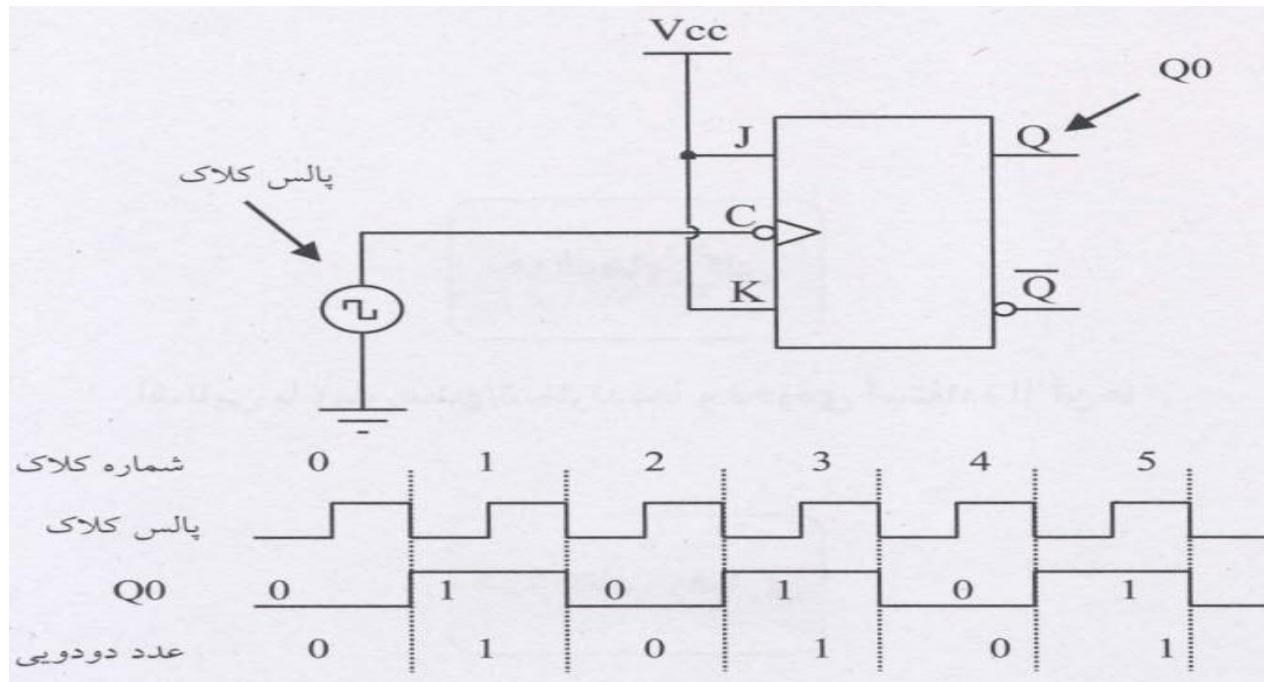


# آشنایی با تایمر / کانتر و نحوه استفاده از آنها

دانشگاه علم و صنعت - دانشکده مهندسی کامپیوتر - مهر ۹۳  
هاشم مشحون

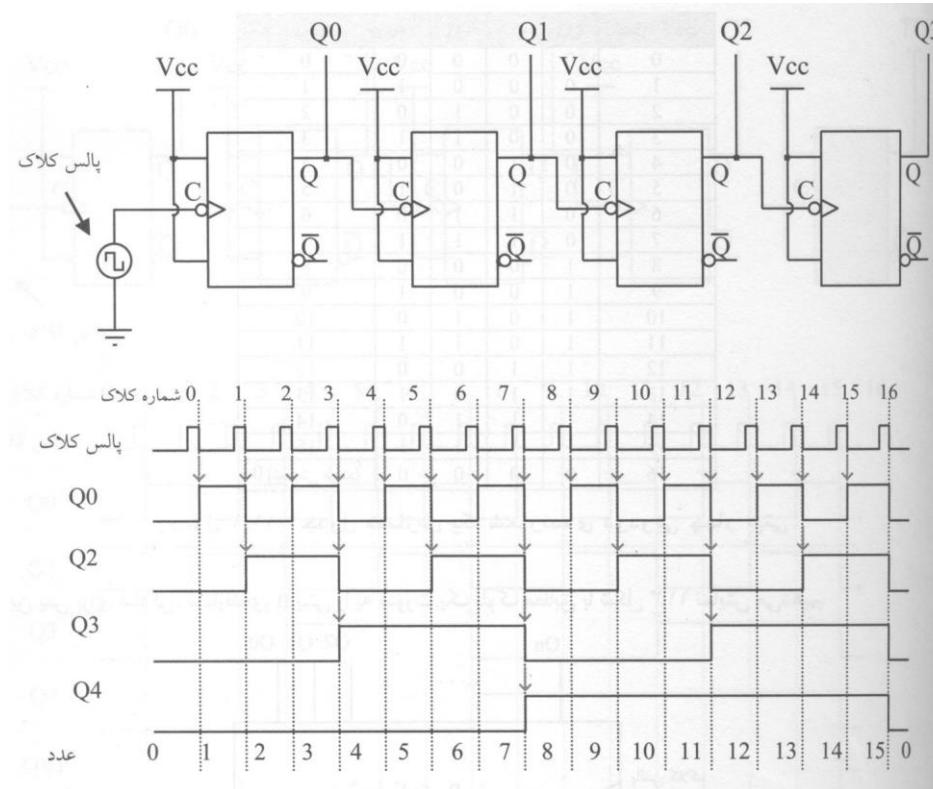
# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## ▶ آشنایی با شمارنده های دیجیتال



▶ در فلیپ فلاپ فوق که از نوع K-L است اگر هر دو ورودی را به یک منطقی وصل کنیم، لبه‌ی پالس کلاسی ورودی باعث تغییر سطح خروجی خواهد شد.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر



با پشت سرهم بستن چهار فلیپ فلاپ طبق شکل فوق میتوان یک شمارنده باینری را ساخت که از صفر تا پانزده را میشمارد. ▶

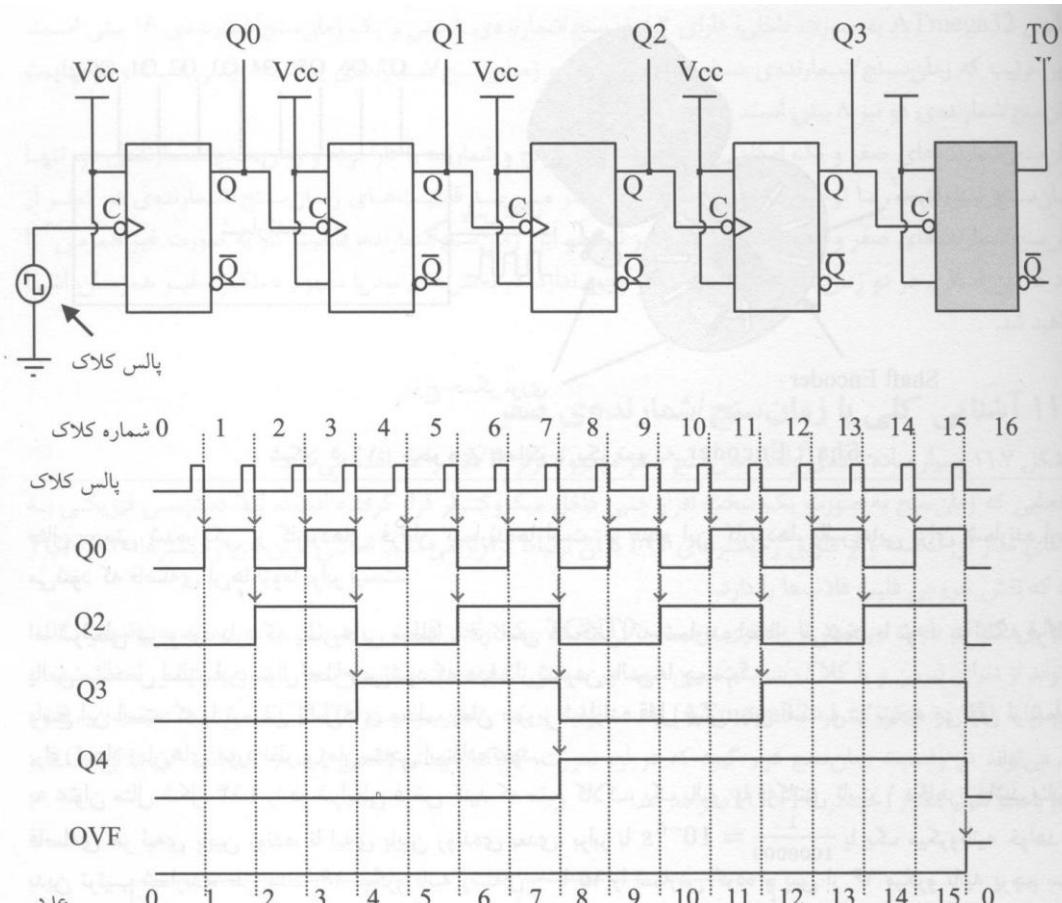
# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

شماره کلاس	Q3	Q2	Q1	Q0	عدد دودویی
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	10
11	1	0	1	1	11
12	1	1	0	0	12
13	1	1	0	1	13
14	1	1	1	0	14
15	1	1	1	1	15
16	0	0	0	0	0 (شمارش مجدد)

در نهایت آنچه در اثر پالس کلاک ورودی، در خروجی فلیپ فلاپها ظاهر میشود مطابق با جدول روبرو است.

- ▶ به همین ترتیب برای ساخت یک شمارنده ۸ بیتی به ۸ فلیپ فلاپ و برای شمارنده  $n$  بیتی،  $n$  فلیپ فلاپ نوع K-L نیاز خواهد بود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر



شکل ۱۱.۴: نمودار بلوکی یک شمارنده ۱۶ بیتی همراه با بیت سرریز

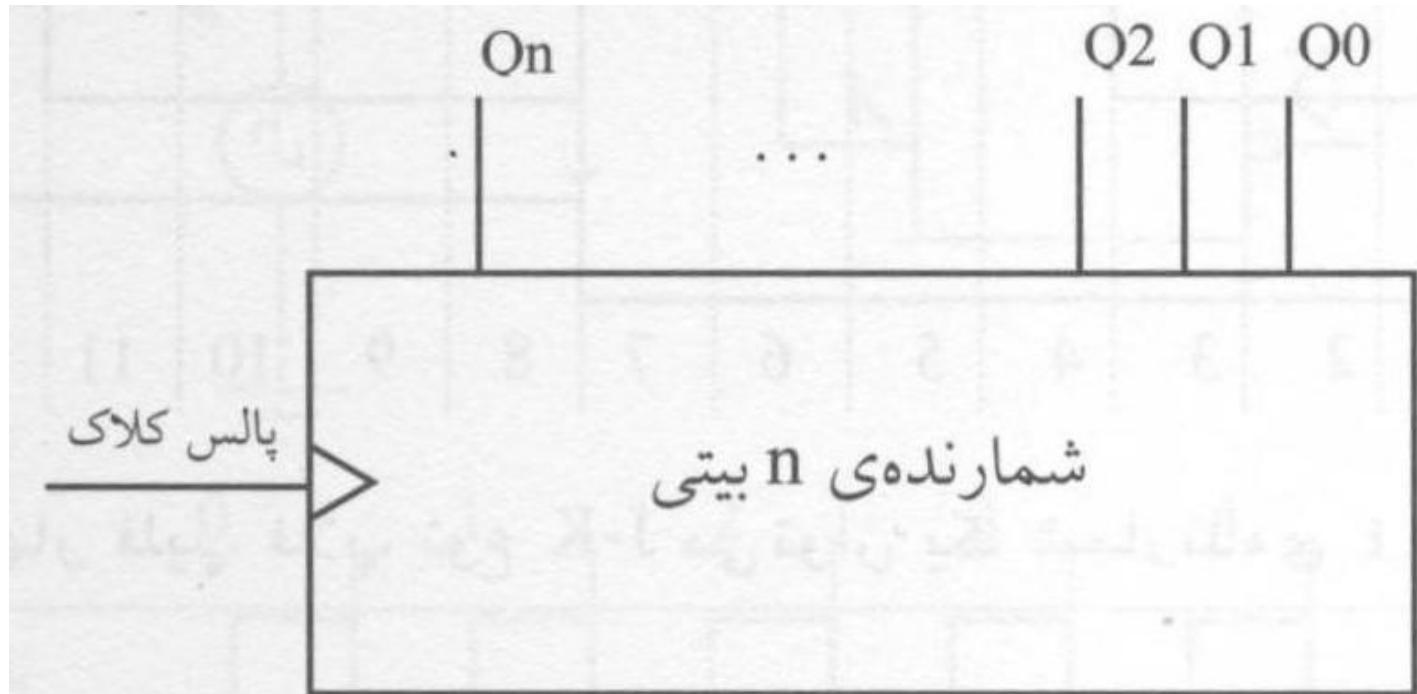
آشنایی با مفهوم سرریز شمارنده

لحظه‌ای که زمان سنج به انتهای رشته‌ی شمارش میرسد و با صفر بازگذاری می‌شود سرریز یا **overflow** نام دارد.

چرا که در این لحظه خروجی تمام فلیپ‌فلاپها یک شده و شمارنده ظرفیت بیشتری ندارد بنابراین با کلاک بعدی خروجی آن صفر می‌شود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

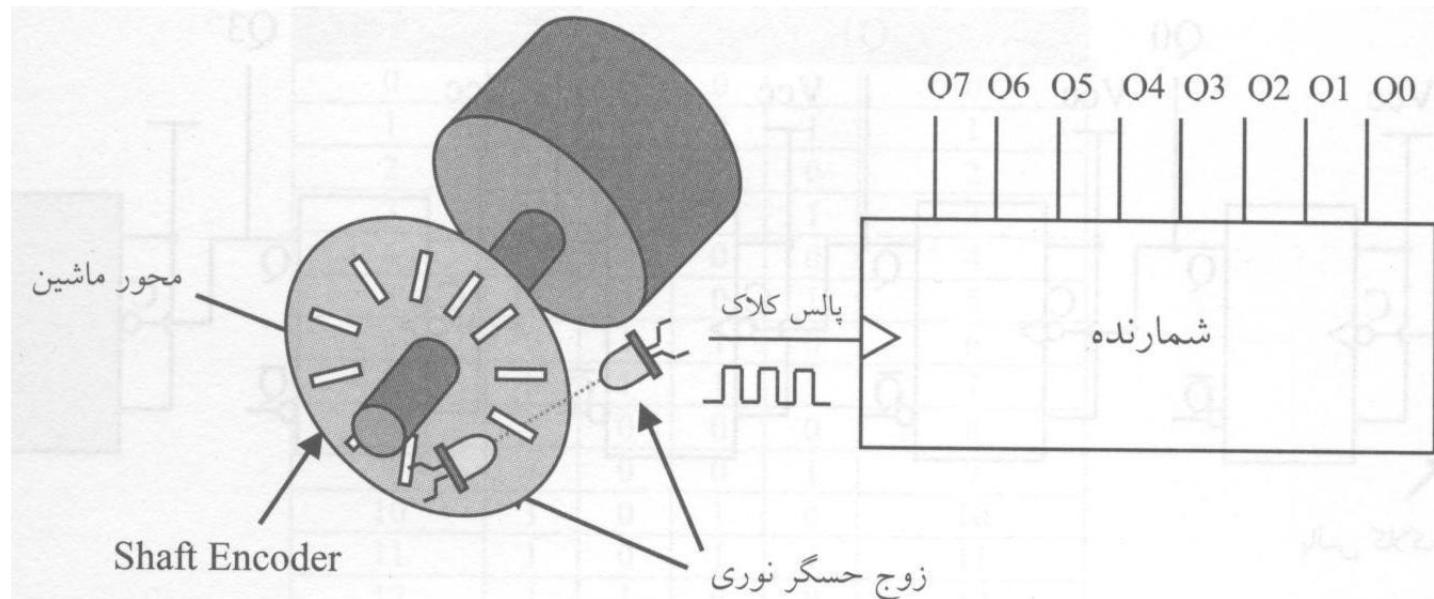
از این پس برای سادگی، شمارنده  $n$  بیتی را به صورت یک بلوک مطابق با شکل زیر نمایش میدهیم.



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

تفاوت تایمر (زمان سنج) و کانتر (شمارنده)

اکنون میدانیم که پالس کلاک، منشاء تغییر در خروجی فلیپ فلاپها و در نتیجه پیشروی در رشته‌ی شمارش است. این پالس از چه طریق تامین میشود؟



با دریافت هر سیگنال نوری توسط حسگر گیرنده، یک پالس برای شمارنده ارسال میشود و مقدار آن را یک واحد افزایش میدهد. بدین ترتیب شمارنده مقدار حرکت چرخشی محور ماشین را ثبت کرده و اندازه جابجایی آن را به شکل دیجیتالی بیان میکند.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## تفاوت تایمر (زمان سنج) و کانتر (شمارنده)

اما شرایطی وجود دارد ه پالسها ی منظم و با فرکانس مشخص، به شمارنده اعمال میشود. با توجه به اینکه فرکانس پالس مشخص است، این سوال مطرح میشود که هدف از شمردن پالسها چیست؟

پاسخ: با شمردن پالسها ی منظم، زمان سرریز شمارنده قابل پیش بینی است و در نتیجه میتوان از شمارنده برای ایجاد زمانهای مورد نظر و زمان سنجی استفاده نمود.

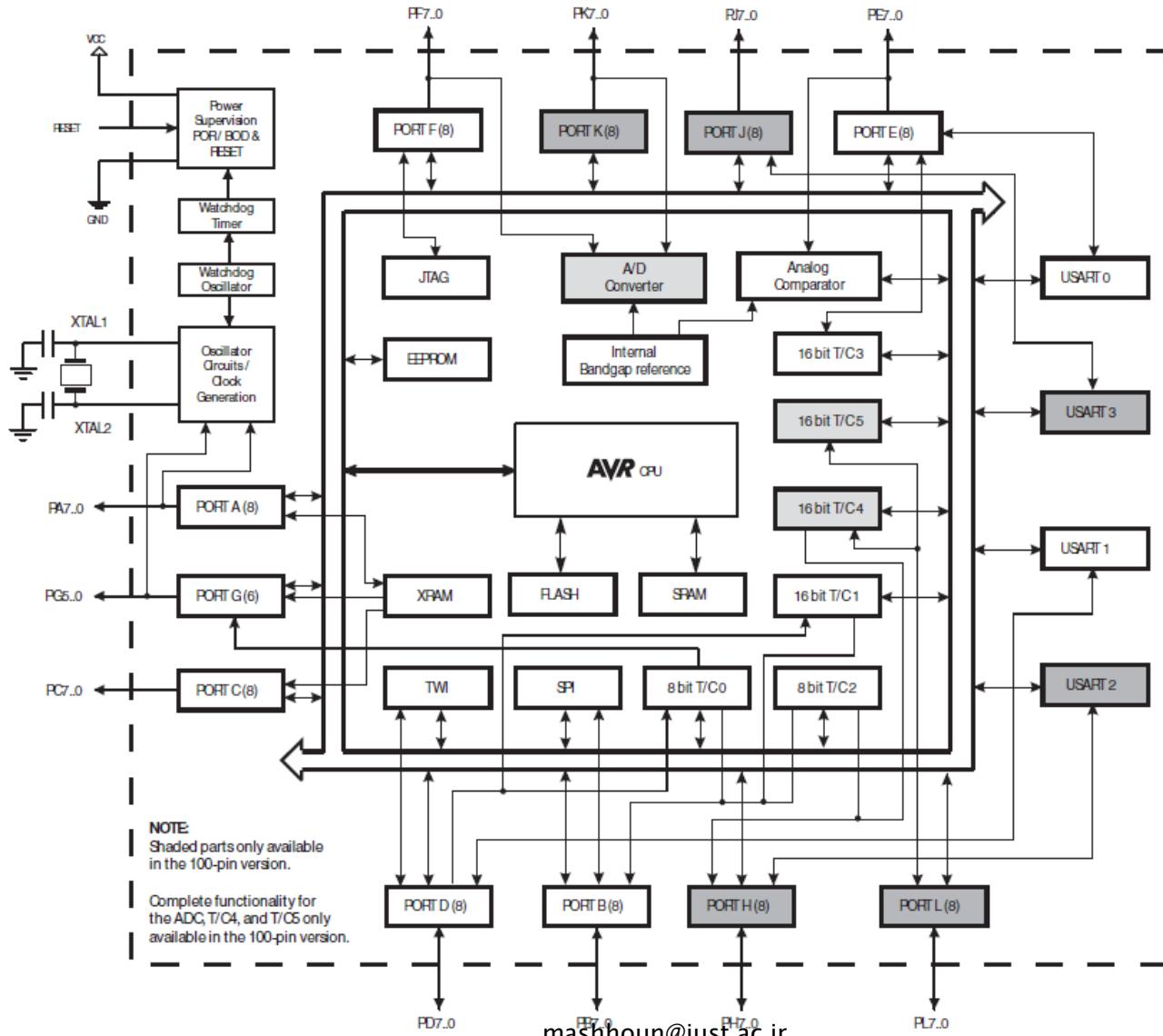
به عنوان مثال فرض کنید که منبع کلاک یک پالس با فرکانس ثابت یک مگاهرتز باشد. بنابراین فاصله ی هر لبه پایین رونده تا لبه ی پایین رونده بعدی، برابر با یک میکروثانیه خواهد بود. بدین ترتیب شمارنده طی مدت ۱۶ میکروثانیه رشته ی صفر تا پانزده را شمارنده کرده و پس از شانزده میکروثانیه پرچم (TOV) یک خواهد شد.

بعدا خواهید دید ه اساس زمان سنجی با میکروکنترلر AVR بر همین مبنای بوده و علاوه بر این امکاناتی پیشترفته نیز در اختیار کاربر قرار میگیرد.

## آشنایی با کانتر/تایмер در میکروکنترلر AVR

ATmega1280

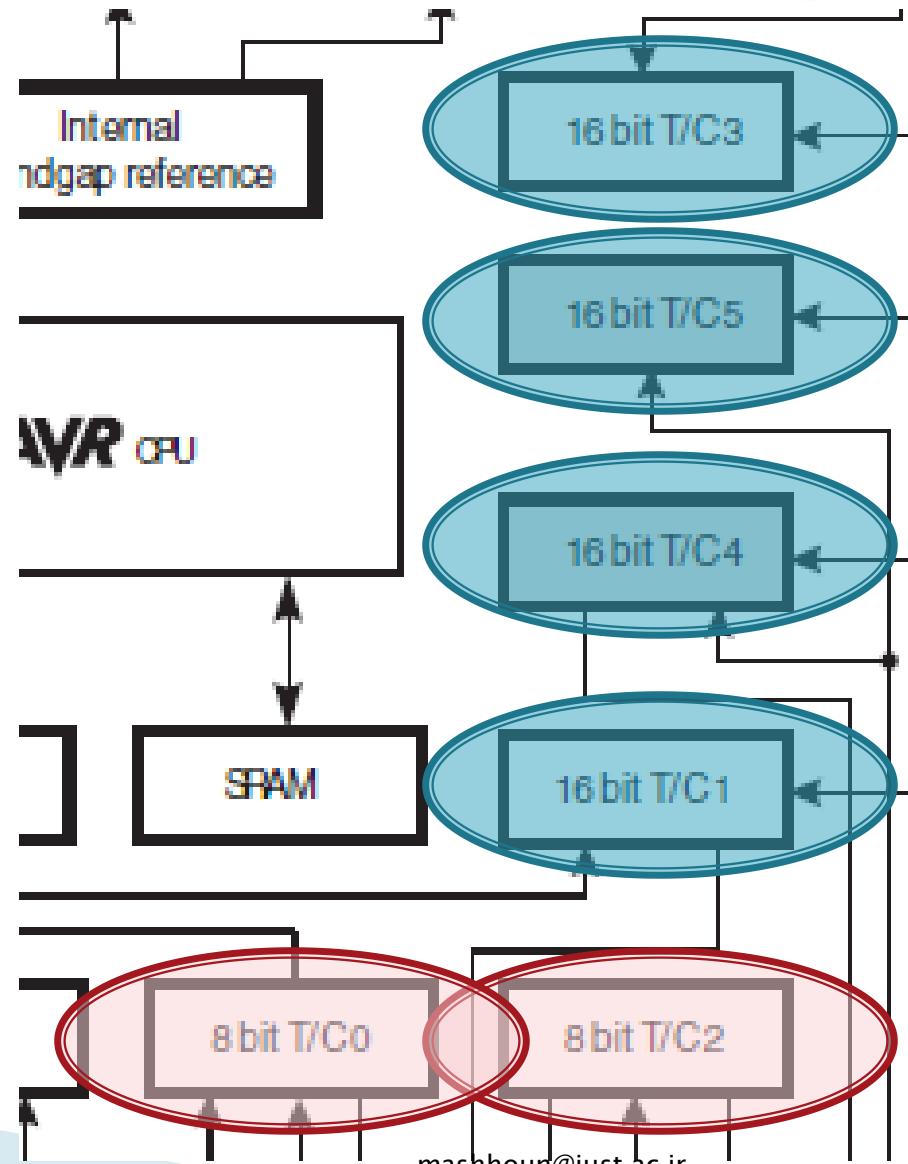
# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

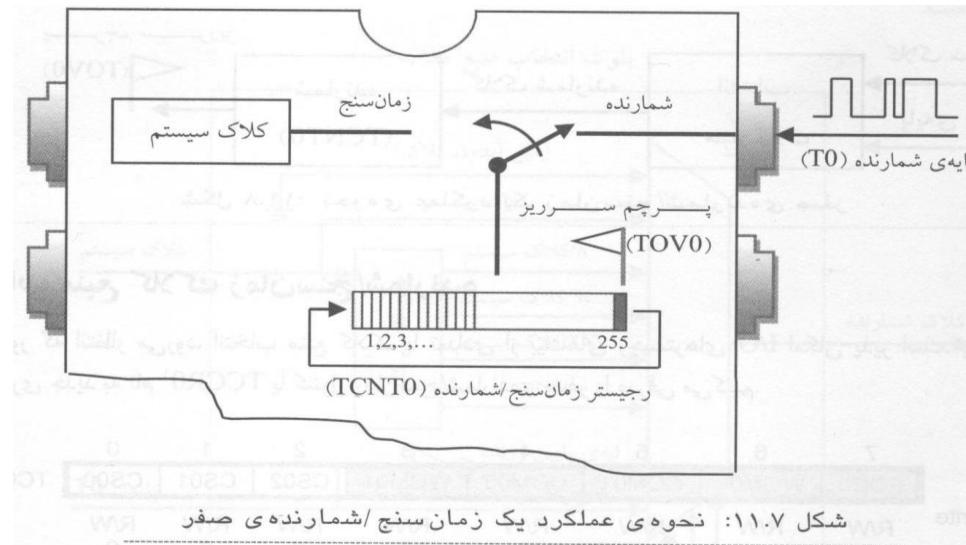
آشنایی با کانتر/تایмер در  
میکروکنترلر ATmega1280

همانطور که در شکل دیده میشود  
میکروکنترلر ATmega1280  
دارای دو تایمر/کانتر ۸ بیتی و چهار  
تایمر/کانتر ۱۶ بیتی است.



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

آشنایی کلی با تایمر/کانتر صفر  
در شکل زیر شکل بسیار ساده شده واحد زمان سنج صفر میکروکنترلر AVR را مشاهده میکنید.



شکل ۱۱.۷: نحوه عملکرد یک زمان سنج/شمارنده صفر

از آنجا که تایمر به صورت یک سخت افزار جنبی داخل میکروکنترلر قرار گرفته است، لذا دسترسی فیزیکی به پایه های مدار آن نداریم و باید از طریق ثباتهای I/O با آن ارتباط برقرار کنیم. اساسی ترین این ثباتها TCNT0 است که نقش خروجی فلایپ فلاپها را دارد.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی کلی با تایمر/کانتر صفر

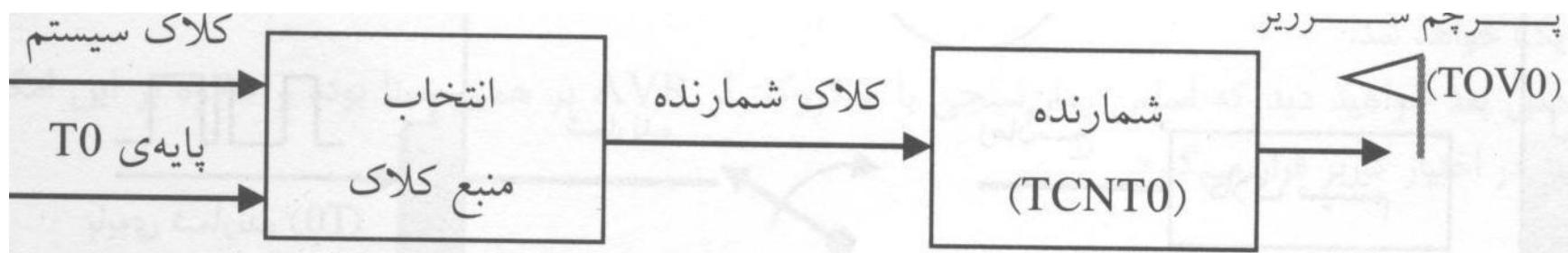
هر پالس کلاک باعث میشود تا محتوی ثبات TCNT0 یک واحد افزایش یابد این پالس بسته به وضعیت کلید، میتواند از دنیای بیرون و یا کلاک سیستم تامین شود. در صورتی که وضعیت شمارنده انتخاب شود. هر پالس اعمال شده از خارج (پایه T0 در میکروکنترلر) باعث افزایش یک واحدی ثبات TCNT0 میشود. در نقطه‌ی مقابل کلید میتواند در وضعیت تایمر قرار گیرد که در آن صورت، کلاک سیستم (تامین شده از اسیلاتور کریستالی، RC و غیره) باعث تغییر مقدار ثبات TCNT0 خواهد شد.

از آنجا که تایمر صفر، هشت بیتی است، ثبات TCNT0 میتواند مقادیر صفر تا ۲۵۵ را اختیار کند. در ابتدا مقدار این ثبات صفر است و با عمال پالس کلاک افزایش پیدا میکند. وقتی مقدار این ثبات ۲۵۵ باشد با اعمال کلاک بعدی، صفر میشود و مجدداً شمارش را از صفر شروع مینماید. و همزمان فلگ سرریز تایمر (TOV0) یک میشود.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی دقق تر با تایمر/کانتر صفر

برای ایجاد زمینه ای جهت بررسی سخت افزار تایمر، اجزاء معرفی شده قبلی را در نمودار بلوکی زیر ملاحظه نمایید.



- تایمر صفر شامل یک ثبات شمارنده به نام **TCNT0** است که نقش خروجی فلیپ فلاپها را دارد.
- این شمارنده با رسیدن به انتهای رشته شمارش (۲۵۵)، سریز شده و مجدداً از صفر میشمارد، در این لحظه فلگ **TOV0** یک میشود.
- ثبات **TCNT0** برای شمارش نیاز با پالس کلاک دارد. این پالس میتواند از دو منبع پایه **T0** یا کلاک سیستم تامین شود که در وضعیت اول، شمارنده یا کانتر نام دارد و در وضعیت دوم تایмер یا زمان سنج نامیده میشود.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## انتخاب منبع کلاک تایمر / کانتر

همانطور که انتظار می‌رود، انتخاب منبع کلاک با تعدادی از بیت‌های ثبات‌های I/O امکان پذیر است. بدین منظور ثبات جدیدی به نام TCCR0 یا کنترل تایمر/کانتر را معرفی می‌کنیم:

Bit	7	6	5	4	3	2	1	0	TCCR0
Read/Write	FOCO	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	
Initial Value	R/W 0								

بیت‌های [TCCR0[2:0] به نامهای CS0[2:0] وظیفه‌ی انتخاب کلاک را بر عهده دارند که عملکرد این بیت‌ها مطابق با جدول اسلاید بعدی است.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

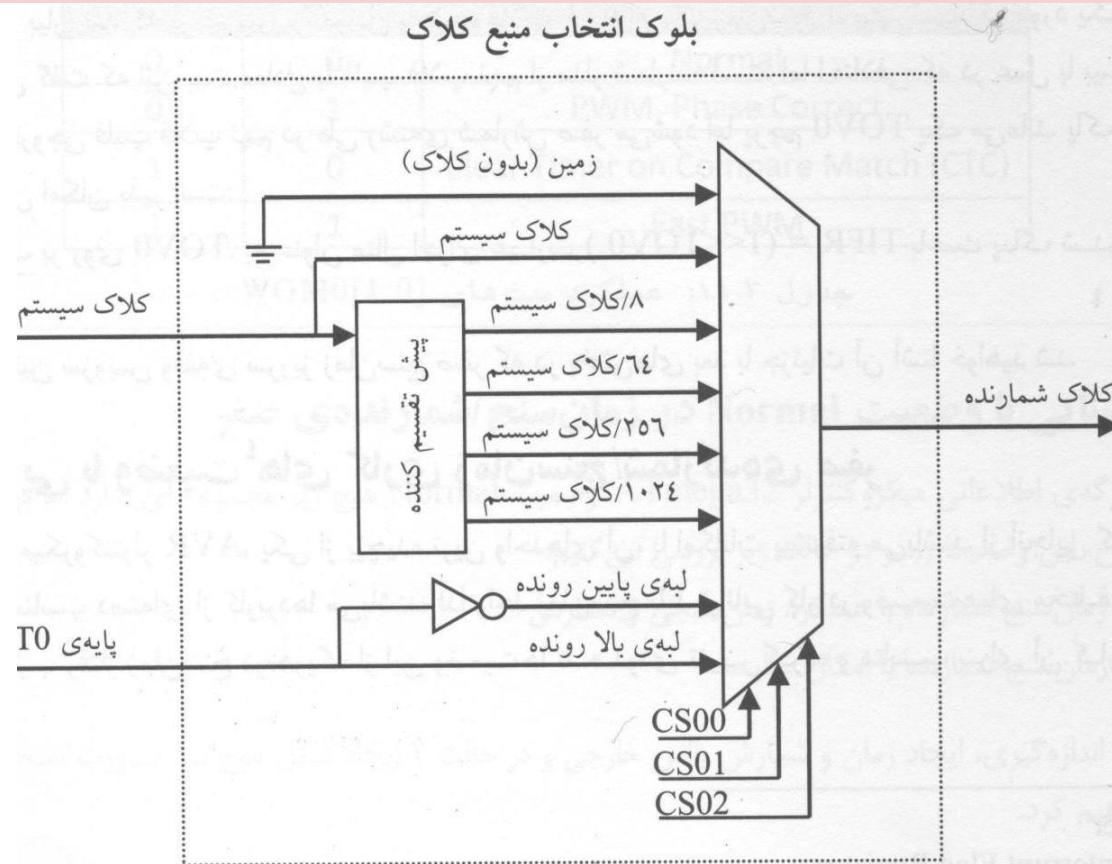
## انتخاب منبع کلاک تایمر / کانتر

CS02	CS01	CS00	منبع کلاک
0	0	0	بدون کلاک (متوقف)
0	0	1	کلاک سیستم (بدون تقسیم)
0	1	0	کلاک سیستم /۸
0	1	1	کلاک سیستم /۶۴
1	0	0	کلاک سیستم /۲۵۶
1	0	1	کلاک سیستم /۱۰۲۴
1	1	0	لبه‌ی پایین رونده‌ی پالس خارجی (T0)
1	1	1	لبه‌ی بالا رونده‌ی پالس خارجی (T0)

مطابق با جدول اگر تمام بیت‌های CS0[2:0] برابر با صفر باشند، کلاکی به ثبات TCNT0 اعمال نمی‌شود و شمارنده غیرفعال است. حال اگر مقدار 001 در این سه بیت قرار گیرد، کلاک سیستم به طور مستقیم به شمارنده اعمال خواهد شد.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## انتخاب منبع کلام/تایمر/کانتر



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## انتخاب منبع کلاک تایمر / کانتر

مطابق با جدول اگر تمام بیتهاي  $CS0[2:0]$  برابر با صفر باشند، کلاکی به ثبات  $TCNT0$  اعمال نمیشود و شمارنده غیرفعال است. حال اگر مقدار ۰۰۱ در این سه بیت قرار گیرد، کلاک سیستم به طور مستقیم به شمارنده اعمال خواهد شد.

از آنجا که معمولاً کلاک سیستم دارای فرکانس بالایی است، لذا اعمال این کلاک به شمارنده‌ی ۸ بیتی باعث سریز شدن سریع آن میشود. بنابراین بهتر است تا این کلاک توسط پیش تقسیم کننده بر عددی تقسیم شود. پیش تقسیم کننده میتواند کلاک ورودی را بر اعداد ۸، ۶۴، ۲۵۶ و ۱۰۲۴ تقسیم نماید. برای این منظور باید بیتهاي  $CS0[2:0]$  را بارگذاری نمود.

# آشنایی با میکروکنترولر AVR بخش تایمر/کانتر

## پرچم سرریز تایمر/کانتر

فلگ TOV0 بیت صفر از ثبات TIFR است.

Bit	7	6	5	4	3	2	1	0	TIFR
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

قبلاً بیان شد که فلگ سرریز، معادل با فلیپ فلاپ  $n+1$  می‌باشد. در مورد یک تایмер ۸ بیتی میتوان گفت که این بیت معادل با فلیپ فلاپ نهم از مدار شمارنده است. اما اختلافی که در عمل با بیت نهم دارد این است که خروجی فلیپ فلاپ در طی رشته‌ی شمارش صفر می‌شود اما فلگ TOV0 یک میماند. پاک کردن این بیت به دو روش امکان پذیر است:

۱- اجرای روتین سرویس وقفه‌ی سرریز تایмер صفر

۲- نوشتمن یک بر روی TOV0

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با وضعیتهای کاری تایمر/کانتر صفر

تایмер/کانتر صفر میکروکنترلر AVR دارای چهار وضعیت کاری است:

- 1– Normal
- 2– CTC (Clear Timer on Compare Match)
- 3– Fast PWM
- 4– Phase correct PWM

باید بدانید که رفتار تایمر کانتر در وضعیتهای ۱ و ۲ مناسب برای شمارش پالس‌های خارجی، اندازه گیری زمان، ایجاد زمانهای مورد نظر و ایجاد موج مربعی است و وضعیتهای ۲ و ۳ مناسب تولید موج PWM هستند.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با وضعيت‌های کاری تایمر/کانتر صفر

انتخاب یکی از این چهار وضعیت کاری، با بارگذاری عدد مناسب در بیت‌های **WGM0[1:0]** امکان پذیر است.

Bit	7	6	5	4	3	2	1	0	TCCR0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

نحوه‌ی انتخاب وضعیت مورد نظر با این دو بیت در جدول زیر آمده است:

WGM01	WGM00	وضعیت زمان‌سنج
0	0	Normal
0	1	PWM, Phase Correct
1	0	Clear Timer on Compare Match (CTC)
1	1	Fast PWM

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با وضعیت نرمال در تایمر/کانتر صفر

این موضوع را در دو حالت زیر بررسی میکنیم:

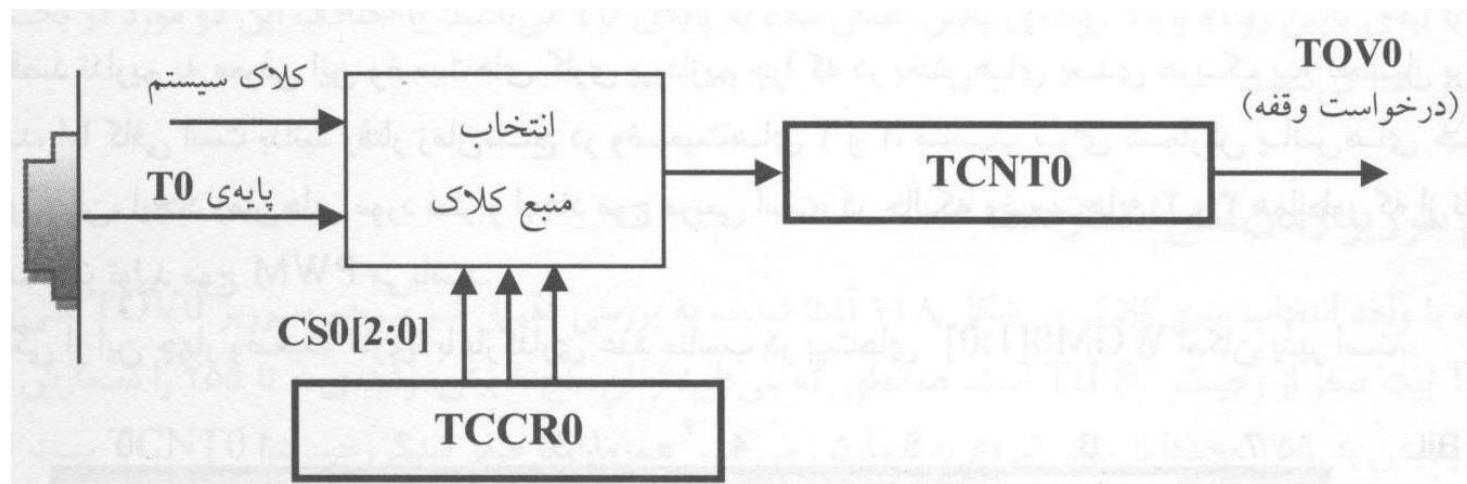
- ۱- تایمر/کانتر با عملکرد زمان سنجی و شمارش
- ۲- تایمر/کانتر با امکان مقایسه

در حالت یک، ایجاد زمان و شمارش پالس خارجی و در حالت دو ایجاد شکل موج به صورت سخت افزاری را بررسی خواهیم کرد.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## تایمر/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

نمودار بلوکی تایмер در این حالت مطابق با شکل زیر است. میدانید که شمارش، در ثبات TCNT0 انجام شده و با سرریز شدن آن پرچم TOV0، یک میشود. در این بخش خواهیم دید که در این لحظه میتواند توسط تایمر، وقفه درخواست شود.



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## تایمر/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

مدت زمانی که لازم است سپری شود تا تایمر/کانتر سرریز شود، بستگی به پیش تقسیم کننده دارد. جدول زیر مقادیر مختلف پیش تقسیم کننده و زمان سرریز را در هر یک از حالات نشان میدهد. در این جدول کلاک سیستم ۸ مگاهرتز فرض شده است.

زمان سنج/شمارنده‌ی صفر (کلاک سیستم: ۸ MHz)			
مقدار پیش تقسیم کننده	کلاک شمارنده (MHz)	هر پالس شمارنده ( $\mu s$ )	زمان سرریز شمارنده ( $\mu s$ )
۱	۸	۰.۱۲۵	۳۲
۸	۱	۱	۲۵۶
۶۴	۰.۱۲۵	۸	۲۰۴۸
۲۵۶	۰.۰۳۱۲۵	۳۲	۸۱۹۲
۱۰۲۴	۰.۰۰۷۸۱۲۵	۱۲۸	۳۲۰۰۰

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

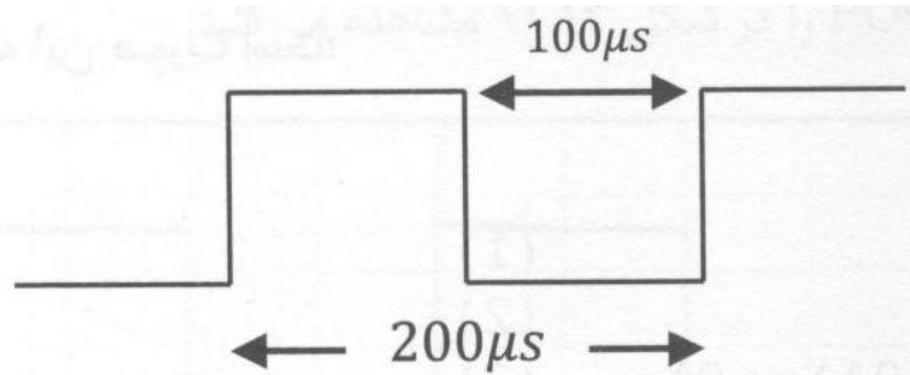
## تایмер/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

مثال : با فرض اینکه منبع کلک میکروکنترلر ۸ مگاهرتز است، پالس مربعی با فرکانس ۵ کیلوهرتز بر روی پایه PD0 ایجاد نمایید.

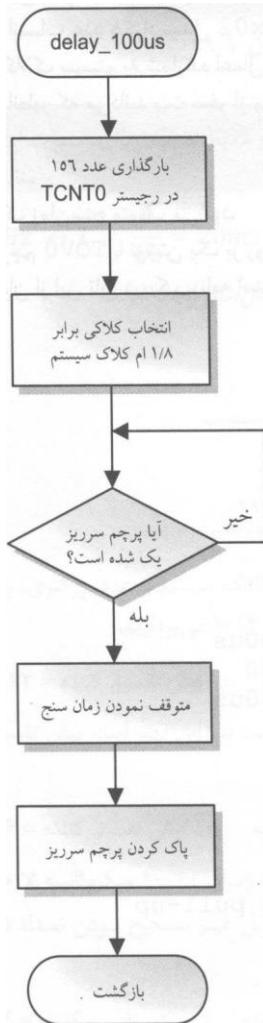
پیش از هر پیز باید زمان قطع و وصل موج مورد نظر را محاسبه نماییم:

$$f = 4\text{KHz} \rightarrow T = \frac{1}{5000} = 200\mu\text{s}$$

بنابراین مطابق شکل زیر لازم است تا مدت زمان ۱۰۰ میکروثانیه توسط تایمر ایجاد شود



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر



## تایمر/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

در صورتی که کلاک سیستم، توسط پیش تقسیم کننده بر عدد ۸ تقسیم شود، پالسی با فرکانس یک مگاهرتز به TCNT0 اعمال خواهد شد، در نتیجه فاصله‌ی هر تیک کلاک برابر با یک میکروثانیه است. بدین ترتیب پس از ۲۵۶ میکروثانیه فلگ TOV0 یک شده و برنامه را از بروز سرریز آگاه میکند.

اما همانطور که عنوان شد برای ایجاد شکل موج مطلوب، زمانی برابر با ۱۰۰ میکروثانیه نیاز است، راه حل این مشکل ساده است: TCNT0 یک ثبات قابل خواندن و نوشتן است، پس با بارگذاری عدد ۱۵۶ در این ثبات، رشته شمارش تایмер محدود به بازه ۱۵۶ تا ۲۵۵ شده و با صفر شدن زمان سنج، فلگ TOV0 یک میشود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## تایمر/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

در ابتدا عدد ۱۵۶ در ثبات TCNT0 بارگذاری شده و با انتخاب پیش تقسیم کننده ۸، زمان سنج شروع به شمارش کرده و پس از آن برنامه منتظر میماند تا تایмер سرریز شود. با یک شدن فلگ TOV0 زمان مورد نظر ایجاد شده است پس میتوان زمان سنج را متوقف نمود.

```
void delay_100us()
{
    TCNT0 = 156;
    TCCR0 = 0x02;
    while((TIFR & 0x01) ==0);
    TCCR0 = 0;
    TIFR |= 0x01;
}
```

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## تایمر/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

اکنون میتوان از این تابع در یک برنامه استفاده کرد و شکل موج مورد نظر مسئله را ایجاد کرد.

```
#include <mega32.h>
void delay_100us();
void init_io();

Void main(void)
{
    init_io();
    while (1)
    {
        PORTD |= (1<<PD0);
        delay_100us();
        PORTD &= ~ (1<<PD0);
        dela_100us();
    }
}
```

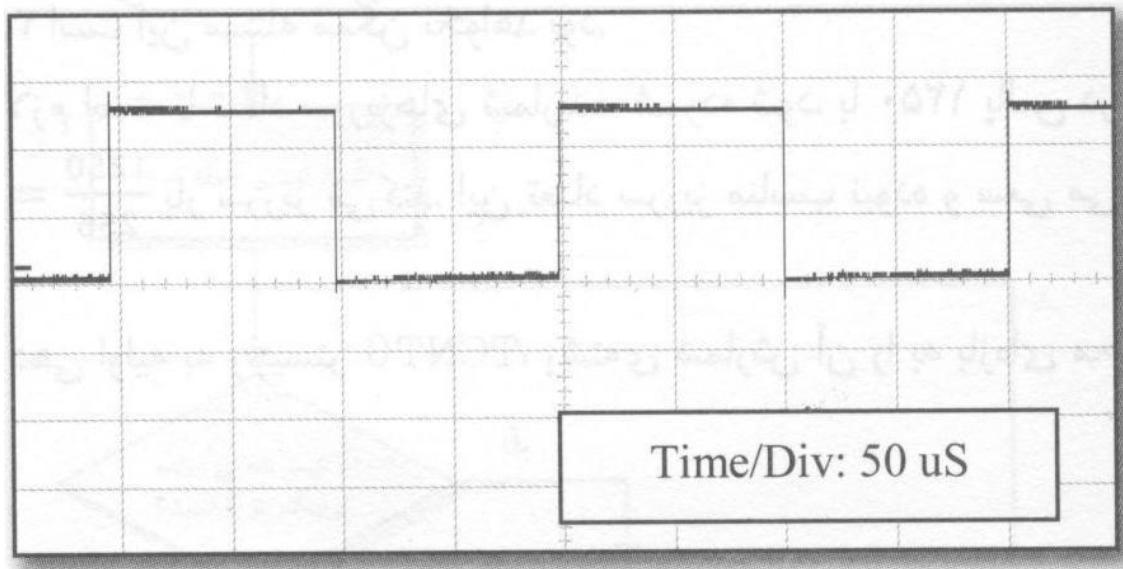
```
void init io()
{
    PORTD = 0x00;
    DDRD = 0x01;
}

void delay_100us()
{
    TCNT = 156;
    TCCR0 = 0x02;
    while((TIFR & 0x01) ==0);
    TCCR0 = 0;
    TIFR |= 0x01;
}
```

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

تایмер/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

شکل موج ایجاد شده روی پایه PDO در شکل زیر دیده میشود:



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

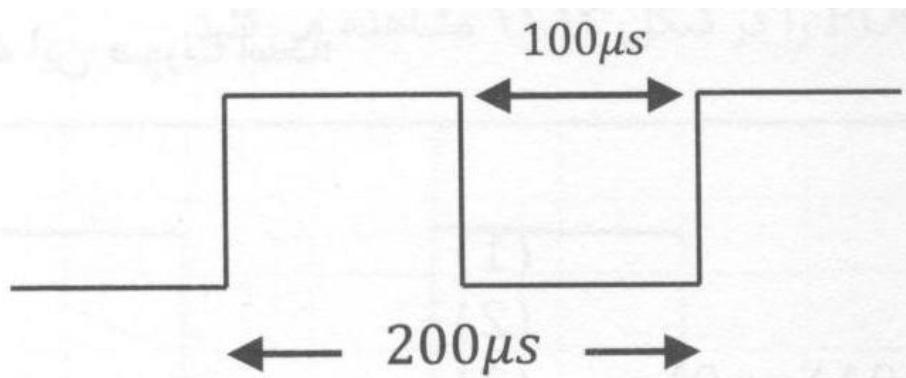
## تایмер/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

مثال : با فرض اینکه منبع کلاک میکروکنترل ۸ مگاهرتز است، پالس مربعی با فرکانس ۵۰ هرتز بر روی پایه PD0 ایجاد نمایید.

پیش از هر پیز باید زمان قطع و وصل موج مورد نظر را محاسبه نماییم:

$$f = 50Hz \rightarrow T = \frac{1}{50} = 20ms$$

بنابراین مطابق شکل زیر لازم است تا مدت زمان ۱۰ میلی ثانیه توسط تایمر ایجاد شود



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## تایمر/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

از آنجا که  $10$  میلی ثانیه زمانی نسبتا طولانی است سعی میکنیم با تقسیم کلاک سیستم بر عدد بزرگتری، پایین ترین فرکانس پالس ممکن را برای شمارنده تامین نماییم. بدین منظور ابتدا به بررسی  $10^{24}$  میپردازیم.

$$Counter_{clock} = \frac{8000000}{1024} = 7812.5Hz$$

با توجه به اینکه نتیجه تقسیم کلاک  $8$  مگاهرتز بر  $10^{24}$  مقداری صحیح نیست بنابراین بهتر است پیش تقسیم کننده  $256$  را بررسی کنیم:

$$Counter_{clock} = \frac{8000000}{256} = 31250Hz$$

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## تایمر/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

با این کلاک هر تیک شمارنده ۳۲ میکروثانیه خواهد بود. در نتیجه برای ایجاد ۱۰۰۰۰ میکروثانیه لازم است تا  $10000 / 32 = 312.5$  پالس به شمارنده اعمال شود که این چاسخ نیز به دلیل غیرصحیح بودن تعداد تیکهای شمارنده مناسب نمیباشد. تقسیم بر ۶۴ را آزمایش میکنیم.

$$Counter_{clock} = \frac{8000000}{64} = 125000Hz$$

با این کلاک هر تیک شمارنده ۸ میکروثانیه خواهد بود که در نتیجه برای ایجاد ۱۰۰۰۰ میکروثانیه لازم است تا  $10000 / 8 = 1250$  پالس به شمارنده اعمال شود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## تایمر/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

اگر برای تامین کلاک شمارنده از این فرکانس استفاده نماییم برای ایجاد زمان ۱۰ میلی ثانیه یا ۱۰۰۰۰ میکروثانیه، نیاز به شمارش رشته صفر تا ۱۲۵۰ خواهد بود. از آنجا که حداکثر مقدار شمارنده ۸ بیتی عدد ۲۵۵ است این مسئله ممکن نخواهد بود.

برای حل این مشکل نیز لازم است تعداد سرریزهای شمارنده شمرده شود. بدیهی است که شمارنده ۸ بیتی  $256/1250 = 0.88$  تعداد سرریز بباییم.

$$\frac{1250}{255} = 4.9$$

$$\frac{1250}{250} = 5.00$$

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

تایمر/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

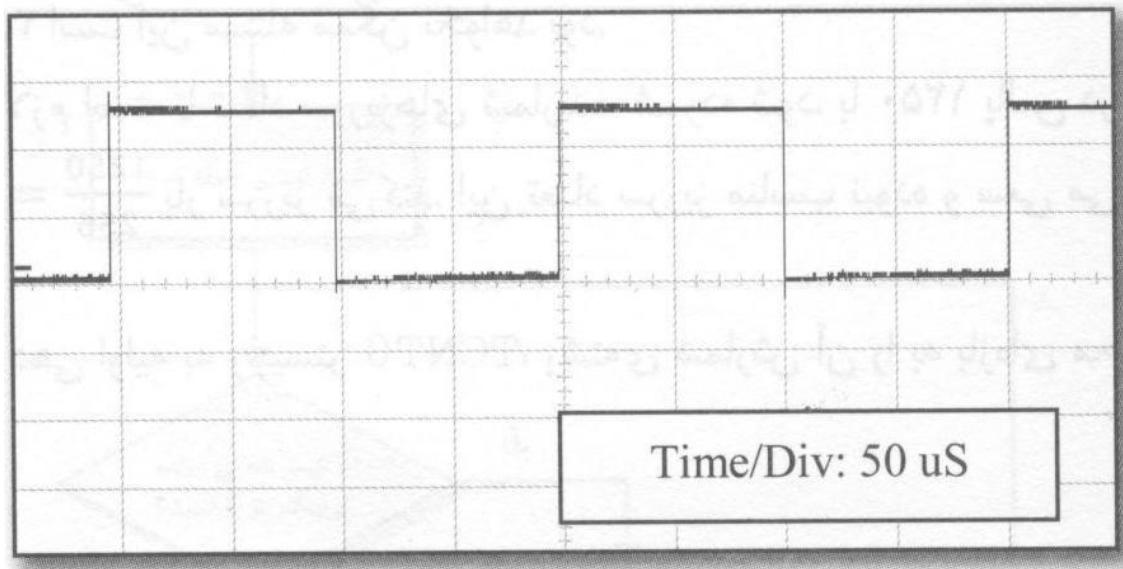
```
void delay_10ms()
{
    unsigned char i=0;
    while (i<5)
    {
        TCNT0 = 6;
        TCCR0 = 0x03;
        while((TIFR & 0x01) ==0);
        TCCR0 = 0;
        TIFR |= 0x01;
        i++;
    }
}
```

بدین ترتیب کافی است تا با هر بار سرریز شمارنده، در رجیستر TCNT0 مقدار اولیه ۶ بارگذاری شود تا رشته شمارش آن محدود به ۲۵۰ عدد محدود شود. بدین ترتیب با ۵ بار سرریز شدن زمان شمارنده ۱۰ میلی ثانیه بدست خواهد آمد.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

تایмер/کانتر با عملکرد زمان سنجی و شمارش در وضعیت نرمال

شکل موج ایجاد شده روی پایه PD0 در شکل زیر دیده میشود:



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با وقفه‌ی سرریز تایمر/کانتر صفر

همانطور که پیشتر اشاره شد، سرریز شمارنده لحظه مهمی محسوب می‌شود و یکی از منابع وقفه است. بدین ترتیب این امکان فراهم می‌شود تا در فاصله بین دو سرریز، به جای سرکشی پرچم TOV0، بخش‌های دیگری از برنامه اجرا شود.

برای فعال نمودن وقفه سرریز شمارنده، میتوان بیت TOIE0 از رجیستر TIMSK را یک نمود. در این شرایط اگر فعال ساز عمومی وقفه‌ها (I) یک شده باشد، یک شدن پرچم TOV0 میتواند باعث ایجاد وقفه شود. در این شرایط با اجرا شدن ISR به صورت خودکار پرچم سرریز پاک می‌شود.

**TIMSK0 – Timer/Counter Interrupt Mask Register**

Bit (0x6E)	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R/W	R/W	R/W	TIMSK0
Initial Value	0	0	0	0	0	0	0	0	

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با وقفه‌ی سرریز تایمر/کانتر صفر

```
#include <mega32.h>           برنامه قبلی را با استفاده از وقفه سرریز شمارنده بازنویسی کنید.  
void init_io();  
  
interrupt [TIM0_OVF] void timer0_ovf_isr(void)  
{  
    TCNT0 = 156;  
    PORTD ^= (1<<PD0);  
}  
  
void main(void)  
{  
    init_io();  
    while(1)  
    { }  
}
```

```
void init_io()  
{  
    PORTD = 0x00;  
    DDRD = 0x01;  
}  
  
    TCNT = 156;  
    TCCR0 = 0x02;  
    TIMSK = 0x01;  
    TCCR0 = 0;  
    #asm("sei");  
}
```

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با وقهه‌ی سرریز تایمر/کانتر صفر

همانطور که ملاحظه می‌کنید در ISR سرریز شمارنده، TCNT0 با عدد ۱۵۶ بارگذاری شده و سپس عبارتی اجرا می‌شود که PD0 را مکمل می‌کند. تفاوت مهم این برنامه و برنامه قبلی در این است که در این برنامه زمان ارزشمند پردازنده صرف چک کردن پرچم سرریز TOV0 نشده و بایک شدن پرچم به طور خودکار ISR اجرا می‌شود.

به عبارت دیگر این امکان وجود دارد تا در حلقه اصلی برنامه وظایف دیگری توسط CPU انجام شود.

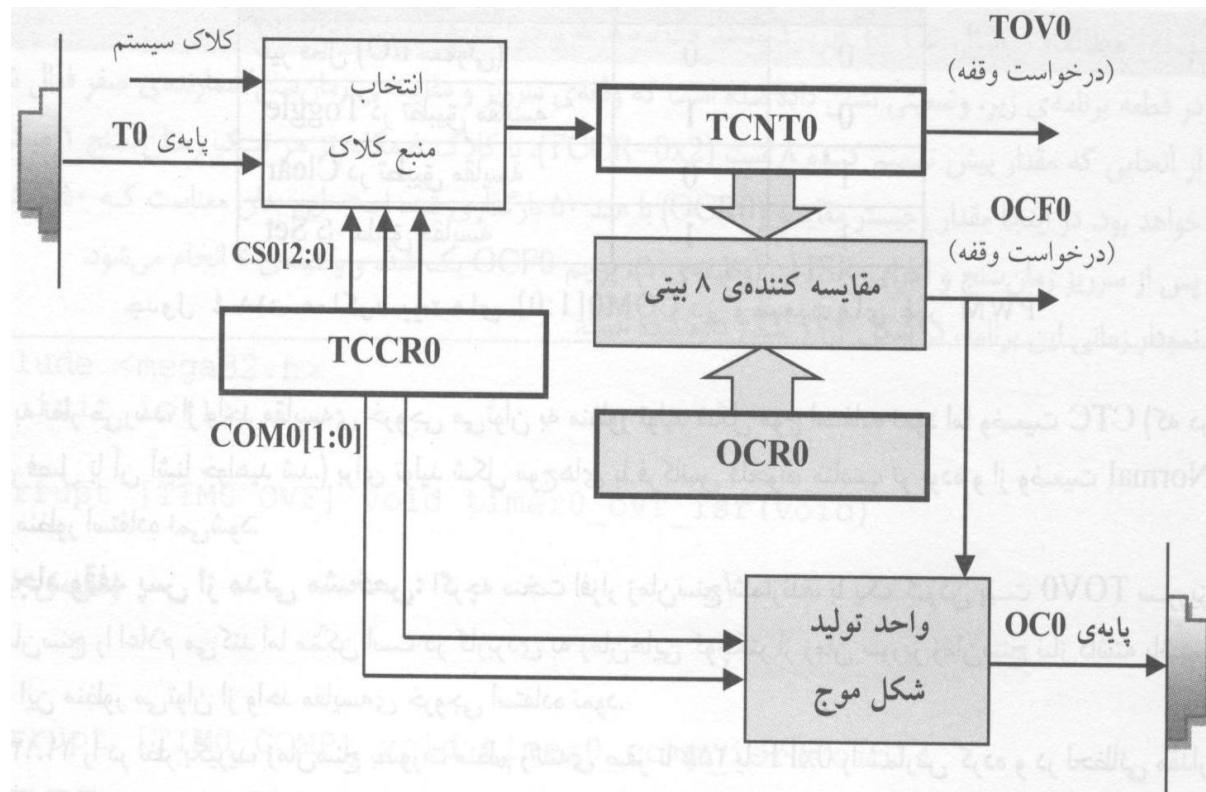
# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آزمایش عملی

برنامه ای بنویسید که یک LED روشن را از چپ به راست و راست به چپ حرکت دهد. تاخیر باید از پورت C (یا همان دیپ سوئیچها) خوانده شود. یک شمارنده چهار رقمی نیز روی سون سگمنتها روشن شود و از صفر رو به بالا به شمارد. شمارنده در هر ثانیه یکبار عدد خود را افزایش دهد. دقیت کنید که اعداد روی سون سگمنت هر ۲۰ میلی ثانیه یکبار باید سون سگمنت ها را بازنویسی نماید تا اطلاعاً بطور صحیح روی آنها دیده شود.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با زمان تایمر/کانتر با امکان مقایسه



همانطور که از نام این حالت مشخص است، تایمر/کانتر قابلیت مقایسه مقدار TCNT0 را به صورت سخت افزاری دارد. برای آشنایی با این قابلیت، ابتدا لازم است تا با واحد مقایسه خروجی در تایмер/کانتر صفر آشنا شوید.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با زمان تایمر/کانتر با امکان مقایسه

همانطور که ملاحظه میکنید، ثبات جدیدی به نام OCRO اضافه شده است. محتوی این ثبات در هر سیکل با مقدار TCNT0 مقایسه شده و در صورت برابری فلگ OCF0 در سیکل بعدی یک میشود. این لحظه برابری را تطبیق مقایسه (Compare Match) مینامیم. کاربردهای این تطبیق مقایسه عبارتند از:

۱- تولید شکل موج

۲- ایجاد وقفه پس از مدتی مشخص

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با زمان تایمر/کانتر با امکان مقایسه

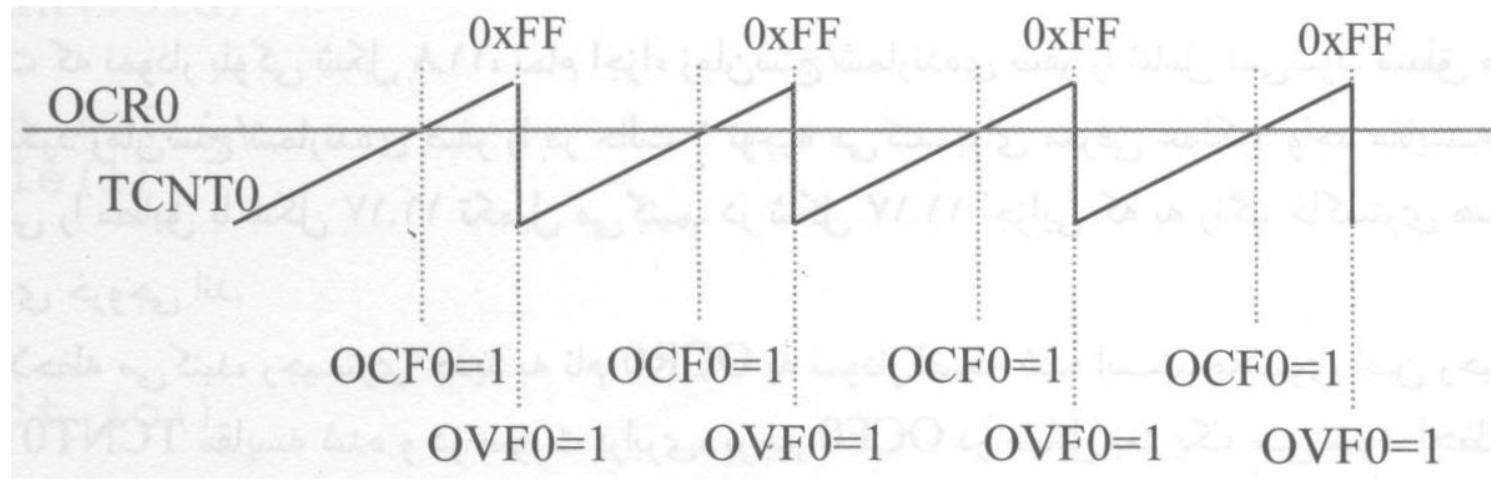
۱- تولید شکل موج - در لحظه یک شدن فلگ OCF0، فرمانی برای واحد تولید شکل موج صادر میشود. این واحد با توجه به تنظیماتی که از بیت‌های COM0[1:0] دریافت میدارد سطح منطقی پایه OC0 را

Bit	7	6	5	4	3	2	1	0	TCCR0
Read/Write	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	
Initial Value	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	تغییر میدهد.
اگرچه به نظر میرسد از واحد مقایسه خروجی میتوان به منظور تولید شکل موج استفاده نمود اما وضعیت CTC برای تولید شکل موجهای با فرکانس دلخواه مناسب تر بوده و از وضعیت نرمال به این منظور استفاده نمیشود.									
COM01		COM00		وضعیت پین OC0					
0		0		غیر فعال (I/O معمولی)					
0		1		در تطبیق مقایسه Toggle					
1		0		در تطبیق مقایسه Clear					
1		1		در تطبیق مقایسه Set					

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با زمان تایمر/کانتر با امکان مقایسه

۱- ایجاد وقفه پس از مدتی مشخص - اگر چه سخت افزار تایمر/کانتر با یک کردن بیت TOV0 سرریز تایмер را اعلام میکند اما ممکن است در کاربردی به زمانهایی کوچکتر از زمان سرریز تایmer نیاز داشته باشیم. به این منظور میتوان از واحد مقایسه خروجی استفاده کرد.



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با زمان تایمر/کانتر با امکان مقایسه

ممکن است این سوال پیش آید که میتوان با مقدار اولیه دادن به TCNT0 رشته شمارش آن را محدود نمود تا در فاصله زمانی مورد نظر فلگ TOV0 یک شود، پس مزیت واحد مقایسه چیست؟ پاسخ این است که اگر چه دو هر روش نتیجه یکسانی خواهند داشت اما با استفاده از واحد مقایسه تمام عملیات به صورت سخت افزاری انجام شده و در نتیجه برای مقدار اولیه دادن به TCNT0 وقت پردازنده تلف نمیشود. شرط لازم برای درخواست وقفه در لحظه تطبیق مقایسه یک بودن بیت OCIE0 از ثبات TIMSK است. همچنین لازم است بیت فعال ساز عمومی وقفه ها با دستور sei یک شود.

Bit	7	6	5	4	3	2	1	0	TIMSK
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## مثال: کاربرد واحد مقایسه خروجی در وضعیت نرمال

در قطعه برنامه زیر وضعیتی نشان داده است که وقفه سرریز و مقایسه زمان تایمر/کانتر صفر فعال شده اند. از آنجایی که مقدار پیش تقسیم کننده ۸ است ( $TCCR=0x02$ ), با کلاک ۸ مگاهرتز هر تیک تایмер یک میکروثانیه خواهد بود در اینجا مقدار ثبات مقایسه ( $OCR0$ ) با عدد ۵۰ بازگذاری شده است.

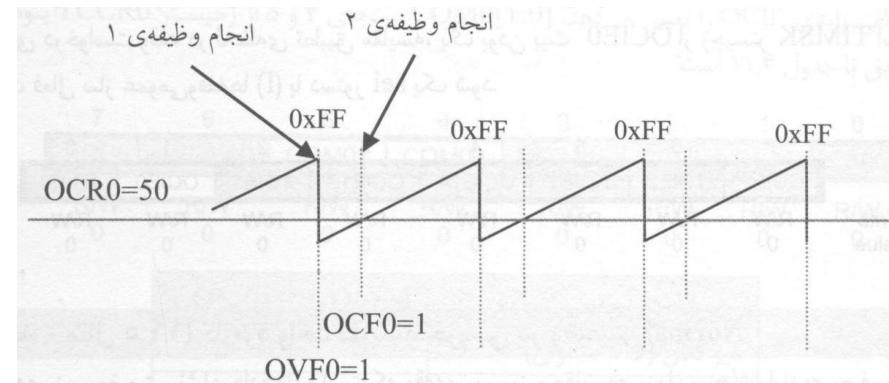
```
#include <mega32.h>
void init_io();

interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
    // وظیفه یک
}

interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
    // وظیفه دو
}

Void masin(void)
{
    init_io();
    while (1) {};
}
```

```
void init_io();
{
    TCNT0 = 60;
    OCR0 = 50;
    TCCR0 = 0x02;
    TIMSK = 0x03;
    #asm("sei");
}
```



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با وضعیت CTC در تایمر/کانتر صفر

اولین نکته در وضعیت جدید این است که مقدار بیتهاي WGM0[1:0] دیگر ۰۰ نبوده و با توجه به جدول مربوطه برابر ۰۱ است. (اسلاید شماره ۲۰)

Bit	7	6	5	4	3	2	1	0	TCCR0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Initial Value

0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---

اصلی ترین کاربرد CTC ایجاد تاخیر و شکل موج به صورت سخت افزاری است. بیاد دارید که در یکی از مثالهای مطرح داده برای ایجاد ۱۰۰ میکروثانیه تاخیر، ثبات TCNT0 را با عدد ۱۵۶ بارگذاری نمودیم، بدین ترتیب رشته شمارش این ثبات، محدود به عدد ۱۵۶ تا ۲۵۵ شد. حال سوال این است که آیا میتوان به جای مقدار اولیه دادن به شمارنده، رشته شمارش آن را از انتهای محدود نمود؟

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

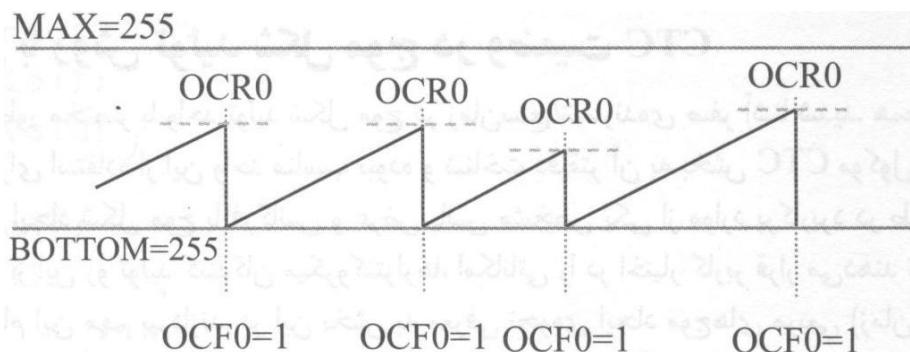
## آشنایی با وضعیت CTC در تایمر/کانتر صفر

قبل از پاسخ به این سوال سه واژه کلید که در ادامه مکرراً از آنها استفاده خواهیم کرد را تعریف میکنیم:

- BOTTOM: شمارنده زمانی که 0x00 شود به رسیده است.
- MAX: شمارنده زمانی که 0xFF (یا ۲۵۵ دسیمال) شود به رسیده است.
- TOP: شمارنده زمانی به TOP میرسد که برابر با بالاترین مقدار در رشته شمارش خود باشد. این مقدار بستگی به وضعیت کاری تایمر/کانتر دارد.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با وضعیت CTC در تایمر/کانتر صفر



حال پاسخ سوال مطرح شده را به این  
ترتیب میدهیم:

در وضعیت CTC میتوان TOP شمارنده را به یک عدد دلخواه کوچکتر از ۲۵۶ تغییر داد. همانطور که میدانید در وضعیت نرمال، TOP تایمر عدد ۲۵۵ بود و این عدد بالاترین مقدار در رشته شمارش ثبات TCNT است. اکنون با این مفهوم آشنا میشوید که در وضعیت CTC مقدار TOP آن عددی است که در ثبات OCR0 بارگذاری شده است. به بیان دیگر، بر خلاف وضعیت نرمال، شمارنده پس از تطبیق مقایسه از عدد OCR0 عبور نمیکند بلکه در همان لحظه مقدار شمارنده صفر میشود.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

شمارنده را در وضعیت CTC به شکلی تنظیم نمایید تا هر ۱۰۰ میکروثانیه وظیفه انجام شود

```
#include <mega32.h>
void init_io();

// Timer 0 output compare interrupt service
routine
interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
    // وظیفه یک
}
Void main(void)
{
    init_io();
    while (1) {};
}

void init_io()
{
    TCCR0 = 0x0A;
    OCR0 = 99;
    TIMSK = 0x02;
    #asm("sei");
}
```

همانطور که مشخص است تایمر/کانتر صفر در وضعیت CTC با  $\text{TOP} = 99$  تنظیم شده است. بنابراین رشته شمارش  $\text{TCNT0}$  محدود به صفر تا ۹۹ خواهد بود. از آنجایی که وقفه مقایسه تایمر/کانتر صفر فعال شده است، پس از ۱۰۰ پالس ساعت، ISR مربوط به آن اجرا شده و در آنجا، وظیفه یک انجام میشود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با روش تولید شکل موج در وضعیت CTC

همانطور که میدانید ایجاد شکل موج با فرکانس و عرض پالس مشخص یکی از موارد پرکاربرد در طراحی سیستمهای میکروکنترلری است. از اینرو تولید کنندگان میکروکنترلرها، امکاناتی را در اختیار کاربر قرار میدهند تا بتوانند به صورت سخت افزاری به انجام این مهم بپردازنند.

برای درک اهمیت مثالی را در نظر بگیرید که وظیفه اصل آن معکوس نمودن یکی از پایه های  $1/O$  بود. در این مثال با هر بار تطبیق مقایسه (پس از گذشت ۱۰۰ میکروثانیه) آن پایه مورد نظر تغییر وضعیت داده میشد و بدین ترتیب پالسی را با دو تناوب ۲۰۰ میکروثانیه ایجاد نمودیم. اما اشکال این روش آن است که اینکار زمانی را از پردازنده اشغال میکند. برای حل این مشکل میتوان از واحد تولید شکل موج استفاده نمود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با روش تولید شکل موج در وضعیت CTC

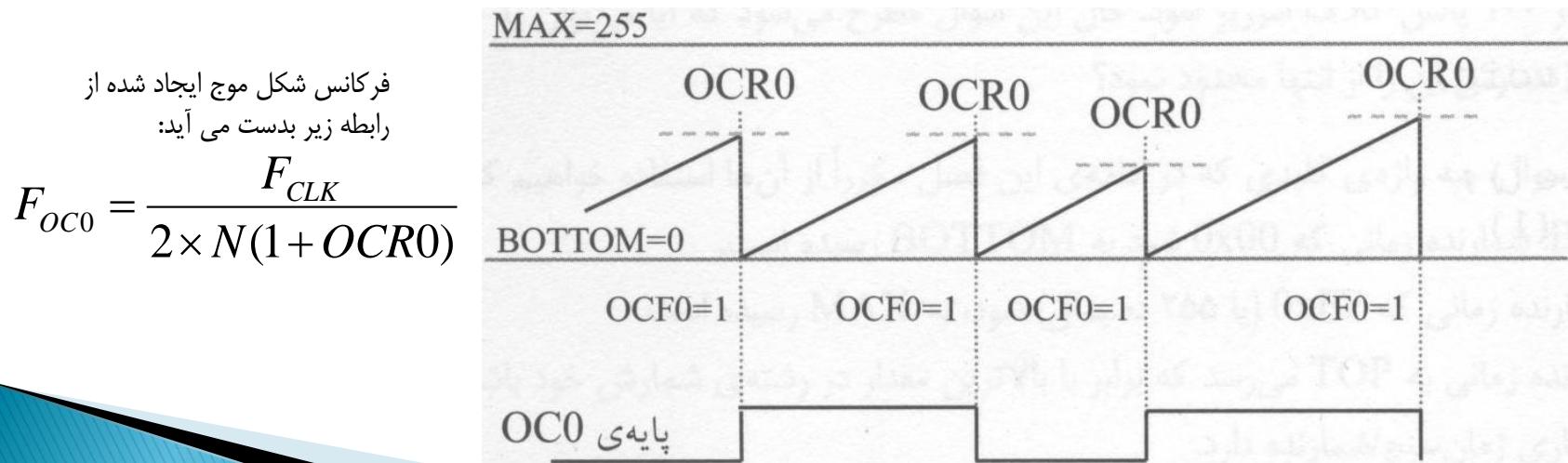
همانطور که میدانید ایجاد شکل موج با فرکانس و عرض پالس مشخص یکی از موارد پرکاربرد در طراحی سیستمهای میکروکنترلری است. از اینرو تولید کنندگان میکروکنترلرها، امکاناتی را در اختیار کاربر قرار میدهند تا بتوانند به صورت سخت افزاری به انجام این مهم بپردازنند.

برای درک اهمیت مثالی را در نظر بگیرید که وظیفه اصل آن معکوس نمودن یکی از پایه های  $1/0$  بود. در این مثال با هر بار تطبیق مقایسه (پس از گذشت  $100$  میکروثانیه) آن پایه مورد نظر تغییر وضعیت داده میشند و بدین ترتیب پالسی را با دو تناوب  $200$  میکروثانیه ایجاد نمودیم. اما اشکال این روش آن است که اینکار زمانی را از پردازنده اشغال میکند. برای حل این مشکل میتوان از واحد تولید شکل موج استفاده نمود.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با روش تولید شکل موج در وضعیت CTC

میدانید که در لحظه تطبیق مقایسه فلگ OCF0 یک شده و امکان درخواست وقفه نیز وجود دارد. یک شدن این فلگ به صورت داخلی فرمانی را برای واحد تولید شکل موج صادر کرده و آن واحد نیز باتوجه به وضعیت بیت‌های [COM0:1] پایه OC0 را تغییر میدهد. مزیت مهم این روش آن است که تمام عملیات لازم برای تولید شکل موج به صورت خودکار انجام می‌شود و برنامه درگیر آن نمی‌گردد و اشکل این خواهد بود که تنها بر روی OC0 (پایه شماره ۴ قطعه ATmega32) می‌توان شکل موج ایجاد نمود.



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## بازنویسی مثال قبلی

در یکی از مثالهای قبلی برای ایجاد فرکانس ۵۰ کیلوهرتز برنامه نقش مهمی را دارا بود. اما در وضعیت CTC همانطور که انتظار می‌رود تمام عملیات را سخت افزار انجام داده و برنامه تنها شامل مقداردهی اولیه در

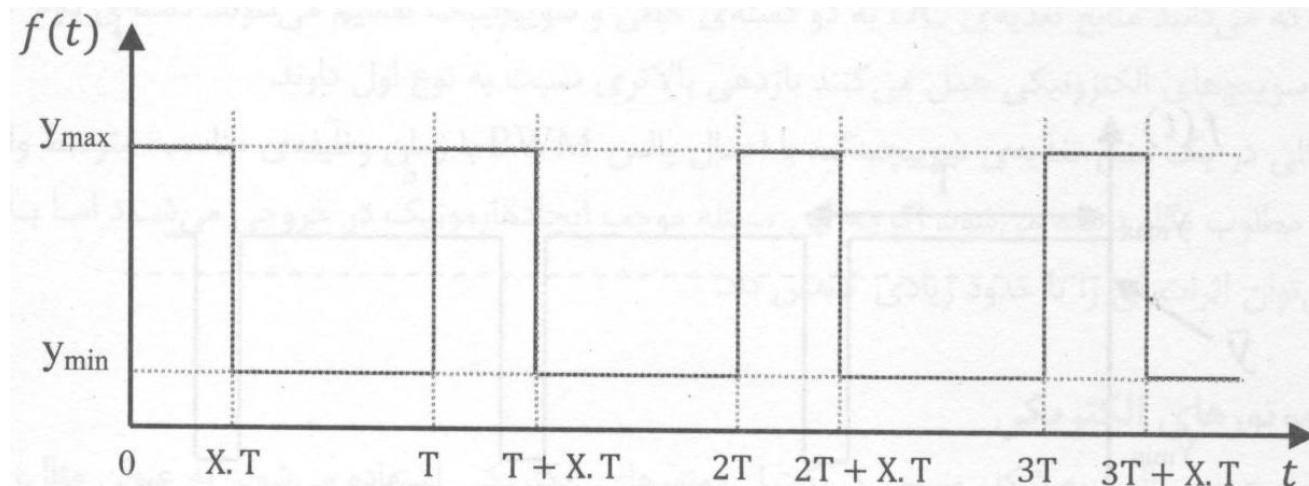
```
#include <mega32.h>
void init_io();
Void main(void)
{
    init_io();
    while (1) {};
}
void init_io();
{
    DDRB = 0x08;           //Output OC0
    TCCR0 = 0x1A;          //Prescaler = 1 and Mode: CTC
                           //Toggle on compare match
    OCR0 = 99;             //Set for 100 count
}
```

تابع `io_init()` است.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با مفهوم مدولاسیون عرض پالس (PWM)

معمولاً هدف از مدوله سازی عرض پالس یک سینگنال یا منبع توان، انتقال اطلاعات و یا کنترل توان تحویلی به بار است. بدین منظور لازم است تا عرض آن پالس، تحت تاثیر یک سینگنال مدوله کننده تغییر نماید. برای روشن شدن موضوع، شکل زیر را در نظر بگیرید:



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با مفهوم مدولاسیون عرض پالس (PWM)

در اینجا یک پالس به نام  $f(t)$  که دامنه آن بین  $y_{\min}$  تا  $y_{\max}$  و زمان وظیفه آن  $X$  است نشان داده شده است. بدیهی است که مقدار متوسط این موج از رابطه زیر بدست می‌آید:

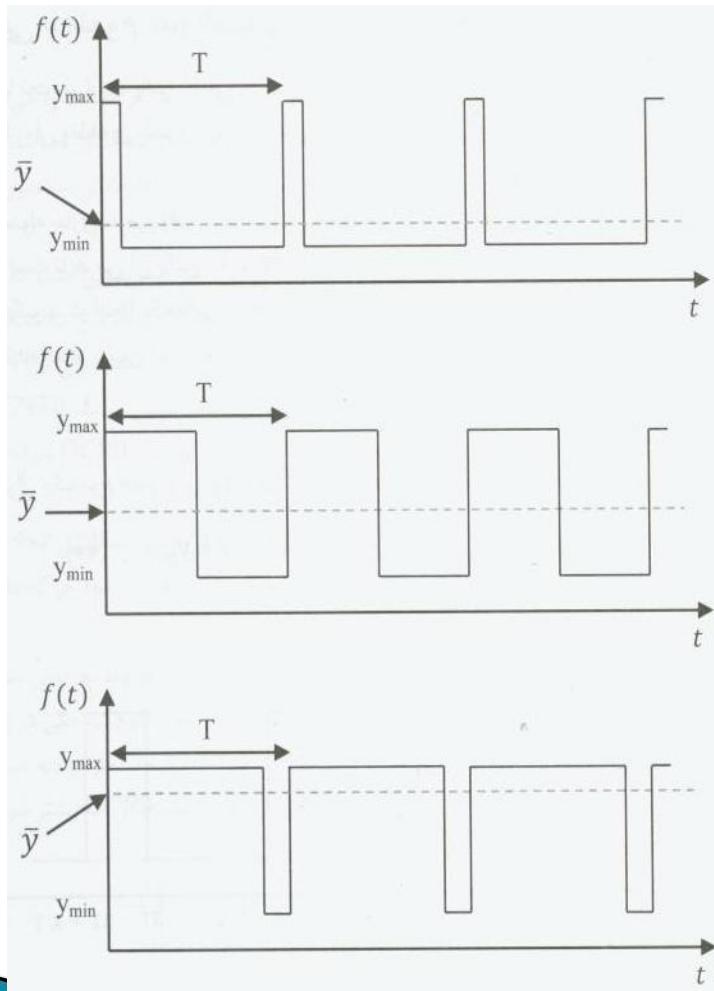
$$\bar{y} = \frac{1}{T} \int_0^T f(t) dt$$

از آنجا که انتگرال یک موج متناوب بر روی یک دوره تناوب برابر سطح زیر آن است  $\bar{y}$  آن برابر است با:

$$\bar{y} = \frac{1}{T} (X \cdot T \cdot y_{\max} + (T - X \cdot T) \cdot y_{\min}) = X \cdot y_{\max} + (1 - X) \cdot y_{\min}$$

با توجه به این رابطه مشخص است که مقدار متوسط سیگنال به طور مستقیم وابسته به زمان وظیفه است.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر



## آشنایی با مفهوم مدولاسیون عرض پالس (PWM)

در شکل‌های زیر مقدار متوسط یک موج PWM به سه زمان وظیفه متفاوت نشان داده شده است. ملاحظه می‌کنند که با افزایش عرض پالس مقدار  $\bar{y}$  افزایش می‌باید.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

برخی از کاربردهای مدولاسیون عرض پالس

- انتقال اطلاعات
- الکترونیک قدرت
- منابع تغذیه سوئیچینگ
- کنترل موتورهای الکتریکی
- کاربردهای صوتی

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با وضیت Fast PWM در تایمر/کانتر صفر

اولین نکته این که برای تنظیم نمودن تایمر/کانتر لازم است تا بیت های WGM0[1:0] برابر 11 تنظیم شوند. دومین نکته این است که عملکرد بیتهاي COM0[1:0] در این وضعیت مطابق جدول زیر است:

COM01	COM00	وضعیت پین OC0
0	0	غیر فعال (I/O معمولی)
0	1	رزرو شده
1	0	کردن OC0 در تطبیق مقایسه و یک کردن آن در TOP (Fast PWM غیرمعکوس)
1	1	یک کردن OC0 در تطبیق مقایسه و پاک کردن آن در TOP (Fast PWM معکوس)

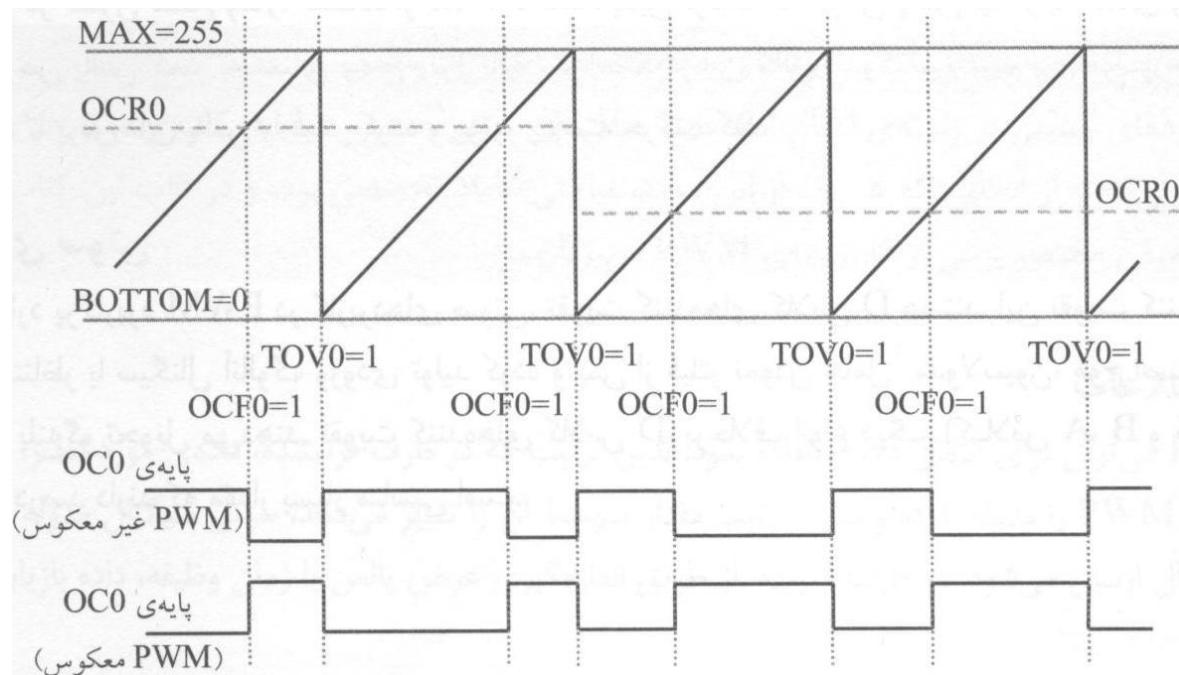
همانگونه که مشاهده میکنید، عملکرد این بیت در وضعیت Fast PWM بسیار متفاوت از وضعیتهاي غیر PWM است.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با وضیت Fast PWM در تایمر/کانتر صفر

در وضعیت PWM غیرمعکوس، شمارنده از BOTTOM تا MAX به صورت صعودی میشمارد، در لحظه تطبیق مقایسه، پایه OC0 پاک شده و با رسیدن به BOTTOM مجدداً یک میشود.

در وضعیت PWM معکوس نیز، شمارنده از MAX تا BOTTOM میشمارد، در لحظه تطبیق مقایسه پایه OC0 یک شده و با رسیدن به BOTTOM صفر خواهد شد.



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با وضیت Fast PWM در تایمر/کانتر صفر

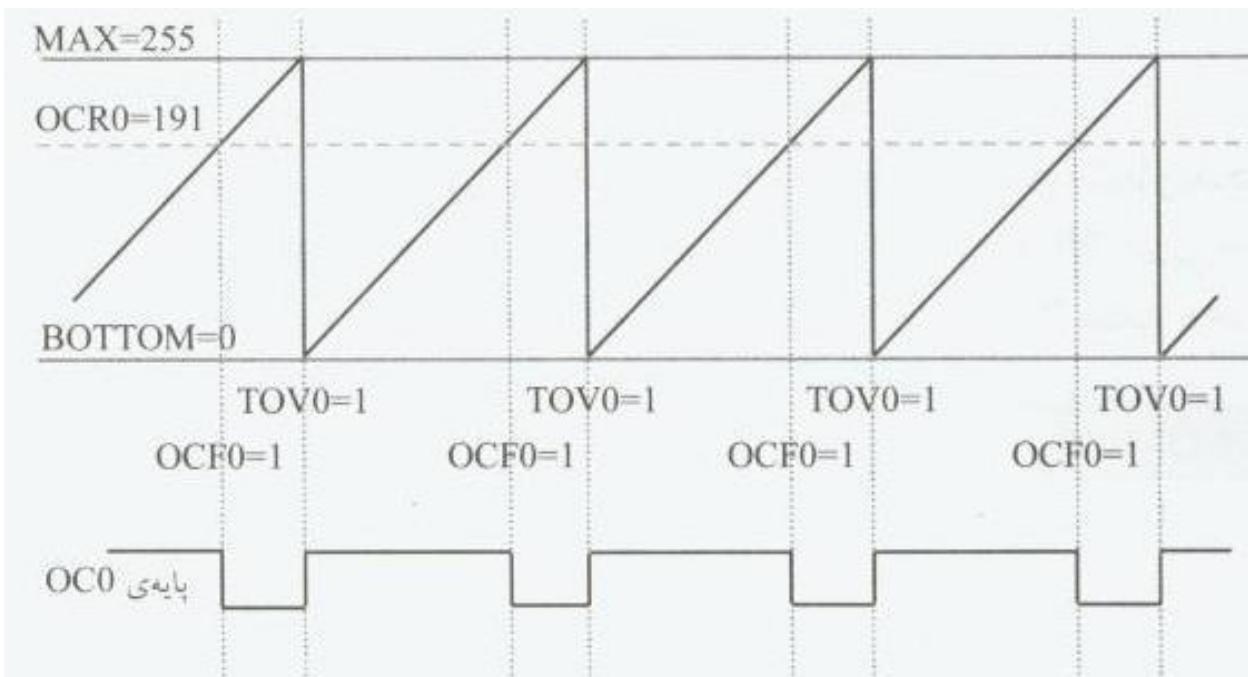
انتساب مقادیر خاص به ثبات OCR0 باعث ایجاد حالت‌های زیر می‌شود:

- در صورتی که مقدار OCR0 برابر با صفر شود، با هر بار سرریز شمارنده یک پالس سوزنی در خروجی ایجاد خواهد شد..
- بارگذاری مقدار ۲۵۵ در OCR0 موجب می‌شود تا بسته به وضعیت خروجی بیتهاي [OC0[1:0] پین OCR0 همواره صفر یا یک باشد. در صوتر یکه COM0[1:0] برابر با ۲ باشد. خروجی همواره یک بوده و در صورتی که این دو بیت برابر ۳ باشند (PWM معکوس) خروجی همواره صفر است.
- فرکانس موج خروجی در وضعیت Fast PWM از رابطه زیر بدست می‌آید: (منظور از N مقدار پیش تقسیم کننده است)

$$f = \frac{f_{CLK}}{N \times 256}$$

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: ایجاد یک موج PWN غیرمعکوس با دوره تناوب ۲۵۶ میکروثانیه و زمان وظیفه %۷۵ روی پایه OC0



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: ایجاد یک موج PWN غیرمعکوس با دوره تناوب ۲۵۶ میکروثانیه

```
#include <mega32.h>
void init_io();
Void main(void)
{
    init_io();
    while (1) {};
}
void init_io();
{
    DDRB = 0x08;           //Output OC0
    PORTB = 0X00
    TCCR0 = 0x6A;          //Prescaler = 8
                           //Mode: non-Inverting Fast PWM
    OCR0 = 191;            //Duty cycle : 75%
}
```

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با وضیت Phase Correct PWM در تایمر/کانتر صفر

اولین نکته این که برای تنظیم نمودن تایمر/کانتر لازم است تا بیت های WGM0[1:0] برابر 01 تنظیم شوند. دومین نکته این است که عملکرد بیتهاي COM0[1:0] در این وضعیت مطابق جدول زیر است:

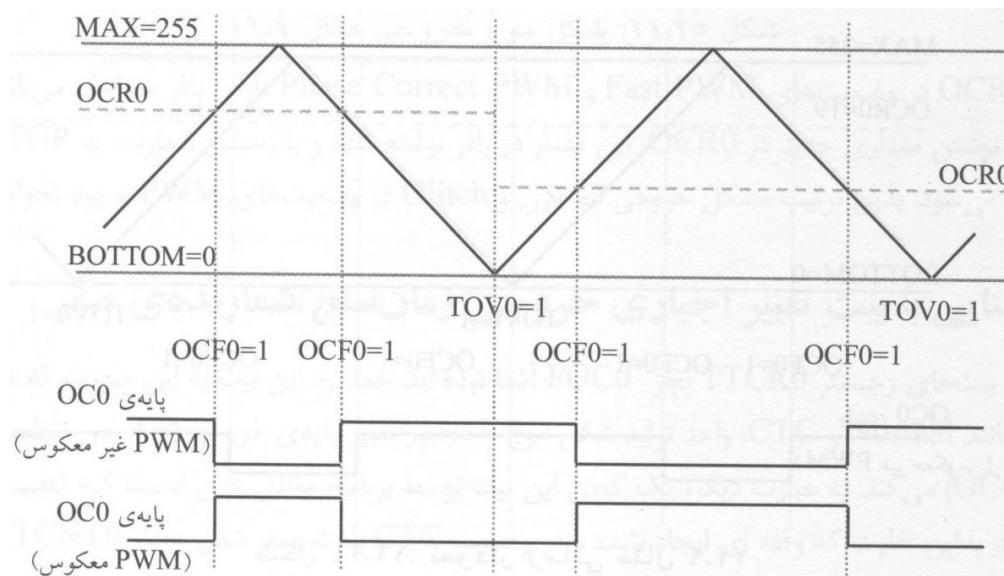
COM01	COM00	وضعیت پین OC0
0	0	غیر فعال (I/O معمولی)
0	1	رزرو شده
1	0	پاک کردن OC0 در تطبیق مقایسه به صعودی و یک کردن آن در به نزولی (PWM غیرمعکوس)
1	1	یک کردن OC0 در تطبیق مقایسه به صعودی و پاک کردن آن در به نزولی (PWM معکوس)

در این وضعیت، شمارنده مرتبا از BOTTOM تا MAX شمرده و با رسیدن به MAX به صورت نزولی تا BOTTOM میشمارد.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با وضیت Phase Correct PWM در تایمر/کانتر صفر

بنابراین طبق شکل زیر مقدار ثبات TCNT0 در دو لبه شمارش با OCR0 برابر میشود. در حالت PWM غیرمعکوس، تطبیق مقایسه روی داده در لبه شمارش صعودی پین را صفر کرده و تطبیق مقایسه در لبه شمارش صعودی پین OC0 را یک میکند. در مورد PWM معکوس، وضعیتی عکس برقرار است.

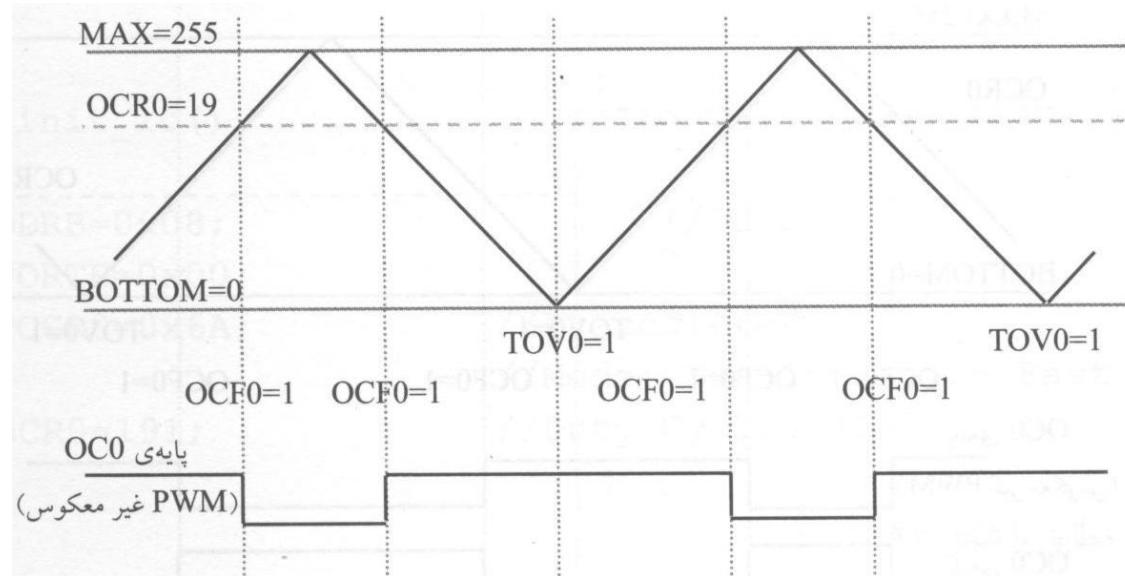


# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال ایجاد یک موج PWM غیرمعکوس با فرکانس تقریبی دو کیلوهرتز و زمان وظیفه ۷۵٪ بر روی پایه ۰ OC0

```
#include <mega32.h>
void init_io();
Void main(void)
{
    init_io();
    while (1) {};
}

void init_io();
{
    DDRB = 0x08;      //Output OC0
    PORTB = 0X00
    TCCR0 = 0x62;    //Prescaler = 8
                    //Mode: non-Inverting Phase Correct PWM
    OCR0 = 192;     //Duty cycle : 75%
}
```



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با سخت افزار تایمر/کانتر یک

یا تایмер/کانتر ۱۶ بیتی از دقت بالاتری برخوردار است. بدین ترتیب ثبات TCNT1 ۱۶ بیتی است و میتواند رشته صفر تا ۶۵۵۳۵ را شمارش کند. مزایای این مسئله از دو دیدگاه قابل بررسی است:

- ۱- در وضعیتهاي PWM شمارنده دیرتر سرریز شده و در نتیجه امکان شمارش پالسهاي بیشتر امکان پذیر است. امكان ایجاد زمانهای طولانی تر وجود خواهد داشت.
- ۲- از نقطه نظر وضعیتهاي PWM شکل موج تولید شده، گامهای کوچکتری در خروجی داشته و به عبارت دیگر رزولوشن بالاتری دارد.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با سخت افزار تایمر/کانتر یک

### دو واحد مقایسه مستقل

بدین ترتیب این امکان وجود دارد تا دو شکل موج (اعم از مربعی یا PWM) به طور همزمان و مستقل بر روی دو کanal خروجی تایمر/کانتر یک ایجاد شود.

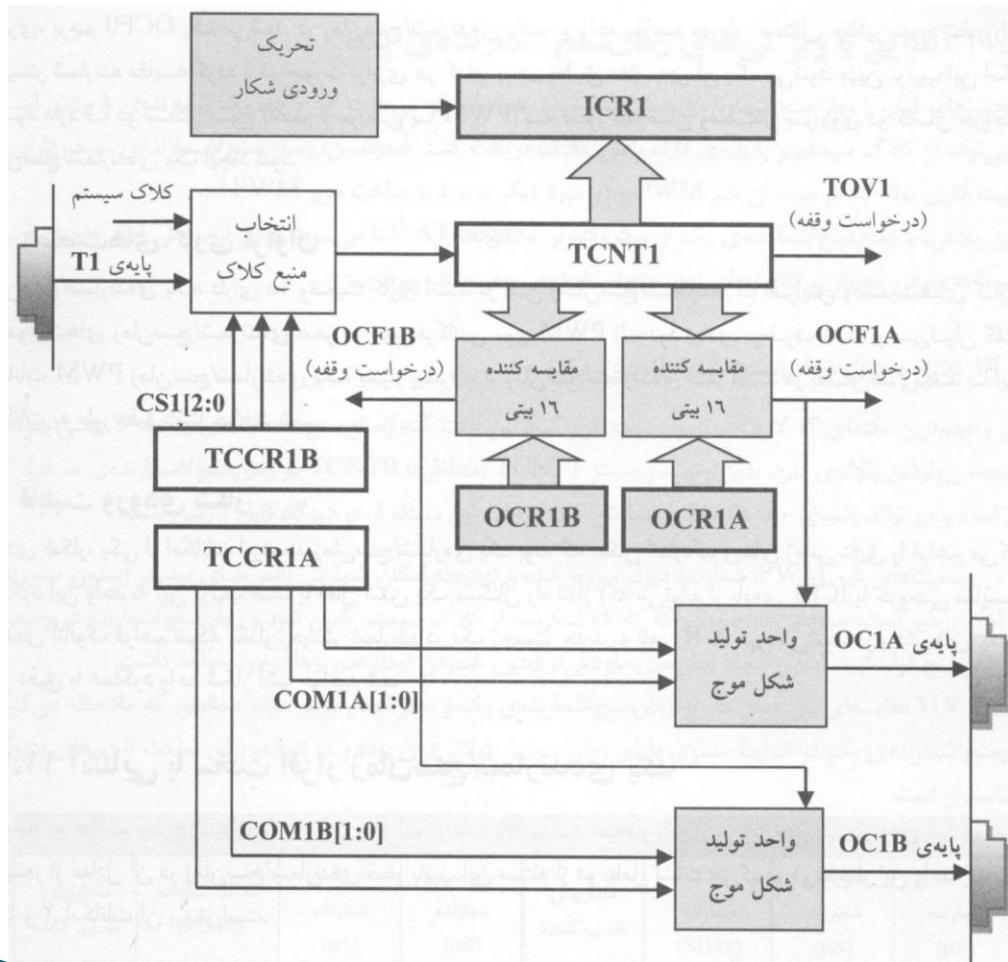
### وضعیت های کاری فراوان

تایمر/کانتر یک دارای ۱۵ وضعیت کاری است. می توان گفت امکانات PWM این تایمر/کانتر بسیار پیشرفته تر از تایمر/کانتر صفر است.

### قابلیت ورودی شکار (Input Capture)

یکی از امکانات ارزشمند بوده که امکان اندازه گیری های زمانی دقیق را فراهم میکند.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر



## آشنایی با سخت افزار تایمر/کانتر یک

ثبت **TCNT1** با دریافت پالس کلاک، رشته صفر تا ۶۵۵۳۵ را شمارش کرده و پس از سرریز شدن، فلگ **TOV1** یک میشود.

تایمر/کانتر یک نیز میتواند در دو وضعیت کانتر و تایмер عمل کند. در وضعیت اول منبع کلاک، پالسهایی خواهد بود که از دنیای بیرون تامین میشود و در وضعیت دوم پالس کلاک سیستم منبع شمارش ثبات **TCNT1** است.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## انتخاب منبع کلاک

انتخاب منبع کلاک مطابق با جدول زیر از طریق بیت‌های CS1[1:0] انجام می‌شود.

Bit	7	6	5	4	3	2	1	0	TCCR1B
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
منبع کلاک									
	0	0	0						بدون کلاک (متوقف)
	0	0	1						کلاک سیستم (بدون تقسیم)
	0	1	0						کلاک سیستم / ۸
	0	1	1						کلاک سیستم / ۶۴
	1	0	0						کلاک سیستم / ۲۵۶
	1	0	1						کلاک سیستم / ۱۰۲۴
	1	1	0						لبه‌ی پایین رونده‌ی پالس خارجی (T1)
	1	1	1						لبه‌ی بالا رونده‌ی پالس خارجی (T1)

# آشنایی با میکروکنترولر AVR بخش تایمر/کانتر

## مقایسه کننده های تایمر/کانتر یک

تایمر/کانتر دارای دو ثبات مقایسه، دو مقایسه کننده دیجیتال، دو فلگ تطبیق مقایسه و دو پایه خروجی مقایسه است. محتوای ثبات ۱۶ بیتی TCNT1 به طور پیوسته و مستقل با دو ثبات OCR1A و OCR1B مقایسه میشود، در صورتی که تطبیق مقایسه در واحد A روی دهد فلگ OCF1A و در صورتی که در واحد B تطبیق مقایسه ایجاد شده باشد، فلگ OCF1B یک خواهد شد.

Bit	7	6	5	4	3	2	1	0	TIFR
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Initial Value	0	0	0	0	0	0	0	0	
---------------	---	---	---	---	---	---	---	---	--

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## فعال کننده های مقایسه کننده تایمر/کانتر یک

در صورتی که بیت فعال ساز وقفه آن فلگ و فعال ساز عمومی وقفه ها یک باشند، یک شدن هر کدام از فلگهای OCF1A و OCF1B میتواند باعث درخواست وقفه شود. این دو بیت فعال ساز به نام های OCIE1A و OCIE1B از ثبات TIMSK میباشند. همچنین بیت ۲ از این ثبات به نام TOIE1 فعال ساز وقفه سرریز کانتر یک است.

Bit	7	6	5	4	3	2	1	0	TIMSK
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## كلمات کلیدی

قبل از پاسخ به اين سوال سه واژه کلید که در ادامه مكرراً از آنها استفاده خواهيم کرد را تعریف ميکنیم:

- BOTTOM: شمارنده زمانی که 0x0000 شود به رسیده است.
- MAX: زمانی که شمارنده 0xFFFF (دسيمال ٦٥٥٣٥) شود به خود رسیده است.
- TOP: زمانی که شمارنده برابر با بالاترين عدد در رشته شمارش خود ميشود به رسیده است.

نکته مهمی که در مورد TOP وجود دارد اين است که در تایmer/کانتر صفر، تغيير دادن TOP به عددی کمتر از MAX تنها در وضعیت CTC ممکن بود اما در تایmer/کانتر يك در وضعیت های PWM نيز میتوان مقدار TOP را به عددی کمتر از MAX تغيير داد.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## وضعیت های کاری تایمر/کانتر یک

تایمر/کانتر یک دارای ۱۵ وضعیت کاری است که یک وضعیت **NORMAL**، دو وضعیت **CTC** و ۱۲ وضعیت نیز مربوط به **PWM** میباشند. این وضعیت ها را در جدول مشاهده میکنید.

شماره	WGM13	WGM12	WGM11	WGM10	وضعیت کاری	TOP	بروزرسانی OCR1x	لحظه‌ای یک شدن TOV1 پرچم
0	0	0	0	0	Normal	0xFFFF	فوری	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	فوری	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	فوری	MAX
13	1	1	0	1	رززو شده	—	—	—
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## وضعیت های کاری تایمر/کانتر یک

انتخاب هر یک از این وضعیتها از طریق بیت‌های WGM1[3:0] از ثباتهای TCCR1A و TCCR1B انجام می‌شود.

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

همانطور که ملاحظه می‌کنید در جدول تنها یک وضعیت NORMAL وجود دارد که در این حالت TOP کانتر عدد 0xFFFF یا همان MAX است.

دو وضعیت CTC (۴ و ۱۲) در TOP با یکدیگر اختلاف دارند، بدین معنا که در وضعیت ۴، کانتر با رسیدن به عددی که در ثبات OCR1A است، پاک شده و در وضعیت ۱۲، حداقل مقدار ثبات TCNT1، برابر با مقدار ICR1 است (بدیهی است که حالات شکار در اینجا غیرفعال است).

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با نحوه دسترسی به ثباتهای ۱۶ بیتی

ثباتهای ICR1، OCR1A، OCR1B و TCNT1 ۱۶ بیتی هستند که میکروکنترلر از طریق باس داده ۸ بیتی به آنها دسترسی پیدا میکند، بنابراین نوشتن یا خواندن یکی از آنها باید طی دو سیکل انجام شود. مشکلی که به این ترتیب به وجود میآید این است که مقدار دهی متوالی به دو بایت این ثباتها، در برخی موارد ممکن است باعث ایجاد مقادیر میانی ناخواسته ای شود.

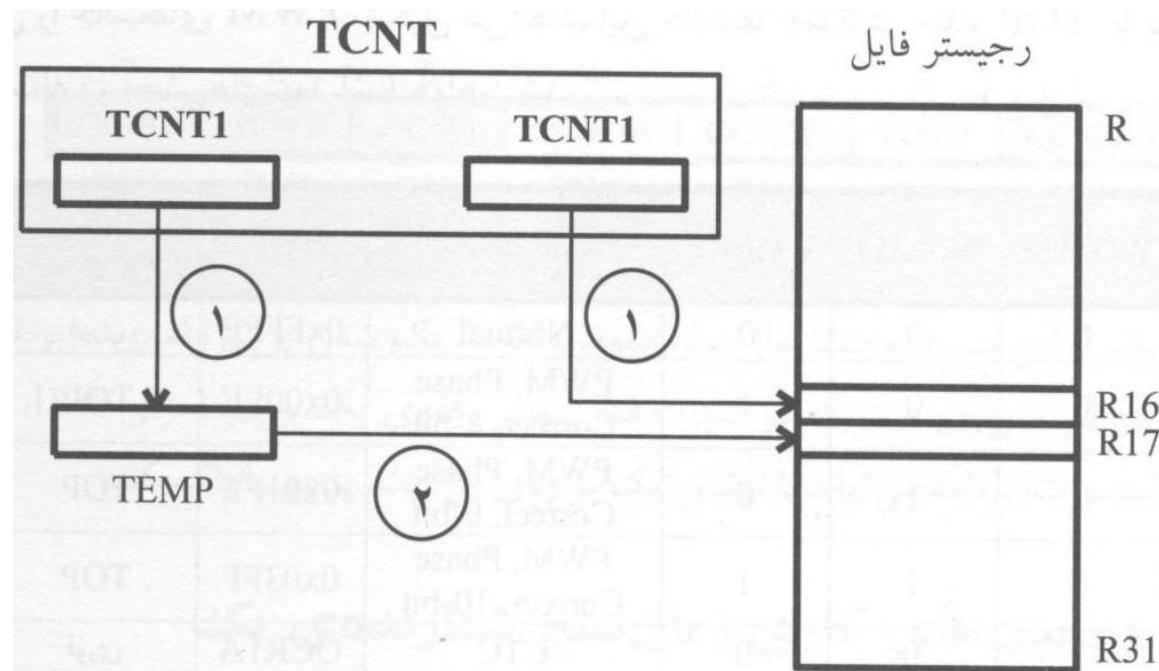
برای حل این مشکل از یک ثبات موقتی به نام TEMP استفاده میشود. این ثبات آدرس در فضای ثباتهای I/O نداشته و تنها برای عملیات داخلی از آن استفاده میشود. این ثبات بین تمام ثباتهای ۱۶ بیتی مشترک است و موقتا بایت بالاتر داده ۱۶ بیتی را نگهداری میکند.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با نحوه دسترسی به ثباتهای ۱۶ بیتی

برای روشن شدن نحوه عملکرد ثبات TEMP شکل زیر را در نظر بگیرید. این شکل به طور نمونه نحوه خواندن از ثبات TCNT1 را نشان میدهد. اعداد داخل دایره نمایشگر شماره سیکل میباشند.

بنابراین برای استفاده از این قابلیت لازم است تا برای خواندن یک ثبات ۱۶ بیتی ابتدا بایت با ارزش کمتر و سپس بایت با ارزش بالاتر خوانده شود.

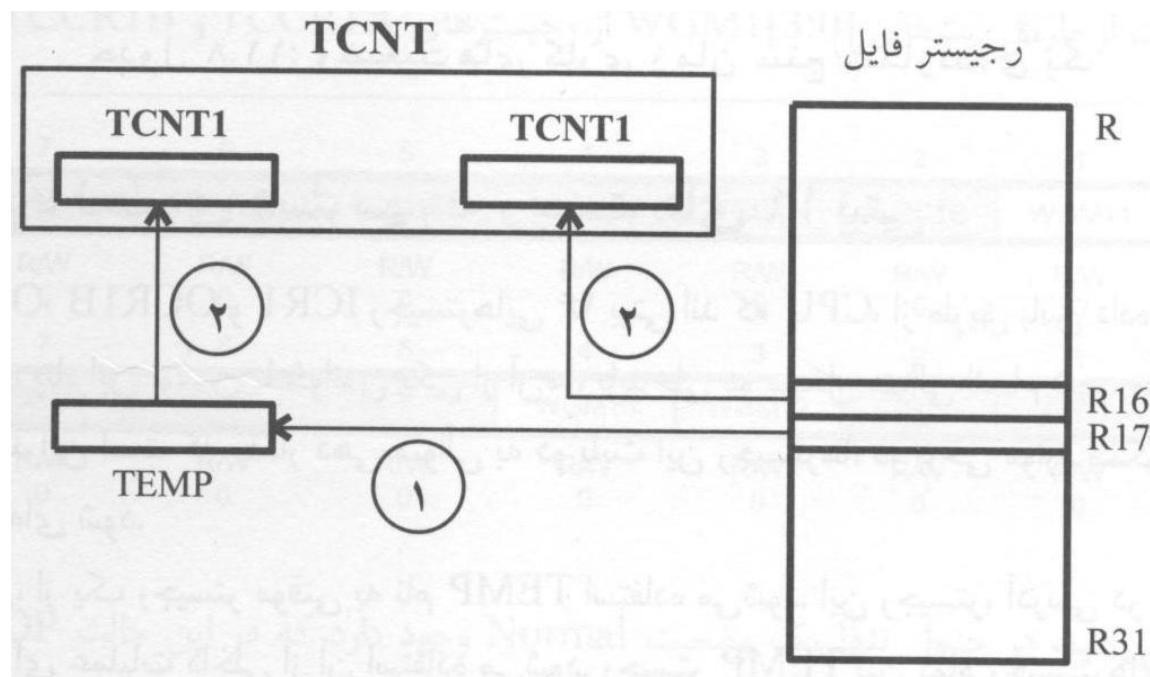


# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با نحوه دسترسی به ثباتهای ۱۶ بیتی

مطابق با شکل زیر، نوشتمن یک بایت در بایت بالاتر موجب ذخیره شدن آن در ثبات TEMP میشود.  
حال نوشتمن بایت پایین تر موجب میشود تا دو بایت بالا و پایین به طور همزمان به ثبات ۱۶ بیتی منتقل شود.

- بنابراین لازم است تا در بروز رسانی یک ثبات ۱۶ بیتی ابتدا بایت با ارزش بیشتر نوشتنه شود.



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با نحوه دسترسی به ثباتهای ۱۶ بیتی

نکته مهمی که در مورد TEMP لازم است مد نظر قرار گیرد این است که این ثبات، بین تمام ثباتهای ۱۶ بیتی مشترک است، بنابراین بین دو سیکل بروز رسانی یک ثبات ۱۶ بیتی، ممکن است وقفه ای درخواست شود و در ISR آن وقفه، مقدار TEMP تغییر کند. این مسئله باعث خطا در نوشتن یا خواندن ثبات میشود. برای جلوگیری از آن میتوان قبل از سیکل اول ، فعال ساز عمومی وقفه‌ها را غیر فعال و پس از سیکل دوم آن را فعال نمود:

```
#asm("cli")
// Writing 16 bit Register
#asm("sei")
```

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با نحوه دسترسی به ثباتهای ۱۶ بیتی

در محیط CodeVisionAVR مانند بیشتر کامپایلرهای سطح بالا، عملیات دسترسی به ثباتهای ۱۶ بیتی توسط کامپایلر انجام میشود. به عنوان مثال با نوشتن دستور **TCNT1=0x4455**، کامپایلر ابتدا بایت با ارزش بیشتر و سپس بایت با ارزش کمتر را در محل ثباتهای **TCNT1H** و **TCNT1L** مینویسید. تنها استثنای این قضیه، ثبات **ICR1** است که در دسترسی به آن، کامپایلر کمکی نکرده و باید بایت بالاتر و پایین تر آن به ترتیب ذکر شده، نوشته یا خوانده شوند. به عنوان مثال برای نوشتن عدد **0x4455** در ثبات **ICR1** نمیتوان از دستور **ICR1=0x4455** استفاده کرد و باید بایت بالاتر و پایین تر را به این ترتیب مقداردهی نمود:

**ICR1H=0X44 ;**

**ICR1L=0X55 ;**

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## شناصایی وضعیت NORMAL در تایمر/کانتر یک

برای انتخاب این وضعیت باید مقدار 0b0000 را به بیتهای WGM[3:0] منتب کرد. وضعیت نرمال در تایمر/کانتر یک مشابه نرمال در تایمر/کانتر صفر است با این تفاوت که رشته شمارش ثبات TCNT1 وسیعتر از TCNT0 میباشد. بدین ترتیب که شمارنده از عدد صفر تا ۶۵۵۳۵ شمرده و در لحظه صفر شدن، فلگ TOV1 یک

وضعیت پین	OC1A/OC1B	COM1A0/COM1B0	COM1A1/COM1B1
غیر فعال (I/O معمولی)		0	0
در تطبیق مقایسه Toggle		1	0
در تطبیق مقایسه Clear		0	1
در تطبیق مقایسه Set		1	1

در وضعیت نرمال میتوان از دو واحد مقایسه A و B برای ایجاد وقفه در فوائل مشخص استفاده نمود. در صورت نیاز میتوان با تنظیم بیتهای [0:1] COM1A[1:0] و [1:0] COM1B[1:0] مطابق با جدول بالا به صورت سخت افزاری بر روی پایه های OC1A و OC1B شکل موج تولید کرد.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## شناسایی اصول استفاده از تایمر/کانتر یک در وضعیت NORMAL

```
#include <mega32.h>
#include <lcd.h>
#include <stdlib.h>
//Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x12 ;PORTD
#endifasm

void ini_io();
unsigned char cnt;
unsigned char str_cnt[3];

Interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
    TCNT1=34286; //Set for 31250 count
    if(cnt<99)
        cnt++;
    else
        cnt=0;
    itoa(cnt,str_cnt); //convert integer to string
    lcd_clear();
    lcd_puts(str_cnt);
}
```

مزیت تایمر/کانتر یک در مقایسه با صفر این است که شمارنده میتواند رشته طولانی‌تری را بدون سرریز شمارش کند و بدین ترتیب امکان ایجاد زمانهای طولانی به صورت سخت افزاری وجود خواهد داشت. اینک میخواهیم یک حلقه تایмер یک ثانیه طراحی نماییم.

از آنجا که با پیش تقسیم کننده ۲۵۶، هر تیک شمارنده  $\frac{۳۲}{۳۲} = ۱$  میکروثانیه طول میکشد، بدیهی است که برای سپری شدن یک ثانیه یک میلیون میکروثانیه لازم است شمارنده  $\frac{۳۱۲۵۰}{۳۲} = ۹۷۵۶$  با افزایش یابد ( $۹۷۵۶ \times ۳۲ = ۳۱۲۸۶$ ). حال کافی است تا ثبات  $TCNT1$  در هر بار سرریز، با  $۳۴۲۸۶ - ۳۱۲۵۰ = ۳۴۲۳۶$  بارگذاری شود.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## شناسایی اصول استفاده از تایمر/کانتر یک در وضعیت NORMAL

```
void main(void)
{
    init_io(0);
    while(1)
    {
    };
}

Void init_io()
{
    // LCD module initialization
    lcd_init(16);
    lcd_putchar('0');

    // Timer/Counter 1 initialization
    TCNT1=34286;      //Set for 31250 count
    TCCR1A=0X00;
    TCCR1B=0X04;      //Prescaler=256
    TIMSK=0X04;        //Enable TOV1 interrupt
    #asm("sei");
}
```

مزیت تایمر/کانتر یک در مقایسه با صفر این است که شمارنده میتواند رشته طولانی‌تری را بدون سرریز شمارش کند و بدین ترتیب امکان ایجاد زمانهای طولانی به صورت سخت افزاری وجود خواهد داشت. اینک میخواهیم یک حلقه تایمر یک ثانیه طراحی نماییم.

از آنجا که با پیش تقسیم کننده ۳۲، هر تیک شمارنده ۳۲ میکروثانیه طول میکشد، بدیهی است که برای سپری شدن یک ثانیه یک میلیون میکروثانیه لازم است شمارنده ۳۱۲۵۰ با افزایش یابد ( $31250 = 32 \times 1000000 / 32$ ). حال کافی است تا ثبات TCNT1 در هر بار سرریز، با ۳۴۲۸۶ بارگذاری شود ( $34286 = 31250 + 3125$ ).  
۶۵۵۳۶

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

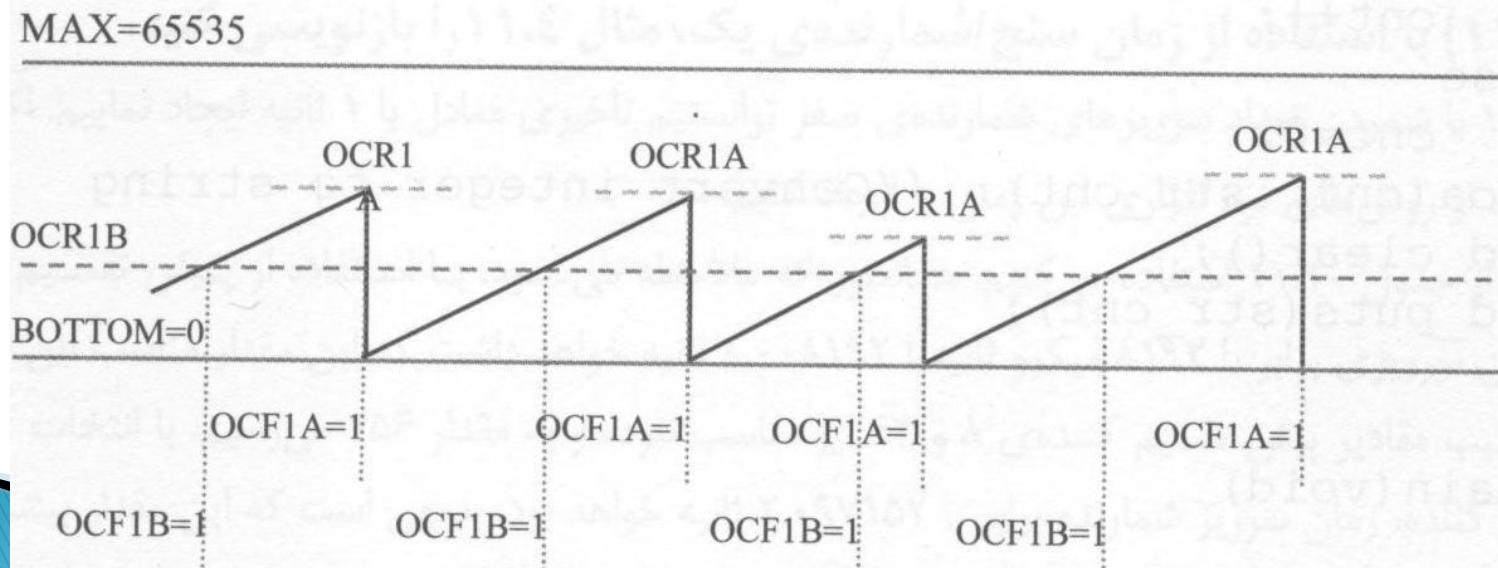
## آشنایی با وضعيت های CTC و تفاوت آنها

عملکرد CTC در تایمر/کانتر یک مشابه صفر است با این تفاوت که در اینجا TOP شمارنده توسط ثبات OCT1A یا ICR1 تعیین میشود.

در وضعیت ۴ از آنجایی که TOP شمارنده برابر مقدار OCR1A است، مطابق با شکل زیر با رسیدن شمارنده به مقدار A، ثبات TCNT1 پاک شده و همزمان در سیکل بعدی فلگ OCF1A نیز یک حواهد شد.

در عین حال، مطابق شکل به طور همزمان محتوى ثبات OCR1B نیز با TCNT1 مقایسه شده و در صورت برابری، فلگ OCF1B یک میشود. با یک شدن هر یک از دو فلگ OCF1A و OCF1B در صورتی که فعال ساز وقفه آن فلگ (در ثبات TIMSK) و فعال ساز عمومی وقفه ها یک باشند، میتواند وقفه درخواست شود.

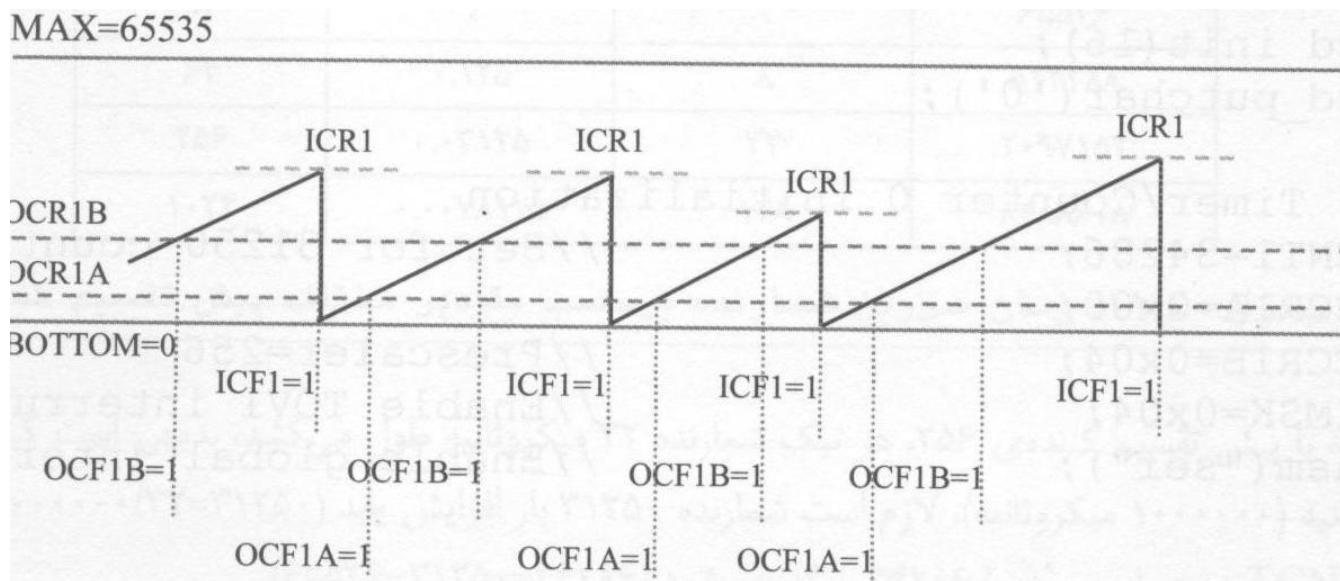
MAX=65535



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با وضعيت های CTC و تفاوت آنها

در وضعیت ۱۲ شمارنده برابر مقدار ICR1 است، بدین ترتیب مطابق با شکل زیر شمارنده از صفر شروع به شمارش کرده و با رسیدن به مقدار ICR1 مقدار آن صفر شده و همزمان فلگ ICF1 یک میشود. بدین ترتیب مقدار ثباتهای OCR1A و OCR1B نفی خواهد بود و تنها لحظه ای یک شدن فلگهای OCF1A و OCF1B را تعیین میکنند.



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## بازنویسی مثال قبلی

```
#include <mega32.h>
#include <lcd.h>
#include <stdlib.h>
//Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x12 ;PORTD
#endasm

void ini_io();
unsigned char cnt;
unsigned char str_cnt[3];

Interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
    if(cnt<99)
        cnt++;
    else
        cnt=0;
    itoa(cnt,str_cnt); //convert integer to string
    lcd_clear();
    lcd_puts(str_cnt);
}
```

در نمونه قبلی برای ایجاد زمان یک ثانیه، در روتین سرویس وقفه، مقدار اولیه ۳۴۲۸۶ را در ثبات TCNT1 بارگذاری کردیم. اکنون باستفاده از CTC میتوانیم TOP شمارنده را به ۳۱۲۴۹ تغییر دهیم تا شمارنده ۳۱۲۵۰ گام را طی کند. بدین منظور در وضعیت CTC تایмер را با TOP=OCR1A پیکربندی کرده ایم.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

بازنویسی مثال قبلی

```
void main(void)
{
    init_io(0);
    while(1)
    {
    };
}

Void init_io()
{
    // LCD module initialization
    lcd_init(16);
    lcd_putchar('0');

    // Timer/Counter 1 initialization
    OCR1A=31249;      //Set for 31250 count
    TCCR1A=0X00;      //CTC: TOP=OCR1A
    TCCR1B=0X0C;      //Prescaler=256
    TIMSK=0X10;        //Enable OCF1A interrupt
    #asm("sei");
}
```

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با رو تولید شکل موج در وضعیت های CTC

مشاهده کردید که یک شدن فلگ OCF1A یا OCF1B، فرمانی برای واحد تولید شکل موج صادر میکند. در این لحظه این واحد با توجه به مقادیر بیتهاي [COM1A[1:0] و COM1B[1:0]] وضعیت پایه OC1A یا OC1B را تغییر میدهد.

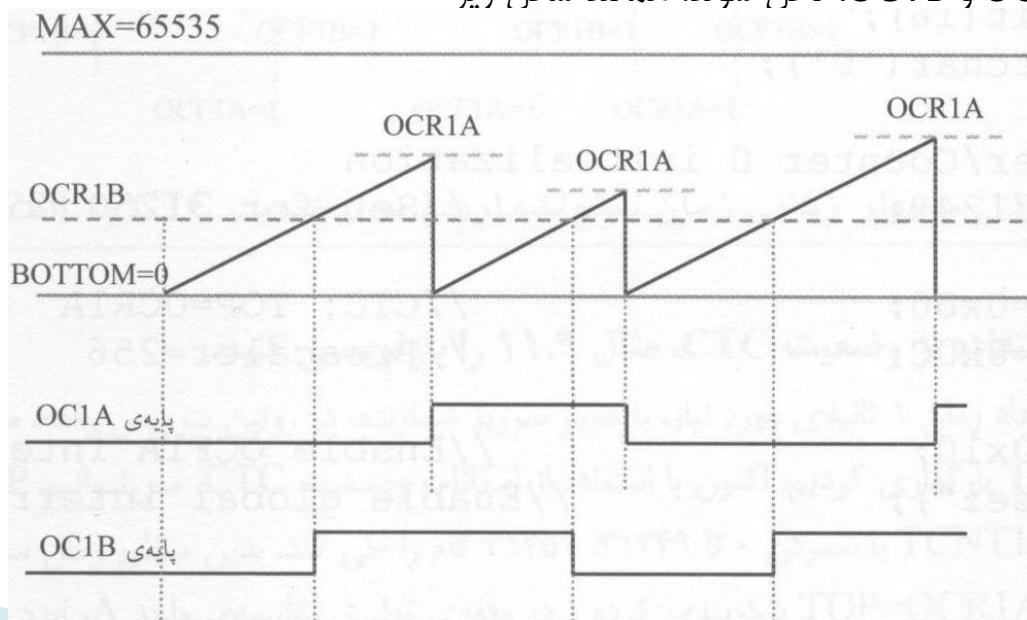
حالات مختلفی که بیتهاي [COM1A[1:0] و COM1B[1:0]] میتوانند اختیار کنند مطابق با جدول زیر است:

COM1A1/COM1B1	COM1A0/COM1B0	وضعیت پین
0	0	غیر فعال (I/O معمولی)
0	1	در تطبیق مقایسه Toggle
1	0	در تطبیق مقایسه Clear
1	1	در تطبیق مقایسه Set

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با رو تولید شکل موج در وضعیت های CTC

با توجه به مطالب بیان شده، میتوان نحوه ایجاد شکل موج در وضعیت های CTC را به طور مجزا بررسی نمود. در وضعیت ۴، TOP شمارنده برابر با مقدار OCR1A بوده و در نتیجه فعال نمودن واحد تولید شکل موج، باعث تغییر وضعیت پایه های OC1A و OC1B میشود. فرض کنید بیت های COM1A[1:0] در وضعیتی باشند که در لحظه تطبیق مقایسه، پایه های OC1A و OC1B، تاگل، شوند. همانند شکل زیر



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## فرکانس شکل موج ایجاد شده

فرکانس شکل موج ایجاد شده از رابطه زیر بدست می آید:

$$f_{OC1A/OC1B} = \frac{f_{CLK}}{2N(1 + OCR1A)}$$

با توجه به این رابطه، بالاترین فرکانس پایه OC1B زمانی خواهد بود که مقدار OC1A صفر بوده و پیش تقسیم کننده نیز برابر یک باشد. بدین ترتیب شکل موجی با نصف فرکانس کلاک سیستم ایجاد میشود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## مثال ایجاد شکل موج

با استفاده از وضعیت CTC، دو موج مربعی با فرکانس یک کیلوهرتز و اختلاف زمانی ۴۰۰ میکروثانیه بر روی پایه های OC1A و OC1B ایجاد نمایید (کلاک سیستم ۸ مگاهرتز است).

اگر مقدار پیش تقسیم کننده ۸ باشد، کلاک شمارنده

برابر یک مگاهرتز بوده و هر تیک آن یک میکروثانیه

خواهد بود. بدین ترتیب در صورتی که TOP شمارنده

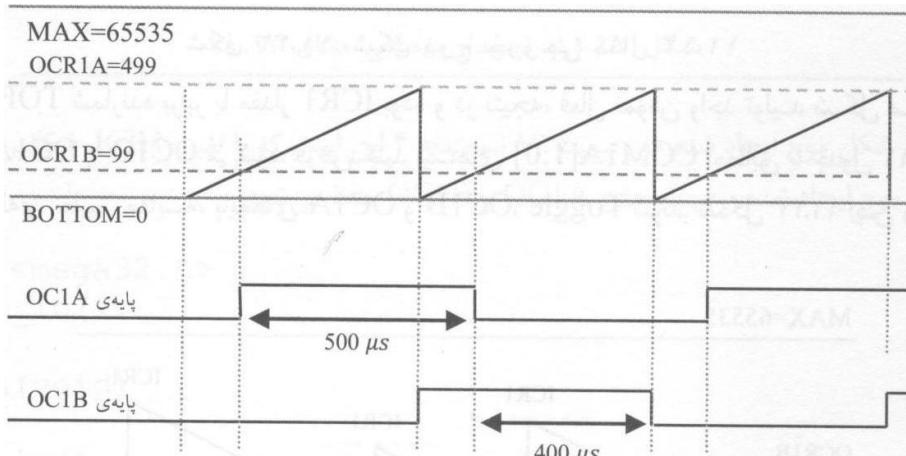
برابر ۴۹۹ باشد، دوره تناب شکل موجهای ایجاد

شده بر روی OC1A و OC1B یک میلی ثانیه و

فرکانس آنها یک کیلوهرتز است. همچنین مطابق با

شکل زیر، برای ایجاد اختلاف زمانی ۴۰۰ میکروثانیه

کافی است تا OCR1B با ۹۹ بارگذاری شود.

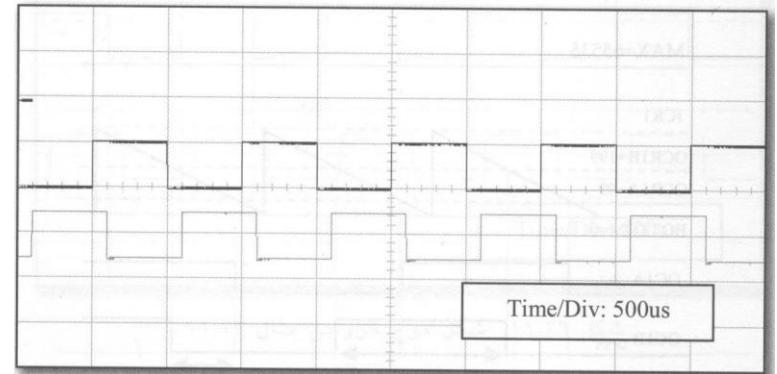


# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## مثال ایجاد شکل موج

```
#include <mega32.h>
void init_io();
void main(void)
{
    init_io();
    while(1)
    {
    };
}

Void init_io()
{
    DDRD=0x30;          //Output OC1A and OC1B
    OCR1A=499;          //Set for 1000 count
    OCR1B=99;           //TO 100us delay
    TCCR1A=0X50;        //CTC: TOP=OCR1A
    TCCR1B=0X0A;        //Prescaler=8
                        //OC1A output: Toggle
                        //OC1B output: Toggle
}
```

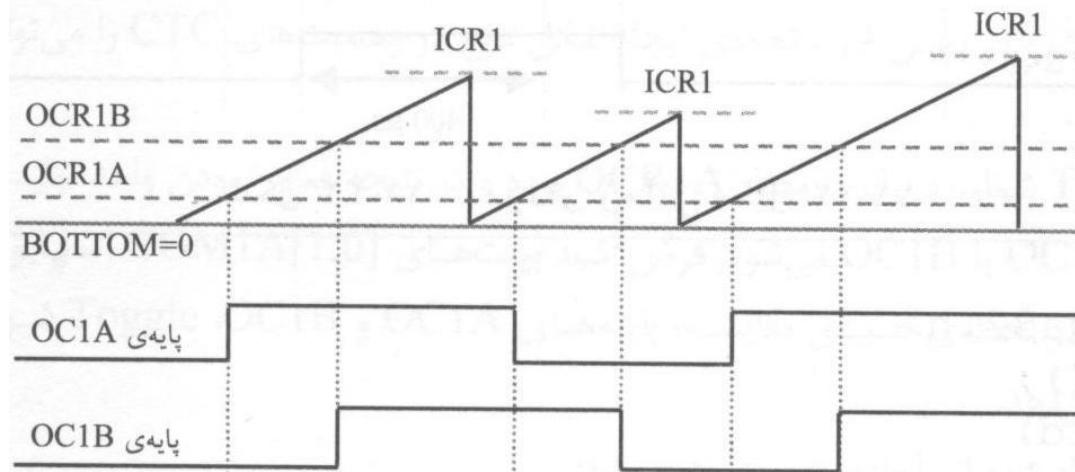


# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با رو تولید شکل موج در وضعیت های CTC

در وضعیت ۱۲، TOP شمارنده برابر با مقدار ICR1 بوده و در نتیجه فعال نمودن واحد تولید شکل موج، باعث تغییر وضعیت پایه OC1B یا OC1A میشود. فرض کنید بیتهای COM1A[1:0] در وضعیتی باشند که در لحظه تطبیق مقایسه، پایه های OC1A و OC1B، تاگل شوند. همانند شکل زیر

MAX=65535



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

فرکانس شکل موج ایجاد شده

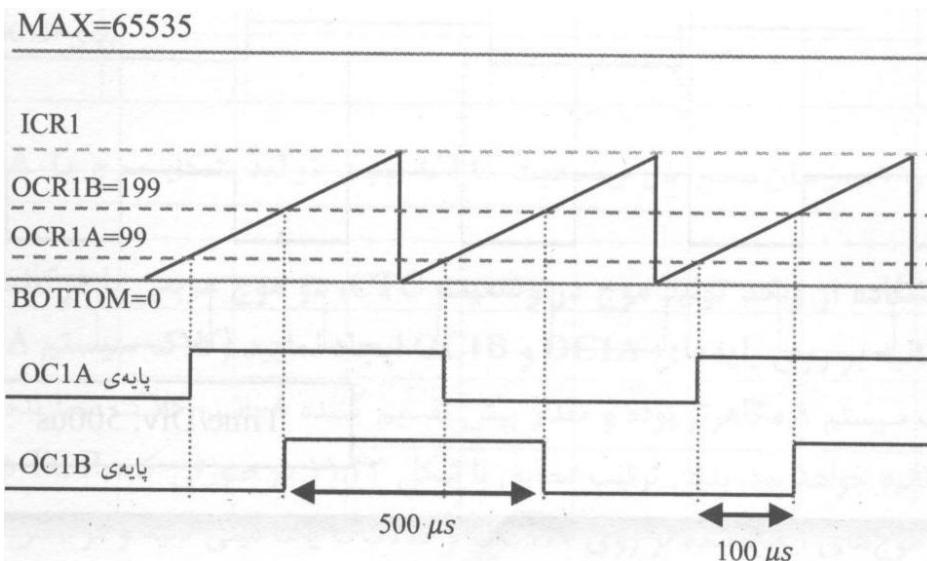
فرکانس شکل موج ایجاد شده از رابطه زیر بدست می آید:

$$f_{OC1A/OC1B} = \frac{f_{CLK}}{2N(1 + ICR1)}$$

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## مثال ایجاد شکل موج

با استفاده از وضعیت CTC، دو موج مربعی با فرکانس یک کیلوهرتز و اختلاف زمانی ۱۰۰ میکروثانیه بر روی پایه های OC1B و OC1A ایجاد نمایید (کلاک سیستم ۸ مگاهرتز است).



اگر مقدار پیش تقسیم کننده ۸ باشد، کلاک شمارنده برابر یک مگاهرتز بوده و هر تیک آن یک میکروثانیه خواهد بود. بدین ترتیب در صورتی که TOP شمارنده برابر ۴۹۹ باشد، دوره تناب شکل موجهای ایجاد شده بر روی OC1A و OC1B یک میلی ثانیه و فرکانس آنها یک کیلوهرتز است. همچنین مطابق با شکل زیر، برای ایجاد اختلاف زمانی ۱۰۰ میکروثانیه کافی است تا مقدار OCR1B و OCR1A ۱۰۰ واحد اختلاف داشته باشند..

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## مثال ایجاد شکل موج

```
#include <mega32.h>
void init_io();
void main(void)
{
    init_io();
    while(1)
    {
    };
}

Void init_io()
{
    DDRD=0x30;      //Output OC1A and OC1B
    OCR1A=99;
    OCR1B=199;
    TCCR1A=0X50;   //CTC: TOP=ICR1
    TCCR1B=0X1A;   //Prescaler=8
                    //OC1A output: Toggle
                    //OC1B output: Toggle
    ICR1H=0x01;    //Set for 500 count
    ICR1L=0xF3;

}
```

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با وضعیت های PWM در تایمر/کانتر یک

قابلیت های عمدۀ تایمر/کانتر یک، ناشی از وضعیت های PWM قابل انعطاف آن است. همانطور که میدانید یک سیگنال PWM دو مشخصه اساسی دارد: عرض پالس و فرکانس.

- در مورد عرض پالس، به لحاظ ۱۶ بیتی بودن تایمر/کانتر یک، موج PWM ایجاد شده رزولوشن نسبتا بالایی دارد.
- فرکانس سیگنال PWM تولید شده توسط تایمر/کانتر یک را میتوان به سادگی و به شکلی پیوسته تغییر داد

تایمر/کانتر یک دارای ۱۲ وضعیت PWM است. از این ۱۲ وضعیت، ۵ مورد مربوط به Fast PWM، ۵ مورد مربوط به Phase & Frequency PWM و دو مورد آن مربوط به یک وضعیت جدید به نام Phase Correct PWM است.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## شناسایی وضعیت های Fast PWM در تایمر/کانتر یک

در صورتی که پنج وضعیت مربوط به Fast PWM را از جدول کلی استخراج کنیم جدول زیر نتیجه میگردد.

شماره	WGM13	WGM12	WGM11	WGM10	وضعیت کاری	TOP	بروزرسانی OCR1X	لحظه‌ی یک شدن پرچم TOV1
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

با کمی دقت در این جدول میتوان دریافت که اختلاف این پنج وضعیت کاری، ناشی از مقدار TOP آنهاست که هر یک را مورد بررسی قرار میدهیم.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## Fast PWM, 8-bit

در این حالت TOP شمارنده برابر با 0x00FF خواهد بود و مشابه با تایمر/کانتر صفر عمل میکند. بدین ترتیب رشته شمارش ثبات TCNT1 محدود به اعداد صفر تا ۲۵۵ شده و در لحظه سرریز شدن شمارنده، فلگ TOV1 یک میشود. فرکانس سیگنال ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{N \times 256}$$

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## Fast PWM, 9-bit

در این حالت TOP شمارنده برابر با 0x01FF خواهد بود. بدین ترتیب رشته شمارش ثبات TCNT1 محدود به اعداد صفر تا ۵۱۱ شده و در لحظه سرریز شدن شمارنده، فلگ TOV1 یک میشود. فرکانس سیگنال ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{N \times 512}$$

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## Fast PWM, 10-bit

در این حالت TOP شمارنده برابر با  $0x03FF$  خواهد. بدین ترتیب رشته شمارش ثبات TCNT1 محدود به اعداد صفر تا ۱۰۲۳ شده و در لحظه سرریز شدن شمارنده، فلگ TOV1 یک میشود. فرکانس سیگنال ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{N \times 1024}$$

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## Fast PWM, TOP=ICR1

در این حالت TOP شمارنده برابر با مقدار ثبات ICR1 خواهد بود و رزولوشن سیگنال ایجاد شده از رابطه زیر بدست می‌آید:

$$resolution = \frac{\log(ICR1 + 1)}{\log(2)}$$

بدین ترتیب، رشته شمارش ثبات TCNT1 محدود به اعداد صفر تا ICR1 شده و در لحظه سرریز شدن شمارنده، فلگ TOV1 یک میشود. در این وضعیت فرکانس سیگنال ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{N \times (ICR1 + 1)}$$

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## Fast PWM, TOP=OCR1A

در این حالت TOP شمارنده برابر با مقدار ثبات OCR1A خواهد بود و رزولوشن سیگنال ایجاد شده از رابطه زیر بدست می‌آید:

$$resolution = \frac{\log(OCR1A + 1)}{\log(2)}$$

بدین ترتیب، رشته شمارش ثبات TCNT1 محدود به اعداد صفر تا OCR1A شده و در لحظه سرریز شدن

شمارنده، فلگ TOV1 یک میشود. در این وضعیت فرکانس سیگنال ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{N \times (OCR1A + 1)}$$

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## شناسایی وضعيت های Fast PWM در تایمر/کانتر یک

حال ممکن است با توجه به جامعیت دو وضعیت آخر، لزوم سه وضعیت نخست زیر سوال برود. به عبارت دیگر ممکن است این سوال مطرح شود:

«با توجه به اینک میتوان در دو وضعیت Fast PWM, TOP=OCR1A و Fast PWM, TOP=ICR1 میتوان شمارند را اعداد 0xFF و 0x1FF و 0x3FF تعیین کرد، چه لزومی دارد تا برای این سه مورد، وضعیتهای جداگانه وجود داشته باشد؟»

پاسخ این است که استفاده از وضعیت Fast PWM, TOP=ICR1 موجب غیرفعال شدن، قابلیت شکار میشود. همچنین استفاده از وضعیت Fast PWM, TOP=OCR1A موجب غیرفعال شدن خروجی PWM واحد A شده به بیان دیگر نمیتوان موج PWM بر روی پایه OCR1A تولید نمود.

همانطور که از روابط مشخص است وضعیت کاری Fast PWM هشت، نه و ده بیتی در رزولوشن و فرکانس با هم متفاوتند.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

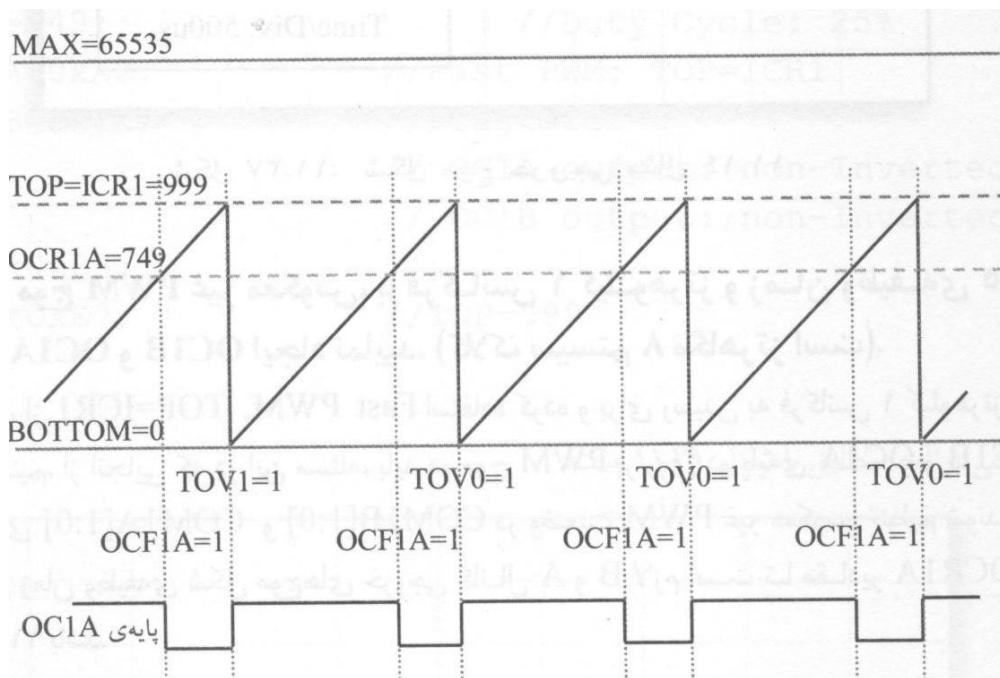
## آشنایی با روش تنظیم واحد تولید شکل موج

مشابه با وضعیت CTC، چنانچه بخواهیم واحد تولید موج را فعال نماییم تا به صورت خودکار شکل موجی را بر روی پایه های OC1A یا OC1B ایجاد نماید، لازم است تا بیتهاي [COM1A[1:0] و COM1B[1:0] تنظیم شوند. حالتهاي مختلفی را که این بیتها شامل میشوند در جدول زیر مشاهده مینمایید.

COM1A1/ COM1B1	COM1A0/ COM1B0	وضعیت پایه های [COM1B[1:0] یا [COM1A[1:0]
0	0	غیر فعال (I/O معمولی)
0	1	در صورتی که [WGM1[3:0] برابر با ۱۵ باشد: پایه OC1A در لحظه‌ی تطبیق مقایسه، OC1A می‌شود و پایه OC1B غیر فعال است. برای سایر وضعیت‌ها، پایه‌های OC1B، عملکرد تولید شکل موج تداشته و I/O معمولی‌اند.
1	0	پاک کردن OC1B یا OC1A در تطبیق مقایسه و یک کردن آن در TOP (PWM غیر معکوس)
1	1	یک کردن OC1A یا OC1B در تطبیق مقایسه و پاک کردن آن در TOP (PWM معکوس)

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: ایجاد شکل موج غیرمعکوس با فرکانس ۱KHz و زمان وظیفه ۷۵٪ بر روی پایه OC1A



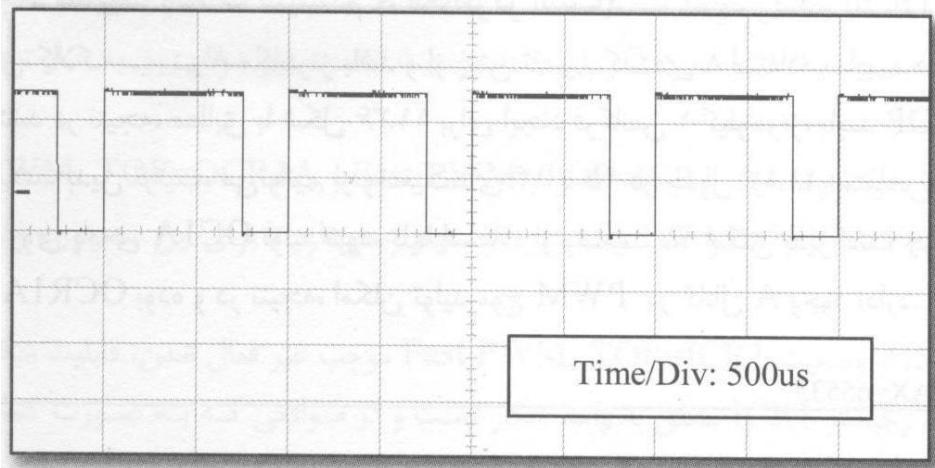
در صورتی که فرکانس کلک سیستم ۸ مگاهرتز باشد و از پیش تقسیم کننده ۸ استفاده نماییم، هر تیک شمارنده یک میکروثانیه طول میکشد. در نتیجه برای ایجاد فرکانس یک کیلوهرتز باید TOP شمارنده برابر با ۹۹۹ (یا ۰x03E7) باشد. بدین ترتیب میتوانیم از وضعیتهای ۱۴ یا ۱۵ استفاده نماییم. اما از آنجا که شکل موج باید بر روی پایه OC1A تولید شود، تنها استفاده از وضعیت ۱۵ امکان پذیر است چرا که در وضعیت ۱۴، TOP شمارنده برابر OCR1A بوده و در نتیجه تولید موج PWM در کanal A وجود ندارد.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: ایجاد شکل موج غیرمعکوس با فرکانس 1KHz و زمان وظیفه 75٪ بر روی پایه OC1A

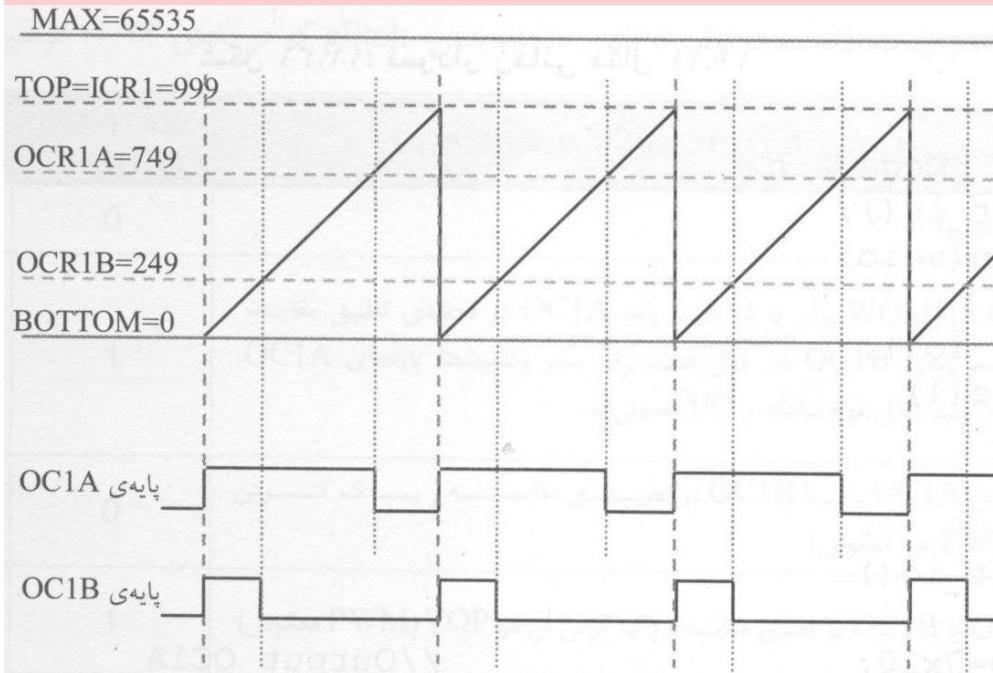
```
#include <mega32.h>
void init_io();
void main(void)
{
    init_io();
    while(1)
    {
    };
}

Void init_io()
{
    DDRD=0x20;          //Output OC1A
    // Timer/counter 1 initialization
    OCR1A=749;         //Duty Cycle: 75%
    TCCR1A=0X82;        //Fast PWM:CTC: TOP=ICR1
    TCCR1B=0X1A;        //Prescaler=8
                    //OC1A output: Non-Inverted
    ICR1H=0x03;
    ICR1L=0xE7;        // TOP=999
}
```



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

مثال: ایجاد دو سیگنال PWM غیرمعکوس، با فرکانس ۱KHz و زمان وظیفه ۷۵٪ و ۲۵٪ بر روی پایه های OC1A و OC1B



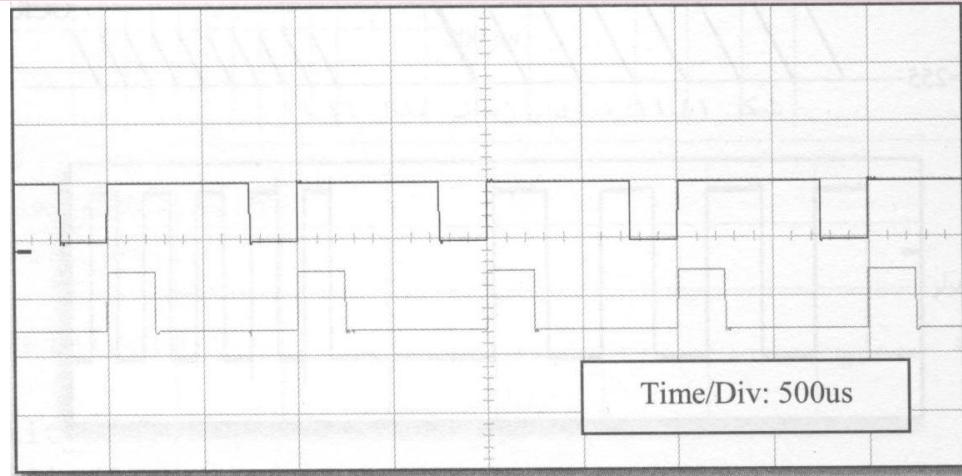
مشابه مثال قبلی، از وضعیت Fast PWM, Top=ICR1 استفاده کرده و برای رسیدن به فرکانس یک کیلوهرتز، ICR1 را با ۹۹۹ بارگذاری میکنیم. از آنجا که در این مثال باید دو سیگنال PWM بر روی دو پایه OC1A و OC1B تولید شود، لازم است تا بیتهاي COM1A[1:0] و COM1B[1:0] در وضعیت PWM غیرمعکوس تنظیم گردند.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

مثال: ایجاد دو سیگنال PWM غیرمعکوس، با فرکانس ۱KHz و زمان وظیفه ۷۵٪ و ۲۵٪ بر روی پایه های OC1A و OC1B

```
#include <mega32.h>
void init_io();
void main(void)
{
    init_io();
    while(1)
    {
    };
}

Void init_io()
{
    DDRD=0x30;          //Output OC1A and OC1B
    // Timer/counter 1 initialization
    OCR1A=749;          //Duty Cycle: 75%
    OCR1B=249;          // Duty Cycle: 25%
    TCCR1A=0X82;        //Fast PWM:CTC: TOP=ICR1
    TCCR1B=0X1A;        //Prescaler=8
                    //OC1A output: Non-Inverted
                    //OC1B output: Non-Inverted
    ICR1H=0x03;
    ICR1L=0xE7;        // TOP=999
}
```

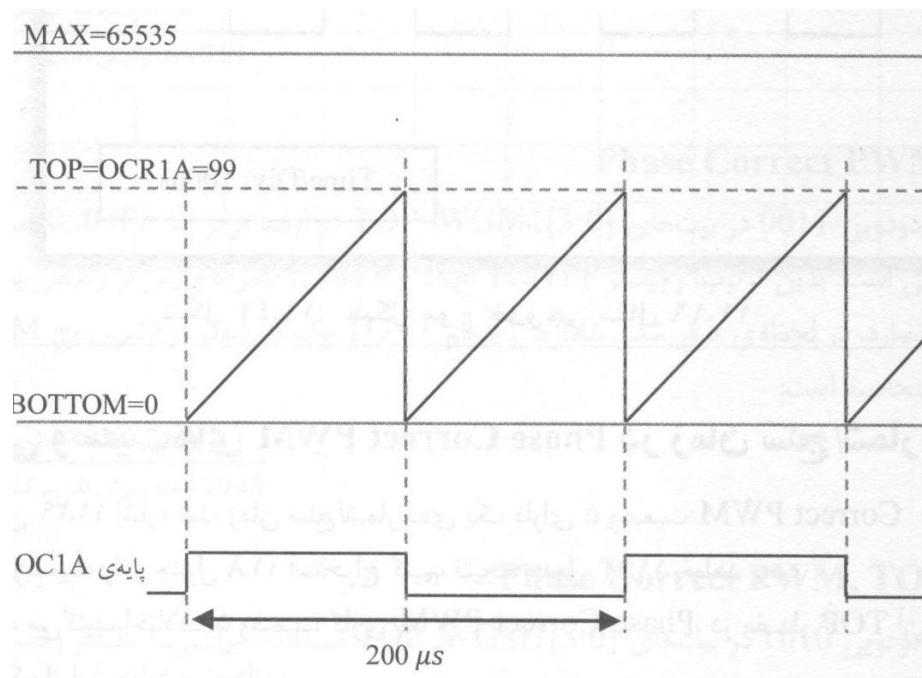


# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## مثال: ایجاد یک موج مربعی با فرکانس ۵ کیلوهرتز با شمارنده یک در وضعیت Fast PWM

برای ایجاد این شکل موج، مطابق با دستورالعمل ارائه شده، شمارنده را در وضعیت ۱۵ تنظیم کرده ( $WGM1[3:0]=15$ ) و خروجی واحد مقایسه را به صورت Toggle تنظیم میکنیم. حال کافی است مقدار OCR1A برابر

با ۹۹ باشد.

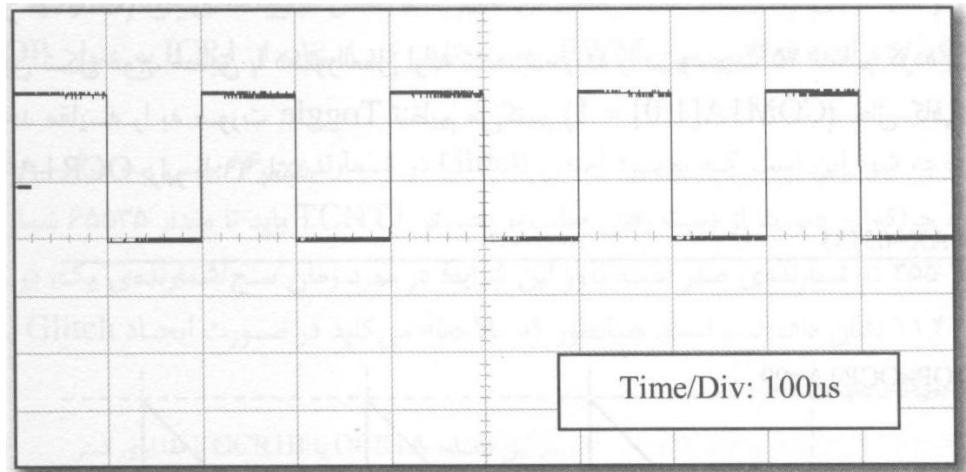


# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: ایجاد یک موج مربعی با فرکانس ۵ کیلوهرتز با شمارنده یک در وضعیت Fast PWM

```
#include <mega32.h>
void init_io();
void main(void)
{
    init_io();
    while(1)
    {
    };
}

Void init_io()
{
    DDRD=0x20;          //Output OC1A
    // Timer/counter 1 initialization
    OCR1A=99;
    TCCR1A=0X43;      //Fast PWM: TOP=OCR1A
    TCCR1B=0X1A;      //Prescaler=8
                      //OC1A output: Non-Inverted
}
```



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## شناسایی وضعیتهای Phase Correct PWM در تایمر/کانتر یک

همانطور که قبلاً اشاره شد، تایمر/کانتر یک دارای ۵ وضعیت Phase Correct PWM است. در جدول زیر این پنج وضعیت را مشاهده میکنید.

شماره	WGM1 3	WGM1 2	WGM1 1	WGM 10	وضعیت کاری	TOP	بروزرسانی OCR1x	لحظه‌ی یک شدن TOV1 پرچم
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM

همانطور که ملاحظه میکند، اختلاف ۵ وضعیت کاری، در مقدار TOP آنهاست که در ادامه به بررسی هر یک از این موارد میپردازیم.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## Phase Correct PWM, 8-bit

در این حالت TOP شمارنده برابر با 0x00FF خواهد بود و مشابه با تایمر/کانتر صفر عمل میکند. بدین ترتیب ثبات TCNT1 اعداد صفر تا ۲۵۵ را شمرده و پس از رسیدن به TOP به صورت نزولی تا صفر میشمارد. در لحظه صفر شدن شمارنده، شده و فلگ TOV1 یک میشود. فرکانس سیگنال ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{2 \times N \times 255}$$

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## Phase Correct PWM, 9-bit

در این حالت TOP شمارنده برابر با 0x01FF خواهد بود و مشابه با تایمر/کانتر صفر عمل میکند. بدین ترتیب ثبات TCNT1 اعداد صفر تا ۵۱۱ را شمرده و پس از رسیدن به TOP به صورت نزولی تا صفر میشمارد. در لحظه صفر شدن شمارنده، شده و فلگ TOV1 یک میشود. فرکانس سیگنال ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{2 \times N \times 511}$$

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## Phase Correct PWM, 10-bit

در این حالت TOP شمارنده برابر با  $0x01FF$  خواهد بود و مشابه با تایمر/کانتر صفر عمل میکند. بدین ترتیب ثبات TCNT1 اعداد صفر تا  $10^{23}$  را شمرده و پس از رسیدن به TOP به صورت نزولی تا صفر میشمارد. در لحظه صفر شدن شمارنده، شده و فلگ TOV1 یک میشود. فرکانس سیگنال ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{2 \times N \times 1023}$$

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## Phase Correct PWM, TOP=ICR1

در این حالت TOP شمارنده برابر با مقدار ثبات ICR1 خواهد بود و رزولوشن سیگنال ایجاد شده از رابطه زیر بدست می‌آید:

$$resolution = \frac{\log(ICR1 + 1)}{\log(2)}$$

بدین ترتیب، رشته شمارش ثبات TCNT1 اعداد صفر تا ICR1 را شمرده و پس از رسیدن به مقدار ICR1، به صورت نزولی تا صفر می‌شمارد. در لحظه صفر شدن شمارنده، فلگ TOV1 یک می‌شود. فرکانس سیگنال ایجاد شده از

رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{2 \times N \times ICR1}$$

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## Phase Correct PWM, TOP=OCR1A

در این حالت TOP شمارنده برابر با مقدار ثبات OCR1A شده و رزولوشن سیگنال ایجاد شده از رابطه زیر بدست می‌آید:

$$resolution = \frac{\log(OCR1A + 1)}{\log(2)}$$

بدین ترتیب، رشته شمارش ثبات TCNT1 اعداد صفر تا OCR1A را شمرده و پس از رسیدن به مقدار OCR1A، به صورت نزولی تا صفر می‌شمارد. در لحظه صفر شدن شمارنده، فلگ TOV1 یک می‌شود. فرکانس سیگنال ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{2 \times N \times OCR1A}$$

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با روش تنظیم واحد تولید شکل موج

مشابه با وضعیت Fast PWM، چنانچه بخواهیم واحد تولید موج را فعال نماییم تا به صورت خودکار شکل موجی را بر روی پایه های OC1B یا OC1A ایجاد نماید، لازم است تا بیتهاي [0:1] COM1B و [1:0] COM1A تنظیم شوند. حالتهاي مختلفی را که اين بیتها شامل میشوند در جدول زير مشاهده مينمایيد.

COM1A1/ COM1B1	COM1A0/ COM1B0	وضعیت پایه های [0:1] COM1B یا [1:0] COM1A
0	0	غیر فعال (I/O معمولی)
0	1	در صورتی که WGM1[3:0] برابر با ۱۱ باشد: پایه OC1A در لحظه‌ی تطبیق مقایسه، OC1B می‌شود و پایه‌ی OC1B غیر فعال است. برای سایر وضعیت‌ها، پایه‌های OC1A و OC1B، عملکرد تولید شکل موج نداشته و I/O معمولی اند. (حالت خاص)
1	0	پاک کردن OC1A یا OC1B در تطبیق مقایسه در شمارش صعودی و یک کردن آن در تطبیق مقایسه در شمارش نزولی (PWM غیر معکوس)
1	1	یک کردن OC1A یا OC1B در تطبیق مقایسه در شمارش صعودی و پاک کردن آن در تطبیق مقایسه در شمارش نزولی (PWM معکوس)

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

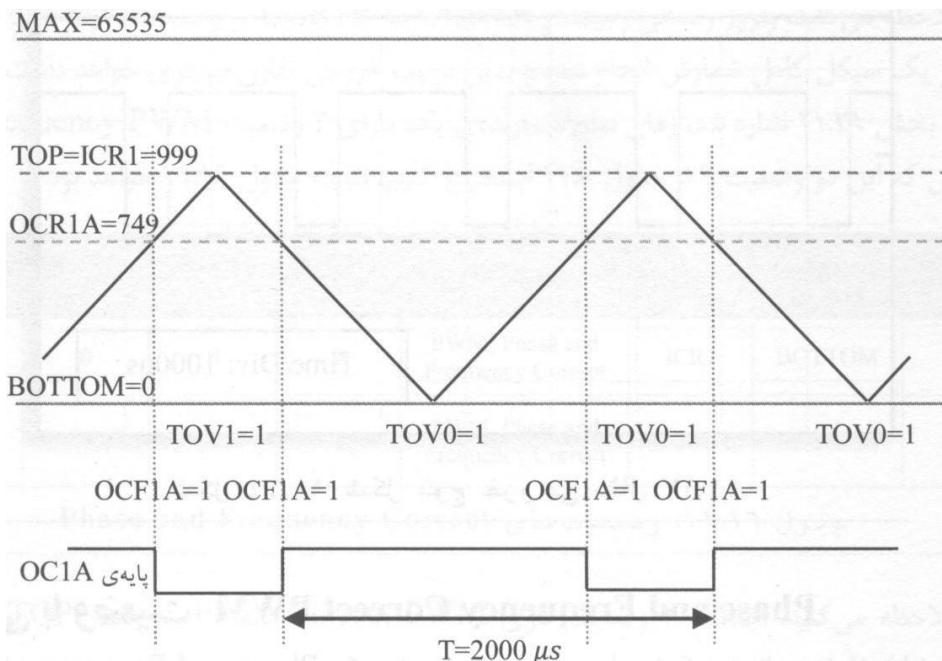
## آشنایی با روش تولید شکل موج در وضیت Phase Correct PWM

برای تولید شکل موج، در اولین گام لازم است تا وضعیت مناسب برای سیگنال PWM مورد نظر را انتخاب نمود. پس از آن میتوان با تنظیم بیتهاي [COM1A[1:0] و [COM1B[1:0] نوع شکل سیگنال خروجی (معکوس، غیرمعکوس و یا حالت خاص را انتخاب نمود.

در اسلاید بعد مثالی را بررسی میکنیم.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: ایجاد سیگنال PWM غیرمعکوس با فرکانس ۵۰۰ هرتز و زمان وظیفه ۷۵٪ بروی پایه OC1A



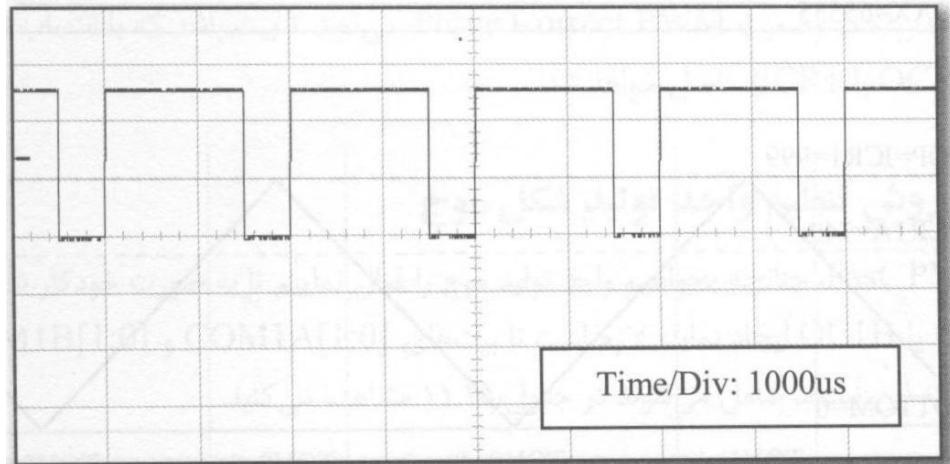
در صورتی که فرکانس کلاک سیستم ۸ مگاهرتز باشد و از پیش تقسیم کننده ۸ استفاده نماییم، هر تیک شمارنده یک میکروثانیه طول میکشد. در نتیجه برای ایجاد فرکانس ۵۰۰ هرتز باید TOP شمارنده برابر با ۹۹۹ (یا 0x03E7) باشد. بدین ترتیب میتوانیم از وضعیتهای ۱۰ یا ۱۱ استفاده نماییم. اما از آنجا که شکل موج باید بر روی پایه OC1A تولید شود، تنها استفاده از وضعیت ۱۰ امکان پذیر است چرا که در وضعیت ۱۱، TOP شمارنده برابر OCR1A بوده و در نتیجه تولید موج PWM در کanal A وجود ندارد.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: ایجاد سیگنال PWM غیرمعکوس با فرکانس ۵۰۰ هرتز و زمان وظیفه ۷۵٪ بروی پایه OC1A

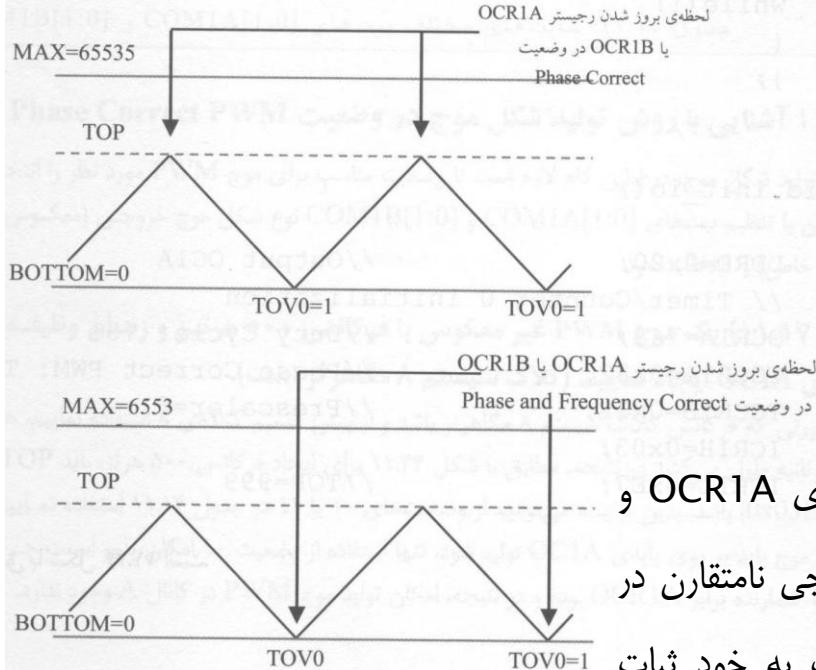
```
#include <mega32.h>
void init_io();
void main(void)
{
    init_io();
    while(1)
    {
    };
}

Void init_io()
{
    DDRD=0x20;          //Output OC1A
    // Timer/counter 1 initialization
    OCR1A=749;         //Duty Cycle: 75%
    TCCR1A=0X82;        //Phase Correct PWM:CTC: TOP=ICR1
    TCCR1B=0X12;        //Prescaler=8
    ICR1H=0x03;
    ICR1L=0xE7;        // TOP=999
}
```



# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با وضعیت Phase and Frequency Correct PWM



این یک وضعیت جدید است که تنها مخصوص تایمر ۱۶ بیتی است و بسیار شبیه به Phase Correct PWM اس این تفاوت که لحظه انتقال مقدار بافر مضاعف به ثبات بافر متفاوت است.

همانطور که قبلاً گفته شد در وضعیتهای PWM، ثباتهای OCR1A و OCR1B دارای بافر مضاعف میباشند، این مسئله از ایجاد خروجی نامتقارن در خروجی جلوگیری میکند. اما لحظه‌ای که مقدار بافر مضاعف به خود ثبات منتقل میشود در وضعیتهای Phase and Frequency PWM و Phase Correct PWM با یکدیگر متفاوت است. این مسئله در شکل نشان داده شده است.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با وضعیت Phase and Frequency Correct PWM

تایмер/کانتر یک دارای دو وضعیت Phase and Frequency PWM است که در جدول زیر آمده اند.

شماره	WGM1 3	WGM1 2	WGM1 1	WGM1 0	وضعیت کاری	TOP	بروزرسانی OCR1x	لحظه‌ی یک شدن TOV1 پرچم
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM

همانطور که ملاحظه میکنید، اختلاف دو وضعیت کاری در مقدار TOP آنهاست که در ادامه به بررسی هر یک از این موارد میپردازیم.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## Phase and Frequency Correct PWM, TOP=ICR1

در این حالت TOP شمارنده برابر با مقدار ثبات ICR1 شده و رزولوشن سیگنال PWM از رابطه زیر به دست می‌آید:

$$resolution = \frac{\log(ICR1 + 1)}{\log(2)}$$

بدین ترتیب، ثبات TCNT1 اعداد صفر تا ICR1 را شمرده و پس از رسیدن به مقدار ICR1، به صورت نزولی تا صفر می‌شمارد. در لحظه صفر شدن شمارنده، فلگ TOV1 یک می‌شود. فرکانس سیگنال PWM ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{2 \times N \times ICR1}$$

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## Phase and Frequency Correct PWM, TOP=OCR1A

در این حالت TOP شمارنده برابر با مقدار ثبات OCR1A شده و رزولوشن سیگنال PWM از رابطه زیر به دست

می‌آید:

$$resolution = \frac{\log(OCR1A + 1)}{\log(2)}$$

بدین ترتیب، ثبات TCNT1 اعداد صفر تا OCR1A را شمرده و پس از رسیدن به مقدار OCR1A، به صورت نزولی تا صفر می‌شمارد. در لحظه صفر شدن شمارنده، فلگ TOV1 یک می‌شود. فرکانس سیگنال PWM ایجاد شده از رابطه زیر قابل محاسبه است:

$$Frequency = \frac{f_{CLK}}{2 \times N \times OCR1A}$$

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با روش تنظیم واحد تولید شکل موج

چنانچه بخواهیم واحد تولید موج را فعال نماییم تا به صورت خودکار شکل موجی را بر روی پایه های OC1B یا OC1A یا COM1A[1:0] و COM1B[1:0] تنظیم شوند. حالتها مختلفی را که این بیتها شامل میشوند در جدول زیر مشاهده مینمایید.

COM1A[1:0] و COM1B[1:0]	COM1A[1:0] و COM1B[1:0]	وضعیت پایه های (I/O معمولی)
0	0	غیر فعال (I/O معمولی)
0	1	در صورتی که WGM1[3:0] برابر با ۹ باشد: پایه OC1A در لحظه‌ی تطبیق مقایسه، Toggle می‌شود و پایه OC1B غیر فعال است. برای سایر وضعیت‌ها، پایه‌های OC1B و OC1A، عملکرد تولید شکل موج نداشته و I/O معمولی‌اند.
1	0	پاک کردن OC1B یا OC1A در تطبیق مقایسه در لبه‌ی شمارش صعودی و یک کردن آن در لبه‌ی شمارش نزولی (PWM غیر معکوس)
1	1	یک کردن OC1B یا OC1A در تطبیق مقایسه در لبه‌ی شمارش صعودی و پاک کردن آن در لبه‌ی شمارش نزولی (PWM معکوس)

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: ایجاد سیگنال PWM غیرمعکوس با فرکانس ۵۰۰ هرتز و زمان وظیفه ۷۵٪ بروی پایه OC1A

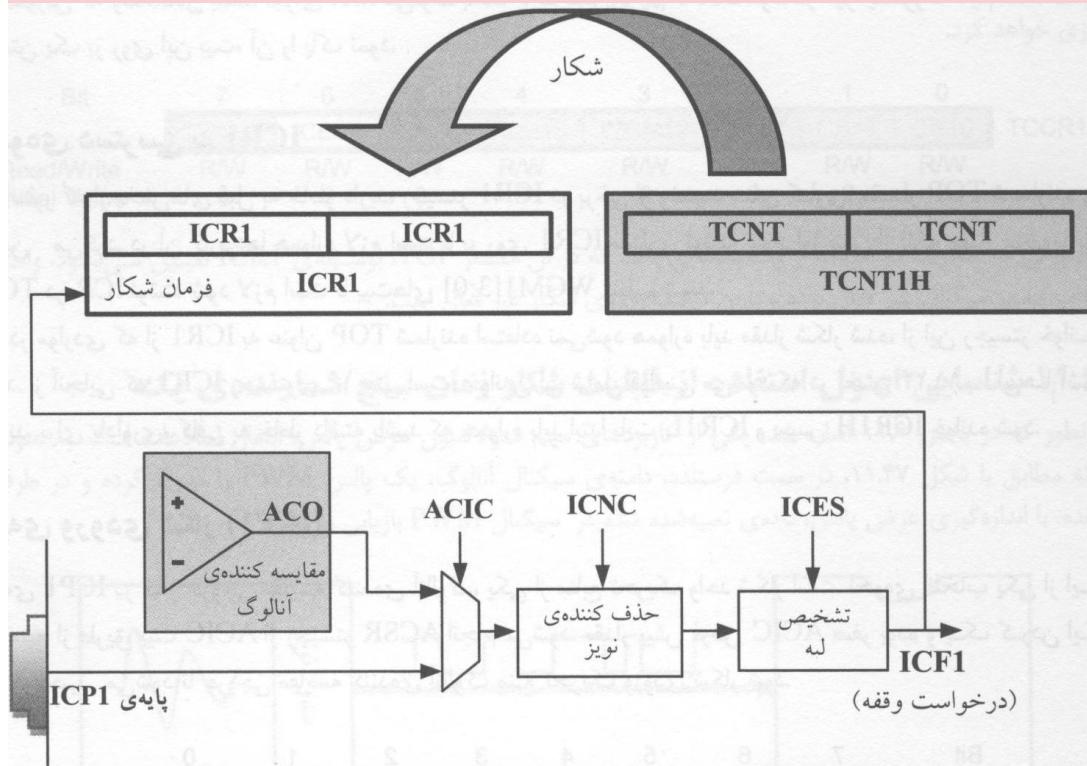
```
#include <mega32.h>
void init_io();
void main(void)
{
    init_io();
    while(1)
    {
    };
}

void init_io()
{
    DDRD=0x20;          //Output OC1A
    // Timer/counter 1 initialization
    OCR1A=749;         //Duty Cycle: 75%
    TCCR1A=0X80;        //Phase & Frequency Correct PWM: TOP=ICR1
    TCCR1B=0X12;        //Prescaler=8
    ICR1H=0x03;
    ICR1L=0xE7;        // TOP=999
}
```

پاسخ این مثال همانند مثال قبلی است و شکل موج خروجی آنها نیز یکسان است، با این تفاوت که در صورت تغییر مقدار OCR1A و OCR1B، لحظه بروز رسانی این ثباتها متفاوت است.  
تفاوت کد این برنامه و مثال قبلی در این است که تایمر/کانتر یک در وضعیت Phase and Frequency Correct تنظیم شده است.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با قابلیت شکار (capture) در تایمر/کانتر یک



ورودی کپچر یا شکار یکی از امکانات ارزشمند تایمر/کانتر یک است که امکان اندازه گیریهای زمانی دقیق را فراهم میکند. در شکل زیر اجزایی که به طور مستقیم مربوط به واحد کپچر نیستند با رنگ خاکستری مشخص شده اند.

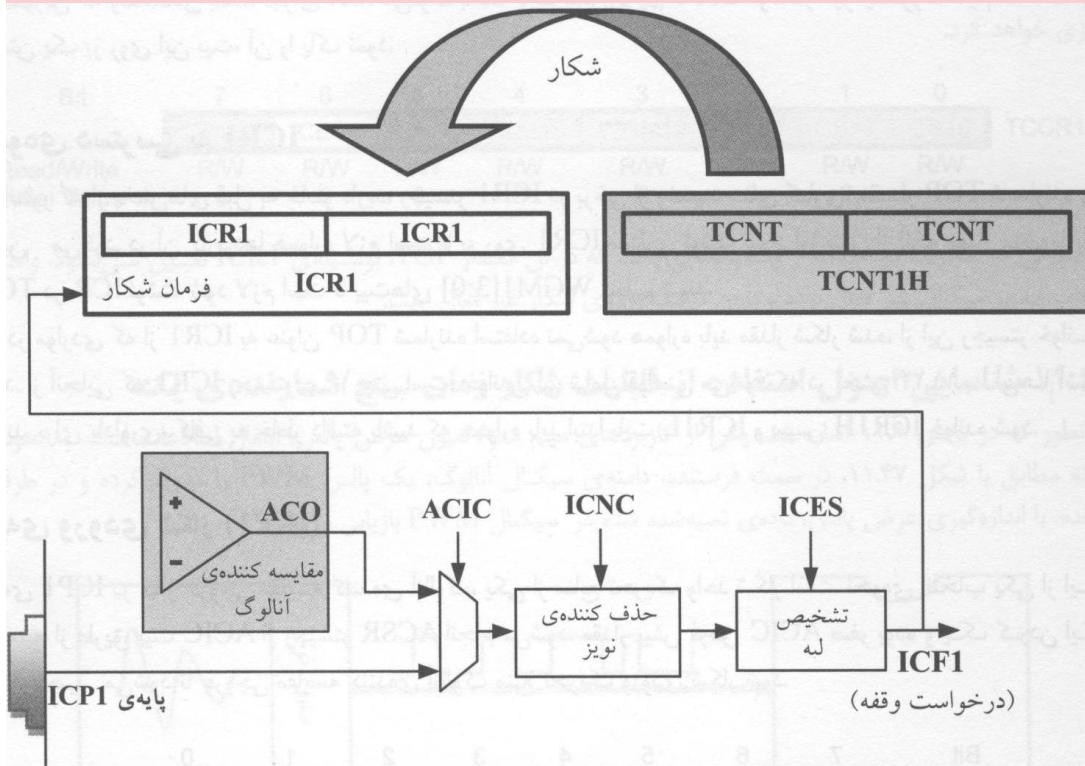
همانطور که در شکل مشخص است، واحد کپچر میتواند به دو روش تحریک گردد:

- ۱- پایه خروجی ICP1 (عملکرد دوم (PD6

- ۲- خروجی مقایسه کننده آنالوگ

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با قابلیت شکار (capture) در تایمر/کانتر یک



زمانی که تغییری در سطح پایه ICP1 یا خروجی مقایسه کننده آنالوگ ایجاد شود، در صورتی که این تغییر با تنظیمات بلوک آشکارساز لبه (بالا رونده یا پایین رونده) همخوانی داشته باشد، واحد کپچر تحریک شده و یک نسخه از محتوای ثبات ICR1 را کپچر کرده و در ثبات TCNT1 ثبت میکند.

همزمان با این کپی شدن، فلگ ICF1 یک شده و تحریک شده واحد کپچر را اعلام مینماید.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با قابلیت شکار (capture) در تایمر/کانتر یک

فلگ ICF1، بیت پنجم از ثبات TIFR است:

Bit	7	6	5	4	3	2	1	0	TIFR
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

در صورتی که بیت TICIE1 از ثبات TIMSK و بیت فعال ساز عمومی وقفه ها یک باشند، یک شدن ICF1 میتواند باعث درخواست وقفه شود.

Bit	7	6	5	4	3	2	1	0	TIMSK
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

در صورتی که وقفه فعال باشد، اجرای ISR میتواند باعث پاک شدن فلگ ICF1 شود در غیر اینصورت لازم است تا با نوشتن یک بر روی این بیت، آن را پاک نمود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با قابلیت شکار (capture) در تایمر/کانتر یک

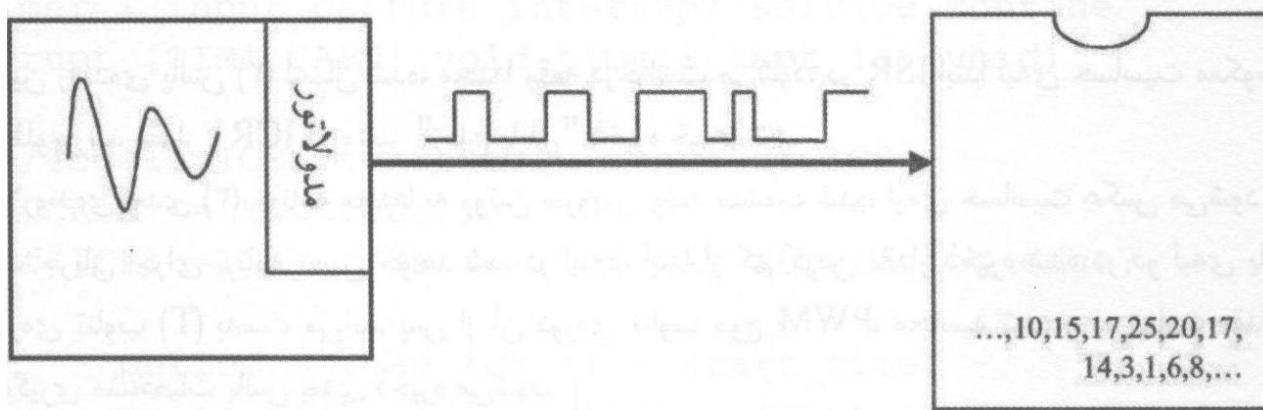
### نحوی دسترسی به ICR1

در مواردی که از ICR1 به عنوان TOP شمارنده استفاده نمیشود همواره باید مقدار کپچر از این ثبات خوانده شود. از آنجا که ICR1 ثباتی ۱۶ بیتی است، خواندن آن شامل قواعدی است که قبل از آن آشنا شدید. برای یادآوری کافی به خاطر داشته باشید که همواره باید ابتدا بایت ICR1L و سپس بایت ICR1H خوانده شود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## شناسایی اصول استفاده از قابلیت کپچر تایمر/کانتر یک

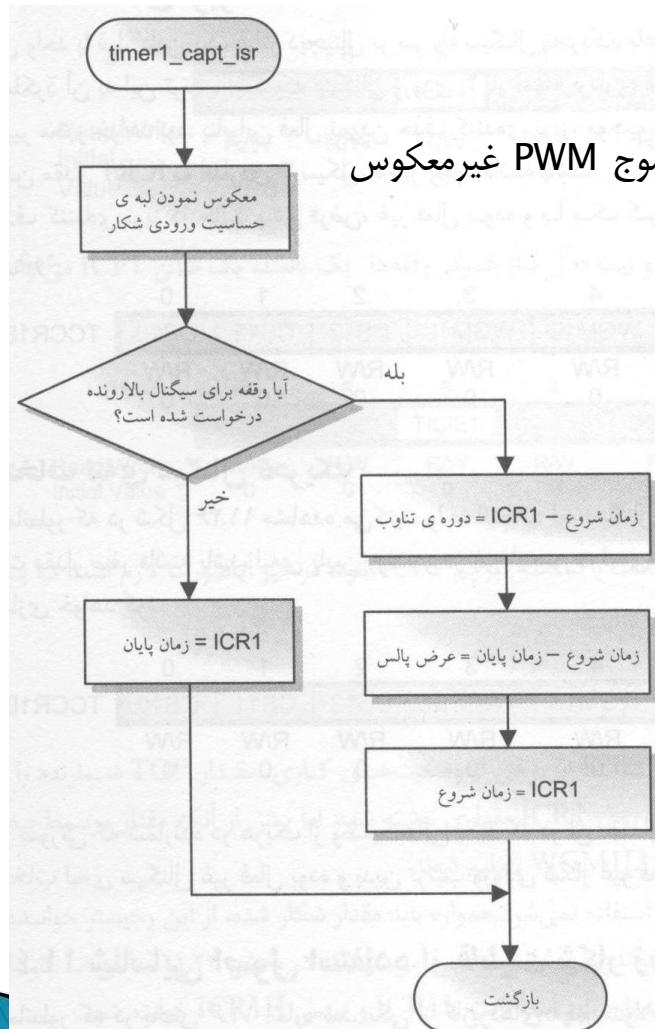
همانطور که قبلاً اشاره شد، یکی از کاربردهای مهم مدولاسیون عرض پالس، انتقال اطلاعات است. به عنوان نمونه مطابق شکل زیر، در سمت فرستنده دامنه سیگنال آنالوگ، یک پالس PWM را مدوله کرده و در طرف گیرنده، با اندازه گیری عرض پالس، داده تعبیه شده در سیگنال PWM بازیابی می‌شود.



اگر چه برای اندازه‌گیری عرض پالس و فرکانس یک سیگنال PWM می‌توان از یک پایه وقفه خارجی به همراه تایmer استفاده کرد اما ورودی کپچر تایمر/کانتر یک، قابلیت وقفه خارجی و شمارنده به طور یکجا در اختیار قرار داده و از آنجا که عملیات اندازه‌گیری به صورت سخت‌افزاری انجام می‌شود دقت اندازه‌گیری بالاتر بوده و مشغولیت پردازندۀ نیز کمتر است.

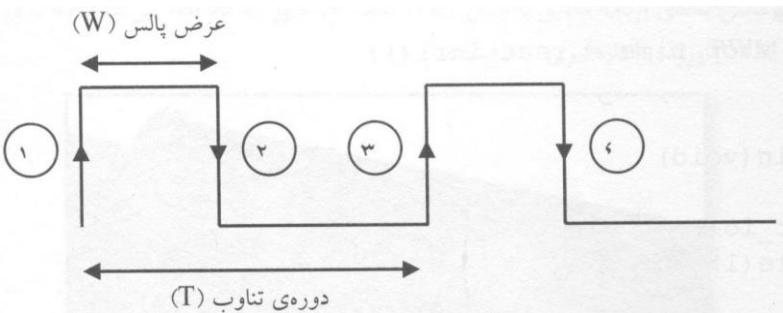
# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## شناسایی اصول استفاده از قابلیت کپچر تایمر/کانتر یک



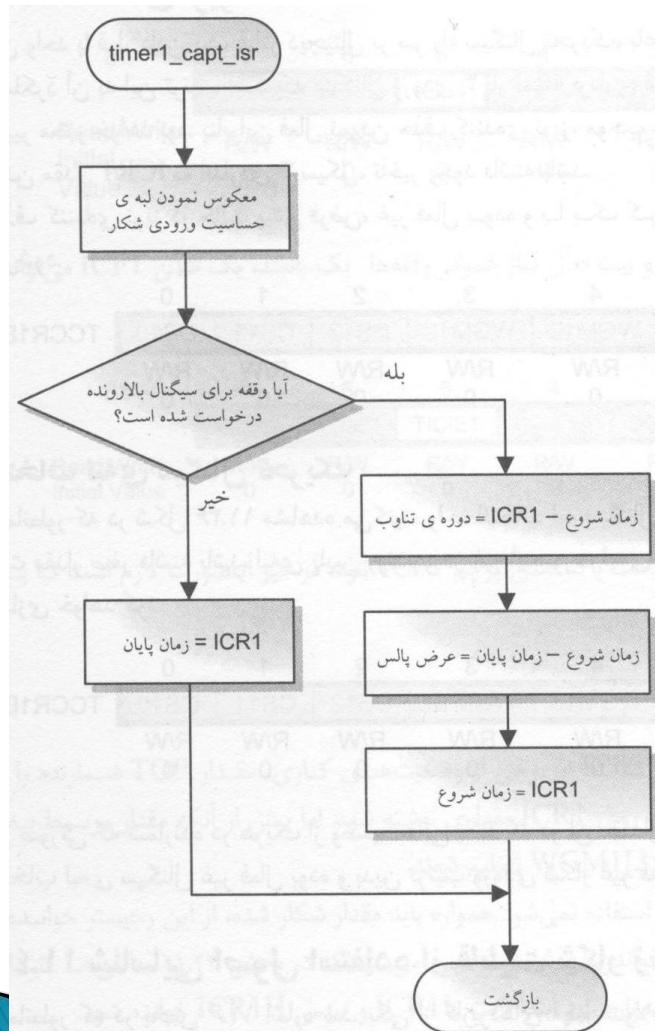
فلوچارت ورودی کپچر، برای اندازه‌گیری دوره تناوب و عرض پالس یک موج PWM غیرمعکوس در شکل روبرو آمده است.

فرض کنید که پالس PWM ورودی، مطابق با شکل زیر است:

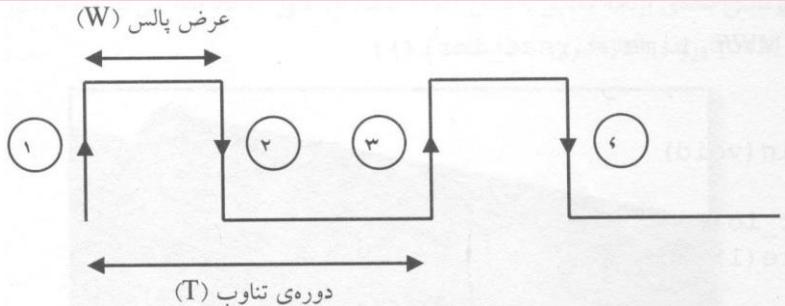


با فرض اینکه قبل از ورودی حساسیت ورودی کپچر به صورت بالا رونده تنظیم شده است، با اعمال اولین لبه بالارونده (لحظه ۱) به پایه ICP1 میتوان تشخیص داد که این وقفه برای لبه بالارونده سیگنال درخواست شده است و بنابراین مطابق با فلوچارت، برنامه منشعب میشود. از آنجایی که اکنون مقدار متغیر مربوط به «زمان شروع» صفر است، بنابراین در متغیر «دورهی تناوب» مقدار ICR1 ذخیره خواهد شد. در گام بعدی به همین دلیل مقدار متغیر «عرض پالس» نیز صفر خواهد شد. حال مقدار ICR1 برای محاسبه دورهی تناوب پالس بعدی در متغیر «زمان شروع» ذخیره میگردد. از آنجایی که همواره میتوان در این ISR مشخصات مربوط به پالس قبلی را محاسبه نمود، با اولین درخواست وقفه، هیچ اطلاعاتی در رابطه با T و W بدست نمی‌آید.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر



## شناسایی اصول استفاده از قابلیت کپچر تایمر/کانتر یک



با لبه پایین رونده پالس (۲) اعمال شده، مجدداً وقفه درخواست می‌شود، در ISR ابتدا لبه هی حساسیت معکوس شده و باتوجه به فلوچارت، مقدار ICR1 در متغیر «زمان پایان» ذخیره خواهد شد.

با لبه بالا رونده بعدی (۳)، برنامه مجدداً به روتین سرویس وقفه منشعب شده، لبه هی حساسیت عکس می‌شود و باتوجه به فلوچارت، جریان اجرای برنامه تعیین خواهد شد. در اینجا ابتدا از کم کردن مقدار ذخیره شده در دو لبه هی بالارونده متوالی، دوره‌ی تناوب (T) بدست می‌آید. پس از دوره‌ی تناوب سیگنال PWM، محاسبه شده و در نهایت مقدار ICR1 برای اندازه‌گیری مشخصات پالس بعدی ذخیره می‌شود.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

مثال: برنامه

```
#include <mega32.h>
#include <lcd.h>
#include <studio.h>
#include <delay.h>

//Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x1B          ;PORTA
#endifasm
#define ICP1 PIND.6

void init_io();
Unsigned int read_icrl();
Unsigned int start_time,stop_time,period,pulse_width;
Unsigned char str[20]
```

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: برنامه

```
//Timer 1 input capture interrupt service routine
Interrupt [TIM1_CAPT] void timer1_capt_isr(void)
{
    //Toggle ICES to reverse sense of ICP
    TCCR1B ^= (1<<ICES1);
    if(ICP1==1)    //Interrupt was for rising edge?
    {
        period = read_incl() - start_time;
        pulse_width = stop_time - start_time;
        start_time = read_icrl();
    }
    else
        stop_time = read_icrl();
}
```

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

مثال: برنامه

```
void main(void)
{
    init_io(0);
    while(1)
    {
        lcd_clear();
        lcd_gotoxy(0,0);
        sprintf(str,"Period: %u uS",period);
        lcd_puts(str);

        lcd_gotoxy(0,1);
        sprintf(str,"PW: %u uS",pulse_width);
        lcd_puts(str);
        delay_ms(1000);
    };
}
```

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

مثال: برنامه

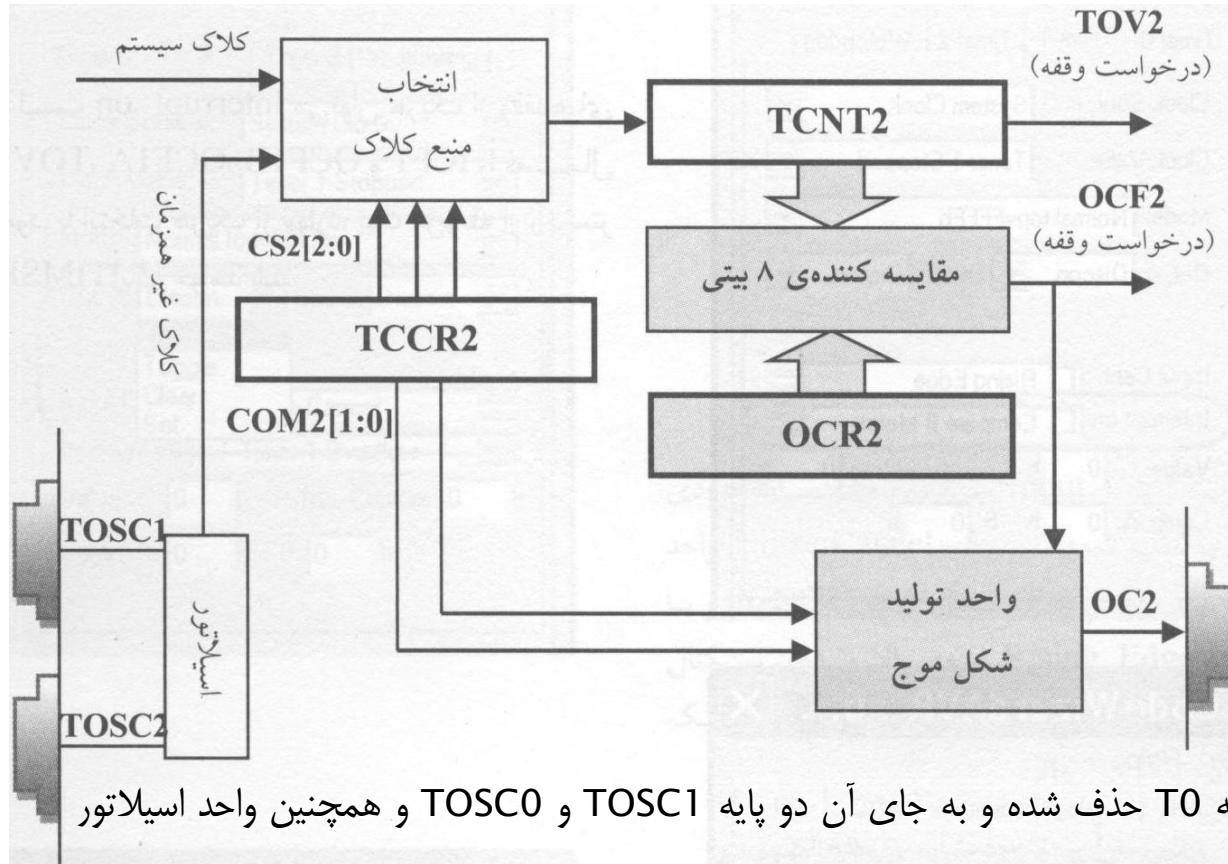
```
Void init_io()
{
    lcd_init(16);          //Initialize LCD
    PORTD |= (1<<PD6);   //Enable ICP1 Pull-up
    DDRB = 0x08;           //Output OC0
    // Timer/counter 1 initialization
    TCCR1A=0X00;           //Mode:Normal
    TCCR1B=0X42;           //Prescaler=8
                           //Input Capture: Rising Edge
                           //Noise Canceler: off
    TIMSK=0X20;             //ICF1 Interrupt Enable
    #asm("sei");           //Global Interrupt Enable

    // Timer/Counter 0 initialization
    TCCR0=0x6A;             //Prescaler=8
                           //Mode: Non-Inverting Fast PWM
    OCR0=191;               //Duty Cycle: 75%
}

Unsigned int read_icr1()
{
    unsigned char temp;
    temp=ICR1L           //First Read Low Byte
    return 256*ICR1H+ICR1L;
}
```

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با سخت افزار تایمر/کانتر دو

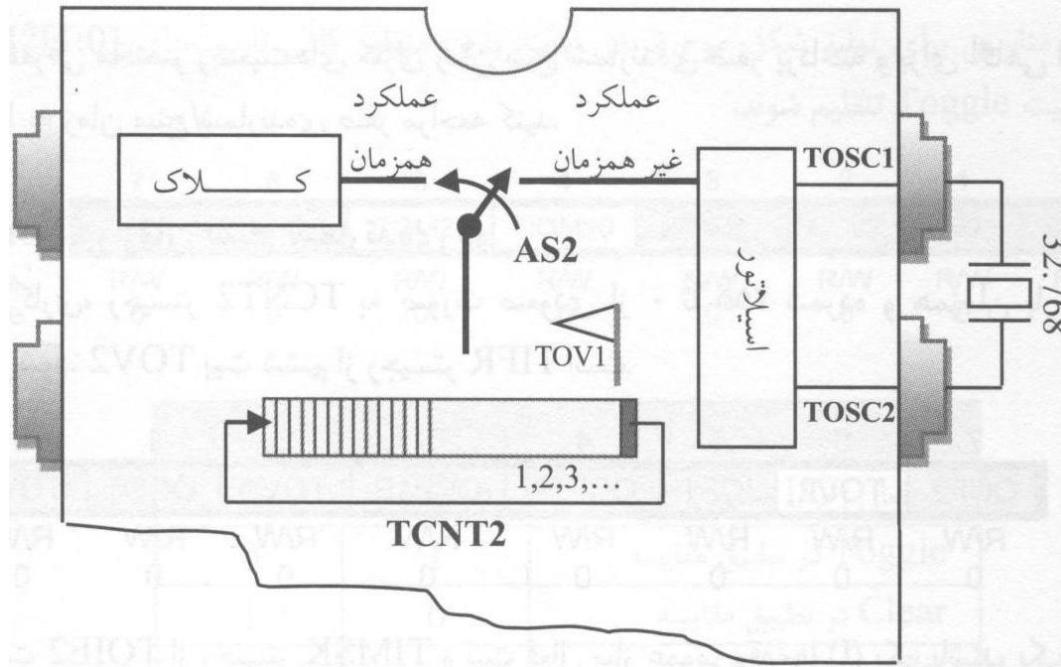


این واحد جنبی همانند تایمر/کانتر صفر، ۸بیتی است با این تفاوت که بر خلاف آن دارای ورودی شمارنده نبوده و به جای آن، قابلیت کار در وضعیت آسنکرون را دارد. بلوک منطقی آن را در شکل رو برو مشاهده می‌نمایید.

همانطور که ملاحظه میکنید، پایه T0 حذف شده و به جای آن دو پایه TOSC0 و TOSC1 و همچنین واحد اسیلاتور اضافه شده است.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با مفهوم عملکرد غیرهمzman در تایمر/کانتر دو



قبل مشاهده کردید که با اعمال یک پالس ساعت منظم به شمارنده، میتوانیم از آن به عنوان تایمر استفاده نماییم. در تمام مواردی که تا کنون از تایمر/کانتر به عنوان تایمر استفاده کردایم این پالس ساعت منظم از کلاک سیستم تامین شده است. حال با وضعیتی آشنا خواهیم شد که پالس تایمر از یک منبع غیرهمzman با کلاک سیستم تامین میشود.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با مفهوم عملکرد غیرهمزمان در تایمر/کانتر دو

همانطور که در شکل مشخص است در صورتی که بیت AS2 از ثبات ASSR، یک شود منبع کلاک تایмер، از کلاک سیستم به یک اسیلاتور با فرکانس ۳۲.۷۶۸ تغییر خواهد کرد.

Bit	7	6	5	4	3	2	1	0	ASSR
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

زمانی که از عملکرد غیرهمزمان استفاده میشود، کلاک سیستم باید حداقل ۴ برابر کلاک غیرهمزمان باشد.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با وضعيت های کاری تایمر/کانتر دو

وضعيت های کاری اين تایمر/کانتر مشابه با تایمر/کانتر صفر بوده و بدیهی است که توسط بیتهای [1:0] انتخاب میشود.

Bit	7	6	5	4	3	2	1	0	TCCR2
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

نحوه انتخاب وضعيت مورد نظر با اين دو بيت، در جدول زير آمده است

WGM01	WGM00	وضعيت زمان سنج
0	0	Normal
0	1	PWM, Phase Correct
1	0	Clear Timer on Compare Match (CTC)
1	1	Fast PWM

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## وضعیت Normal در تایمر دو

در این وضعیت، ثبات TCNT2 به صورت صعودی از صفر تا ۲۵۵ شمرده و همزمان با صفر شدن آن، فلگ TOV2 یک میشود. TOV2 بیت ششم از ثبات TIFR است.

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

در صورتی که بیت TOIE2 از ثبات TIMSK و بیت فعال ساز عمومی وقفه‌ها (I) یک باشد، یک شدن فلگ TOV2 میتواند باعث درخواست وقفه گردد.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## وضعیت Normal در تایمر دو

همچنین از واحد مقایسه تایمر/کانتر دو میتوان برای ایجاد وقفه‌های تطبیق مقایسه با اختلاف زمانی مشخص از رخدادهای سرریز استفاده کرد. بدین منظور کافی است تا مقدار مقایسه، در ثبات OCR2 بارگذاری شود. همانطور که میدانید، محتوى ثبات 2 به طور پیوسته با مقدار OCR2 مقایسه شده و در لحظه برابری (تطبیق مقایسه) فلگ OCF2 یک میشود.

Bit	7	6	5	4	3	2	1	0	TIFR
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

در صورتی که بیت OCIE2 از ثبات TIMSK و بیت فعال ساز عمومی وقفه‌ها (I) یک باشد، یک شدن فلگ OCF2 میتواند باعث درخواست وقفه شود.

Bit	7	6	5	4	3	2	1	0	TIMSK
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

اگر چه میتوان با تنظیم بیتهای COM2[1:0] با استفاده از واحد تولید موج، موج مربعی ایجاد نمود اما به دلیل اینکه برای مقدار اولیه دادن به شمارنده، زمان زیادی از CPU گرفته میشود، از وضعیت CTC برای تولید شکل موج استفاده میکنیم.

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## وضعیت CTC در تایمر دو

در این وضعیت با رسیدن شمارنده به مقدار OCR2، ثبات TCNT2 پاک شده و مجدداً از صفر شروع به شمارش می‌کند. بدین ترتیب رشته شمارش، محدود به اعداد صفر تا OCR2 شده و همزمان با رسیدن شمارنده به مقدار TOP، فلگ OCF2 یک می‌شود.

این وضعیت برای تولید شکل موج مربعی بسیار مناسب است. بدین منظور کافی است تا بیتهاي COM2[1:0] مطابق با جدول زیر تنظیم شوند.

Bit	7	6	5	4	3	2	1	0	TCCR2
FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

COM21	COM20	وضعیت پین OC2
0	0	غیر فعال (I/O معمولی)
0	1	در تطبیق مقایسه Toggle
1	0	در تطبیق مقایسه Clear
1	1	در تطبیق مقایسه Set

عملکرد بیتهاي COM2[1:0] در وضعیتهای غیر PWM

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## وضعیت Fast PWM در تایمر دو

وضعیت Fast PWM یکی از دو وضعیت PWM تایمر/کانتر دو است. در این حالت، ثبات TCNT2 به طور افزایشی تا ۲۵۵ شمرده و با رسیدن به TOP، مجدداً از صفر شروع به شمارش می‌کند. در لحظه‌ی تطبیق مقایسه، بسته به وضعیت بیت‌های COM2[1:0] پایه OC2 مطابق با جدول زیر تغییر وضعیت میدهد.

COM21	COM20	وضعیت پین OC2
0	0	غیر فعال (I/O معمولی)
0	1	رزرو شده
1	0	پاک کردن OC2 در تطبیق مقایسه و یک کردن آن در PWM) TOP غیر معکوس)
1	1	یک کردن OC2 در تطبیق مقایسه و پاک کردن آن در PWM) TOP معکوس)

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## وضعیت Phase Correct PWM در تایمر دو

مشابه با تایمر صفر، ثبات TCNT2 به طور صعودی از صفر تا ۲۵۵ شمرده و پس از آن به صورت نزولی تا صفر شمارش میکند. از آنجایی که در این حالت، حامل مدولاسیون دو شیبه است در هر سیکل شمارش، دو تطبیق مقایسه وجود خواهد داشت. در هر یک از دو تطبیق مقایسه، مطابق با تنظیمات بیت‌های COM2[1:0] وضعیت پایه‌ی OC2 تغییر میکند.

COM21	COM20	وضعیت پین OC2
0	0	غیر فعال (I/O معمولی)
0	1	رزو شده
1	0	پاک کردن OC2 در تطبیق مقایسه‌ی شمارش صعودی و یک کردن آن در شمارش نزولی (PWM غیر معکوس)
1	1	یک کردن OC2 در تطبیق مقایسه‌ی شمارش صعودی و پاک کردن آن در شمارش نزولی (PWM معکوس)

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با واحد انتخاب منبع کلک در تایمر/کانتر دو

در این جدول حالت‌های لبھی بالارونده و پایین رونده T2 وجود نداشته و به جای آنها، دو مقدار تقسیم بر ۳۲ و ۱۲۸ اضافه شده است.

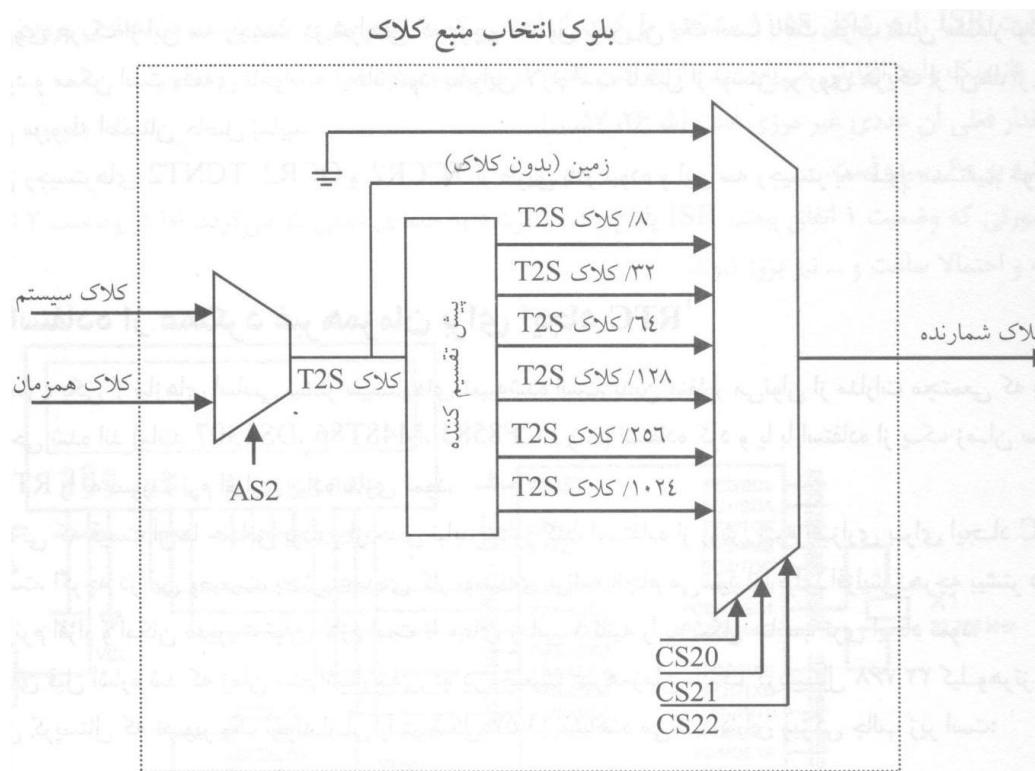
Bit	7	6	5	4	3	2	1	0	TCCR2
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

منبع کلک	CS22	CS21	CS20
بدون کلک (متوقف)	0	0	0
کلک سیستم (بدون تقسیم)	0	0	1
کلک ای/ت2S	0	1	0
کلای/ت2S	0	1	1
کلای/ت2S	1	0	0
کلای/ت2S	1	0	1
کلای/ت2S	1	1	0
کلای/ت2S	1	1	1

# آشنایی با میکروکنترلر AVR بخش تایمر/کانتر

## آشنایی با واحد انتخاب منبع کلاک در تایمر/کانتر دو

ممکن است این سوال مطرح شود که منظور از T2S چیست؟ در پاسخ باید گفت که منبع کلاک تایмер دو میتواند از کلاک سیستم و یا کلاک غیرهمزمان باشد که توسط بیت AS2 تعیین میشود.



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## آشنایی با سایر بیتهای ثبات وضعیت غیرهمزمان

با بیت AS2 از ثبات ASSR آشنا شدید و میدانید که با یک کردن این بیت، منبع کلاک شمارنده از کلاک غیرهمزمان تامین میشود. اکنون به بررسی سایر بیتهای ثبات ASSR میپردازیم.

Bit	7	6	5	4	3	2	1	0	ASSR
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

زمانی که بر روی یکی از سه ثبات TCCR2، OCR2 و TCNT2 مقداری نوشته میشود، آن مقدار به یک ثبات موقتی منتقل شده و پس از دو لبه بالارونده TOSC1 مقدار مورد نظر به مقصد منتقل میشود.

از آنجا که کلاک غیرهمزمان (۳۲۷۶۸ هرتز) بسیار پایین تر از کلاک سیستم است، لذا اعمال این دو لبه بالارونده، ممکن است زمان نسبتاً زیادی به طول انجامد. برای جلوگیری از بازنویسی مقدار ثبات موقت قبل از انتقال به خود ثبات، سه بافر موقت در نظر گرفته شده است. این بافرها تنها در عملکرد غیرهمزمان فعال بوده و در شرایطی که بیت AS2 صفر است عیرفعال میشوند. با نوشتن یک مقدار جدید در هر یک از این سه ثبات، فلگ مشغول بودن آن یک شده و تا اتمام بروز رسانی یک باقی میماند. زمانی که محتوى بافر موقت به خود ثبات منتقل شد، این فلگ صفر میشود. فلگ مشغول بودن ثباتهای TCCR2، TCNT2، OCR2 و TCR2UB به ترتیب OCR2UB، TCNT2UB و TCR2UB میباشند. بنابراین لازم است تا قبل از نوشتن بر روی هر یک از آنها، از صفر بودن فلگ مربوطه اطمینان حاصل نمایید.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## استفاده از عملکرد غیرهمزان برای ایجاد RTC

آگاهی از زمان، یکی از نیازهای اساسی بیشتر سیستم‌های تعبیه شده است. بدین منظور میتواند از مدارات مجتمعی که بدین منظور طراحی شده اند (مانند DS1307، M48T86، PCF8583 و ....) استفاده کرد و یا با استفاده از یک تایمر، عملکرد RTC را به صورت نرم افزاری پیاده سازی نمود.

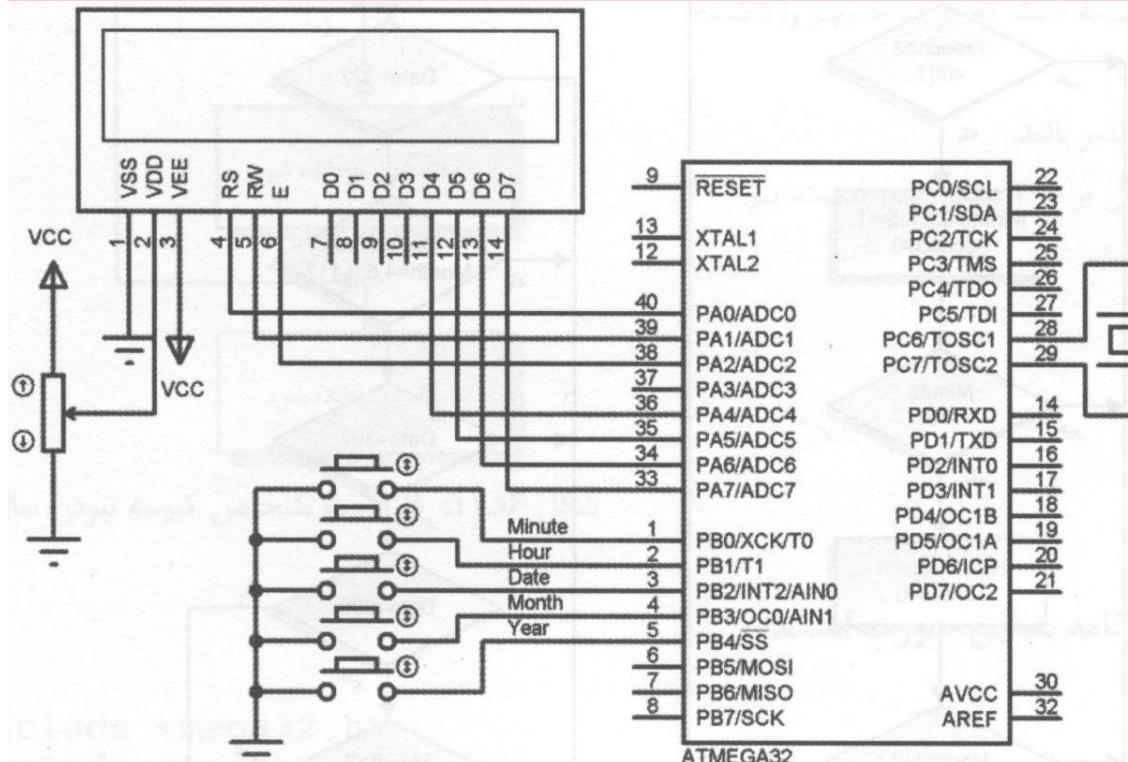
قبل اشاره شد که تایمر/کانتر دو در وضعیت غیرهمزان با یک کریستال ۳۲۷۶۸ کیلوهرتز کار میکند. این کریستال دارای ویژگی جالب زیر است:

$$\frac{32768}{128} = 256$$

این بدان معناست که اگر فرکانس کریستال ساعت توسط پیش تقسیم کننده بر عدد ۱۲۸ تقسیم شود و به تایمر/کانتر هشت بیتی امال گردد، زمان سرریزی معادل با یک ثانیه خواهد داشت. بدین ترتیب بدون هیچ گونه بار نرم افزاری میتوانیم به زمان سرریز یک ثانیه دست پیدا کنیم. این نتیجه ای ایده آل برای پیاده سازی عملکرد RTC است.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

مثال: طرح یک ساعت و تقویم با نمایش بر روی LCD کاراکتری با استفاده از وضعیت غیرهمزمان تایمر/کانتر دو

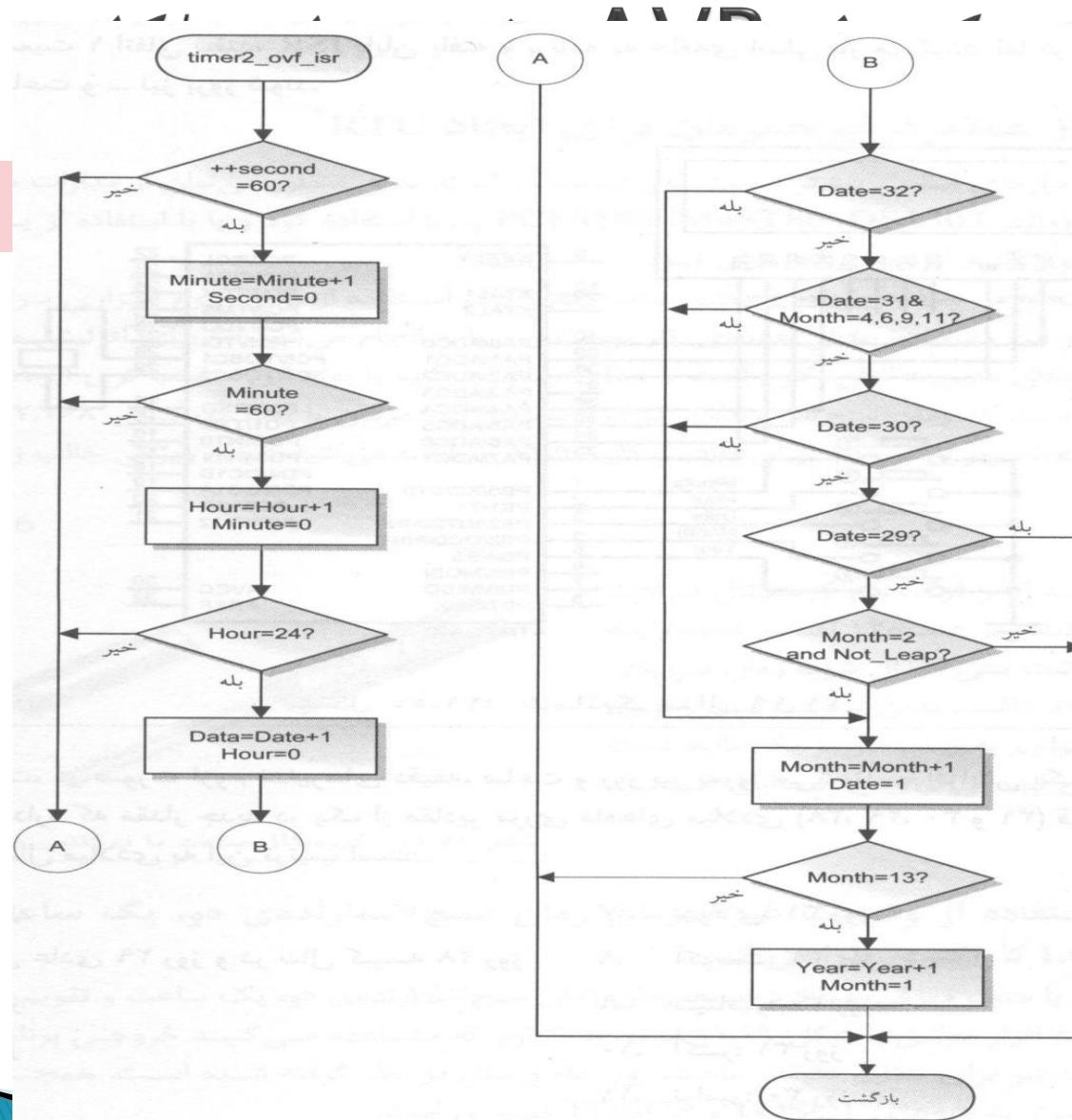


شماییک اتصالات سخت افزار مطابق با شکل زیر است. همانطور که مشاهده میکنید خروجی برنامه یک LCD کاراکتری است و ۵ کلید نیز برای تنظیم دقیقه، ساعت، روز، ماه و سال در نظر گرفته شده است. همچنین اتصال یک کریستال ۳۲۷۶۸ هرتزی به پایه های TOSC1 و TOSC2 ضروری است.

آشنایی با مید

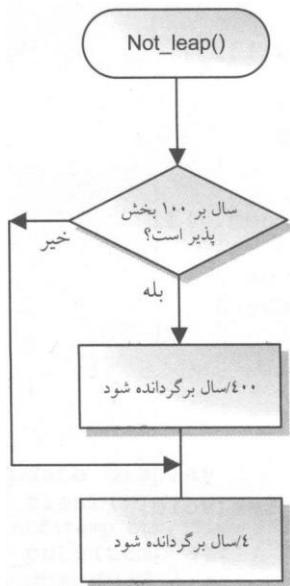
### مثال: طرح یک ساعت و تقدیر

مهمترین بخش برنامه، روتین وقفه تایمر است. با فرض اینکه تایمر/کانتر دو در وضعیت غیرهمزمان و پیش تقسیم کننده ۱۲۸ تنظیم شده باشد، پس زا TCNT2 هر یک ثانیه ثبات سرریز شده و به این ترتیب برنامه به ISR منشعب میشود.



# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## مثال: طرح یک ساعت و تقویم



- در صورتی که سال کبیسه نباشد، مقدار **True** و در صورت کبیسه بودن **False** را برمیگرداند.  
یک سال میلادی در صورتی کبیسه است که دو شرط زیر را داشته باشد:
- سال بر  $4$  بخش پذیر باشد.
  - در صورتی که سال بر  $100$  بخش پذیر است، بر  $400$  نیز بخش پذیر باشد.

کد برنامه در فایل `main.c` آمده است. آنرا بررسی کنید.

# آشنایی با میکروکنترلر AVR بخش تایмер/کانتر

## مثال: طرح یک ساعت و تقویم

همانطور که ملاحظه میکنندی این برنامه دارای دو زیربرنامه دیگر نیز میباشد:

تابع **init\_io()**: در این تابع، تمام تنظیمات مربوط به ورودی و خروجیها و همچنین تایمر/کانتر دو انجام میشود. نکته ای که در این تابع وجود دارد ایناست که در آن تاخیری به اندازه یک ثانیه برای پایدار شدن اسیلاتور در نظر گرفته شده است. علت این مسئله این است که کریستال فرکانس پایین، زمان شروع به کار نسبتاً زیادی دارد.

تابع **check\_keys()**: بوسیله این تابع، کلیدها ببررسی شده و در صورت لزوم متغیرهای مربوط به زمان بروز میشوند. در این تابع از متغیر Flag برای جلوگیری از افزایش بی مورد متغیر مورد نظر استفاده شده است. تاخیر ۲۰۰ میلی ثانیه ای نیز برای Debounce نمودن کلید بوده که بسته به نوع کلید استفاده شده میتوان آن را تغییر داد و یا حذف نمود.

در این برنامه برای نگهداری متغیرهای مربوطه به زمان از یک ساختار به نام `time` استفاده شده است.