« به نام خدا »

با عرض سلام خدمت همه ی دوستان عزیز

✓ **استفاده از L298 در ربات مسیریاب ساده ، ULN2003 ، یک ترفند در مسیریابی...**

همونطور که در جلسه ی پیش گفتیم ، این جلسه ابتدا سعی می کنیم از L298 در راه اندازی موتور ربات استفاده کنیم .

نحوه ی استفاده از L298 در یک ربات مسیریاب ساده

همونطور که در جلسات پیش توضیح داده شد ، یک آی سی L298 قابلیت راه اندازی ۲ موتور به صورت همزمان را دارد . البته L298 یک درایور موتور نسبتاً حرفه ایست و در این ربات ما ضرورتی در استفاده از این آی سی نیست ، و این مطالب بیشتر جنبه ی آموزشی دارد ، یعنی هدف ما اینه که دوستان کاربرد عملی این آی سی را در ربات ببینند .

در این آی سی برای هر موتور ۲ ورودی و ۲ خروجی وجود دارد . ۲ پایه ی خروجی را که مستقیماً به پایه های موتور متصل می کنیم

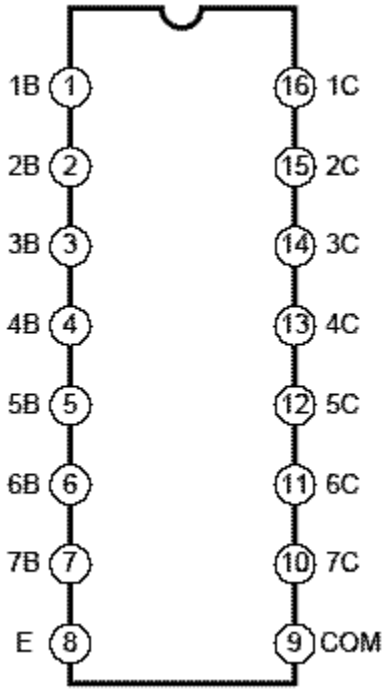
اما ۲ پایه ی ورودی هر موتور ...

در اینجا هم یکی از ۲ پایه ی ورودی را مستقیماً به - متصل کرده و پایه ی دیگر را به خروجی AND متناظر موتور وصل می کنیم .

برای موتور دیگر هم دقیقاً همین روند را تکرار می کنیم ، یعنی ابتدا خروجی ها را به موتور متصل کرده و سپس ورودی ها را یکی به – و دیگری به خروجی AND متناظر وصل می کنیم .

آی سی ULN2003

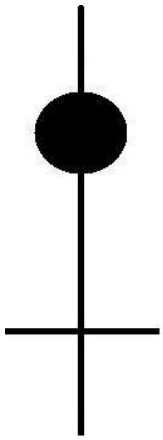
این آی سی نیز عملکردی شبیه بافر ۷۴۲۴۵ دارد ، با این تفاوت که اصطلاحاً (Open collector) است ، یعنی در هیچ شرایطی آی سی خروجی + نمی دهد یا صفر یا منفی در خروجی داریم . درست مثل یک ترانزیستور npn که روی کلکتور یا صفر داریم (gnd) – ترانزیستور روشن – و یا ول داریم – ترانزیستور خاموش – .



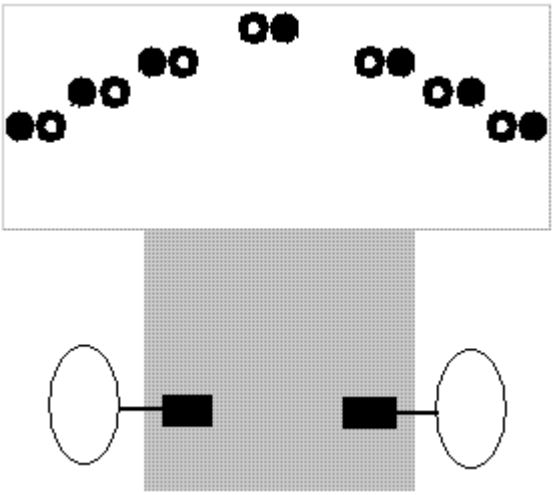
در این آی سی پایه ۸ باید به – یا همان GND متصل شود و پایه ی ۹ نیز به ولتاژ مورد نظر ما برای خروجی ها .
پایه های سمت چپ ، ورودی ها ، و پایه های سمت راست خروجی های آی سی هستند .

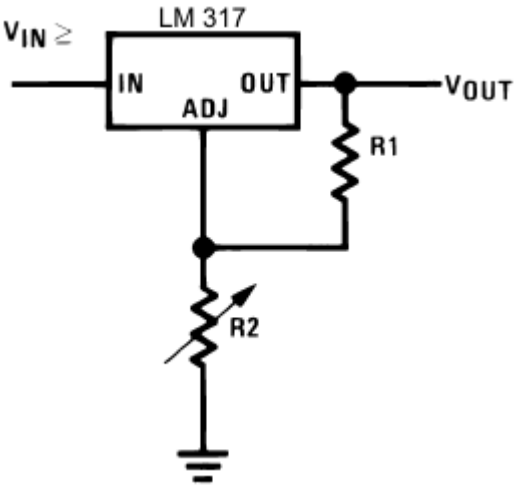
یک ترفند در مسیریابی

در مسابقات مسیریاب ، گاهاً در مسیر مسابقه خط هایی به صورت عمود به خط اصلی ، و یا دایره ی سیاه رنگ در بعضی قسمتهای مسیر قرار می دهند و ربات باید بتواند بدون توجه به آنها ، مسیر اصلی را دنبال کند .



برای حل ای مشکل ، یک سنسور دقیقاً در وسط سنسورهای ۲ طرف به گونه ای تعبیه می کنند که وقتی ربات دقیقاً روی خط قرار دارد ، این سنسور خط را ببیند . حال مدار را به گونه ای طراحی می کنند که وقتی سنسور وسط روی خط است ، بدون توجه به بقیه ی سنسورها ، ربات به سمت جلو حرکت کند . البته پیاده سازی این روش روی ربات ما کار خیلی ساده ای نیست و این روش برای ربات های مسیر یاب حرفه ای میکروکنترلر دار (دارای برنامه نویسی) است .

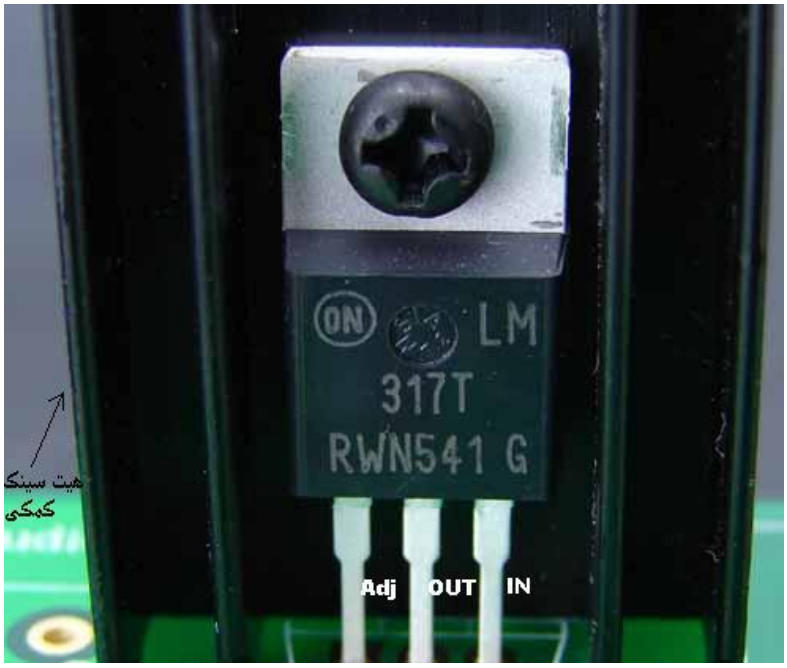




رگولاتورهایی که ما تا به حال با آن ها آشنا شده‌ایم همگی ولتاژ خروجی ثابتی داشتند، مثلاً ۷۸۰۵ خروجی ثابت ۵ ولت به ما می‌دهد و ۷۸۰۹ خروجی ثابت ۹ ولت .

اما با رگولاتور LM317 و به کمک یک مقاومت ثابت و یک پتانسیومتر ، می توانیم سطح ولتاژ خروجی را به دلخواه خود تنظیم کنیم . درست مثل رگولاتورهای بالا سطح ولتاژ خروجی این رگولاتور نیز از ولتاژ ورودی حداقل حدود یک ولت کمتر است .

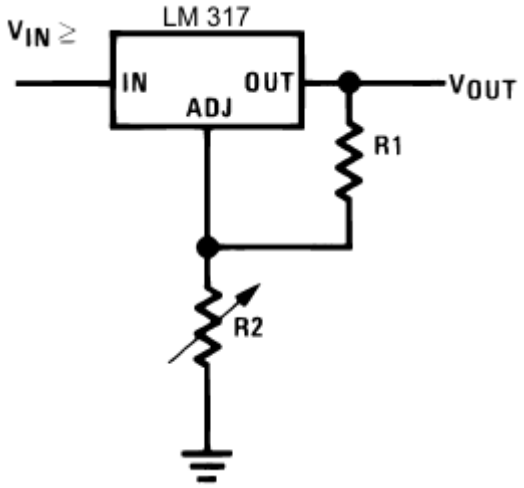
ترتیب پایه های LM317 در شکل زیر نشان داده شده است .



همان‌طور که در شکل می‌بینید ، برای پایین آوردن دمای آی سی در مدارهایی که نیاز به جریان دهی بالا دارند ، علاوه بر هیت سینک (صفحه فلزی سفید) خود آی سی ، از یک هیت سینک کمکی بزرگتر نیز استفاده 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 .

توصیه : برای انتقال بهتر حرارت ، بین هیت سینک آی سی و هیت سینک کمکی از ورق یا خمیر سیلیکون استفاده شود .

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



در مدار بالا ، R1= ۴۷۰Ω است و R2 ، یک پتانسیومتر یا مولتی‌ترن ۱۰KΩ .

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

برای محاسبه‌ی سطح ولتاژ خروجی ، فرمول زیر وجود دارد :

$$V = 1.25\ (1 + (R_2 \div R_1)) + I_{adj} \times R_2$$

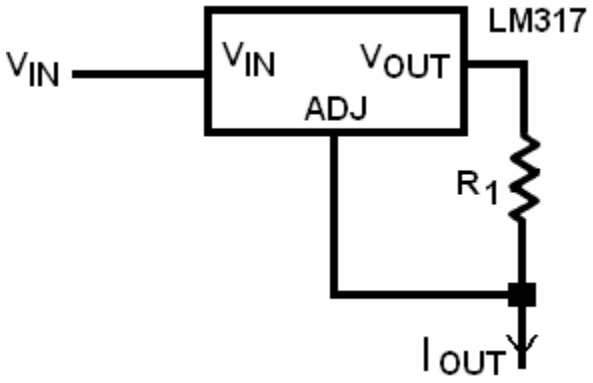
در فرمول بالا $I_{adj} = 100\ \mu A$ که بسیار ناچیز و قابل چشم پوشی است . بنابراین :

$$V = 1.25\ (1 + (R_2 \div R_1))$$

طبیعتاً نیازی نیست شما هردفعه برای محاسبه‌ی ولتاژ خروجی از این فرمول استفاده کنید ، شما می‌توانید با چرخاندن پیچ پتانسیومتر ، 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 است .

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

به مدار دقت کنید :

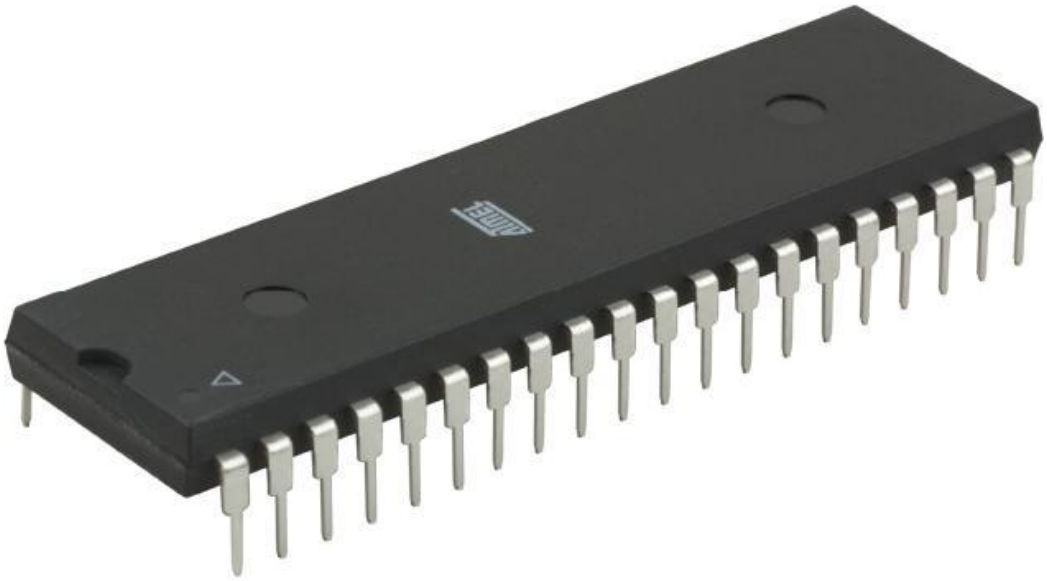


به وسیله‌ی رابطه $I_{out}=Vin \div R_1$ می‌توان میزان جریان خروجی را حساب کرد . البته این مدار کاربرد بسیار کمی دارد ، و برای کنترل جریان در مدارهای ساده ، معمولاً از مقاومت‌های معمولی استفاده می‌کنیم .

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

میکروکنترلر 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 (مغزِ ربات) . 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 میکروکنترلر 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 میکروکنترلرهای AVR و ساخت شرکت اَتمِل (Atmel) است و به دلیل ویژگی های خاص و قیمت مناسبش (حدوداً ۴۰۰۰ تومان) ، به عنوان یکی از پرکاربردترین و معروف ترین انواع میکروکنترلرهای آموزشی 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 میکروکنترلر ATmega16L را آموزش ببینیم فقط می توانیم فقط با همین میکروکنترلر کار کنیم ، بلکه کار

کردن با سایر میکروکنترلرهای خانواده 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 .

میکروکنترلر یک ریز پردازنده (Processor) است که می تواند ورودی و خروجی های متعدد داشته باشد . یعنی تعدادی 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 ، یک Object 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 Chip یا Device (قطعه) ، 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 به ما طی 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 قالب یک نرم افزار به نام "Code Vision" توسط شرکت HP به بازار عرضه شده است ، البته کدویژن تنها یک نرم افزار است و چندین نرم افزار دیگر هم با همین مشخصات وجود دارد . کاربر با نصب این نرم افزار بر روی کامپیوتر شخصی خود ، در حقیقت هر ۳ برنامه را ، به علاوه 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 .

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 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 یا تهیه 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 .

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

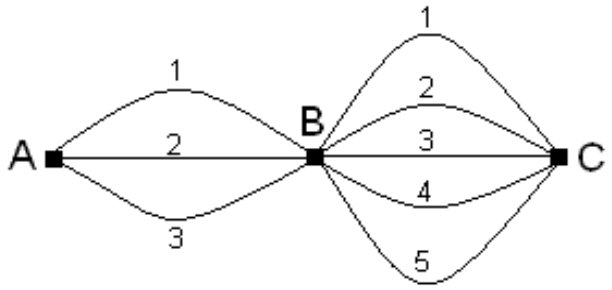
همانطور که می دانید، کوچک ترین واحد ذخیره سازی اطلاعات در حافظه، Bit است
هر 8 بیت را یک Byte می گویند. در حقیقت یک بایت اطلاعات، 8 تا ، 0 یا 1 است که در مجموع 256 حالت مختلف را پدید می آورند.

توضیح بیشتر :

یک بیت، فقط 2 حالت دارد، 0 یا 1. وقتی 2 بیت در کنار هم قرار می گیرند، هر کدام 2 حالت را پدید می اورند و در مجموع طبق اصل ضرب ، 4 حالت به وجود می آید . یعنی:

1 و 1   0 و 1
1 و 0   0 و 0

به مثال زیر توجه کنید.



در شکل بالا، برای رفتن از A به B، 3 مسیر وجود دارد؛ 5 مسیر هم برای رفتن از B به C وجود دارد. طبق اصل ضرب، برای رفتن از A به C مجموعاً 15=5*3 حالت وجود دارد .

در اینجا، در حقیقت تعداد کل حالت ها ، برابر است با حاصل ضرب حالت های هر بیت (2 حالت) می باشد . به عنوان مثال برای محاسبه تعداد حالت های 3 بیت اطلاعات، داریم: 8=3^2=2*2*2 .

طبق همین رابطه، یک بایت ، 256=8^2 حالت مختلف می تواند باشد .

هر 1024 بایت را 1 **کیلوبایت** می گویند و هر 1024 کیلو بایت ، یک **مگابایت** است . هر 1024 مگا بایت هم یک **گیگابایت** و هر 1024 گیگا بایت هم یک **ترابایت** نام دارد. (1024=10^2)

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 256 کاراکتر (یک کاراکتر عبارتست از یک عدد، رقم یا یک علامت مثل + و -) را کد گذاری کرد و به هر کدام یک عدد 8 رقمی در مبنای 2(یعنی یک بایت) نسبت داد. این کارکترها شامل همه ی حروف الفبای لاتین، اعداد 0 تا 9، 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 می یابد. مثلاً برای ذخیره سازی کلمه ی"ALI" به 3 بایت حافظه نیاز داریم . جدول کدهای ASCII را می توانید در کتاب های برنامه نویسی یا با جستجو در اینترنت به راحتی ببینید .

انواع زبان های برنامه نویسی

زبان ماشین :

پایین ترین سطح زبان برنامه نویسی زبان ماشین است . در این زبان شما باید به جای گذاشتن علامت + برای جمع کردن مقدار 2 عدد ، باید از کد 00 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 کامپایلر

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 2 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. در زبان 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.

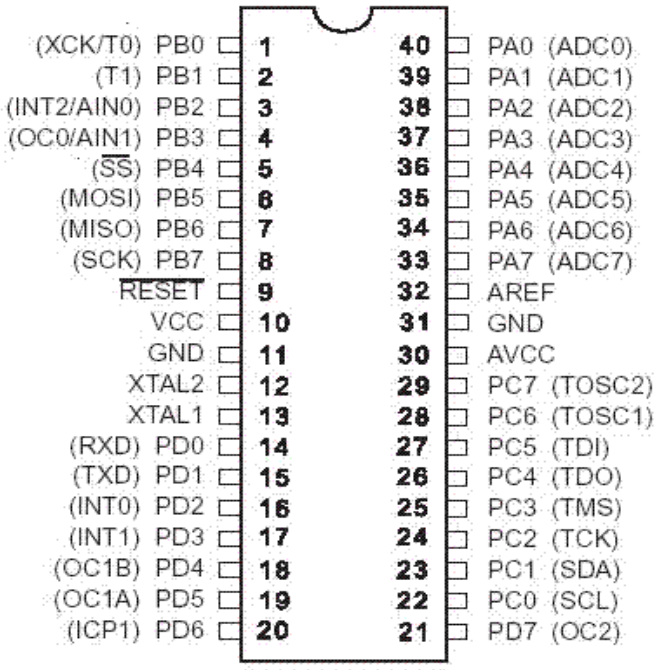
در زبان C وجود تابعی با نام main الزامیست. یعنی ما باید حتماً تابعی با نام main در برنامه ی خود داشته باشیم و اجرای برنامه هم از تابع main شروع می شود.

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.

رجیستر چیست ؟ رجیستری‌های PORTx, PINx , DDRx ، 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...



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

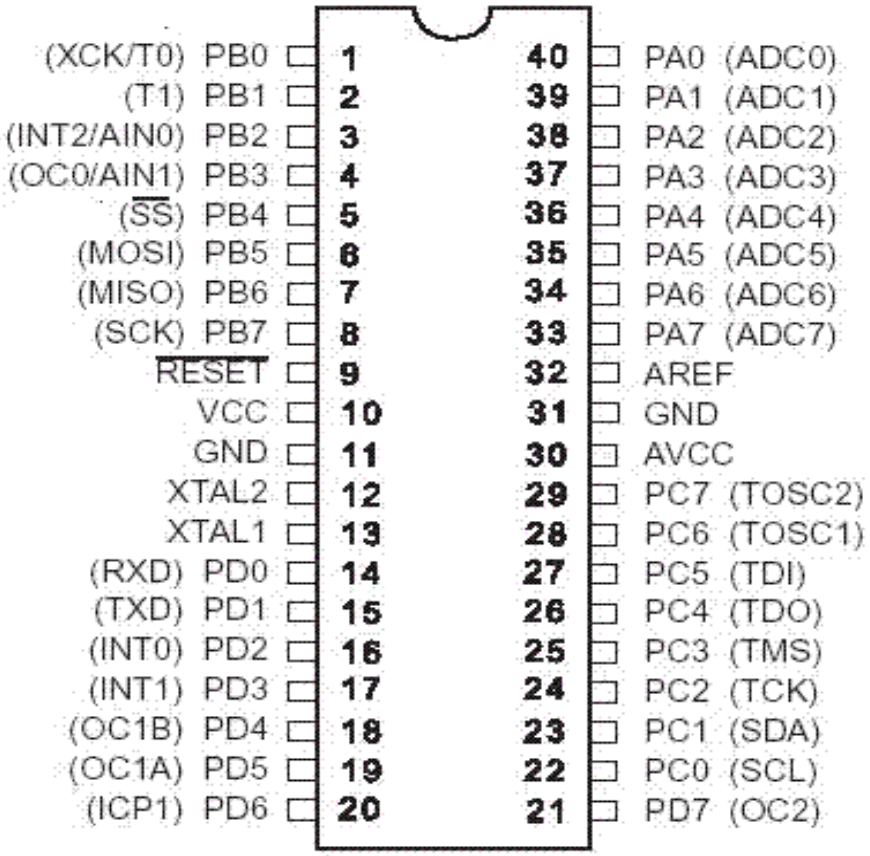
همونطور که گفته شد AT Mega۱۶ دارای پایه های متعددی برای تبادل اطلاعات با مدار است. هر ۸پایه ی مجاور که این وظیفه را دارند یک پورت نامیده می شوند (به شکل نگاه کنید) . AT Mega۱۶ دارای ۴ پورت با نام های A ، B، C و D می باشد. پایه های هر پورت به این شکل نمایش داده می شود :

شماره ی پایه . نام پورت

مثلاً اولین پایه ی پورت D به این صورت نشان داده می شود: D.۰

و پایه ی سوم پورت C به صورت : C.۲

حال به ترتیب پایه های ATMEGA۱۶L 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 نوعی حافظه هستند که به طور مستقیم با بخش 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.۳) 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.۳ خروجی باشد باید این جمله را بنویسیم: DDRD.۳=۱; و اگر بخواهیم این پایه ورودی باشد: DDRD.۳=۰;

رجیستر PORTx :

در صورتی که پایه ها به صورت **خروجی** تنظیم شده باشند، هر چه در این رجیستر نوشته شود سطح منطقی پایه متناظر را تعیین می کند، مثلاً اگر بنویسیم PORTB.۳=۱ 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.۱=۰، 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.۱=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 Wizard Code 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 wizard Code 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.۵را به صورت خروجی (برای کنترل حرکت موتورها) تنظیم کرده می کنیم.

B۲ و B.۳ برای کنترل موتور سمت راست و B.۰ برای سنسور سمت راست .

B.۴ و B.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 دیگر را هم به صورت هماهنگ با سنسور متناظر آن سمت ۰ و ۱ می کنیم، یعنی اگر خروجی سنسور ۰ بود، پایه 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

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 B.۳ نیز ۱ منطقی می شود.

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

```
PORTB.۲=۰;

PORTB.۴=۰;

PORTB.۳=PINB.۰;

PORTB.۵=PINB.۱;
```

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 .