

به نام خدا

در این جلسه نیز مبحث PWM را دنبال می‌کنیم . برای درک مطالب این جلسه حتماً باید جلسه قبل را مطالعه کرده باشید . در این جلسه به آبی‌سی ۲۹۸ نیز اشاراتی شده است که دوستان می‌توانند جهت یادآوری ، بخش مربوط به ۲۹۸ در جلسه ی ۱۷ را نیز مرور کنند .

ابتدا با توابعی که برای ایجاد وقفه در اجرای دستورات برنامه توسط CodeVision برای کاربران در نظر گرفته شده آشنا می‌شویم.

همان‌طور که در جلسه پیش دیدیم ، در قسمت‌هایی از برنامه ممکن است نیاز پیدا کنیم تا برای لحظاتی روند اجرای دستورات را متوقف کنیم . CodeVision برای این کار توابعی را از پیش تنظیم کرده است . (در مورد مبحث توابع در زبان C در آینده مفصل توضیح خواهیم داد)

Delay دستوری برای توقف یا تاخیر

برای ایجاد تاخیر در روند اجرای دستورات ، CodeVision دو تابع زیر را در اختیار ما قرار داده است.

```
delay_ms( );
delay_us( );
```

تابع delay_ms() برای ایجاد تاخیرهایی در حد میلی ثانیه به کار می‌رود. در داخل پرانتز، یک عدد صحیح مثبت می‌نویسیم که نشان دهنده اندازه تاخیر مورد نیاز ما بر حسب میلی ثانیه است. به بیان ساده‌تر، مثلاً اگر داخل پرانتز عدد ۱۰۰ را بنویسیم، روند اجرای برنامه به اندازه ۱۰۰ میلی ثانیه در همان خط متوقف خواهد شد.

تابع delay_us() برای ایجاد تاخیرهایی در حد میکروثانیه به کار می‌رود. نحوه‌ی استفاده از آن دقیقاً مانند delay_ms() است.

به عنوان یک مثال عملی ، همان برنامه ایجاد PWM ۲.۵ ولت را با استفاده از توابع delay بازنویسی می‌کنیم.

```
while(۱)
{
PORTB.۴=۱;
delay_ms(۵); // ۵ milliseconds delay
PORTB.۴=۰;
delay_ms(۵); // ۵ milliseconds delay
}
```

تنها نکته بسیار مهم در استفاده از توابع delay اضافه کردن `#include<delay.h>` دقیقاً بعد از `#include<mega16.h>` در ابتدای برنامه ی شماست بدون اضافه کردن `#include<delay.h>` شما نمی‌توانید از دستورات delay استفاده کنید . دستور `#include` به معنی اضافه کردن و `delay.h`

به معنی دستورات و امکانات فایل delay.h است که در این خصوص در آینده بیشتر صحبت می کنیم ولی بیاید در حال حاضر از بحث اصلی منحرف نشویم.

دقت کنید که این دستور نیازی به « ; » ندارد !

با آموختن تابع delay ، می‌توانید هر نسبت ولتاژ یا PWM که می‌خواهید بر روی پایه‌های خروجی ایجاد کنید . به عبارت دیگر اگر نیاز به ۲.۵ ولت برای کمرنگ روشن کردن یک چراغ داشتید کافی است یک pwm ۵۰ درصد روی پایه ی متصل به چراغ بدهید در آن صورت طبق مطالب درس قبل رفتار چراغ شبیه به این است که ۲.۵ ولت به آن وصل شده.

همانطور که می‌دانید موتورهای متعرفی که برای ساخت ربات‌ها استفاده می‌شود ، ممکن است ولتاژهای کاری مختلفی داشته باشند (مثلاً ۱۲ ولت، ۲۴ ولت، ۶ ولت و ...) و برای راه‌اندازی آن‌ها باید از درایورهای موتور مثل L۲۹۸ استفاده کنیم. سوالی که ممکن است پیش آید این است که وقتی ما میکروکنترلر را به درایورهای موتور (مثل L۲۹۸) وصل می‌کنیم و از تکنیک PWM برای کنترل سرعت موتور استفاده می‌کنیم، چه وضعیتی پیش می‌آید؟ مثلاً وقتی ما PWM مربوط به ولتاژ ۲.۵ ولت را تولید می‌کنیم، درایور ما چه عکس‌العملی نشان می‌دهد؟ آیا ولتاژ ۲.۵ ولت بر روی پایه‌های موتور قرار می‌گیرد؟

برای پاسخ به این سوال باید به ساختار PWM دقت کنیم، ما وقتی PWM مربوط به ۲.۵ ولت را تولید می‌کنیم، در حقیقت سطح ولتاژ خروجی را با فواصل زمانی برابر ۰ و ۱ می‌کنیم، پس اگر این خروجی را، به ورودی L۲۹۸ وصل کنیم (مثلاً پایه ۷)، L۲۹۸ نیز موتور را با همین الگو کنترل می‌کند و ولتاژی که به موتور می‌دهد را ۰ و ۱ می‌کند. و همانطور که می‌دانید، L۲۹۸ هر ولتاژی که بر روی پایه‌ی شماره‌ی ۴ آن قرار گرفته باشد را بر روی موتور قرار می‌دهد (اگر ولتاژ کاری موتور ۱۲ ولت باشد، باید این پایه به ۱۲ ولت متصل شود). پس جواب سوال بالا منفی است ! وقتی ما PWM مربوط به ۲.۵ ولت را تولید می‌کنیم، در حقیقت سطح ولتاژ خروجی در ۵۰ درصد زمان ۱ و بقیه‌ی زمان ۰ است. پس اگر همان طور که در بالا اشاره شد، این PWM به درایوری مثل L۲۹۸ داده شود، و ولتاژ پایه‌ی ۴ آن ۱۲ ولت باشد، درایور، ولتاژ ۶ ولت را به موتور می‌دهد. در نتیجه اهمیتی ندارد چه ولتاژی بر روی پایه‌ی ۴ L۲۹۸ قرار گرفته باشد، وقتی که ما PWM مربوط به ۲.۵ ولت را تولید می‌کنیم، درایور ولتاژی که به موتور می‌دهد را ۵۰ درصد می‌کند. در نتیجه بهتر است از این به بعد به جای آن که بگوییم PWM مربوط به ۲.۵ ولت، بگوییم PWM پنجاه درصد. یا به جای PWM مربوط به ۱ ولت، بگوییم PWM بیست درصد .



بسته آموزشی تازه کار ، یک مجموعه از قطعات مکانیک و الکترونیکه که علاقمندان و تازه کاران دنیای رباتیک می تونن با مونتاژ قطعاتش به شکل های مختلف مکانیزم ها و بدنه های مختلفی رو برای ربات هاشون درست کنن . این بسته رو به کسانی که دوست دارنند برای اولین بار رباتیک رو تجربه کنن پیشنهاد می کنم .

ROBOCHIP.IR

PWM در میکروکنترلرهای AVR

انجام تنظیمات اولیه برای استفاده از PWM برای راه اندازی موتور در میکروکنترلرهای AVR کمی پیچیده است، اما در اینجا هم CodeWizard به کمک ما آمده است و کار را کمی ساده تر کرده است. ما در جلسه آینده بخشی از تنظیمات CodeWizard را بدون توضیح مطرح می نمایم، زیرا توضیح هر بخش از آن نیازمند مقدمات مفصلی است و تاثیر چندانی هم در روند کار ما ندارد، اما به دوستانی که می خواهند میکروکنترلر را کاملاً حرفه ای دنبال کنند، پیشنهاد می کنم از منابعی که قبلاً معرفی شده است، مطالب را تکمیل کنند .

به هر حال دوستان عزیز با انجام این تنظیمات اولیه مختصر در CodeWizard، می توانند از الگویی به مراتب ساده تر از آنچه تا به حال آموخته ایم، برای ایجاد PWM برای هدایت موتورهای ربات استفاده نمایند.

در میکروکنترلرهای خانواده AVR ، نیازی نیست در هر بار استفاده از PWM، چندین خط برنامه بنویسیم. در ATmega16 چهار پایه مشخص از آی سی به این موضوع اختصاص داده شده

است. یعنی این چهارپایه علاوه بر کاربردهای معمولی خود، این قابلیت را دارند که در مواقع لزوم برای تولید PWM استفاده شوند.

حال سوال اینجاست که این چهارپایه چه تفاوتی با بقیه پایه‌های خروجی آی‌سی دارند که آن‌ها را از سایر پایه‌های خروجی میکروکنترلر متمایز می‌سازد؟

برای این چهارپایه نیازی به اجرای الگویی که تا به حال برای ایجاد PWM فراگرفته‌اید نیست. در این روش، فقط شما باید یک عدد صحیح بین ۰ تا ۲۵۵ انتخاب کنید، و طبق الگویی زیر آن را در برنامه‌ی خود بنویسید.

یک عدد صحیح بین ۰ تا ۲۵۵ = نام رجیستر مربوطه;

این عدد، بیانگر توان PWM شماست، و شما توان PWM مورد نیاز خود را با این عدد مشخص می‌کنید. که ۲۵۵ بالاترین توان و مربوط به PWM ۱۰۰ درصد است، و ۰ پایین‌ترین توان و مربوط به PWM ۰ درصد است. به عنوان مثال اگر این عدد را ۱۲۸ قرار دهید، همان PWM ۵۰ درصد را ایجاد کرده‌اید. یا مثلاً اگر این عدد ۵۱ باشد، PWM ۲۰ درصد بر روی پایه قرار داده‌اید.

رجیسترهای مربوط به این ۴ پایه

همانطور که می‌دانید، برای پایه‌هایی که در CodeWizard به صورت خروجی تعریف شده‌اند، رجیستری به نام «PORTx» وجود دارد که هر مقداری در این رجیستر قرار داده شود، مقدار پایه‌های خروجی متناظر با آن رجیستر را مشخص می‌کند. (رجوع به جلسه‌ی ۲۴، تعریف رجیستر PORTx)

در این جلسه با ۴ رجیستر دیگر آشنا می‌شویم، که وقتی تنظیمات مربوط به PWM موتور در CodeWizard را انجام دهیم، هر مقداری که در آن‌ها ریخته شود، توان PWM پایه‌ی متناظر را مشخص می‌کنند.

این رجیسترها OCR₀، OCR_{1A}، OCR_{1B} و OCR₂ نام دارند که به ترتیب، متناظر پایه‌های PB.۳، PD.۵، PD.۴ و PD.۷ هستند.

پس مثلاً اگر در بخشی از برنامه‌ی خود بنویسیم:

OCR₀ = ۱۲۷;

در حقیقت بر روی پایه PB.۳ میکروکنترلر، PWM ۵۰ درصد به وجود آورده‌ایم.

به مثال‌های دیگری توجه کنید : (توضیح هر دستور در جلوی دستور و بعد از // آورده شده است)

OCR1AL=۵۱; // ۲۰% Duty Cycle on PD.۵

OCR1BL=۲۵۵; //۱۰۰% Duty Cycle on PD.۴

OCR2=۰; //۰% Duty Cycle on PD.۷

در جلسه‌ی آینده، در مورد نحوه انجام تنظیمات اولیه جهت تولید PWM در CodeWizard توضیح خواهیم داد.

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 - ویرایش علمی و گرافیکی نهایی : زهره دارابیان



فروشگاه عرضه قطعات الکترونیک ، مکانیک و رباتیک

RoboChip.ir