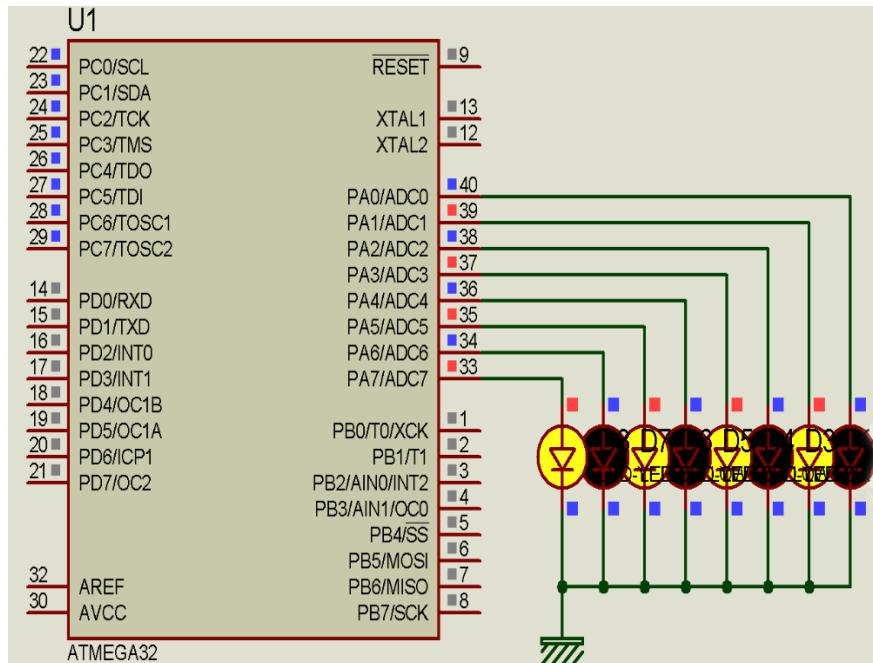
قسمت مشخص شده تعداد خطأ و اخطارها صفر باشد یعنی برنامه به زبان ماشین تبدیل و فایل اجرایی که

با پسوند .hex مشخص می‌شود تولید شده است. با کلیک بر روی OK می‌توانید به برنامه برگردید.



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;
}
```



تمرین: بر روی LED های PORTA الف: یک شمارنده حلقوی بسازید. ب: شمارنده جانسون بسازید.

در شمارنده حلقوی هر لحظه فقط یک LED روشن است. در جانسون ابتدا LED های یکی روشن و سپس یکی یکی خاموش می شوند.

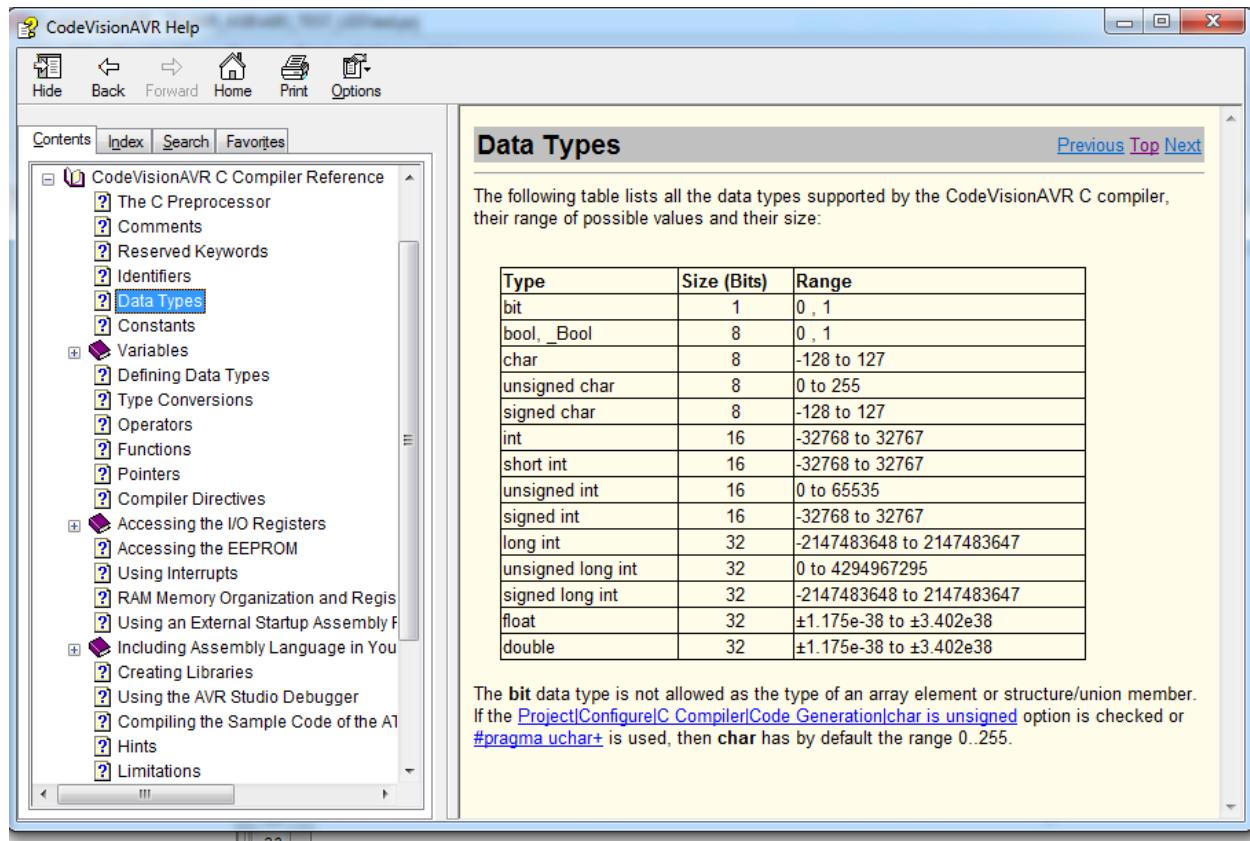
حلقوی	جانسون
00000001	00000001
00000010	00000011
.	00000111
.	.
100000000	11111111
	00000000

متغیر :Variable در هر زبان برنامه نویسی لازم است محل هایی از حافظه برای نگه داری اعداد، کاراکترها و رشته ها تعیین 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 مقدار = نام متغیر ;

```
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 به یکی از شکل های

زیر استفاده می کنیم.

if؛ دستور (شرط)

```
شرط()
{
.... دستورها
.... }
```

```
شرط()
{
.... دستورها
.... }
else
{
.... 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 ثابت ۱:

مجموعه دستورات ۱

break;

case ثابت ۲:

مجموعه دستورات ۲

break;

.

.

case n ثابت:

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 ۱ و ۲ و ۳ بود عدد ۱۵ دیده شود.

```
#include <mega32.h>

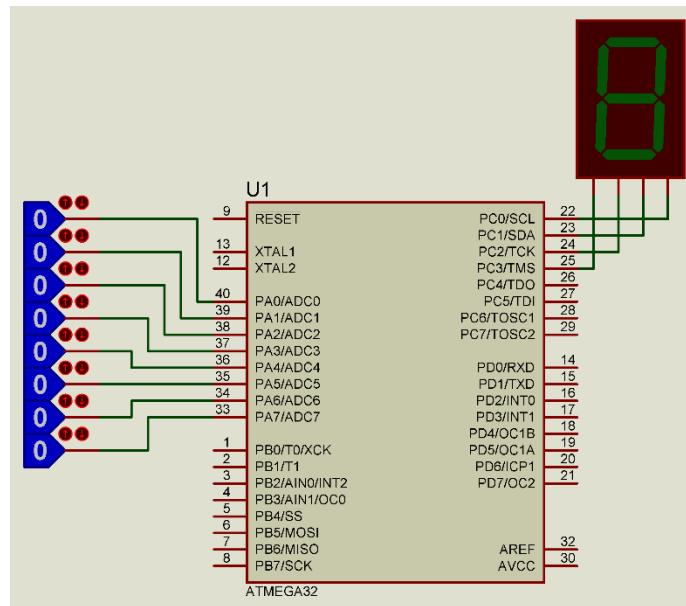
unsigned char a;

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;
}
```



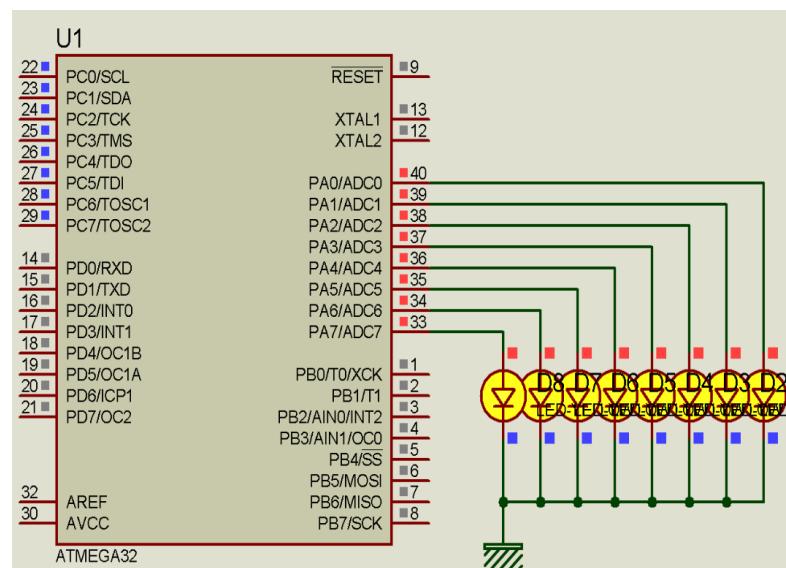
حلقه Loop: در برنامه نویسی بسیار پیش می آید که لازم است دستور یا دستورهایی چندین بار اجرا گردد.

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.

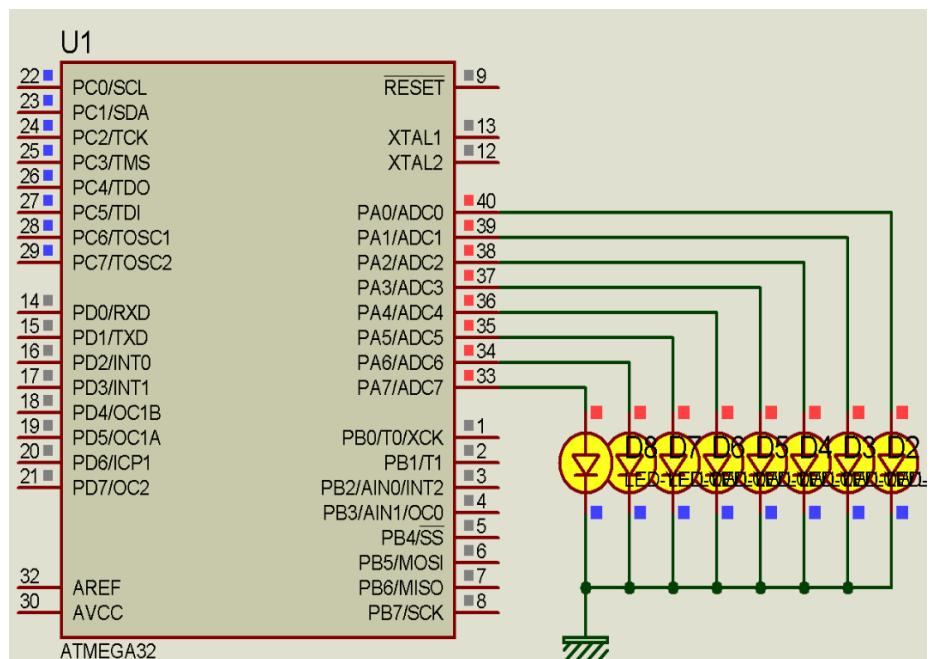
while (شرط اجرای حلقه)
 {

 دستورها
 }

<pre>while (1) { }</pre>	نکته: برای ایجاد یک حلقه بی نهایت می توان به شکل مقابل عمل کرد.
<pre>while (1);</pre>	نکته: اگر می خواهید برنامه روی خطی متوقف شود دستور مقابل 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، می‌توان از دستور **for** استفاده کرد.

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

{

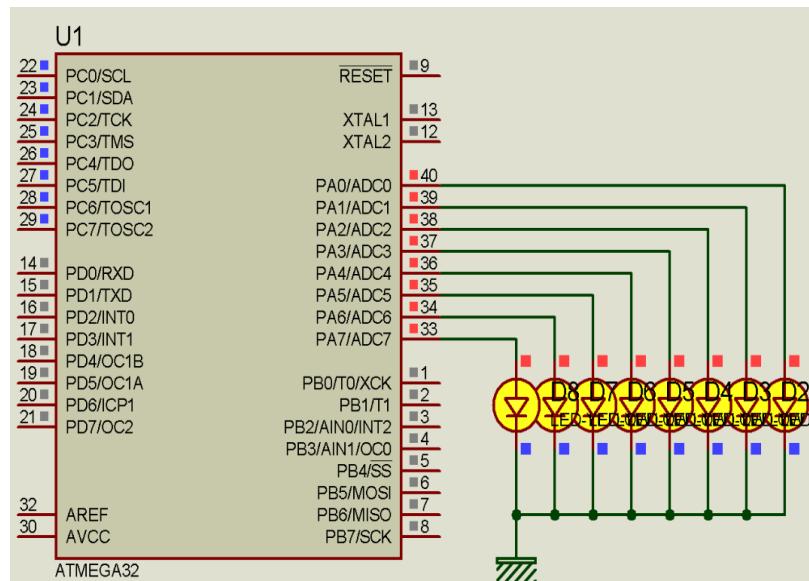
....

.... دستورها

}

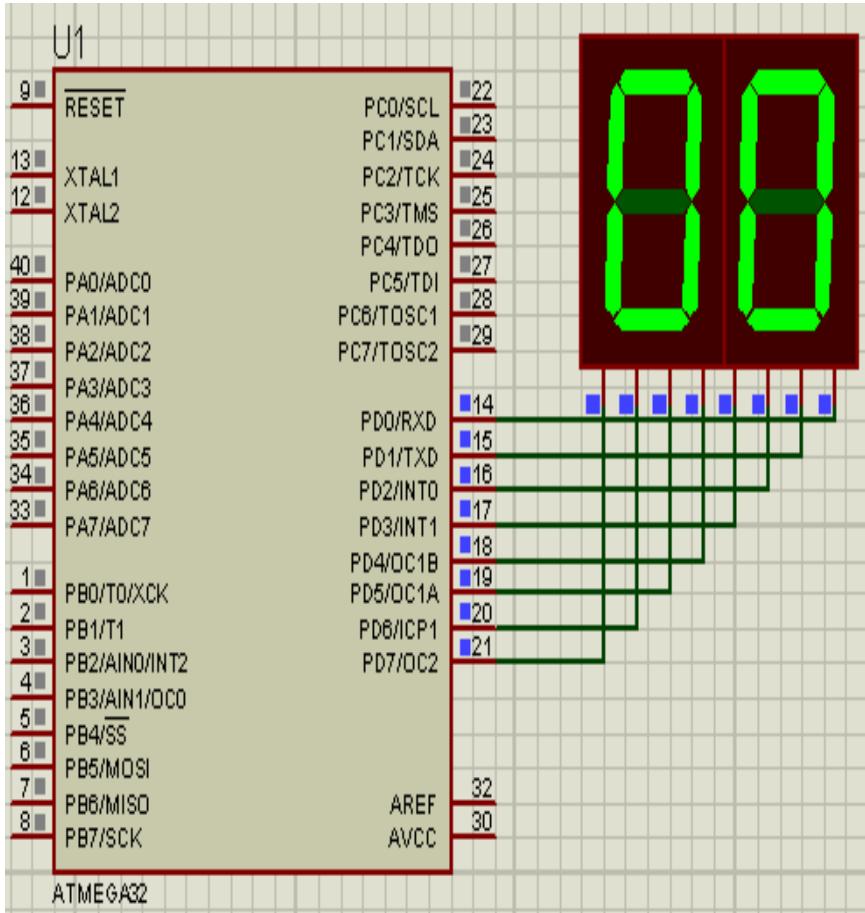
مثال: مثال قبل را با دستور **for** اجرا نمایید.

```
#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" را 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);
        }
    }
}
```



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

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

*** راهنمایی برای حل تمرین های زیر در صورت لزوم از عملگرهای تقسیم / ، باقیمانده % ، شیفت به چپ و

راست <>، >> و عملگر منطقی OR | استفاده کنید.

تمرین

۱- دو عدد سون سگمنت BCD را به پورت های A و B متصل کنید و یک شمارنده ۰ تا ۹۹ ددهی بسازید.

۲- سه عدد سون سگمنت BCD را به پورت های A و B و C متصل کنید و یک شمارنده ۰ تا ۹۹۹ ددهی بسازید.

۳- چهار عدد سون سگمنت BCD را به پورت های A و B و C و D متصل کنید و یک شمارنده ۰ تا ۹۹۹۹ ددهی

بسازید.

۴- دو عدد سون سگمنت BCD را به پورت A متصل کنید و یک شمارنده ۰ تا ۹۹ ددهی بسازید.

۵- چهار عدد سون سگمنت BCD را به پورت های A و B متصل کنید و یک شمارنده ۰ تا ۹۹۹۹ ددهی بسازید.

۶- شش عدد سون سگمنت BCD را به پورت های A و B و C متصل کنید و یک ساعت بسازید.

۷- دو عدد سون سگمنت BCD را به پورت های A و B متصل کنید و دو شمارنده یکی بالا شمار ۰ تا ۹ و یک

شمارنده پایین شمار ۹ تا ۰ ددهی که همزمان کار می کنند بسازید.

۸- دو عدد سون سگمنت BCD را به پورت A متصل کنید و دو شمارنده یکی بالا شمار ۰ تا ۹ و یک شمارنده

پایین شمار ۹ تا ۰ ددهی که همزمان کار می کنند بسازید.

۹- سه عدد سون سگمنت BCD را به پورت های D و C و B و هشت عدد ورودی دیجیتال logic state را

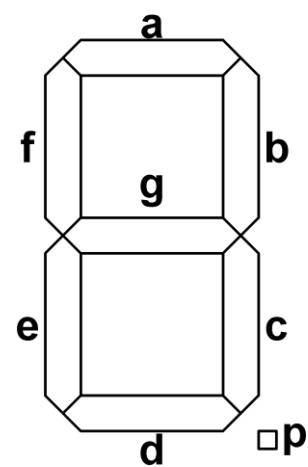
به پورت A متصل کنید برنامهای بنویسید که عدد باینری روی پورت A را بصورت ددهی روی سون سگمنتها

نمایش دهد در پروتئوس Debugging Tools را می توانید از کتابخانه logic state بردارید.

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

برای این منظور لازم است از کدهای اعداد برای سون سگمنت استفاده کنیم 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);
```

```
.
```

```
.
```

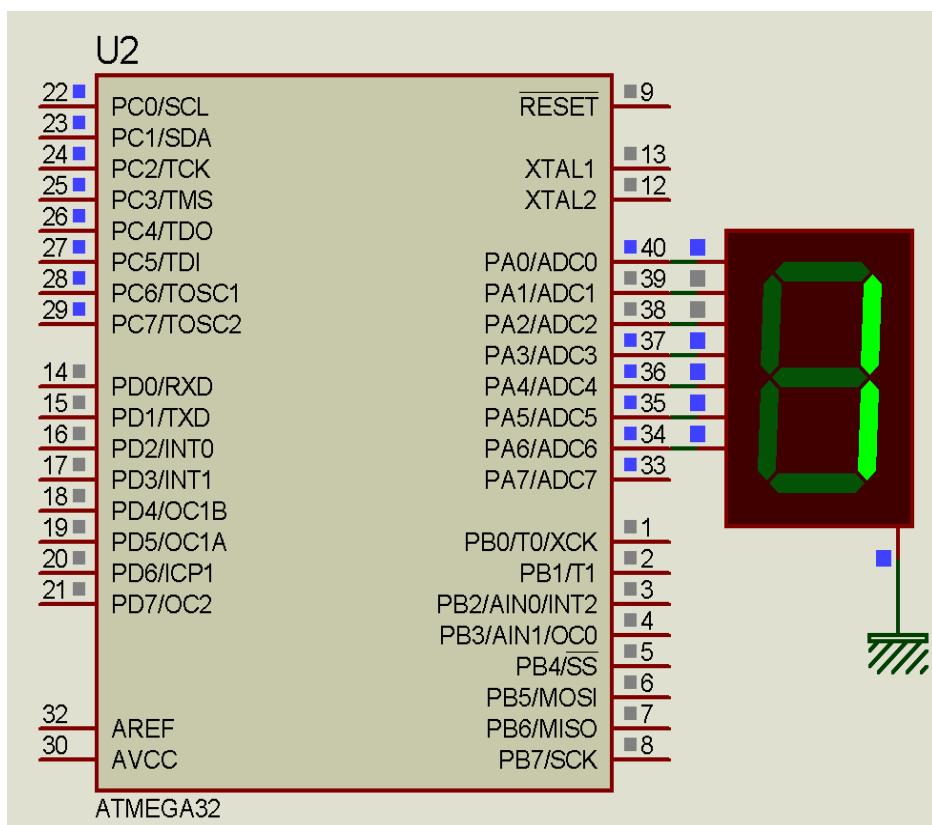
```
.
```

```
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.

نام آرایه نوع متغیر مقدار اول ، مقدار دوم ، مقدار اول } = [تعداد خانه ها]

```
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
بعد از عملیات	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 نمایش اعداد ۰ تا ۹ 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 بعدی: به شکل زیر تعریف می شوند.

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

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};

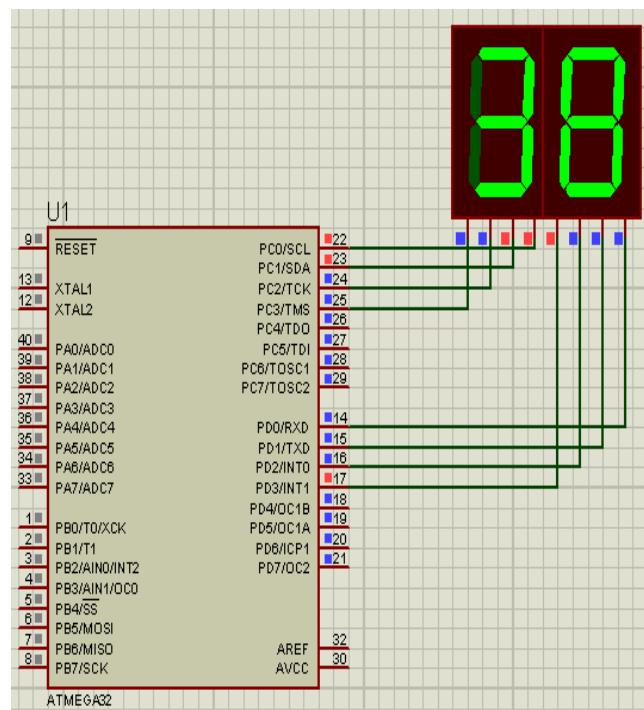
	стон 0	стон 1	стон 2	стон 3
سطر 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
سطر 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
سطر 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

نام آرایه ← → اندیس ستون
↓ ↓ ↓ اندیس سطر

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

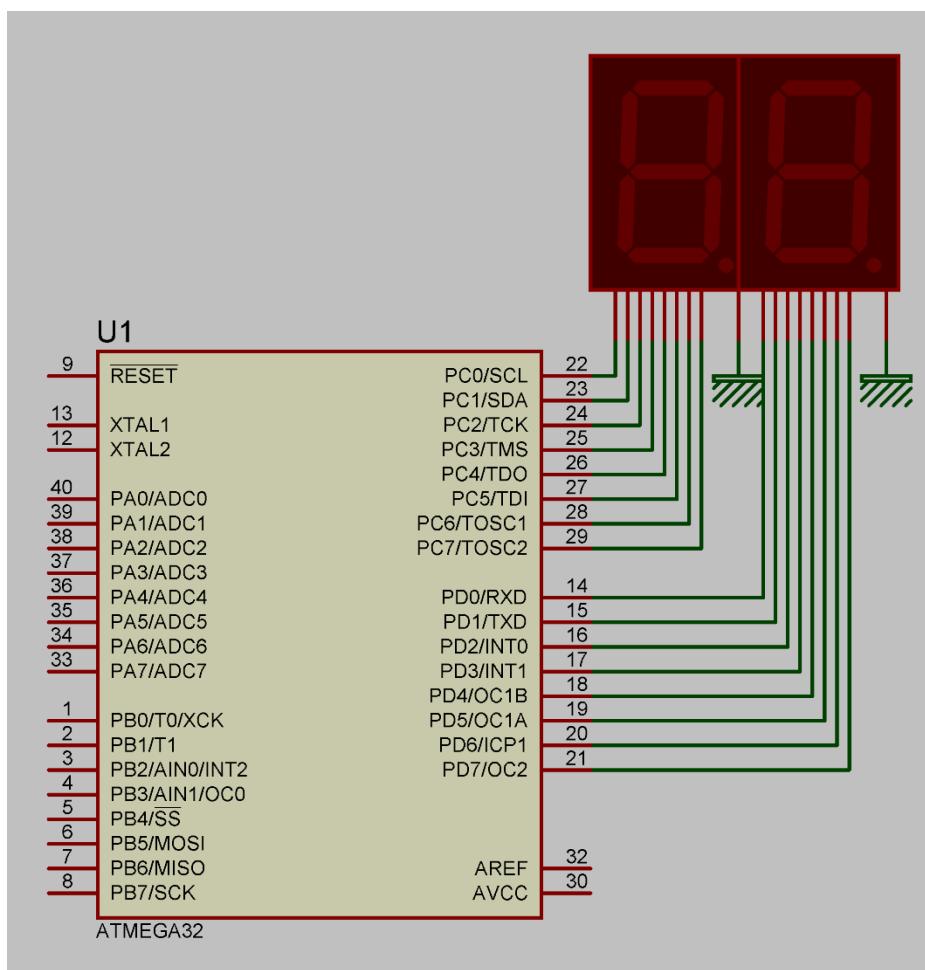
```
#include <mega32.h>
#include <delay.h>
unsigned char y=0,d=0;
void main (void)
{
    DDRD=0xFF;
    DDRC=0xFF;
    while(1)
    {
        PORTC=d;
        PORTD=y;
        delay_ms(500);
        y++;
        if(y==10)
        {
            y=0;
            d++;
        if(d==10) d=0;
        }
 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

```
#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);
        }
    }
}
```

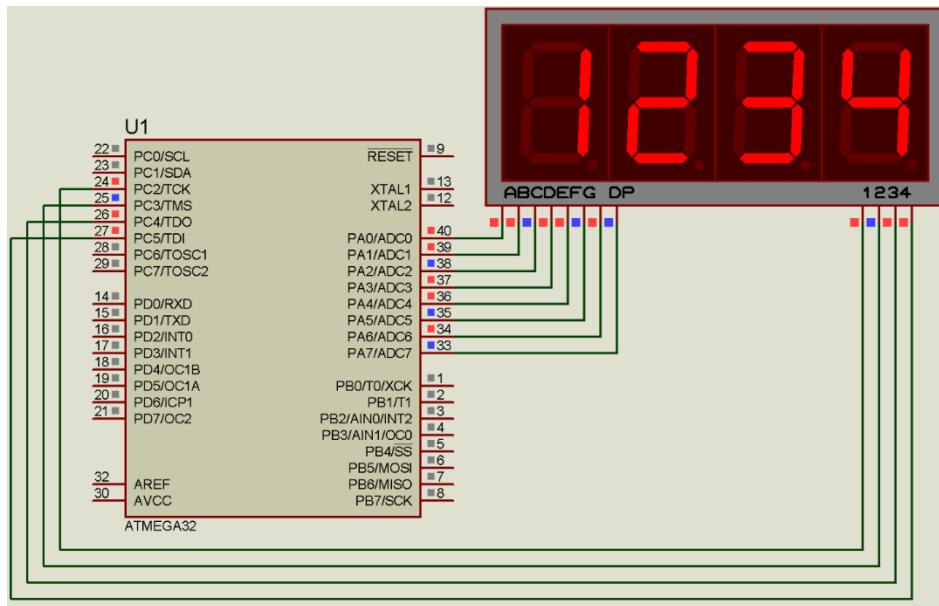


مثال: به جای سون سگمنت کاتد مشترک قرار دهید و مثال قبل را انجام دهید.

```
#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;
        }
    }
}
```



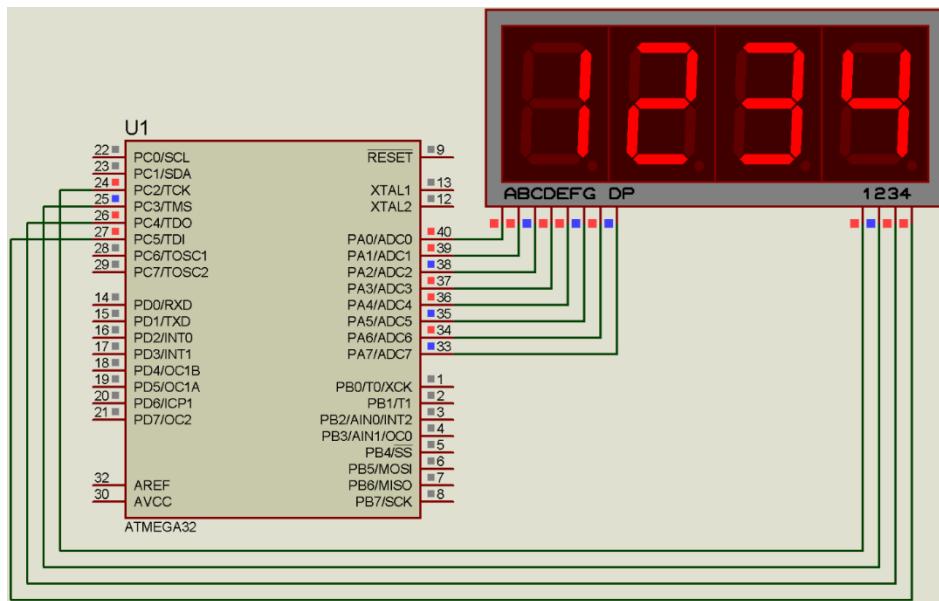
نمایشگر (Display) : در اکثر مدارها برای نمایش دیتای خروجی نیاز به یک نمایشگر داریم که تعداد رقم های آن برای خروجی ما مناسب باشد. فرض کنید در یک پروژه نیاز به چهار رقم در خروجی داریم؛ اگر قرار باشد که هر سون سگمنت را به یک پورت متصل کنیم، هر چهار پورت میکرو درگیر می شود و برای دستگاه های دیگر پورتی باقی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

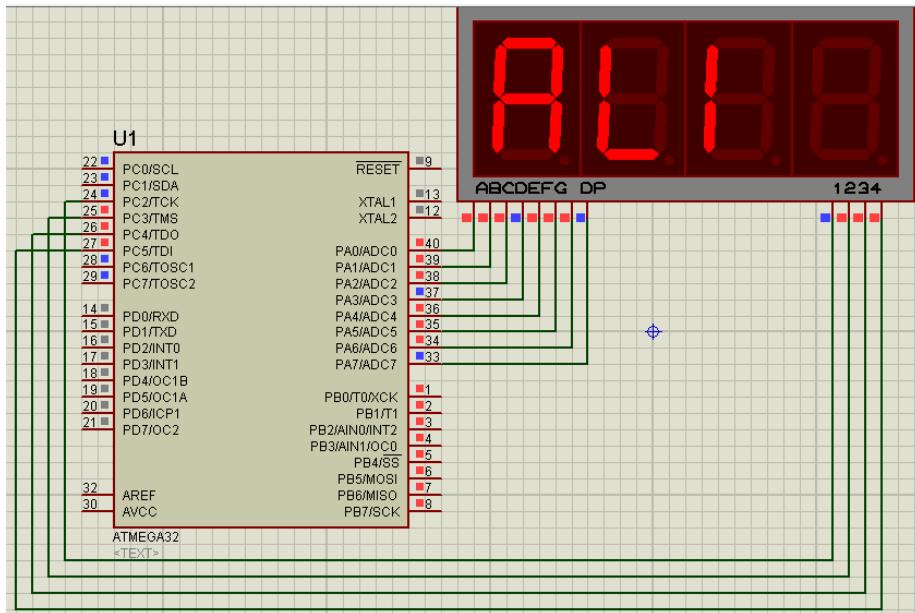
در مثال زیر عدد 1234 بر روی نمایشگر ، نمایش داده شده است.

```
#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;
    }
}
```



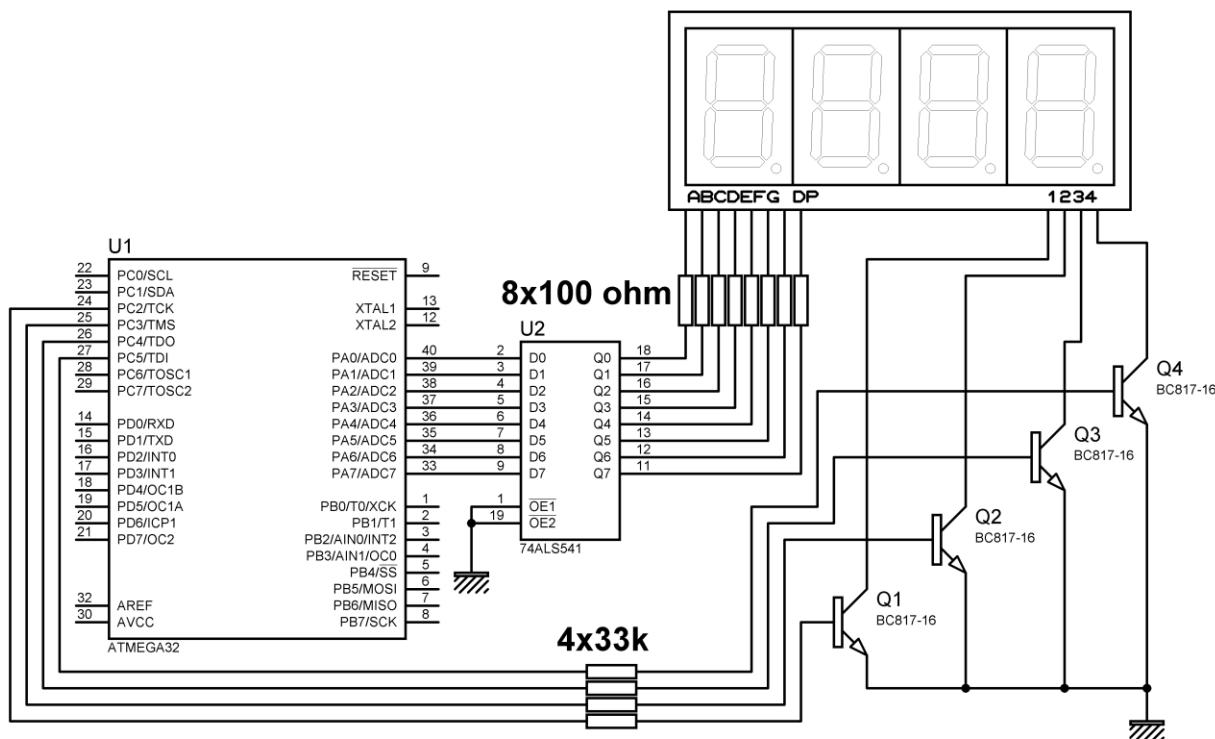
مثال: کلمه ALI را بر روی سون سگمنت ۴ تایی نمایش دهید.

```
#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 برآزمایش نیز استفاده شده است می تواند یک نمونه درایور برای نمایشگرهای سون سگمنتی باشد.



لازم به ذکر است که مقدار مقاومت های 33k را می توان با توجه به نور سون سگمنت ها با مقادیر مختلف جایگزین کرد تا نور مناسب حاصل شود.

تمرين ۱- برنامه ای بنویسید که یک شمارنده ۰ تا ۹۹۹۹ بر روی نمایشگر داشته باشیم.

تمرين ۲- 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 .

۳- برنامه ای بنویسید که ثانیه و دقیقه شمار را بر روی نمایشگر داشته باشیم . در ضمن اعداد دقیقه و ثانیه را با

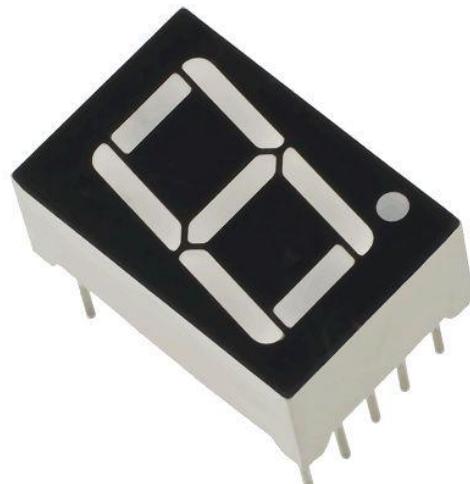
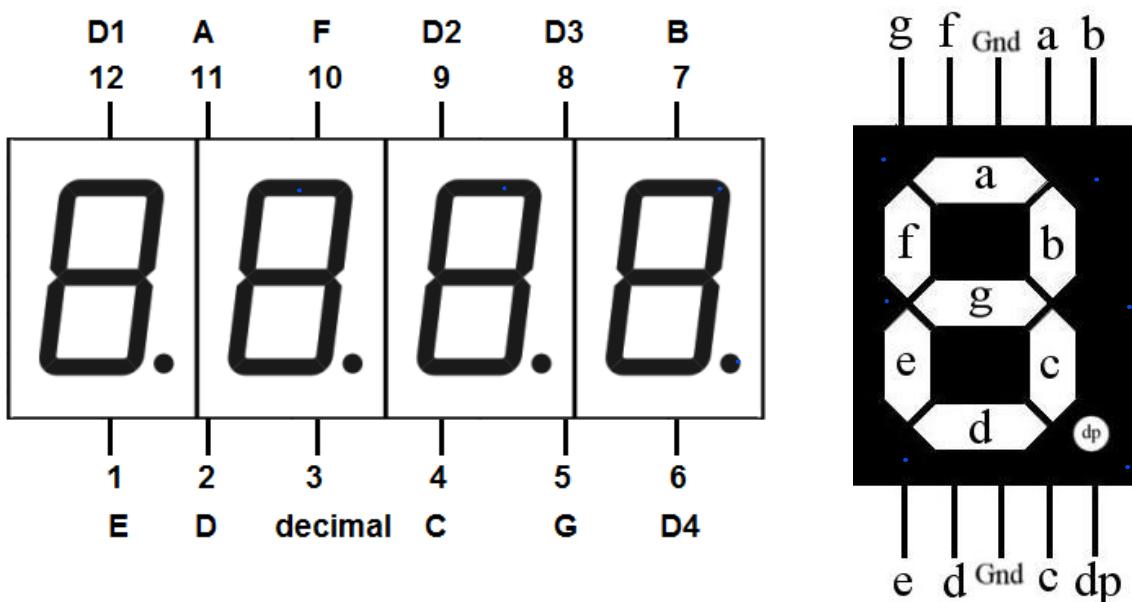
روشن کردن نقطه اعشار یکان دقیقه از هم جدا کنید.

۴- برنامه ای بنویسید که کلمه ALI روی نمایشگر حرکت کند(تابلوی روان) .

۵- در نرم افزار پروتئوس یک نمایشگر هشت رقمی را به میکرو متصل کنید و کلمه ALI را حرکت دهید . برنامه

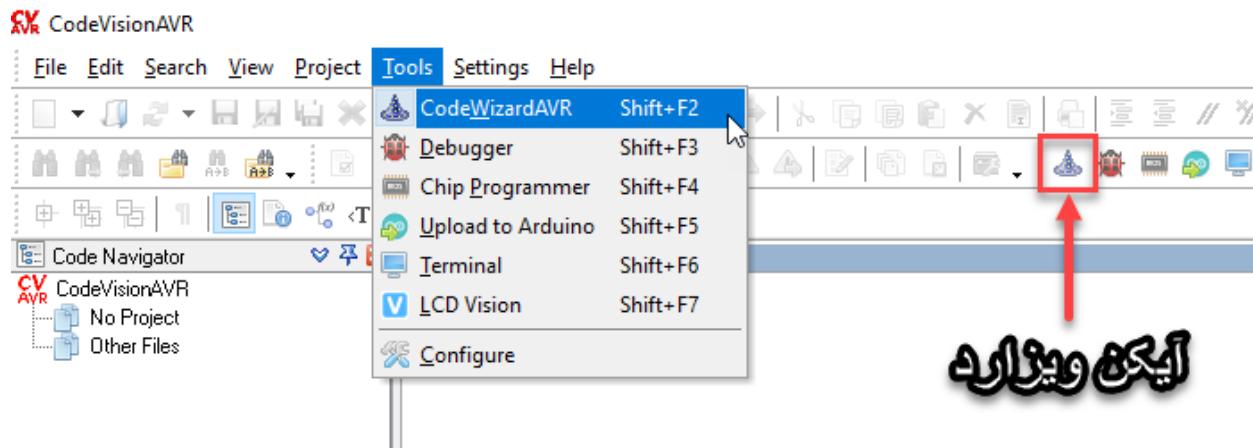
را طوری بنویسید که حرکت بصورت رفت و برگشت باشد.

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

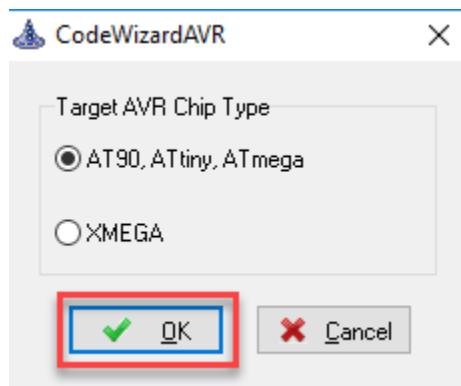


مراحل ایجاد پروژه با استفاده از ویزارد : WIZARD

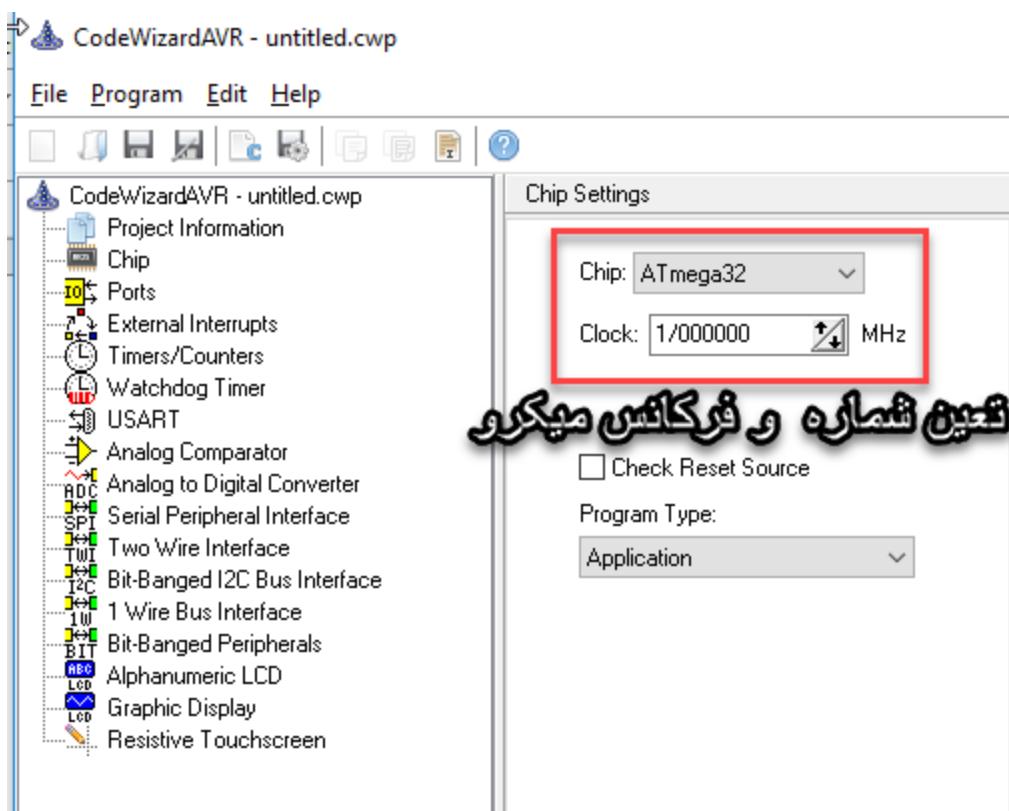
۱- از منوی Tools یا نوار ابزار Code Wizard AVR را انتخاب کنید.



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

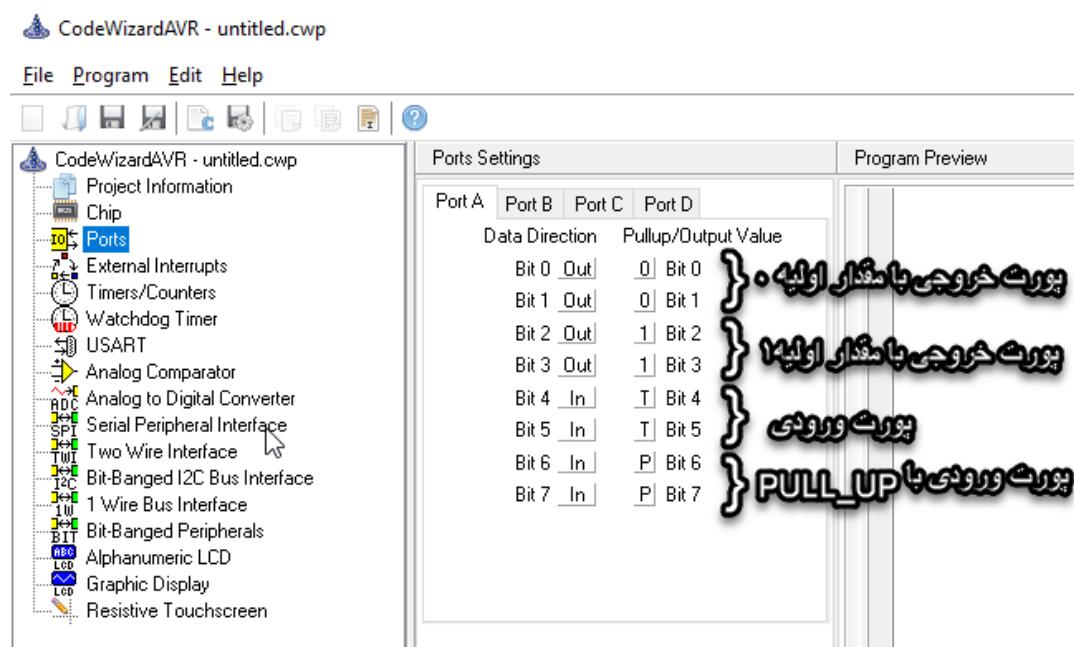


۳- پنجره Wizard در حالی که برگه Chip انتخاب شده باز می شود در این برگه شماره میکرو و فرکانس کاری آن را مشخص می کنیم.

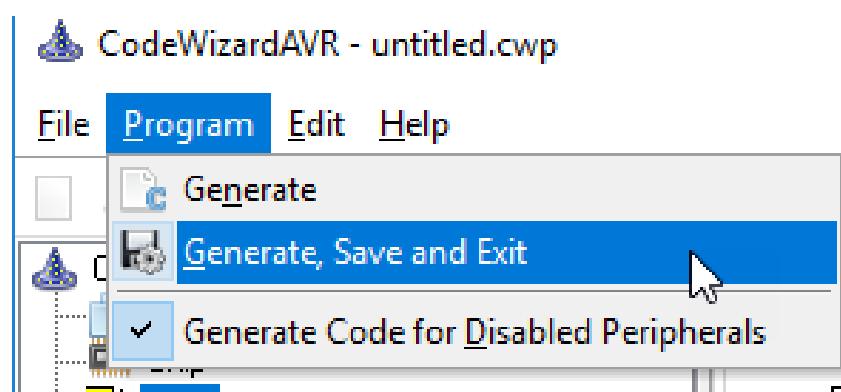


۴- با توجه به نیاز پروژه ، برگه های دیگر را باز و تنظیم های لازم را انجام می دهیم.

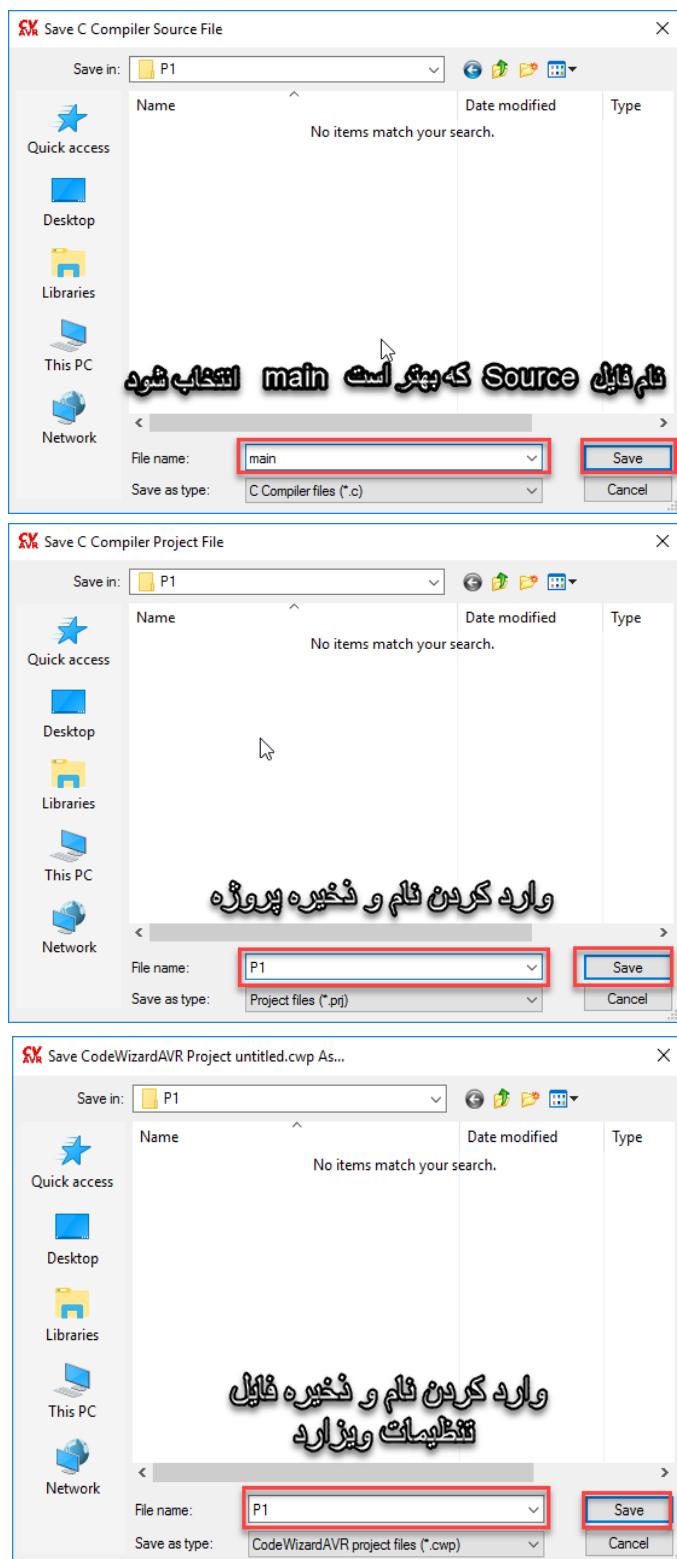
*نکته: در برگه پورت ها اگر پورتی ورودی تنظیم شود و در قسمت pull_up حرف T باشد یعنی این پایه HI_Z است و منطق لاجیکی آن بستگی به ورودی دارد . اگر حرف T را با کلیک به P تبدیل کنید یعنی این پایه از داخل میکرو pull_up شده و منطق یک دارد.



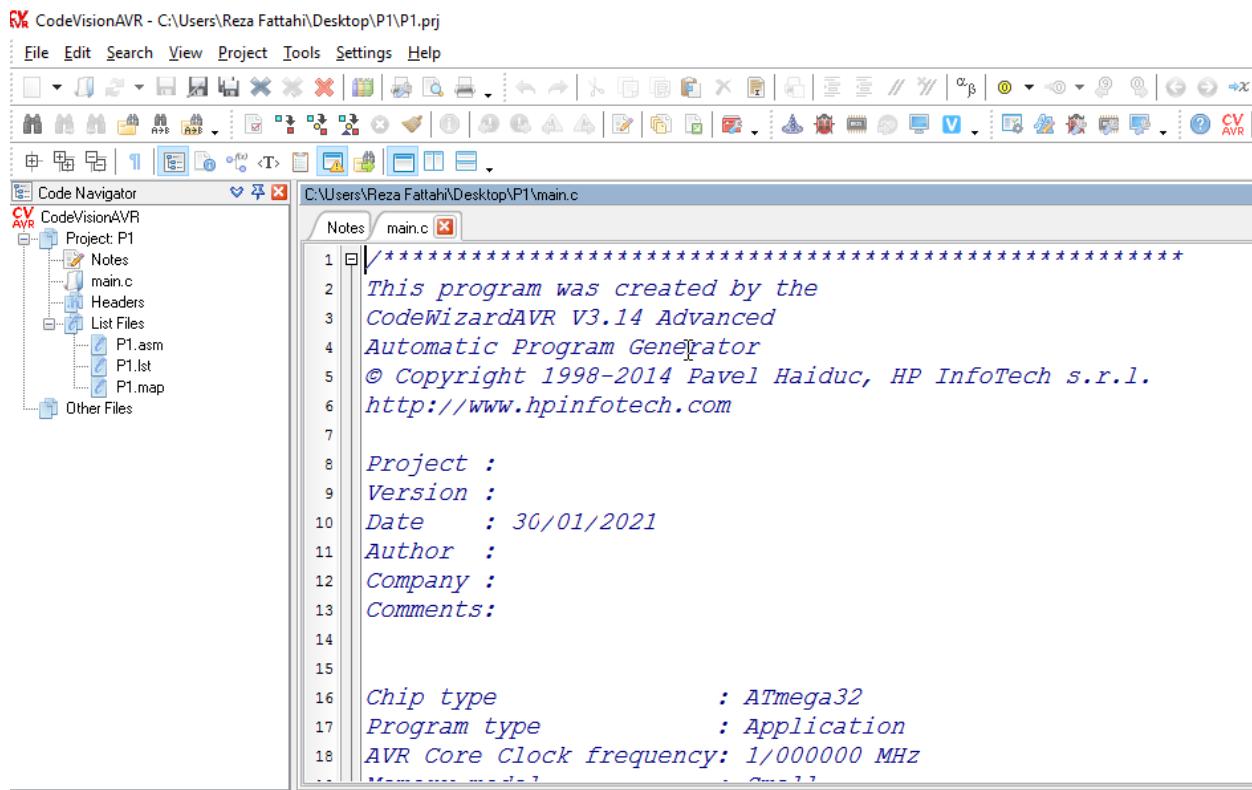
۵- پس از اتمام تنظیمات ، از مسیر زیر پروژه و فایل های مربوط به آن را با نام مناسب و در محلی مشخص ذخیره می کنیم. Program/Generate,Save and Exit



*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.



```

CodeVisionAVR - C:\Users\Reza Fattahi\Desktop\P1\P1.prj
File Edit Search View Project Tools Settings Help
Code Navigator Project: P1
CV AVR Notes main.c
This program was created by the
CodeWizardAVR V3.14 Advanced
Automatic Program Generator
© Copyright 1998-2014 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date : 30/01/2021
Author :
Company :
Comments:

Chip type : ATmega32
Program type : Application
AVR Core Clock frequency: 1/000000 MHz

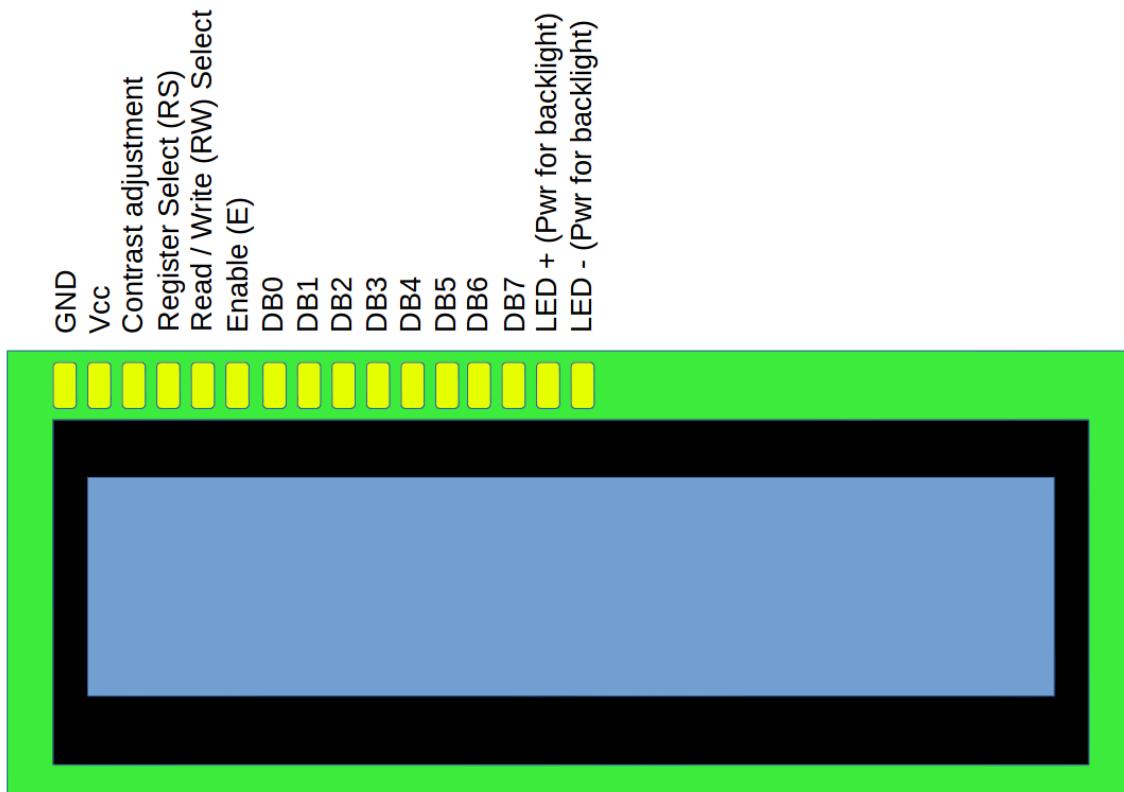
```

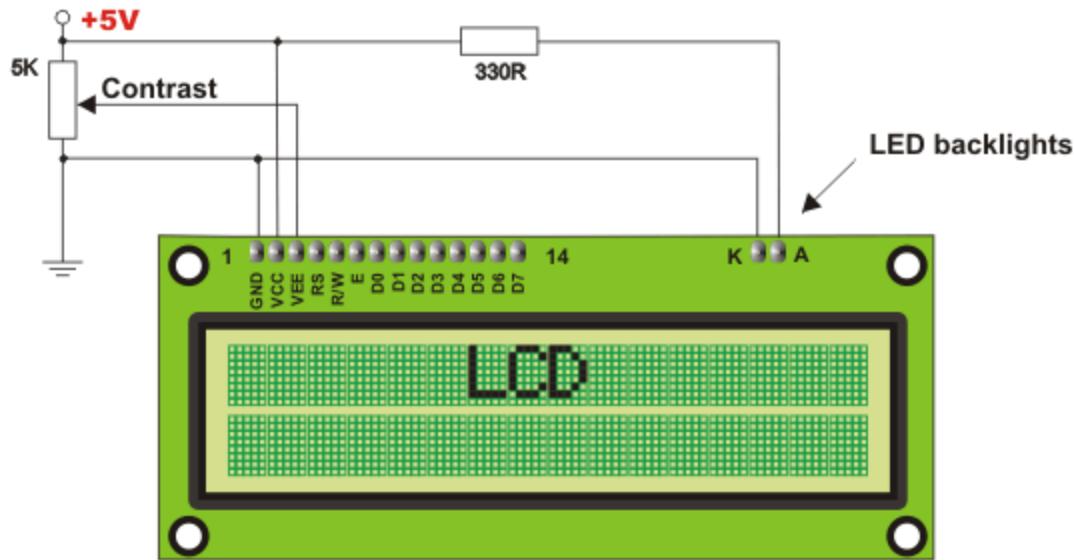
(Liquid Crystal Display)LCD

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: معمولاً ۱۶ پایه به شرح زیر دارد.





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

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

: دستور و دیتا را دریافت می کند.

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

: اگر یک باشد Read و اگر صفر باشد یعنی Write

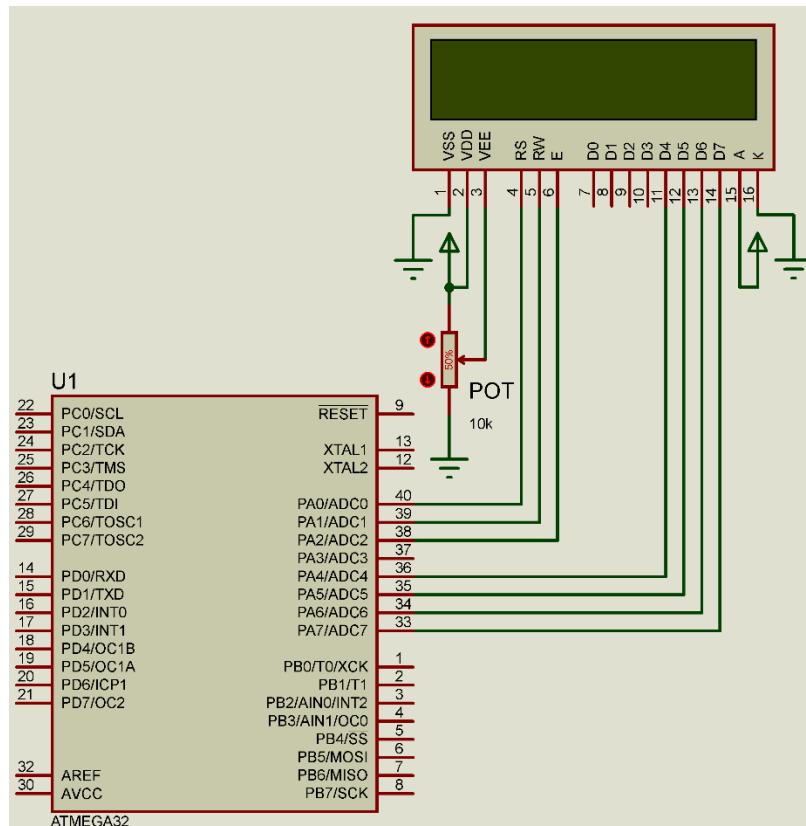
: با اعمال یک لبه ی پایین رونده می توان اجازه ورود دیتا یا دستور را صادر کرد.

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

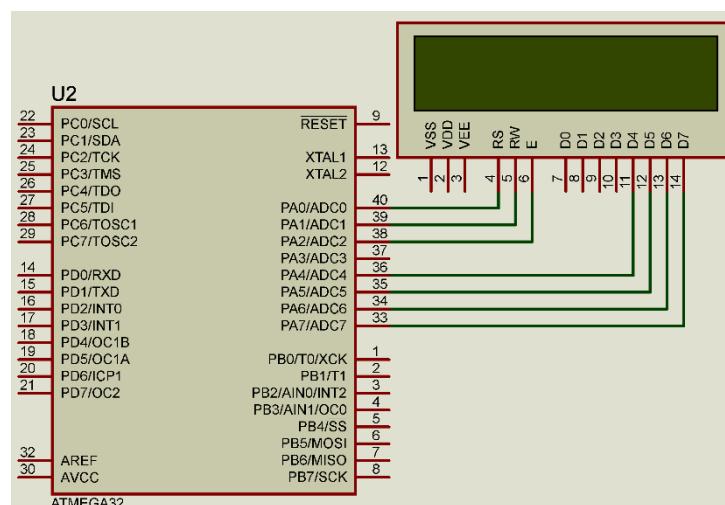
: تغذیه LCD از این پایه ها تأمین می شود.

نکته: LCD های کاراکتری را می توان با ۴ خط دیتا راه 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.

در شکل زیر نحوه اتصال کامل یک LCD کاراکتری را به PORTA میکرو مشاهده می کنید.

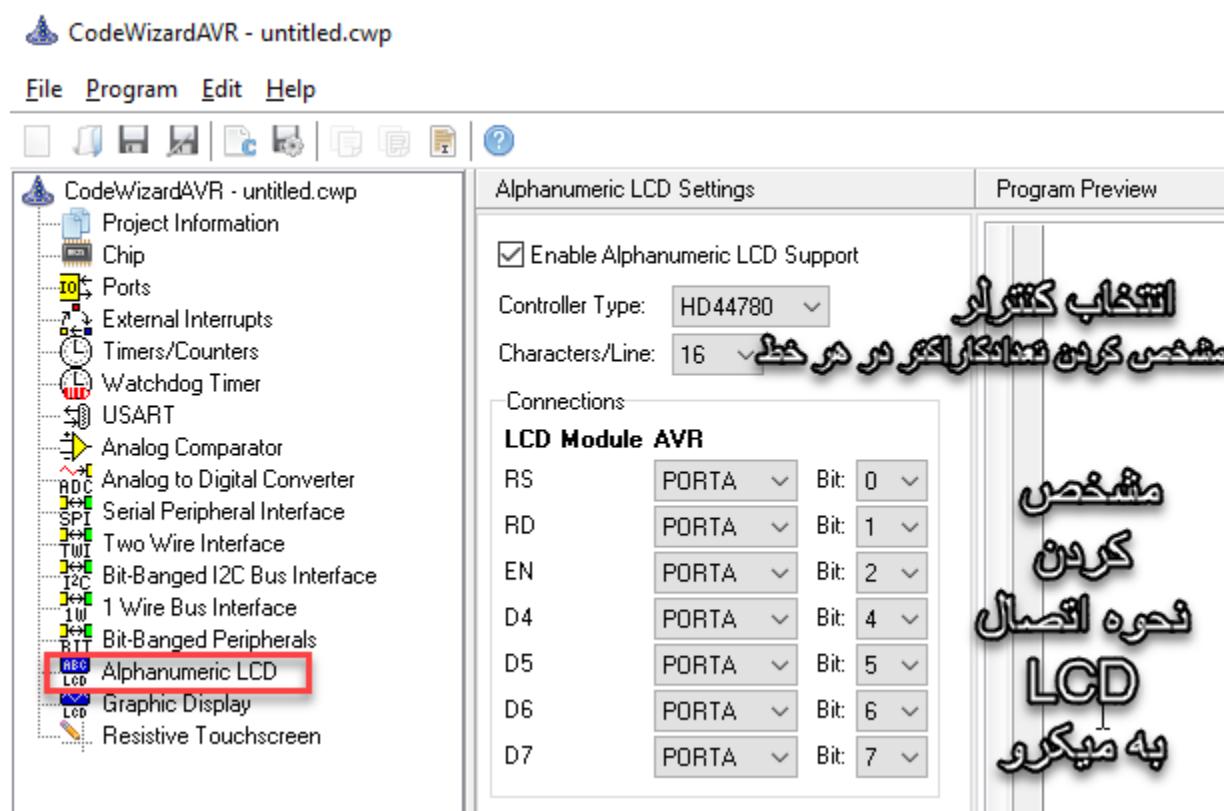


- لازم به ذکر است که در پروتئوس نیاز به اتصال تغذیه، پتانسیومتر، آند و کاتد نیست. مانند شکل زیر:



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

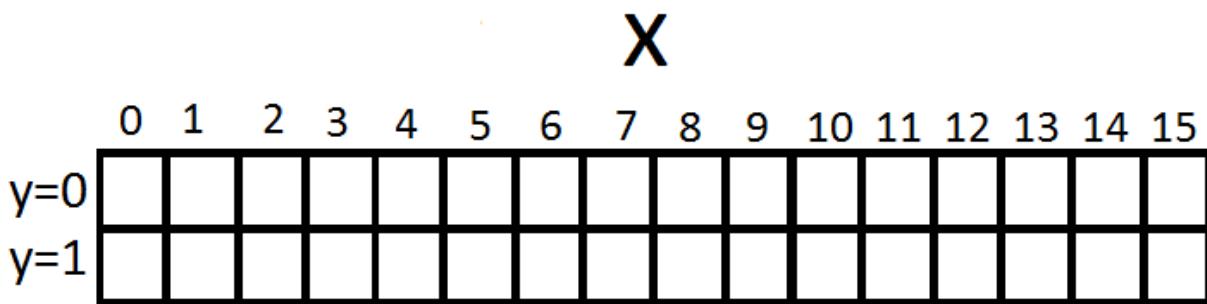
ابتدا در ویزارد برگه Controller Type Alphanumeric LCD را باز و آن را فعال می کنیم. در قسمت مشخص می کنیم که کنترلر LCD کدام است (معمولا HD44780 می باشد) در قسمت Connections مشخص می کنیم که در هر خط چند کاراکتر وجود دارد. در قسمت Character/Line نیز نحوه اتصال LCD به میکرو مشخص شده که قابل ویرایش است.



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

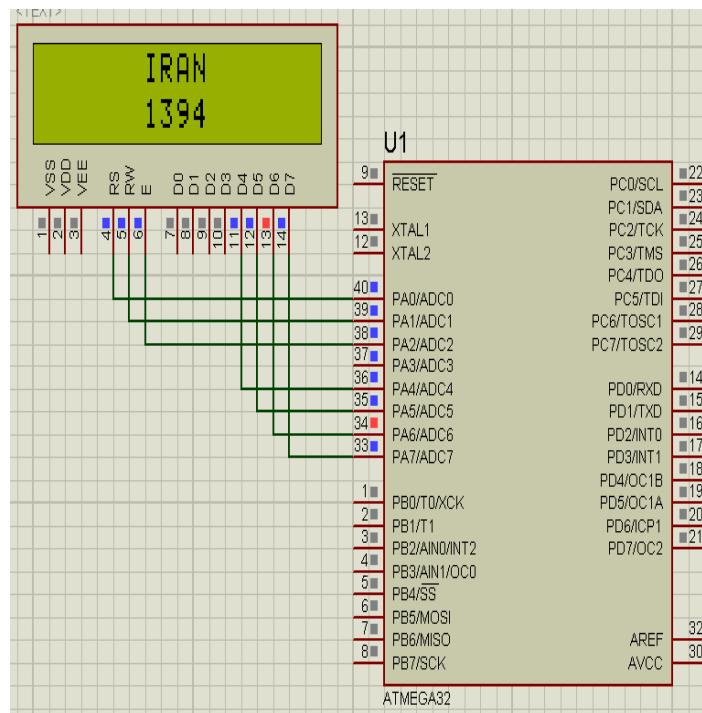


`lcd_putchar('کارکتر');` `lcd_putchar('R');`

`lcd_putsf("رشته");` `lcd_putsf("REZA");`

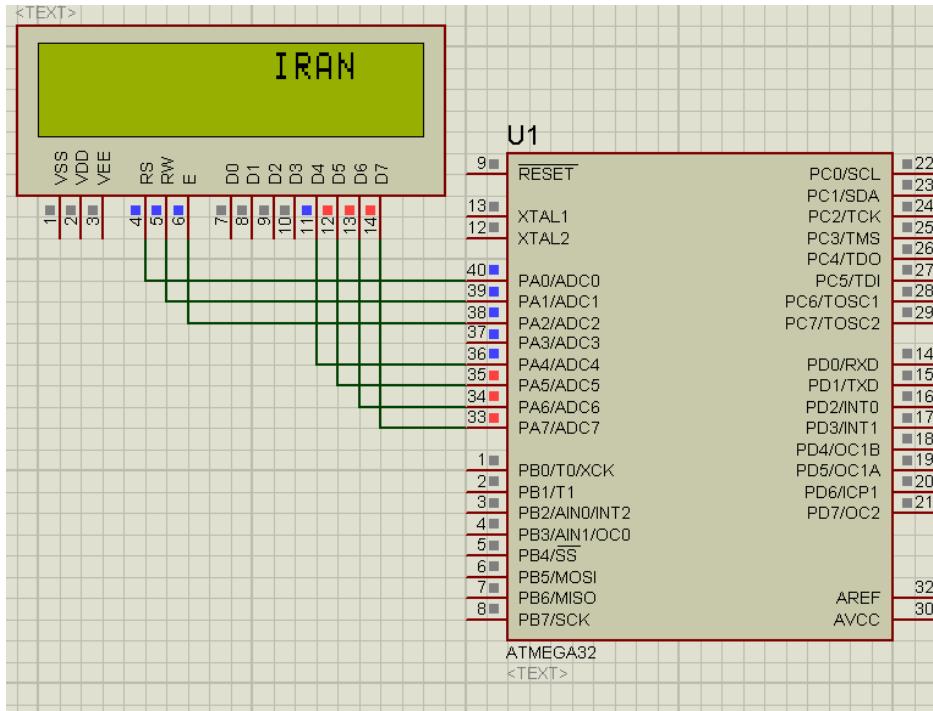
`lcd_puts(s);` متغیر رشته ای (`s`); `char s[]="FATTAHI";` `lcd_puts(s);`

```
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);
        }
    }
}
```



American Standard Code Information Interchange : (ASCII)

با توجه به اینکه کامپیوتر از بخش های مختلفی ساخته شده و لازم است که این بخش ها به یکدیگر اطلاعات ارسال و دریافت نمایند، باید کدهایی استاندارد و معروفی می شد تا تمام سازندگان کامپیوتر و دستگاه های جانبی از آن پیروی کنند. انجمنی در آمریکا این کار را برای ارسال و دریافت اطلاعات در مخابرات تلگرافی و الکترونیکی انجام داده و کد تولیدی را به نام ASCII معرفی کرده بود . 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			ASCII printable characters								Extended ASCII characters							
DEC	HEX	Simbolo ASCII	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	
00	00h	NULL (carácter nulo)	32	20h	espacio	64	40h	@	96	60h	`	128	80h	Ç	160	A0h	á	
01	01h	SOH (inicio encabezado)	33	21h	!	65	41h	A	97	61h	a	129	81h	ü	161	A1h	í	
02	02h	STX (inicio texto)	34	22h	"	66	42h	B	98	62h	b	130	82h	é	162	A2h	ó	
03	03h	ETX (fin de texto)	35	23h	#	67	43h	C	99	63h	c	131	83h	à	163	A3h	ú	
04	04h	EOT (fin transmisión)	36	24h	\$	68	44h	D	100	64h	d	132	84h	ä	164	A4h	ñ	
05	05h	ENQ (enquiry)	37	25h	%	69	45h	E	101	65h	e	133	85h	à	165	A5h	N	
06	06h	ACK (acknowledgement)	38	26h	&	70	46h	F	102	66h	f	134	86h	å	166	A6h	ä	
07	07h	BEL (timbre)	39	27h	*	71	47h	G	103	67h	g	135	87h	ç	167	A7h	º	
08	08h	BS (retroceso)	40	28h	(72	48h	H	104	68h	h	136	88h	ê	168	A8h	¸	
09	09h	HT (tab horizontal)	41	29h)	73	49h	I	105	69h	i	137	89h	ë	169	A9h	ø	
10	0Ah	LF (salto de linea)	42	2Ah	*	74	4Ah	J	106	6Ah	j	138	8Ah	è	170	AAh	¬	
11	0Bh	VT (tab vertical)	43	2Bh	+	75	4Bh	K	107	6Bh	k	139	8Bh	í	171	A Bh	½	
12	0Ch	FF (form feed)	44	2Ch	,	76	4Ch	L	108	6Ch	l	140	8Ch	í	172	A Ch	¼	
13	0Dh	CR (retorno de carro)	45	2Dh	-	77	4Dh	M	109	6Dh	m	141	8Dh	í	173	A Dh	¾	
14	0Eh	SO (shift Out)	46	2Eh	.	78	4Eh	N	110	6Eh	n	142	8Eh	À	174	A Eh	«	
15	0Fh	SI (shift In)	47	2Fh	/	79	4Fh	O	111	6Fh	o	143	8Fh	Ã	175	A Fh	»	
16	10h	DLE (data link escape)	48	30h	0	80	50h	P	112	70h	p	144	90h	È	176	B0h	»	
17	11h	DC1 (device control 1)	49	31h	1	81	51h	Q	113	71h	q	145	91h	æ	177	B1h	¶	
18	12h	DC2 (device control 2)	50	32h	2	82	52h	R	114	72h	r	146	92h	Æ	178	B2h	µ	
19	13h	DC3 (device control 3)	51	33h	3	83	53h	S	115	73h	s	147	93h	ô	179	B3h	‰	
20	14h	DC4 (device control 4)	52	34h	4	84	54h	T	116	74h	t	148	94h	ò	180	B4h	–	
21	15h	NAK (negative acknowle.)	53	35h	5	85	55h	U	117	75h	u	149	95h	ö	181	B5h	À	
22	16h	SYN (synchronous idle)	54	36h	6	86	56h	V	118	76h	v	150	96h	ü	182	B6h	Ã	
23	17h	ETB (end of trans. block)	55	37h	7	87	57h	W	119	77h	w	151	97h	ú	183	B7h	Ã	
24	18h	CAN (cancel)	56	38h	8	88	58h	X	120	78h	x	152	98h	ÿ	184	B8h	©	
25	19h	EM (end of medium)	57	39h	9	89	59h	Y	121	79h	y	153	99h	Ô	185	B9h	±	
26	1Ah	SUB (substitute)	58	3Ah	:	90	5Ah	Z	122	7Ah	z	154	9Ah	Ù	186	BAh	»	
27	1Bh	ESC (escape)	59	3Bh	:	91	5Bh	[123	7Bh	{	155	9Bh	ø	187	B Bh	»	
28	1Ch	FS (file separator)	60	3Ch	<	92	5Ch	\	124	7Ch		156	9Ch	£	188	BCh	»	
29	1Dh	GS (group separator)	61	3Dh	=	93	5Dh]	125	7Dh	}	157	9Dh	Ø	189	B Dh	¢	
30	1Eh	RS (record separator)	62	3Eh	>	94	5Eh	^	126	7Eh	~	158	9Eh	×	190	B Eh	¥	
31	1Fh	US (unit separator)	63	3Fh	?	95	5Fh	-				159	9Fh	f	191	B Fh	—	
127	20h	DEL (delete)													223	DFh	■	

برای ارسال متغیرهای عددی به LCD لازم است که ابتدا آن ها را توسط توابع موجود به کد ASCII تبدیل کرده و سپس ارسال نماییم. برای این کار توابعی وجود دارد که یکی از آن ها تابع `sprintf` موجود در کتابخانه `<stdio.h>` است. ساختار این تابع مانند شکل زیر است.

(متغیر عددی ، "متن و نوع متغیرها و نحوه چاپ" ، متغیر رشته ای)

برای تعریف متغیر رشته ای طبق جدول زیر عمل کنید.

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

برای مشخص کردن نوع و چگونگی چاپ، علاوه بر نوع متغیر، از اعداد نیز در کنار آن استفاده می شود. مثلاً از علامت `%d` به همراه کاراکترهای خاص استفاده می شود. مثلاً از علامت `%f` یا `%s` برای مشخص کردن اعداد صحیح و از `%f` برای اعداد اعشاری استفاده می شود .

برای مشخص کردن چگونگی چاپ، علاوه بر نوع متغیر، از اعداد نیز در کنار آن استفاده می شود، مثلاً `%02d` یعنی این که عدد بصورت دو رقمی چاپ شود حال اگر عدد یک رقمی باشد با قرار دادن یک ۰ قبل از عدد، آن را در دو رقم چاپ می کند (نمایش می دهد) یا اگر مقدار متغیر ۳ باشد روی LCD عدد دورقمی ۰۳ دیده خواهد شد.

همچنین برای تبدیل متغیر های عددی به متغیر رشته ای می توانید از توابع `itoa` و `ftoa` که در کتابخانه `<stdlib.h>` قرار دارند استفاده کنید.

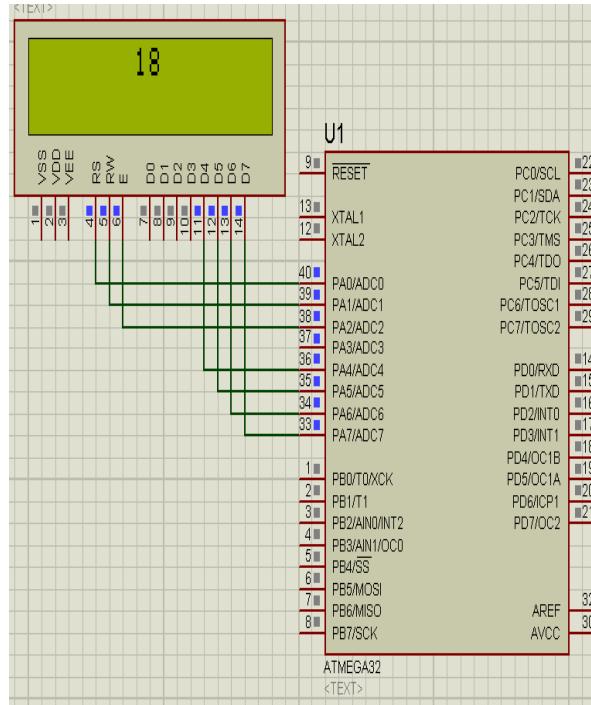
(متغیر رشته ای , متغیر عددی) `itoa`

(متغیر رشته ای , تعداد رقم اعشار , متغیر عددی) `ftoa`

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.

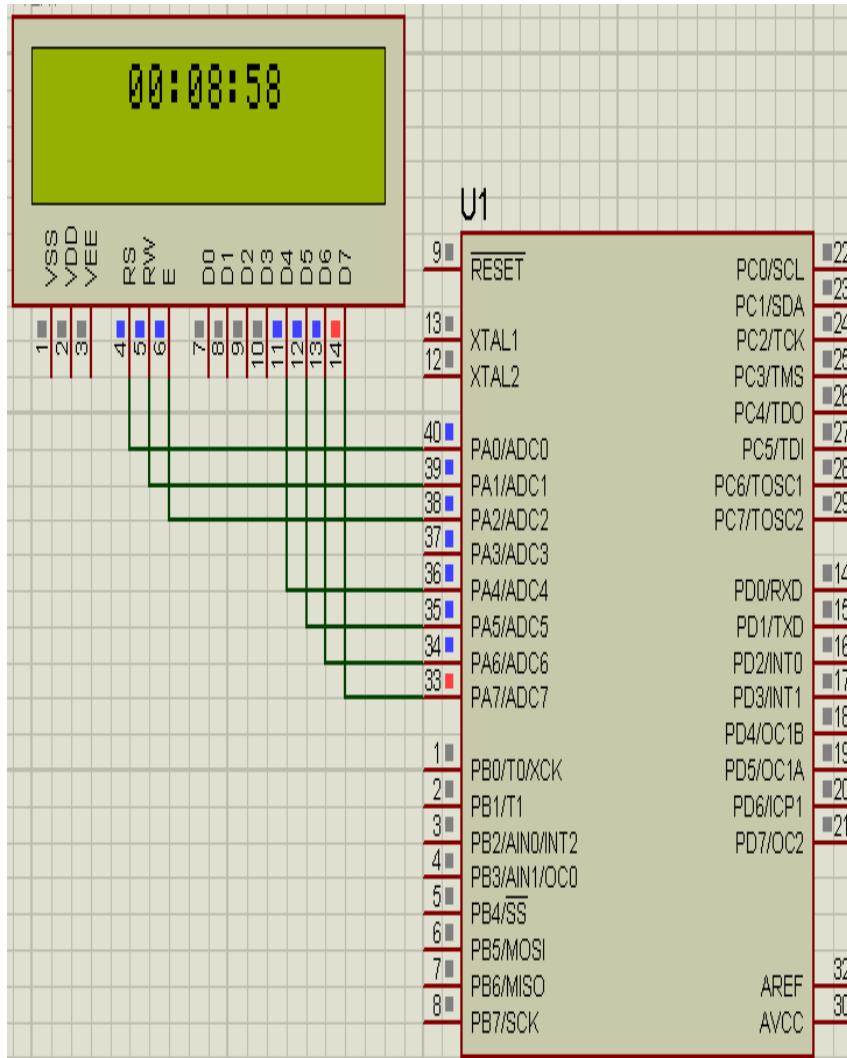
مثال: بر روی LCD یک شمارنده ۰ تا ۱۹ بسازید.

```
#include <delay.h>
#include <stdio.h>
unsigned char i;
char s[4];
void main(void)
{
while (1)
{
for(i=0;i<20;i++)
{
sprintf(s,"%02d",i);
lcd_gotoxy(7,0);
lcd_puts(s);
delay_ms(500);
}}}
```



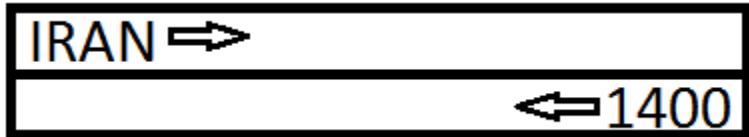
مثال: یک ساعت بر روی LCD طراحی کنید.

```
#include <stdio.h>
#include <delay.h>
unsigned char h,m,s;
char t[16];
void main(void)
{
    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,d,s);
        lcd_gotoxy(4,0);
        lcd_puts(t);
        delay_ms(1000);
    }
}
```



تمرین:

- ۱- برنامه ای بنویسید که کلمه IRAN روی خط اول و عدد 1400 روی خط دوم LCD برعکس یکدیگر



حرکت کند.

- ۲- برنامه ای بنویسید که اینیمیشن Load شدن را روی خط دوم و در صد پیشرفت را وسط خط اول LCD نمایش دهد. راهنمایی: دستور lcd_putchar(255); باعث نمایش کاراکتر █ می شود.



- ۳- برنامه ای بنویسید که یک جمله حاوی حداقل ۳۲ کاراکتری را روی خط اول LCD حرکت دهد. برای

مثال جمله می تواند نام، نام خانوادگی، تاریخ تولد، کد ملی و شماره دانشجویی شما باشد.

- ۴- بر روی خط اول LCD یک ساعت و روی خط دوم، ساعت و دقیقه آلام را نمایش دهید. به میکرو یک

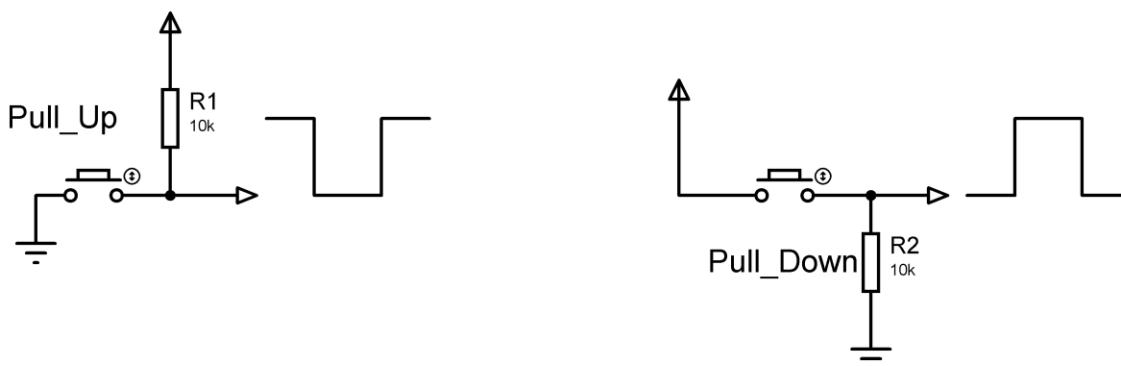
Buzzer نیز متصل است برنامه‌ای بنویسید که هر گاه زمان به آلام رسید بازر فعال شود و تا یک دقیقه

فعال بماند و سپس قطع شود.

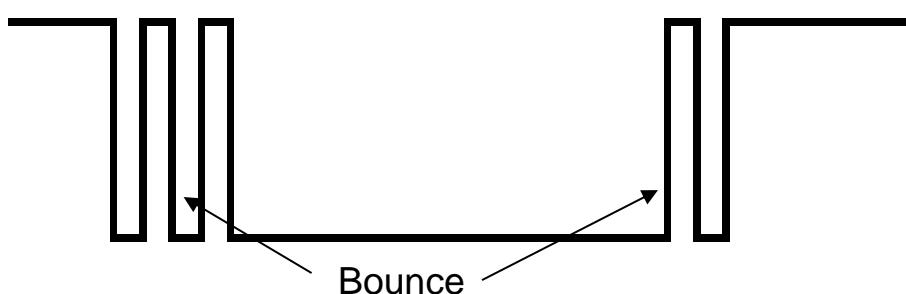
کلید (key) : یکی از ساده‌ترین و پرکاربردترین دستگاه‌های ورودی کلید است. توسط کلید می‌توان صفر یا یک را تولید و به مدار اعمال کرد. کلید‌های مد نظر ما در حالت عادی باز هستند و فشار بر روی آن‌ها باعث وصل شدن و اتصال کنکات‌ها می‌شود. به این کلیدها Tact Switch گفته می‌شود.

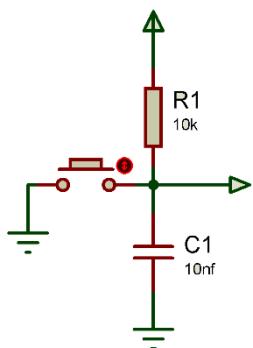


معمولاً دو مدار زیر برای کلیدها بسته می‌شوند.



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





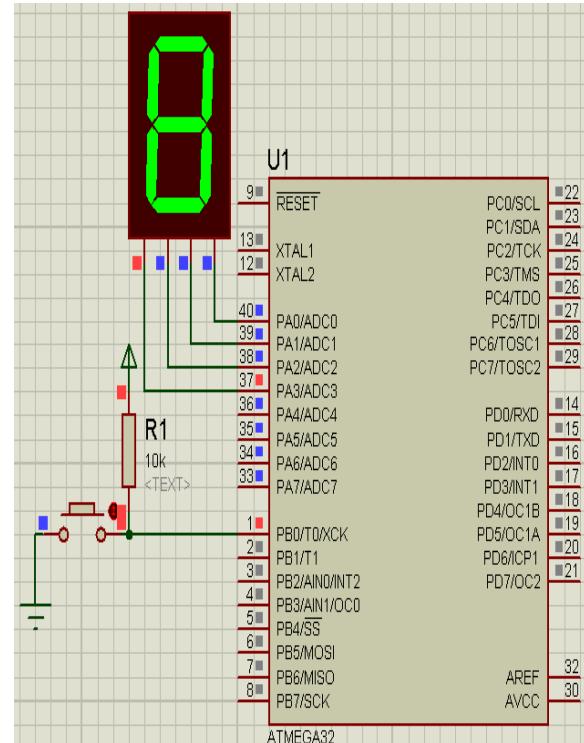
برای رفع این مشکل پس از هر تغییر در خروجی کلید ، حدود ۲۰ تا ۳۰ میلی ثانیه تاخیر ایجاد می کنیم تا لرزش ها تمام شود. همچنین می توان یک خازن 10nf را به مانندشکل مقابله به مدار کلید اضافه کرد. با توجه به فرکانس پالس های Bounce این پالس های اضافی از طریق خازن فیلتر و به زمین هدایت شده ، وارد میکرو نمی شوند.

مثال: یک سون سگمنت BCD و یک کلید را به میکرو متصل می کنیم. برنامه ای بنویسید که با هر بار 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 با یک کلید عدد خروجی را افزایش و با کلید دوم کاهش دهید.

برنامه را طوری بنویسید که عدد خروجی از ۹ بیشتر و از صفر کمتر نشود.

```
if(PINB.0==0 && i<9)
{
    delay_ms(25);

    i++;

    PORTA=i;

    while(PINB.0==0);
    delay_ms(25);

}

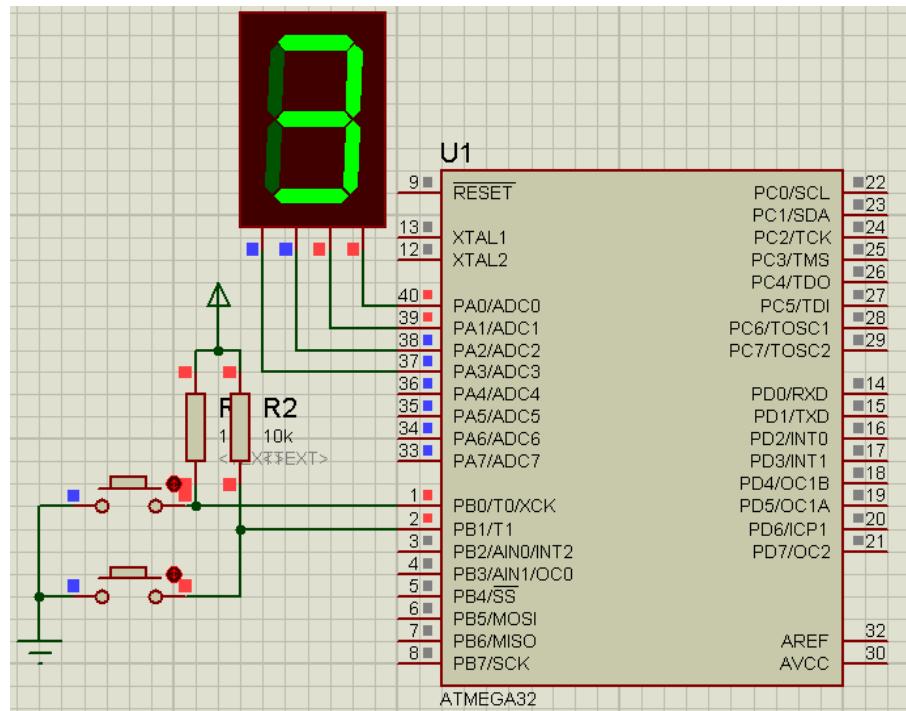
if(PINB.1==0 && i>0)
{
    delay_ms(25);

    i--;

    PORTA=i;

    while(PINB.1==0);
    delay_ms(25);

}
```



تمرین:

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

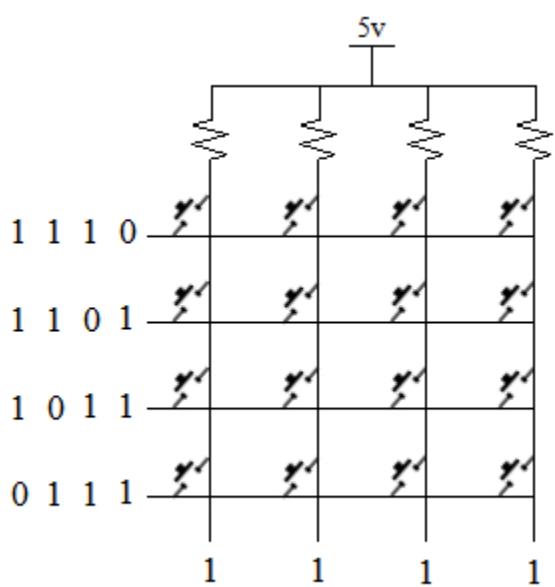
۲- دو عدد کلید به PD.2 و یک Buzzer به PD.3 و یک Buzzer به PD.7 متصل است برنامه ای بنویسید که با زدن کلید PD.2، Buzzer روشن و بوق بزند و با کلید PD.3 قطع شود.

۳- 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.

۴- 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.

۵- یک lcd به PORTA و سه عدد کلید به PD.2 و PD.3 و PD.0 و یک Buzzer به PD.7 متصل کنید برنامه ای بنویسید که روی خط اول lcd ساعت و روی خط دوم lcd زمان آلام 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 آلام با ساعت برابر شد بطور مناسب بوق بزند.

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



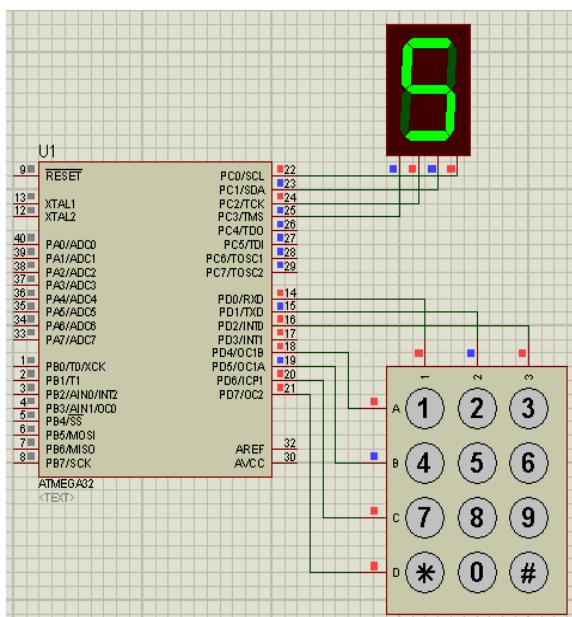
برای خواندن صفحه کلید به شکل زیر عمل شود:

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 مجدداً ستون هارا می خوانیم. این کار را برای همه ی سطرهای انجام می دهیم. اگر سطحی را صفر کردیم و ستونی صفر شد با توجه به سطر و ستونی که صفر شده، کلید فعل 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);}
```

PORTE.7=1;

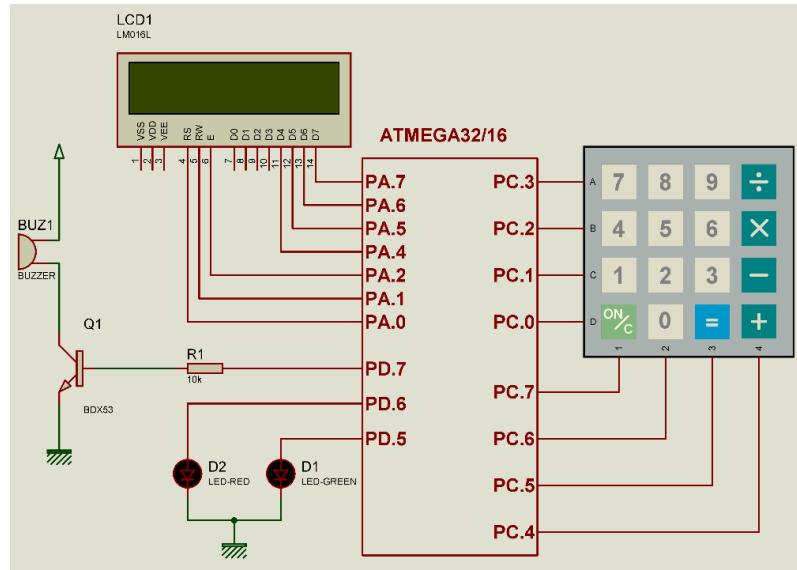
if(k!=12) PORTC=k;

}

تمرین:

- ۱- 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

نیز LCD پاک شود.



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

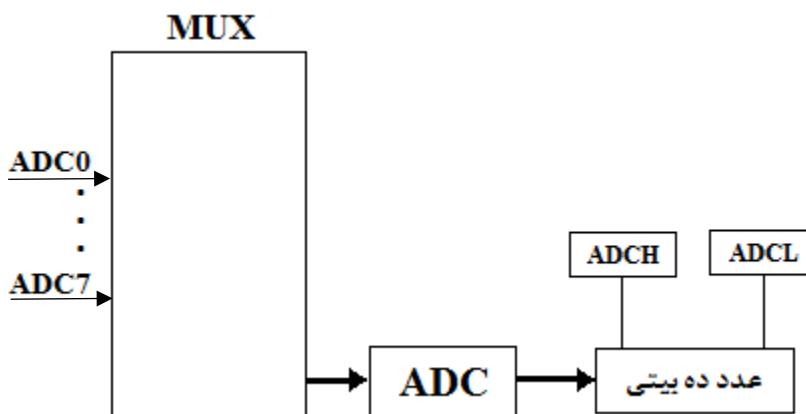
۳- یک ماشین حساب بسازید. الف) یک رقمی ب) چند رقمی

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

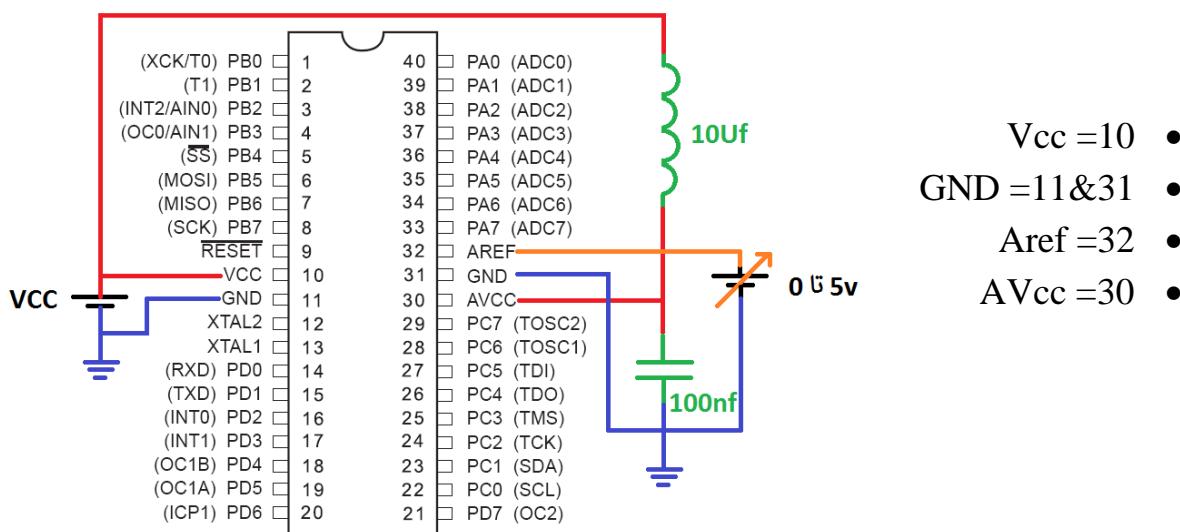
- ۵- تمرین قبل را طوری کامل کنید که مدیر بتواند رمز دستگاه را نیز عوض کند . در ضمن هنگامی که کاربران رمز را وارد می کنند اعداد زده شده بصورت * روی lcd دیده شوند. همچنین اگر کاربر رمز ورودی را سه بار اشتباه وارد کرد ، کیبورد قفل ، و هر دو led روشن شوند.

Mبدل آنالوگ به دیجیتال (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.



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

۱. ولتاژ تغذیه: یعنی AVcc مرجع باشد. (5v)

۲. ولتاژ منبع خارجی: یعنی 5v. Aref (0 تا 5v)

۳. ولتاژ منبع داخلی: یعنی 2.56v (یک خازن در حد $1\mu F$ را روی پایه Aref قرار دهید)

ضریب تفکیک Step Size: پارامتری است که مشخص می‌کند حساسیت یا دقت ADC چقدر است و از رابطه زیر محاسبه می‌شود.

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

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

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

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

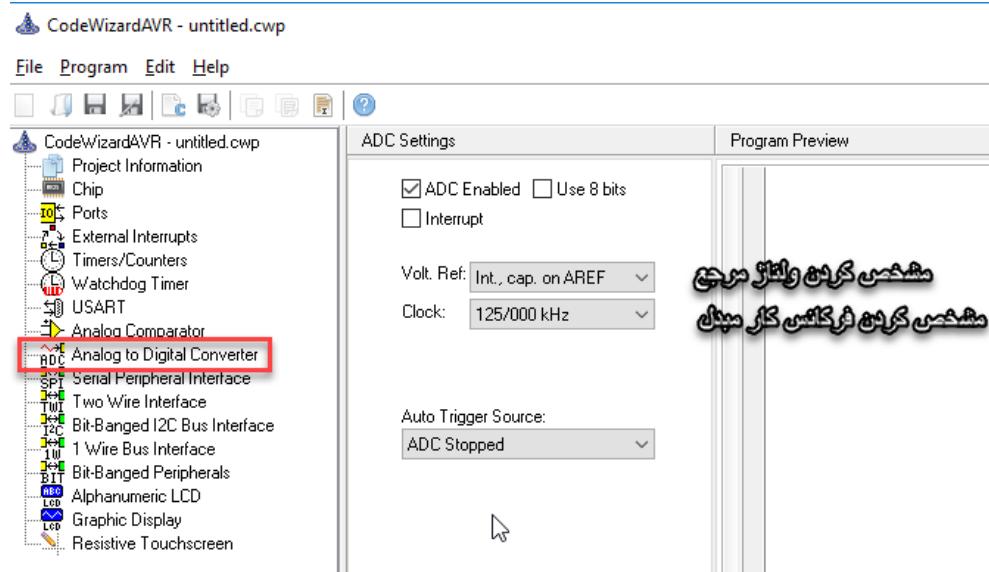
$$\frac{2.56}{2^{10} - 1} = 2.5mv$$

$$\frac{1}{2.5mv} = 400$$

۱. ضریب تفکیک

۲. عدد خروجی مبدل

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



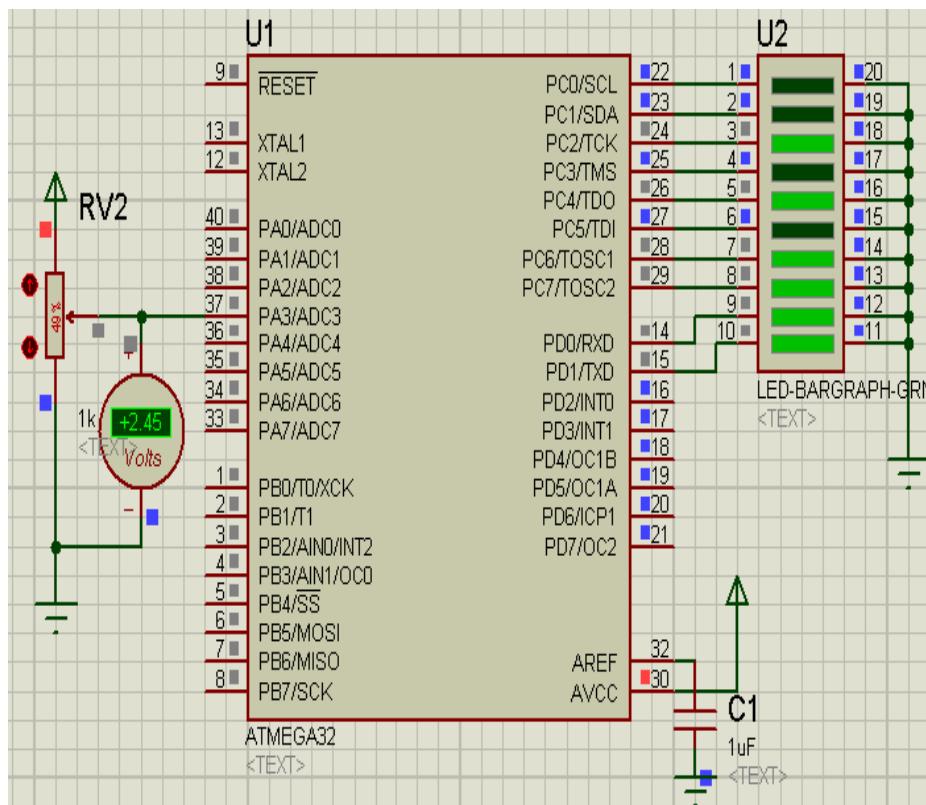
نکته: وقتی از مرجع داخلی استفاده می‌شود، باید یک خازن که طرف دیگر آن به زمین متصل است، در حد Aref بر روی پایه 1uF قرار گیرد.

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 ADC Stopped را نیز روی Trigger قرار دهید.

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

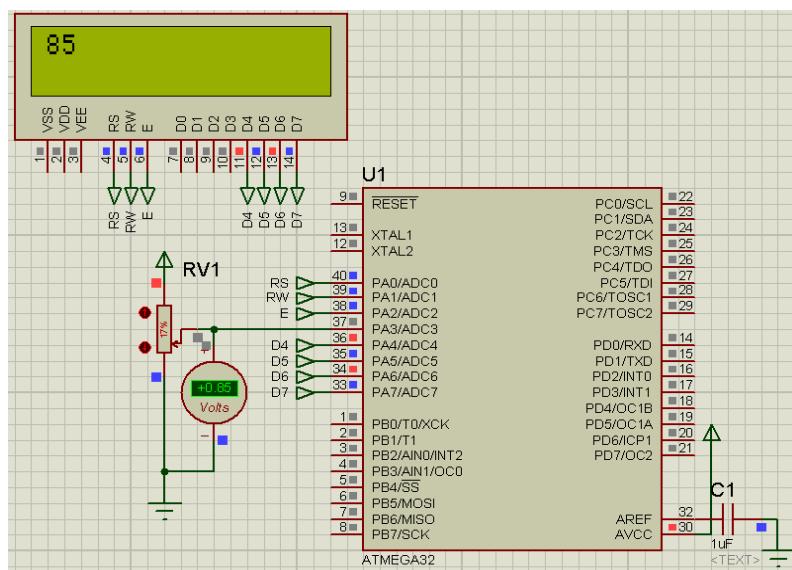
پس از تولید کد ملاحظه می شود تابع `read_adc` به برنامه اضافه شده است. این تابع مشخص می کند که از کدام کanal مقدار آنالوگ خوانده شود و خروجی آن عدد خروجی مبدل است.

```
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 <stdio.h>
#include <delay.h>
unsigned int a;
char s[6];
void main(void)
{
while (1)
{
a=read_adc(3);
sprint(s,"%04d",a);
lcd_gotoxy(0,0);
lcd_puts(s);
delay_ms(200);
}}
```



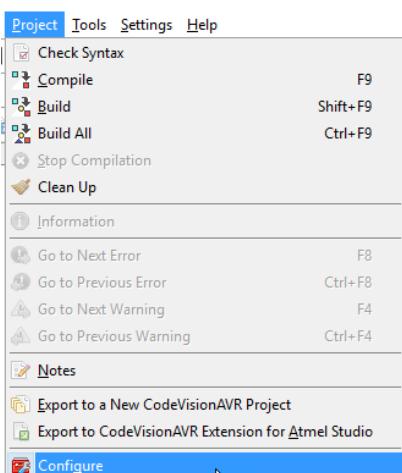
مثال: یک LCD به پورت A متصل کرده و با استفاده از کanal ۳ یک ولتمتر بسازید که مقدار ولتاژ را بر روی آن نمایش دهد.

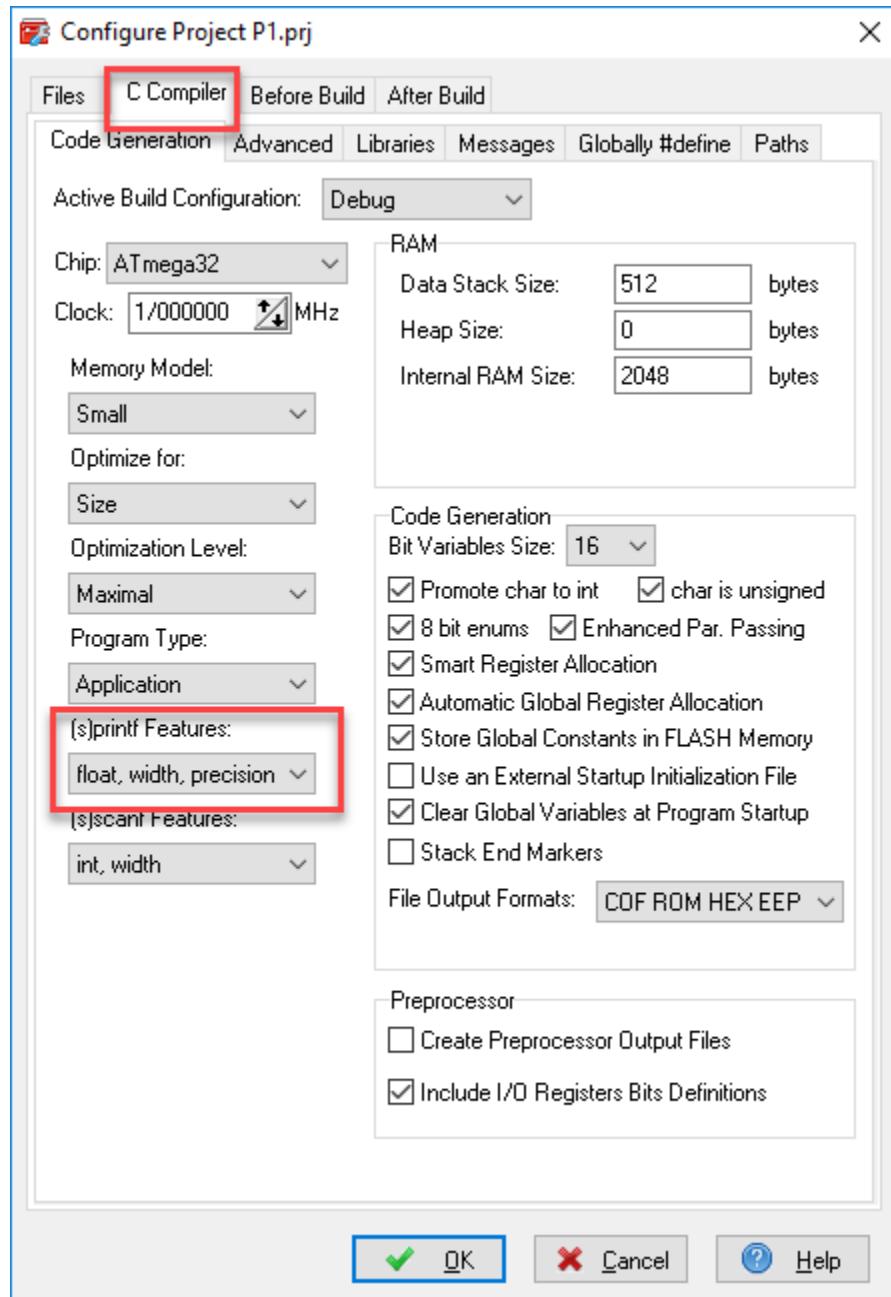
LCD نمایش دهد.

نکته: با توجه به این که در این برنامه به نمایش عدد اعشاری نیاز داریم لازم است از مسیر زیر در کدویزن تبدیل

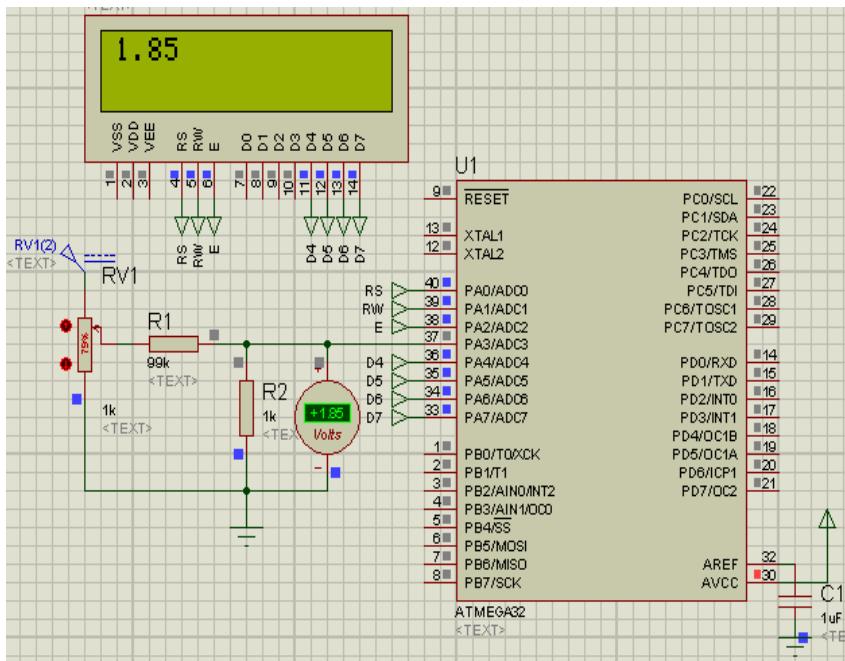
اعداد اعشاری به رشته را فعال کنیم.

Project/Configure/C Compiler/(s)print Features/float,width,precision

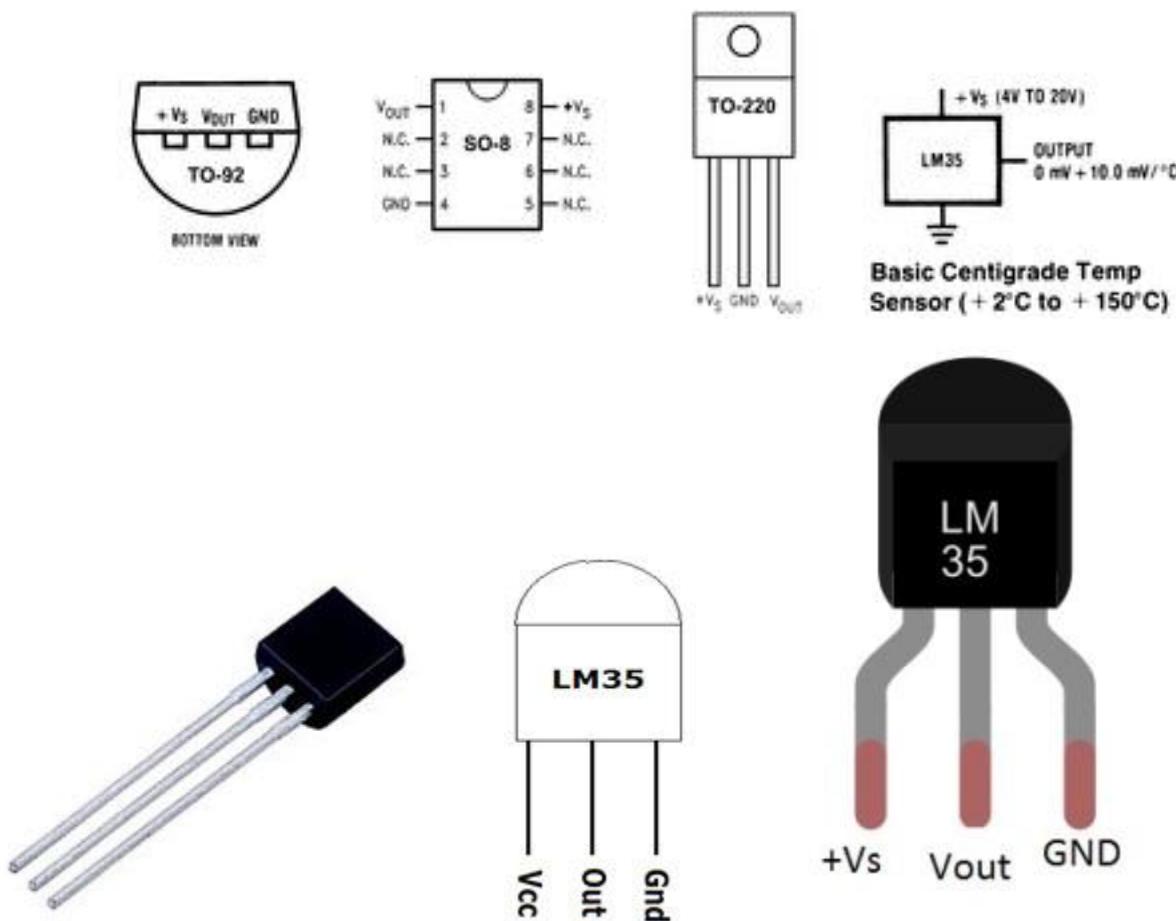




```
#include <stdio.h>
unsigned int a;
float v;
char s[8];
void main(void)
{
while (1)
{
a=read_adc(3);
v=(float)a/400;
sprintf(s,"%4.2f",v);
lcd_gotoxy(0,0);
lcd_puts(s);
delay_ms(100);
}
}
```



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 15 درجه سانتیگراد باشد، خروجی سنسور 150mV خواهد بود.



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

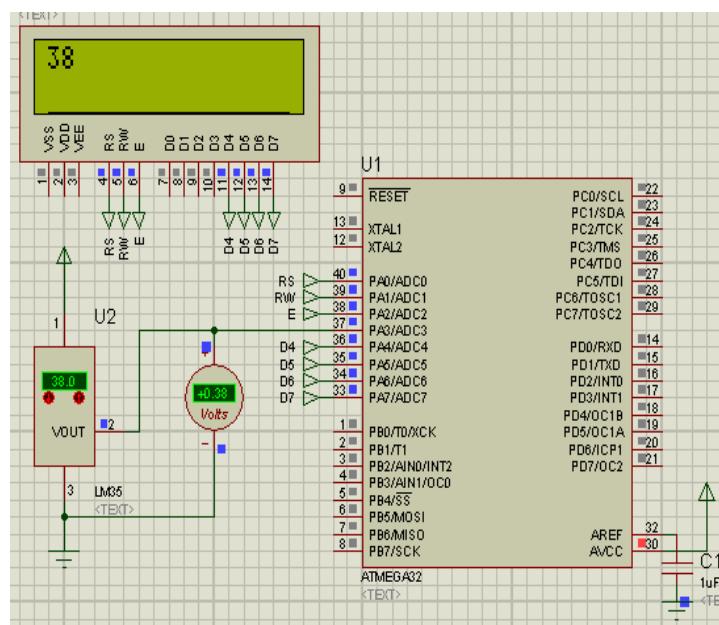
پاسخ: فرض می کنیم دما یک درجه سانتیگراد باشد با توجه به ضریب تفکیک 2.5mV عدد خروجی مبدل 4 خواهد شد.

$$\text{عدد خروجی} = \frac{\text{خروجی}}{\text{ضریب}} = \frac{10\text{mV}}{2.5\text{mV}} = 4$$

با توجه به محاسبه بالا هر عددی که خوانده شد را برابر 4 تقسیم می کنیم تا دما بدست آید.

برای این برنامه کافی است در مثال بالا (ولتومتر) ، بجای 400 عدد خروجی مبدل را به 4 تقسیم کنیم.

$$v = (\text{float}) a / 4;$$



تمرین:

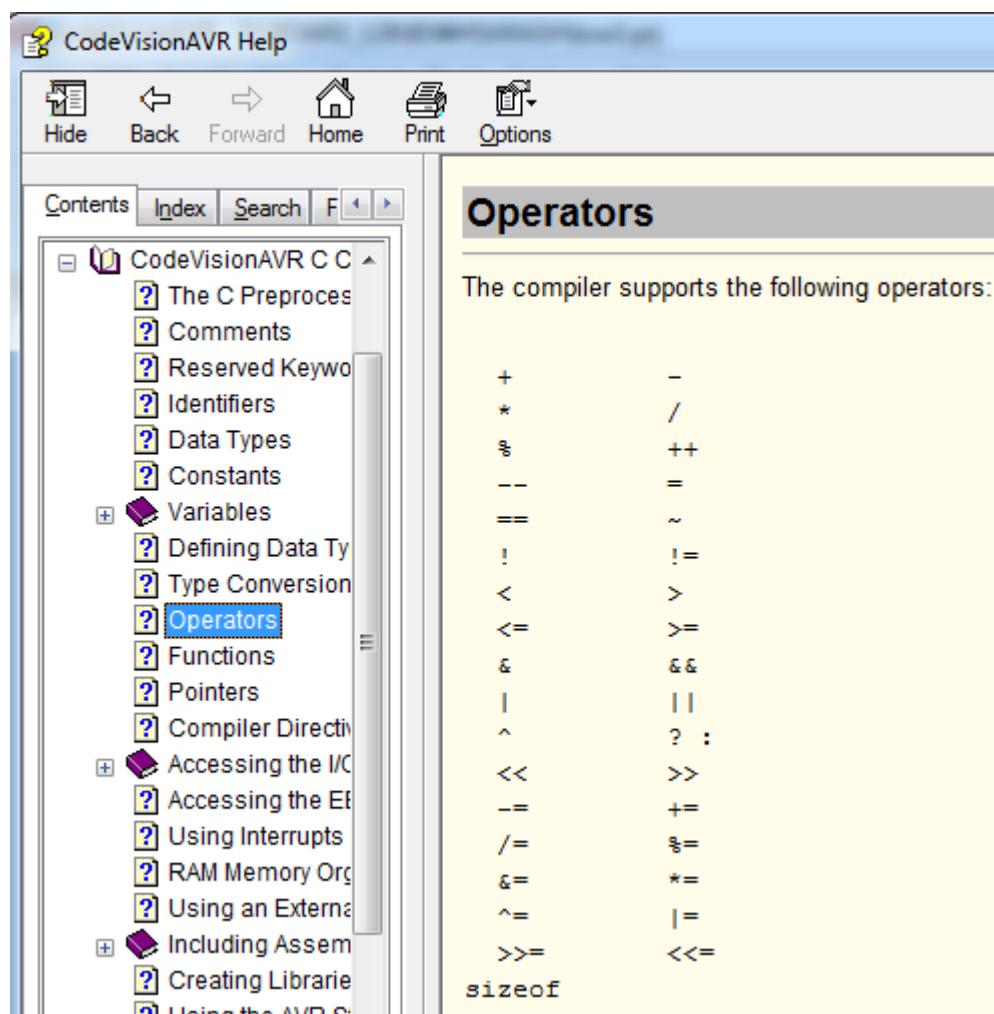
۱- یک سنسور دما،یک LCD و یک فن به میکرو متصل کنید برنامه ای بنویسید که اگر دما از ۳۰ درجه بیشتر شد فن روشن و اگر دما از ۲۵ درجه کمتر شد فن خاموش شود.در ضمن همواره دما و روشن یا خاموش بودن فن بطور مناسب روی LCD نمایش داده شود.

۲- دو سنسور دما،یک LCD و یک فن به میکرو متصل کنید برنامه ای بنویسید که اگر میانگین دمای دو سنسور از ۳۰ درجه بیشتر شد فن روشن و اگر میانگین دما از ۲۵ درجه کمتر شد فن خاموش شود. در ضمن همواره دمای هر دو سنسور و میانگین دما و روشن یا خاموش بودن فن بطور مناسب روی LCD نمایش داده شود.

۳- یک سنسور دما،یک LCD ، یک فن و ۳ عدد کلید ، به میکرو متصل کنید برنامه ای بنویسید که روی LCD دو دمای T_{\max} و T_{\min} نمایش داده شود و کاربر بتواند با یکی از کلید ها مشخص کند کدام آیتم را می خواهد تغییر دهد و با دو کلید دیگر دمای مورد نظر را مشخص کند . اگر دما از T_{\max} بیشتر شد فن روشن و اگر از T_{\min} کمتر شد فن خاموش شود.در ضمن همواره دما و روشن یا خاموش بودن فن بطور مناسب روی LCD نمایش داده شود.

OPERATORS

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



عملگرهای جمع، تفریق و ضرب نکته خاصی ندارد.

عملگرهای تقسیم معمولی و باقی مانده در مثال زیر تفاوت و نتیجه هر یک را مشاهده می‌کنید.

$$13 / 5 = 2$$

$$13 \% 5 = 3$$

عملگرهای افزایش و کاهش به مقدار یک واحد.

$$a=5 \quad a++; \quad \longrightarrow \quad a=6$$

$$a=5 \quad a--; \quad \longrightarrow \quad a=4$$

عملگرهای 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

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

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

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.

: عملگرهای بیتی و منطقی : $\wedge, \vee, \neg, \&\&, \&, \sim, !$

عملگر	بیتی	منطقی
NOT	\sim	!
AND	$\&$	$\&\&$
OR	$ $	\parallel
XOR	\wedge	ندارد

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، مقادیر زیر را به دست آورید.

$PORTC=a \& b;$ $PORTC=a | b;$ $PORTC=a \wedge b;$ $PORTC=\sim a;$

$a=0x56$ 01010110

$b=0x9C$ 10011100

$a \& b$ 00010100  $PORTC=00010100=0x14$

$a=0x56$ 01010110

$b=0x9C$ 10011100

$a | b$ 11011110  $PORTC=11011110=0xDE$

a=0x56 01010110
 b=0x9C 10011100
 a^b 11001010  PORTC=00010100=0xCA

a=0x56 01010110
 ~a 10101001  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.

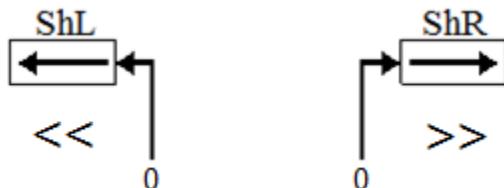
```
if ((a>6)&&(b<10))  

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  PORTC=00011000=0x18

:<, > 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 $\text{PORTC}=99$ اجرا می شود.

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

؟ : عملگر شرطی:

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.

$(a>b) ? (\text{PORTC}=99) : (\text{PORTC}=55)$

عملگرهای تخصیص مرکب:

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 \rightarrow $x=1$

`int b; x=sizeof (b)` نتیجه \rightarrow $x=2$

`float c; x=sizeof (c)` نتیجه \rightarrow $x=4$

تمرین:

۱- برنامه بنویسید که دو عدد را از PORTA و PORTB دریافت کند، و روی PORTC (الف) AND و (ب) OR (ج) XOR ورودی ها د) NOT ورودی A را نمایش دهد.

۲- برنامه بنویسید که دو عدد را از PORTA و PORTB دریافت کند، و روی PORTC مطابق با جدول زیر عمل کند.

شرط	PORTC
A>B	1
A<B	2
A>B و B>10	3
A>B یا B>10	4
A>10 و A<20	5
A=10 یا A=20	6
(A>10 و A<20) و (B>30 و B<40)	7
(A>10 و A<20) و (B>30 یا B<40)	8
(A>B+5) یا (B=A-3)	9

۳- برنامه بنویسید که عددی را از PORTA دریافت کند، اگر فرد بود روی PORTC عدد ۱ و اگر عدد زوج بود عدد ۲ را نمایش دهد.

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

PORTA	PORTB	PORTC	PORTD
10110110	1	8	2

۵- 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

۶- 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

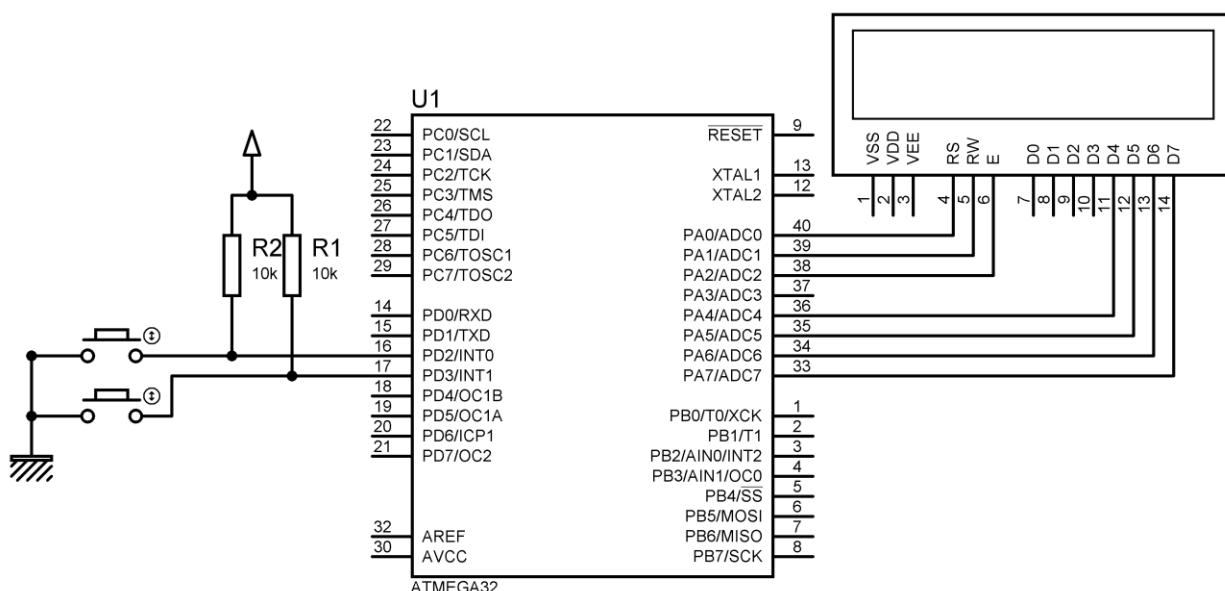
وقفه : Interrupt

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

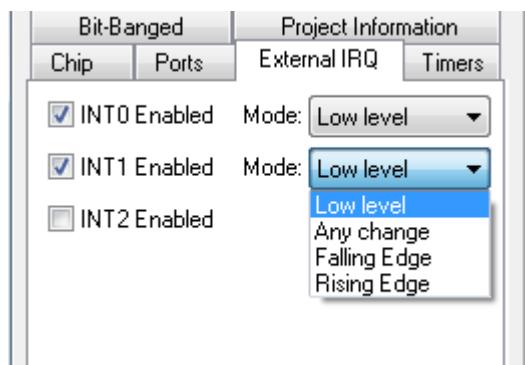
در میکرو مورد بحث ما سه وقفه خارجی INT0, INT1, INT2 وجود دارد. با یک مثال روش تنظیم، راه اندازی و استفاده از آن ها را فرامی گیریم.

مثال: یک LCD به PORTA به LCD 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 INT1 کلمه lcd 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 External IRQ برگه wizard 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  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  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();
}

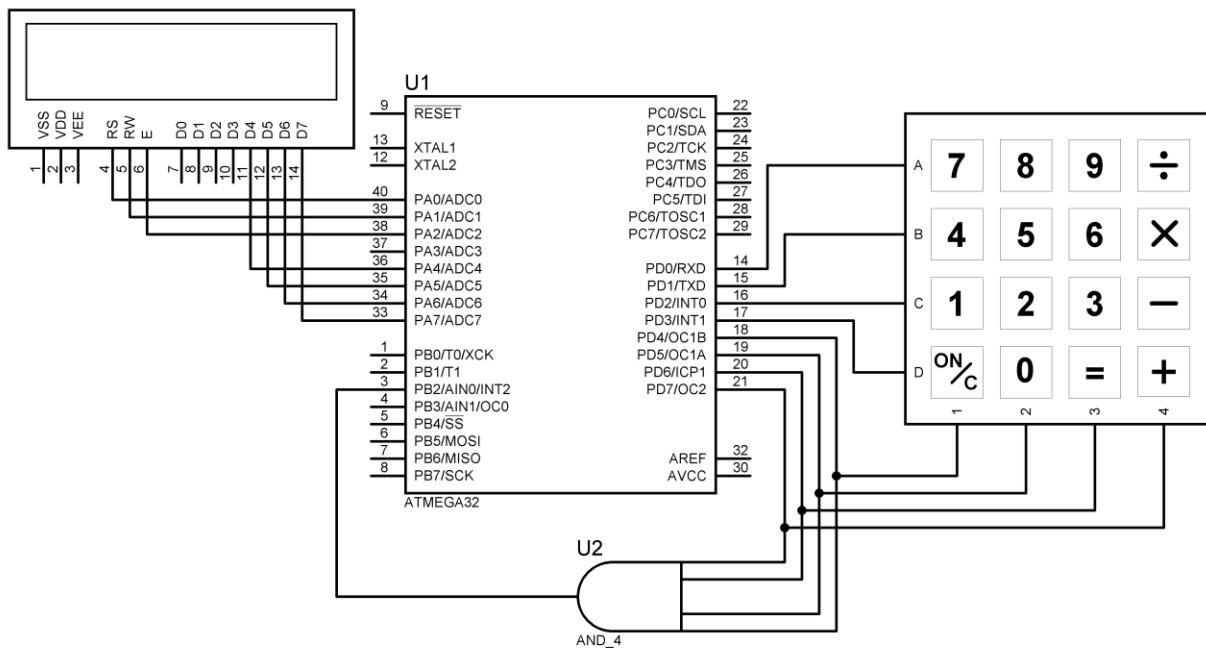
//Global enable interrupts

#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 ۰ تا ۹ 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).

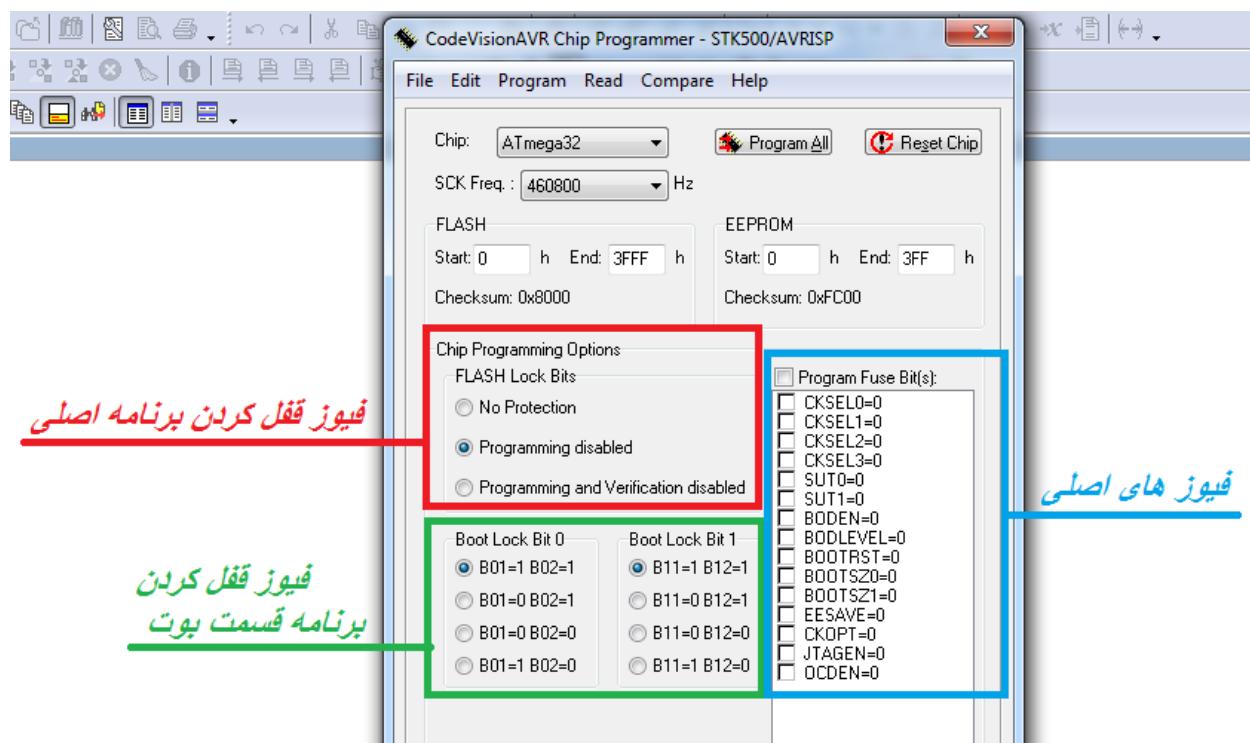


: Fuse bit

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

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

فیوز های ATMEGA32 عبارتند از:



فیوز قفل کردن برنامه اصلی

**فیوز قفل کردن
برنامه قسمت بوت**

فیوز های اصلی

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

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

کاربرد این فیوز بیت ها به شرح زیر است:

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

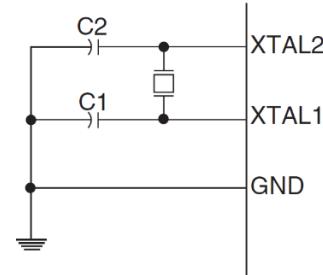
CLKSEL3_2_1_0	منبع کلک پالس
1111-1010	کریستال خارجی / تشدیدگر سرامیکی
1001	کریستال فرکانس پایین خارجی
1000-0101	نوسان ساز RC خارجی
0100-0001	نوسان ساز RC کالیبره شده داخلی
0000	نوسان ساز خارجی

تنظیم فیوزها برای نوسان ساز RC داخلی :

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

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 (Start-up Time) SUT0,1

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 و عملا 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 (Oscillator options) CKOPT

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 نویز پذیری کمتر شود اما دقت کنید که با فعال کردن این بیت توان مصرفی بیشتر خواهد شد.

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

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

$0 = 4 \text{ volt}$

$1 = 2.7 \text{ volt}$

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

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

(XCK/T0) PB0	1	40	PA0 (ADC0)
(T1) PB1	2	39	PA1 (ADC1)
(INT2/AIN0) PB2	3	38	PA2 (ADC2)
(OC0/AIN1) PB3	4	37	PA3 (ADC3)
(SS) PB4	5	36	PA4 (ADC4)
(MOSI) PB5	6	35	PA5 (ADC5)
(MISO) PB6	7	34	PA6 (ADC6)
(SCK) PB7	8	33	PA7 (ADC7)
RESET	9	32	AREF
VCC	10	31	GND
GND	11	30	AVCC
XTAL2	12	29	PC7 (TOSC2)
XTAL1	13	28	PC6 (TOSC1)
(RXD) PD0	14	27	PC5 (TDI)
(TXD) PD1	15	26	PC4 (TDO)
(INT0) PD2	16	25	PC3 (TMS)
(INT1) PD3	17	24	PC2 (TCK)
(OC1B) PD4	18	23	PC1 (SDA)
(OC1A) PD5	19	22	PC0 (SCL)
(ICP1) PD6	20	21	PD7 (OC2)

(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

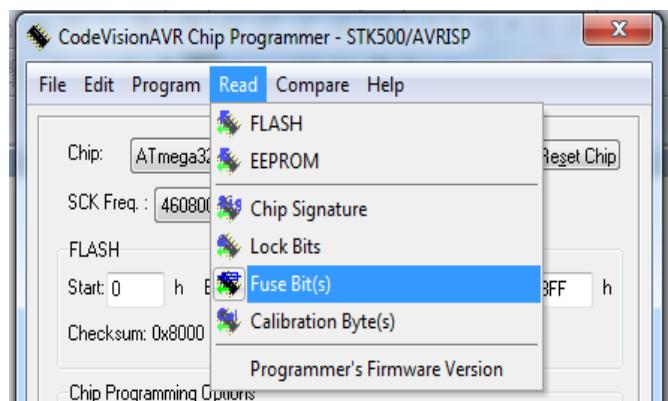
اگر این فیوز بیت فعال باشد وقتی میکروکنترلر ریست شود به آدرس بوت لودر پرش خواهد کرد. می توان بخشی از حافظه Boot Flash را به Flash اختصاص داد . بوت این امکان را فراهم می کند تا بتوان بدون استفاده از پروگرامر بخشی از حافظه Flash را برنامه ریزی کرد .

(Boot Size): BOOTSZ0,1

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

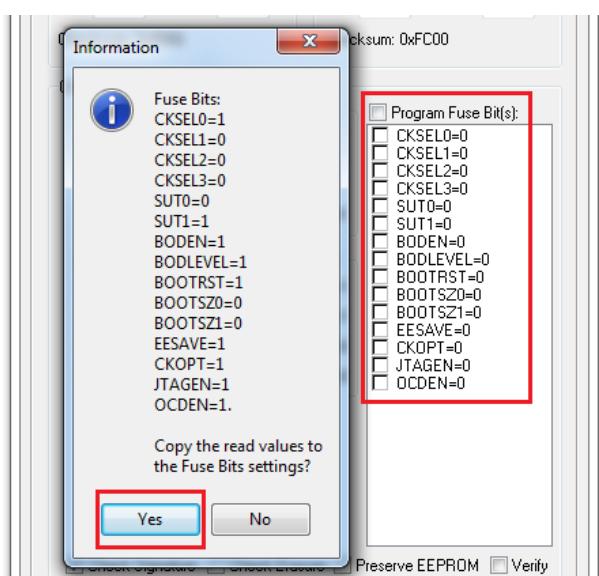
بعد از این مرحله فیوزها خوانده شده ، و در یک

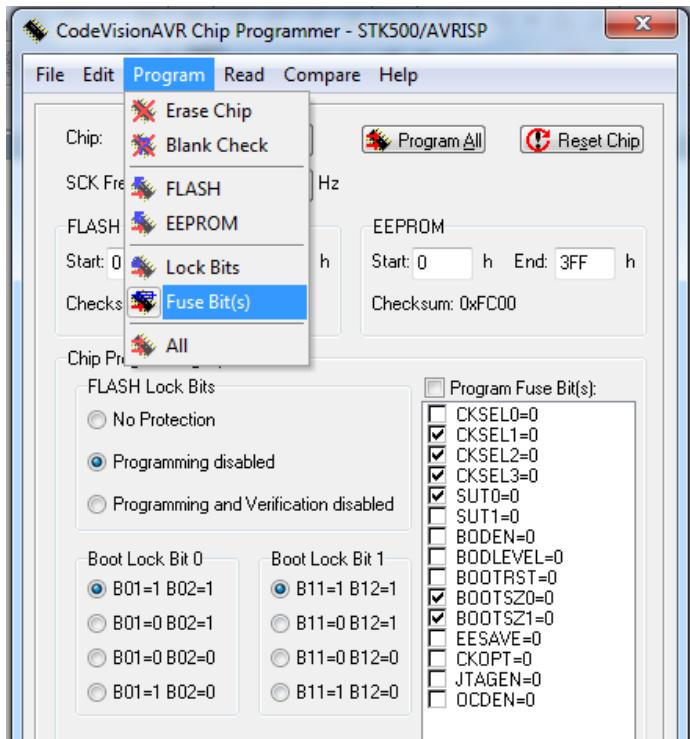
پنجره نمایش داده می شود، اگر گزینه yes را

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.

**توجه داشته باشید که ابتدا با مراجعه به Data sheet 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. نوشتمن 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:

void f1 (int x,y,z) نادرست

void f1(int x,int y,int z) درست

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

return ; نام متغیر return ; مقدار ثابت

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

(ب)

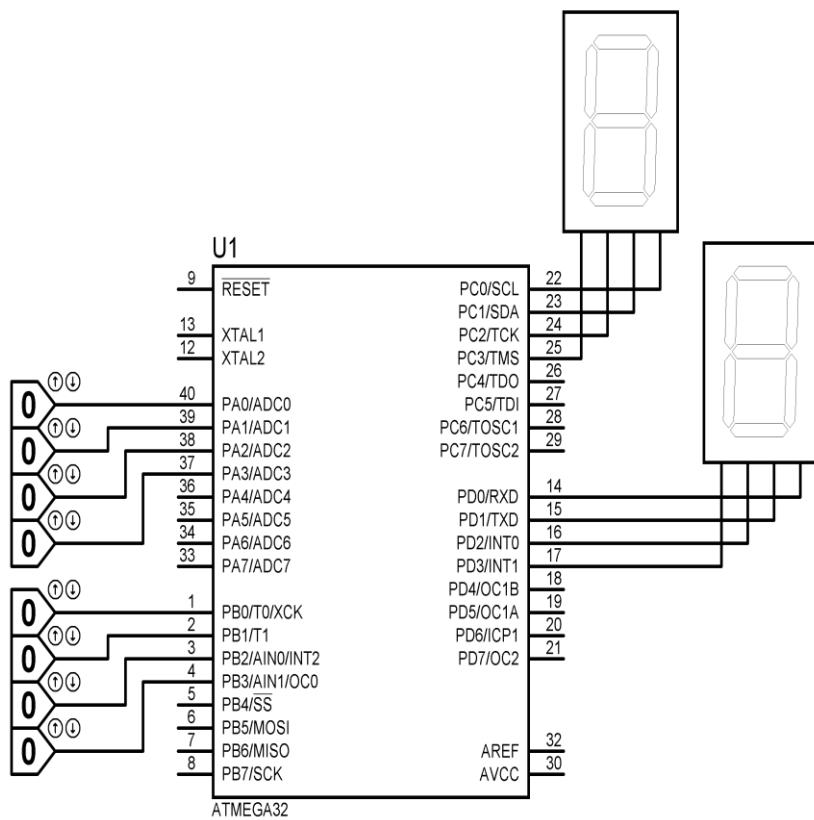
(الف)

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

اگر تابع 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 PORTB و PORTA وارد کند سپس حاصل جمع آن 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
}
```

تمرین:

۱- برنامه ای بنویسید که دو عدد چهار بیتی را از PORTA و PORTB وارد کند سپس مطابق جدول زیر روی PORTC اعدادی را نمایش دهد. این برنامه را یک بار بدون استفاده از تابع و یک بار با استفاده از تابع بنویسید.

شرط	PORTC
A>B	1
A<B	2
A>B و B>10	3
A>B یا B>10	4
A>10 و A<20	5
A=10 یا A=20	6
(A>10 و A<20) یا (B>30 و B<40)	7
(A>10 و A<20) یا (B>30 یا B<40)	8
(A>B+5) یا (B=A-3)	9

۲- برنامه ای بنویسید که عددی را از PORTA وارد کند و مساوی با آن عدد، LED های روی PORTC را خاموش و روشن کند. این برنامه را یک بار بدون استفاده از تابع و یک بار با استفاده از تابع بنویسید.

۳- برنامه ای بنویسید که یک عدد چهار بیتی را از PORTA وارد کند و دو برابر آن را روی PORTC و سه برابر آن را روی PORTD نمایش دهد. این برنامه را یک بار بدون استفاده از تابع و یک بار با استفاده از تابع بنویسید.

۴- هشت عدد LED به PORTA متصل کنید و یک چشمک زن بسازید . برای تاخیر از تابع delay موجود در کد کدویژن استفاده نکنید و یک تابع تاخیر بنویسید. این برنامه را یک بار بدون استفاده از تابع و یک بار با استفاده از تابع بنویسید.

asherه گر : POINTER

هرگاه لازم باشد به جای متغیر از آدرس آن متغیر استفاده کنیم می توان توسط اشاره گر با آدرس آن حافظه کار کرد. کاربرد اشاره گر در تخصیص حافظه پویا، دسترسی آسان تر و کار کردن 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 از طریق 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

تعريف 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) $\text{PORTC} = a + b;$

در سه خط اول ، آدرس حافظه هایی که متغیر های a, b, c در آن جا ذخیره شده اند درون اشاره گر های p, q, r

$$p=60 \quad q=62 \quad r=64 \quad \text{قرار می گیرند یعنی:}$$

در خط چهارم محتوای محل هایی که p و q به آن ها اشاره می کنند با 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.

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	سايه	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;
sprint(f.name,"ali");
sprint(f.family,"barani");
f.age=23;
f. average=15.75;
```

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

`DDRA.5=1;`

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

تمرین : برنامه‌ای بنویسید که دو عدد چهار بیتی را از `PORTB` و `PORTA` وارد کند سپس حاصل جمع آنها را روی `PORTC` و تفریق را روی `PORTD` نمایش دهد. این برنامه را با استفاده از متغیرهایی که بصورت `struct` تعریف شده‌اند بنویسید.

ویرایش نسخه ۹ زمستان ۱۳۹۹

