جلسه پنجم

موضوع جلسه: معرفی مفهوم interrupt (وقفه) و کار با وقفه‌های خارجی

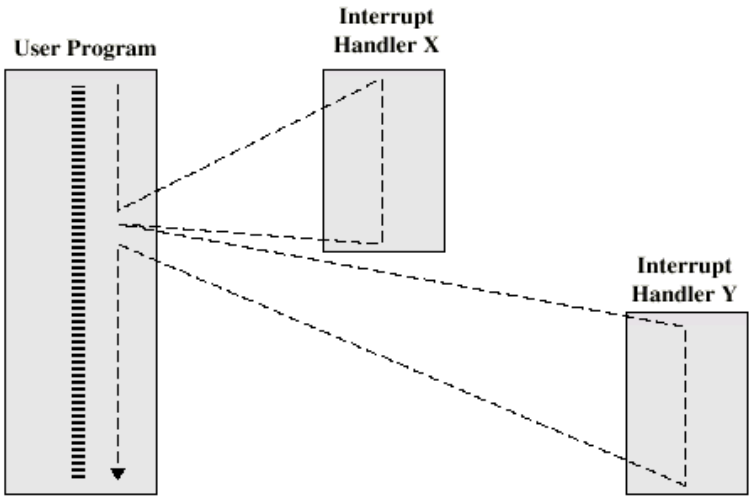
در حالت عادی میکروکنترلر برنامه ()main را از ابتدا تا انتها اجرا می‌کند. اگر در طول این برنامه نیاز به بررسی دستگاهی باشد (به عنوان مثال می‌خواهیم ببینیم آیا کلیدی فشرده شده است یا خیر تا مثلاً سرعت حرکت LED ها را تغییر دهیم) باید داخل حلقه آن پایه بررسی شود مثل برنامه زیر:

```
while(1) {
    if (PIND.2==0)
       d=100;
    else
       d=1000;
     delay_ms(d);
     }
```

یکی از مشکلات این روش این است که اگر تعداد دستگاه‌ها و کلیدهای که باید بررسی شوند زیاد شود، همه آنها باید در حلقه بررسی شوند که این کار برنامه‌نویسی را پیچیده می‌کند. (به این روش اصطلاحاً polling یا سرکشی می‌گویند.)

روش بهتر استفاده از interrupt یا وقفه است.

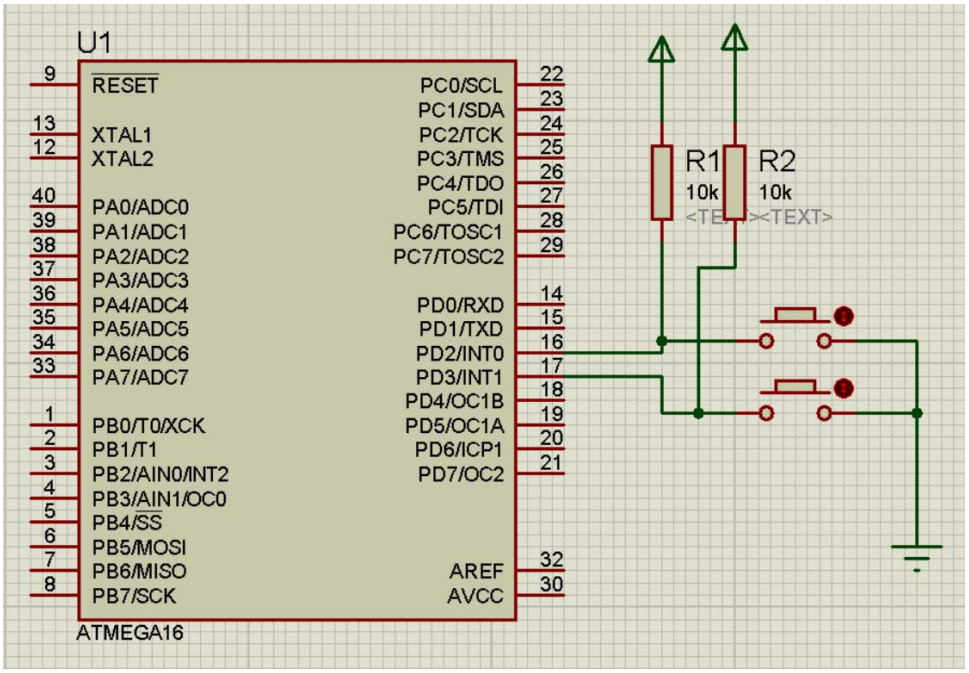
در این روش اگر عامل ایجاد کننده وقفه اتفاق بیافتد. (به عنوان مثال فشردن کلید که ولتاژ پایه را تغییر می دهد.) برنامه ()main در خطی که در حال اجرا است موقتاً متوقف شده سپس زیر برنامه رسیدگی به وقفه (که زیر برنامه‌ای است که خودمان پیش از این نوشته‌ایم) **یک بار** اجرا می‌شود. و سپس به برنامه ()main برمی‌گردد و اجرای آن را از سر می‌گیرد. مانند شکل زیر



همه دستگاه‌های I/O در میکروکنترلر دارای وقفه هستند و برای هر کدام عامل ایجاد کننده وقفه تعیین شده است.

به عنوان مثال برای پایه‌های PD2، PD3 و PB2 وقفه تعریف شده است و اگر تغییری در ولتاژ این پایه‌ها صورت گیرد (که البته نوع این تغییر هم قابل انتخاب است.) وقفه درخواست می‌شود. این سه وقفه به ترتیب با نام‌های INT0، INT1 و INT2 نام گذاری شده‌اند.

روی برد آزمایشگاه هم پایه‌های PD2 و PD3 به دو کلید فشاری (پایین LEDها) متصل شده‌اند.



در این حالت (استفاده از وقفه های پایه‌های PD2 و PD3) می‌توانیم با تعریف وقفه کاری را که می‌خواهیم با فشردن کلید انجام شود در زیر برنامه مربوطه بنویسیم و دیگر نیازی به چک کردن پایه در حلقه نیست.

آزمایش اول:

با هر بار فشردن کلید PD2 عدد روی LEDها یکی افزایش یابد. (دقت کنید که عدد به صورت مبنای ۲ روی LEDها نمایش داده می‌شود.)

آزمایش دوم

لرزش کلید را حذف کنید (debounce)

کلیدهای واقعی (و نه در شبیه‌سازی) در هنگام قطع وصل نوسان ایجاد می‌کنند (یا به عبارتی دیگر شما یک بار کلید را فشار داده اید ولی در زمانی کوتاه چندین بار قطع و وصل می‌شوند) که باید در برنامه حذف شود.

برای این کار هنگام رسیدگی به وقفه (یعنی زمانی که اولین تغییر را دریافت کردیم) ابتدا وقفه را غیر فعال می‌کنیم تا نوسانات بعدی به عنوان فشرده شدن کلید در نظر گرفته نشوند سپس با اعمال تأخیر (delay_ms()) فرصت می‌دهیم که نوسانات تمام شوند. و بعد دوباره وقفه را فعال می‌کنیم)

آزمایش سوم

در ادامه آزمایش قبل با هر بار فشردن کلید متصل به PD3 عدد روی LED کاهش یابد. کاری کنید که عدد هیچگاه منفی نشود.

آزمایش چهارم

در ادامه آزمایش قبل عدد ایجاد شده را (که با دو کلید کاهش و افزایش می‌یابد) را با دو رقم روی 7segmant نمایش دهید.

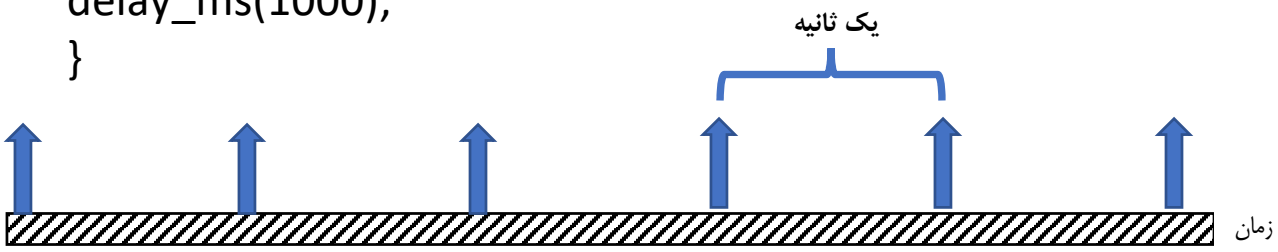
جلسه ششم

موضوع جلسه: اندازه‌گیری زمان به کمک تایمر

تنها ابزار ایجاد زمان در برنامه‌نویسی تا کنون استفاده از تابع (delay_ms()) بوده است. به عنوان مثال در خاموش و روشن کردن و حرکت LEDها، نمایش اعداد روی 7segment و لرزش‌گیری کلید از این تابع استفاده کردیم.

این تابع با متوقف کردن اجرای برنامه این زمان را ایجاد می‌کند به عنوان مثال در برنامه زیر برای خاموش و روشن کردن یک LED یک ثانیه اجرای برنامه متوقف می‌شود:

```
DDRB=0xFF;
while (1) {
    PORTB=0xFF;
    delay_ms(1000);
    PORTB=0x00;
    delay_ms(1000);
    }
```



در شکل بالا محور پایین نشاندهنده زمان است و هر یک ثانیه یکبار LEDها خاموش یا روشن می‌شوند. ولی در بین این زمان یک ثانیه برنامه کاملاً متوقف می‌شود و قادر به انجام کار دیگری نیست.

به عنوان مثال اگر در این برنامه بخواهیم همزمان هم عددی را روی 7sement چند رقمی نمایش دهیم امکان‌پذیر نیست. زیرا برای نمایش اعداد باید در حدود ۵ میلی‌ثانیه رقم جدیدی را نمایش دهیم تا امکان نمایش ارقام متفاوت ایجاد شود.

ولی در برنامه بالا به مدت ۱ ثانیه برنامه متوقف می‌شود. و در این یک ثانیه نمی‌توان کاری انجام داد.

برای حل این مشکل برای زمان‌سنجی ابزاری مستقل در میکروکنترلر به نام Timer طراحی شده است. این ابزار قادر است به طور مستقل از برنامه (و بدون ایجاد توقف در آن) زمان را اندازه‌گیری کند.
در میکروکنترلر AVR مدل 32 یا ATmega16 سه تایمر مستقل وجود دارد. یعنی اندازه گیری سه زمان متفاوت امکان پذیر است.
این تایمرها با عنوان‌های Timer0، Timer1 و Timer2 نام‌گذاری شده‌اند. تایمر 0 و 2 ۸بیتی و تایمر 1 ۱۶بیتی است. درباره تفاوت این دو در ادامه صحبت می‌کنیم.
هر تایمر دارای ۳ رجیستر است.

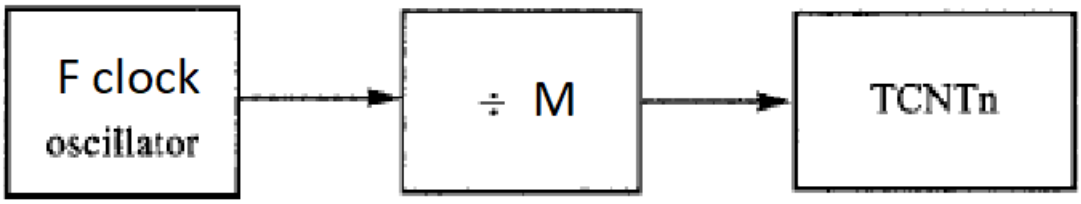
- رجیستر TCNT که شمارنده است.
- رجیستر TCCR که تنظیمات تایمر با آن انجام می‌شود.
- رجیستر OCR که رجیستر مقایسه نام دارد.

به عنوان مثال برای تایمر 0 نام این رجیسترها TCNT0، TCCR0 و OCR0 است.

نحوه اندازه گیری زمان:
برای درک روش زمان‌سنجی باید ابتدا نحوه کار رجیستر TCNT را متوجه شویم.
TCNT یک رجیستر است که عددی در آن ذخیره می‌شود. این رجیستر به فرکانس کلاک میکروکنترلر وصل شده است و مقدار آن با هر لبه کلاک یکی افزایش می‌یابد. به عنوان مثال اگر فرکانس کلاک f=1 MHz باشد فاصله لبه‌های بین کلاک برابر $f = \frac{1}{T} = 1\mu s$ است و هر $1\mu s$ مقدار آن یکی افزایش می‌یابد.

- TCNT در تایمرهای ۸ بیتی دارای ۸ بیت است و محدوده شمارش آن از ۰ تا ۲۵۵ است.
- TCNT در تایمرهای ۱۶ بیتی دارای ۱۶بیت است و محدوده شمارش آن از ۰ تا ۶۵۵۳۵ است.

همچنین طبق شکل زیر امکان این وجود دارد که فرکانس ورودی به TCNT با ضریب M کاهش یافته و در نتیجه زمان شمارش هر کلاک با ضریب M افزایش یابد.
M را می‌توان یکی از اعداد ۱، ۸، ۶۴، ۲۵۶ یا ۱۰۲۴ انتخاب کرد.



- به طور خلاصه می توان برای تنظیم زمان تایمر از فرمول زیر استفاده کرد:

$$MN = F_{clk}T$$

که در آن T زمان دلخواه و F_{clk} فرکانس کلاک میکروکنترلر است که معمولاً با توجه به نیاز مسأله مشخص هستند. M که تقسیم کننده فرکانس است و N مقدار شمارش مورد نیاز برای TCNT است. باید به دست بیایند.

مثال: زمان ۲۰ میلی‌ثانیه را با تایمر ایجاد کنید.

$$MN = F_{clk}T$$
$$\rightarrow MN = 1 \times 10^{6} \times 20 \times 10^{-3}$$
$$\rightarrow MN = 20 \times 10^{3} = 20000$$

در اینجا مقادیر امکان پذیر M را امتحان می‌کنیم:

اگر $M=1$ باشد $N=20000$ می‌شود. یعنی $TCNT$ باید ۲۰۰۰۰ تا بشمرد تا زمان مورد نظر به دست آید. این موضوع برای تایمر ۱۶ بیتی امکان پذیر است. (چون حداکثر شمارش آن ۶۵۵۳۵ است.) ولی برای تایمر ۸ بیتی امکان پذیر نیست. برای اینکه ببینیم آیا می‌توان از تایمر ۸ بیتی استفاده کرد بقیه مقادیر M را هم امتحان می‌کنیم

M	N	تایمر ۸ بیتی	تایمر ۱۶ بیتی
۱	۲۰۰۰۰	✗	✓
۸	۲۵۰۰	✗	✓
۶۴	۳۱۲,۵→۳۱۲	✗	✓
۲۵۶	۷۸,۱۲۵→۷۸	✓	✓
۱۰۲۴	۱۹,۵۳→۲۰	✓	✓

مودهای شمارش در تایمرها

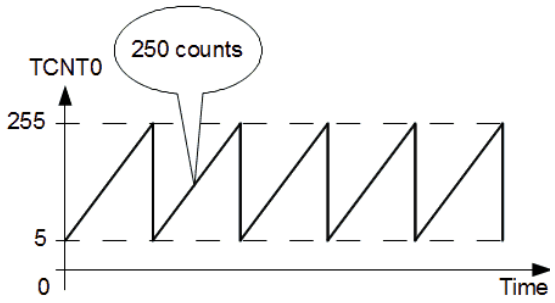
در تایمر روش‌های مختلفی برای شمارش طراحی شده که به آنها مود شمارش می‌گوییم برای زمان‌سنجی دو مود مهم است:

- مود نرمال:

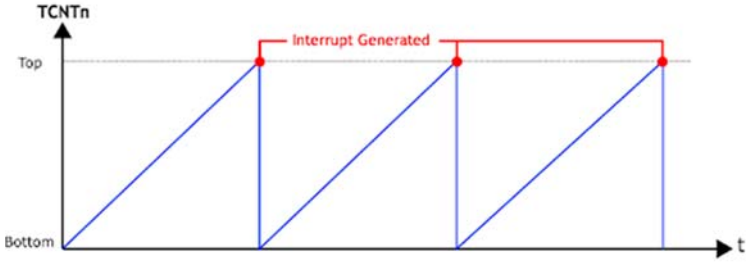
در این مود شمارش TCNT از ۰ تا حداکثر (مثلا ۲۵۵) انجام شده و سپس به صفر برمی‌گردد. در هنگام برگشت به صفر یک درخواست وقفه سرریز (overflow) صادر می‌شود.

اگر بخواهیم به عنوان مثال ۷۸ با TCNT بشمریم، مقدار اولیه TCNT را ۱۷۷=۷۸-۲۵۵ می‌گذاریم تا از ۱۷۷ شروع کند و با شمارش ۷۸ عدد سرریز کرده و با وقفه به ما اعلام سپری شدن زمان کند.

در شکل زیر از ۵ کرده و ۲۵۰ تا شمرده و سرریز کرده است. بعد از سرریز هم باید اگر می‌خواهیم دوباره همان زمان را اندازه‌گیری کنیم TCNT را دوباره مقدار مجدد بدهیم چون مقدار آن بعد از سرریز صفر می‌شود.



Normal Mode

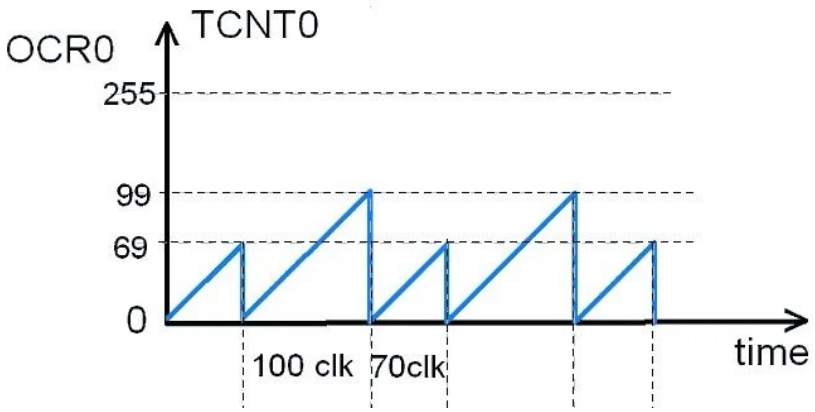


به کارگیری مود نرمال کمی مشکل تر از مود بعدی است. بنابرین در اکثر موارد از مود بعدی استفاده می‌کنیم.

- مود CTC

در این مود شمارش TCNT از ۰ شروع می‌شود و تا حداکثر (مثلاً ۲۵۵) نمی‌رسد بلکه هر وقت با مقدار رجیستر OCR برابر شد، صفر می‌شود. مقدار OCR در اختیار ماست و می‌توانیم آن را مقدار دهیم و معمولاً برابر با عدد n به دست آمده از فرمول قرار می‌دهیم.

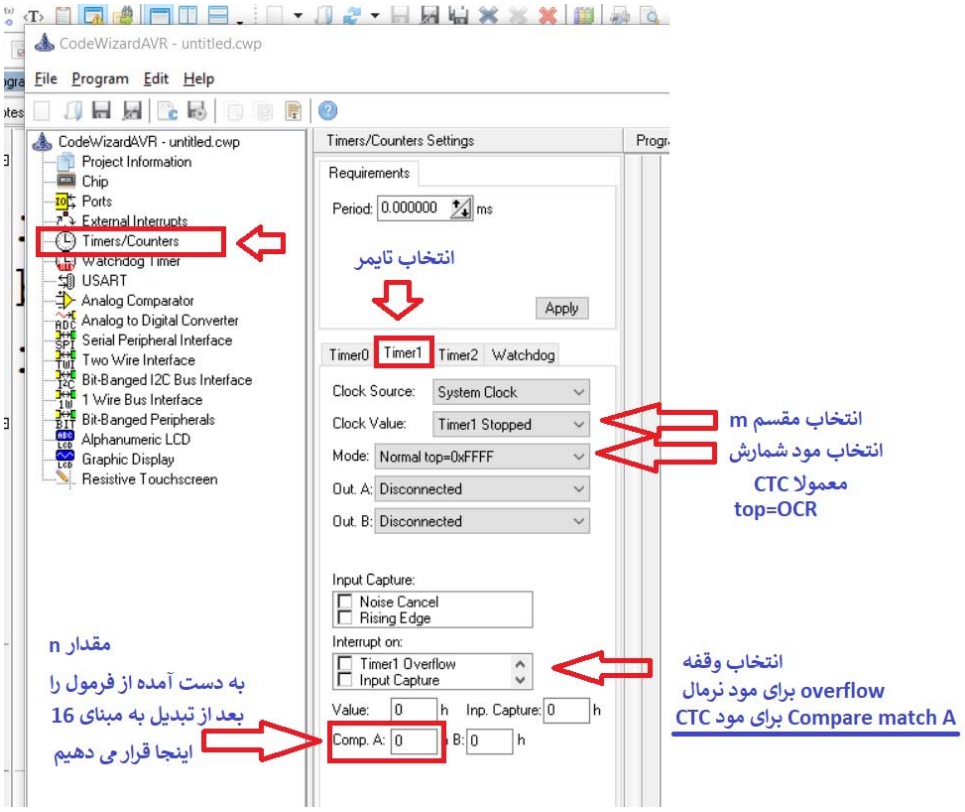
در این مود وقفه در هنگام برابری TCNT و OCR اتفاق می‌افتد.



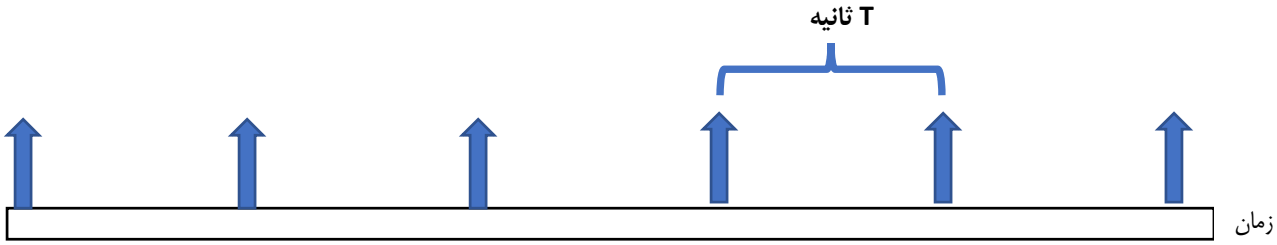
- به طور خلاصه برای به کارگیری تایمر برای زمان‌سنجی:

۱. از فرمول m و n را محاسبه می‌کنیم

۲. تنظیمات تایمر را طبق شکل زیر انجام می‌دهیم (شکل برای تایمر ۱ است که ۱۶ بیتی است.)



حالا هر دستور متناسب با هر کاری را که قرار است هر T ثانیه انجام شود در زیر برنامه وقفه مربوط به تایمر قرار می‌دهیم. و همانطور که در شکل زیر دیده می‌شود حالا بین T ثانیه برنامه آزاد است و می‌توان برنامه دیگری نیز به دلخواه انجام داد. در آزمایش دوم این مورد را بررسی می‌کنیم.



آزمایش اول: عدد روی LEDها را (در مبنای ۲) هر ثانیه یکی افزایش دهید.

$f_{clk}=1x10^6$ و T=1

$$\rightarrow MN = 1 \times 10^6 = 1000000$$

M	N	تایمر ۸ بیتی	تایمر ۱۶ بیتی
۱	۱۰۰۰۰۰۰	✗	✗
۸	۱۲۵۰۰۰	✗	✗
۶۴	۱۵۶۲۵	✗	✓
۲۵۶	۳۹۰۶,۲۵	✗	✓
۱۰۲۴	۹۷۶,۵۶	✗	✓

آزمایش دوم:

عدد به دست آمده را با سه رقم روی 7segment نمایش دهید.

آزمایش سوم:

ثانیه شمار را به دهم ثانیه شمار تبدیل کنید: (T=0.1 s)

$$\rightarrow MN = 1 \times 10^5 = 100000$$

تایمر ۱۶ بیتی	تایمر ۸ بیتی	N	M
✗	✗	۱۰۰۰۰۰	۱
✓	✗	۱۲۵۰۰	۸
✓	✗	۱۵۶۲,۵	۶۴
✓	✗	۳۹۰,۶۲۵	۲۵۶
✓	✓	۹۷,۶۵۶	۱۰۲۴

آزمایش چهارم:

با کلید INT0 و INT1 زمان سنج را متوقف و راه اندازی کنید. (با روش polling یا وقفه)

جلسه هفتم

موضوع جلسه: تولید موج PWM توسط تایمر

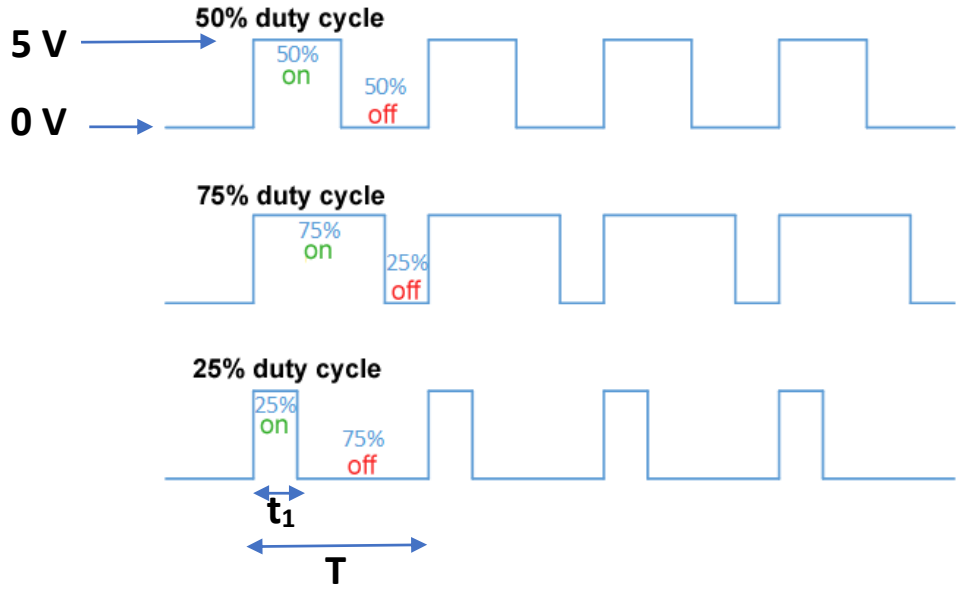
می‌دانیم که همه پایه‌های ورودی/خروجی میکروکنترلر (۳۲ پایه در مدل‌های Mega16 و Mega32) دیجیتال هستند. یعنی پس از خروجی شدن می‌توان آنها را فقط در دو وضعیت 0 (۰ ولت) یا 1(۵ ولت) قرار داد. و اگر Led به آنها وصل باشد، فقط دو وضعیت روشن و خاموش خواهد داشت.

اما در بسیاری از کاربردها نیاز داریم که ولتاژ آنالوگ (مقدار پیوسته قابل تغییر از ۰ تا ۵ ولت) داشته باشیم. به عنوان مثال:

- کنترل سرعت موتور: برای موتورهای DC می‌توان با تغییر ولتاژ به صورت پیوسته سرعت موتور را تنظیم کرد.
- تغییر شدت نور یک یا چند LED یا چراغ و در نتیجه تولید رنگ‌های مختلف: در صورتی که ولتاژی کمتر به LED داده شود شدت نور آن هم کاهش می‌یابد. مثلا به جای ۰ و ۵ ولت، ولتاژی بین این ولتاژها بدهیم: ۲,۵ ولت (شدت نور نسبت به ۵ ولت کاهش می‌یابد.
- تولید صدا روی بلندگو: صدا سیگنالی آنالوگ است و اگر بخواهیم روی بلندگو از طریق میکروکنترلر صدایی ضبط شده را پخش کنیم، باید ولتاژ آنالوگ تولید کنیم.
- و چندین کاربرد دیگر...

یک راه برای اینکه بتوانیم ولتاژ آنالوگ روی پایه‌های دیجیتال تولید کنیم، استفاده از موج PWM است.

موج PWM (Pulse Width Modulation) یک موج مربعی با فرکانس (دوره تناوب T) ثابت و پهنای پالس (dutycyle)متغیر است. این موضوع در شکل زیر نمایش داده شده است.



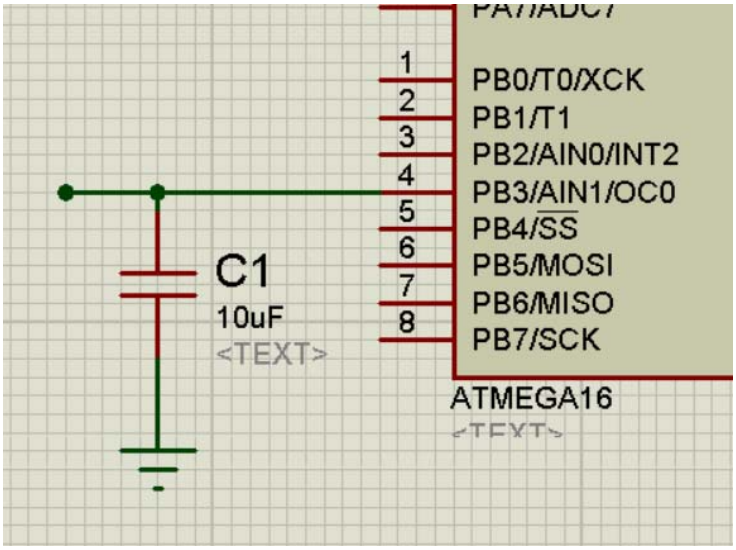
همانطور که در شکل مشخص است، این موج قابل ایجاد توسط پایه دیجیتال می‌باشد. چون مقدار آن فقط ۰ و ۱ است. اما دوره روشن و خاموش بودن آن در اختیار ماست و از ۰ تا ۱۰۰ درصد قابل تغییر است.

اگر مقدار میانگین (مقدار DC) این موج را به دست آوریم:

$$V_{DC} = \frac{t_1}{T} \times 5V$$

یعنی می‌توان این موج را در پایه دیجیتال میکرو تولید کرد و سپس فقط با حفظ مقدار DC آن ولتاژ بالا را تولید کرد. در این رابطه می‌توان با تغییر t_1 از صفر تا T مقدار ولتاژ V_{DC} را از ۰ تا ۵ ولت تغییر داد.

برای حفظ مقدار DC اگر فرکانس موج به اندازه کافی زیاد باشد می‌توان از مدار زیر (یک خازن با ظرفیت به اندازه کافی بزرگ) بهره گرفت.



می‌توان موج pwm را برنامه نویسی عادی در یک حلقه بی پایان و با تأخیرهای متفاوت برای روشن و خاموش بودن در هر پایه‌ای ایجاد کرد.

اما در اینجا می‌خواهیم از امکانی که در تایمرها قرار داده شده است استفاده کنیم.

در تایمرها می‌توان به طور مستقل از برنامه اصلی موج PWM تولیدکرد.

برای اینکار باید نکات زیر را در نظر گرفت:

۱- هر تایمر در پایه‌ای مشخص امکان تولید این موج را دارد:

تایمر 0	تایمر 1	تایمر 2
پایه PB3	پایه‌های PD4 و PD5	پایه PD7

۲- یکی از مودهای شمارش PWM برای تایمر انتخاب شود. (معمولا مود Phase correct را انتخاب می‌کنیم.)

۳- خروجی تایمر برای تولید موج باید با انتخاب گزینه non-inverted در بخش output فعال شود. واگرنه موجی تولید نمی‌شود.

۴- برای ایجاد ولتاژ دلخواه در پایه، طبق رابطه زیر به رجیستر OCR تایمر مورد نظر مقدار داده شود.

$$OCR = \frac{V_{\text{دلخواه}}}{5} \times \text{تایمر شمارش حداکثر}(65535 \text{ یا} 255)$$

آزمایش اول:

یک LED را با ولتاژ ۲,۵ ولت روشن کنیم.و ولتاژ ۱,۲۵ ولت

$$OCR = \frac{2.5}{5} \times 255 = 127.5 = 128$$

آزمایش دوم:

برنامه‌ای بنویسید که در آن هر ۰,۴ دهم ثانیه ولتاژ خروجی 0.5 ولت افزایش یابد. و این روند تکرار شود. بدون استفاده از تابع delay

$$Mn=fT$$

$$Mn=1*10^{6*}0.4=4*10^{5}=400000$$

$$M=8\ N=50000$$

هر 0.4 ثانیه OCR باید ۲۵ اضافه شود.

آزمایش سوم:

در ادامه آزمایش قبلی هر ۰,۴ ثانیه ولتاژ ۰,۵ ولت کمتر شود.

آزمایش چهارم:

موج تولید شده را به پایه های دیگر انتقال دهید (سایر LED ها)

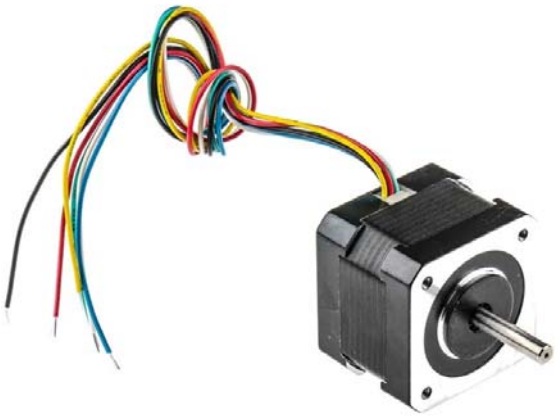
جلسه هشتم

موضوع جلسه: راه‌اندازی موتور پله‌ای (Stepper Motor)

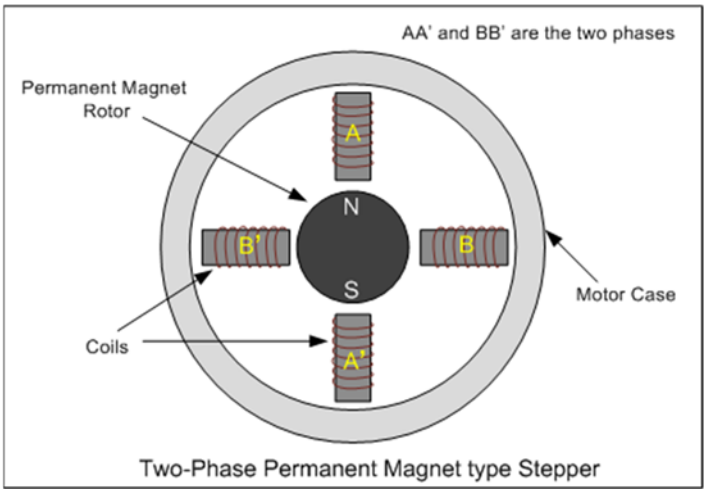
ساختار درونی هر موتور الکتریکی دارای دو بخش استاتور (بخش ثابت و بدون حرکت) و بخش روتور (بخش متحرک) است روتور به محور بیرونی موتور متصل است که معمولا بار متحرک روی آن قرار دارد.

موتورهای الکتریکی با انواع مختلفی طراحی شده‌اند که یکی از آنها موتور پله‌ای است. در موتور پله‌ای استاتور سیم‌پیچی شده است. یعنی می توان با گذراندن جریان الکتریکی از آن بخش‌های دلخواه را به آهنربای موقتی تبدیل کرد. اما روتور در موتورهای پله‌ای آهنربای دائمی است و سیم‌پیچی شده نیست.

هدف طراحی موتور پله‌ای حرکت با زاویه دقیق و سرعت دقیق بوده است. در بسیاری از کاربردهایی نیاز به حرکت دقیق یا سرعت دقیق دارند به کار می‌روند.



ساختار درونی ساده شده درون موتورهای پله‌ای



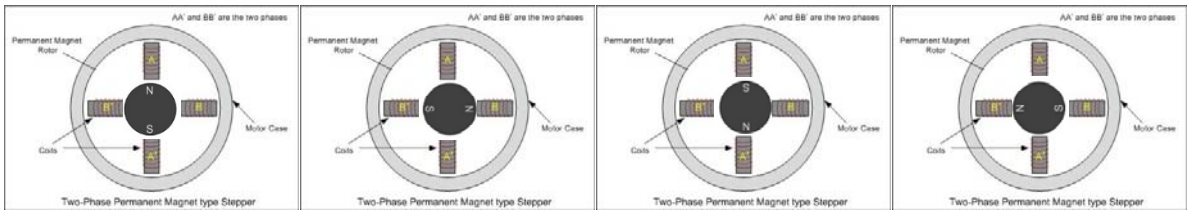
در شکل بالا ساختار ساده شده درون موتور پله‌ای دیده می‌شود (ساختار واقعی پیچیده تر از شکل بالا است که در ادامه درباره آن صحبت می‌کنیم). اکثر موتورهای پله‌ای دارای ۴ سیم‌پیچ مستقل روی استاتور خود هستند. با عبور جریان الکتریکی سیم‌پیچ‌ها به آهنربا تبدیل شده و قطب مخالف روتور را به خود جذب می‌کنند.

روش‌های راه‌اندازی موتور پله‌ای

۱- روش اول:

یک مجموعه فرمان کامل

شماره	سیم‌پیچ A	سیم‌پیچ B	سیم‌پیچ A+	سیم‌پیچ B+	زاویه چرخش	زاویه چرخش مجموع
۱	**1**	0	0	0	-	-
۲	0	**1**	0	0	Step=S=90درجه	S= ۹۰
۳	0	0	**1**	0	Step=S=90درجه	۲S=۱۸۰درجه
۴	0	0	0	**1**	Step=S=90درجه	۳S=۲۷۰درجه
تکراری	**1**	0	0	0	Step=S=90درجه	۴S=۳۶۰درجه



۱ ۲ ۳ ۴

در جدول بالا برای چرخش موتور به اندازه 4s درجه ۵ فرمان دادیم که پنجمی تکراری است. در نتیجه ۴ فرمان اول یک مجموعه فرمان کامل موتور پله‌ای محسوب می‌شود.

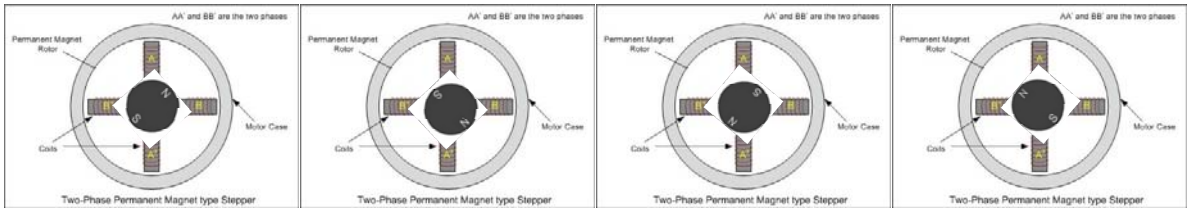
- **<u>قانون مهم موتورهای پله‌ای: هر مجموعه فرمان کامل= 4s درجه چرخش</u>**

<u>نکته مهم: در موتورهای واقعی s یا گام موتور بسیار کمتر از ۹۰ درجه است و شکل بالا فقط برای آموزش است.</u>

۲- روش دوم راه اندازی: این روش قدرت بیشتری دارد زیرا همزمان دو سیم‌پیچ را به‌کارمی‌گیرد.

یک مجموعه فرمان کامل

شماره	سیم‌پیچ A	سیم‌پیچ B	سیم‌پیچ A+	سیم‌پیچ B+	زاویه چرخش	زاویه چرخش مجموع
<u>۱</u>	**<u>1</u>**	**<u>1</u>**	0	0	-	-
۲	0	**<u>1</u>**	**<u>1</u>**	0	Step=S=90درجه	S= ۹۰
۳	0	0	**<u>1</u>**	**<u>1</u>**	Step=S=90درجه	۲S=۱۸۰درجه
۴	**<u>1</u>**	0	0	**<u>1</u>**	Step=S=90درجه	۳S=۲۷۰درجه
<u>تکراری</u>	**<u>1</u>**	**<u>1</u>**	0	0	Step=S=90درجه	۴S=۳۶۰درجه



۱ ۲ ۳ ۴

در جدول بالا برای چرخش موتور به اندازه 4s درجه ۵ فرمان دادیم که پنجمی تکراری است. در نتیجه ۴ فرمان اول یک مجموعه فرمان کامل موتور پله‌ای محسوب می‌شود.

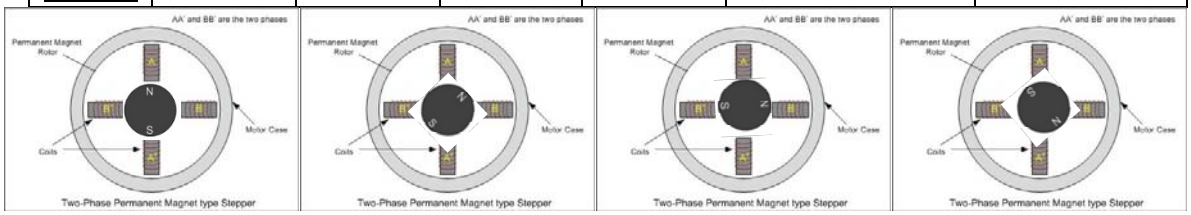
۳- روش سوم راه اندازی: این روش دقت بیشتری دارد و در هر مرحله نیم s چرخش داریم

در جدول زیر برای چرخش موتور به اندازه 4s درجه ۹ فرمان دادیم که نهمی تکراری است. <u>در نتیجه ۸ فرمان اول یک مجموعه فرمان کامل موتور پله‌ای</u> محسوب می‌شود.

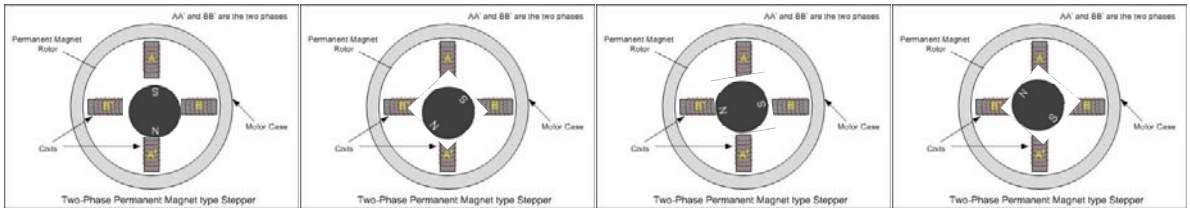
ولی باز هم با یک فرمان کامل 4s درجه چرخش داریم.

شماره	سیم‌پیچ A	سیم‌پیچ B	سیم‌پیچ A+	سیم‌پیچ B+	زاویه چرخش	زاویه چرخش مجموع

۱	**1**	**1**	0	0	-	-
۲	0	**1**	0	0	Step=S/2=۴۵درجه	S/2= ۴۵
۳	0	**1**	**1**	0	Step=S/2=۴۵درجه	S= ۹۰
۴	0	0	**1**	0	Step=S/2=۴۵درجه	2s/3=135
۵	0	0	**1**	**1**	Step=S/2=۴۵درجه	2s=180
۶	0	0	0	**1**	Step=S/2=۴۵درجه	5s/2=225
۷	**1**	0	0	**1**	Step=S/2=۴۵درجه	3s=270
۸	**1**	0	0	0	Step=S/2=۴۵درجه	7s/2=315
تکراری	**1**	**1**	0	0	Step=S/2=۴۵درجه	4s=360

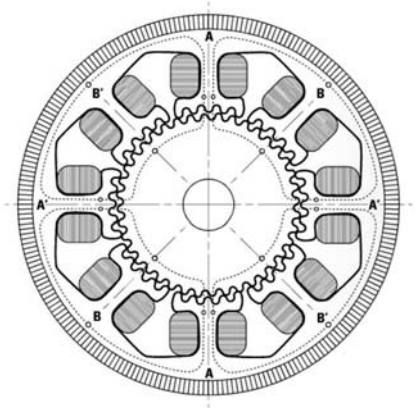


۱ ۲ ۳ ۴



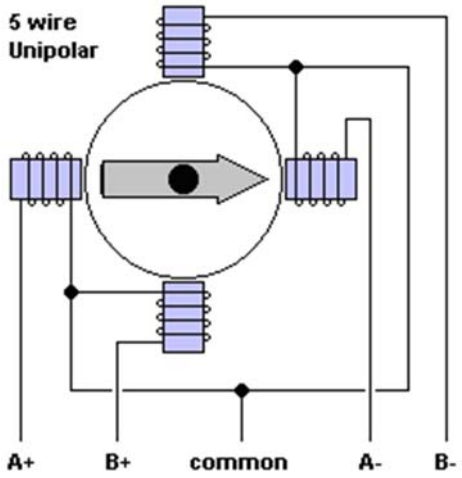
۵ ۶ ۷ ۸

همانطور که گفتیم در موتورهای واقعی زاویه s بسیار کمتر از ۹۰ است (چیزی مشابه شکل زیر است.) و شکل‌های بالا برای آموزش فرمان‌ها آورده شده است.



Stepper motor rotor & stator cross section

هر stepper موتور دارای ۵ سیم خروجی است که یک سیم مشترک بین همه سیم‌پیچ‌ها است و چهار سیم دیگر سر دیگر سیم‌پیچ ها می‌باشد. روی برد آزمایشگاه سیم مشترک به زمین وصل شده است. و **۴ سیم دیگر به پایه‌های PC4 تا PC7 میکروکنترلر** (البته از طریق یک driver که جریان کافی برای راه‌اندازی سیم‌پیچ‌ها را تأمین می‌کند.) متصل شده است. بنابرین می‌توان با یک کردن هر یک از این پایه‌ها در میکروکنترلر سیم‌پیچ مربوطه را روشن کرد.



- آزمایش اول: مقدار گام s موتور روی برد را به دست آورید. s برابر چند درجه است؟

پاسخ یک فرمان کامل می‌دهیم تا موتور 4s درجه بچرخد مقدار چرخش را مشاهده کرده و s را به دست می‌آوریم.

از روش دوم راه‌اندازی استفاده کنید. بین فرمان‌ها حتما تاخیر نیاز است. (فعلا ۲۰ میلی‌ثانیه بگذارید)

با ۶ فرمان کامل ۱۸۰ درجه چرخش مشاهده شد. یعنی یک فرمان کامل ۳۰ درجه چرخش است. 4s=30 پس s=7.5

- اگر فرمان کامل در یک حلقه با تعداد تکرار مشخص مثلا n قرار گیرد موتور 4ns درجه چرخش خواهد داشت. و اگر در یک حلقه بی‌نهایت قرار گیرد برای همیشه می چرخد.

برای پیدا کردن سرعت موتور در حال چرخش جدول زیر را در نظر می‌گیریم:

t مقدار تاخیری است که بین فرمان‌ها قرار داده‌ایم و T زمان یک چرخش کامل موتور است.

	مقدار چرخش	زمان لازم
یک فرمان کامل	4s درجه	4t
یک دور کامل	۳۶۰ درجه	T=?

در نتیجه $T = \frac{360 \times 4t}{4s} = \frac{360t}{s}$

سرعت موتور (برحسب دور در ثانیه) برابر $V = \frac{1}{T} = \frac{s}{360t}$

و برحسب دور در دقیقه: $V_{rpm} = \frac{s}{360t} \times 60 = \frac{s}{6t}$ خواهد بود

پس طبق رابطه زیر می‌توان با تغییر t (تاخیر بین فرمان‌ها) سرعت را به دقت تعیین کرد:

$$t = \frac{s}{360V}$$

- آزمایش دوم: سرعت موتور را
  - ۱ دور در ثانیه

$$t = \frac{7.5}{360} = 0.02083\ s = 20.83\ ms$$

  - $\frac{1}{60}$ دور در ثانیه

$$t = \frac{7.5}{360\frac{1}{60}} = \frac{7.5 * 60}{360} = \frac{7.5}{6} = 1.25\ s = 1250\ ms$$

  - ۲ دور در ثانیه

$$10.4ms$$

  - ۴ دور در ثانیه
- آزمایش سوم: جهت حرکت موتور را معکوس کنید. کافیست ترتیب فرمان‌ها تغییر کند (از روی جدول از پایین به بالا)
- آزمایش چهارم: با کلیدهای متصل به پایه PD2 و PD3 جهت حرکت موتور را معکوس کنید. (یکی راست‌گرد و دیگری چپ‌گرد کند.)
- آزمایش پنجم: طبق برنامه زیر موتور را به حرکت درآورید. (الزاما باید از تایمر استفاده کنیم)

زمان	وضعیت حرکت موتور
۰ تا ۵ ثانیه	راست‌گرد
۵ تا ۱۰ ثانیه	توقف
۱۰ تا ۱۵ ثانیه	چپ‌گرد
۱۵ تا ۲۰ ثانیه	توقف