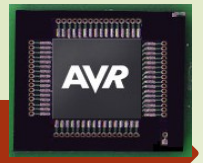


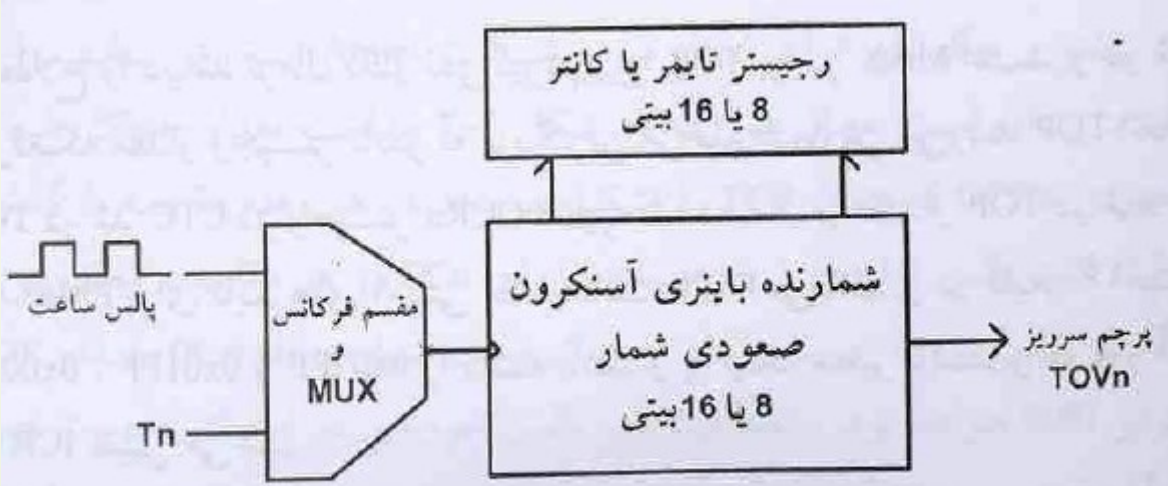
تایمر یا کانتر
(Timer / Counter)

اهداف

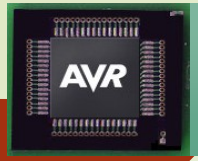
1. آشنایی با انواع مد های تایمر یا کانتر
2. معرفی و نحوه ی کار با رجیستر های تایمر یا کانتر
3. آشنایی با نوشتن برنامه های تایمر یا کانتر به روشهای polling وinterrupt

تایمر یا کانتر (Timer/Counter)

یکی از امکانات جانبی CPU واحد تایمر یا کانتر می باشد که این امکان در همه میکروکنترلر های AVR وجود . ابتدا می خواهیم شما را با مفهوم تایمر یا کانتر آشنا کنیم.



شکل ۶-۱ بلوک دیاگرام تایمر یا کانتر در حالت کلی



تایمر یا کانتر (Timer/Counter)

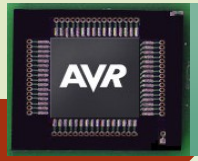
Timer: تایمر به معنی زمان سنج می باشد در واقع تایمر ها همان شمارند های باینری آسنکرون هستند که در مدار های منطقی با آنها آشنا شدیم وقتی به یک شمارنده ۸ بیتی پالس ساعت اعمال می کردیم شمارنده از مقدار 00H تا FFH شروع به شمارش می کرد و در پالس ساعت ۲۵۶ اعمالی ،مقدار شمارنده به صفر برگشت و در اینجا سرریز اتفاق می افتد.

در تایمر درونی میکروکنترلر نیز به همین صورت است با این تفاوت که اجازه اعمال پالس ساعت ،مقدار اولیه و تشخیص سرریز توسط نرم افزار تعیین می گردد.

پس ما به شمارنده ی باینرب که پالس ساعت خود را از کلاک سیستم داخلی تامین می کند اصلاح تایمر را بکار می گیریم تا با کانتر اشتباه نشود.

از کاربرد تایمر ها می توان به اندازه گیری زمان جهت ساعت دیجیتال,زمان کار یک دستگاه,زمان رفت و برگشت سیگنال,اندازه گیری .مان وظیفه (Duty Cycle) ,کنترل موتور و... اشاره نمود.

Counter: کانتر به معنی شمارش کننده می باشد. کانتر نیز مانند تایمر یک شمارنده باینری صعودی شمار آسنکرون می باشد و فرق آن با تایمر در این است که این شمارنده برای شمارش,پالس ساعت خود را از پایه بیرونیTn (nاندیس کانتر است) در لبه بالا رونده یا پایین رونده می تواند دریافت کند. در کانتر نیز اجازه اعمال پالس ساعت,مقدار اولیه و تشخیص سرریز توسط نرم افزار تعیین می گردد.از کاربرد کانتر ها می توان به شمارش رخداد های بیرونی مانند: اندازه گیری RPM دور موتور,شمارش تعداد بسته های یک خط تولید یا شمارش شیشه های



تایمر یا کانتر (Timer/Counter)

نوشابه و ... اشاره کرد.

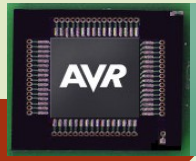
انواع مدهای تایمر یا کانتر

در حالت کلی می توان عملکرد تایمر ها را در چهار مد دسته بندی کرد. از آنجایی که هر چهار مد در تمامی تایمر ها مشترک است ابتدا مدهای تایمر را بررسی می کنیم و سپس رجیستر های مربوط به هر تایمر ها را توضیح می دهیم.اما قبل از توضیح مدها با سه اصطلاح زیر آشنا شوید:

BOTTOM:رسیدن مقدار رجیستر تایمر به 0x00(۸بیتی) یا 0x0000(۱۶ بیتی) را مقدار حداقل تایمر یا Bottom می نامیم. توجه داشته باشید که مقدار Bottom الزاما صفر نیست و می تواند تعیین شود.

MAX: رسیدن مقدار رجیستر تایمر به 0xFF(۸ بیتی) یا 0xFFFF(۱۶ بیتی) را مقدار حداکثر تایمر یا (مقدار ماکزیمم تایمر)MAX می نامیم.

TOP: این اصطلاح را در مد نرمال بکار نمی گیریم زیرا TOPبرابر MAXاست و در مد های دیگر کاربرد دارد. موقعیکه مقدار رجیستر تایمر به بزرگترین مرحله شمارش میرسد TOPاتفاق افتاده است یعنی مقدار TOPدر مد CTCدر رجیستر OCRnذخیره شده است و مقدار TOPدر مد Fast PWM وPhase Correct برای تیمر های ۸ بیتی مقدار ثابت 0xFF می باشد و در تایمر ۱۶بیتی می تواند اعداد ثابت 0x00FF, 0x01FFو 0x03FF



تایمر یا کانتر (Timer/Counter)

را داشته باشد و یا اینکه متغیر باشد و مقدار آن را رجیستر OCR1 و یا ICR1تعیین می کند.

1. مد نرمال (Normal Mode)

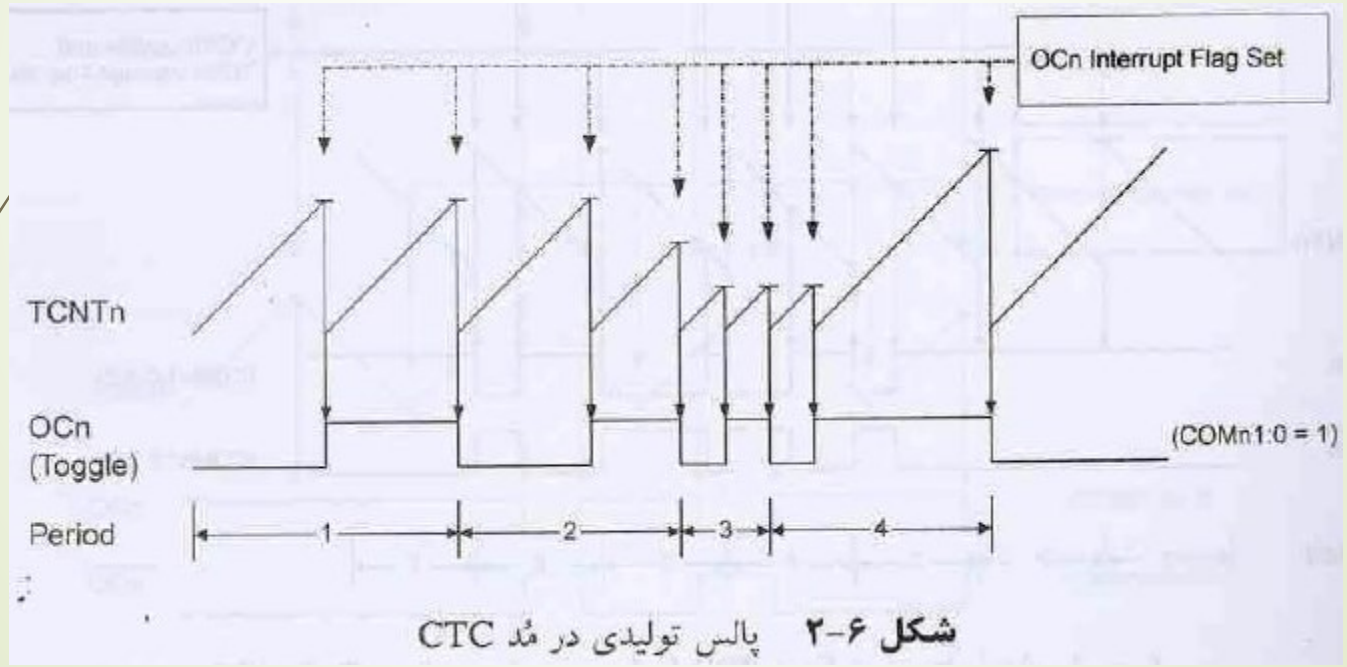
در این حالت مقدار TOP با مقدار MAX برابر است و این مقدار برای تایمر ۸ بیتی 0xFF و برای تایمر ۱۶ بیتی OxFFFFمی باشد. در این مد تایمر از مقدار Bottomشروع به شمارش می کند و تا MAXادامه می یابد.پس از رسیدن به مقدار نهایی با اعمال کلاک بعدی سرریز اتفاق می افتد و پرچم TOVn(nاندیس تایمر) یک می شود و پس از پاک کردن پرچم شمارش مجددا تکرار می گردد توجه داشته باشید که مقدار Bottomرا می توان به عنوان مقداردهی اولیه تایمر در نظر گرفت و پس از هر بار سرریز مقدار Bottomرا تعین نمود. از این مد که در ادامه مثال هایی ارایه خواهد شد برای زمان سنجی دقیق استفاده می کنیم.

۲.مد مقایسه ای CTC(Clear Timer on Compare Match Mode)

در این حالت محتوای رجیستر تایمر TCNTn(nاندیس تایمر) دایما توسط سخت افزار با رجیستر مقایسه ای تایمر OCRn (n اندیس تایمر) مقایسه می شود و هر موقع این برابری رخ دهد محتوی رجیستر تایمر ،صفر می شود . در این مد مقدار TOP را محتوی OCRn تعیین می کند .

تایمر یا کانتر (Timer/Counter)

همانطور که در شکل ۶-۲ پیداست مقدار حداکثر شمارش تغییر کرده است یعنی محتوی OCRn که همان TOP را تشکیل می دهد در هر وقفه مقایسه بارگذاری مجدد شده است و اگر دقت کنید پالس تولیدی بر روی پایه بیرونی Ocn در هر مقایسه معکوس شده است و زمان تناوب آن نیز تغیر کرده است



شکل ۶-۲ پالس تولیدی در مُد CTC

تایمر یا کانتر (Timer/Counter)

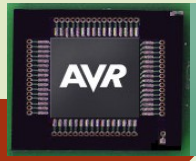
اگر در یک زمان تناوب (period) مقدار TOP ثابت باشد Duty cycle برابر 50% خواهد بود .

رابطه فرکانسی پالس خروجی در این مد بصورت زیر است:

۸بیتی $$focn = \frac{Fclk\_I/O}{2N(1+OCRn)} = \frac{Fosc}{2N(1+OCRn)}$$

۱۶ بیتی $$focA = \frac{Fclk\_I/O}{2N(1+OCRnA)} = \frac{Fosc}{2N(1+OCRnA)}$$

عدد N می تواند یکی از اعداد ثابت ۱ ،۸ ،۶۴،۲۵۶ و یا ۱۰۲۴ انتخاب شود.



تایمر یا کانتر (Timer/Counter)

۳. مد مدولاسیون عرض پالس سریع (Fast PWM Mode)

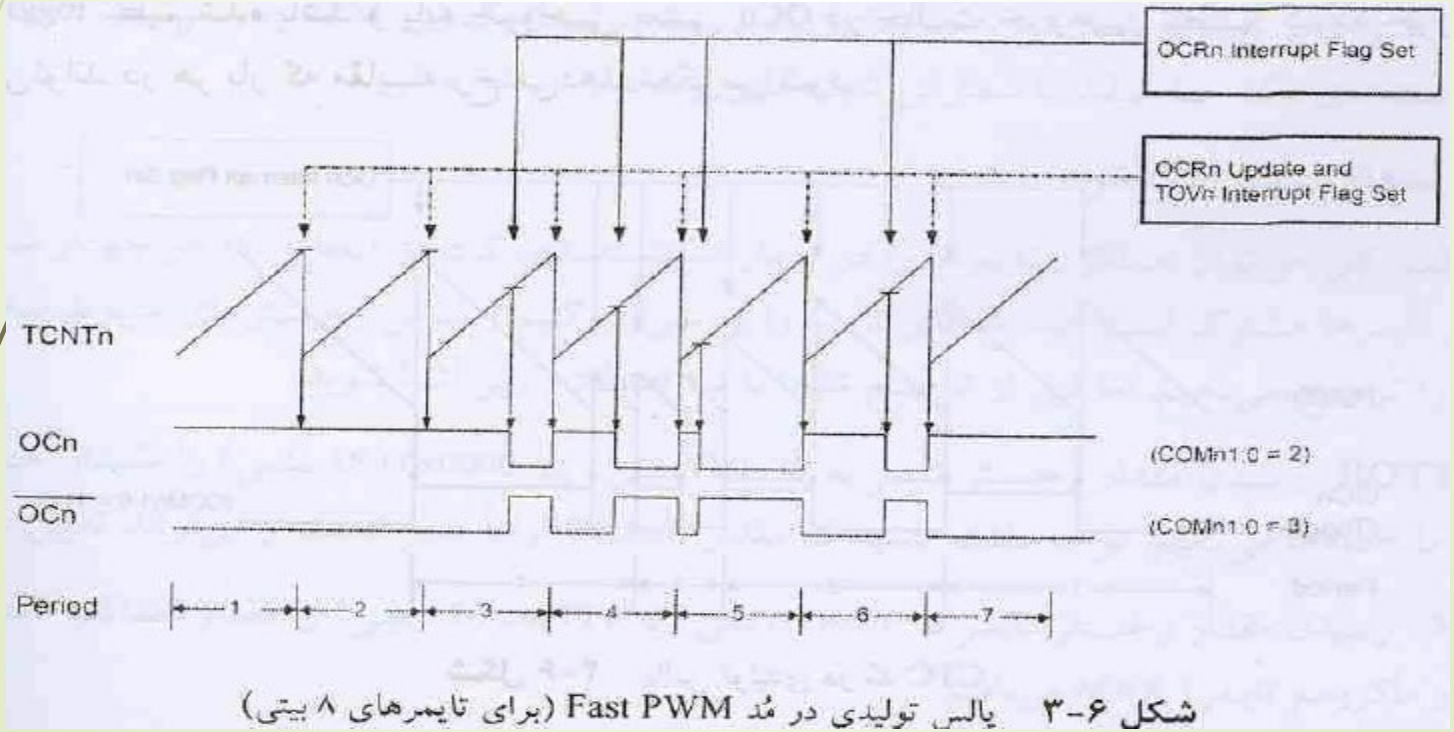
در این حالت محتوی رجیستر تایمر TCNTn از مقدار اولیه Bottom شروع به افزایش می کند و هر موقع که رجیستر تایمر با محتوی رجیستر OCRn برابر شود پایه بیرونی Ocn (در صورت فعال بودن و تنظیم به صورت خروجی) معکوس می شود .

پس از سر ریز تایمر ،محتوی TCNTn از مقدار رجیستر OCRn کمتر شده و خروجی Ocn مجددا معکوس می شود و شمارش دوباره از Botton تا MAX (۸ بیتی) و از Botton تا MAX (۱۶بیتی) ادامه می یابد .

شکل موج تولیدی در تایمر های ۸ بیتی در مد Fast PWM بر خلاف مد CTC دارای زمان تناوب ثابت است و این به دلیل آن است که بعد از مقایسه ، شمارش تا MAX ادامه می باد اما Dute cycle را می توان توسط محتوی OCRn تعیین نمود یعنی زمانی که وقفه سرریز تایمر ، بعد از رسیدن به MAX یا TOP رخ دهد مقدار OCRn را برای دوره ی بعدی تنظیم می کنیم.

تایمر یا کانتر (Timer/Counter)

در شکل ۶-۳ به زمان تناوب دقت کنید که ثابت است و فقط زمان وظیفه (dute cycle)تغییر کرده است ضمن آنکه نحو ی تغیی خروجی نیز توسط رجیستر ها که در ادامه تضیح خواهیم داد قابل تنظیم است.
از کاربرد مهم این نوع پالس می توان به استفاده در کنترل دور موتور DC و منابع تغذیه Switching ،شبیه سازی مبدا D/A اشاره کرد.



شکل ۶-۳ پالس تولیدی در مُد Fast PWM (برای تایمرهای ۸ بیتی)

تایمر یا کانتر (Timer/Counter)

شمارش در تایمر های ۸ بیتی از bottom تا MAX و در تایمر ۱۶ بیتی از bottom تا TOP می باشد .

رابطه فرکانسی پالس خروجی در مدFast PWM بصورت زیر است:

۸بیتی $$focnPWM = \frac{Fclk\_I/O}{N(TOP1)} = \frac{Fosc}{N\times256}$$

۱۶ بیتی $$focnPWM = \frac{Fclk\_I/O}{N(TOP+1)} = \frac{Fosc}{N(TOP+1)}$$

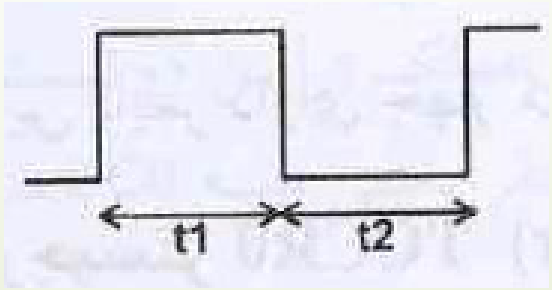
عدد N می تواند یکی از اعداد ثابت ۱ ،۸ ،۶۴،۲۵۶ و یا ۱۰۲۴ انتخاب شود.

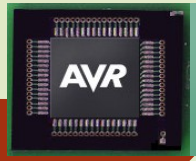
تایمر یا کانتر (Timer/Counter)

رابطه D.C (duty cycle) یا زمان وظیفه بصورت زیر است:

$$\mathrm{D}.C = \frac{t1}{t1 + t2} \times 100$$

$$\mathrm{D}.C = \frac{OCRn - Bottom}{[(TOP + 1) - Bottom]} \times 100$$





تایمر یا کانتر (Timer/Counter)

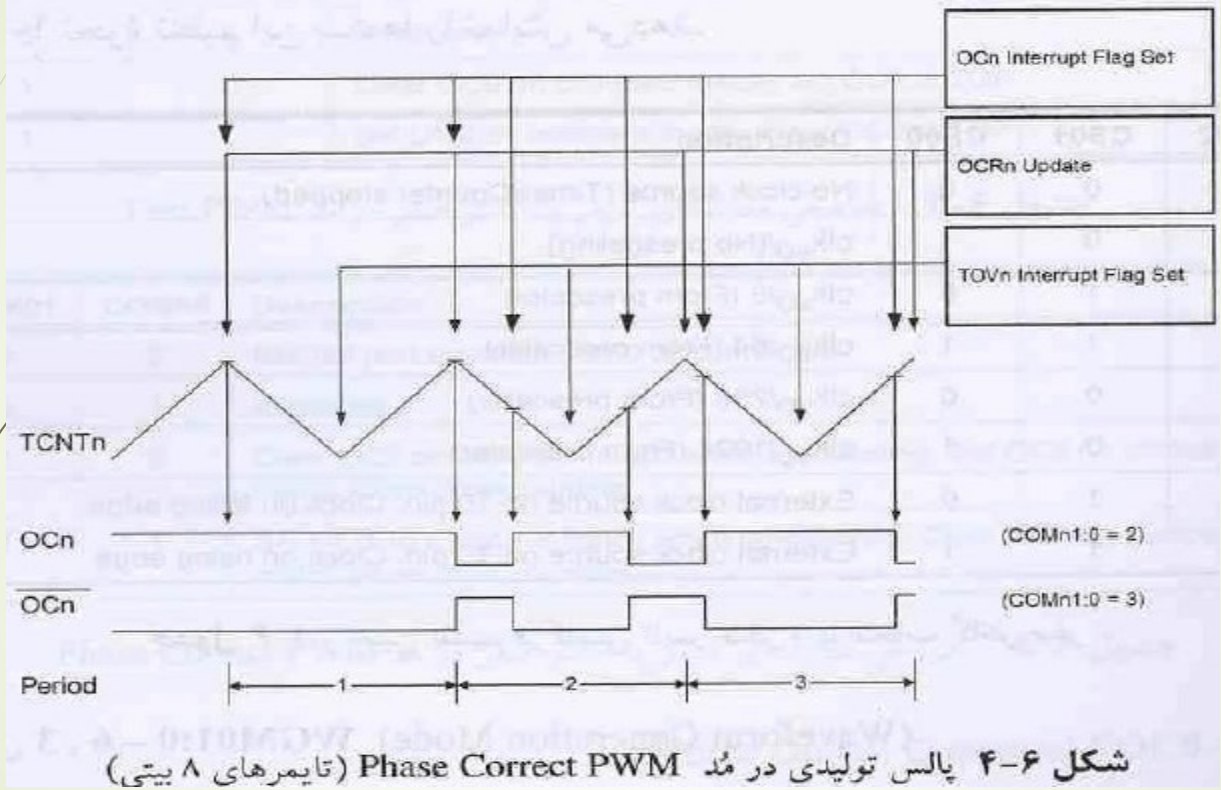
۳. مد مدولاسیون عرض پالس صحیح فاز(Phase correct PWM Mode)

در این حالت رجیستر تایمر TCNTn از مقدار اولیه Bottom شروع به شمارش بصورت صعودی می کند و پس از رسیدن به مقدار نهایی یعنی 0xFF (۸بیتی) وtop (۱۶ بیتی) به صورت نزولی شروع به شمارش می کندتا به Bottom برسد به همین دلیل در شکل ۶-۴ سیکل شمارش تایمر را بصورت مثلثی نمایش می دهیم تا مفهوم بهتری داشته باشد.

زمانی که شمارش بصورت صعودی یا نزولی صورت می گیرد مقدار رجیستر تایمر TCNTn با مقدار OCRn مقایسه می شود و در هر بار برابری ،پایه خروجی Ocn تغییر وضعیت می دهد.

در این حالت اصطلاح دو شیبه را بکار می گیریم که فرکانس آن نصف مد fast PWM است.

تایمر یا کانتر (Timer/Counter)



شکل ۶-۴ پالس تولیدی در مُد Phase Correct PWM (تایمرهای ۸ بیتی)

همانطور که در شکل ۶-۴ مشاهده می کنید بعد از هر بار رسیدن مقدار تایمر به MAX شمارش بصورت نزولی تا رسیدن به Bottom ادامه می یابد و پایه خروجی Ocn در موقع برابری مقدار تایمر با OCRn معکوس شده است

تایمر یا کانتر ۸ بیتی صفر

این تایمر دارای چهار مد ذکر شده می باشد . ابتدا رجیسترهای این تایمر را معرفی می کنیم.

رجیستر TCCR0 (Timer/Counter Control Register)

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

بیت های CS02:0-2,1,0 (clock select)

در صورتی که بخواهیم از تایمر استفاده کنیم وظیفه این بیت ها تعیین تقسیم فرکانسی کلاک تایمر صفر می باشد . این بدان معنی است که تایمر می تواند با سرعتی کمتر از سرعت CPU کار کند در ادامه با این قابلیت بیشتر آشنا خواهید شد.

تایمر یا کانتر ۸ بیتی صفر

اما در مد کانتر این بیت ها نوع لبه کلاک خارجی را مشخص می کنند . اگر این بیت ها صفر باشند تایمر یا کانتر خاموش خواهند شد.

جدول ۶-۱ نحوه تنظیم این بیت ها را نمایش می دهد.

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

جدول ۶-۱ تعیین تقسیم فرکانسی تایمر صفر و یا انتخاب کانتر صفر

تایمر یا کانتر ۸ بیتی صفر

بیت های 3, 6- WGM01:0 (waveform generation mode)

توسط این این بیت ها مد عملکرد تایمر را تعیین می کنیم . در خصوص مد ها ابتدای فصل صحبت کردیم.
طبق جدول ۶-۲ می توان یکی از چهار مد را انتخاب کرد.

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

جدول ۶-۲ انتخاب یکی از چهار مُد برای تایمر یا کانتر صفر

تایمر یا کانتر ۸ بیتی صفر

بیت های 4, 5- COM01:0(compare match output mode)

در صورتی که از یکی از مد های Normal و یا CTC استفاده نمائیم تنظیم و عملکرد این دو بیت به صورت جدول ۶-۳ می باشد. البته باید پایه خروجی مقایسه ای تایمر صفر یعنی پایه OC0 در حالت خروجی پیکربندی شود.

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

جدول ۶-۳ تنظیم خروجی تایمر یا کانتر صفر در مُد CTC و مُد Normal

تایمر یا کانتر ۸ بیتی صفر

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at TOP
1	1	Set OC0 on compare match, clear OC0 at TOP

جدول ۶-۴ خروجی مقایسه‌ای تایمر یا کانتر صفر در مُد Fast PWM

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.

جدول ۶-۵ خروجی مقایسه‌ای تایمر یا کانتر صفر در مُد Phase Correct PWM

تایمر یا کانتر ۸ بیتی صفر

بیت های 7- FOC0 (force output compare)

از این بیت فقط زمانی می توان استفاده نمود که تایمر در مد PWM نباشد . با یک کردن این بیت یک مقایسه اجباری ایجاد می کنیم و بسته به نوع تنظیم خروجی مقایسه ای که توسط بیت های com01:0 تعیین کرده ایم پایه خروجی OC0 تغییر حالت می دهد این بیت در کاربرد های خاص مورد استفاده قرار می گیرد و در حالت عادی آن را صفر می کنیم.

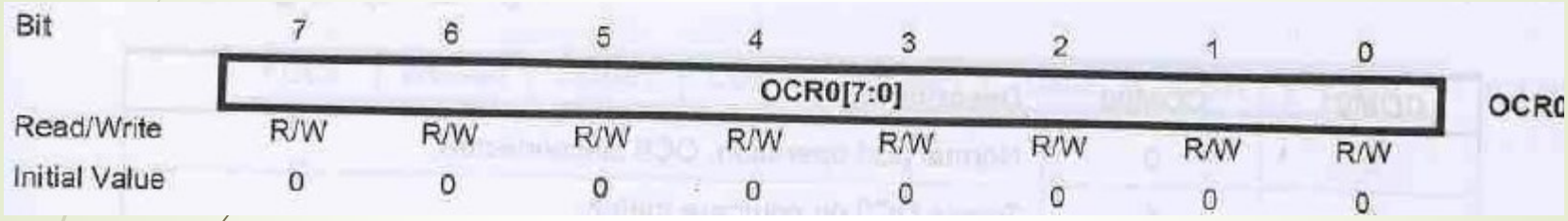
رجیستر **TCNT0** (timer/counter register)

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

رجیستر TCNT0 به طور مستقیم قابل دسترسی است و محتوی ۸ بیتی تایمر یا کانتر صفر را در خود جای می دهد.

تایمر یا کانتر ۸ بیتی صفر

رجیستر **OCR0** (output compare register)



Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
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	

محتوی این رجیستر مقدار TOP را در مد CTC و مبنای مقایسه را در مد های PWM تعیین می کند.

تایمر یا کانتر ۸ بیتی صفر

رجیستر **TIMSK** (timer / counter interrupt mask register)

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	

بیت 1- OCIE (timer /counter output compare match interrupt enable)

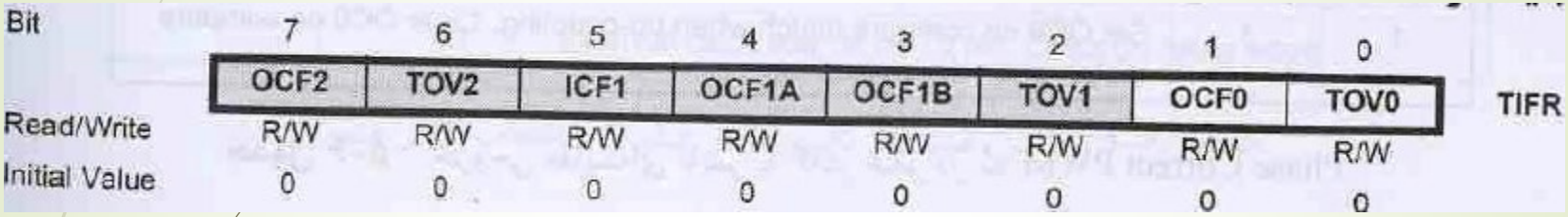
اگر در این بیت یک نوشته شود و بیت I در رجیستر وضعیت یک باشد یعنی وقفه همگانی فعال باشد و تطبیق مقایسه ای تایمر یا کانتر صفر صورت گیرد وقفه مربوط به مقایسه اجرا می شود.

بیت TOIE0-0 (timer / counter overflow interrupt enable)

اگر این بیت را یک کنیم وقفه سرریز تایمر صفر فعال می گردد و در صورتی که وقفه همگانی فعال باشد و محتوی تایمر از حداکثر خود سرریز کند وقفه سریز تایمر صفر اجرا می شود.

تایمر یا کانتر ۸ بیتی صفر

رجیستر **TIFR** (timer / counter interrupt flag register)



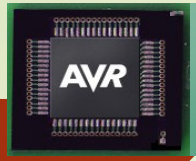
Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت 1- OCF0 (output compare flag 0)

زمانی که رجیستر مقایسه ای تایمر و کانتر صفر یعنی OCR0 با محتوی TCNT0 برابر می شود این پرچم یک می شود و در صورت فعال بودن وقفه مقایسه ای ،وقفه اجرا می شود و پس از برگشت از وقفه این بیت توسط سخت افزار پاک گردد.

بیت 0-TOV0 (timer / counter overflow flag)

زمانی که محتوی رجیستر تایمر یا کانتر صفر یعنی TCNT0 به حداکثر خود یعنی 0xff می رسد این پرچم یک می شود و در صورت فعال بودن وقفه سرریز تایمر یا کانتر صفر ،وقفه اجرا می شود و پس از برگشت از وقفه این بیت توسط سخت افزار پاک می گردد.



تایمر یا کانتر ۸ بیتی صفر

مثال ٦-١ : برنامه ای بنویسید که با استفاده از تایمر صفر یک موج مربعی 1khz بر روی پایه خروجی PA.0 ایجاد نماید. (کریستال را FOSC=8MHZ)

حل :

مرحله اول : ابتدا باید زمان فرکانس موج مورد نظر را بدست آوریم .

$$Tpulse = \frac{1}{1KHZ} = 1ms = 1000\mu s$$

مرحله دوم : باید فرکانس کاری تایمر صفر را با توجه به کریستال داده شده بدست آوریم.

$$Ftimer0 = \frac{Fosc}{N} = \frac{8MHZ}{64} 0.125MHZ$$

عدد N می تواند توسط بیت های CS02:0 یکی از اعداد ثابت ۱ ،۸ ،۶۴ ، ۲۵۶ و یا ۱۰۲۴ انتخاب شود . در واقع عدد N تقسیم فرکانسی تایمر صفر را تعیین می کند.

تایمر یا کانتر ۸ بیتی صفر

مرحله سوم : مدت زمان شمارش تایمر در هر کلاک را بدست می آوریم.

$$Ttimer0 = \frac{1}{Ftimer0} = \frac{1}{0.125MHZ} = 8\mu s$$

مرحله چهارم : بدست آوردن عدد مورد نظر برای مقدار دهی اولیه به رجیستر تایمر صفر می باشد.

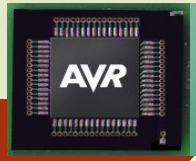
$$TCNT0 = MAXtimer0 - \frac{Tpulse \times \frac{1}{2}}{Ttimer0} = 256 - \frac{Tpulse \times \frac{1}{2}}{Ttimer0} = 256 - 62.5 = 193.5$$

مقدار حداکثر تایمر صفر 0xff برابر 255 دسیمال می باشد اما در رابطه فوق مقدار حداکثر تایمر را 256 قرار دادیم چون یک کلاک نیز جهت پرچم تایمر صفر در نظر گرفتیم .
حاصل رابطه فوق عدد 193.5 می باشد که آن را به 193 گرد کردیم و به مبنای هگزاد یعنی 0xc1 می نویسیم .

تایمر یا کانتر ۸ بیتی صفر

عدد زمان پالس بدست آمده از مرحله اول را در رابطه فوق به دو تقسیم می کنیم زیرا 50% دیگر آن صفر است.
حال بهتر است به توضیح خط به خط برنامه بپردازیم.

```
#include  <mega16.h>
Void  main ( ) {
PORTA.0=0;                    // وضعیت پایه خروجی صفر
DDRA.0=1;                     // پایه PA0 به عنوان خروجی
While  (1) {                  //ایجاد کردن حلقه بی نهایت
TCNT0=0xc1;                   //مقدار دهی اولیه به رجیستر تایمر صفر
TCCR0=0xc3;                   //شروع تایمر صفر با تقسیم فرکانسی ۶۴
while (!(TIFR & 0x01));       // منتظر ماندن برای یک شدن پرچم سرریز تایمر صفر
TCCR0=0x00;                   // توقف تایمر صفر
PORTA.0=! (PORTA.0);          // معکوس کردن خروجی جهت تولید پالس
TIFR=0x01;                    //با نوشتن یک منطقی پرچم را برای مرحله بعدی صفر می کنیم
};
}
```

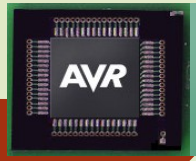


تایمر یا کانتر ۸ بیتی صفر

مثال ۶-۲ : برنامه مثال قبل یعنی تولید موج 1KHZ را این بار توسط روش وقفه انجام دهد. (کریستال را Fosc=8MHZ در نظر بگیرید)

حل : محاسبات برای مقدار دهی به تایمر صفر شبیه مثال ۶-۱ می باشد.

```
#include  <mega16.h>
Interrupt [TIM0_OVF] void timer0_ovf_isr (void) {          //تابع وقفه تایمر
TCCR0=0x00;                                          // توقف تایمر صفر
TCNT0=0xC1;                                    //مقدار دهی اولیه به رجیستر تایمر صفر
PORTA.0=! (PORTA.0);                          // معکوس شدن پایه خروجی جهت تولید موج
TCCR0=0x03;                                   //شروع تایمر صفر با تقسیم فرکانسی ۶۴
Void  main ( ) {
PORTA.0=0;                                 // وضعیت اولیه  پایه خروجی صفر
DDRA.0=1;                                        // پایه PA0 به عنوان خروجی
TCNT0=0xC1;                                  //مقدار دهی اولیه به رجیستر تایمر صفر
TIMSK=0x01;                            //فعال کردن وقفه سرریز تایمر صفر
TCCR0=0x03;                                   //شروع تایمر صفر با تقسیم فرکانسی ۶۴
#asm ("sei"                                    // فعال کردن وقفه کلی
While  (1);                                   // حلقه بی نهایت برنامه
}
```



تایمر یا کانتر ۸ بیتی صفر

مثال ٦-٣: برنامه ای بنویسید که توسط تایمر صفر ،با دو کلید فشاری بتوان یک موج PWM با قابلیت تنظیم dute cycle و زمان تناوب 20 میلی ثانیه تولید نماید. (کریستال را FOSC=8MHZ)

حل : از مد Fast PWM استفاده می کنیم و زمان کل پالس را خود صورت مسئله داده است.

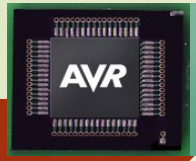
$$Tpluse = 20ms = 20000\mu s$$

$$Ftimer0 = \frac{Fosc}{N} = \frac{8MHZ}{1024} = 7.813KHZ$$

$$Ttimer0 = \frac{1}{Ftimer0} = \frac{1}{7.813KHZ} = 127.99\mu s$$

$$TCNT0 = MAXtimer0 - \frac{Tpulse}{Ttimer0} = 256 - \frac{20000\mu S}{127.99\mu S} = 256 - 156.26 = 99.74 \approx 100$$

$$\rightarrow Bottom = 100$$



تایمر یا کانتر ۸ بیتی صفر

توجه داشته باشید موقع استفاده از فرمول فرکانس خروجی در مد fast PWM اگر مقدار Bottom صفر نباشد باید مقدار Bottom را از TOP+1 تایمر کم کنیم و مقدار OCR0 نباید از Bottom کمتر باشد.

$$FocnPWM = \frac{Fosc}{N \times 256}$$

$$FocnPWM = \frac{Fosc}{N \times [(TOP + 1) - Bottom]} = \frac{8 \times 10^6}{1024 \times [(255 + 1) - 100]} = 50HZ$$

برای محاسبه dute cycle باید مقدار Bottom را نیز در نظر بگیریم به طور نمونه در این مثال مقدار پیش فرض برای پیش فرض OCR0=180 در نظر گرفته شده است.

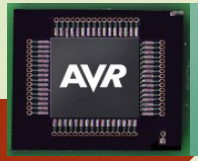
$$Dute\ Cycle = \frac{OCRn - Bottom}{[(TOP + 1) - Bottom]} \times 100 = \frac{(180 - 100)}{[(255 + 1) - 100]} \times 100 = \%51.25$$

تایمر یا کانتر ۸ بیتی صفر

اگر بخواهیم طوری طراحی کنیم که مقدار Bottom را صفر در نظر بگیریم برای تولید موج با فرکانس 50HZ باید مقدار N=625 انتخاب شود که استاندارد نیست و غیر قابل قبول است.

$$FocnPWM = \frac{Fosc}{N \times 256} \rightarrow 50HZ = \frac{8 \times 10^6}{N \times 256} \rightarrow N = 625$$

```
#include <mega16.h>
#define up PINA.0                    // معرفی کلمه up به جای PINA.0
#define down PINA.1                  // معرفی کلمه up به جای PINA.0
Unsigned int x=180;                  // معرفی متغییر کلی ۱۶ بیتی با نام x
interrupt [TIM0_OVF] void timer0_ovf_isr(void) { تابع وقفه
        //تایمر
TCCR0=0x00;                          //توقف تایمر صفر
TCNT0=100;                           // مقدار دهی اولیه به تایمر ،تعیین Bottom
  OCR0=x;                            //به روز سازی مقدار top
TCCR0=0x7D;                          // شروع تایمر در مد fast PWM با تقسیم فرکانسی 1024
}
```



تایمر یا کانتر ۸ بیتی صفر

```
Void  main () {
PORTA=0x03;                  // PA1,PA0 فعال کردن مقاومت pull-up درونی پایه های
DDRA=0x00;                     //تعیین پورت A به عنوان ورودی
PORTB=0x00;                    //وضعیت پایه خروجی OC0 را در ابتدا صفر قرار می دهیم
DDRB=0x08;                   //تعیین پایه  PB3(OC0) را در ابتدا قرار می دهیم
TCNT0=100;                  // Bottom مقدار دهی اولیه به تایمر ،تعیین
OCR0=180;                    //مقدار top را در ابتدا طوری قرار دادیم تا زمان وظیفه 51.25% باشد
 TIMSK=0x01;                // فعال کردن وقفه سرریز تایمر صفر
TCCR0=0x7D;                 // شروع تایمر در مد fast PWM با تقسیم فرکانسی 1024
 #asm ("sei")                 // فعال کردن وقفه کلی
While  (1) {
  If (down ==0) {            //تست کلید متصل به پایه PA1 برای صفر شدن
If (x>=110 && x<250) x+=10;                    // افزایش ۱۰ پله ای متغیر x
While (down == 0);           // تست برای رها شدن کلید فشرده شده
}
 If (up == 0) {              //تست کلید متصل به پایه PA0 برای صفر شدن
If (x!=110)  x-=10 ;                          //کاهش ۱۰ پله ای متغییر x
   While (up == 0);   }   };   }          // تست برای رها شدن کلید فشرده شده
```

تایمر یا کانتر ۱۶بیتی یک

این تایمر از نظر عملکرد ،تمام خصوصیات تایمر صفر را دارد با این تفاوت که ۱۶ بیتی است و مقدار TOP آن در مد های PWM قابل تنظیم است و ضمنا این تایمر دارای ویژگی مهم captuer می باشد ابتدا کمی در مورد واحد تسخیر کننده (capture unit) می پردازیم.

واحد ورودی تسخیر کننده (Input Capture Unit)

تایمر یک می تواند توسط واحد capture سیگنالهای خارجی را تشخیص دهد و از زمان جاری با سرعت بالا نمونه بگیرد . این واحد دارای رجیسترهای ۸ بیتی ICRIL وICRIH می باشد که با هم یک محتوی ۱۶ بیتی را تشکیل می دهند.

به عبارت ساده تر تایمر را که روشن می کنیم با توجه به سرعت کلاک خود شروع به شمارش می کند حال اگر یک سیگنال خارجی در لحظه t به ورودی پایه ICP اعمال گردد محتوی رجیستر تایمر یک ،یعنی TCNT1 در همان لحظه t به رجیستر ICR1کپی می شود.

از کاربردهای واحد capture می توان به اندازه گیری dute cycle اشاره کرد. برخی المانها پارامتر duty cycle را تغیر می دهند.

از کاربرد دیگر این واحد می توان به اندازه گیری زمان برگشت پالس با استفاده از سنسور آلتراسونیک اشاره کرد.

تایمر یا کانتر ۱۶بیتی یک

این سنسور به این صورت عمل می کند که سیگنال را با سرعت صوت انتشار می دهد و مدت زمان رفت و برگشت این سیگنال رابطه مستقیمی با طول موج دارد.
بلوک دیاگرام واح capture در شکل ۶-۵ نمایش داده شده است.

رجیستر های تایمر یا کانتر یک

رجیستر TCCR1A (timer / counter control register A)

بیت های 1,0- WGM11,WGM10 (waveform generation mode)
بیت های 0,1 این رجیستر به همراه بیت های 3و4 واقع در رجیستر TCCR1B جهت تعیین مد عملکرد تایمر یا کانتر یک استفاده می شوند که در جدول ۶-۶ ملاحظه می کنید.

بیت های 3 -FOC1B , 2 و FOC1A (force output mode for chanel)
تایمر یک ۱۶ بیتی است و برای مد تطبیق مقایسه از دو قسمت مجزا ۱۶ بیتی تشکیل شده است . در زمانی که مد PWM انتخاب نشده باشد با یک شدن این بیت ها توسط نرم افزار می توان یک تطبیق مقایسه اجباری ایجاد کردکه در صورتی که خروجی مقایسه ای فعال باشند تغییر وضعیت خواهند داد.

تایمر یا کانتر ۱۶بیتی یک

بیت های 6, 7- COM1A1:0 (compare output mode for channel A)
بیت های 4, 5- COM1B1:0 (compare output mode for channel B)

ترکیبات این بیت ها برای تنظیمات خروجی تایمر یا کانتر یک ، یعنی پایه های بیرونی OC1A وoc1b می باشد . در مد های مختلف ترکیبات این بیت ها حالت خروجی ها را تعیین می کند.
جدول ۶–۷ عملکرد این بیت ها را برای تنظیم پایه های خروجی در مد غیر PWM نشان می دهد.
جدول ۶–۸ عملکرد این بیت ها را برای تنظیم پایه های خروجی در مد fast PWM نشان می دهد.
جدول ۶–۹ عملکرد این بیت ها را برای تنظیم پایه های خروجی در مد phase correct PWM نشان می دهد.

تایمر یا کانتر ۱۶بیتی یک

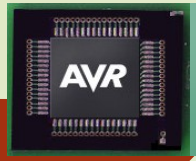
رجیستر **TCCR1B**(timer / vounter1 control register B)

بیت های CS12:0-2,1,0(clock select)

این بیت مطابق جدول ۶-۱۰ وظیفه تعیین فرکانسی کلاک تایمر یک و همچنین خاموش بودن تایمر و انتخاب تایمر یا کانتر را بر عهده دارند.

CS12	CS11	CS10	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

جدول ۶-۱۰ تعیین تقسیم فرکانسی تایمر یک و یا انتخاب کانتر یک



تایمر یا کانتر ۱۶بیتی یک

بیت های WGM13:2-4,3 (wafeform generation mode)

در توضیحات رجیستر TCCR1A درباره ایت بیت ها و عملکرد آنها توضیح دادیم.

بیت 5- -- (Rserved bit)

این بیت برای تولید های بعدی میکروکنترلر رزرو شده است و باید همیشه در آن صفر قرار دهیم.

بیت ICES1-6 (Input capture edge select)

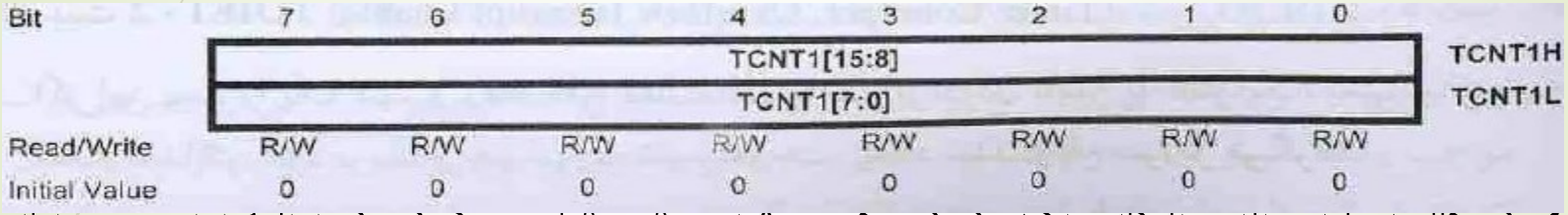
این بیت نحو ه ی تحریک شدن واحد capture را تعیین می کند . اگر این بیت صفر باشد یک لبه پایین رونده سیگنال و اگر یک باشد یک لبه بالا رونده سیگنال می تواند واحد capture را تحریک کند. واحد capture با سطح صفر و سطح یک پالس عمل نمی کند.

بیت ICNC1-7 (input capture noise canceler)

با یک کردن این بیت سیگنال اعمال شده به پایه ورودی ICP از یک فیلتر حذف نویز دیجیتال عبور می کند این فیلتر در چهار کلاک اسیلاتور از سیگنال نمونه گیری می کند و در صورت برابری نمونه ها خروجی فیلتر دیجیتال ،واحد capture را تحریک می کند.

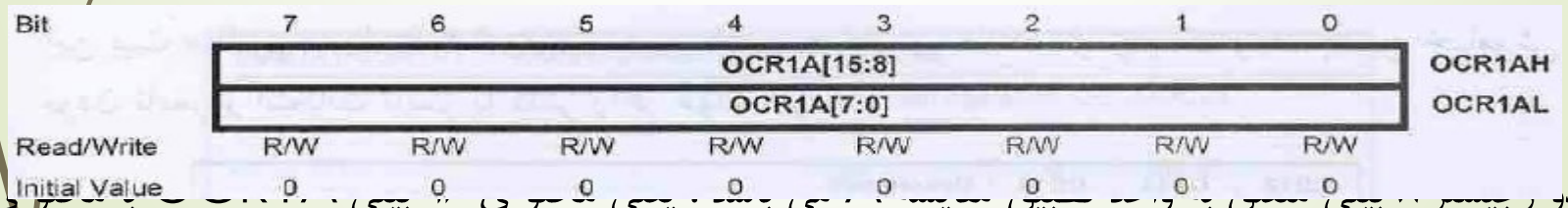
تایمر یا کانتر ۱۶بیتی یک

رجیستر های TCNT1Hو TCNT1L



محتو ی در حال سمارس تایمر یا کانتر یک ، در دو رجیستر ۸ بیتی فوق قرار می گیرد و در برنامه نویسی می توانیم با نام TCNT1 محتو ی تایمر یا کانتر یک را به صورت ۱۶ بیتی بخوانیم.

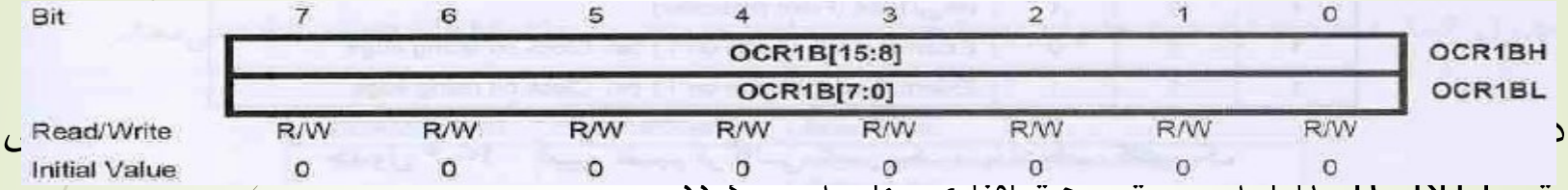
رجیستر های OCR1AHو OCR1AL



این دو رجیستر ۸ بیتی متصل به واحد مقایسه کننده ۱۶ بیتی هستند و مقدار ۱۶ بیتی موجود در OCR1A با مقدار ۱۶ بیتی TCNT1 مداما به صورت سخت افزاری مقایسه می گردد.

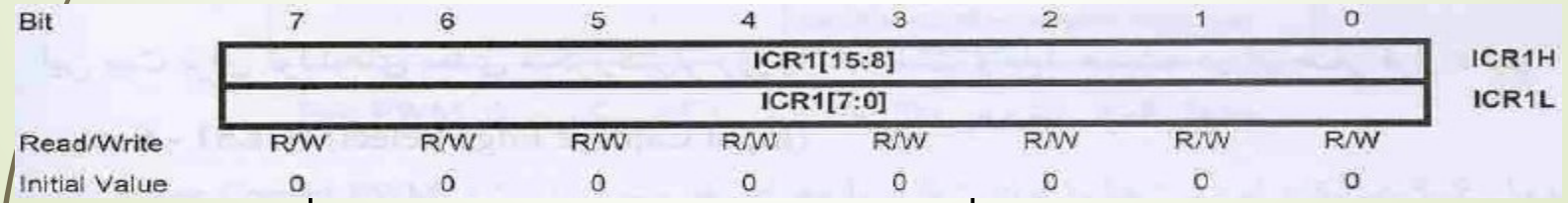
تایمر یا کانتر ۱۶بیتی یک

رجیستر های OCR1BHوOCR1BL



این د
۱۶ بیتی ICNT1 مداما به صورت سخت افزاری مقایسه می گردد.

رجیستر های ورودی تسخیر کننده (input capture register ICR1H and ICR1L)



این دو رجیستر ۸ بیتی محتو ی واحد capture را شامل می شوند . اگر واحد capture توسط خروجی مقایسه کننده آنالوگ یا سیگنال اعمال شده به پایه ICP واحد تسخیر کننده را تحریک کند محتو ی همان لحظه تایمر در این رجیستر ها بروزرسانی می شود.

تایمر یا کانتر ۱۶بیتی یک

رجیستر TIMSK(timer/counter interrupt mask register)

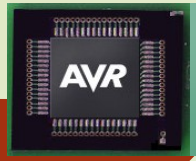
Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت TOIE1-2(timer/counter overflow interrupt enable)

اگر این بیت را یک کنیم و وقفه کلی فعال باشد با سرریز کردن تایمر یا کانتر یک ،یعنی اینکه به مقدار حداکثر خود برسد پرچم سرریز تایمر موجب ایجاد شدن وقفه سرریز می گردد.

بیت OCIE1B -3(timer/counter output compare B match interrupt enable)

اگر این بیت را یک کنیم و وقفه کلی فعال باشد با تطبیق مقایسه واحد B یعنی برابری رجیستر TCNT1 با رجیستر OCR1B پرچم تطبیق واحد B تایمر یا کانتر یک ،موجب ایجاد شدن وقفه تطبیق مقایسه می گردد.



تایمر یا کانتر ۱۶بیتی یک

بیت OCIE1A-4(timer/counter output compare A match interrupt enable)

اگر این بیت را یک کنیم و وقفه کلی فعال باشد با تطبیق مقایسه واحد A یعنی برابری رجیستر TCNT1 با رجیستر OCR1B پرچم تطبیق واحد A تایمر یا کانتر یک ،موجب ایجاد شدن وقفه تطبیق مقایسه می گردد.

بیت TICIE1-5 (timer counter input captuer interrupt enable)

اگر این بیت را یک کنیم و وقفه کلی فعال باشد ، در صورت تحریک شدن واحد capture توسط خروجی مقایسه کننده آنالوگ داخلی یا سیگنال اعمال شده به پایه ICP،وقفه مربوط به واحد تسخیر کننده فعال می گردد و می توان محتوی رجیستر ICR1 را بررسی نمود.

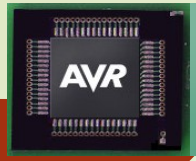
تایمر یا کانتر ۱۶بیتی یک

رجیستر TIFR(timer / counter overflow flag)

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	

بیت TOV1-2 (timer/counter Toverflow flag)

تنظیم این بیت بستگی به تنظیمات بیت های WGM13:0 دارد در مد normal و مد CTC زمانی که سرریز رخ دهد ،این پرچم فعال می شود و می تواند موجب وقفه سرریز تایمر یک شود .زمانی که بردار وقفه اجرا می شود ، در برگشت از وقفه به طور اتوماتیک توسط سخت افار این پرچم پاک می گردد .



تایمر یا کانتر ۱۶بیتی یک

بیت 3- OCF1B (timer / counter1, output compare B match flag)

این پرچم زمانی فعال می گردد که مقدار TCNT1 یا مقدار OCR1B برابر شود . این پرچم می تواند موجب وقفه تطبیق B تایمر یک شود زمانی که بردار وقفه اجرا می شود در برگشت از وقفه به طور اتوماتیک توسط سخت افزار این پرچم پاک می گردد.

بیت 4-OCF1A(timer / counter1, output compare A match flag)

این پرچم زمانی فعال می گردد که مقدار TCNT1 یا مقدار OCR1A برابر شود . این پرچم می تواند موجب وقفه تطبیق A تایمر یک شود زمانی که بردار وقفه اجرا می شود در برگشت از وقفه به طور اتوماتیک توسط سخت افزار این پرچم پاک می گردد.

بیت 5-ICF1(timer /counter ,input capture flag)

زمانی که واحد captuer تحریک گردد این پرچم فعال می شود و موجب ایجاد وقفه تسخیر کننده می گردد و در برگشت از وقفه ،توسط سخت افزار پاک می گردد.

تایمر یا کانتر ۱۶بیتی یک

مثال ٦-٤: برنامه ای بنویسید که یک LED متصل به پایه PC0 هر یک ثانیه یک بار چشمک بزند برای اینکار از تایمر یک استفاده نمایید و به روش polling این کار را انجام دهید. (کریستال را FOSC=4MHZ)

حل :

$$Tpluse = 1s = 1000ms = 1000000\mu s$$

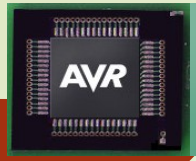
$$Ftimer1 = \frac{Fosc}{N} = \frac{4MHZ}{256} = 0.015625MHZ$$

$$Ttimer1 = \frac{1}{Ftimer1} = \frac{1}{0.015625MHZ} = 64\mu s$$

$$TCNT1 = MAXtimer1 - \frac{Tpulse}{Ttimer1} = 65536 - \frac{1000000\mu S}{64\mu S} = 65536 - 15625 = 49911$$
$$= C2F7H$$

تایمر یا کانتر ۱۶بیتی یک

```
#include <mega16.h>
Void main () {
PORTC.0=0;                  وضعیت ال ای دی ابتدا خاموش است.//
DDRC.0=1;          پایه متصل به ال ای دی را به صورت خروجی تنظیم می کنیم//
TCCR1A=0x00;                  انتخاب مد نرمال تایمر یک //
While (1) {                 ایجاد حلقه بی نهایت //
TCNT1H=0xC2;                  مقدار دهی اولیه به رجیستر بالایی تایمر یک //
TCNT1H=0xC2;                  مقدار دهی اولیه به رجیستر پایینی تایمر یک //
TCCR1B=0x04;          شروع تایمر یک با تقسیم فرکانسی ۲۵۶//
While (!(TIFR & 0x04));       منتظر ماندن برای فعال شدن پرچم سرریز تایمر یک //
TCCR1B=0x00;                  تایمر یک را خاموش می کنیم//
PORTC.0=! (PORTC.0);          پایه متصل به ال ای دی را معکوس می کنیم//
TIFR=0x04;                  پاک کردن پرچم سرریز بصورت نرم افزاری //
};
}
```



تایمر یا کانتر ۱۶بیتی یک

مثال ٦-٤: برنامه ای بنویسید که یک موج با فرکانس 2KHZ با duty cycle %70 سطح منطقی یک بر روی پایه خروجی PC0 ایجاد نماید. (کریستال را =MHZ۸FOSC)

حل : برای تولید چنین موجی ما تایمر یک را طوری تنظیم می کنیم که یک پالس 2MHZ ایجاد نماید اما از تطبیق مقایسه ای A استفاده می کنیم و ۷۰ درصد پالس را یک و ۳۰ درصد آن را صفر می کنیم.
برای این مثال از مد نرمال استفاده کردیم البته می توانستیم از مد Fast PWM نیز استفاده کنیم.

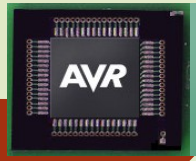
$$Ftimer1 = \frac{Fosc}{N} = \frac{8MHZ}{8} = 1MHZ$$

$$Ttimer1 = \frac{1}{Ftimer1} = \frac{1}{1MHZ} = 1\mu s$$

$$Tpulse = \frac{1}{2KH} = 0.5ms = 500\mu s$$

$$TCNT1 = MAXtimer1 - \frac{Tpulse}{Ttimer1} = 65536 - \frac{500\mu S}{1\mu S} = 65536 - 500 = 65036$$

$$= FE0CH$$



تایمر یا کانتر ۱۶بیتی یک

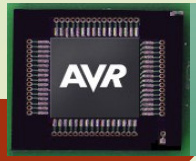
$$Dute\ cycle = D.C = 500\mu S \times \%70 = 350$$

$$OCR1A = MAXtimer1 - \frac{D.C}{Ttimer1} = 65536 - \frac{350\mu S}{1\mu S} = 65536 - 350 = 65186$$

$$= FEA2H$$

```
#include <mega.h>
Interrupt [TIM1_OVF] void timer1_ovf_isr(void) { وقفه تایمر سریز یک//
TCCR1B=0x00; توقف تایمر یک//
TCNT1H=0xfe; مقدار دهی اولیه به رجیستر بالایی تایمر یک//
TCNT1L=0x0C; مقدار دهی پایینی به رجیستر بالایی تایمر یک//
PORTC.0=! (PORTC.0); معکوس کردن خروجی//
TCCR1B=0x02; شروع تایمر با تقسیم فرکانسی ۸//
}
Interrupt [TIM1_COMPA] void timer1_compa_isr (void) // وقفه مقایسه A
{
PORTC.0=! (PORTC.0); معکوس کردن خروجی//
}
```

تایمر یا کانتر ۱۶بیتی یک

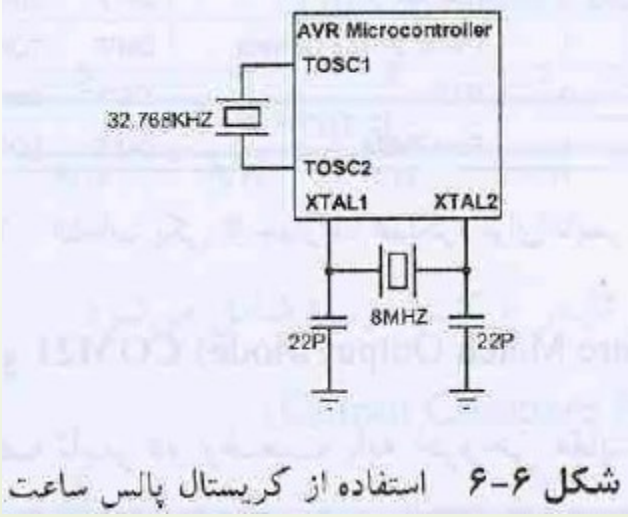



```
Void main () {
PORTC.0=0;                          وضعیت اولیه پایه خروجی//
DDRC.0=1;                           تنظیم پایه PC.0به عنوان خروجی//
TCNT1H=0xFE;                        مقداردهی اولیه به رجیستر بالایی تایمر یک//
TCNT1L=0x0C;                        مقداردهی اولیه به رجیستر پایینی  تایمر یک//
 OCR1AH=0XFE;                       مقدار دهی اولیه به رجیستر بالایی تطبیق مقایسه ای A//
 OCR1AL=0XA2;                       مقدار دهی اولیه به رجیستر پایینی تطبیق مقایسه ای A//
TIMSK=0X14;                         فعال کردن وقفه سرریز و تطبیق مقایسه ای تایمر یک //
TCCR1A=0x00;                        انتخاب مد نرمال //
TCCR1B=0x02;                        شروع تایمر با تقسیم فرکانسی ۸//
#asm (“sei”)                        فعال کردن وقفه  کلی//
While  (1) ;                        حلقه بی نهایت و بیکار برنامه //
}
```

تایمر یا کانتر ۸ بیتی دو

این تایمر شباهت بسیار زیادی به تایمر صفر دارد اما این تایمر دارای یک عملکرد غیر همزمان است و می تواند به طور مجزا با یک اسیلاتور پالس ساعت کار کند به این عملکرد RTCگفته می شود . یعنی وقتی تایمر عملکرد آسنکرون داشته باشد پالس مورد نیاز خود را از کریستال پالس ساعت یعنی 32.768KHZ تامین می کند و می تواند به طور دقیق زمان واقعی را اندازه گیری کند .

شکل ۶-۶ نحوه ی اتصال کریستال پالس ساعت به ورودی تایمر دو را نشان می دهد همانطور که می بینید این کریستال، مجزا از کریستال کلاک سیستم می باشد و همچنین کریستال پالس ساعت بهینه شده و نیازی به قرار دادن خازن های بالانس نمی باشد.



شکل ۶-۶ استفاده از کریستال پالس ساعت

تایمر یا کانتر ۸ بیتی دو

رجیستر های تایمر یا کانتر دو

رجیستر TCCR2 (timer/counter control register)

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

تایمر یا کانتر ۸ بیتی دو

بیت های CS22:0-2,1,0 (clock select)

ترکیبات این سه بیت می تواند تایمر دو را خاموش کند یا با یک تقسیم فرکانسی از کلاک سیستم تایمر را روشن کند . عملکرد مختلف این بیت ها مطابق جدول ۶-۱۱ است.

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)
1	0	1	clk_{T2S}/128 (From prescaler)
1	1	0	clk_{T2S}/256 (From prescaler)
1	1	1	clk_{T2S}/1024 (From prescaler)

جدول ۶-۱۱ تعیین تقسیم فرکانسی تایمر یا کانتر دو

تایمر یا کانتر ۸ بیتی دو

بیت های WGM21وWGM20 (wafeform generation mode)

توسط این دو بیت می توان مد عملکرد را انتخاب نمود. جدول ۶-۱۲ نحوی انتخاب مد تایمر دو را نمایش می دهد.

Mode	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Mode of Operation	TOP	Update of OCR2	TOV2 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR2	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

جدول ۶-۱۲ انتخاب یکی از چهار مُد عملکرد برای تایمر یا کانتر دو

تایمر یا کانتر ۸ بیتی دو

بیت های 4 ,5- COM20وCOM21 (compare match output mode)

این دو بیت در مد های مختلف تایمر دو وضعیت پایه خروجی مقایسه ای OC2 را تعیین می کند.

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Toggle OC2 on compare match
1	0	Clear OC2 on compare match
1	1	Set OC2 on compare match

جدول ۶-۱۳ خروجی مقایسه‌ای تایمر یا کانتر دو، در مُد غیر PWM

تایمر یا کانتر ۸ بیتی دو

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Reserved
1	0	Clear OC2 on compare match, set OC2 at TOP
1	1	Set OC2 on compare match, clear OC2 at TOP

جدول ۶-۱۴ خروجی مقایسه‌ای تایمر یا کانتر دو، در مُد Fast PWM

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Reserved
1	0	Clear OC2 on compare match when up-counting. Set OC2 on compare match when downcounting.
1	1	Set OC2 on compare match when up-counting. Clear OC2 on compare match when downcounting.

جدول ۶-۱۵ خروجی مقایسه‌ای تایمر یا کانتر دو، در مُد Phase Correct PWM

تایمر یا کانتر ۸ بیتی دو

بیت 7- FOC2(force output compare)

کاربرد این بیت در مد های غیر PWM می باشد . با یک شدن این بیت یک سیگنال به واحد تطبیق مقایسه فرستاده می شود و به طور اجباری خروجی مقایسه ای با توجه به تنظیمات انجام شده ،تغییر وضعیت می دهد.

رجیستر TCNT2(timer/counter register)

Bit	7	6	5	4	3	2	1	0	
	TCNT2[7:0]								TCNT2
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	

این رجیستر محتوی شمارش تایمر یا کانتر دو را شامل می شود.

تایمر یا کانتر ۸ بیتی دو

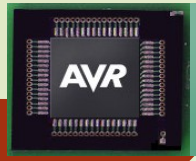
رجیستر OCR2(output compare register)

Bit	7	6	5	4	3	2	1	0	
	OCR2[7:0]								OCR2
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	

این رجیستر محتوی شمارش مقایسه ای تایمر یا کانتر دو را شامل می شود.

رجیستر ASSR(asynchronous status register)

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	ASSR
Read/Write	R	R	R	R	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	



تایمر یا کانتر ۸ بیتی دو

بیت های TCR2UB -0 (timer /counter control register2 update busy)

زمانی که تایمر دو بصوررت غیر همزمان کار می کند اگر در رجیستر TCCR2 مقداری نوشته شود این بیت یک می گردد و زمانی که رجیستر TCCR2 توسط رجیستر ذخیره موقت به روز می شود،این بیت توسط سخت افزار پاک می شود.

بیت های OCR2UB-1 (output compare register2 update busy)

زمانی که تایمر دو بصوررت غیر همزمان کار می کند اگر در رجیستر OCR2 مقداری نوشته شود این بیت یک می گردد و زمانی که رجیستر OCR2 توسط رجیستر ذخیره موقت به روز می شود،این بیت توسط سخت افزار پاک می شود.

بیت های TCN2UB-1 (timer/compare2 update busy)

زمانی که تایمر دو بصوررت غیر همزمان کار می کند اگر در رجیستر TCNT2مقداری نوشته شود این بیت یک می گردد و زمانی که رجیستر TCNT2توسط رجیستر ذخیره موقت به روز می شود.

تایمر یا کانتر ۸ بیتی دو

بیت AS2-3 (asynchronous timer counter2)

زمانی که این بیت صفر باشد تایمر دو کلاک مورد نیاز خود را از کلاک سیستم دریافت می کند و زمانی که این بیت را یک کنیم تایمر به صورت RTCعمل می کند و کلاک خود را از کریستال پالس ساعت متصل به پایه های TOSC1وTOSC2 دریافت می کند.

رجیستر TIMSK(timer /counter interrupt mask register)

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	

بیت TOIE2-6(timer/counter2 overflow interrupt enable)

با یک کردن این بیت وقفه سرریز تایمر با کانتر دو فعال می گردد و در صورتی که وقفه کلی نیز فعال باشد با سرریز مقدار تایمر از 0xff وقفه مربوطه به آن اجرا می شود.

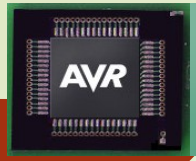
تایمر یا کانتر ۸ بیتی دو

بیت OCIE2-7(timer/counter2 output compare match interrupt enable)

با یک کردن این بیت وقفه تطبیق مقایسه تایمر با کانتر دو فعال می گردد و در صورتی که وقفه کلی نیز فعال باشد در صورت برابر شدن TCNT2 مقدار OCR2 وقفه تطبیق مقایسه تایمر یا کانتر دو اجرا می گردد.

رجیستر TIFR(timer /counter interrupt flag register)

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	



تایمر یا کانتر ۸ بیتی دو

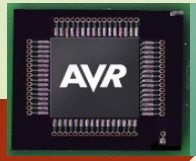
بیت TOV2-6(timer/counter2 overflow flag)

زمانی که محتوی رجیستر TCNT2 از حداکثر خود یعنی 0xFF سرریز می کند ،پرچم TOV2 فعال می گردد و در صورت فعال بودن وقفه سرریز تایمر یا کانتر دو،وقفه ایجاد می گردد و بعد از اجرا وقفه ،سخت افزار این پرچم را پاک می کند و به روش polling باید بصورت نرم افزاری توسط کاربر با نوشتن یک منطقی در این پرچم ،پاک گردد.

بیت OCF2-7(output compare flag 2)

اگر وقفه تطبیق مقایسه تایمر یا کانتر دو فعال باشد و محتوی TCNT2 با مقدار OCR2 برابر شود پرچم OCF2 فعال می گردد و وقفه تطبیق مقایسه تایمر یا کانتر دو اجرا می گردد.

بعد از برگشت وقفه ،این پرچم توسط سخت افزار پاک می گردد و به روش polling باید به صورت نرم افزاری توسط کاربر با نوشتن یک منطقی در این پرچم پاک گردد.



تایمر یا کانتر ۸ بیتی دو

مثال ٦-٧ : برنامه ای بنویسید که بتواند مقدار ساعت ،دقیقه و ثانیه را محاسبه کرده و در سه متغیر ذخیره نماید. برای این کار از کریستال پالس ساعت (32.768khz)استفاده کنید.یعنی تایمر دو را بصورت غیر همزمان بکار بگیرید.

حل: چون در این مثال از کریستال مجزا RTC استفاده شده است بنابراین در رابطه زیر مقدار این کریستال را به مقدار N (تقسیم فرکانسی) تقسیم می کنیم تا فرکانس کاری تایمر مشخص گردد.

$$Ftimer2 = \frac{Fcrystal}{N} = \frac{32.768KHZ}{128} = 0.256KHZ$$

$$Ttimer2 = \frac{1}{Ftimer2} = \frac{1}{0.256KHZ} = 3.90625\mu s$$

در رابطه زیر مدت زمان سرریز تایمر ۲ را محاسبه می کنیم.

$$Tovf2 = MAXtimer2 \times Ttimer2 = 256 \times 3.90625ms = 1000ms = 1\mu S$$

تایمر یا کانتر ۸ بیتی دو

```
#include <mega16.h>
Unsigned char oclock=0,minute=0,second=0; معرفی متغیر های
                برنامه//
Interrupt [TIM2_OVF] void timer2_ovf_isr (void) { تابع وقفه تایمر
               //۲
Second++;                                  افزایش متغیر ثانیه شمار به اندازه یک واحد//
If (second==60) {                          اگر ثانیه شمار برابر عدد ۶۰ شود//
Second=0;                                  ثانیه شمار صفر گردد//
Minute++;                                  دقیقه شمار به اندازه یک واحد افزایش یابد//
If (minute==60) {                          اگر دقیقه شمار برابر عدد ۶۰ شود//
Minute=0;                                  دقیقه شمار صفر می گردد//
Oclock++;                                  ساعت شمار را به اندازه یک واحد افزایش دهیم//
If (clock==24) {                           اگر ساعت شمار برابر عدد ۲۴ شود//
Oclock=0;                                  ساعت شمار را صفر می کنیم//
}
}
}
}
```

تایمر یا کانتر ۸ بیتی دو

```
Void  main () {
ASSR=0x08;
TCCR2=0x05;
TCNT2=0x00;
TIMSK=0x40;
#asm (“sei”)
While  (1) ;
}
```

انتخاب عملکرد آسنکرون تایمر دو//
شروع تایمر با تقسیم فرکانسی ۱۲۸//
مقدار اولیه تایمر صفر است//
فعال کردن وقفه سرریز تایمر دو//
فعال کردن وقفه کلی//
حلقه بی نهایت و بیکار برنامه//