

## پودمان ۲

### برنامه‌نویسی به زبان C



در کتاب مونتاژ و دمونتاز SMD با میکرو کنترلر AVR و پروژه‌های ساده الکترونیکی آشنا شدید. همچنین مثال‌ها، تمرین‌ها و برنامه‌های آماده را که از طریق نرم‌افزار کد ویژن نوشته شده بود، در میکروکنترلر بارگذاری و آزمایش کردید. در این پودمان با چگونگی شکل‌گیری و نوشتن برنامه‌ها به زبان C آشنا می‌شوید تا بتوانید توانایی نوشتن برنامه‌های پیچیده‌تر را کسب کنید. در فرایند آموزش این پودمان، علائم الگوریتم، رسم فلوچارت و تبدیل آن به برنامه را فرا خواهید گرفت. یکی دیگر از اهداف آموزشی، آشنایی با قطعات و تجهیزات ورودی و خروجی است که به وسیله آن می‌توانید داده‌های دیجیتال و آنالوگ را به میکروکنترلر وارد کنید یا از آن دریافت نمایید. فرایند پردازش داده‌ها و اطلاعات توسط میکروکنترلر از مواردی است که هر متخصص الکترونیک باید با آن آشنایی داشته باشد که در ادامه آموزش به این موارد نیز می‌پردازیم.

## واحد یادگیری ۳

### کسب شایستگی در طراحی الگوریتم (فلوچارت) مدار پروژه ساده الکترونیکی

#### آیا تا به حال فکر کرده‌اید:

- با چه زبان‌ها و کامپایلرهایی می‌توان اقدام به برنامه‌نویسی میکروکنترلرها نمود؟
- اولین زبان برنامه‌نویسی اختراع شده چه بوده است؟
- فلوچارت یا روند نما چیست و چه کاربردی دارد؟
- در فلوچارت هر یک از علائم چه معنایی دارد و با چه نرم‌افزاری رسم می‌شود؟
- از توابع و عملگرها در زبان C چگونه استفاده می‌شود؟
- به کارگیری توابع و عملگرها چگونه باعث افزایش سرعت برنامه‌نویسی و تفهیم بهتر آن می‌شود؟
- اتصال و راه‌اندازی صفحه کلید به میکروکنترلر چگونه صورت می‌گیرد؟
- راه‌اندازی سنسورهایی مانند سنسور دما و شدت نور به کمک واحد ADC در میکروکنترلرها چگونه انجام می‌شود؟

از زمان پیدایش و توسعه زبان برنامه‌نویسی C تا به امروز، این زبان همواره در حال توسعه بوده و در سیستم‌های نرم‌افزاری و سخت‌افزاری به کار می‌رود. با توجه به قدیمی بودن این زبان هیچ‌گاه دستور زبان به کار رفته در آن منسوخ نشده است. امروزه زبان‌های بسیار زیادی ایجاد شده‌اند که شباهت به C دارند و به شکل گسترده‌ای توسعه یافته‌اند. نفوذ زبان C در صنعت به قدری است که توسعه‌دهندگان نرم‌افزارهای میکروکنترلر، در حد گسترده از آن استفاده می‌کنند. امروزه زبان‌های برنامه‌نویسی متنوعی برای میکروکنترلرهایی مانند AVR، PIC و سری ARM با استفاده از کامپایلرهای مبتنی بر زبان C مانند Codevision، PICC، keil و IAR طراحی شده‌اند که بخش عظیمی از صنایع دنیا را پوشش می‌دهند. در این واحد یادگیری ابتدا علائم و چگونگی ترسیم الگوریتم یا روند برای حل یک مسئله را مطرح می‌کنیم. سپس با مفاهیم ساختار کلی برنامه‌نویسی به زبان C آشنا خواهیم شد. همچنین به کارگیری برنامه C در برنامه‌نویسی میکروکنترلرهای AVR و توسعه سخت‌افزار مبتنی بر آن را مورد بحث قرار خواهیم داد.

#### استاندارد عملگرد

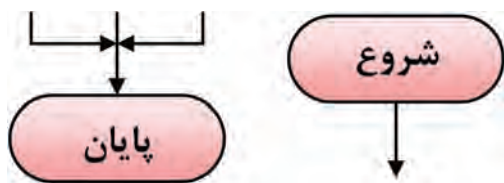
ترسیم فلوچارت برنامه‌های مختلف و پیاده‌سازی آن روی میکروکنترلر

## ۱-۳- طراحی الگوریتم (روند) برنامه

عناصر اصلی برای حل یک مسئله در کامپیوتر، ورودی‌ها و خروجی‌ها هستند. برای مثال، برای جمع دو عدد  $a$  و  $b$  ورودی‌ها  $a$  و  $b$  هستند. این برنامه یک خروجی دارد که حاصل جمع این دو عدد است. بنابراین نوشتن این برنامه بسیار ساده است، اما برای نوشتن برخی از برنامه‌ها به تفکر بیشتری نیاز داریم. در چنین مرحله‌ای لازم است ابتدا گام‌های برنامه مورد نظر را به دست آورید تا بتوانید برنامه مورد نظر را پیاده‌سازی کنید. غالباً اجرای این گام‌ها چندین ساعت طول می‌کشد. همچنین نیاز به محاسبات ریاضی فراوانی دارد. برای رسیدن به این هدف، ترسیم فلوجارت برای مسئله مورد نظر، امری کاربردی است.

به مجموعه‌ای از تصاویر و نمادها که الگوریتم (روند) برنامه را به صورت تصویری و نموداری نشان می‌دهد فلوجارت یا روندنما می‌گویند. رسم فلوجارت تا حد زیادی درک مسئله را برای ذهن ما ساده‌تر و قابل فهم‌تر می‌کند.

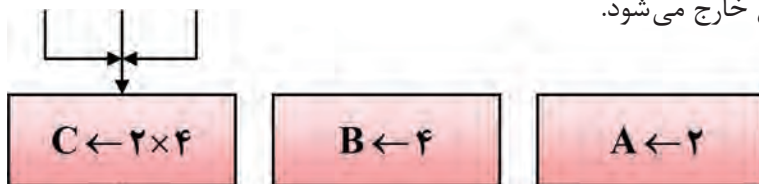
■ **علائم استاندارد برای طراحی فلوجارت:** برای رسم یک فلوجارت لازم است ابتدا با نمادهای آن آشنا شویم. نمادهایی که در روند نما به کار می‌رود شامل بیضی، مستطیل، متوازی‌الاضلاع و لوزی است که هر یک مفهومی به شرح زیر دارند.



شکل ۳-۱

■ **نماد شروع و پایان:** برای شروع و پایان هر الگوریتم در فلوجارت از نماد بیضی مطابق شکل ۳-۱ استفاده می‌شود. معمولاً از نماد شروع یک پیکان خارج و به نماد پایان یک یا چند پیکان وارد می‌شود.

■ **علامت پردازش یا اجرای عملیات:** برای نمایش انجام یک عملیات مانند عملیات محاسباتی و پردازشی از علامت مستطیل مطابق شکل ۳-۲ استفاده می‌شود، مثلاً شکل ۳-۲ می‌گوید در  $A$  مقدار ۲، در  $B$  مقدار ۴ و حاصل ضرب محتوای  $A$  در محتوای  $B$  در  $C$  قرار می‌گیرد. به این نماد یک یا چند پیکان وارد شده و یک پیکان از آن خارج می‌شود.



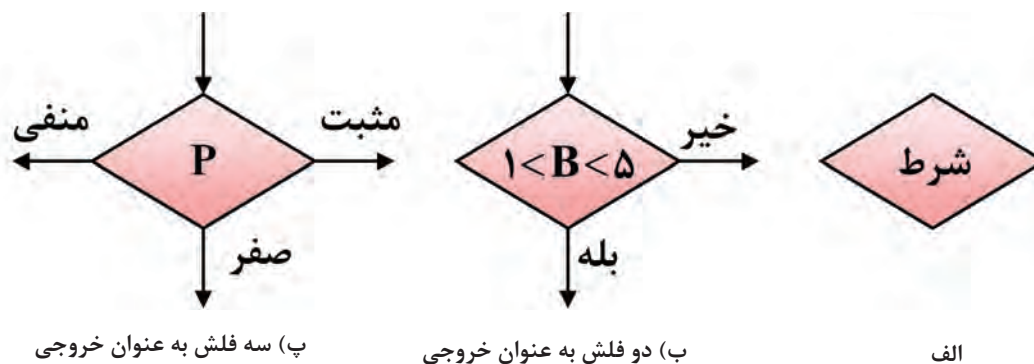
شکل ۳-۲



شکل ۳-۳

■ **نماد ورودی و خروجی:** از علامت متوازی‌الاضلاع برای نمایش عملیات مربوط به مقادیر ورودی و خروجی داده‌ها استفاده می‌شود. در شکل ۳-۳ این نماد و مثال‌هایی از آن را مشاهده می‌کنید.

■ **علامت شرط یا if:** شرط یا شرطها را طبق شکل ۳-۴ داخل لوزی قرار می‌دهیم. در عبارات شرطی، انجام عملیات منوط به برقرار بودن شرط یا شروط مورد نظر است. طبق شکل ۳-۴ به این نماد یک پیکان وارد و متناسب با نیاز دو یا سه پیکان خارج می‌شود. مثلاً در شکل پ ۳-۴ اگر P، مثبت، منفی یا صفر باشد در هر حالت عملیات مشخصی انجام می‌گیرد.



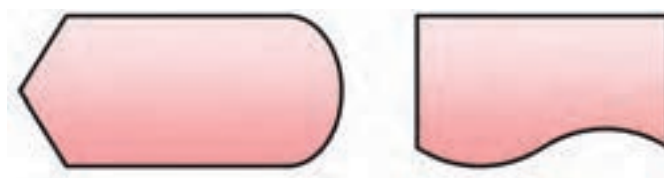
پ) سه فلش به عنوان خروجی

ب) دو فلش به عنوان خروجی

الف

شکل ۳-۴

علامت چاپ: برای چاپ مقدار مورد نظر روی کاغذ یا صفحه نمایش، می‌توان از دو علامت شکل ۳-۵ استفاده کرد. مقدار مورد نظر برای چاپ، در داخل نماد نوشته می‌شود.



برای چاپ روی کاغذ چاپ روی صفحه نمایش

شکل ۳-۵

- تحقیق کنید چه کامپایلرهای دیگری برای AVR به زبان C و پاسکال (PASCAL) وجود دارد.
- با مراجعه به رسانه‌های مختلف انواع نرم‌افزارهای رسم فلوچارت را پیدا کنید و با پرسش از اهل فن، نرم‌افزار مناسب را انتخاب نمایید و در ساعت‌های غیر درسی با آن کار کنید.

فعالیت



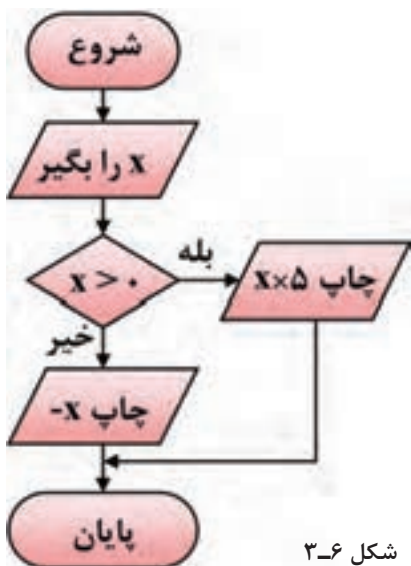
### الگوی پرسش:

- ۱- به مجموعه‌ای از تصاویر و نمادها که الگوریتم (روند) یک برنامه را به صورت تصویری نشان می‌دهد ..... یا ..... می‌گویند.
- ۲- از نماد ..... برای شروع و پایان الگوریتم استفاده می‌شود.
- ۳- از نماد ..... برای پردازش استفاده می‌شود.
- ۴- از نماد لوزی تعداد ..... پیکان داخل و ..... پیکان خارج می‌شود.

مثال ۱: فلوجارت برنامه‌ای را ترسیم کنید که عدد  $x$  را به عنوان ورودی دریافت، سپس اگر عدد  $x$  مثبت بود آن را در ۵ ضرب کرده و چاپ کند در غیراین صورت آن را در یک منفی ضرب کرده و سپس آن را چاپ نماید، شکل ۳-۶.

مثال ۲: فلوجارتی رسم کنید که دو عدد  $a$  و  $b$  را بگیرد و حاصل جمع آنها را چاپ کند. در شکل ۳-۷ الگوریتم و فلوجارت برنامه ترسیم شده است.

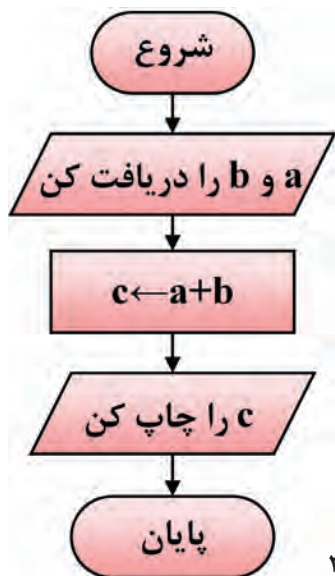
پاسخ مثال ۱



شکل ۳-۶

الگوریتم  
 ۱- شروع  
 ۲-  $x$  را بگیر  
 ۳- اگر  $x > 0$ ، آنگاه  $x \times 5$  را چاپ کن  
 و برو به پایان  
 ۴-  $-x$  را چاپ کن  
 ۵- پایان

پاسخ مثال ۲



شکل ۳-۷

الگوریتم  
 ۱- شروع  
 ۲-  $a$  و  $b$  را دریافت کن  
 ۳-  $c \leftarrow a + b$   
 ۴-  $c$  چاپ کن  
 ۵- پایان

مثال ۳: فلوجارتی رسم کنید که عدد حقیقی  $a$  را دریافت و مقدار تابع چندضابطه‌ای داخل کادر را محاسبه کند و در نهایت نمایش دهد، شکل ۸-۳.

الگوریتم

۱- شروع

۲-  $a$  را دریافت کن

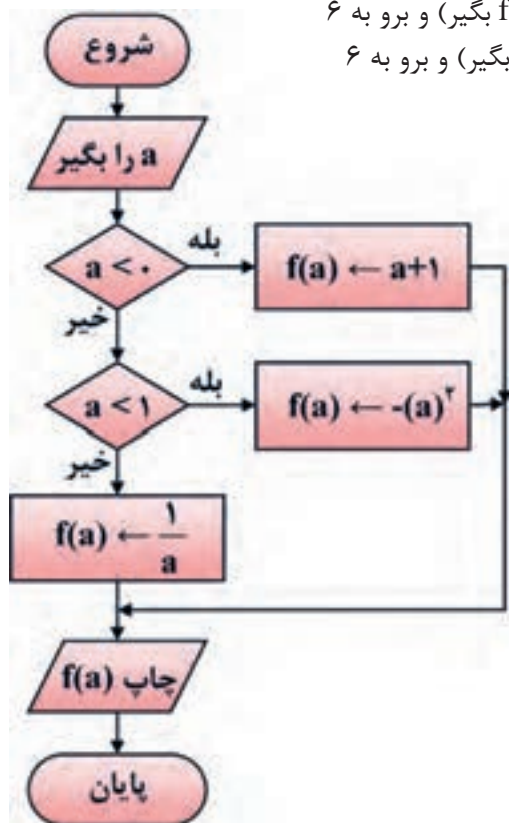
۳- اگر  $a < 0$  سپس  $f(a) \leftarrow a + 1$ ،  $f(a)$  را مساوی  $(a + 1)$  بگیر و برو به ۶

۴- اگر  $0 < a < 1$  سپس  $f(a) \leftarrow -(a^2)$ ،  $f(a)$  را مساوی  $(-a^2)$  بگیر و برو به ۶

۵-  $f(a) \leftarrow \frac{1}{a}$ ،  $f(a)$  را مساوی  $(\frac{1}{a})$  بگیر

۶- چاپ  $f(a)$

۷- پایان



$$f(a) = \begin{cases} a + 1 & a < 0 \\ -a^2 & 0 \leq a < 1 \\ \frac{1}{a} & 1 \leq a \end{cases}$$

شکل ۸-۳- فلوجارت مثال ۳

■ برای روند (الگوریتم) زیر یک فلوجارت رسم کنید.

۱- شروع

۲-  $a$  و  $b$  را دریافت کن

۳-  $d = a \cdot b$  ( $d$  را مساوی  $a$  ضربدر  $b$  بگیر)

۴-  $d$  را چاپ کن

۵- پایان

■ برای رسم فلوجارت‌های ذکر شده از نرم‌افزاری که در مراحل قبل انتخاب کرده‌اید استفاده کنید. سپس فلوجارت رسم شده با نرم‌افزار را با فلوجارت‌های داده شده مقایسه کنید، در صورتی که تعارضی دارد درباره آن بحث کنید.

فعالیت





با مراجعه به رسانه‌های مختلف، جست‌وجو کنید آیا نرم‌افزاری وجود دارد که با دادن الگوریتم، فلوچارت آن را مورد استفاده قرار دهد.

### الگوی پرسش

- ۱- فلوچارت برنامه‌ای را رسم کنید که طول و عرض یک مستطیل را دریافت و مساحت و محیط آن را محاسبه و چاپ کند.
- ۲- روندنمای برنامه‌ای را رسم کنید که سه عدد را دریافت کرده و آنها را با هم مقایسه کند، سپس به ترتیب صعودی چاپ نماید.
- ۳- فلوچارت برنامه‌ای را رسم کنید که ضرایب یک معادله درجه دوم را دریافت کند و در صورت داشتن ریشه‌های حقیقی آنها را محاسبه و چاپ نماید. در غیر این صورت با پیغام مناسب مشخص نماید که معادله ریشه حقیقی ندارد.
- ۴- فلوچارت پرسش‌های ۱ تا ۳ را با نرم‌افزار مناسب رسم کنید.

## ۲-۳- یادآوری ساختار برنامه‌نویسی به زبان C

زبان برنامه‌نویسی C در سال ۱۹۷۳ در آزمایشگاه بل طراحی و ارائه شد. خالق این زبان برنامه‌نویسی دنیس ریچی نام دارد. وی در کنار کن تامپسون علاوه بر خلق زبان C سیستم عامل یونیکس را توسعه داد. لذا در زمره افراد نامدار در زمینه محاسبات جدید و به عنوان یک فرد مشهور و نامی شناخته شده است. از ویژگی‌های زبان برنامه‌نویسی C، امکان دسترسی به سخت‌افزار و حافظه‌ها، امکان برنامه‌نویسی مستقل (ماژولار - Modular)، استفاده از برنامه‌های یک میکروکنترلر با کمی تغییر برای میکروکنترلرهای دیگر به خاطر استاندارد بودن این زبان نام برد.

در کتاب کارگاهی مونتاژ و دمونتاژ پایه یازدهم با ساختار و چگونگی برنامه‌نویسی به زبان C و کامپایلر کد ویژن آشنا شدید. علت استفاده از کدویژن فراگیر بودن آن در بین کاربران AVR، قابل اجرا بودن آن با حداقل سخت‌افزار رایانه‌ای و ساده و سبک بودن نرم‌افزار است. در این بخش مباحث تکمیلی و پیشرفته‌تری را درباره این موضوع آموزش می‌دهیم.

- با استفاده از رسانه‌های موجود، در مورد زبان‌های برنامه‌نویسی قبل از پدید آمدن زبان C گزارشی تهیه و به کلاس ارائه دهید.
- کامپایلر دیگری علاوه بر کد ویژن را بررسی و سعی کنید یک برنامه ساده را در آن محیط بنویسید و با میکروکنترلر اجرا کنید.



### ■ نرم‌افزار

پس از طراحی و ساخت سخت‌افزارهای میکروکنترلری، لازم است برنامه‌ای برای کنترل آن نوشته شود. تنها زبان قابل فهم برای پردازنده‌ها، زبان ماشین است که از کدهای دو دویی یا باینری (۱۱۰۰۰۱۰۰) تشکیل می‌شود. نوشتن برنامه و رفع اشکال آن به زبان ماشین (low level) کاری دشوار و طاقت فرسا است؛ لذا برای حل این مشکل، از زبان‌های سطح بالا (high level) مانند بیسیک، پاسکال و C استفاده می‌کنیم.



زبان‌های سطح بالا زبان‌هایی هستند که از نظر ساختاری به زبان محاوره انسان شباهت بیشتری دارند. هر قدر زبان برنامه‌نویسی به زبان ماشین که همان صفر و یک است نزدیک‌تر شود، آن زبان را اصطلاحاً زبان سطح پایین می‌گویند.

در مورد زبان اسمبلی Assembly تحقیق کنید و نتیجه را در قالب یک گزارش ارائه دهید.

**ساختار زبان C:** برای برنامه‌نویسی میکروکنترلرها به زبان C باید از ساختار کلی این زبان که در جدول ۳-۱ نشان داده شده است، پیروی کنیم:

**جدول ۳-۱- ساختار و تعاریف مربوط به زبان C همراه با یک نمونه از برنامه**

ساختار تعاریف مربوط به زبان C	نمونه‌ای از برنامه زبان C
(هدر فایل) #include ماکروها متغیرهای عمومی معرفی توابع void main (void) { .... .... برنامه اصلی .... .... .... } بدنه تابع { .... برنامه تابع .... .... }	<pre>#include &lt;mega.h&gt; #define key PINB.۳ unsigned char i=۰; void wait (void); void main (void) {   DDRD=۰xff;   while (۱)   {     if ( key==۰ )     {       i++;       PORTD=i;       wait( );     }   } } void wait (void) {   int j;   for(j=۰;j&lt;۳۰۰۰۰;j++); }</pre>
⇒ ادامه در ستون مقابل	





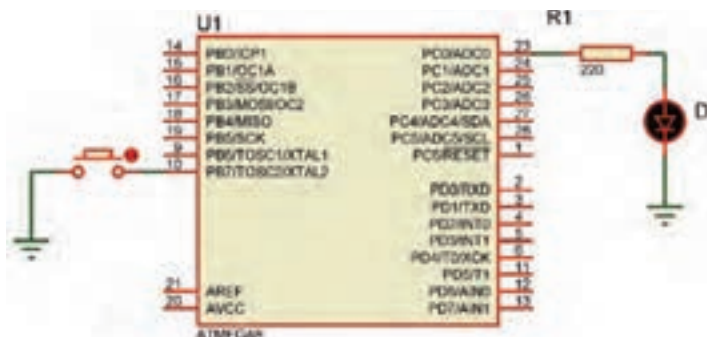
جدول ۳-۱ را از طریق بارش فکری به بحث بگذارید و نتیجه را جمع‌بندی کنید و در قالب یک گزارش ارائه دهید.

□ **هدر (سرتیتر) فایل (Header):** برای تعریف ثابت‌ها، متغیرهای عمومی، دستورات پیش پردازنده و همچنین تعریف اولیه توابع از هدر فایل‌ها استفاده می‌کنیم. در نمونه ارائه شده در جدول ۳-۱ عبارت `#include <mega8.h>` ویژگی‌ها و مشخصات سخت‌افزاری قطعه ATMEGA8 و آدرس حافظه‌ها و رجیسترهای آن را برای کامپایلر معرفی می‌کند.

□ **ماکرو (Macro):** ماکرو رشته‌ای است که می‌تواند شامل حرف، عدد، مقادیر ثابت، توابع و مانند آن باشد. برای تعریف ماکرو از پیش پردازنده `#define` استفاده می‌شود. همچنین در انجام تعاریف جدید به جای مقادیر استاندارد و از پیش تعیین شده در مترجم نیز به کار می‌رود. استفاده از ماکروها باعث می‌شود تا با نام‌گذاری‌های جدید به جای نام‌های استاندارد، برنامه‌نویسی ساده‌تر و قابل فهم‌تر شود و از مراجعه پی در پی به نقشه سخت‌افزار جلوگیری به عمل آید. پیش پردازنده `#define` به صورت شکل ۳-۹ نوشته می‌شود: به مثال ۴ توجه کنید.

نام استاندارد    نام جدید    `#define`

شکل ۳-۹- روش نوشتن پیش پردازنده در ماکرو



شکل ۳-۱۰

جدول ۳-۲- برنامه مثال ۴

بدون استفاده از ماکرو	با استفاده از ماکرو
<code>if (PINB.Y==0) PORTC.0=1;</code>	<code>#define key PINB.Y</code> <code>#define led PORTC.0</code> <code>if (key==0) led=1;</code>

مثال ۴: در شکل ۳-۱۰ می‌خواهیم با اتصال کلید `PINB.7`، LED متصل به `PORTC.0` روشن شود، برنامه را یک بار بدون استفاده از ماکرو و سپس با استفاده از ماکرو می‌نویسیم.  
**حل:** در جدول ۳-۲ برنامه نوشته شده با استفاده از ماکرو و بدون استفاده از ماکرو را ملاحظه می‌کنید.

□ **متغیر Variable:** در هر زبان برنامه‌نویسی لازم است محل‌هایی از حافظه را برای نگهداری اعداد، کاراکترها و رشته‌ها در نظر بگیریم تا بتوانیم در هنگام اجرای برنامه آنها را بخوانیم یا روی آن بنویسیم. در نمونه ارائه شده در جدول ۳-۱ این متغیر با عبارت `unsigned char i=0` نوشته شده که متغیر آن `i` و مقدار اولیه صفر اعلان شده است.

□ **تابع (Function):** در برنامه‌های طولانی و پیچیده که شامل چندین بخش منطقی و مستقل از هم هستند، بهتر است برای هر قسمت منطقی، برنامه جداگانه‌ای نوشته شود. برنامه‌هایی که برای هر یک از

بخش‌ها نوشته می‌شود را تابع می‌نامند. با استفاده از تابع، از نوشتن تکراری دستورها تا حد زیادی جلوگیری به عمل می‌آید. معمولاً توابع در ابتدای برنامه معرفی و بعد از تابع main تعریف می‌شوند. در نمونه ارائه شده در جدول ۳-۱ یک تابع با نام wait معرفی و در انتهای برنامه تعریف شده است. این تابع با هر بار اجرا یک تاخیر ایجاد می‌کند.

مقدار = نام متغیر	نوع متغیر
نام متغیر	نوع متغیر

□ **تابع main:** ساختار زبان C بر پایه توابع بنا شده است. برای ترجمه و اجرای برنامه لازم است حتماً یکی از توابع را به نام main معرفی کنیم تا به عنوان تابع اصلی شناخته شود. اجرای برنامه با دستورهایی نوشته شده در تابع main آغاز می‌شود.

شکل ۳-۱۱- ساختار چگونگی اعلان یک متغیر در برنامه C

جدول ۳-۳

unsigned char	a=۵ ;
int	b,c,d=۲۰۰۰;
float	pi=۳.۱۴;
char	t='R';
char	s[ ]="REZA";

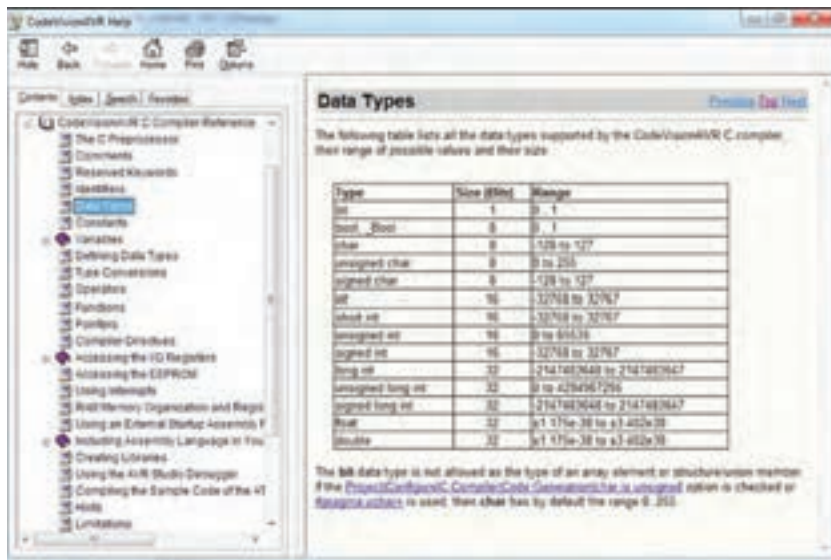
### ■ روش‌های بیان و نوشتن متغیر Variable:

همان‌طور که گفته شد، در برنامه‌نویسی لازم است محل‌هایی از حافظه را برای نگهداری اعداد کاراکتر و رشته‌ها در نظر بگیریم. در زبان C برای اعلان یک متغیر از ساختار شکل ۳-۱۱ استفاده می‌کنیم. مثال ۵: نمونه‌هایی از معرفی و اعلان متغیر در برنامه را در جدول ۳-۳ آورده‌ایم.



مقدار دهی اولیه اختیاری است.

■ با توجه به یافته‌های خود این مثال را بررسی کنید و با انطباق با جدول ۳-۱ فرایند آن را تعریف نمایید.  
 ■ انواع متغیرها و رنج اعداد قابل نمایش به وسیله آنها در Help برنامه و در بخش Data Types طبق شکل ۳-۱۲ در دسترس است.



شکل ۳-۱۲- جدول متغیرهای C در کدوین

■ بازه اعداد قابل ذخیره در متغیرهای بدون علامت و علامت‌دار از روابط داده شده در جدول ۳-۴ محاسبه می‌شود.

جدول ۳-۴

بدون علامت	علامت‌دار
$0 \leq N \leq 2^n - 1$	$-2^{n-1} \leq N \leq 2^{n-1} - 1$
در روابط بالا n تعداد بیت و N رنج عدد قابل نمایش می باشد	

مثلاً برای دو متغیر signed char و unsigned char که هشت بیتی هستند، بازه اعداد قابل ذخیره را در جدول ۳-۵ ملاحظه می‌کنید:

جدول ۳-۵

unsigned char	signed char
$0 \leq N \leq 2^8 - 1$	$-2^{8-1} \leq N \leq 2^{8-1} - 1$
$0 \leq N \leq 255$	$-128 \leq N \leq 127$

با اعضاء گروه خود محاسبات مربوط به بازه اعداد قابل نمایش را برای یک متغیر ۱۶ بیتی انجام دهید.

فعالیت



### ۳-۳- نوشتن اعداد در مبناهای مختلف در زبان C

در کد ویژن می‌توانید اعداد را در مبناهای ۲، ۸، ۱۰ و ۱۶ بنویسید. در جدول ۳-۶، اعداد ۰ تا ۱۵ در مبناهای ۲، ۸، ۱۰ و ۱۶ جهت یادآوری نوشته شده است.

جدول ۳-۶- تبدیل مبنا

مبنای ۱۰	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
مبنای ۲	۰۰۰۰	۰۰۰۱	۰۰۱۰	۰۰۱۱	۰۱۰۰	۰۱۰۱	۰۱۱۰	۰۱۱۱	۱۰۰۰	۱۰۰۱	۱۰۱۰	۱۰۱۱	۱۱۰۰	۱۱۰۱	۱۱۱۰	۱۱۱۱
مبنای ۱۶	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	A	B	C	D	E	F
مبنای ۸	۰	۱	۲	۳	۴	۵	۶	۷	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵	۱۶	۱۷

جدول ۳-۷

$(12)_{10} = (1100)_2 = (14)_8 = (C)_{16}$	
PORTD=۱۲;	مبنای ۱۰
PORTD=۰b۱۱۰۰;	مبنای ۲
PORTD=۰O۱۴;	مبنای ۸
PORTD=۰xC;	مبنای ۱۶

مثال ۶: می‌خواهیم عدد ۱۲ را در مبناهای مختلف به PORTD ارسال کنیم. برنامه مربوط به مثال ۶ به صورت جدول ۳-۷ نوشته می‌شود. همان‌طور که

### جدول ۳-۸

```
#include <mega.h>
#include <delay.h>
unsigned char i=0;
void main(void)
{
  DDRB=0xFF;
  while(1)
  {
    PORTB=i;
    delay_ms(500);
    i++;
  }
}
```

در جدول ۳-۶ دیده می‌شود برای نوشتن اعداد در مبنای ۲ از پیشوند ۰b، برای مبنای ۸ از پیشوند ۰O و برای مبنای ۱۶ از پیشوند ۰X استفاده می‌شود. مبنای ۱۰ نیازی به پیشوند ندارد. توجه داشته باشید هر سه پیشوند با صفر شروع می‌شود. مثال ۷: طبق شکل ۳-۹ هشت عدد LED را به PORTB متصل کنید و یک شمارنده بالا شمار بر روی آن ایجاد نمایید. در جدول ۳-۸ برنامه مربوط به مثال ۷ را ملاحظه می‌کنید.

در نوشتن برنامه به زبان C باید دستورها با حروف کوچک و نام رجیسترها با حروف بزرگ نوشته شود.

نکته



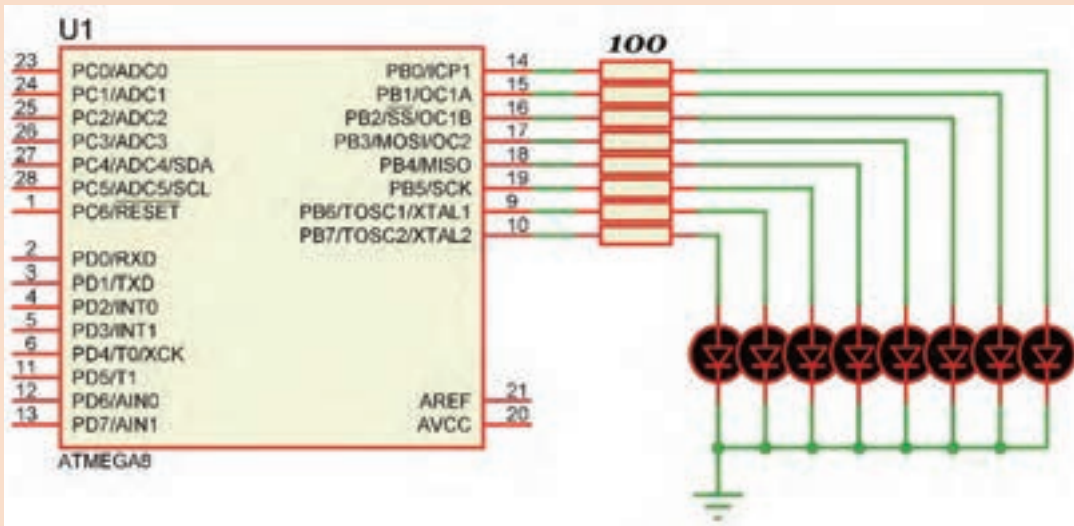
برنامه مثال ۷ را در گروه خود بررسی کنید و فرایند اجرای آن را طبق الگوی برنامه داده شده به بحث بگذارید. نتیجه را در قالب یک گزارش ارائه دهید.

فعالیت



با توجه به مدار شکل ۳-۱۳ برنامه‌ای بنویسید که روی PORTB یک شمارنده پایین شمار ایجاد کند.

فعالیت



شکل ۳-۱۳

## ۳-۴- عملگرها (Operators)

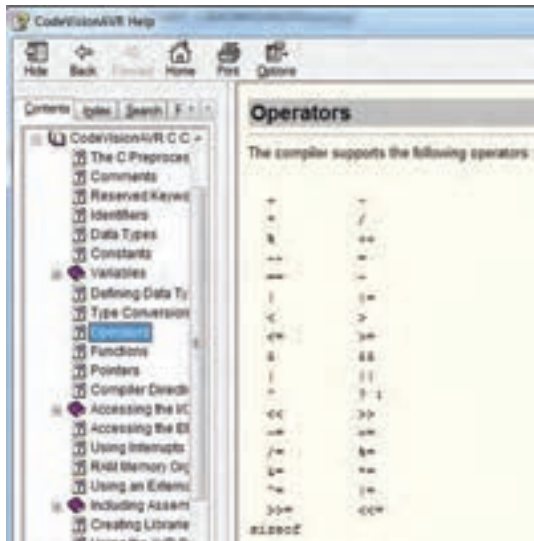
### ■ تعریف عبارت

در ریاضیات، به مجموعه‌ای مرکب از اعداد، متغیرها و عملگرها، عبارت گفته می‌شود. برای مثال هر یک از جملات زیر یک عبارت ریاضی است.

$$-x \quad 3/14 \times R \times R \quad 7 \times 2 + 4 \quad 3 + 6$$

در جمله  $3+6$  علامت  $+$  را عملگر و دو عدد  $3$  و  $6$  را عملوند می‌گویند. از آنجایی که عملگر جمع، بر روی دو عملوند عمل می‌کند به آن عملگر دوتایی گفته می‌شود.

در جمله  $-x$  عملگر قرینه‌ساز فقط یک عملگر دارد و به آن عملگر یکتایی گفته می‌شود.



شکل ۳-۱۴- عملگرها در کدویژن

جدول ۳-۹- اولویت عملگرهای ریاضی در زبان C همراه با مثال

الویت	نام عملگر	نشانه	مثال	نوع عملگر
۱	قرینه	-	-7	یکتایی
۲	ضرب تقسیم باقیمانده تقسیم	*	7*12	دوتایی
		/	17/4	
۳	جمع تفریق	+	67+9	دوتایی
		-	56-78	

جدول ۳-۱۰- عملگرهای افزایش و کاهش همراه با مثال

نتیجه	عملگر	مقدار اولیه
a=6	a++	a=5
a=4	a--	a=5

عملگرهایی که می‌توانید در زبان C و در برنامه کدویژن استفاده کنید، طبق شکل ۳-۱۴ در Help برنامه و در بخش Operators آورده شده است. در جدول ۳-۹ فهرست عملگرهای ریاضی را به ترتیب اولویت مشاهده می‌کنید:

**عملگرهای جمع (+)، تفریق (-) و ضرب (\*):** این عملگرها در واقع همان محاسبات ریاضی هستند که تاکنون آموخته‌اید.

**عملگرهای تقسیم معمولی (/)، باقی‌مانده (/) و اعشاری (.):** عملکرد این عملگرها به صورت زیر تعریف می‌شود.

$$\begin{array}{r}
 13 \overline{) 5} \\
 \underline{10} \phantom{0} \\
 3
 \end{array}$$

$13/5=2$   
 $13\%5=3$

در عملگر تقسیم معمولی اگر هر دو عملوند از نوع عدد صحیح باشند، نتیجه نیز عددی صحیح خواهد بود. اگر یکی یا هر دو عملوند از نوع عدد اعشاری باشند، نتیجه عدد اعشاری خواهد شد.

$7/2=3$  ,  $7/2.0=3.5$  ,  $7.0/2=3.5$  ,  $7.0/2.0=3.5$   
 عملگرهای افزایش به مقدار یک واحد (++) و کاهش به مقدار یک واحد (--): در جدول ۳-۱۰ عملکرد این عملگرها با مثال نشان داده شده است.

عملگرهای انتساب (=)، شرط مساوی (==) و شرط نامساوی (!=): نماد = مقداری را به یک متغیر نسبت می‌دهد و نماد == بررسی می‌کند که آیا دو مقدار با هم مساوی هستند یا خیر. همچنین نماد != بررسی می‌کند که آیا دو مقدار با هم نامساوی هستند یا خیر.

□ در دستور K=۵ مقدار ۵ در متغیر K قرار می‌گیرد.

□ در دستور K==۵ مقدار K با عدد ۵ مقایسه می‌شود. اگر برابر باشند، نتیجه دستور مقدار یک یا TRUE است. در صورتی که مقدار K با عدد ۵ برابر نباشد، نتیجه دستور مقدار صفر یا FALSE خواهد بود.

□ در دستور K!=۵ مقدار K با عدد ۵ مقایسه می‌شود. اگر برابر نباشند، نتیجه دستور مقدار یک یا TRUE و در صورتی که برابر باشند مقدار برابر با صفر یا FALSE است.

### جدول ۱۱-۳- عملگرهای بیتی و منطقی

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

### عملگرهای بیتی و منطقی: نماد این عملگرها

به صورت !، ~، &، &&، |، || و ^ است. در جدول ۱۱-۳ عملگرهای بیتی و منطقی و نماد آنها را ملاحظه می‌کنید.

در عملگرهای بیتی ابتدا عملوندها به صورت باینری نوشته و سپس بیت به بیت عمل مورد نظر بر روی بیت‌ها انجام می‌شود.

مثال ۸: اگر  $a = 0x56$  و  $b = 0x9C$  باشد، مقادیر زیر را به دست آورید.

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

در جدول ۱۲-۳ عملگرها، نوع عملیات و نتیجه حاصل شده را ملاحظه می‌کنید.

در عملگرهای منطقی ابتدا درستی یا نادرستی هر عبارت مشخص و سپس عمل مورد نظر بر روی آنها انجام می‌شود.

### جدول ۱۲-۳- پاسخ مثال ۷ عملگرهای بیتی

عملگر	عملیات	نتیجه
AND	$a = 0x56$ ۰۱۰۱۰۱۱۰ $b = 0x9C$ ۱۰۰۱۱۱۰۰ $a \& b$ ۰۰۰۱۰۱۰۰	$PORTD = 00010100 = 0x14$
OR	$a = 0x56$ ۰۱۰۱۰۱۱۰ $b = 0x9C$ ۱۰۰۱۱۱۰۰ $a   b$ ۱۱۰۱۱۱۱۰	$PORTD = 11011110 = 0xDE$
XOR	$a = 0x56$ ۰۱۰۱۰۱۱۰ $b = 0x9C$ ۱۰۰۱۱۱۰۰ $a \wedge b$ ۱۱۰۰۱۰۱۰	$PORTD = 00010100 = 0xCA$
NOT	$a = 0x56$ ۰۱۰۱۰۱۱۰ $\sim a$ ۱۰۱۰۱۰۰۱	$PORTD = 10101001 = 0xA9$



در گروه کاری خود، جدول ۳-۱۲ را به بحث بگذارید و نتیجه آن را جمع‌بندی کرده و در قالب گزارش ارائه دهید.

مثال ۹: در جدول ۳-۱۳ که قطعه‌ای از یک برنامه است، اگر  $a=5$  و  $b=8$  باشد، بر روی پورت D چه عددی نمایش داده خواهد شد؟

در این مثال نتیجه عبارت  $a > 6$  نادرست (false) و نتیجه عبارت  $b < 10$  درست (true) می‌باشد. پس نتیجه عملگر منطقی AND نادرست خواهد بود و به دلیل برقرار نبودن شرط، دستور مربوط به else اجرا و روی PORTD عدد ۵۵ نمایش داده می‌شود.



شکل ۳-۱۵ شیفت به چپ و راست

جدول ۳-۱۳- کد مثال ۹

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

شیفت به چپ و راست: در شکل ۳-۱۵ علامت شیفت

یا جابه‌جایی به چپ و راست را مشاهده می‌کنید.

□ با اجرای عملگرهای شیفت، از یک سمت «صفر» وارد و از سمت دیگر «یک بیت» خارج می‌شود. نماد

شیفت به چپ با دو علامت کوچکتر «<<» و نماد شیفت به راست با دو علامت بزرگتر «>>» نشان

داده می‌شود. در عبارت  $a >> 3$  یعنی عدد موجود در متغیر a سه بار به سمت راست انتقال یابد.

مثال ۱۰: اگر  $b = 0 \times FE$  و  $a = 0 \times CF$  باشد، حاصل عبارت  $PORTD = ((a >> 3) \& (b << 2))$  را به دست آورید.

در جدول ۳-۱۳ پاسخ مربوط به مثال ۱۰ را مشاهده می‌کنید.  
 $a = 11001111$        $b = 11111110$



درباره جدول ۳-۱۴ و چگونگی کار عملگرها در آن بحث کنید و نتیجه را در یک پاراگراف بنویسید.

جدول ۳-۱۴- پاسخ مثال ۹ عملگرهای شیفت و AND

جدول ۳-۱۵- کد مثال ۱۱

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

عملگر	نتیجه
$a >> 3$	0011001
$b << 2$	11111000
$\&$	0011000

عملگرهای شرط بزرگ‌تر (>) و شرط کوچک‌تر (<):

مثال ۱۱: در قطعه برنامه‌ای که در جدول ۳-۱۵ نشان داده شده است، اگر  $a=5$  و  $b=8$  باشد، پورت D چه عددی را نمایش می‌دهد؟

در این مثال به دلیل برقرار بودن شرط، دستور  $PORTD = 99$  اجرا می‌شود.



جدول ۳-۱۵ را از طریق بارش فکری به بحث بگذارید و نتیجه را با پاسخ داده شده مقایسه کنید.

### جدول ۳-۱۶- عملگرهای تخصیص مرکب

شکل اصلی	شکل خلاصه شده
$a = a + 5$	$a += 5$
$b = b * 7$	$b *= 7$
$c = c \& 8$	$c \& = 8$
$d = d \ll 4$	$d \ll = 4$

عملگر شرطی «?:» این عملگر مانند دستور if else

عمل می کند و به صورت زیر نوشته می شود:

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

□ اگر شرط موجود در عبارت شرطی برقرار باشد دستور

اول و در غیر این صورت، دستور دوم اجرا می شود.

(PORTD = ۵۵) : (PORTD = ۹۹) ? (a > b)

عملگرهای تخصیص مرکب

توسط این روش می توان عبارات محاسبه ای را مانند

جدول ۳-۱۶ به صورت خلاصه نوشت.

با مراجعه به رسانه های مختلف، درباره عملگرهای تخصیص مرکب تحقیق کنید و نتیجه را در قالب گزارش ارائه دهید.



### جدول ۳-۱۷ مثال های مربوط به

عملگر sizeof و نتیجه آنها

متغیر	عملیات	نتیجه
char a;	$x = \text{sizeof}(a)$	$x = 1$
int b;	$x = \text{sizeof}(b)$	$x = 2$
float c;	$x = \text{sizeof}(c)$	$x = 4$

### جدول ۳-۱۸

PORTB	PORTC	PORTD
۱۰۱۱۰۱۱۰	۳	۵

### جدول ۳-۱۹

PORTB	PORTC	PORTD
S <sub>1</sub>	S <sub>2</sub>	S <sub>1</sub>
۱۰۰۱	۰۱۱۱	۹
		۷

عملگر sizeof:

خروجی عملگر sizeof، مقدار حافظه ای است که یک

متغیر، بر حسب بایت اشغال می کند. در جدول ۳-۱۷

مثال هایی آورده شده است.

الگوی پرسش

۱- برنامه ای بنویسید که عددی را از PORTB

دریافت کند، اگر عدد دریافت شده، فرد بود روی

PORTD عدد ۱ و اگر زوج بود عدد ۲ را روی

PORTD نمایش دهد.

۲- برنامه ای بنویسید که یک عدد باینری را مشابه

جدول ۳-۱۸ از PORTB دریافت کند، و تعداد

صفرهای آن را روی PORTC و تعداد یک های آن

را روی PORTD نمایش دهد.

۳- برنامه ای بنویسید که مشابه جدول ۳-۱۹ یک عدد

باینری را از PORTB دریافت کند، در صورتی که

خروجی دو سنسور به این پورت متصل شده باشند،

عدد ارسالی هر سنسور را جدا کرده و به پورت های

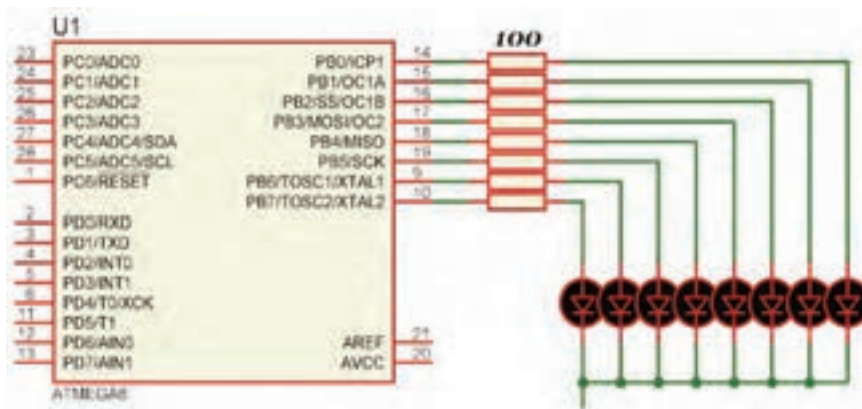
C و D ارسال نماید و آن را نمایش دهد.



مثال ۱۲: با توجه به شکل ۳-۱۶ که هشت عدد LED به PORTB متصل است برنامه‌ای بنویسید که LEDها به صورت چشمک‌زن عمل کنند.

جدول ۳-۲۰

```
#include <mega.h>
#include <delay.h>
void main(void)
{
    DDRB=0xff;
    while(1)
    {
        PORTB=0b00011111;
        delay_ms(500);
        PORTB=0b11110000;
        delay_ms(500);
    }
}
```



شکل ۳-۱۶ مدار مثال ۱۲

در ساعات غیر درسی، برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی نمایید. در صورت امکان، مدار را به صورت سخت‌افزاری نیز اجرا کنید. نتیجه را در قالب یک گزارش ارائه دهید.

فعالیت



مثال ۱۳: برای شکل ۳-۱۶ برنامه‌ای بنویسید که یک شمارنده حلقوی روی آن ایجاد شود.

جدول ۳-۲۱

#include <mega.h>	delay_ms(500);
#include <delay.h>	PORTB=0b00010000;
void main(void)	delay_ms(500);
{	PORTB=0b00010000;
DDRB=0xff;	delay_ms(500);
while(1)	PORTB=0b01000000;
{	delay_ms(500);
PORTB=0b00000001;	PORTB=0b01000000;
delay_ms(500);	delay_ms(500);
PORTB=0b00000010;	PORTB=0b10000000;
delay_ms(500);	delay_ms(500);
PORTB=0b00000100;	}
⇒ ادامه در ستون مقابل	}

در مثال ۱۳ چون برنامه با دستورهای ساده نوشته شد تعداد خطوط برنامه نیز بسیار زیاد شده است. مثال ۱۴: برنامه مثال ۱۳ را به کمک حلقه و دستور انتقال به چپ << بنویسید. این کار باعث کاهش تعداد خطوط برنامه می‌شود. برنامه نوشته شده را با برنامه داده شده در جدول ۳-۲۲ مقایسه کنید.

## جدول ۳-۲۲

<pre>#include &lt;mega.h&gt; #include &lt;delay.h&gt; unsigned char i; void main(void) {   DDRB=0xff;    ⇒ ادامه در ستون مقابل</pre>	<pre>while(1) {   for (i=0 ; i&lt;8 ; i++)   {     PORTB=1&lt;&lt;i;     delay_ms(500);   } }</pre>
--	---

در این برنامه متغیر  $i$  با هر بار اجرای حلقه `for` یک واحد افزایش می‌یابد. در این حالت، در دستور `PORTB=1<<i;` عدد یک به اندازه عدد متغیر  $i$  به سمت چپ انتقال می‌یابد.

نکته



برای شکل ۳-۱۲ برنامه‌ای بنویسید که یک شمارنده جانسون بر روی آن ایجاد شود. این فعالیت را یک بار با دستوره‌های ساده و بار دیگر با استفاده از حلقه و دستور انتقال انجام دهید.

فعالیت



**راهنمایی:** از عملگر  $\wedge$  (XOR) استفاده کنید.

□ در شمارنده جانسون LEDها یکی پس از دیگری روشن می‌شوند و روشن باقی می‌مانند تا همه آنها روشن شوند. سپس به همان ترتیب که روشن شده بودند خاموش خواهند شد.

## ۳-۵- دستور شرطی if

اگر قرار باشد دستور یا دستورهایی بنا به شرایط خاص انجام شود، از دستوره‌های شرطی استفاده می‌کنیم. یکی از دستوره‌های شرطی پرکاربرد دستور `if` است که در شکل الف - ۳-۱۷ مشاهده می‌کنید. □ توجه داشته باشید که نتیجه عبارت شرطی همواره یکی از دو حالت (`true`) یا (`false`) خواهد بود. فلوجارت دستور `if` را در شکل ب - ۳-۱۷ مشاهده می‌کنید.

اگر تعداد دستوره‌های بعد از `if` بیش از یک دستور باشد، طبق شکل پ - ۳-۱۷ باید بین دو آکولاد `{}` قرار گیرد.

نکته



شکل ۳-۱۷

### جدول ۳-۲۳

<pre>#include &lt; mega8.h &gt; #include &lt; delay.h &gt; signed char i=0; void main (void) {   DDRB = 0xff;   while (1)   ⇒ ادامه در ستون مقابل</pre>	<pre>{   PORTB =i;   delay_ms (۵۰۰);   i++;   if (i==۱۰) i=0; }</pre>
---	---

مثال ۱۵: برای شکل ۳-۱۶ برنامه‌ای بنویسید که یک شمارنده باینری بالا شمار صفر تا ۹ ساخته شود.

در جدول ۳-۲۳ برنامه داده شده است. این برنامه را بررسی کنید و درباره آن بحث کنید.

در ساعات غیر درسی، برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی نمایید. در صورت امکان، مدار را به صورت سخت‌افزاری نیز اجرا کنید. نتیجه را در قالب یک گزارش ارائه دهید.

فعالیت



فعالیت

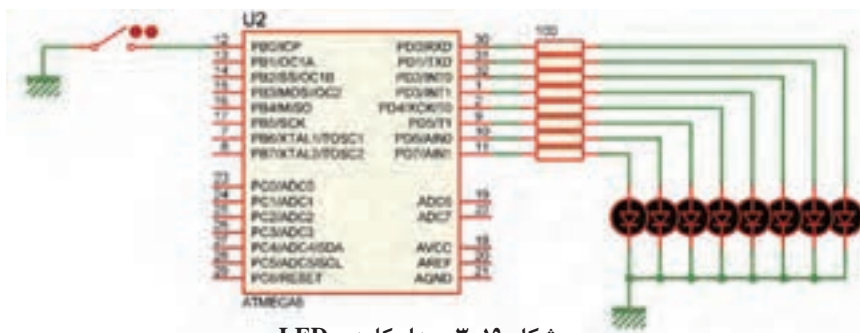


برنامه مثال ۱۵ را طوری بنویسید که یک شمارنده باینری پایین شمار ۹ تا صفر داشته باشیم.

### ۳-۶- دستور if-else

اگر لازم باشد که در صورت برقرار بودن یک شرط، دستور یا دستورهایی اجرا شود، و در صورت برقرار نبودن آن شرط، دستورهایی دیگری اجرا شود، از دستور if-else استفاده می‌کنیم. ساختار این دستور در شکل ۳-۱۸ نشان داده شده است.

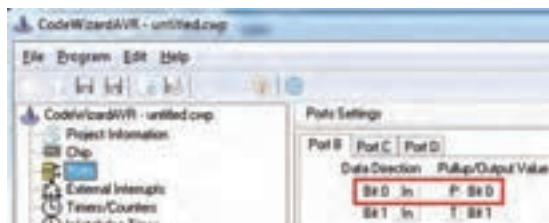
مثال ۱۶: هشت عدد LED به PORTD و یک کلید به PINB.۰ متصل است مانند شکل ۳-۱۹. برنامه‌ای بنویسید که اگر کلید باز بود تمام LEDهای روی PORTD روشن و اگر کلید بسته بود همه LEDها خاموش شوند.



شکل ۳-۱۹- مدار کلید و LED



شکل ۳-۱۸- دستور if-else



شکل ۳-۲۰ تنظیم

توجه داشته باشید هنگام تنظیم wizard باید مانند شکل ۳-۲۰ بیت صفر مربوط به PORTB، PB.۰ را در حالت ورودی قرار دهیم و Pull\_Up داخلی آن را فعال کنیم.

در جدول ۳-۲۴ برنامه داده شده است. این برنامه را بررسی و درباره آن بحث کنید.

### جدول ۳-۲۴

<pre>#include &lt;megaA.h&gt; #include &lt;delay.h&gt; void main (void) { while(1) { if ( PINB.0 == 1) { ⇒ ادامه در ستون مقابل</pre>	<pre>PORTD = 0b11111111; } else { PORTD = 0b00000000; } } }</pre>
--	---

برای سخت‌افزار شکل ۳-۱۹ برنامه‌ای بنویسید که اگر کلید باز بود روی PORTD یک شمارنده بالا شمار و اگر کلید بسته بود شمارنده پایین شمار داشته باشیم.



### ۳-۷- حلقه (Loop)

در برنامه‌نویسی حالت‌هایی پیش می‌آید که لازم است دستور یا دستورهای چندین بار به صورت حلقه و تکراری اجرا شود. در این شرایط باید آنها را درون یک حلقه قرار دهیم تا به تعداد دفعات مورد نیاز تکرار شوند. در هر حلقه، یک شمارنده (Counter) وجود دارد که آن را با عدد حلقه پر می‌کنیم و با هر بار اجرا یک واحد از آن کم می‌کنیم. وقتی محتوای شمارنده صفر شود، از حلقه خارج می‌شویم.

■ **دستور while**: یکی از روش‌های ایجاد حلقه استفاده از دستور while است که در شکل الف - ۳-۲۱ آن را ملاحظه می‌کنید.

به نکات ذکر شده در شکل‌های ب - ۳-۲۱ و پ - ۳-۲۱ توجه کنید و آنها را مورد بحث قرار دهید. سپس نتیجه را جمع‌بندی کنید.



نکته: اگر می‌خواهید برنامه روی خطی متوقف شود دستور زیر را بنویسید.

```
while (1);
```

به نقطه ویرگول ؛ انتهای دستور دقت کنید.

پ

نکته: برای ایجاد یک حلقه بی‌نهایت می‌توان به شکل زیر عمل کرد.

```
while (1)
{
.....
.....
}
```

ب

شرط اجرای حلقه (while)

```
{
....
....
}
```

دستورها

الف

شکل ۳-۲۱

### جدول ۳-۲۵

<pre>#include &lt;mega.h&gt; #include &lt;delay.h&gt; signed char i; void main (void) {   DDRB=0xff;   i=5;   while(i &gt; 0) //     تا زمانی که i بزرگ‌تر از صفر است     حلقه انجام می‌شود     =&gt; ادامه در ستون مقابل</pre>	<pre>{   PORTB = 0xff;   delay_ms(500);   PORTB = 0x00;   delay_ms(500);   i--; } while (1);}</pre>
---	---

مثال ۱۷: برای مدار شکل ۳-۱۵ با استفاده از دستور while برنامه‌ای بنویسید که LEDها را پنج بار روشن و خاموش کند. در جدول ۳-۲۵ برنامه داده شده است. این برنامه را بررسی کنید و درباره آن بحث کنید.

با استفاده از دستور while برنامه‌ای بنویسید که بر روی PORTB یک شمارنده باینری بالا شمار صفر تا ۹ داشته باشیم.

فعالیت



■ **حلقه for:** هرگاه تعداد دفعات تکرار حلقه مشخص باشد، بهتر است از دستور for با ساختار داده شده در شکل ۳-۲۲ استفاده کنیم. مثال ۱۸: برای مدار شکل ۳-۱۵ با استفاده از حلقه for برنامه‌ای بنویسید که LEDها را پنج بار روشن و خاموش کند. در جدول ۳-۲۶ برنامه داده شده است. این برنامه را بررسی و درباره آن بحث کنید.

### جدول ۳-۲۶

<pre>#include &lt;mega.h&gt; #include &lt;delay.h&gt; unsigned char i; void main(void) {   DDRB=0xff;   for(i=0; i&lt;5; i++)     =&gt; ادامه در ستون مقابل</pre>	<pre>{   PORTB=0xFF;   delay_ms(500);   PORTB=0x00;   delay_ms(500); } while(1); }</pre>
---	--



شکل ۳-۲۲

برای مدار شکل ۳-۱۹ با استفاده از دستور for برنامه‌ای بنویسید که بر روی PORTB یک شمارنده باینری بالا شمار از صفر تا ۹ داشته باشیم.

فعالیت



## ۸-۳- تابع (Function)

■ ساختار زبان C بر پایه توابع است. کاربر می تواند هر بخش از برنامه را به صورت یک تابع بنویسد و زمانی که لازم بود آن را فراخوانی و اجرا نماید. نوشتن برنامه به صورت توابع دارای مزایایی به شرح زیر است:

✓ خواندن برنامه و درک آن آسان تر است.

✓ رفع اشکال از برنامه ساده تر است.

✓ اصلاح و ارتقای برنامه ساده تر صورت می گیرد.

✓ می توان از توابع نوشته شده در برنامه های دیگر هم استفاده کرد.

■ بسته به این که تابع ورودی یا خروجی داشته باشد چهار حالت زیر به وجود می آید:

بدون ورودی - بدون خروجی      void f1 (void)

با ورودی - بدون خروجی      void f2 (unsigned char x)

بدون ورودی - با خروجی      char f3 (void)

با ورودی - با خروجی      int f4 (float y)

■ اگر تابعی دارای چند ورودی از یک نوع باشد باید مانند نمونه زیر تک تک آنها معرفی شوند:

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

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

return مقدار ثابت      یا      return نام متغیر

نکته



### ■ انواع توابع در زبان C

✓ توابع کتابخانه ای استاندارد (standard library function)

✓ توابع تعریف شده توسط کاربر (user defined function)

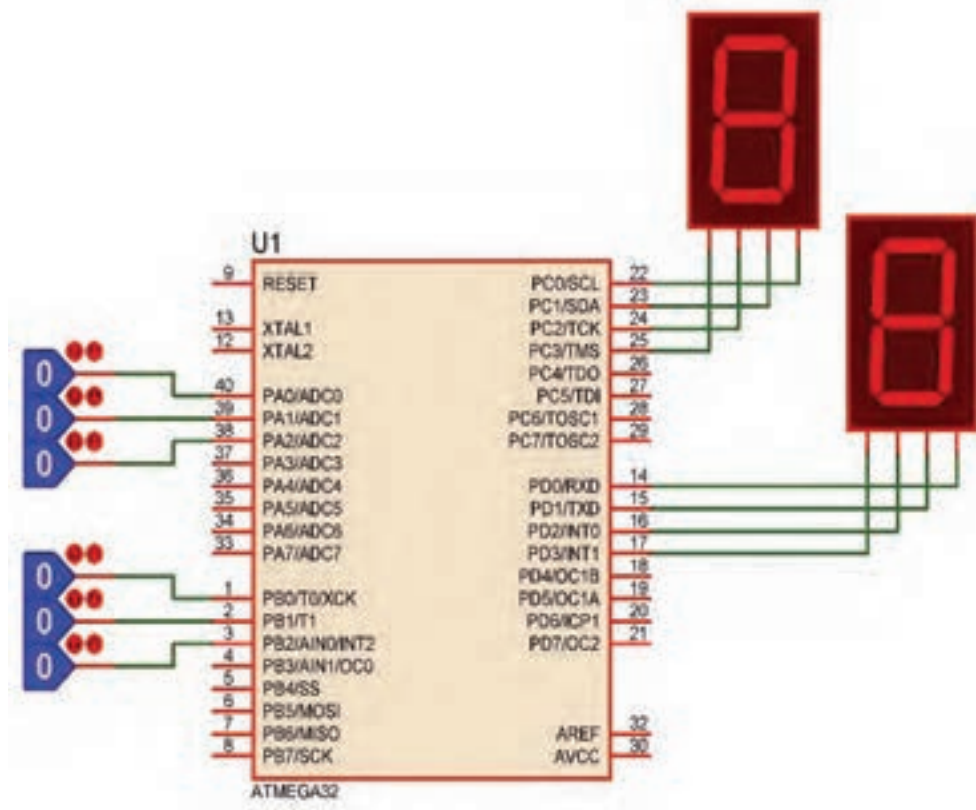
برخی از توابع پر کاربرد که در اغلب برنامه ها مورد استفاده قرار می گیرند، از قبل نوشته شده و در فایل هایی با پسوند h هنگام نصب برنامه در کنار بقیه فایل های کامپایلر C ذخیره می شوند. این توابع را توابع کتابخانه ای می نامند. برای مثال توابع ریاضی مانند sin() و cos() در کتابخانه math.h قرار دارند. فایل های دیگری نیز وجود دارند که در آرشیوهای دیگر جمع می شوند. و در صورت لزوم می توانیم از آنها استفاده کنیم.

### ■ ایجاد تابع جدید

□ محل قرار گرفتن تابع می تواند به یکی از شکل های نشان داده شده در جدول ۳-۲۷ باشد.

اگر توابع مستقل باشند، و از داخل یک تابع، تابع دیگری فراخوانی نشود، انتخاب شکل (الف) یا (ب) تفاوتی ندارد. در غیر این صورت باید روش الف را استفاده کنید.

مثال ۱۹: برنامه ای بنویسید که مانند شکل ۳-۲۳ دو عدد سه بیتی را از PORTA و PORTB دریافت کند، سپس حاصل جمع آنها را روی PORTC و حاصل تفریق آنها را روی PORTD نمایش دهد. این برنامه را یک بار بدون استفاده از تابع و یک بار با استفاده از تابع بنویسید. این مثال را در نرم افزار پروتئوس انجام دهید. برای ورود اعداد باینری از کتابخانه Debugging Tools ابزار Logicstate شکل الف- ۳-۲۴ و برای نمایش اعداد از کتابخانه Optoelectronics قطعه SEG - BCD ۷، شکل ب- ۳-۲۴ را بردارید. در جدول ۳-۲۸ برنامه داده شده است. این برنامه را بررسی کنید و فرآیند اجرای آن را با دقت بیاموزید و درباره آن بحث کنید.



شکل ۲-۲۳

جدول ۳-۲۷ محل قرارگیری تابع

جدول ۳-۲۸

#include <mega32.h>	{	
unsigned char a,b,c,d;	a=PINA & 0x0F;	۱
void main (void)	b=PINB & 0x0F;	۲
{	c=a+b;	
DDRA=0x00;	d=a-b;	
DDRB=0x00;	PORTC=c;	
DDRC=0xff;	PORTD=d;	
DDRD=0xff;	}	
While (1)	}	
⇒ ادامه در ستون مقابل		

ب	الف
بدنه تابع	; معرفی تابع
{	void main (void)
.....	{
.....	.....
}	.....
	فراخوانی تابع
	.....
void main (void)	}
{	.....
.....	بدنه تابع
.....	{
فراخوانی تابع	.....
.....	.....
}	}



ب - انتخاب BCD\_VSEG

الف - انتخاب LOGICSTATE

شکل ۲۴-۳

- در خطوط مشخص شده با شماره‌های ۱ و ۲ به عمل انجام شده ماسک (Mask) کردن می‌گویند با این کار فقط سه بیت اول پورت خوانده شده و بقیه بیت‌ها صفر می‌شوند. (در پودمان سوم به این نکته اشاره می‌شود که  $A.0=0$  و  $A.1=A$ )
- حال طبق جدول ۲۹-۳ هر قطعه از برنامه را به صورت یک تابع مستقل می‌نویسیم و به ترتیب فراخوانی می‌کنیم.

جدول ۲۹-۳- مثال برای تابع همراه با توضیح هر بخش

#include <mega32. h>	معرفی میکرو MEGA۳۲
<pre>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);</pre>	معرفی توابع نوشته شده در برنامه و در صورت وجود، همراه با مشخص کردن نوع ورودی و خروجی آنها، توابعی که ورودی یا خروجی ندارند با کلمه void مشخص شده‌اند.
<pre>void main (void) {   init_port();   while (1)   {     input();     c=sum(a,b);     d=sub(a,b);     output(c,d);   } }</pre>	تابع اصلی برنامه که توابع در هنگام لزوم در آن فراخوانی شده‌اند.
<pre>void init_port (void) {   DDRA=0x00;   DDRB=0x00;   DDRC=0xff;   DDRD=0xff; }</pre>	بدنه تابع init_port که در برنامه اصلی یک بار فراخوانی شده و با اجرای آن پورت‌ها به صورت ورودی یا خروجی پیکربندی شده‌اند. init مخفف کلمه Initialization به معنی مقدار دهی اولیه می‌باشد. این تابع، ورودی و خروجی ندارد.



### ادامه جدول ۲۹-۳

<pre>void input (void) { a=PINA &amp; 0x0f; b=PINB &amp; 0x0f; }</pre>	<p>بدنه تابع input که هر بار فراخوانی شود دو عدد از پورت‌های A و B را خوانده و بعد از ماسک کردن، آنها را در متغیرهای a و b قرار می‌دهد. این تابع، ورودی و خروجی ندارد.</p>
<pre>unsigned char sum (unsigned char x, unsigned char y) { unsigned char z; z=x+y; return z; }</pre>	<p>بدنه تابع sum که هر بار فراخوانی شود دو عدد ورودی خود را با هم جمع و نتیجه را که در Z قرار دارد با استفاده از دستور return به برنامه اصلی برمی‌گرداند. این تابع، هم ورودی و هم خروجی دارد.</p>
<pre>unsigned char sub (unsigned char x, unsigned char y) { unsigned char z; z=x-y; return z; }</pre>	<p>بدنه تابع sub که هر بار فراخوانی شود دو عدد ورودی خود را از هم تفریق و نتیجه را که در Z قرار دارد با استفاده از دستور return به برنامه اصلی برمی‌گرداند. این تابع، هم ورودی و هم خروجی دارد.</p>
<pre>void output (unsigned char x, unsigned char y) { PORTC=x; PORTD=y; }</pre>	<p>بدنه تابع output که هر بار فراخوانی شود دو عدد ورودی خود را به پورت‌های C و D ارسال می‌کند. این تابع ورودی دارد ولی خروجی ندارد.</p>

فعالیت



فعالیت‌های زیر را در محیط پروتئوس و با میکرو ATMEGA۳۲، یک بار بدون تابع و یک بار با تابع انجام دهید.

- یک عدد چهار بیتی را از پورت A دریافت کنید. اگر عدد ورودی زوج بود، روی پورت C عدد ۲ و اگر فرد بود عدد ۳ را نمایش دهید.
- یک عدد هشت بیتی را از پورت A دریافت و تعداد یک‌های آن را روی پورت C و تعداد صفرهای آن را روی پورت D نمایش دهید.

### الگوی آزمون نرم‌افزاری واحد یادگیری ۳

مشابه یکی از مثال‌ها یا فعالیت‌های داده شده در متن واحد یادگیری را به صورت نرم‌افزاری اجرا کنید.

#### الگوی آزمون نظری واحد یادگیری ۳

- ۱- فلوجارت برنامه‌ای را بنویسید که بتواند از رابطه  $y=(x+1)^2$  جذر بگیرد و مقدار y را به ازاء مقادیر بزرگ‌تر از یک محاسبه کند.
- ۲- اصلاحات Variable، Fuction و main را تعریف کنید.
- ۳- هر یک از دستورات زیر را تفسیر کنید.

unsigned char a = ۶ ;

int b,c,d = ۱۰۰۰ ;

float pi = ۰,۲۸;

char s[] = "ALI";

۴- هریک از دستورات; PORTD =12; و PORTD =0b11000; چه مفهومی دارند.

۵- دستور a++ نشانه ..... و دستور a-- نشانه ..... است.

۶- در دستور شرطی if برای اجرای چند دستور، دستورها همواره باید در بین دو آکولاد قرار گیرد.

درست  نادرست

۷- در ویزارد، پول آپ داخلی چه کاربردی دارد؟ شرح دهید. چنانچه فعال نشود چه اشکالی برای برنامه پدید می آورد؟

### ارزشیابی واحد یادگیری ۳: کسب شایستگی در طراحی الگوریتم (فلوچارت) مدار پروژه ساده الکترونیکی

#### شرح کار:

۱- طراحی الگوریتم و فلوچارت ۲- عملگرها و کاربرد آن در برنامه C ۳- نوشتن چند برنامه ساده با دستورهای حلقه، شرطی و حلقه شرطی در زبان C ۴- تشریح توابع و توابع کتابخانه‌ای استاندارد و استفاده از آنها در نوشتن برنامه ساده به زبان C ۵- استفاده از شبیه‌سازی در نرم‌افزار در برنامه C

استاندارد عملکرد: ترسیم اجرای فلوچارت و نوشتن چند برنامه ساده با انواع دستورات اجرای آن به صورت نرم‌افزاری

#### شاخص‌ها:

انتخاب فضای مورد نظر و مناسب بودن آن (میز کار)  
تدوین و طراحی الگوریتم و فلوچارت (۲۰ دقیقه)  
استفاده از عملگر در برنامه‌های ساده به زبان C (۲۰ دقیقه)  
نوشتن برنامه با دستورهای مختلف (۲۵ دقیقه)  
استفاده از توابع کتابخانه در برنامه C (۱۵ دقیقه)  
اجرای برنامه در نرم‌افزار (۳۰ دقیقه)

شرایط انجام کار و ابزار و تجهیزات: شرایط انجام کار مشابه بقیه واحدهای یادگیری

#### معیار شایستگی:

ردیف	مراحل کار	حداقل نمره قبولی از ۳	نمره هنرجو
۱	طراحی الگوریتم، فلوچارت با استفاده از عملگرها	۱	
۲	نوشتن برنامه ساده، تبدیل مبناها و تحلیل و اجرای آن با نرم‌افزار	۲	
۳	نوشتن برنامه ساده با دستورهای مختلف و اجرای آن با نرم‌افزار	۲	
	شایستگی‌های غیر فنی، ایمنی، بهداشت، توجهات زیست محیطی و نگرش: ۱- رعایت نکات ایمنی دستگاه‌ها ۲- دقت و تمرکز در اجرای کار ۳- شایستگی تفکر و یادگیری مادام‌العمر ۴- اخلاق حرفه‌ای	۲	
	میانگین نمرات		*

\* حداقل میانگین نمرات هنرجو برای قبولی و کسب شایستگی، ۲ می‌باشد.

## واحد یادگیری ۴

### کسب شایستگی در برنامه‌نویسی به زبان C (یا هر زبان به روز دیگر) و تحلیل برنامه‌های آماده پروژه‌های الکترونیکی

#### آیا تا به حال فکر کرده‌اید:

- برنامه‌نویسی مدارهای الکترونیکی چگونه انجام می‌شود؟
- چگونه می‌توان یک بارگراف (نمودار میله‌ای) را با استفاده از میکروکنترلر به نمایش درآورد؟
- برنامه‌نویسی برای نمایش گر LCD چگونه انجام می‌شود؟
- کد اسکی ASCII چیست و چه کاربردی دارد؟
- برای صفحه کلیدها از چه کدهایی استفاده می‌شود؟
- برنامه‌نویسی صفحه کلیدها چگونه انجام می‌شود؟
- برای اجرای پروژه‌های الکترونیکی برنامه‌های آماده وجود دارد؟

دنیای برنامه‌نویسی به قدری گسترده است که هرگز نمی‌توان برای آن انتهایی در نظر گرفت. امروزه تمام دستگاه‌های الکترونیکی الکترونیکی خانگی، اداری، تجاری یا صنعتی که دارای کنترل‌هایی مانند دما، سرعت، نور و حرکت هستند از میکروکنترلر و برنامه‌های آن استفاده می‌کنند. کمی به دستگاه‌هایی که در اطراف خود می‌بینید توجه کنید. تقریباً در تمام آنها میکروکنترلر به کار گرفته شده است، ساده‌ترین این نوع دستگاه‌ها، دستگاه چای‌ساز است که برای کنترلر دما طی سه مرحله برنامه‌ریزی شده است، زمانی که آب به جوش می‌آید و زمانی که باید جوشیدن آب متوقف شود و دما در محدوده ۹۰ درجه ثابت بماند. در این واحد یادگیری بررسی نظری و نرم‌افزاری چند نمونه از پروژه‌های الکترونیکی که در آنها از برنامه‌نویسی و میکروکنترلر استفاده کرده‌اند می‌پردازیم.

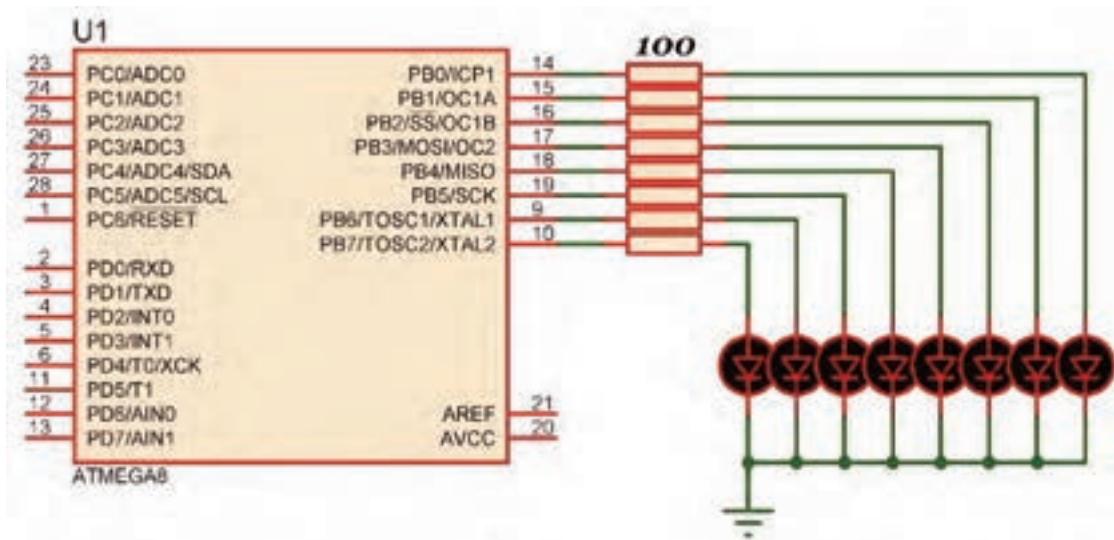
#### استاندارد عملکرد

برنامه‌نویسی به زبان C (یا هر زبان به روز دیگر) برای پروژه‌های عملی و تحلیل برنامه‌های سامانه‌های کنترلی

#### ۴-۱- برنامه‌نویسی برای مدارهای الکترونیکی با LED

یکی از خروجی‌های پرکاربرد، ساده و متنوع از نظر شکل و رنگ LED است. در این بخش چند پروژه ساده را با این قطعه انجام می‌دهیم.

مثال ۱: هشت عدد LED مانند شکل ۴-۱ به پورت B متصل کنید و یک چشمک زن بسازید. برنامه این مدار در جدول ۴-۱ آمده است.



شکل ۴-۱- مدار چشمک‌زن

#### جدول ۴-۱

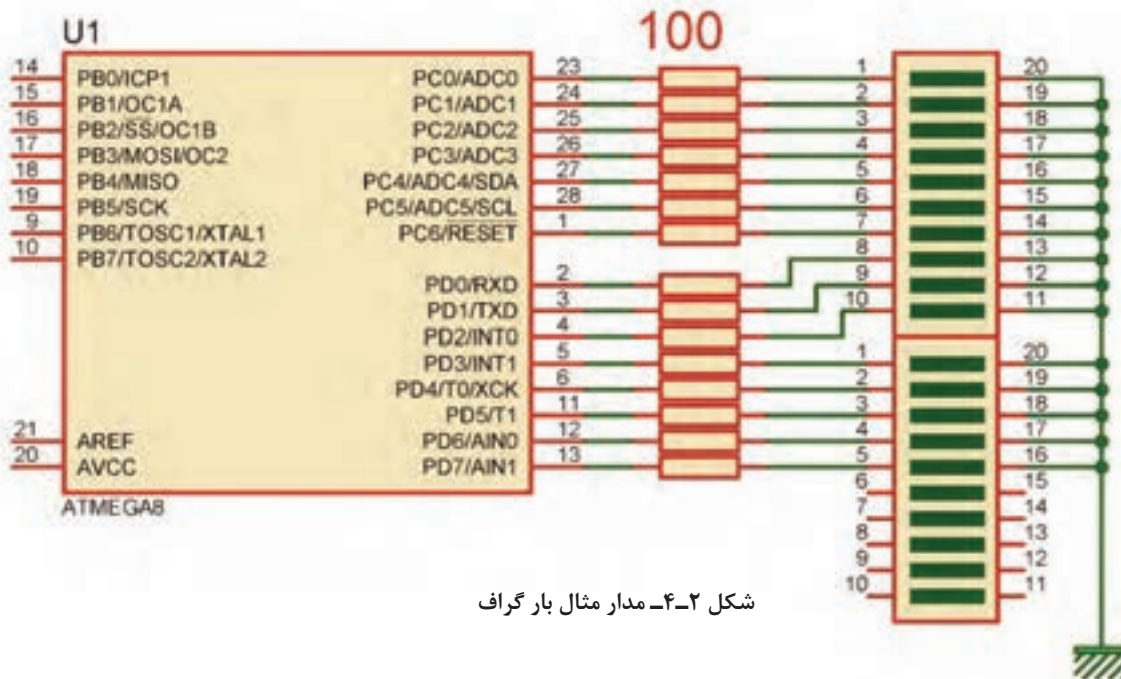
```
#include < mega8. h > // معرفی کتابخانه میکروکنترلر
#include < delay. h > // معرفی کتابخانه تأخیر
void main (void) // شروع تابع اصلی
{
  DDRB=0xff; // قرار دادن پورت در حالت خروجی
  while(1) // شروع حلقه while
  {
    PORTB=0b 11000011; // روشن کردن چهار بیت کناری و خاموش کردن چهار بیت وسط
    delay_ms(500); // ایجاد تاخیر به اندازه نیم ثانیه
    PORTB=0b 01111000; // روشن کردن چهار بیت وسط و خاموش کردن چهار بیت کناری
    delay_ms(500); // ایجاد تاخیر به اندازه نیم ثانیه
  } // while حلقه
} // انتهای تابع اصلی
```

در ساعت‌های غیردرسی، برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی کنید. نتیجه را در قالب یک گزارش ارائه دهید.

فعالیت



مثال ۱: با اتصال پانزده عدد LED به میکروکنترلر مانند شکل ۴-۲ یک نمودار میله‌ای (bar graph) بسازید.



شکل ۲-۴- مدار مثال بار گراف

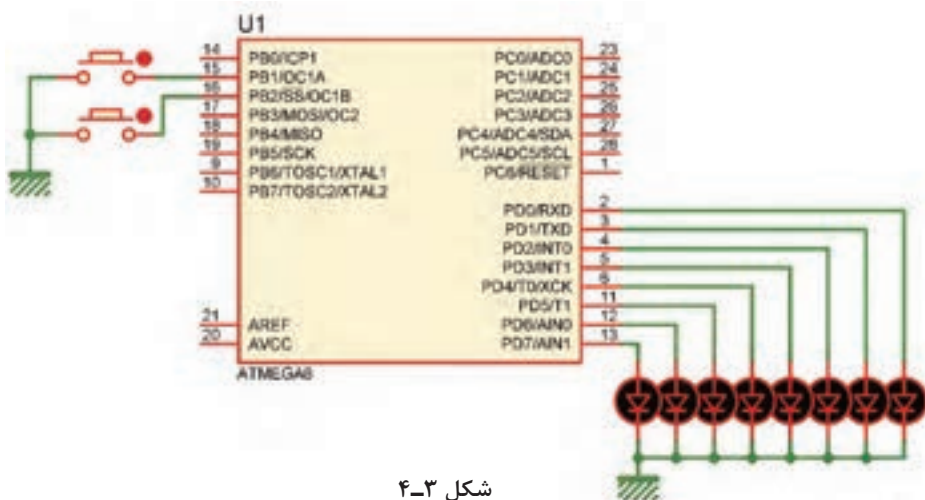
## جدول ۲-۴

```
#include <mega8.h > // معرفی کتابخانه میکروکنترلر
#include <delay.h > // معرفی کتابخانه تاخیر
signed char i; // تعریف متغیر علامت‌دار
void main(void) // شروع تابع اصلی
{
    DDRC=0xff; // قرار دادن پورت در حالت خروجی
    DDRD=0xff; // قرار دادن پورت در حالت خروجی
    while(1) // شروع حلقه while
    { // ----- LED ON -----
        for(i=7;i>=0;i--) // شروع حلقه for
        {
            روشن کردن با استفاده از شیفت و عملگر xor
            PORTD=PORTD^(1<<i); //
            delay_ms(100); // ایجاد تأخیر
        }
        for(i=6;i>=0;i--) // شروع حلقه for
        {
            // ادامه برنامه در ستون مقابل از بالا به پایین
        }
    }
}
```

```
روشن کردن با استفاده از شیفت و عملگر xor
PORTC=PORTC^(1<<i); //
delay_ms(100); // ایجاد تاخیر
}
//----- LED OFF -----
for(i=0;i<7;i++) // شروع حلقه for
{
    خاموش کردن با استفاده از شیفت و عملگر xor
    PORTC=PORTC^(1<<i); //
    delay_ms(100); // ایجاد تاخیر
}
for(i=0;i<8;i++) // شروع حلقه for
{
    خاموش کردن با استفاده از شیفت و عملگر xor
    PORTD=PORTD^(1<<i); //
    delay_ms(100); // ایجاد تاخیر
}
} // while حلقه
} // انتهای تابع اصلی
```

## ۴-۲- استفاده از کلید (KEY)

یکی از ساده ترین و پرکاربردترین قطعات ورودی کلید فشاری است که در اکثر دستگاه‌های الکترونیکی مانند ریموت کنترل تلویزیون، کنترل دمای یخچال و کولر و مانند آنها به کار می‌رود. مثال ۳: می‌خواهیم با استفاده از دو کلید، عدد باینری خروجی روی LEDهای متصل به پورت D را کم و زیاد کنیم. هشت عدد LED به PORTD و دو عدد کلید به پین‌های ۱ و ۲ PINB متصل شده است، شکل ۳-۴. برنامه‌ای بنویسید که با هر بار زدن کلید ۱ PINB عدد روی PORTD افزایش و با هر بار زدن کلید ۲ PINB عدد روی PORTD کاهش یابد. (ورودی کلیدها pull\_up داخلی شوند)



شکل ۳-۴

■ ابتدا فلوجارت برنامه را به صورت شکل ۴-۴ رسم می‌کنیم:

■ برنامه را طبق جدول ۴-۳ می‌نویسیم و پس از بارگذاری راه‌اندازی می‌کنیم.

جدول ۴-۳



شکل ۴-۴- فلوجارت خواندن کلید و شمارش اعداد

<pre>#include &lt; mega8. h &gt; #include &lt; delay. h &gt; unsigned char i=۰; void main (void) {   DDRB=۰x۰۰;   DDRD=۰xff;   while (۱)   {     if (PINB.۱==۰) // کلید اول     {       delay_ms(۲۵);       i++;     }     if (PINB.۲==۰) // کلید دوم     {       delay_ms(۲۵);       i--;     }   }   PORTD=i;   while (PINB.۲==۰);   delay_ms(۲۵); } }</pre>	<pre>PORTD=i; while (PINB.۱== ۰); delay_ms(۲۵); } if (PINB.۲==۰) // کلید دوم {   delay_ms(۲۵);   i--; } PORTD=i; while (PINB.۲== ۰); delay_ms(۲۵); } }</pre>
--	--

ادامه برنامه در ستون مقابل از  
⇒ بالا به پایین

در ساعت‌های غیردرسی، برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی کنید. نتیجه را در قالب یک گزارش ارائه دهید.

فعالیت



### ۴-۳- راه‌اندازی LCD



شکل ۴-۵- LCD کاراکتری ۲×۱۶

یکی دیگر از دستگاه‌های خروجی، LCD کاراکتری است. روی LCD می‌توان متن، اعداد و علائم را نمایش داد. در LCD کاراکتری پارامترهای مهم، تعداد خط و تعداد کاراکترها در هر خط است. نوع ۲×۱۶ آن کاربرد بیشتری در مقایسه با سایر LCDها دارد.

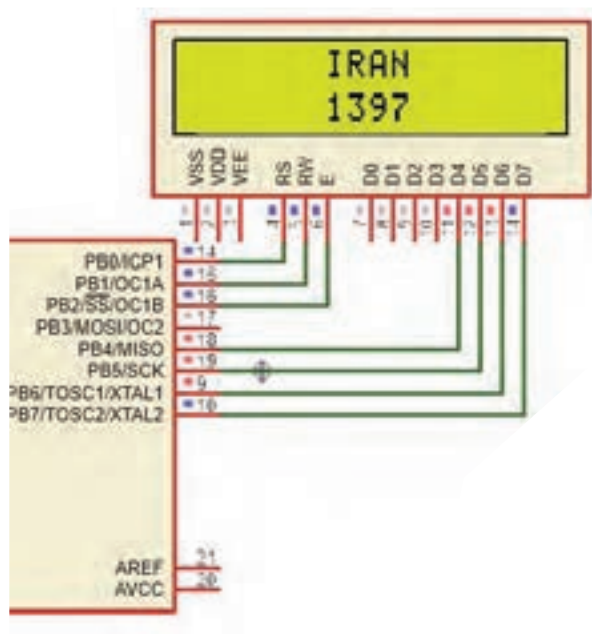
■ LCDهای کاراکتری را می‌توان با ۴ خط دیتا راه‌اندازی کرد (خطوط D۴ تا D۷). برای مثال در شکل ۴-۶- طرز اتصال یک LCD به PORTB با ۴ خط دیتا نشان داده شده است. ■ کد ویژن از این نوع اتصال پشتیبانی می‌کند.

نکته

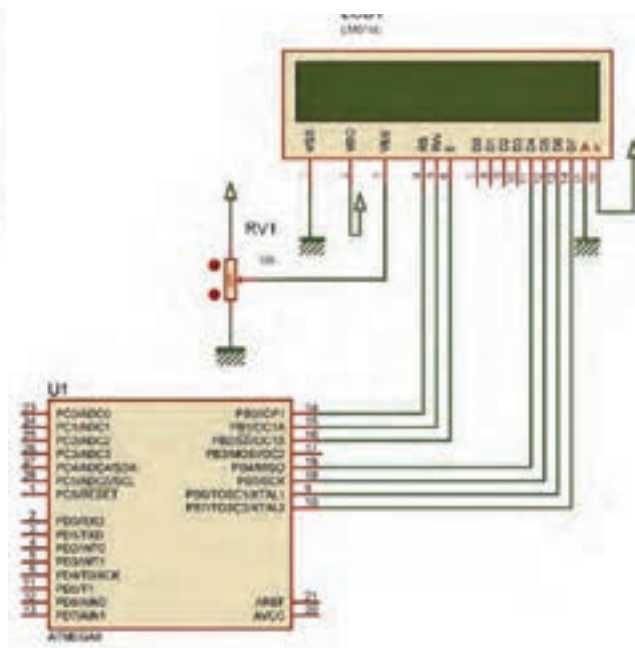


مثال ۳: یک LCD را مانند شکل ۴-۶ به پورت B متصل کنید.

مانند شکل ۴-۷ برنامه‌ای بنویسید که وسط خط اول کلمه IRAN و وسط خط دوم ۱۳۹۷ نوشته شود.



شکل ۴-۷



شکل ۴-۶- نحوه اتصال LCD به میکروکنترلر

### ■ مراحل اجرای کار

ابتدا در ویزارد برگه Alphanumeric LCD را باز و آن را فعال می‌کنیم.  
 □ در قسمت Character/Line مشخص می‌کنیم که در هر خط چند کاراکتر وجود دارد.

جدول ۴-۴

	X=۰	X=۱	.....	X=۱۵	ستون‌های LCD
سطرهای LCD	Y=۰		.....		
	Y=۱		.....		

□ در قسمت Connections مانند شکل ۴-۸ چگونگی اتصال LCD به میکروکنترلر مشخص شده و قابل ویرایش است.

□ پس از تولید کد، کتابخانه <alcd.h> به برنامه اضافه می‌شود که توابع کار با LCD در این کتابخانه قرار دارد. این توابع عبارت‌اند از:

lcd\_clear(); پاک کردن صفحه نمایش

lcd\_gotoxy (x,y); مشخص کردن ستون و سطری که نوشتن از آنجا شروع می‌شود

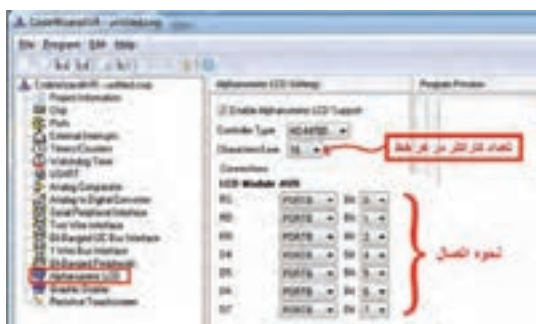
□ در جدول ۴-۴ شماره سطرها و ستون‌های LCD نشان داده شده است.

□ برای نوشتن متن روی LCD می‌توانید یکی از توابع موجود در جدول ۴-۵ را به کار ببرید.

□ کدهای مندرج در جدول ۴-۶ زیر را به کدهای تولید شده توسط ویزارد اضافه کنید.

مثال ۴: برنامه‌ای بنویسید که کلمه IRAN را از ابتدای خط اول LCD تا انتهای خط اول حرکت دهد.

□ کدهای وارد شده در جدول ۴-۷ را به کدهای تولید شده توسط ویزارد، اضافه کنید.



شکل ۴-۸- تنظیم ویزارد برای اتصال LCD به میکروکنترلر

جدول ۴-۵ توابع نوشتن بر روی LCD همراه با مثال

مثال	تابع
lcd_putchar('A');	lcd_putchar('ک کاراکتر');
lcd_putsf("ARASH");	lcd_putsf("رشته");
char s[]="IRAN"; lcd_puts(s);	lcd_puts(متغیر رشته‌ای);

جدول ۴-۶	جدول ۴-۷
<pre>char s[]="IRAN"; void main(void) {     lcd_init(۱۶);     lcd_clear();     lcd_gotoxy(۶,۰);     lcd_puts(s);     lcd_gotoxy(۶,۱);     lcd_putsf("۱۳۹۷");     while (۱)     {         } }</pre>	<pre>#include &lt;delay.h&gt; unsigned char x; void main(void) {     while (۱)     {         for(x=۰;x&lt;۱۳;x++)         {             lcd_clear();             lcd_gotoxy(x,۰);             lcd_putsf("IRAN");             delay_ms (۳۰۰);         }     } }</pre>



### ■ کد اسکی (ASCII): American Standard Code for Information Interchange

☑ کدهای اسکی توسط انجمن استاندارد آمریکا برای تبادل اطلاعات در سیستم‌های کامپیوتری و مخابراتی به وجود آمد. این کدها به صورت استاندارد برای همه دستگاه‌های ارتباطی در نظر گرفته شده‌اند تا تمامی سیستم‌های مخابراتی از استاندارد یکسانی جهت ارتباط با یکدیگر تبعیت کنند.

☑ ASCII کاراکترهای مورد استفاده برای ارسال و دریافت اطلاعات را به صورت کدهایی تعریف می‌کند که برای همه به صورت یکسان از قبل تعریف شده است. مثلاً حرف A با کد اسکی ۶۵ به صورت یک مشخصه استاندارد تعریف شده است، حال اگر سیستمی این کد را دریافت و یا ارسال کند همواره این کد برای سامانه‌های مختلف به عنوان حرف A تلقی خواهد شد.

☑ در LCDهای کاراکتری نیز از کد ASCII استفاده می‌شود. در ادامه، کدهای ASCII برخی کاراکترهای ASCII را ملاحظه می‌کنید:

A=۶۵      a=۹۷      c=۴۰      +=۴۳      @=۶۴      ?=۲۳

☑ در جدول ۸-۴ کدهای اسکی مربوط به کاراکترها را ملاحظه می‌کنید.

جدول ۸-۴- کدهای اسکی

Character	Binary Code	Character	Binary Code	Character	Binary Code
A	01000001	a	01100001	!	00100001
B	01000010	b	01100010	"	00100010
C	01000011	c	01100011	#	00100011
D	01000100	d	01100100	\$	00100100
E	01000101	e	01100101	%	00100101
F	01000110	f	01100110	&	00100110
G	01000111	g	01100111	'	00100111
H	01001000	h	01101000	(	00101000
I	01001001	i	01101001	)	00101001
J	01001010	j	01101010	*	00101010
K	01001011	k	01101011	+	00101011
L	01001100	l	01101100	,	00101100
M	01001101	m	01101101	-	00101101
N	01001110	n	01101110	.	00101110
O	01001111	o	01101111	/	00101111
P	01010000	p	01110000	0	00110000
Q	01010001	q	01110001	1	00110001
R	01010010	r	01110010	2	00110010
S	01010011	s	01110011	3	00110011
T	01010100	t	01110100	4	00110100
U	01010101	u	01110101	5	00110101
V	01010110	v	01110110	6	00110110
W	01010111	w	01110111	7	00110111
X	01011000	x	01111000	8	00111000
Y	01011001	y	01111001	9	00111001
Z	01011010	z	01111010	?	00111111
				@	01000000

✓ برای ارسال اعداد به LCD لازم است که ابتدا آنها را توسط توابع موجود به کد ASCII تبدیل کرده و سپس ارسال نماییم. برای این کار توابعی وجود دارد که یکی از آنها تابع `sprintf` موجود در کتابخانه `<stdio.h>` است. ساختار این تابع مانند شکل ۴-۹ است.

(متغیر عددی، "متن و نوع متغیر ها ونحوه چاپ"، متغیر رشته‌ای) `sprintf`

شکل ۴-۹- ساختار تابع `sprintf`

جدول ۴-۹

تعریف متغیر رشته‌ای	مثال
[طول رشته] نام رشته <code>char</code>	<code>char s[۶];</code>

✓ برای تعریف متغیر رشته‌ای طبق جدول ۴-۹ عمل کنید.  
 ✓ برای مشخص کردن نوع و چگونگی چاپ اعداد روی LCD از علامت `%` به همراه کاراکترهای خاص استفاده می‌شود. مثلاً از علامت `%d` یا `%i` برای مشخص کردن اعداد صحیح و از `%f` برای اعداد اعشاری استفاده می‌شود.

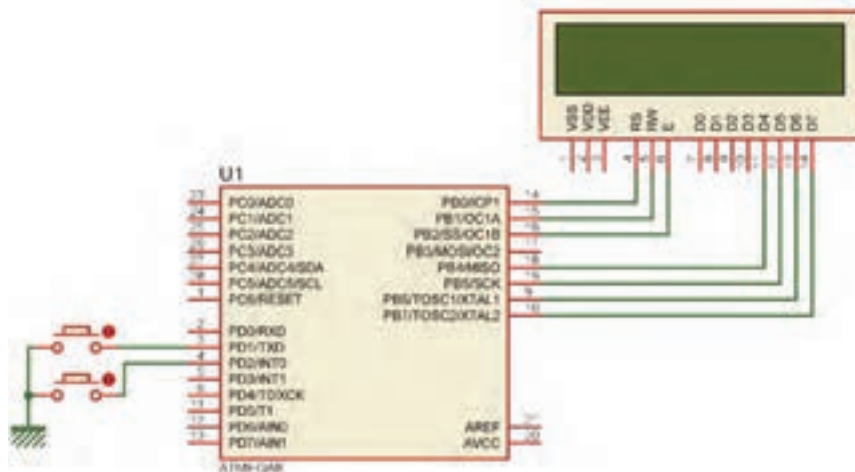
جدول ۴-۱۰

```
i=۳;
char s[۶];
sprintf (s , "%۰۲d" , i);
lcd_puts(s);
```

✓ برای مشخص کردن چگونگی چاپ، علاوه بر نوع متغیر، از اعداد نیز در کنار آنها استفاده می‌شود، مثلاً `%۰۲d` یعنی این که عدد به صورت دو رقمی چاپ شود حال اگر عدد یک رقمی باشد با قرار دادن یک `۰` قبل از عدد، آن را در دو رقم چاپ می‌کند (نمایش می‌دهد). برای مثال اگر مقدار متغیر `۳` باشد روی LCD عدد `۰۳` دیده خواهد شد.

✓ در جدول ۴-۱۰ یک نمونه برنامه مربوط به چاپ یا نمایش اعداد روی LCD را ملاحظه می‌کنید.

**مثال ۵:** دو عدد کلید و یک LCD را مانند شکل ۴-۱۰ به میکروکنترلر متصل کنید، سپس برنامه‌ای بنویسید که با نگه داشتن یکی از کلیدها، عدد روی LCD زیاد و با کلید دیگر کاهش یابد و بازه تغییر عدد صفر تا `۲۰` باشد.



شکل ۴-۱۰

پاسخ: در جدول ۴-۱۱ برنامه مربوط به مثال ۵ آمده است.

### جدول ۴-۱۱

<pre>#include &lt;mega.h&gt; #include &lt;alcd.h&gt; #include &lt;delay.h&gt; #include &lt;stdio.h&gt; unsigned char i=0; char s[۴]; void main(void) {   lcd_init(۱۶);   lcd_gotoxy(۷,۰);   lcd_putsf("۰۰");   while (۱)   {     ادامه برنامه در ستون مقابل از بالا     =&gt; به پایین</pre>	<pre>if(PIND.۱==۰ &amp;&amp; i&lt;۲۰) // باشد ۲۰ کوچک‌تر از i {   delay_ms(۲۵);   i++;   sprintf(s,"%۰۲d",i);   lcd_gotoxy(۷,۰);   lcd_puts(s);   delay_ms(۳۰۰); }   اگر کلید دوم زده شده و i بزرگ‌تر از صفر باشد   if(PIND.۲==۰ &amp;&amp; i&gt;۰)   {     delay_ms(۲۵);     i--;     sprintf(s,"%۰۲d",i);     lcd_gotoxy(۷,۰);     lcd_puts(s);     delay_ms(۳۰۰);   } }</pre>
--	--

برنامه داده شده در جدول ۴-۱۱ را تحلیل کنید و فرایند اجرای آن را در قالب یک گزارش ارائه دهید.

فعالیت



مثال ۶: با توجه به مدار شکل ۴-۱۰ یک ساعت بر روی LCD طراحی کنید.

### جدول ۴-۱۲

<pre>#include &lt;stdio.h&gt; #include &lt;delay.h&gt; unsigned char h,m,s; char time[۱۶]; void main(void) {   lcd_clear();   while (۱)   {     s++;     if(s==۶۰)     {       ادامه برنامه در ستون مقابل از بالا       =&gt; به پایین</pre>	<pre>s=۰; m++; if(m==۶۰) {   m=۰;   h++;   if(h==۲۴) h=۰; } }   sprintf(time,"%۰۲d: %۰۲d: %۰۲d",h,m,s);   lcd_gotoxy(۴,۰);   lcd_puts(time);   delay_ms(۱۰۰۰); }</pre>
--	--

- برنامه این مثال در جدول ۴-۱۲ آمده است.
- در این برنامه با گذشت هر ثانیه به مقدار متغیر S که ثانیه را در خود نگه می دارد یک واحد اضافه می شود.
- هر گاه  $S=60$  شود شرط موجود مقدار S را صفر و به مقدار m که دقیقه را در خود نگه می دارد یک واحد اضافه می کند.
- هر گاه  $m=60$  شود، شرط موجود m را صفر و به مقدار h که ساعت را در خود نگه می دارد یک واحد اضافه می کند.
- هر گاه  $h=24$  شود این متغیر نیز صفر خواهد شد.
- در تابع printf هر سه مقدار ساعت، دقیقه و ثانیه به صورت دو رقمی همراه با جدا کننده ":" به صورت یک رشته در متغیر رشته ای time قرار می گیرد؛ سپس وسط خط اول چاپ می شود.
- دستور `delay_ms(1000)` نیز باعث می شود هر ثانیه یک بار متغیر S زیاد شود.

توجه داشته باشید که برنامه ساعت نوشته شده دقیق نیست و برای داشتن یک ساعت دقیق لازم است از واحد تایمر و یا RTC استفاده شود.

تحقیق کنید و دریابید که RTC چه مفهوم و چه کاربردهایی دارد.

با توجه به مدار شکل ۴-۱۰ یک ساعت بر روی LCD طراحی کنید، که بتوان توسط دو کلید جداگانه، ساعت و دقیقه را تنظیم کرد.

نکته



پژوهش

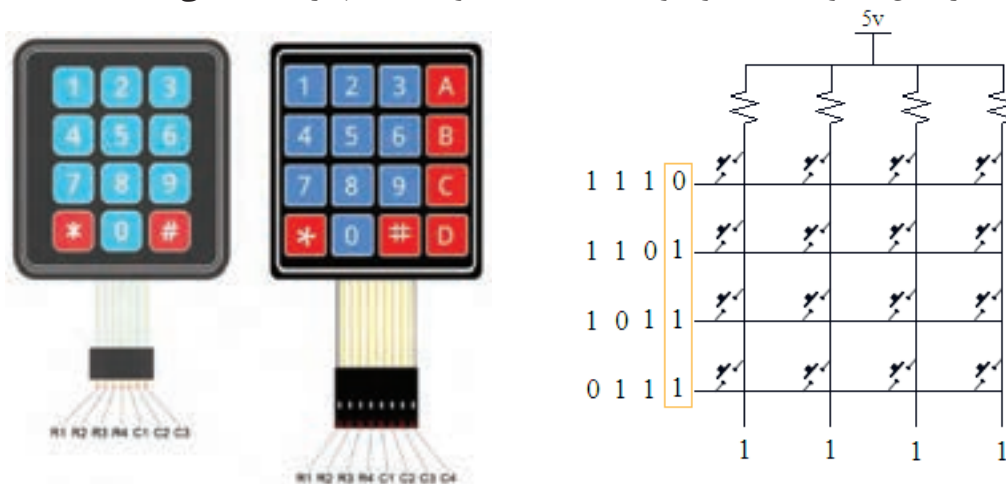


فعالیت



## ۴-۴- صفحه کلید

- اگر در مداری به تعداد زیادی کلید نیاز داشته باشیم و بخواهیم هر یک را به یک پین از پورتها متصل کنیم، تعداد زیادی از پورتها اشغال می شود. برای رفع این مشکل از صفحه کلید که یکی دیگر از وسایل ورودی است استفاده می کنیم. در این دستگاه ورودیها، کلیدها هستند که طبق شکل الف - ۴-۱۱ به صورت ماتریسی چیده می شوند. در شکل الف و ب ۴-۱۱ دو نمونه صفحه کلید با سوکت های آنها را ملاحظه می کنید.



شکل ب ۴-۱۱- دو نمونه صفحه کلید

شکل الف ۴-۱۱- مدار صفحه کلید

### روش خواندن صفحه کلید

ابتدا یک سطر را صفر و بقیه را یک می‌کنیم، سپس ستون‌ها را می‌خوانیم. اگر همه آنها یک باشند، یعنی در آن سطر کلیدی فعال نشده است. آن سطر را یک و سطر بعدی را صفر می‌کنیم و دوباره ستون‌ها را می‌خوانیم. این کار را برای همه سطرها انجام می‌دهیم.

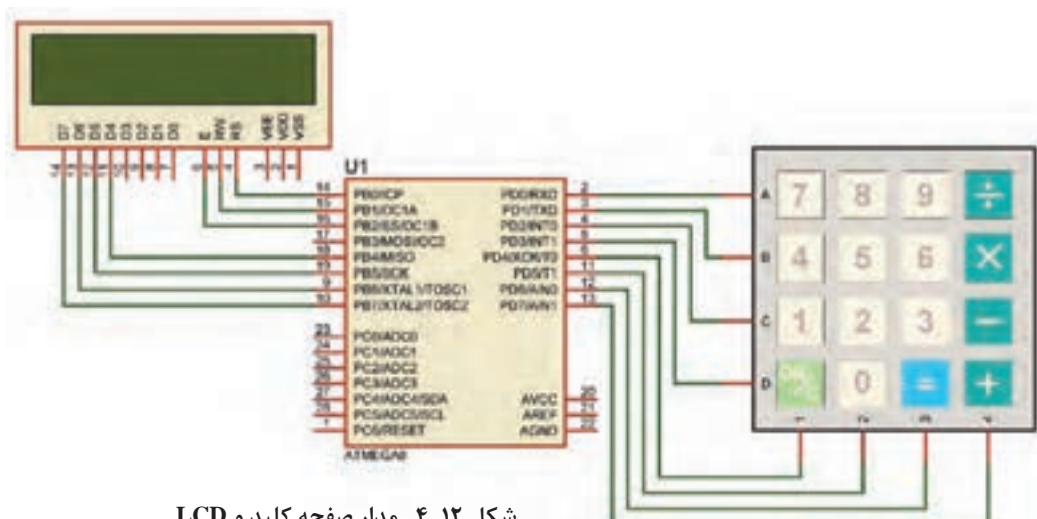
اگر سطری را صفر کردیم و ستونی صفر شد با توجه به سطر و ستونی که صفر شده، کلید فعال مشخص می‌شود. برای مثال اگر سطر دوم را صفر کنیم و ستون سوم صفر شود یعنی، کاربر کلید ۶ را زده است. مثال ۷: یک صفحه کلید ۴×۴ و یک LCD را مطابق شکل ۱۲-۴ به میکروکنترلر متصل کنید. برنامه‌ای بنویسید که با فعال کردن هر کلید، عدد نوشته شده روی کلید در خروجی ظاهر شود. برای کلیدهای غیر عددی یعنی نمادهایی مانند جمع و تفریق، کدهای داده شده در جدول ۱۳-۴ روی نمایشگر به نمایش درآید، چاپ کنید.

جدول ۱۳-۴ نمایش کدهای عددی متناظر برای کاراکترها

÷	×	-	+	=	on/c
۱۰	۱۱	۱۲	۱۳	۱۴	۱۵

توجه داشته باشید که برای ستون‌ها از Pull-Up داخلی استفاده شده است.

نکته



شکل ۱۲-۴ مدار صفحه کلید و LCD

توجه: جهت خوانا تر شدن (کاربر پسندتر شدن User Friend) برنامه، با استفاده از ماکرو برای سطر و ستون‌ها از نام‌های جدید استفاده کرده‌ایم. همچنین برنامه خواندن صفحه کلید را به صورت یک تابع نوشته‌ایم.

توجه



جدول ۴-۱۴ برنامه مربوط به مثال ۷ را نشان می‌دهد.

جدول ۴-۱۴

<pre>#include &lt;mega.h&gt; #include &lt;delay.h&gt; #include &lt;stdio.h&gt; #include &lt;alcd.h&gt; #define R1 PORTD.0 #define R2 PORTD.1 #define R3 PORTD.2 #define R4 PORTD.3 #define C1 PIND.4 #define C2 PIND.5 #define C3 PIND.6 #define C4 PIND.7 unsigned char kb(void); unsigned char key; char s[4]; void main (void) {   DDRD=0xf;   lcd_init(16);   while(1)   {     key=kb();     if (key !=16)     {       sprintf (s,"%d",key);       lcd_puts(s);     }   } } //----- kb function ----- unsigned char kb (void) {   ⇒ ادامه برنامه در ستون مقابل از بالا به پایین</pre>	<pre>unsigned char k=16; PORTD=0xFF; //----- ROW1 //----- R1=0; delay_ms(3); if (C1==0) {k=7 ; while (C1==0);} if (C2==0) {k=8 ; while (C2==0);} if (C3==0) {k=9 ; while (C3==0);} if (C4==0) {k=10 ; while (C4==0);} R1=1; //----- ROW2 //----- R2=0; delay_ms(3); if (C1==0) {k=4 ; while (C1==0);} if (C2==0) {k=5 ; while (C2==0);} if (C3==0) {k=6 ; while (C3==0);} if (C4==0) {k=11 ; while (C4==0);} R2=1; //----- ROW3 //----- R3=0; delay_ms(3); if (C1==0) {k=1 ; while (C1==0);} if (C2==0) {k=2 ; while (C2==0);} if (C3==0) {k=3 ; while (C3==0);} if (C4==0) {k=12 ; while (C4==0);} R3=1; //----- ROW4 //----- R4=0; delay_ms(3); if (C1==0) {k=15 ; while (C1==0);} if (C2==0) {k=0 ; while (C2==0);} if (C3==0) {k=14 ; while (C3==0);} if (C4==0) {k=13 ; while (C4==0);} R4=1; return k; }</pre>
---	--

در ساعت‌های غیر درسی برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی نمایید. نتیجه را در قالب یک گزارش ارائه دهید.

فعالیت



### جدول ۴-۱۵

<pre>while(1) { key=kb(); if (key &lt; ۱۰) { sprintf (s,"%d",key); lcd_puts(s); } }</pre> <p>ادامه برنامه در ستون مقابل ⇒ از بالا به پایین</p>	<pre>if (key==۱۰) lcd_putchar('۰'); if (key==۱۱) lcd_putchar('*'); if (key==۱۲) lcd_putchar('_'); if (key==۱۳) lcd_putchar('+'); if (key==۱۴) lcd_putchar('='); if (key==۱۵) lcd_clear(); }</pre>
--	---

**مثال ۸:** در مثال ۷ برنامه را طوری تغییر دهید که با فعال کردن کلیدهای مربوط به علائم، به جای اعداد ۱۰ تا ۱۴ علامت مشخصه آن کلید مانند تقسیم روی صفحه نمایشگر ظاهر شود و با فعال کردن کلید ON/C صفحه LCD پاک شود.  
پاسخ: در برنامه مثال ۷ در حلقه while تغییرات داده شده در جدول ۴-۱۵ را انجام دهید.

فعالیت



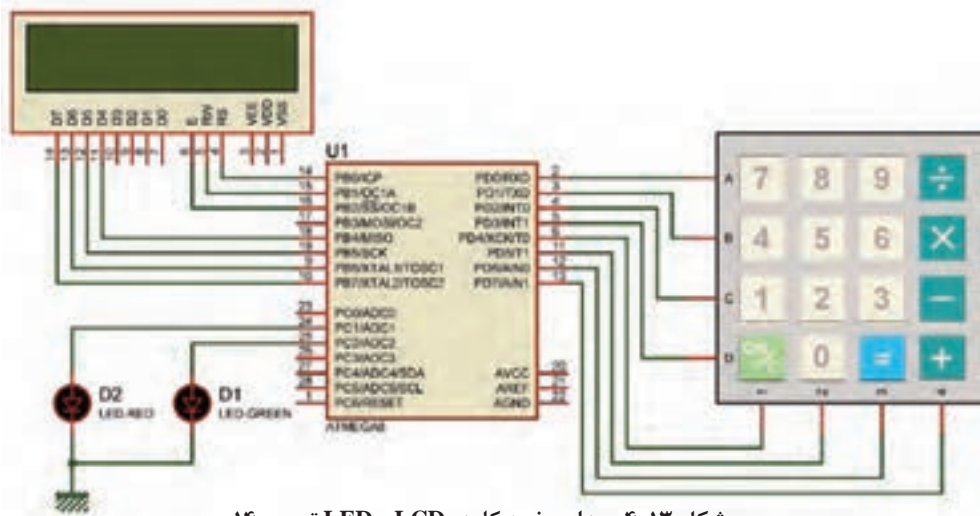
برای مدار شکل ۴-۱۳ برنامه‌ای بنویسید که کاربر بتواند:

- یک رمز چهار رقمی را وارد کند.
  - با فعال کردن کلید مساوی، عدد وارد شده با رمز تعیین شده مقایسه شود.
  - اگر رمز صحیح بود LED سبز (متصل به PC.۲) و اگر رمز اشتباه بود LED قرمز (متصل به PC.۱) روشن و بعد از یک ثانیه خاموش شود.
- راهنمایی:** برای نوشتن برنامه قفل رمز لازم است موارد زیر را اجرا کنید:
- یک آرایه تعریف کنید.
  - شماره کلیدهای انتخاب شده را در داخل آرایه ذخیره کنید.
  - اگر کاربر کلید مساوی = را فشار داد، اعداد ذخیره شده در آرایه با ارقام رمز از پیش تعیین شده مقایسه شود.
  - چنانچه عدد وارد شده با رمز برابر بود، LED ۱ سبز رنگ و در غیر این صورت LED ۲ قرمز رنگ روشن شود.

نکته



تعریف یک آرایه با نام d و تعداد ۱۰ خانه به صورت: Char d[۱۰] بیان می‌شود.



شکل ۴-۱۳- مدار صفحه کلید، LCD و LED تمرین ۱۴

#### جدول ۴-۱۶

```

unsigned char i;
while(1)
{
for(i=۰;i<۱۰;i++)
{
lcd_gotoxy(V,۰);
lcd_puts(i);
delay_ms(۵۰۰);
}
}

```

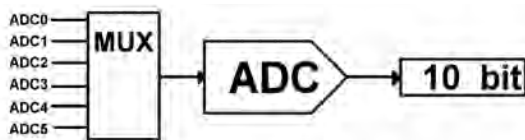
**الگوی پرسش**

۱- اگر در برنامه داده شده در جدول ۴-۱۶ متغیر  $i$  از نوع `unsigned char` تعریف شده باشد، چه خطایی وجود دارد و برای تصحیح آن چه باید کرد. با مراجعه به جدول شرح دهید و جدول را اصلاح کنید.

۲- یک LCD و دو عدد کلید به میکروکنترلر متصل است، برنامه‌ای بنویسید که با فشار دادن کلیدها یک نمودار میله ای افقی بر روی LCD ترسیم شده و کوتاه و بلند شود.

**راهنمایی:** دستور `lcd_putchar(۲۵۵)` باعث چاپ یک مستطیل تو پر ■ بر روی LCD می‌شود.

#### ۴-۵- ADC (Analog to Digital Converter)



شکل ۴-۱۴- واحد ADC

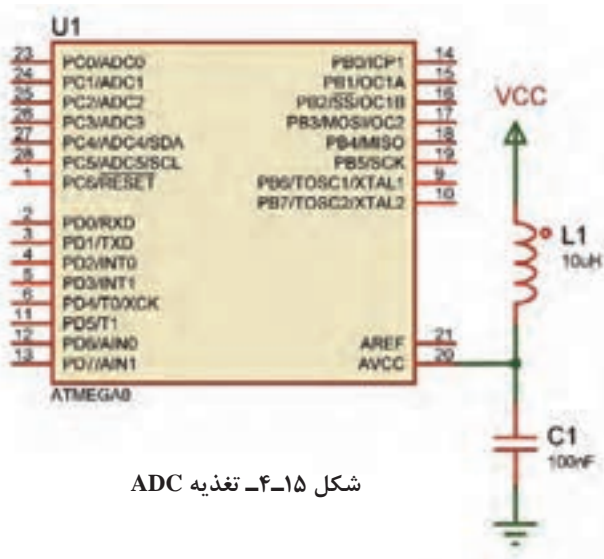
■ تمام کمیت‌های اطراف ما مانند دما، فشار و رطوبت کمیت‌های آنالوگ هستند. برای اندازه‌گیری و پردازش آنها لازم است ابتدا آنها را به یک کمیت دیجیتال تبدیل و سپس پردازش شوند. در ATMEGA8 یک مبدل ۶ کاناله و ۱۰ بیتی پیش‌بینی شده است، شکل

۴-۱۴. توسط Multiplexer یکی از ۶ کانال ورودی انتخاب و کمیت آنالوگ آن کانال به مبدل وارد و در نهایت به یک عدد ۱۰ یا ۸ بیتی تبدیل می‌شود.

■ **تغذیه ADC:** برای بالا بردن دقت ADC و کاهش تأثیر نویزهای احتمالی که ممکن است از ولتاژ تغذیه اصلی به بخش ADC وارد شود، تغذیه این قسمت را از تغذیه سایر مدارهای مرتبط با میکروکنترلر جدا می‌کنند. کاربر می‌تواند تغذیه این بخش را جداگانه تأمین کرده یا با استفاده از یک فیلتر LC مانند شکل ۴-۱۵ به ولتاژ اصلی متصل نماید. برای تغذیه میکروکنترلر و بخش ADC مطابق جدول ۴-۱۷ عمل کنید.

#### جدول ۴-۱۷- پایه‌های تغذیه اصلی و بخش ADC

شماره پایه	عملکرد
۷	تغذیه اصلی VCC
۸	تغذیه اصلی GND
۲۰	تغذیه ADC AVCC
۲۲	تغذیه ADC AGND
۲۱	ولتاژ مرجع خارجی Aref



شکل ۴-۱۵- تغذیه ADC



■ **ولتاژ مرجع:** در این نوع مبدل نیاز به یک ولتاژ مرجع داریم که می‌توانیم آن را از سه طریق تأمین کنیم:  
 ✓ ولتاژ تغذیه روی پایه AVCC علاوه بر تغذیه واحد ADC می‌تواند به عنوان ولتاژ مرجع نیز در نظر گرفته شود.

✓ ولتاژ روی پایه Aref، که می‌تواند بین ۰ تا ۵ ولت باشد و به عنوان ولتاژ مرجع خارجی در نظر گرفته شود.  
 ✓ ولتاژ مرجع داخلی، یعنی ۲/۵۶۷ ولت نیز می‌تواند ولتاژ مرجع باشد.

■ **ضریب تفکیک:** پارامتری (مشخصه‌ای) است که مشخص می‌کند حساسیت یا دقت ADC چقدر است و از رابطه زیر محاسبه می‌شود.

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

در این رابطه n تعداد بیت خروجی مبدل است که می‌تواند با توجه به تنظیمات انجام شده در بخش ADC ۸ یا ۱۰ باشد.

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

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

در این رابطه  $V_{\text{in}}$  ولتاژ ورودی به مبدل است. توجه داشته باشید که حداکثر ولتاژ ورودی به یک مبدل برابر با ولتاژ مرجع است.

**مثال ۹:** اگر ولتاژ ورودی به مبدل، یک ولت و ولتاژ مرجع، از نوع داخلی و برابر ۲/۵۶۷ باشد، مطلوب است:  
 الف) ضریب تفکیک

ب) عدد خروجی مبدل

$$\text{ضریب تفکیک} = \frac{2/567}{2^{10} - 1} = 2/5\text{mv} \qquad \text{عدد خروجی مبدل} = \frac{1\text{v}}{2/5\text{mv}} = 400$$

برای تنظیم ADC در ویزارد مطابق شکل ۱۶-۴، برگه ADC را باز کنید و آن را فعال نمایید. در قسمت volt. ref می‌توانیم مشخص کنیم که ولتاژ مرجع از کدام منبع تأمین شود. در این مثال ولتاژ مرجع را داخلی و برابر با ۲/۵۶۷ انتخاب کرده‌ایم.

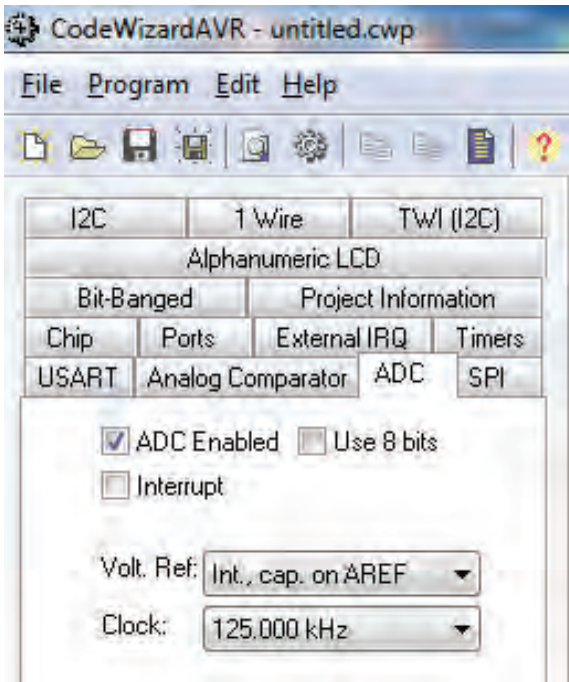
نکته



برای استفاده از مرجع داخلی لازم است، یک خازن در حد ۱uF، که طرف دیگر آن به زمین متصل است، بر روی پایه Aref قرار گیرد. در قسمت Clock فرکانس ورودی به قسمت ADC را مشخص می‌کنیم. این فرکانس باید بین ۵۰k تا ۲۰۰k هرتز باشد.

نکته

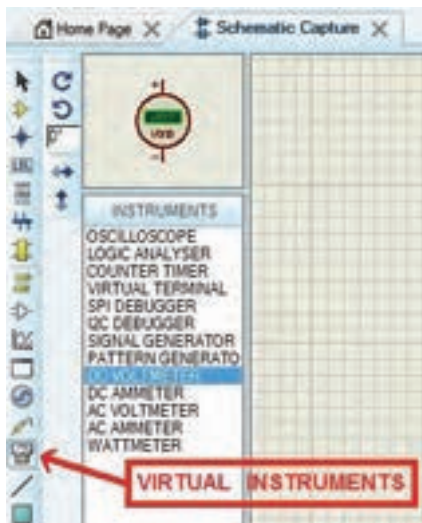




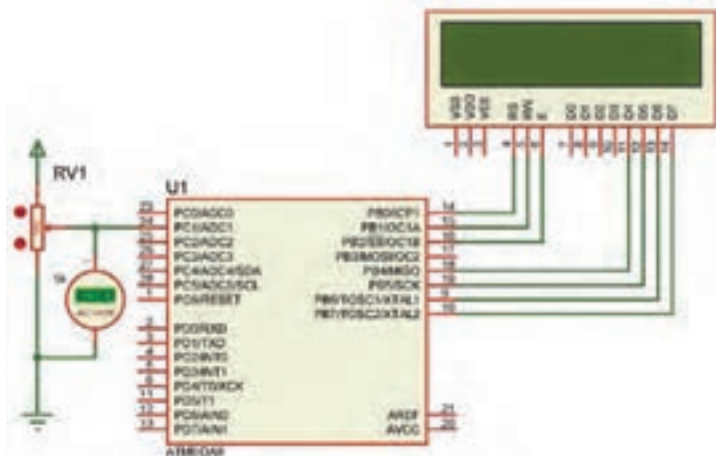
شکل ۴-۱۶- تنظیم ADC در ویزارد

مثال ۱۰: یک پتانسیومتر و یک ولت‌متر DC را مانند شکل ۴-۱۶ الف به کانال ADC۱ متصل کنید، برنامه‌ای بنویسید که عدد خروجی مبدل بر روی LCD نمایش داده شود. با توجه به روابط مربوط به ضریب تفکیک و عدد خروجی، محاسبات را برای ولتاژ ورودی به دست آورید و نتایج را با اعداد نمایش داده شده بر روی LCD مقایسه کنید.

راهنمایی: در نرم‌افزار Proteus مطابق شکل الف و ب ۴-۱۷ می‌توانید پتانسیومتر را با تایپ عبارت POT\_HG در قسمت Keywords و ولت‌متر را در صفحه اصلی از بخش VIRTUAL INSTRUMENT بردارید.



ب



الف

شکل ۴-۱۷

■ پس از تولید کد ملاحظه می‌شود که تابع (read\_adc) به برنامه اضافه شده، ورودی این تابع شماره کانال و خروجی آن که از نوع unsigned int می‌باشد، عدد خروجی مبدل ADC را نشان می‌دهد.

```
// Read the AD conversion result
```

```
unsigned int read_adc(unsigned char adc_input)
```

### جدول ۴-۱۸

```
#include <stdio.h>
unsigned int a;
char s[۶];
void main(void)
{
while (۱)
{
a=read_adc(۱);
sprintf (s,"%۰۴d",a);
lcd_gotoxy(۰,۰);
lcd_puts(s);
delay_ms(۱۰۰);
}}
```

بنابراین نوع متغیری که برای دریافت عدد خروجی تعریف می‌شود را unsigned int انتخاب می‌کنیم. اگر کدهای نوشته شده در جدول ۴-۱۸ را در محل مناسب به برنامه اضافه کنید می‌توانید نتیجه را بر روی lcd ببینید.

با توجه به مدار شکل ۴-۱۸ برنامه را طوری تغییر دهید که یک ولت‌متر داشته باشیم.

فعالیت



### مثال ۱۱: کار با سنسور دمای LM۳۵

- در کتاب مونتاژ و دمونتاز پایه یازدهم با این سنسور آشنا شدید. حساسیت آن  $۱۰\text{mV}/^{\circ}\text{C}$  است. یعنی اگر دمای محیط  $۱^{\circ}\text{C}$  باشد خروجی آن  $۱۰$  میلی‌ولت خواهد بود.
- می‌خواهیم عدد خروجی مبدل ADC را با ولتاژ مرجع داخلی  $۲/۵۶۷$  محاسبه کنیم و برنامه‌ای بنویسیم که دمای محیط را بر روی LCD نمایش دهد.
- همچنین یک موتور FAN را که به میکروکنترلر متصل است در دمای  $۳۰$  درجه سانتی‌گراد روشن و در دمای  $۲۵$  درجه سانتی‌گراد خاموش کند. (چنانچه در محیطی که این آزمایش را انجام می‌دهید این دو دما قابل دسترس نیست می‌توانید آنها را تغییر دهید).

پاسخ:

### جدول ۴-۱۹

#include <stdio.h>	PORTD.۴=۱;
unsigned int a;	lcd_gotoxy(۰,۱);
unsigned char t;	lcd_putsf("MOTOR ON
char s[۱۶];	");
void main(void)	}
{	if(t<=۲۵)
while (۱)	{
{	PORTD.۴=۰;
a=read_adc(۱);	lcd_gotoxy(۰,۱);
t=a/۴;	lcd_putsf("MOTOR OFF
sprintf (s,"Temp=%۰۲d",t);	");
lcd_gotoxy(۰,۰);	}
lcd_puts(s);	delay_ms(۱۰۰);
if(t>=۳۰)	}
{	}
ادامه برنامه در ستون مقابل از بالا	
⇒ به پایین	

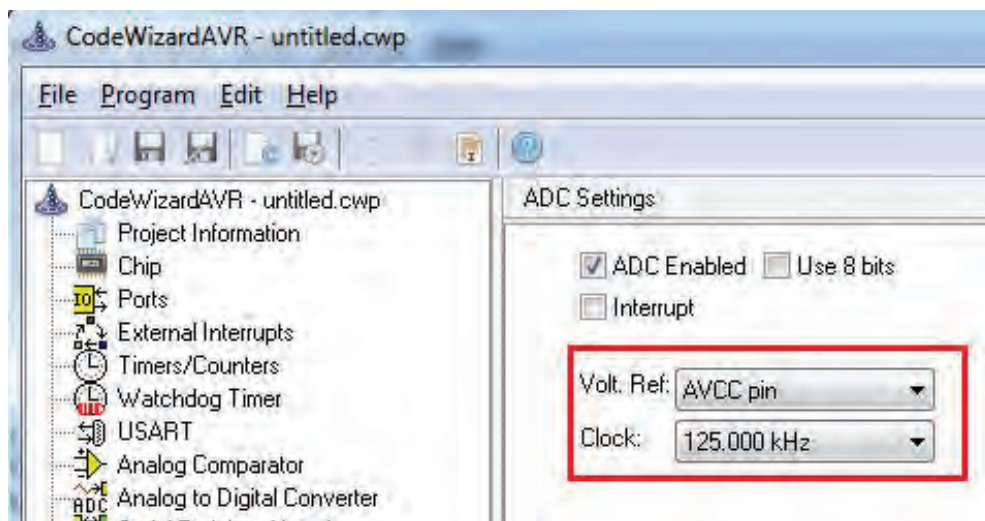
- ☑ موارد مطابق شکل ۴-۱۸ است. در شکل ۴-۱۹ پایه‌های سنسور LM۳۵ و ترانزیستور BDX۵۳C آمده است.
- ☑ مقدار عدد خروجی و ضریب تفکیک را محاسبه می‌کنیم.

$$\text{ضریب تفکیک} = \frac{۲/۵۶۷}{۱۰۲۴} = ۲/۵\text{mv}$$

$$\text{عدد خروجی} = \frac{۱۰\text{mv}}{۲/۵\text{mv}} = ۴$$

- ☑ با توجه به این که به ازای دمای  $۱^{\circ}\text{C}$  عدد خروجی مبدل ۴ است پس برای نمایش دما روی LCD باید عدد خروجی مبدل را بر ۴ تقسیم کنیم.
- ☑ کدهای داده شده در جدول ۴-۱۹ را به کدهای تولید شده توسط ویزارد اضافه کنید.





شکل ۴-۲۱- تنظیم ولتاژ مرجع

جدول ۴-۲۰

<pre>#include &lt;stdio.h&gt; unsigned int a; char s[۱۶]; void main(void) { while (۱) { a=read_adc(۱); sprintf (s,"Light=%۰۴d",a); lcd_gotoxy(۰,۰); lcd_puts(s); ادامه برنامه در ستون مقابل از بالا =&gt; به پایین</pre>	<pre>if(a&gt;=۵۵۰) { PORTD.۴=۱; Lcd_gotoxy(۰,۱); lcd_putsf("LAMP ON "); } if(a&lt;=۵۰۰) { PORTD.۴=۰; Lcd_gotoxy(۰,۱); lcd_putsf("LAMP OFF "); } delay_ms(۱۰۰);}}</pre>
--	--

یکی از درگاه‌های بسیار پرکاربرد در میکروکنترلرها UART است. درباره چگونگی اتصال سخت‌افزاری ماژول‌های آماده شکل ۴-۲۲ از قبیل حسگر اثر انگشت، کارت‌خوان‌های RFID، GPS و GSM به میکروکنترلر تحقیق کنید و نتیجه را در قالب یک گزارش به کلاس ارائه دهید.

تحقیق کنید





شکل ۲۲-۴- اتصال ماژول به میکروکنترلر

#### الگوی آزمون نرم‌افزاری واحد یادگیری ۴

مشابه یکی از مثال‌ها یا فعالیت‌های داده شده در متن واحد یادگیری را به صورت نرم‌افزاری اجرا کنید.

#### الگوی آزمون نظری واحد یادگیری ۴

۱- دستور `#include <mega8.h>` در برنامه، بارگراف کتابخانه میکرو را معرفی می‌کند.

درست  نادرست

۲- دستور مربوط به شرط حلقه `for` و دستور تأخیر را بنویسید.

۳- LCD کاراکتری  $2 \times 16$  نسبت به سایر LCDها پرکاربردتر است.

درست  نادرست

۴- برای اتصال LCD کاراکتری به میکروکنترلر، باید پایه‌های دیتای ..... تا ..... متصل شود.

۵- برای نوشتن کلمات در LCD با استفاده از میکروکنترلر حتماً باید از نرم‌افزاری مانند ویزارد استفاده کنیم.

درست  نادرست

۶- در مثال `chars [۶];` و عدد ۶ چه مفهومی دارد؟

۷- کد اسکی یک کد ..... که در ..... به کار می‌رود و در سطح جهان کاربرد یکسانی دارد.

۸- خروجی ۸ بیت یا ۱۰ بیت در ADC چه تفاوتی با هم دارد و روی چه مشخصه‌ای تأثیر می‌گذارد؟

### ارزشیابی واحد یادگیری ۴: کسب شایستگی در برنامه‌نویسی به زبان C

<p><b>شرح کار:</b></p> <p>۱- تحلیل برنامه‌های نوشته شده به زبان C برای شمارنده ۲- تحلیل برنامه‌های نوشته شده به زبان C برای LCD ۳- تحلیل برنامه‌های نوشته شده به زبان C برای کنترل دما و نور ۴- اجرای انواع برنامه‌های کنترلی با نرم‌افزار</p>			
<p><b>استاندارد عملکرد:</b> تحلیل انواع برنامه‌های آماده به زبان C برای مدارهای کنترلی ساده با استفاده از خروجی‌های LCD و LED و ورودی‌های حسگر دما، رطوبت و نور</p>			
<p><b>شاخص‌ها:</b></p> <p>انتخاب فضای مورد نظر و مناسب بودن آن (میز کار)</p> <p>تحلیل برنامه نوشته شده به زبان C برای شمارنده یا LCD (۳۰ دقیقه)</p> <p>تحلیل برنامه نوشته شده به زبان C کنترل دما، رطوبت و نور (۳۰ دقیقه)</p> <p>اجرای نرم‌افزاری انواع برنامه‌ها (۴۰ دقیقه)</p>			
<p><b>شرایط انجام کار و ابزار و تجهیزات:</b> شرایط انجام کار مشابه بقیه واحدهای یادگیری</p>			
<p><b>معیار شایستگی:</b></p>			
ردیف	مراحل کار	حداقل نمره قبولی از ۳	نمره هنرجو
۱	تحلیل انواع برنامه‌های آماده ساده	۱	
۲	تحلیل انواع برنامه‌ها برای شمارنده با LCD	۲	
۳	تحلیل انواع برنامه‌ها برای کنترل دما، نور و رطوبت	۲	
	<p>شایستگی‌های غیرفنی، ایمنی، بهداشت، توجهات زیست محیطی و نگرش:</p> <p>۱- رعایت نکات ایمنی دستگاه‌ها ۲- دقت و تمرکز در اجرای کار</p> <p>۳- شایستگی تفکر و یادگیری مادام‌العمر ۴- اخلاق حرفه‌ای</p>	۲	
<p><b>میانگین نمرات</b></p>			*
<p>* حداقل میانگین نمرات هنرجو برای قبولی و کسب شایستگی، ۲ می‌باشد.</p>			

