**باسمه تعالی**

# صنایع رباتیک آراد

## بزرگترین خانواده رباتیک ایران

# فصل سوم

**طراح و برنامه نویس : مهندس تالیا براری**

**نویسنده :مهندس فراز امیرغیاثوند**

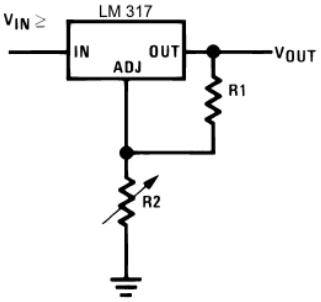
**تاریخ انتشار آبان ۱۳۹۰**

کسی که می خواهد کاری را انجام دهد ، راهش را پیدا می کند و کسی که نمی خواهد ، بهانه اش را

## فصل سوم

### جلسه‌ی بیست و یکم

### LM317، کنترل کننده‌ی ولتاژ و جریان...

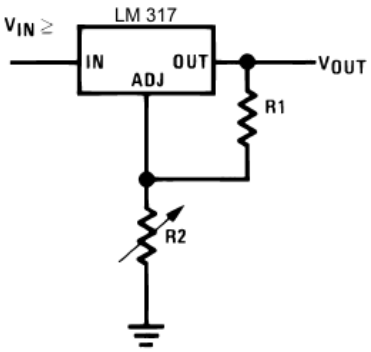



این جلسه می‌خواهیم شما رو با رگولاتور LM317 آشنا کنیم. رگولاتورهایی که ما تا به حال با آن ها آشنا شده‌ایم همگی ولتاژ خروجی ثابتی داشتند، مثلاً ۷۸۰۵ خروجی ثابت ۵ ولت به ما می‌دهد و ۷۸۰۹ خروجی ثابت ۹ ولت!!! اما با رگولاتور LM317 و به کمک یک مقاومت ثابت و یک پتانسیومتر، می توانیم سطح ولتاژ خروجی را به دلخواه خود تنظیم کنیم. البته طبیعتاً سطح ولتاژ خروجی نمی تواند از ولتاژ ورودی بیشتر باشد!

ترتیب پایه های ۳۱۷LM در شکل زیر نشان داده شده است.

همان‌طور که در شکل می‌بینید، خود آی سی یک هیت سینک دارد، ولی معمولاً برای پایین‌تر آوردن دمای آی سی در مدارهایی که نیاز به جریان دهی بالا دارند، هیت سینک آی سی، به وسیله‌ی یک پیچ، به یک هیت سینک کمکی بزرگتر متصل می‌شود. هیت سینک یک قطعه فلزی است که گرما را به خوبی انتقال می‌دهد و نمی‌گذارد دمای آی سی بیش از حد بالا رود. این قطعه به صورت آماده در اندازه‌های مختلف موجود است.

برای استفاده از این آی سی در مَد کنترل کننده‌ی ولتاژ، باید مدار زیر را ببندیم:



در مدار بالا، R۱=۴۷۰اهم است و R۲، یک پتانسیومتر یا مولتی‌ترن ۱۰ کیلو اهمی.

حالا با تغییر مقاومت پتانسیومتر، سطح ولتاژ خروجی تغییر می کند و می‌توانیم آنرا تنظیم کنیم.

برای محاسبه‌ی سطح ولتاژ خروجی، فرمول زیر وجود دارد:

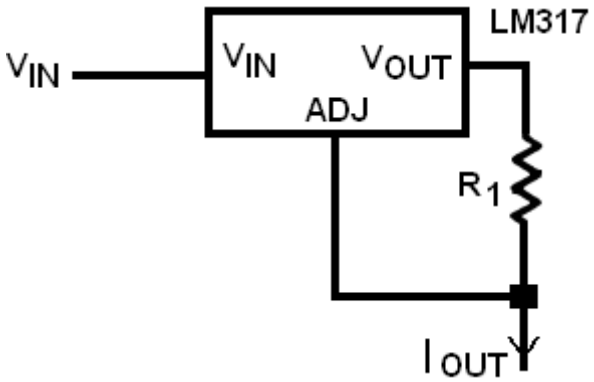
$$V = 1.25(1+ (R2/R1))$$

طبیعتاً نیازی نیست شما هردفعه برای محاسبه‌ی ولتاژ خروجی از این فرمول استفاده کنید، شما می‌توانید با چرخاندن پیچ مولتی‌ترن، ولتاژ خروجی را در سطح ولتاژ مورد نظر تنظیم کنید.

حداقل ولتاژ خروجی در این آی سی ۱.۲۵ ولت می تواند باشد، و حداکثر ولتاژ خروجی نیز، ۳۷ ولت.!

همچنین این آی سی می‌تواند با یک مدار کوچک دیگر، به عنوان کنترل‌کننده‌ی میزان جریان خروجی استفاده شود.

به مدار دقت کنید:




به وسیله‌ی رابطه **I out= Vin/R۱** می‌توان میزان جریان خروجی را حساب کرد.
البته این مدار کاربرد بسیار کمی دارد، و برای کنترل جریان در مدارهای ساده، معمولاً از مقاومت‌های معمولی استفاده می‌کنیم.

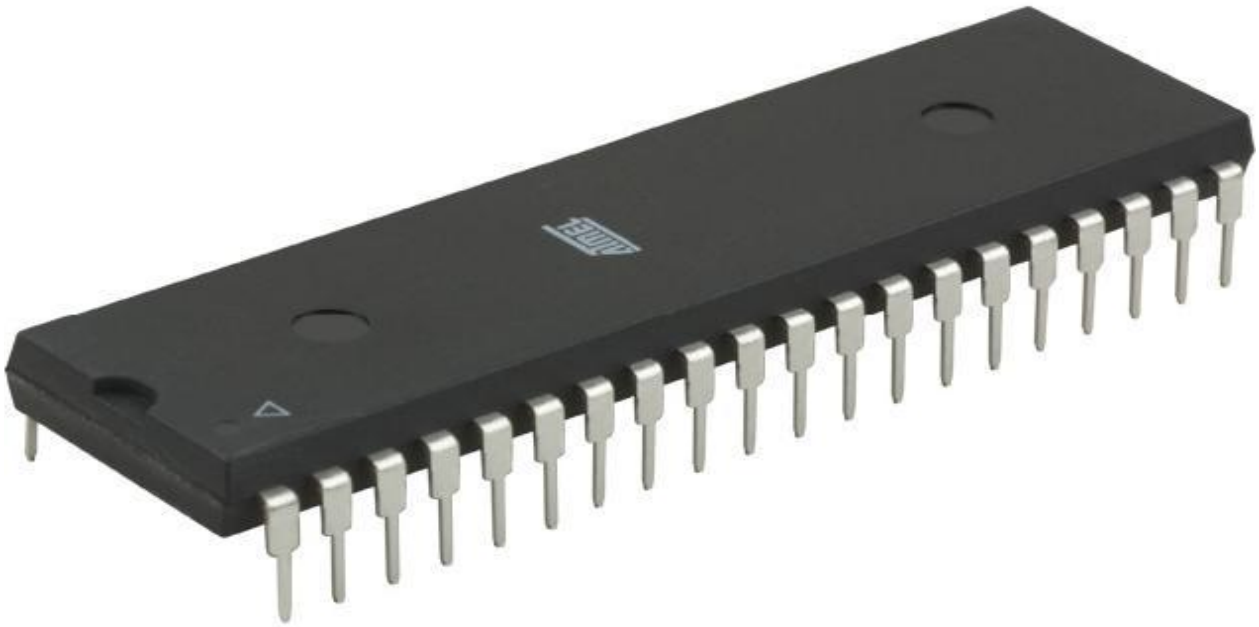
**جلسه‌ی بیست و دوم**

## میکروکنترلر و مقدمات...

از این جلسه ما وارد مبحث میکروکنترلر می‌شویم. این مبحث نسبت به مباحث قبلی، نیازمند توجه و دقت بیشتری است و دوستان باید زمان بیشتری برای آموزش این مبحث صرف کنند. بخشی از این مبحث مربوط به برنامه نویسی تحت زبان C است که ما سعی می کنیم دوستان رو در حد کمی با مقدمات برنامه نویسی در این زبان نیز آشنا کنیم.

میکروکنترلر در زبان انگلیسی به معنی «ریز کنترل کننده" است. این قطعه در واقع یک کنترل کننده ی مرکزی و یک مرکز تصمیم گیری و هدایت برای مدارهای ماست. این قطعه یک آی سی است که می تواند توسط کاربر برنامه ریزی شود. برنامه ریزی آن نیز توسط زبان های مختلف برنامه نویسی مانند C، اسمبلی و basic انجام می شود.

فقط کافیست تمام ورودی و خروجی های مدار یا ربات خود را در اختیار میکرو کنترلر قرار دهیم و سپس الگوریتم مورد نظر خود را تحت یکی از این زبان های برنامه نویسی پیاده سازی کرده و میکروکنتلر را برنامه ریزی کنیم، حالا این قطعه به راحتی ربات یا مدار ما را به طور کامل کنترل می کند.



تصویر بالا تصویر یک میکروکنترلر Atmega16L است. این میکرو کنترلر یک آی سی ۴۰ پایه از خانواده ی میکروکنترلرهای AVR است و به دلیل ویژگی های خاص و قیمت مناسبش(حدوداً ۲۰۰۰تومان)، به عنوان یکی از پرکاربردترین و معروف ترین انواع میکروکنترلرها در جهان شناخته شده است. به همین دلیل ما نیز آموزش کار با همین میکروکنترلر را خواهیم داد. البته این به این مفهوم نیست که ما اگر میکروکنترلرAtmega16L را آموزش ببینیم فقط می تونیم فقط با همین

میکروکنترلر کار کنیم، بلکه کار کردن با سایر میکروکنترلرهای خانواده ی AVR را نیز فرا می گیریم و فقط کافیست چند نکته ی کوچک در مورد میکروکنترلر های دیگر این خانواده یاد بگیریم تا بتوانیم با آن ها نیز کار کنیم.

جالبه بدونید که اولین میکروکنترلرها در دهه ی ۸۰ میلادی ساخته شد، که هنوز هم کار با آن میکروکنترلرها در بسیاری از دانشگاه ها و مراکز مختلف آموزشی، آموزش داده می شود.

خوب، حالا کمی با این قطعه ی جادویی بیشتر آشنا بشویم میکروکنترلر یک ریز پردازنده (Processor) است که می تواند ورودی و خروجی های متعدد داشته باشد. یعنی تعددادی ورودی از محیط دریافت کند و طبق برنامه ریزی هایی که روی آن انجام شده، خروجی هایی متناسب با آن ها صادر کند.

ما برای برنامه ریزی این قطعه، از زبان C که یکی از کاملترین زبان های برنامه نویسی روز دنیاست، استفاده می کنیم.

## توضیحات ابتدایی در مورد قسمت های نرم افزاری:

به برنامه ای که توسط کاربر نوشته می شود، Source گفته می شود. این برنامه باید توسط یک نرم افزار، به زبان قابل فهم برای میکروکنترلر تبدیل شود. به این نرم افزار کامپایلر می گویند. به این برنامه ی کامپایل شده نیز، یکObject می گویند. حالا باید این Object توسط نرم افزار دیگری به چیپ (Chip) یا همان آی سی منتقل شود. به این عمل، یعنی انتقال برنامه ی کامپایل شده به چیپ، پروگرام کردن می گویند و به نرم افزاری که این کار را انجام می دهد پروگرامر (Programmer) می گویند. محیطی که ما در آن برنامه ی مورد نظر خود را می نویسیم (تایپ می کنیم) Editor نام دارد. این نرم افزار ما را در خلل برنامه نویسی بسیار کمک می کند، مثلاً کلمات رزرو شده و غیر قابل تعویض را با رنگها و فونت های گوناگون برای ما برجسته می کند.

این ۳ برنامه، یعنی کامپایلر، پروگرامر و ادیتور، در غالب نرم افزاری به نام "Code Vision" توسط شرکت HP به بازار عرضه شده است . کاربر با نصب این نرم افزار بر روی کامپیوتر شخصی خود، در حقیقت هر ۳ برنام را، به علاوه ی چندین قابلیت و برنامه ی جانبی دیگر را که در جلسات آینده با آن ها آشنا خواهید شد، بر روی دستگاه خود نصب کرده است. در واقع Code vision یک بسته‌ی نرم افزاری کامل و جامع برای خانواده ی AVR است که تمام نیازهای نرم افزاری ما را برای کار کردن با میکروکنترلرهای این خانواده برطرف می کند.
در جلسات آینده در مورد این نرم افزار بیش تر توضیح خواهیم داد.

## توضیحات مقدماتی در مورد قسمت های سخت افزاری:

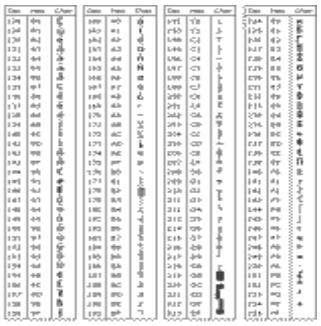
میکروکنترلر Atmega16L دارای ۴ پورت (Port) یا درگاه است. هر پورت دارای ۸ پایه است که می توانند به عنوان ورودی یا خروجی استفاده شوند. در حقیقت این میکروکنترلر دارای ۳۲ پایه برای دریافت اطلاعات و یا صدور دستورات مختلف برای کنترل سایر قطعات است. ۸ پایه ی دیگر نیز وظایف مختلفی بر عهده دارند که در جلسات آینده در مورد آن ها نیز توضیح

داده خواهد شد.

در بعضی از میکروکنترلرها برای انتقال برنامه به چیپ (پروگرام کردن چیپ)، از یک مدار جانبی به نام "Micro controller programmer" استفاده می کنند و چیپ را در آن مدار قرار داده و چیپ باید فقط روی آن مدار پروگرام شود. Atmega16L این قابلیت را دارد که بدون هیچگونه مدار خارجی و فقط به وسیله ی چند رشته سیم معمولی، بر روی خود ربات یا مدار اصلی پروگرام شود. این قابلیت به اختصار ISP یا (In System programing) نام دارد. این قابلیت یکی از بزرگترین مزیت های این نوع میکروکنترلر به شمار می رود. زیرا دیگر نیازی به صرف هزینه ی اضافی برای خرید این مدار نیست. علاوه بر این دیگر نیازی نیست چیپ هر بار برای پروگرام شدن از روی ربات جدا شود.

در مورد میکروکنترلر مطالب بسیار گسترده و زیادی وجود دارد، تا جایی که به عنوان یکی از درس های تخصصی رشته های برق و کامپیوتر به دانشجویان مقطع کارشناسی ارائه می شود. بدیهی است ما نمی توانیم در اینجا تمامی مطالب موجود در مورد میکروکنترلر ها را آموزش دهیم. اما به هر حال در جلسات آینده سعی می کنیم شما رو تا حد مناسبی با این قطعه ی با ارزش آشنا کنیم.

**جلسه‌ی بیست و سوم**



**شروع بحث‌های تخصصی نرم‌افزاری در میکروکنترلر، ASCII Code، اصل ضرب و ...**

---

از این جلسه ما وارد مبحث آموزش مقدماتی زبان C می شویم تا دوستان کمی با مقدمات برنامه نویسی آشنا بشوند. در استفاده از میکروکنترلرها برای ساخت ربات های مقدماتی مثل مسیر یاب و آتش نشان و ... ما نیازی به آموختن برنامه نویسی در حد حرفه ای نداریم و کمی آشنایی با مقدمات برای ما کافیست!!!

بدون مقدمه بیشتر وارد بحث می شویم.

همانطور که می دانید، کوچک ترین واحد ذخیره سازی اطلاعات در حافظه، Bit است. (جلسه ی شانزدهم در مورد یک Bit توضیح داده شده).
هر ۸ بیت را یک Byte می گویند. در حقیقت یک بایت اطلاعات، ۸ تا ۰ یا ۱ است که در مجمع ۲۵۶ حالت مختلف را پدید می اورند.

**توضیح بیشتر:**

یک بیت، فقط ۲ حالت دارد، ۰ یا ۱. وقتی ۲ بیت در کنار هم قرار می گیرند، هر کدام ۲ حالت را پدید می اورند و در مجموع طبق اصل ضرب، ۴ حالت به وجود می آید. یعنی:

۱ و ۱

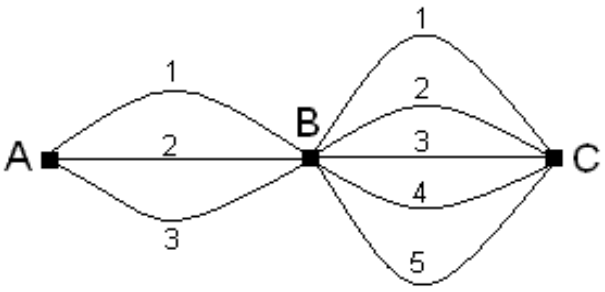
۱ و ۰

۰ و ۱

۰ و ۰

## اصل ضرب:

به مثال زیر توجه کنید.




در شکل بالا، برای رفتن از A به B، ۳ مسیر وجود دارد؛ ۵ مسیر هم برای رفتن از B به C وجود دارد. طبق اصل ضرب، برای رفتن از A به C مجموعاً ۱۵=۵*۳ حالت وجود دارد.

در اینجا، در حقیقت تعداد کل حالت ها، برابر است با حاصل ضرب حالت های هر بیت (که ۲ حالت می باشد) می باشد. به عنوان مثال برای محاسبه ی تعداد حالت های ۳ بیت اطلاعات، داریم: ۸=۲^۳=۲*۲*۲.

طبق همین رابطه، یک بایت، ۲۵۶=۲^۸ حالت مختلف می تواند باشد.

هر ۱۰۲۴ بایت را ۱ **کیلوبایت** می گویند و هر ۱۰۲۴ کیلو بایت، یک **مگابایت** است. هر ۱۰۲۴ مگا بایت هم یک**گیگابایت** و هر ۱۰۲۴ گیگا بایت هم یک **ترابایت** نام دارد.( ۱۰۲۴=۲^۱۰)

حافظه های کامپیوترهای خانگی امروزی، می تواند تا چند صد گیگابایت هم باشد.

## کد ASCII چیست:

موسسه ی استاندارد آمریکا، استانداردی برای ذخیره سازی اطلاعات معرفی کرد. این استاندارد ۲۵۶ کاراکتر (یک کاراکتر

عبارتست از یک عدد، رقم یا یک علامت مثل + و -) را کد گذاری کرد و به هر کدام یک عدد ۸ رقمی در مبنای ۲(یعنی یک بایت) نسبت داد. این کارکترها شامل همه ی حروف الفبای لاتین، اعداد ۰ تا ۹، علامت های مختلف مثل نماد جمع(+) و تفریق(-) و ... هستند.

در حقیقت طبق این استاندارد، برای ذخیره سازی هر کاراکتر، یک بایت از حافظه به آن اختصاص میابد. مثلاً برای ذخیره سازی کلمه ی"ALI" به ۳ بایت حافظه نیاز داریم. جدول کدهای ASCII را می تونید در کتاب های برنامه نویسی یا با جستجو در اینترنت به راحتی ببینید.

## انواع زبان های برنامه نویسی:

### زبان ماشین:

سطح پایین ترین زبان برنامه نویسی زبان ماشین است. در این زبان شما باید به جای گذاشتن علامت + برای جمع کردن مقدار ۲ عدد، باید از کد ۰۰ استفاده کنید. این زبان، زبان قابل فهم برای کامپیوتر است، به همین خاطر به آن زبان ماشین می گویند. برنامه های ما در هر زبان برنامه نویسی دیگری، حتی اسمبلی، باید توسط کاپایلر مخصوص آن زبان، به زبان قابل فهم برای کامپیوتر یعنی زبان ماشین ترجمه شود.

### زبان اسمبلی:

این زبان کمی پیشرفته تر از زبان ماشین است و کارکردن با آن خیلی راحت تر از زبان ماشین است. به عنوان مثال برای جمع کردن ۲ مقدار با یکدیگر می توان از دستورADD استفاده کرد. در این زبان سیستم کد گذاری ASCII هم تعریف شده است و کاربر به عنوان مثال فقط کافیست کلمه ی ALI را تایپ کند، کامپایلر در اینجا کدهای مربوط به این کلمه را از جدول استخراج کرده و جایگزین می کند.

بعد از این ها نوبت به زبان های برنامه نویسی سطح بالا می رسد. این زبان ها سعی کرده اند تا حد امکان به زبان گفتار انسان نزدیک شوند. زبان C یکی از زبان های سطح بالا می باشد.

یک برنامه، شامل چندین دستور مختلف هستش که ما آنها را پشت سرهم با ترتیب مشخصی می نویسیم. درزبان C دستورات باید حتماً داخل توابع باشند. یک تابع عبارتست از چند دستور که در داخل یک آکولاد ({}) نوشته می شوند و نام مشخصی هم برای ان ها گذاشته می شود. همچنین توابع می توانند اطلاعاتی را به عنوان ورودی و خروجی از برنامه دریافت و به آن بازگردانند.

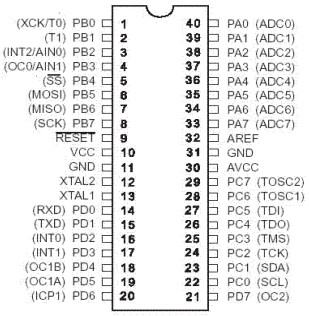
در زبان C وجود تابعی با نام main الزامیست. یعنی ما باید حتماً تابعی با نام main در برنامه ی خود داشته باشیم و اجرای برنامه هم از تابع main شروع می شود.

در Codevision، بعد از انجام تنظیمات اولیه، خود برنامه برای شما قالبی را آماده می کند که در آن تنظیمات اولیه ی پورت ها و .... همچنین بعضی تعاریف اولیه مثل تابع main انجام شده است. فقط کافیست شما دستورات خود را در داخل آن فضای مشخص شده (در داخل تابع main) تایپ کنید.

در جلسه آینده برای آشنایی با نحوه ی برنامه نویسی در فضای Codevision بعد از تعریف متغیرها، برنامه ی یک ربات مسیر یاب بسیار ساده را با هم خواهیم نوشت.

## جلسه‌ی بیست و چهارم

### رجیستر چیست؟ رجیستری‌هایDDRx , PINx, PORTx ، قسمتی از برنامه‌ی یک ربات مسیریاب بسیار ساده و...




رجیسترها توعی حافظه هستند که به طور مستقیم با بخشش پردازش‌گر میکروکنترلر در ارتباط هستند. هر رجیستر یک بایت یا ۸ بیت است. یکی از ویژگی‌های رجیسترها این است که به خاطر ارتباط نزدیک با پردازنده، سرعت بسیار بالاتری نسبت به سایر خانه‌های حافظه دارند...

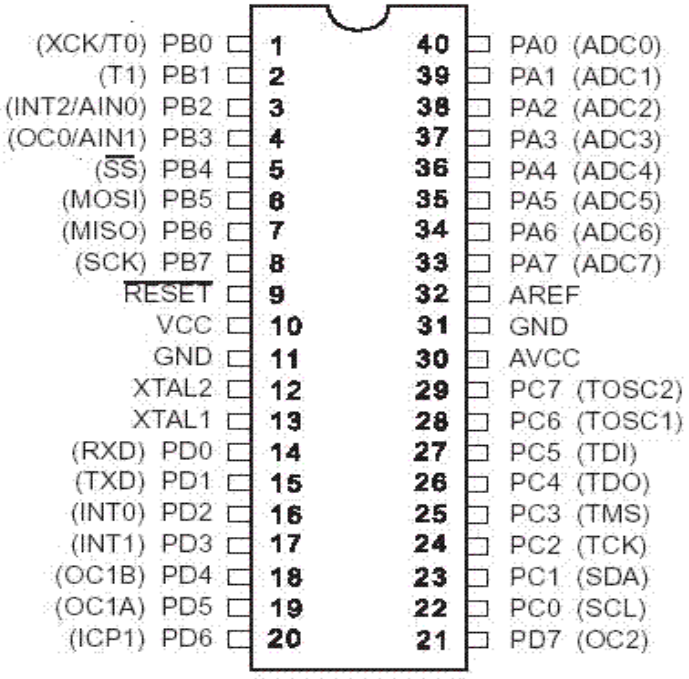
خوب، بر می گردیم سر مبحث میکروکنترلرها....

قرار بود این جلسه برنامه ی یک مسیریاب بسیار ساده ی ۲ سنسوره را با هم بنویسیم. اما ابتدا باید چندتا نکته ی دیگه هم یاد بگیریم.

همونطور که گفته شد AT Mega۱۶ دارای پایه های متعددی برای تبادل اطلاعات با مدار است. هر ۸پایه ی مجاور که این وظیفه را دارند یک پورت نامیده می شوند (به شکل نگاه کنید) . AT Mega۱۶ دارای ۴ پورت با نام های A، B، C و D می باشد. پایه های هر پورت به این شکل نمایش داده می شود:

شماره ی پایه+"."+نام پورت

مثلاً اولین پایه ی پورت D به این صورت نشان داده می شود: .D۰
و پایه ی سوم پورت C به صورت : .C۲
حال به ترتیب پایه های ATMEGA16L دقت کنید




دقت کنید که شماره گذاری پایه ها در پورت ها از ۰ شروع می شود.

همچنین گفته شد، پایه های میکروکنترلر می توانند به صورت ورودی یا خروجی تنظیم شوند، مثلاً در یک ربات مسیریاب میتوان چند پایه را تنظیم کرد که ورودی باشند و اطلاعات سنسورها را دریافت کنند، یا انها را تنظیم کرد تا خروجی باشند و موتورها را هدایت کنند. این تنظیم به صورت نرم افزاری و با تنظیم رجیستر DDRx انجام می گیرد. اما ابتدا باید رجیستر را تعریف کنیم.

### رجیستر چیست؟

رجیسترها توعی حافظه هستند که به طور مستقیم با بخشش پردازشگر میکروکنترلر در ارتباط هستند. هر رجیستر یک بایت یا ۸ بیت است. یکی از ویژگی های رجیسترها این است که به خاطر ارتباط نزدیک با پردازنده، سرعت بسیار بالاتری نسبت به سایر خانه های حافظه دارند.

### رجیستر **DDRx**:

رجیستر DDRx (Data Direction) برای تنظیم ورودی یا خروجی بودن پایه های میکروکنترلر است. برای تنظیم پایه ها در برنامه، باید به جای x باید آدرس پایه ی مورد نظر(مثل B.3) را بنویسیم. اگر بخواهیم آن پایه خروجی باشد باید بیت رجیستر مربوط به آن را ۱ کنیم، و اگر بخواهیم آن پایه ورودی باشد، باید بیت رجیستر مربوط به آن را ۰ کنیم. به عنوان مثال اگر بخواهیم پایه۱۷ یعنی D.3 خروجی باشد باید این جمله را بنویسیم: DDRD.3=1;

و اگر بخواهیم این پایه ورودی باشد: DDRD.3=0;

### رجیستر **PORTx**:

در صورتی که پایه ها به صورت **خروجی** تنظیم شده باشند، هر چه در این رجیستر نوشته شود سطح منطقی پایه ی متناظر را

تعیین می کند، مثلاً اگر بنویسیم PORTB.3 =1 پایه B.۳ یعنی پایه ی ۴، ۱ منطقی خواهد شد(یعنی ولتاژ ۵ ولت بر روی این پایه قرار می گیرد). و اگر بنویسیم PORTC.1=0، پایه ی C.۱ یعنی پایه ی ۲۳، ۰ منطقی خواهد شد (یعنی ولتاژ این پایه ۰ می شود.).

**رجیستر PINx**:

در صورتی که پایه ها به صورت **ورودی** تنظیم شده باشند، محتویات این رجیستر حاوی اطلاعات دریافتی از پایه های میکروکنترلر است. مثلاً اگر PINB.1=۰ باشد، یعنی بر روی پایه شماره ی ۲ یا همان B. ۱ ؛ ۰ منطقی اعمال شده است(مثلاً اگر به سنسوری وصل شده است، خروجی سنسور ۰ منطقی بوده است). در حقیقت این رجیستر برای خواندن وضعیت پایه های ورودی مورد استفاده قرار می گیرد.

**نکته ی بسیار مهم:** دقت کنید که در زبان C، باید در انتهای هر خط از برنامه یک علامت ";" گذاشته شود. به این علامت در زبان انگلیسی سِمی کالِن می گویند.

**نکته ی مهم:**

در حقیقت برای هر پورت ۳ رجیستر(حافظه ۱ بایتی) در داخل میکروکنترلر وجود دارد که به مجموع این ۱۲ رجیستر، رجیسترهای I/O (Input/Output) می گویند.

بسیار خوب، حالا نوبت نوشتن برنامه ی ۱ ربات مسیریاب ساده است که فقط ۲ تا سنسور داره!!!

نرم افزاری کمکی به نام Code Wizard در داخل همان Codevision وجود دارد که کار ما را برای انجام تنظیمات اولیه مانند تنظیم ورودی یا خروجی بودن پایه ها آسان می کند. یعنی دیگه نیازی نیست برای هر پایه تک تک با رجیستری DDR سرو کله بزنیم، و به راحتی با چند تا تیک ساده همه ی پایه ها رو تنظیم می کنیم. البته Code wizard همونطور که از اسمش هم معلومه بسیاری امکانات جادویی دیگری هم داره که در جلسات آینده به تدریج با آن ها آشنا خواهیم شد. Code Wizard در حقیقت برای ساده تر کردن و سریع تر کردن برنامه نویسی در فضای Codevision طراحی شده است و کارش این است که قسمت های زیادی از برنامه را به صورت خود کار و طبق خواسته های ما برای ما می نویسد.

پس با این حساب نیازی نیست تنظیمات رجیستری DDRx رو ما در برنامه خودمون انجام بدیم و این کار رو به Code wizard واگذار می کنیم. با Code wizard در جلسه ی آینده آشنا خواهیم شد.

پس در این جلسه فرض می کنیم تنظیمات اولیه مثل رجیستری DDRx و ... انجام شده است. پایه های B. ۰ و B.۱ را به صورت ورودی(برای دریافت اطلاعات سنسورها)، و پایه های B.۲، B.۳، B.۴ و B. ۵ را به صورت خروجی (برای کنترل حرکت موتورها) تنظیم کرده می کنیم.

B.۲ و B.۳ برای کنترل موتور سمت راست و B.۰ برای سنسور سمت راست!

B.۴ و B.۰ ۵ برای کنترل موتور سمت چپ و B.۱ برای سنسور سمت چپ!

حال مانند ربات قبلی، یک پایه از هر موتور را ۰ می کنیم؛ و روشن و خاموش کردن هر موتور را، با اعمال ۰ یا ۱ منطقی بر روی پایه ی دیگر کنترل می کنیم.

پایه ی دیگر را هم به صورت هماهنگ با سنسور متناظر آن سمت ۰ و ۱ می کنیم، یعنی اگر خروجی سنسور ۰ بود، پایه ی موتور را ۰ می کنیم و اگر ۱ بود ، پایه را ۱ کرده و موتور را فعال می کنیم.(به شرطی که از مدار گیرنده ی شماره ۲ استفاده شود(جلسه ی ۱۵))

در زبان C علامت "=" یک عملگر است که عملوند سمت راست خود را خوانده و در عملوند سمت چپ خود می ریزد. مثلاً وقتی می نویسیم:

PORTB.۳=PINB.۰ ;

ابتدا مقداری B.۰ خوانده می شود و سپس بر روی B.۳ ریخته می شود. یعنی مثلاً اگر روی B. ۰ ، ۱ منطقی اعمال شده باشد، پایه ی B. ۳ نیز ۱ منطقی می شود.

حال با توضیحات داده شده به برنامه ی ربات مسیر یاب ساده دقت کنید:

```
PORTB.2=0;

PORTB.4=0;

PORTB.3=PINB.0;

PORTB.5=PINB.1;
```

همانطور که می بینید این برنامه بسیار ساده و کوتاه است.

در جلسات آینده سعی می کنیم شما رو با Code wizard بیشتر آشنا کنیم. منتظر سوالات و نظرات دوستای خوبم هستم.

**جلسه‌ی بیست و پنجم**

## درباره‌ی متغیرها، Identifierها، نکات مهم در برنامه نویسی و...


در این جلسه نیز در ادامه‌ی مطالب جلسه پیش، سعی می‌کنیم کمی بیشتر با نحوه‌ی برنامه نویسی در زبان Cآشنا شویم...

### متغیّر چیست؟

متغیر قسمتی از حافظه است که ما برای آن یک نام دلخواه انتخاب می کنیم و از آن برای نگه داری اطلاعات مورد نیاز خود در روند اجرای رنامه استفاده می کنیم.

۱- متغیرها با خاموش شدن مدار پاک می شوند و حافظه ی دائمی نیستند.

۲- باید نوع اطلاعاتی که قرار است در متغیر نگه داری شود، معلوم گردد، مثلاً قرار است در آن عدد ذخیره شود یا حروف، یا عدد اعشاری یا ...

۳- کامپیلر به صورت خودکار بخشی از حافظه را به متغیر مورد نیاز ما اختصاص می دهد و نیازی نیست ما برای آن مشخص کنیم که طلاعات را در کجای حافظه ذخیره کند. البته می توان در صورت نیاز آدرس بخشی از حافظه را مشخص کنیم تا اطلاعات ما در ان جا ذخیره شود(که فعلاً به آن نمی پردازیم).

### تعریف متغیر:

رای تعریف یک متغیر ابتدا باید نوع یا تایپ (Type) اطلاعاتی که قرار است در آن ذخیره شود، نوشته شود، و بعد از یک فاصله (Space) ام متغیر نوشته شود. به مثال زیر دقت کنید:

```
int a;
```

ر اینجا متغیری با نام "a" و از نوع **int**eger یا همان عددی تعریف شده است، یعنی در این متغیر فقط می توان یک عدد صحیح (غیر عشاری) را ذخیره کرد.

**کته**: اگر یک عدد اعشاری در آن ریخته شود، بخش اعشاری آن حذف می شود.

ر متغیر از جنس int، دو بایت حافظه را به خود اختصاص می دهد و می توان در آن اعداد در گستره ی۳۲۷۶۷ تا ۳۲۷۶۸- را ذخیره کرد.

رای ذخیره سازی حروف(Character) باید متغیر از نوع Char تعریف شود. متغیرهای Char یک بایت حافظه را به خود اختصاص می

هند و در آن ها می توان تنها یک حرف را ذخیره سازی کرد. برای ذخیره سازی حروف در حافظه، کد اسکی (ASCII code) حروف در
ن ذخیره می شود.

ر جدول زیر چند نوع داده(Data Type) ی دیگر نیز معرفی شده است.

| Type | اندازه(size) | بازه ی تحت پوشش |
|---|---|---|
| Long int | 4بایت | ۲۱۴۷۴۸۳۶۴۷ تا ۲۱۴۷۴۸۳۶۴۸ - |
| Unsigned long int | 4بایت | 4294967295از ۰ تا |
| Float | 4بایت | برای اعداد اعشاری |
| Unsigned int | 2بایت | 65535از ۰ تا |

رای ذخیره سازی اطلاعات در داخل متغیرها نیز از همان عملگر "=" استفاده می کنیم. مثلاً:

```
sum1=75;
```

ی توانیم متغیرها در همان موقع تعریف مقدار دهی کنیم. به این کار مقدار دهی اولیه یا "Initialize" کردن میگویند. مثلاً:

```
int sum1=75;
```

طلاعاتی که در داخل متغیرها ذخیره می شود ثابت نیست و می توان در هر جای برنامه که لازم بود، مقدار دیگری در متغیر ذخیره کرد.
ثلاً:

```
int Cross1=34;
.
.
.
Cross1= 68;
```

گر بخواهیم مقدار متغیر ثابت و غیر قابل تغییر باشد باید قبل از تعیین نوع متغیر، کلمه ی "const" را بنویسیم. مثلاً

```
Const float pi=3.14;
```

ی توان چند متغیر را با هم تعریف کرد و انها را مقدار دهی کرد. مثلاً:

```
char a1='a', a2, a3, a4='B';
```

وجه:برای مقدار دهی متغیرهایی که از جنس **"char"** تعریف میشوند، باید مقدار در داخل ' ' قرار بگیرد، به مثال بالا دقت کنید.

## قوانین نام گذاری شناسه ها (Identifiers)در زبان C:

شناسه ها همان نام هایی هستند که برای متغیرها، توابع و ... انتخاب می شوند.

رای انتخاب یک شناسه فقط می توانیم از حروف زیر استفاده کنیم:

۱- اعداد ۹ تا ۰

۲- حروف z تا a (حروف کوچک)

۳- حروفZ تا A (حروف بزرگ)

۴- خط فاصله "_" (Under Line)

۵- علامت $

ه غیر از این کاراکترها مجاز به استفاده از هیچ کاراکتر دیگری (حتی فاصله(Space) ) نیستیم.

همچنین در ابتدا شناسه ها نمی توانیم از اعداد استفاده کنیم. مثلاً شناسه ی ۱loop غلط است، ولی ۱Loop درست است.

طول شناسه ها نیز نمیتواند بیش از ۳۲ کاراکتر باشد.

عضی کلمات در این زبان جزو **کلمات رزرو شده** (Reserved word) هستند و نمی توانند به عنوان شناسه استفاده شوند مانند: int, float, void, char, while, i ، و ...

## نکات مهم در مورد برنامه نویسی در زبان C:

۱- در پایان هر دستور باید یک ";" گذاشته شود.

۲- جملات و عبارات غیر عددی را باید در داخل " " قرار دهیم. مثلاً اگر می خواهیم کارکتر B را در داخل متغیری با نام Temp که از جنس char تعریف شده است ذخیره کنیم، باید بنویسیم:

```
Temp='B';
```

۱- زبان C در اصطلاح یک زبان Case sensitive است، یعنی در این زبان بین حروف بزرگ و کوچک تفاوت وجود دارد. مثلاً در یک برنامه ما می توانیم دو متغیر با نام های "temp" و "Temp" داشته باشیم که ارتباطی هم با یکدیگر ندارند.

۲- اگر بخواهیم در هر قسمت از برنامه توضیحاتی رو بنویسیم، باید یک "//" در ابتدای جمله بنویسیم. مثلاً:

```
int a; // etelaate porte C dar in moteghayer rikhte mishavad
```

همچنین اگر بخواهیم چند خط پشت سر هم را موقتاً از روند اجرای برنامه حذف کنیم، باید علامت"*/" را در ابتدا، و "/*" را در انتهای ان خطوط قرار دهیم. هرگاه این ۲ علامت را پاک کنیم، دوباره آن قسمت، به روند اجرای برنامه اضافه می شود.

۳- در ساختار زیر، هردستور یا دستوراتی که در داخل {} نوشته شود، بی نهایت بار انجام می شود. در حقیقت while(1)، یک حلقه ی بی پایان است که دستورات داخل آن تا وقتی که مدار فعال باشد، تکرار می شوند. در جلسات آیند شما با ساختار حلقه ها بیشتر آشنا خواهید شد.

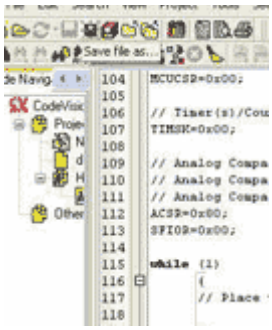
```
while(1)

{

PORTD.3=PINA.2;

PORTD.4=PINA.3;

}
```

این ۲ دستور مکرراً تا زمانیکه میکروکنترلر فعال باشد، اجرا می شوند.

هر بخش برنامه نویسی مطالب بسیار گسترده ای برای آموزش هست، ولی جلسه ی آینده سعی می کنیم کمی هم از سایر بخش های نرم افزاری میکروکنترلر یعنی همان Code vision صحبت کنیم تا تنوعی هم در مطالب ارائه شده داشته باشیم.

**جلسه‌ی بیست و ششم**

## آشنایی باCodevision ، لبه‌یPort ، لبه‌یChip ، آشنایی با Pullup و Output valueو...



Crackشده‌ی این نرم افزار را، برای دانلود دوستان قرار دادیم.(حجم ۸.۱۵مگابایت)

این جلسه همان‌طور که قول داده بودیم، قراره کمی در مورد Codevision توضیحاتی بدیم.

در ابتدا دوستان عزیز برای اینکه بتونن مطلب رو با ما دنبال کنند، لازمه که این نرم افزار تهیه کرده و روی کامپیوتر شخصی خود نصب کنند. در زیر نسخه ی۲.۰۳.۴ Crack شده‌ی این نرم افزار را، برای دانلود دوستان قرار دادیم.(حجم ۸.۱۵مگابایت)
برای دانلود نرم افزار **اینجا** کلیک کنید
پس انجام مراحل نصب برنامه، برنامه را باز کنید.

### چگونه یک پروژه‌ی جدید تعریف کنیم؟

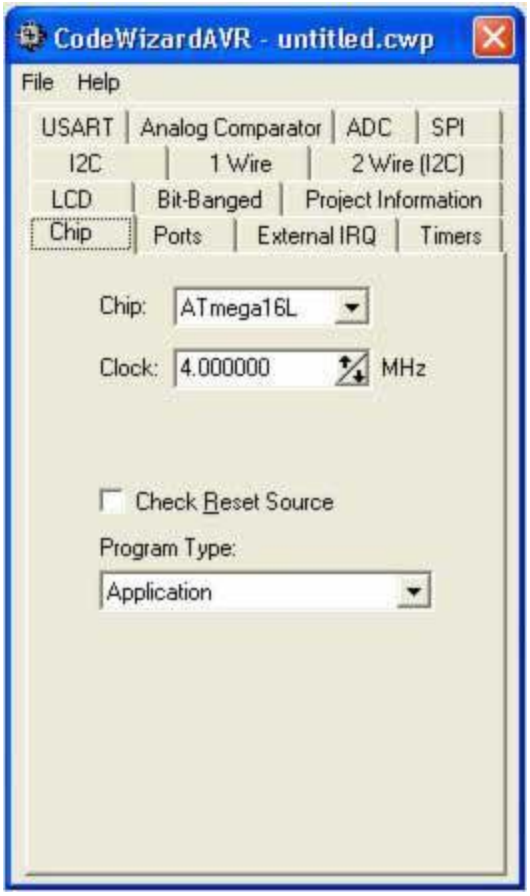
برای نوشتن یک پروژه ی جدید، باید ابتدا از منوی File، گزینه ی New را انتخاب کنید. یک پنجره ی کوچک در وسط صفحه باز می شود که در آن باید گزینه ی Project را انتخاب کرده و تایید کنید. بلافاصله پنجره ی دیگری باز می شود که از شما سوال می کند آیا تمایل دارید برای انجام پروژه ی خود از CodeWizard استفاده کنید؟

همانطور که گفته شد، CodeWizard یکی از نرم افزارهای جانبی CodeVision است که به وسیله ی یک واسط گرافیکی، در نوشتن برنامه ی اصلی و انجام تنظیمات اولیه پورت ها و ....، کمک بسیار زیادی به ما می کند.

پس گزینه ی Yes را انتخاب می کنیم و CodeWizard باز می شود.

### چگونه از CodeWizard استفاده کنیم:

شکل زیر، نمای کلی از CodeWizard است:




همانطور که میبینید، لبه های متعددی برای انجام تنظیمات مختلف میکروکنترلر در آن وجود دارد.

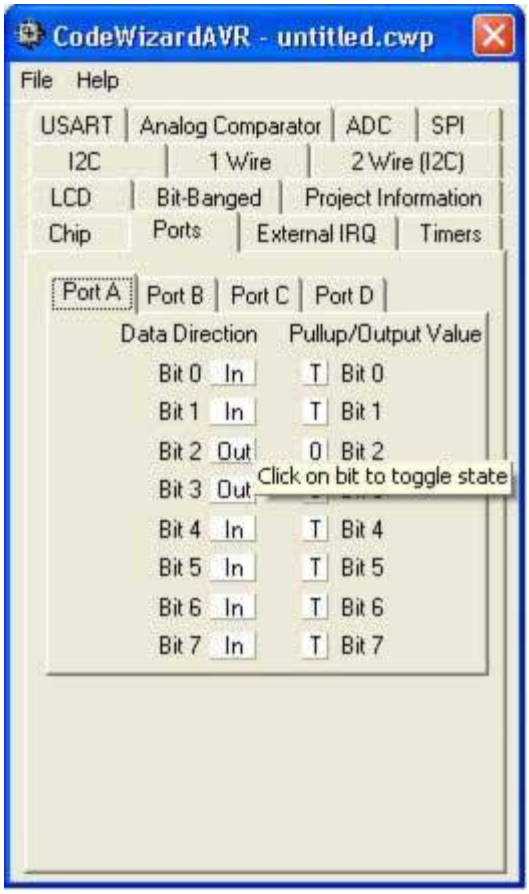
## لبه‌ی:Chip

اولین لبه ای که ما با آن کار داریم لبه ی Chip است. در این قسمت ما باید نوع میکروکنترلر خودمان را انتخاب کنیم. همانطور که در شکل بالا می بینید، میکرو کنترلر Atmega16L را انتخاب کرده ایم.

قسمت Clock مربوط به تنظیم فرکانس کاری آی سی است که ما فعلاً وارد این مبحث نمی‌شویم و آنرا با همان مقدار پیش فرض می پذیریم. با قسمت های دیگر این لبه هم ما کاری نداریم و آن ها را به همان صورت پیش فرض می پذیریم.

## لبه‌ی:Port

این لبه مربوط به تنظیمات ورودی خروجی پایه هاست.




همانطور که می بینید هر پایه از هر پورت را در این قسمت می توان به راحتی به صورت In (ورودی) و یا Out(خروجی) تنظیم کرد. فقط کافیست لبه ی مربوط به پورت مورد نظر را انتخاب کنید، حالا برای تغییر وضعیت هر پایه باید روی آن کلیک کنید.

**Output valueیا مقدار اولیه:**

وقتی پایه ای را به صورت خروجی تنظیم می کنیم، می توان با تنظیم رجیستری PORTx تعیین کرد که سطح ولتاژ خروجی این پایه به صورت پیش فرض ۰ باشد یا ۱. یعنی در زمانیکه هنوز برنامه‌ی ما برای پایه ها تعیین وضعیت نکرده است، می توان به این طریق سطح ولتاژ اولیه ی پایه را تعیین نمود.

CodeWizard در این جا هم کار ما را راحت تر کرده است، در ستون مقابل یعنی ستون "Pullup/Output value" برای پایه هایی که به صورت **خروجی** تعریف شده اند، می توان با یک کلیک وضعیت خروجی پایه را مشخص کرد. مثلاً الآن پایه ی شماره ی ۲ از پورت A به صورت خروجی تعریف شده و در ستون مقابل نیز مقدار پیش فرض خروجی ۰ تعیین شده است.

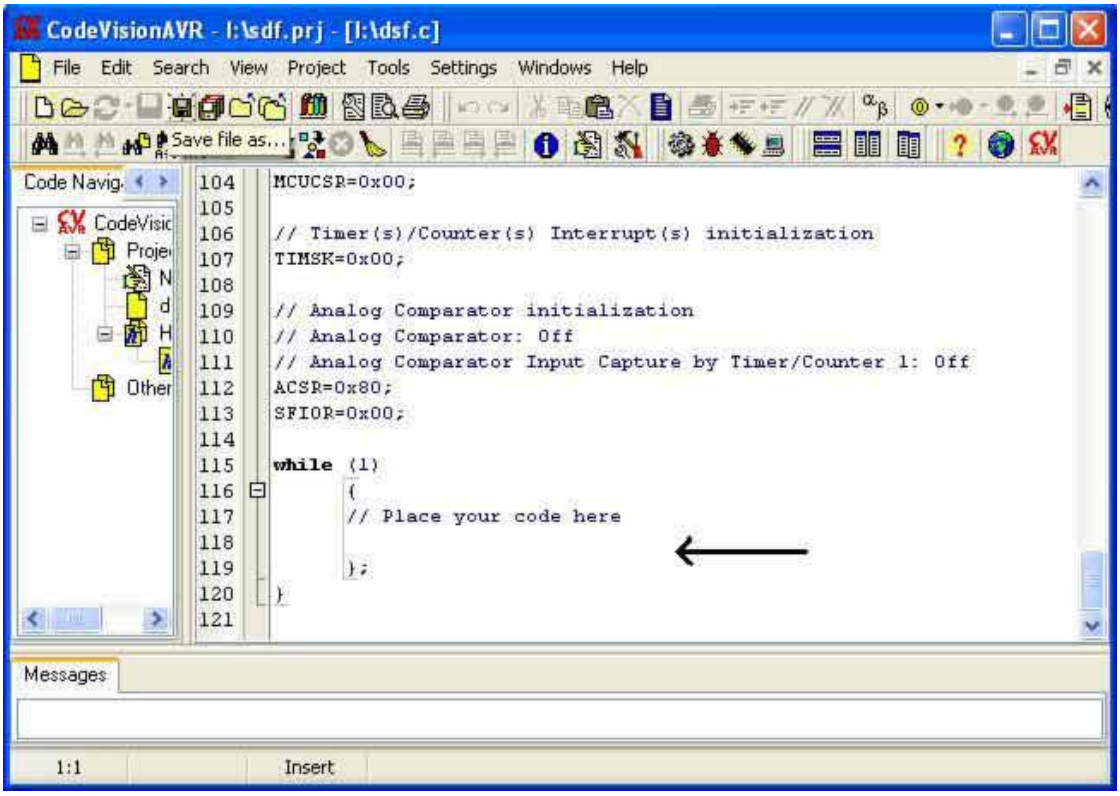
**Pullup:**

این قابلیت سخت افزاریست و در خانواده ی AVR نیز وجود دارد. Pullup کردن به این معناست که پایه ای را با یک مقاومت بالا(مثلاً ۱۰ کیلو اهم) به + وصل کنیم. اگر هم پایه را با این مقاومت به GND وصل کنیم، می گوییم پایه را Pulldown کرده ایم. مقاومت بزرگ باعث می شود که جریان عبوری به حداقل کاهش یابد، ولی به وسیله ی ولتاژی که بر روی پایه قرار می گیرد، می‌توان ورودی مورد نظر را در هنگامیکه هنوز ورودی از خارج دریافت نکرده به صورت پیش فرض ۰ یا ۱ کرد.

وقتی پایه ای را به صورت **ورودی** تعریف می کنیم، با تنظیم رجیستری PORTx می توان تعیین کرد که پایه ی ورودی به صورت پیش فرض Pullup باشد یا نباشد. دقت کنید که در اینجا نمی توان تنظیم کرد که مقدار ورودی پیش فرض ۰ باشد، چون خانواده ی AVR قابلیت Pulldown ندارند وفقط می توان آنرا به صورت Pullup تنظیم نمود، و در نتیجه پایه ای که Pullup شده است در هنگامیکه هنوز از خارج مقداری را دریافت نکرده است، به صورت پیش فرض ۱ منطقی شود.حالا بعد از انجام تنظیمات اولیه پورت ها و خود آی سی ، باید از Codewizard بخواهیم تا یک برنامه ی نیمه آماده با توجه به تنظیماتی که تنجان داده ایم در اختیار ما بذاره.

برای این کار از منوی File گزینه ی "Generate, Save and Exit" را انتخاب کنید. حالا باید جایی که می خواهید برنامه ی شما Save شود را مشخص کنید. Codevision در اینجا ۳ فایل برای برنامه ی شما می سازد که باید آن ها را نام گذاری کنید. بهتر است نام این ۳ فایل و محل ذخیره سازی آن ها یکی باشد.

بعد از ساخته شدن این ۳ فایل توسط Codevision برنامه آماده است، حالا شما باید دستورات خود را در محل تعیین شده بنویسید.




بعد از نوشتن برنامه باید آنرا کامپایل کرده و و سپس فایلHex آنرا بسازید و بعد از آن، فایل Hex را در میکرو کنترلر Load کنید. حالا میکروکنترلر شما پروگرام شده و آماده ی استفاده است. در جلسه ی آینده با روند اجرای این مراحل در Codevision آشنا خوهید شد.

**جلسه‌ی بیست و هفتم**

مراحل کامپایل کردن، پروگرام کردن میکروکنترلر و رفع نقص برنامه و...

در این جلسه قراره شما رو با مراحل کامپایل کردن ، پروگرام کردن میکروکنترلر و رفع نقص برنامه آشنا کنیم.

همانطور که گفته شد فقط «زبان ماشین»(Machine Language)، زبان قابل فهم برای پردازنده ی کامپیوتر است، و برنامه هایی که در زبان های دیگر می نویسیم برای اینکه بتوانند توسط پردازنده اجرا شوند باید حتماً توسط کامپایلرها به «زبان ماشین» ترجمه شوند. اما نوشتن برنامه در این زبان برای ما بسیار مشکل است، زیرا دستورات قابل فهم برای این زبان بسیار ابتدایی و ساده هستند و به سختی می توان برنامه های حرفه ای و الگوریتم های پیچیده را در آن پیاده سازی کرد. مثلاً حتی برای انتقال داده از یک متغیر به متغیر دیگر، باید چندین خط برنامه بنویسید، اما در زبان C این کار در ۱ عبارت انجام می شود. برنامه نویسی در این زبان دشواری های مختلفی دارد که فعلاً به آن ها نمی پردازیم.

به همین خاطر ما برنامه های خود را در زبان C می نویسیم و باقی کارها را به کامپایلر می سپاریم. **کامپایلر** ابتدا برنامه ی ما را از زبان C به زبان اسمبلی ترجمه می کند، سپس برنامه ی دیگری به نام «**اسمبلر**»(" Assembler") برنامه ی ما را از

اسمبلی به «زبان ماشین» تبدیل می کند.

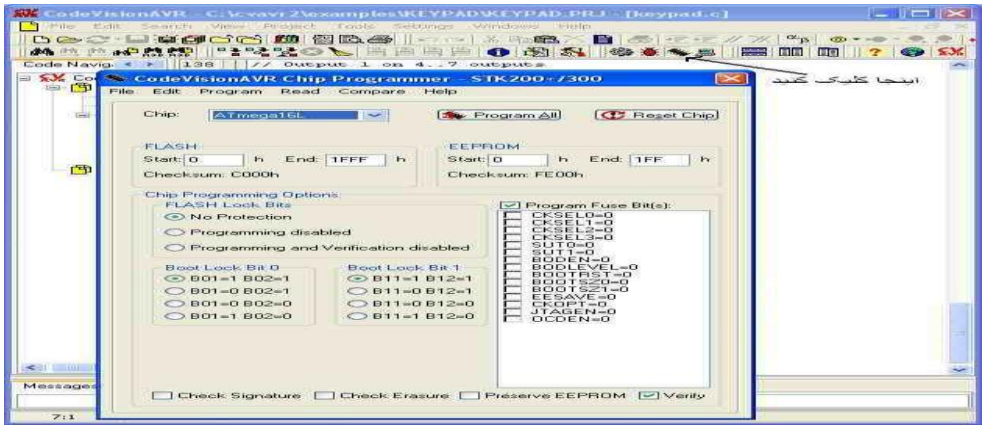
**زبان اسمبلی** یک پله کاملتر از زبان ماشین است. برنامه نویسی در این زبان بسیار ساده تر از زبان ماشین است و بعضی از مشکلاتی که در زبان ماشین وجود داشت در این زبان برطرف شده و یکی از زبان های رایج فعلی برای برنامه نویسی میکروکنترلرها همین زبان اسمبلی است که بیشتر هم در برنامه نویسی میکروکنترلرهای سری ۸۰۵۱ استفاده می شود. اما برنامه نویسی در این زبان هم بسیار پیچیده تر از زبان C است و نوشتن برنامه های حرفه ای و طولانی در این زبان بسیار دشوار است.

**حال چگونه باید این مراحل را در محیط CodeVision انجام داد:**

بعد از نوشتن برنامه، شما می توانید با فشار دادن کلید ۹F برنامه ی خود را کامپایل کنید. با فشار دادن همزمان ۹Shift+F برنامه ی شما ابتدا کامپایل شده و به اسمبلی تبدیل می شود وسپس توسط اسمبلر، به زبان ماشین تبدیل می شود. سپس فایلی با پسوند .HEX در محلی که شما مشخص کرده اید(در هنگام ساختن پروژه) ساخته می شود.این فایل همان برنامه ی شماست و شما باید این فایل را طی مراحلی که در ادامه توضیح داده می شود، در میکروکنترلر Load کنید.

در اینجا ما نیاز به نرم افزارِ پروگرامر "Programmer" داریم تا اطلاعات ما رو با پرتوکل های مشخصی که در جلسات آینده در مورد آن ها توضیح خواهیم داد، به میکروکنترلر منتقل کند.

همانطور که در جلسات پیش مطرح شد، CodeVision مجموعه ای از چند برنامه ی مختلف است که در کنار هم جمع شده اند تا همه ی نیازهای کاربر را برطرف کنند. در اینجا هم پروگرامر CodeVision مشکل ما رو حل می کند. برای استفاده از پروگرامر، باید در نوارابزار بالا روی "Chip Programmer" کلیک کنید تا پنجره ای به شکل زیر باز شود.




حال از منوی File ِ همین پنجره، گزینه ی Load Flash را انتخاب کنید. حالا فایلی که در قسمت بالا ساختید (.Hex) را از پوشه ی "exe" انتخاب کنید. البته به صورت پیش فرض این کار انجام می شود و فایل .HEX برنامه ی شما در پروگرامر Load می شود، اما ممکن است گاهی به دلایل مختلف نیاز باشد فایل دیگری را Load کنید.

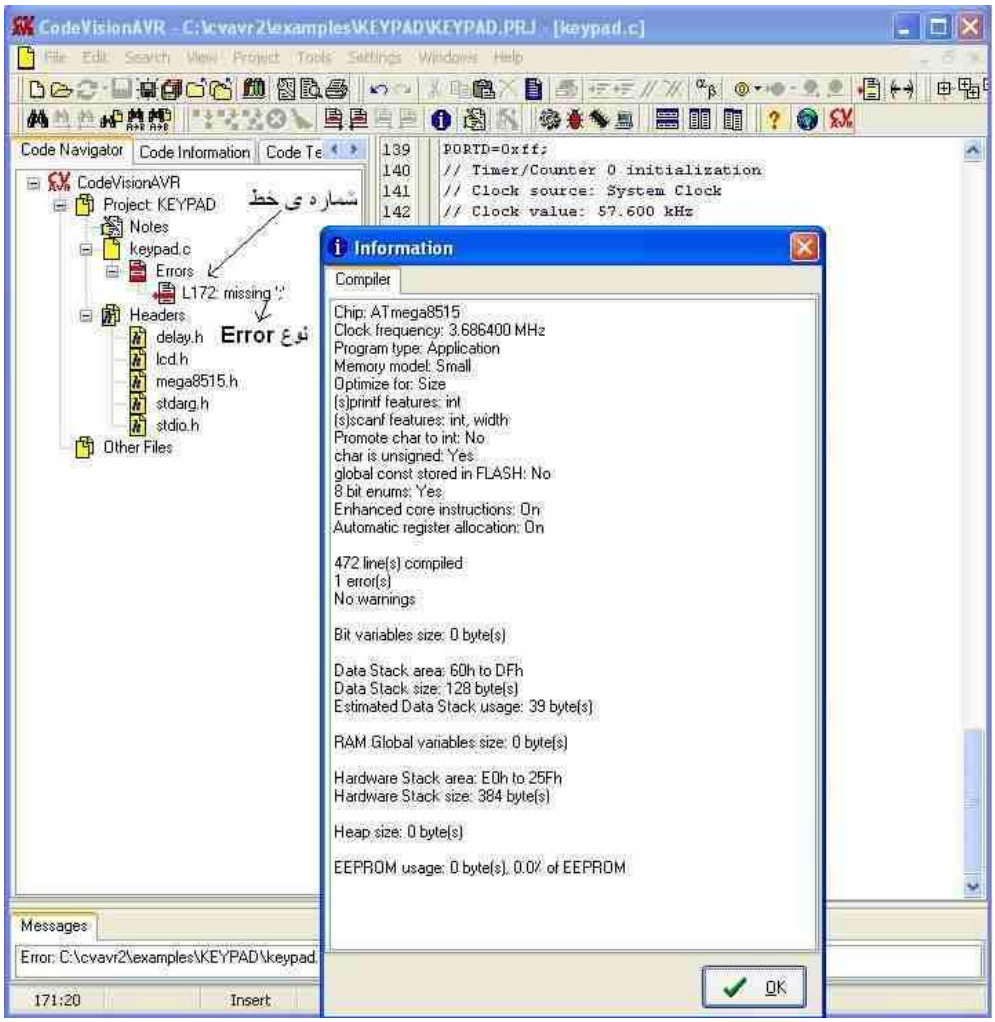
حالا شما باید کابل ارتباطی بین میکروکنترلر و کامپیوتر را متصل کنید ارتباط بین میکروکنترلر و کامپیوتر برقرار شود. توضیحات مفصل در مورد این قسمت و نحوه ی ساخت این کابل در جلسات آینده به تفصیل توضیح داده خواهد شد.

حالا از منوی "Program" گزینه ی "Erase Chip" را انتخاب کنید تا برنامه های قبلی ای که رو میکروکنترلر شما هست

پاک شود و میکروکنترلر آماده دریافت برنامه ی جدید شود. سپس از همین قسمت، گزینه ی "Flash" را انتخاب کنید تا برنامه ی جدید شما در داخل میکروکنترلر Load شود. حالا میکروکنترلر شما پروگرام شده و آماده ی استفاده است.

**مشکلات احتمالی:**

در بخش کامپایل کردن برنامه، ممکن است برنامه ی شما ایرادات مختلفی داشته باشد که مانع کامپایل شدن برنامه شود. این ایرادات(Errors)، همراه با شماره ی خطی که در آن ایراد وجود دارد، بعد از هربار که برنامه را کامپایل می کنید در قسمت سمت چپ، در لبه ی"Code Navigator" نمایش داده می شوند .




**نکته ی مهم:**یکی از رایج ترین ایرادات که مربوط به نگذاشتن ";" در پایان جملات است، ایراد "';'missing " است.

سپس بعد از رفع ایراد ، دوباره برنامه را کامپایل کنید و اگر Error در آن قسمت نبود، برنامه ی شما کامل است.

در جلسه ی آینده، به شما نحوه ی ساخت و استفاده از کابل پروگرامر را آموزش می دهیم. بعد از آن در مورد سخت افزار و نحوه ی استفاده و راه اندازی میکروکنترلر در مدار را آموزش می دهیم. بعد از طی شدن این مراحل، شما می توانید یک ربات مسیریاب **ساده ی** میکروکنترلر دار طراحی کنید و بسازید.

**جلسه ی بیست و هشتم**

**آشنایی با مسابقات رباتیک. لیگ ربات‌های جستجوگر، لیگ ربات‌های فوتبالیست و...**

در این جلسه، یه کم از بحث تخصصیمون خارج می شویم و میریم سراغ حاشیه!!!

ولی قبل از شروع مطلب، بنا به درخواست بسیاری از دوستان، یک کتاب در مورد نحوه ی کار با میکروکنترلرهای خانواده ی AVR معرفی می کنیم. کتاب میکروکنترلرهای AVR، تالیف مهندس ره افروز، کتاب مناسبی هست و دوستان می توانند برای مبحث میکروکنترلر کار ما، این کتاب را تهیه کنند.

در این جلسه قراره کمی در مورد مسابقات رباتیکی که در کشور ما برگزار میشه، و لیگ ها و قوانین و ... اونها توضیح بدیم.

فدراسیون جهانی روبوکاپ، فقط ۲ رشته را به عنوان لیگ‌های رسمی در بخش دانش‌آموزی معرفی کرده است و مسابقات جهانی روبوکاپ هر ساله در  بخش دانش‌آموزی، فقط در همین ۲ لیگ برگزار می‌شود.

**۱- لیگ ربات‌های فوتبالیست**

**۲- لیگ ربات‌های امدادگر**

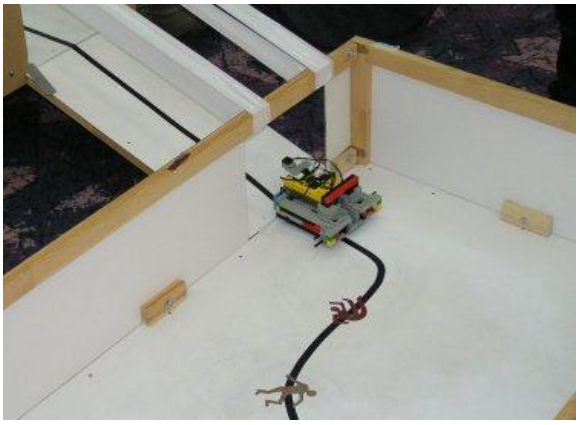
ما هم در این جلسه فقط در مورد همین ۲ لیگ توضیح خواهیم داد. اما این بدین معنا نیست که در کشور مسابقات دیگری در بخش دانش‌اموزی برگزار نمی‌شود، هر ساله در کشورِما ،مسابقات متعددی از جمله مسیریاب، آتش نشان، هزار تو(ماز)، جنگجو و ... در بخش های دانشجویی و دانش اموزی برگزار می شود، اما فدراسیون جهانی روبوکاپ، در بخش دانش آموزی فقط ۲ لیگ مذکور را به رسمیت می شناسد.

**Rescue junior league:**
**لیگ ربات های امدادگر دانش آموزی**

این لیگ شباهت زیادی به ربات های مسیریاب پیشرفته دارد. اما کمی پیچیده‌تر از آن است. ربات‌ها علاوه بر تعقیب خط مشکی رنگ، باید قادر باشند رنگ خطهای زمین مسابقه را بخوانند. مثلاً در بخشهایی از زمین، خطهای سبز یا قرمز یا نقره ای رنگ کشیده شده است، این خط‌ها همان مصدومین فرضی هستند و ربات باید قادر به تشخیص آن‌ها باشد.

همچنین در بخش‌هایی از مسیر، موانعی در مسیر حرکت ربات تعبیه شده است که ربات باید بتواند بدون برخورد با موانع، آن را رد کند. همچنین ممکن است در برخی از نقاط مسیر، شیب زمین به ۲۵ درجه نسبت به سطح افق نیز برسد.


در شکل زیر به مصدومین فرضی که به وسیله ی آدمک رنگی نشان داده شده اند دقت کنید.



این مسابقات در ۲ بخش Primary(مقدماتی)، و Secondary(پیشرفته) برگزار می‌شود. در بخش نخست، شرکت کنندگان نمی‌توانند سن بیش از ۱۴ سال داشته باشند. در بخش Secondary نیز، سن شرکت گنندگان نمی‌تواند بیش از ۱۸ سال باشد.

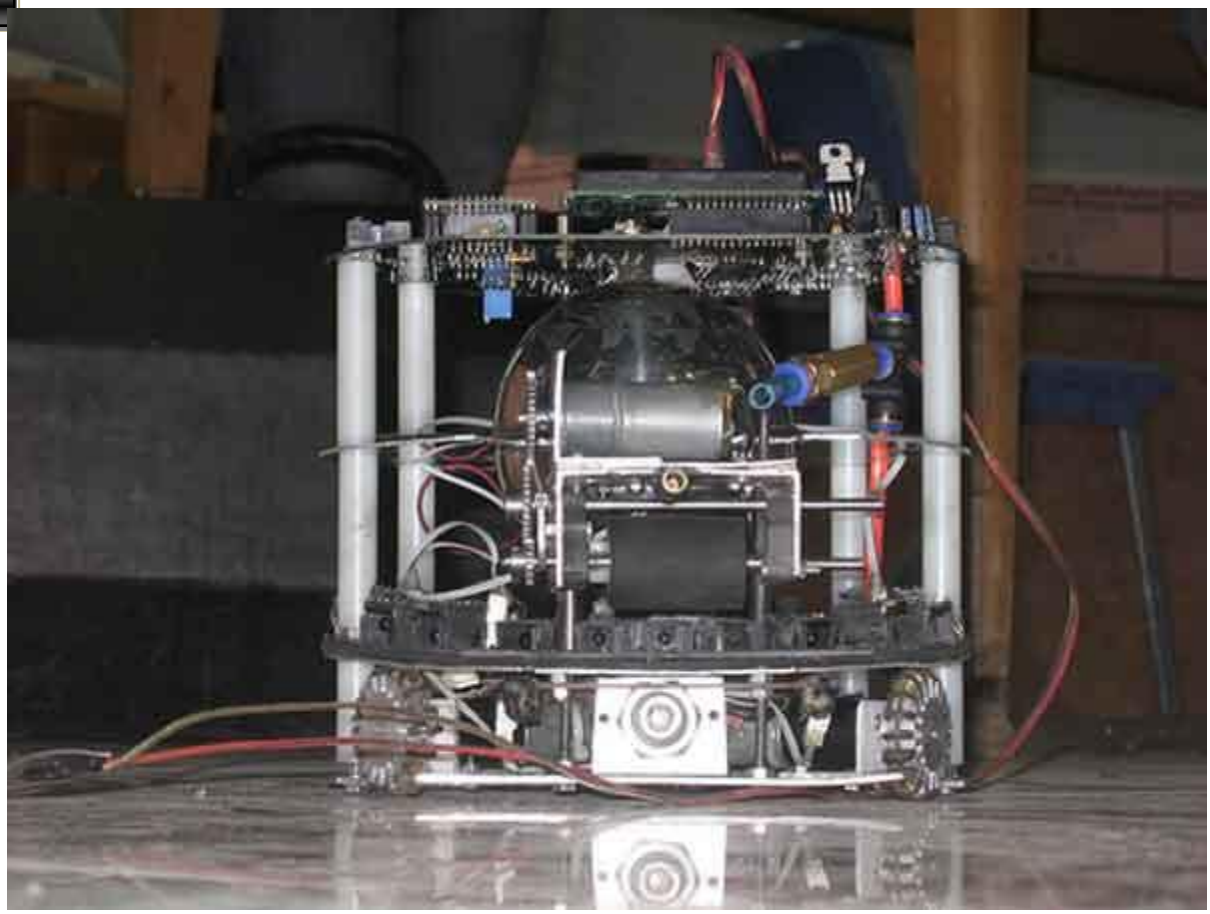
برای دریافت قوانین **اینجا** کلیک کنید.

**Soccer junior league:**
**لیگ ربات های فوتبالیست دانش آموزی**

در این لیگ، ربات‌ها باید بتوانند با توپ‌های خاصی که از خود مادون قرمز ساتع می کنند، فوتبال بازی کنند. دلیل استفاده از این نوع توپ خاص، ساده بودن روش‌های تشخیص آن به وسیله‌ی سنسورهای مادون قرمز معمولی(فتو ترانزیستورهایی که در مورد آن توضیح داده شده) است. یافتن این توپ‌ها توسط ربات، به سادگی یافتن آتش یا هر منبع نور دیگری است.

ربات‌های هر تیم باید تلاش کنند تا توپ مذکور را، در دروازه‌ی تیم مقابل جای دهند. در پایان زمان مسابقه، هرتیمی که گل بیشتری زده باشد، برنده‌ی بازی خواهد بود.

تیم‌های حرفه‌ای در این لیگ، سیستم‌های مختلفی را برای هدایت و شوت کردن توپ به سمت دروازه‌ی تیم مقابل بر روی ربات‌های خود تعبیه می کنند.



در کف زمین مسابقه، برای کمک به مکان یاب ربات‌ها، یک طیف رنگی از سیاه تا سفید، بین ۲ دروازه کشیده شده است. تیم‌ها با تشخیص رنگ کف زمین، می توانند مکان تقریبی خود را در زمین مسابقه به دست بیاورند. داشتن مختصات تقریبی، به ربات کمک می‌کند تا بتوانند استراتژی‌های کاملتری را در زمین مسابقه پیاده سازی کند.

این لیگ، در دو بخش ۲در۲ و ۱در۱ برگزار می‌شود. در لیگ ۲در۲، هر تیم می‌تواند حداکثر ۲ ربات در زمین مسابقه حاضر کند.
در بخش ۱در ۱ نیز، تیم‌ها فقط ۱ ربات می‌توانند در زمین مسابقه داشته باشند.

ابعاد زمین مسابقه، ۱۲۲ در ۱۸۳ سانتی‌متر است.

برای دریافت قوانین **اینجا** کلیک کنید.

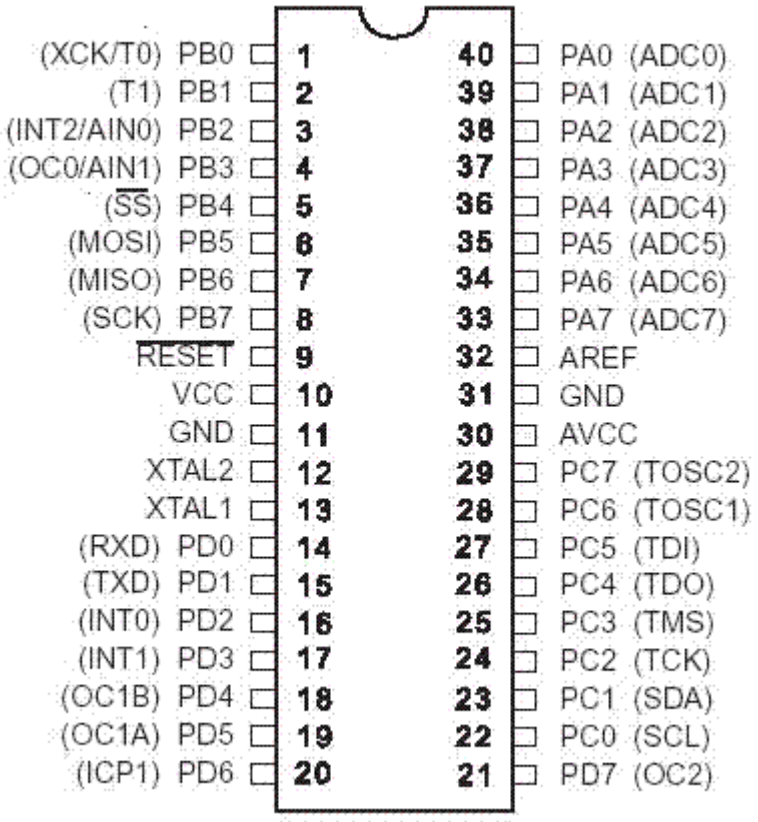
در جلسه ی آینده دوباره بر میگردیم سر مبحث میکروکنترلر و بحث را با مدارهای راه اندازی میکروکنترلر دنبال می کنیم.

**جلسه‌ی بیست و نهم**

مدارهای جانبی برای راه‌اندازیATMEGA16 ، اسیلاتور، مدار Reset و...

این جلسه همانطور که قبلاً گفته بودیم، سعی می‌کنیم کمی از مقدمات سخت افزاری و مدارهای راه اندازی میکروکنترلرهای AVR صحبت کنیم تا دوستان بتوانند به تدریج کار عملی با Atmega16 را شروع کنند.

در شکل زیر شمای کلی ATMEGA16 آورده شده است




**پایه‌ی ۱۰:** تغذیه‌ی آی سی است و باید به ۵ولت متصل گردد. ولتاژ تغذیه برای میکروکنترلرهای Atmega16، بین ۵.۵_۴.۵ ولت باید باشد، و برای Atmega16**L**، بین ۵.۵_۲.۷ ولت است.

**پایه‌های ۱۱ و ۳۱:** این ۲ پایه GND هستند و باید به قطب – منبع تغذیه متصل شوند.

**پایه‌ی ۳۰:** این پایه، تغذیه‌ی مبدل آنالوگ به دیجیتال است(ADC) و اگر بخواهیم از این امکان میکروکنترلرهای AVR استفاده کنیم، باید این پایه را به همان ۵ولت منبع تغذیه متصل کنیم.

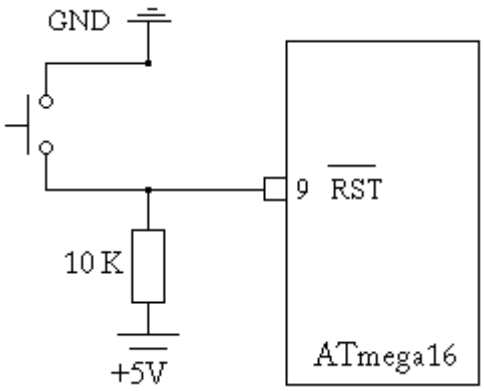
**پایه‌ی ۳۲**: این پایه نیز مربوط به همان امکان تبدیل آنالوگ به دیجیتال است، در مورد آن در جلسات آینده توضیح خواهیم داد. وقتی از این امکان استفاده نمی‌کنیم، نیازی نیست این پایه به جایی متصل باشد.

## مدار پایه‌ی:Reset

این پایه برای Reset کردن آی سی به کار می رود. Reset شدن میکروکنترلر مثل Reset شدن کامپیوتر است و باعث می‌شود که آی سی همه‌ی برنامه‌های خود را دوباره از اول اجرا کند.

این پایه باید در حالت عادی ۱ منطقی باشد و هرگاه بخواهیم آی‌سی را Reset کنیم، باید آنرا ۰ منطقی کنیم(حداقل ۱۶ میلی ثانیه) و سپس ۱ منطقی کنیم.

برای این پایه، می‌توان مدار زیر را بست.




در این مدار، پایه‌ی Reset به وسیله‌ی یک مقاومت ۱۰ کیلواهمی به VCC وصل شده است، و هر گاه کلید را فشار دهیم، پایه مستقیماً به GND وصل می‌شود و آی سی Reset می‌شود.

## اسیلاتور خارجی:

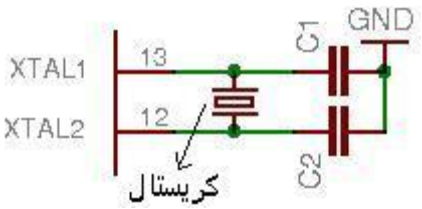
میکروکنترلر هم مثل کامپیوتر شما یک فرکانسِ کاری دارد، مثلاً وقتی می‌گویید CPU کامپیوتر شما ۲.۵ گیگا هرتز است، در حقیقت شما فرکانس کاری پردازنده‌ی کامپیوتر خود را گفته‌اید.

برای تولید این فرکانس، ما نیاز به یک نوسان ساز یا اسیلاتور داریم. این قطعه در اصطلاح تجاری به کریستال معروف است.

یکی از مزیت‌های Atmega16 این است که یک نوسان ساز در داخل خود میکروکنترلر تعبیه شده است و نیازی نیست شما از این کریستال‌ها استفاده کنید.

اما در Atmega16 این نوسان ساز دقت خوبی ندارد و برای کارهایی که نیاز به دقت بالا دارند(بعداً در این مورد توضیح خواهیم داد)، باید از کریستال یا نوسان ساز خارجی استفاده کرد. اما فعلاً برای کار ما نیازی به کریستال خارجی نیست.

پایه‌های ۱۲ و ۱۳ برای این منظور در نظر گرفته شده‌اند. برای اتصال کریستال به آی‌سی باید مدار زیر را که شامل ۲ عدد خازن عدسی ۳۰ پیکوفاراد است به این ۲ پایه متصل کنید.




دقت کنید که پایه‌های کریستال تفاوتی با هم ندارند و در نتیجه فرقی نمی‌کند از کدام طرف در مدار قرار گیرد.

(مثل LED مثبت و منفی ندارد)

برای میکروکنترلرهای ATMEGA16L، حداکثر از اسیلاتورهای ۸ مگا هرتز می‌توان استفاده نمود، اما برای ATMEGA16 می‌توان از ۱۲ یا ۱۶ مگاهرتز هم استفاده نمود.

یکی دیگر از ویژگی‌های میکروکنترلرهای AVR این است که برای پروگرام کردن آن‌ها نیازی به دستگاه‌پروگرامر نیست، و فقط با یک کابل ساده‌ی ۵ رشته می‌توان آن‌ها را به سادگی توسط کامپیوتر پروگرام کرد.

در جلسه‌ی آینده، نحوه‌ی ساخت این پروگرامر را برای میکروکنترلرهای خانواده‌ی AVR آموزش می‌دهیم.

**جلسه‌ی سی‌ام**

معرفی پروتکل STK200\300 برای پروگرام کردن میکروکنترلرهای خانوادهیAVR ، ساخت یک پروگرامر بسیار ساده و...

ابتدا باید یک نکته رو از مطالب جلسه‌ی پیش گوش زد کنم، برای بستن مدار Reset و همچنین کرستال خارجی، در ATMEGA16L هیچ الزامی وجود ندارد و صرفاً برای دقت بیشتر می‌باشند.
در ضمن یاد آوری می‌کنم که میکروکنترلرهای ATMEGA16L و ATMEGA16 تفاوت خاصی در ترتیب پایه‌ها و کارایی با یکدیگر ندارند. مهمترین تفاوت این ۲ آی سی در فرکانس کاری این ۲ آی سی است که ATMEGA16L نمی‌تواند با فرکانس بیش از ۸ مگاهرتز کار کند.
خوب، همانطور که قول داده بودیم، قراره این جلسه ساخت یک پروگرامر بسازیم که بتوانیم به وسیله‌ی آن، برنامه‌هایی که در کامپیوتر می‌نویسیم را به میکروکنترلر منتقل کنیم. برای پروگرام کردن میکروکنترلرهای خانواده‌ی AVR روش‌ها و پورتکل‌های متعددی وجود دارد. یکی از معروفترین و پرکاربردترین پروتکل‌های موجود، STK۲۰۰\۳۰۰ نام دارد که ما در این جلسه سعی می‌کنیم نحوه‌ی استفاده از این پروتکل را آموزش دهیم.

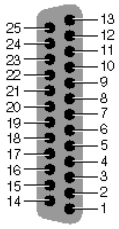
همانطور که گفته شد، میکروکنترلرهای خانواده‌ی AVR این قابلیت را دارند که می‌توان آن‌ها را مستقیماً به وسیله‌ی یک کابل ۵ رشته به کامپیوتر متصل نموده و پروگرام کرد، و در نتیجه، نیازی به یک دستگاه مجزا برای پروگرام کردن ندارند. این روش پروگرام کردن STK۲۰۰/۳۰۰ نام دارد. این روش، به خاطر عدم نیاز به هرگونه مدار جانبی و سهولت کار با آن، از محبوبیت زیادی در بین کاربران حرفه‌ای برخوردار است.

اولین نکته این است که اگر کامپیوتر شما پورت LPT(موازی) نداشته باشد، شما نمی توانید به این روش(یعنی فقط با یک کابل ۵ رشته‌ی ساده) میکروکنترلر خود را پروگرام کنید و باید از مدارهای پروگرامر USB استفاده کنید. با استفاده از پروگرامرهای USB، شما می‌توانید با استفاده از درگاه USB هم میکروکنترلر خود را پروگرام کنید. ساختن این پروگرامرها کار ساده‌ای نیست، اما انواع مختلف آن‌ها در بازار موجود است که بین ۲۰ تا ۲۰۰ هزار تومان هم قیمت دارند.

برای دیدن پورت یا درگاه LPT (موازی) ، به پشت کیس کامپیوتر خود نگاه کنید.

این درگاه، درگاه ارتباط موازی(Parallel) یا LPT نام دارد و یکی از رایج‌ترین کاربردهای آن برای ارتباط با پرینتر است. البته اکثر پرینترهای امروزی از طریق درگاه USB با کامپیوتر ارتباط برقرار می‌کنند.

درگاه LPT دارای ۲۵ پایه است که به شکل زیر شماره گذاری می‌شوند.




برای ساختن این پروگرامر ساده، شما احتیاج به نیم متر کابل فِلَت ۵ رشته و یک عدد سوکت نَری LPT دارید. البته الزامی در استفاده از این نوع کابل نیست و می‌توان از هر کابل ۵ رشته‌ی دیگری برای این منظور اسفاده کرد.

سوکت نَری LPT.

کابل فلت نیز در شکل زیر نشان داده شده است.

شما باید این ۵ رشته را به پایه‌های شماره‌ی ۶و۷و۹و۱۰و۲۴ از این سوکت لحیم کنید.

حالا می‌توانید به وسیله‌ی این ۵ سیم میکروکنترلر خود را پروگرام کنید. کافیست این سیم‌ها را به ترتیب زیر به پایه‌های میکروکنترلر وصل کنید.

سیمی که به پایه‌ی شماره‌ی ۶ سوکت متصل شده است، باید به پایه‌ی SCK در میکروکنترلر شما وصل شود. در SCK, Atmega16 پایه‌ی شماره‌ی ۸ است.

پایه‌ی شماره‌ی ۷ سوکت، باید به پایه‌ی MOSI در میکروکنترلر وصل شود. در MOSI ,Atmega16 پایه‌ی شماره‌ی ۶ است.

پایه‌ی شماره‌ی ۹ سوکت، باید به پایه‌ی Reset در میکروکنترلر وصل شود. در Reset Atmega16 پایه‌ی شماره‌ی ۹ است.

پایه‌ی شماره‌ی ۱۰ سوکت، باید به پایه‌ی MISO در میکروکنترلر وصل شود. درMISO,Atmega16 پایه‌ی شماره‌ی ۷ است.

و در نهایت، پایه‌های شماره‌ی ۱۸تا ۲۵ نیز، باید به GND یا همان زمین در میکروکنترلر وصل شود. پایه‌ی ۱۱و ۳۱ در

Atmega16L ، - یاهمان GND است.بهتر است برای اتصال این پایه به میکروکنترلر، از یک مقاومت ۱ کیلو اهم استفاده کنید.

در ضمن دقت کنید، که اگر طول سیم بیش از نیم متر باشد، ممکن است در پروگرام کردن دچار مشکل شوید، به همین خاطر بهتر است تا جای ممکن طول سیم را کوتاه انتخاب کنید.

جلسه‌ی آینده در مورد نحوه‌ی انجام تنظیمات مربوط به پروگرامر را در CodeVision نیز توضیح خواهیم داد.

پایان فصل سوم

گرد آورنده و طراح : مهندس تالیا براری