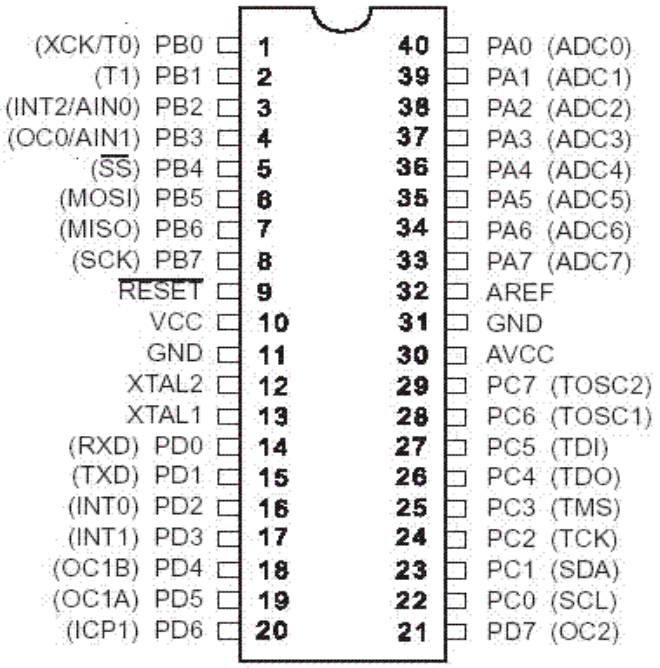
«به نام خدا»

با عرض سلام خدمت همه ی دوستان عزیز

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

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 .

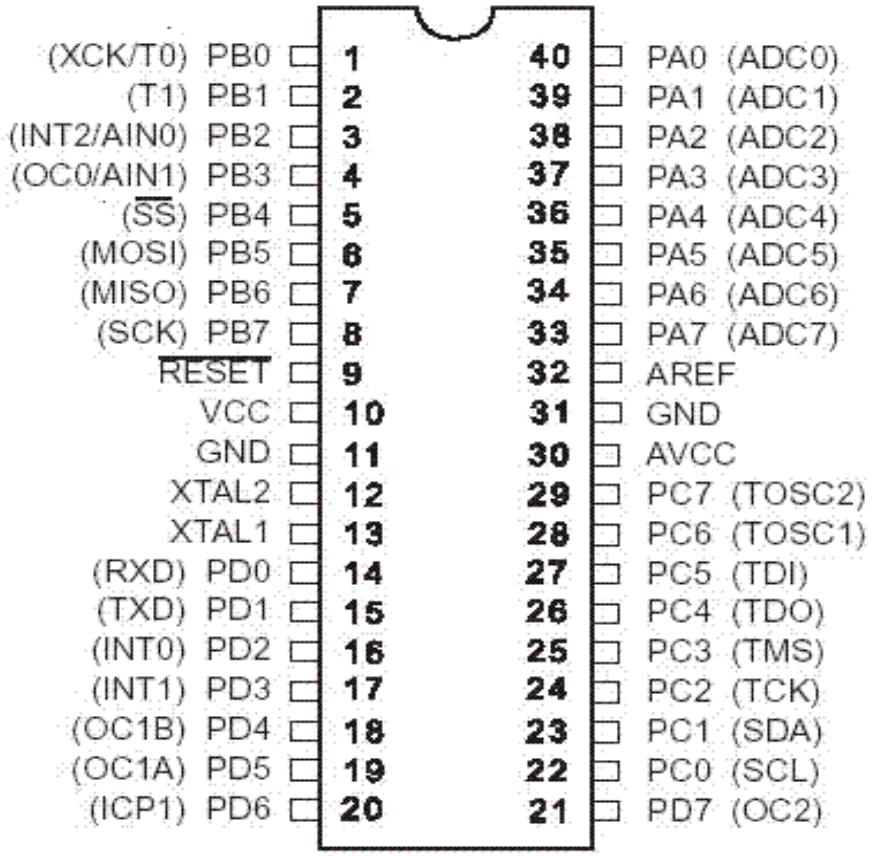
همونطور که گفته شد AT Mega۱۶ دارای پایه های متعددی برای تبادل اطلاعات با مدار است. هر ۸پایه ی مجاور که این وظیفه را دارند یک پورت نامیده می شوند (به شکل نگاه کنید) . AT Mega۱۶ دارای ۴ پورت با نام های A ، B، C و D می باشد. پایه های هر پورت به این شکل نمایش داده می شود :

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(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 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 .

آموزشهای رباتیک طبقه بندی شده توسط کمیته مهندسی رباتیک / **nrec.ir** (طرح ساماندهی آموزش رباتیک در اینترنت) برگرفته از سایت رشد مخصوص رده سنی ۱۳ تا ۲۵ سال

گردآوری و ویرایش اولیه : خانم فرناز عطاءالهی

ویرایش علمی و گرافیکی نهایی : زهره دارابیان

فـــروشگاه عرضه قطعات الکترونیک ، مکانیک و رباتیک

RoboChip.ir