تایمر – کانتر

Timer / Counter

میکروکنترلرها دارای یک واحد جهت شمارش و زمان سنجی موسوم به تایمر کانتر هستند که از تایمر جهت زمان سنجی و از کانتر جهت شمارش استفاده می شود . کانتر در واقع یک شمارنده می باشد که با هر پالسی که دریافت می کند مقدار رجیسترش یک واحد افزوده می شود . این پالس از خارج اعمال می شود و می تواند توسط یک کلید یا هر چیز دیگری در لبه بالارونده یا پایین رونده صورت گیرد .
میکروکنترلرهای خانواده mega8 و mega16 و mega32 دارای 3 تایمر کانتر هستند که دوتای آنها 8 بیتی و یکی از آنها 16 بیتی می باشد . تایمر کانتر صفر و یک 8 بیتی و تایمر کانتر 2 ، 16 بیتی می باشد .

Bottom

کمترین مقدار رجیستر است و شمارش از این عدد شروع می شود و می تواند صفر یا عدد دیگری باشد این مقدار را می توان با مقدار دهی اولیه به رجیستر های تایمر کانتر تعیین کرد .

Max

حداکثر مقدار یک رجیستر است که در رجیستر 8 بیتی 0xff و در رجیستر 16 بیتی 0xffff می باشد .

Top

در صورتی که مقدار شمارش در تایمر یا کانتر به عدد Top برسد شمارش از صفر یا عدد Bottom ی که خودمان تعییین کرده ایم آغاز می شود . در مد نرمال عددTop همان عدد Max می باشد اما در مد های دیگر می تواند می تواند دلخواه باشد .

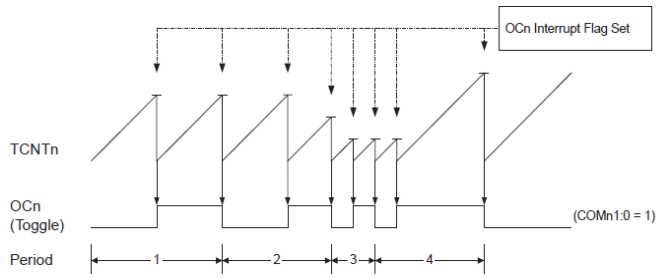
مد ها در تایمر کانتر

الف - مد نرمال

در این مد تایمر کانتر به صورت معمولی شمارش می کند هرگاه به مقدار Top برسد در صورت فعال بودن وقفه تایمر کانتر، پرچم Over Flow یا OVF یا همان سرریز یک می شود و در صورت فعال بودن وقفه cpu یه داخل وقفه سرریز می رود و برنامه را اجرا می کند . پس در این مد می توان تعداد شمارش یا زمان سرریز را تغییر داد . مثلا اگر عدد Bottom مقدار 200 در نظر گرفته شود شمارش از 200 شروع شده و هرگاهبه 255 برسد سرریز اتفاق می افتد پس بعد از هر 55 شمارش سرریز اتفاق می افتد .

ب– مد CTC (clear timer on compare)

رجیستر OCRn مقایسه می شود و در صورت برابری اولا محتوای تایمر کانتر صفر شده و دوم اینکه پایه خروجی OCn مطابق تنظیمات تغییر می کند پس در این حالت فرکانس را می توان تغییر داد .



شکل7-1

در واقع مقدار رجیستر OCR نقش مقدار TOP را بازی می کند . همانگونه که در شکل مشاهده می شود شمارش تا زمان برابری ادامه پیدا می کند و در صورت برابری از صفر دوباره شروع می شود .مقدار فرکانس خروجی در این مد از رابطه زیر بدست می آید :

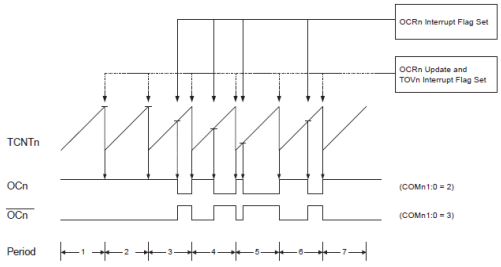
مقدار N می تواند 1 ، 8 ، 64 ،256 یا 1024 باشد .

16 بیتی $f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$

8 بیتی $f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$

ج – مد Fast PWM

بر خلاف مد CTC در این مد T/C در لحظه برابری صفر نمی شود بلکه شمارش ادامه پیدا می کند تا به مقدار TOP برسیم . در این مد با شروع شمارش هرگاه محتوای T/C با محتوای رجیستر OCR برابر شود خروجی OC مطابق تنظیمات Inverted یا Non Inverted تغییر خواهد کرد و در ان حالت می ماند تا زمانی که تایمر به مقدار بیشینه خود برسد در این زمان تایمر به حالت اولیه بر می گردد . پس در این مد مقدار فرکانس تغییر نمی کند بلکه می توان مقدار موثر پالس (زمان یک بودن) را تغییر داده و در مواردی همچون کنترل دور موتور ها یا میزان نور لامپ از مد استفاده کرد .



شکل7-2

فرکانس خروجی از رابطه زیر بدست می آید :

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

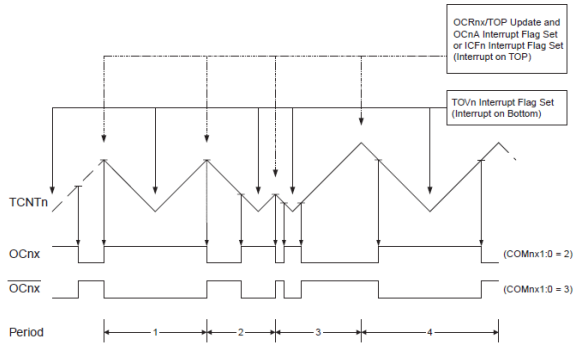
مقدار N همانند حالت قبل می باشد .

حالت Inverted

در این حالت خروجی OC یک است تا زمانی که برابری اتفاق افتد در این لحظه خروجی OC صفر می شود و این وضعیت ادامه دارد تا T/C به حداکثر مقدار خود (Top) برسد . در این لحظه خروجی OC دوباره یک شده و کار از نو شروع می شود (حالت Non Inverted عکس این موضوع است) .

مد P.C.PWM (Phase correct PWM)

در این مد T/C ابتدا صعودی می شمارد و وقتی به مقدار TOP برسد به جای آنکه شمارش از صفر شروع شود نزولی شمارش می کند تا به مقدار صفر برسد و در این لحظه کار از اول شروع می شود و در حالت Inverted خروجی OC یک می باشد تا در زمان صعودی شمار برابری اتفاق بیافتد . در این لحظه خروجی OC صفر می گردد و این وضعیت ادامه دارد تا در حالت نزولی شمار برابری دوباره اتفاق بیافتد و خروجی OC دوباره یک گردد .



شکل3-7

فرکانس خروجی در حالت هشت بیتی از رابطه $f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$ بدست می آید و در حالت 16 بیتی از رابطه $f_{OCnxPFCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$ بدست می آید .

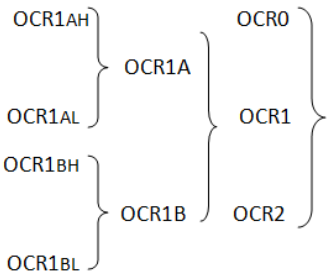
رجیستر های مهم مرتبط با تایمر کانتر

1- رجیستر TCNTn (Timer Counter Register)

این رجیستر در هر لحظه مقدار T/C را نشان می دهد به بیانی محتوای تایمر کانتر در هر لحظه در این رجیستر ریخته می شود . تعداد بیت های این رجیستر در تایمر کانترها متفاوت می باشد و به صورت زیر می باشد :

رجیستر TCNTn (timer counter register) { 1- TCNT0 (8 بیتی) ; 2- TCNT1 (16 بیتی) { TCNT1L ; TCNT1L } ; 3- TCNT2 (8 بیتی) }

2- رجیستر OCRn (output compare register)



این رجیستر یک رجیستر مقایسه ای می باشد و در هر نقطه محتوای آن را با محتوای

رجیستر TCNTn مقایسه کرده و در صورت برابری (compare match) بر اساس تنظیمات انجام شده کار ویژه ای را انجام می دهد .

پین OCn

چند پین خروجی بر روی میکرو می باشد که وضعیت آنها متناسب با تنظیمات در زمان تطابق مقایسه ای تغییر می کند که شامل 4 پین OC0 ، OC1A ، OC1B ، OC2 می باشد.

3- رجیستر TCCRn (timer +counter control register)

تنظیمات کنترلی مربوط با تایمر کانتر در این رجیستر انجام می شود . بیت های این رجیستر در تایمر کانترهای صفر و دو یکسان، اما در تایمر کانتر دو متفاوت می باشد .

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	

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	

سه بیت کم ارزش این رجیستر جهت تنظیم کلاک تایمر کانتر می باشد .

The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 42. Clock Select Bit Description

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 1-7

بیت های WGM(wave form generation mode) جهت انتخاب مد تایمر کانتر می باشد .
این مدها در ادامه توضیح داده خواهند شد .

Table 38. Waveform Generation Mode Bit Description(1)

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 2-7

بیت های COM تعیین می کنند در حالت تطابق (برای دو رجیستر OCRn وTCNTn)
وضعیت پین OCn چه حالتی باشد. که در مد های مختلف این عملکرد متفاوت می باشد
و در زیر این حالت ها در مدهای مختلف نشان داده شده است .

Compare Output Mode, Fast PWM Mode(1)

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 3-7

Compare Output Mode, non-PWM Mode

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 4-7

Compare Output Mode, Phase Correct PWM Mode(1)

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 5-7

رجیستر TCCR1A (timer counter control register A)

همانطور که گفته شد این رجیستر در تایمر کانتر یک متفاوت می باشد . این تایمر کانتر 16 بیتی بوده و شامل دو رجیستر کنترلی TCCR1A و TCCR1B می باشد . بیت های این دو رجیستر به صورت زیر است :

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	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت WGM10 و WGM11

این دو بیت به همراه بیت های سه و چهار در رجیستر TCCR1B جهت تعیین مد کاری می باشد .

Table 47. Waveform Generation Mode Bit Description(1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	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	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	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	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

جدول 6-7

بیت FOC1A و FOC1B

در مدهای غیر از PWM یک کردن این بیت ها بدون اینکه وقفه ایجاد کند در صورت تطبیق مقایسه باعث تغییر وضعیت پین های OC1A و OC1B مطابق با وضعیت بیت های COM در رجیستر TCCR1 می شود .

بیت COM1B0 ، COM1B1 ، COM1A0 و COM1A1

این بیت ها وضعیت پایه OC1A و OC1B در مد های مختلف و به هنگام تطبیق مشخص می کنند .

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on compare match
1	0	Clear OC1A/OC1B on compare match (Set output to low level)
1	1	Set OC1A/OC1B on compare match (Set output to high level)

جدول 7-7

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OCnA/OCnB disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at TOP
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at TOP

جدول 8-7

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OCnA on Compare Match, OCnB disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when downcounting.

جدول 9-7

رجیستر TCCR1A (timer counter control register B)

بیت های این رجیستر به صورت زیر است :

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت CS10 ، CS11 و CS12

توسط این سه بیت می توان مطابق جدول زیر کلاک تایمر را تعیین کرد .

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.

جدول10-7

بیت های WGM12 و WGM13

همانطور که قبلا گفته شد در تعیین مد کاری نقش دارند .

بیت ICES1

این بیت نحوه تحریک واحد Capture را تعیین می کند بدین صورت که اگر صفر باشد تحریک در لبه پایین رونده و اگر یک باشد تحریک در لبه بالا رونده صورت می گیرد .

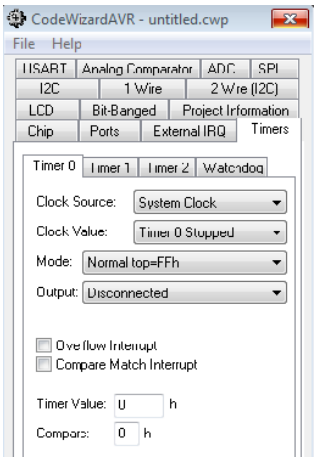
بیت ICNC1

این بیت حذف نویز در پایه ICP1 را فعال می کند .

تنظیمات CodeWizard

در CodeWizard با زدن گزینه timers همانگونه که در شکل پیداست چهار گزینهtimer0 , timer1 , timer2 و watch dog نمایان می شود . با زدن هر یک از آنها شما می توانید تایمر مورد نظر خود را انتخاب کرده وتنظیمات دلخواه را انجام دهید .

در گزینه clock source شما می توانید نحوه تامین کلاک تایمر را تعیین کنید .



1- تامین کلاک توسط cpu (system clock)

2- کلاک با لبه بالارونده (toping falling edge)

3- کانتر با لبه پایین رونده (toping rising edge)

شکل4-7

clock value : شما می توانید فرکانس تایمر را انتخاب کنید .

mode : امکان انتخاب مد کاری را به شما می دهد .

output : می توان خروجی را فعال کرد و مشخص کنید که هنگام compare match خروجی چگونه تغییر کند .در صورت انخاب مد ctc سه گزینه وجود خواهد داشت . مثلا اگر گزینه toggle on compare match را انتخاب کنید خروجی شما با هر برابری تغییر وضعیت خواهد داد .

اما اگر مد Fast Pwm یا Phase Correct Pwm را انتخاب کنید دو گزینه Inverted و Non-Inverted وجود خواهد داشت که در حالت Inverted خروجی یک بوده و هنگام

برابری خروجی صفر می شود و تا زمانی که تایمر به مقدار Top برسد در آن حالت مانده و سپس دوبار خروجی یک می شود . در حالت Non-Inverted عکس این اتفاق رخ می دهد .

Overflow Interrupt :

با زدن تیک این گزینه در در صورت رسیدن T/C به مقدار Top وقفه رخ می دهد .

Compare match Interrupt :

وقفه برابری را فعال می کند .

Timer value : مقدار اولیه T/C را می توان درآن قرار داد . (به صورت هگز)

Compare : مقدار رجیستر OCRn در اینجا تعیین می شود .

تذکر : در صورت فعال کردن وقفه CTC با ایجاد برابری , cpu به داخل تابع وقفه پرش می کند و در این تابع می توان مقدار رجیستر OCR را تغییر داد .

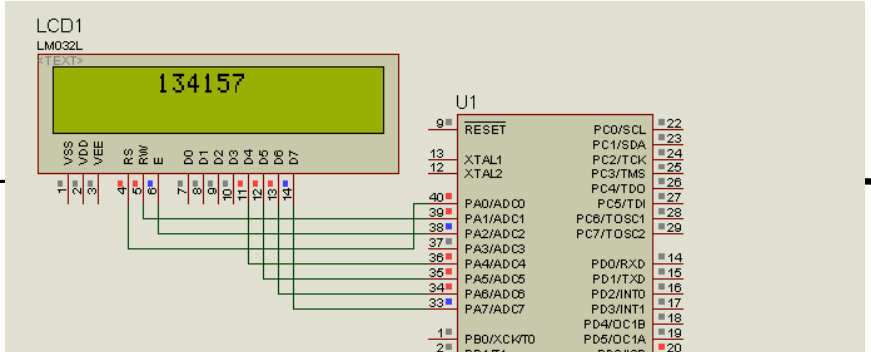
حالت Capture

تایمر کانتر یک دارای یک واحد موسوم به Capture می باشد که هرگاه پایه ICP میکرو را از بیرون تحریک کنیم محتوای تایمر کانتر در آن لحظه در داخل دو رجیستر هشت بیتی با نام ICRL و ICRH کپی می شود . توجه کنید که نویز ممکن است با عث تحریک ICP شود . برای جلوگیری از این کار باید در کد ویزارد noise cancel را فعال کنیم تا تحریکی مورد قبول می باشد که حداقل چهار ماشین سیکل طول بکشد .

مد capture دارای وقفه نیز می باشد یعنی با تحریک پایه ICP، CPU علاوه بر نمونه برداری از محتوای T/C به تابع وقفه نیز پرش می کند .

در مثال زیر با هر بار تحریک پایه ICP از بیرون محتوای T/C بر روی LCD نمایش داده می شود :

```
#include <mega16.h>                                  // میکرو مورد استفاده
#include <stdio.h>                                   //فراخوانی کتابخانه stdio.h
#asm                                                 //شروع برنامه اسمبلی
  .equ __lcd_port=0x1B ;PORTA                        //معرفی پورت A به عنوان پورت نمایشگر
#endasm                                              //پایان برنامه اسمبلی
#include <lcd.h>                                     //فراخوانی کتابخانه lcd.h
Char    a ,b , str[10] ;                             //معرفی متغیرها
interrupt [TIM1_CAPT] void timer1_capt_isr(void)     {        // وقفه capture
sprintf(str,"%d%d",ICR1H,ICR1L);       // خواندن مقدار تایمرکانتر و تبدیل آنها به رشته
   lcd_clear();                                      // پاک کردن نمایشگر
   lcd_gotoxy(6,0);                                  // رفتن به سطر و ستون مورد نظر
   lcd_puts(str);       }                            // نمایش رشته ذخیره شده
void main(void)    {                                 //برنامه اصلی
lcd_init(16);                                        //آماده سازی نمایشگر
while (1)   ;     }                                  //حلقه بینهایت
```



شکل5-7

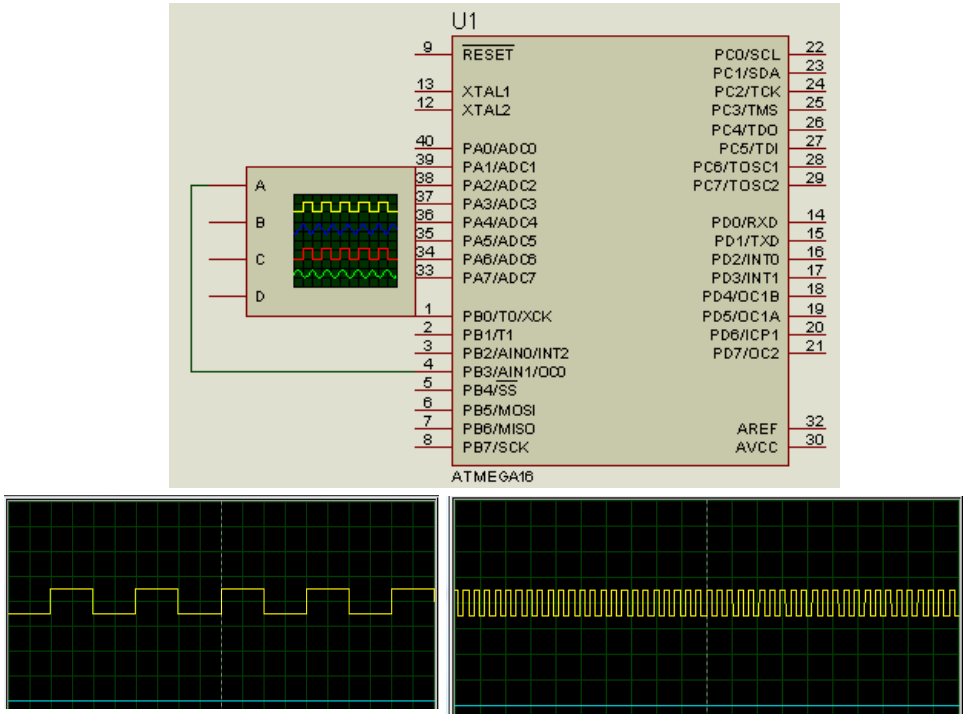
مثال : برنامه ای بنویسید که فرکانس یک پالس ابتدا رفته رفته زیاد شود و بعد از نهایی شدن رفته رفته کم گردد و این کار دائما تکرار گردد .

این کار را به کمک مد CTC انجام می دهیم :

```
#include <mega16.h>                                   // معرفی نوع میکرو
unsigned char a,b;                                    // معرفی متغیرها
interrupt [TIM0_COMP] void timer0_comp_isr(void)  {   // وقفه برابری
a++ ;                                                 // افزایش یک واحدی متغیر در هر بار برابری
if(a==200&&b==0)  {                                   // اگر دویست بار برابری رخ داد و در حالت افزایش بودیم
 a=0 ;                                                // صفر کردن متغیر شمارشگر برابری
 OCR0=OCR0+10;  }                                     // OCR به اندازه ده واحد افزایش یابد

   if(OCR0>230)  {                                    // اگر مقدار OCR بیشتر از 230 بود
   b=1;  }                                            // یک کردن متغیرb جهت ورود به حالت کاهشی
```

```
if(a==200&&b==1)    {          //اگر دویست بار برابری رخ داد و در حالت کاهشی بودیم
a=0;                           // صفر کردن متغیر شمارشگر برابری
OCR0=OCR0-10;    }             // OCR به اندازه ده واحد کاهش یابد
if(OCR0<3) {                   //اگر مقدارOCR کوچکتر از سه بود
b=0 ;  } }                     //صفر کردن متغیرb جهت ورود به حالت افزایشی
```



شکل6-7

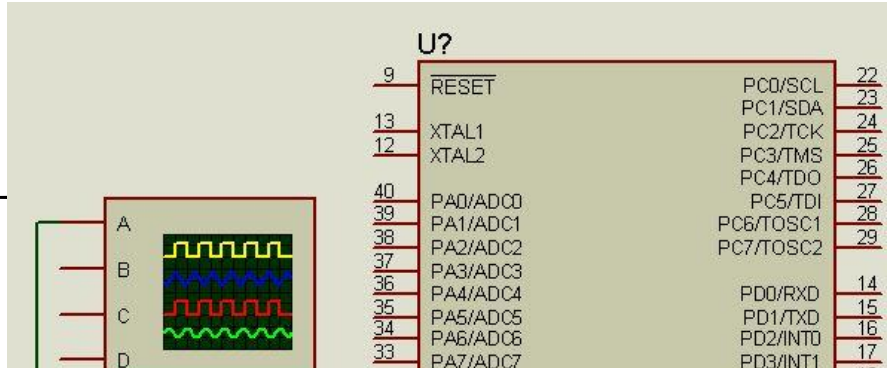
جهت مشاهده شکل موج فوق در Proteus از مسیر Virtual Instrument Mode گزینه VIRTUAL TERMINAL را انتخاب کنید و آن را در صفحه اصلی قرار دهید و یکی از پایه های آن را به پایه OCn ی که انتخاب کرده اید وصل کنید .

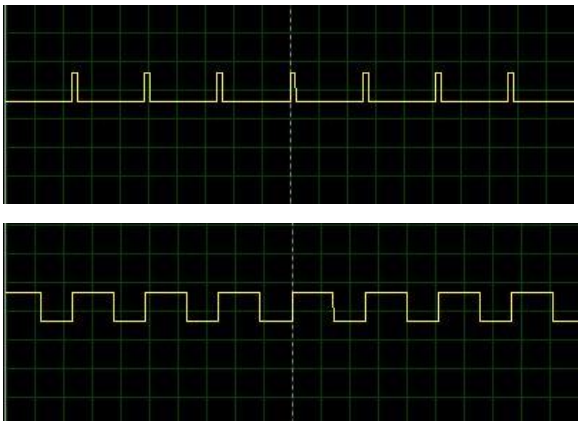
مشاهده می شود که عرض پالس و فرکانس خروجی تغییر می کند اما میزان موثر تغییری نمی یابد . در واقع زمان یک بودن سیگنال با زمان صفر بودن آن برابر است .

مثال :

به کمک مد Fast PWM برنامه ای بنویسید که نور یک لامپ آرام آرام افزایش و سپس همانگونه کاهش یابد .

```
#include <mega16.h>                                   //معرفی میکرو مورد استفاده
unsigned char a=0 , b=0 ;                             // معرفی متغیرها
interrupt [TIM0_OVF] void timer0_ovf_isr(void){       //وقفه برابری
a++;                                                  //افزایش یک واحدی متغیر جهت شمارش برابری
if((a==200)&&(b==0))  {                               //اگر دویست بار برابری رخ داد و در حالت افزایش بودیم
OCR0=OCR0+5;                                          // OCR به اندازه پنج واحد افزایش یابد
a=0;      }                                           //صفر کردن متغیر شمارشگر برابری
if(OCR0>240)          {                               //اگر مقدار OCR بیشتر از 240 بود
 b=1;         }                                       //یک کردن متغییرb جهت ورود به حالت کاهشی
 if(OCR0<10)      {                                   //اگر مقدارOCR  کوچکتر از سه بود
 b=0;          }                                      //صفر کردن متغییرb جهت ورود به حالت افزایشی
 if((a==200)&&(b==1)){                                //اگر دویست بار برابری رخ داد و در حالت کاهشی بودیم
a=0;                                                  //صفر کردن متغییر شمارشگر برابری
OCR0=OCR0-5 ;       } }                               // OCR به اندازه ده واحد کاهش یابد
```





شکل7-7

همانگونه که در شکل مشخص ، میزان موثر بودن پالس دائما در حال تغییر است و این سبب کم می شود که اگر یک LED به خروجی ما وصل باشد میزان نور آن کم وزیاد شود . همچنین مشاهده می کنید که در این حالت فرکانس تغییر نمی کند .

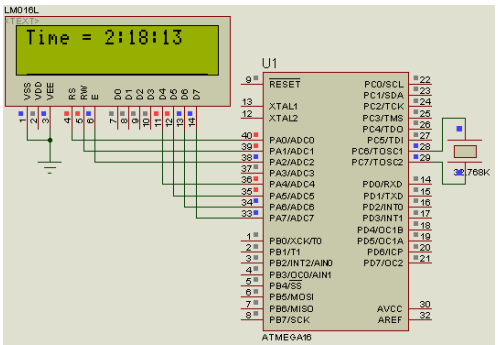
RTC (Real Time Clock)

تایمر دو دارای قابلیتی موسوم به RTC می باشد که توسط آن و با وصل یک اسیلاتور به پایه های TOSC1 و TOSC1 کلاک تایمر از طریق این کریستال تامین می شود. اگر این کریستال 32.768 MH انتخاب شود. با انتخاب تقسیم کلاک تایمر بر عدد 128مقدار 256 بدست می آید یعنی هر ثانیه یک بار تایمر2 سرریز می کند حال اگر برنامه ساعت را داخل وقفه سرریز تایمر 2بنویسیم می توان یک ساعت دقیق مانند مثال زیر طراحی کرد.

مثال : برنامه ساعت بکمک RTC

```
#include <mega16.h>                                    // میکرو مورد استفاده
#include <stdio.h>                                     //stdio.h فراخوانی کتابخانه
#asm                                                   // شروع برنامه اسمبلی
.equ __lcd_port=0x1B ;PORTA                            // معرفی پورت A به عنوان پرت نمایشگر
#endasm                                                // پایان برنامه اسمبلی
#include <lcd.h>                                       //lcd.h فراخوانی کتابخانه
unsigned char second, minute,hour;                     // معرفی متغیرها
unsigned char lcd_buff[10];                            // معرفی متغیرها
interrupt [TIM2_OVF] void timer2_ovf_isr(void)  {      // وقفه سرریز
if(second==59)  {                                      // اگر ثانیه برابر 59 شد
second=0;                                              // ثانیه را صفر کن
if(minute==59)  {                                      // اگر دقیقه برابر 59 بود
        minute=0;                                      // دقیقه را صفر کن
```

```
if(hour==24)                    //اگر ساعت برابر 24 بود
hour=0;                         //ساعت را صفر کن
else                            //اگر ساعت 24 نبود
hour++;     }                   //ساعت را یک واحد افزایش ده
else                            // اگر دقیقه 59 نبود
minute++;   }                   //دقیقه را یک واحد افزایش بده
else                            //اگر ثانیه 59 نبود
second++;                       //ثانیه را یک واحد افزایش بده
sprintf(lcd_buff,"Time = %d:%d:%d",hour, minute, second);
                                // خواندن زمان و تبدیل کردن آن به رشته جهت نمایش
lcd_clear();                    // پاک کردن نمایشگر
lcd_puts(lcd_buff);    }        //نمایش رشته ذخیره شده
void main(void)     {           // برنامه اصلی
lcd_init(16);                   //آماده سازی نمایشگر
while(1);  }}
```



شکل8-7

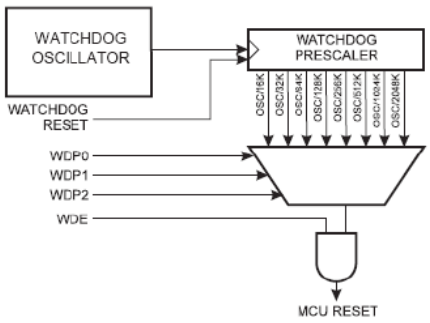
Watch dog Timer

میکرو کنترلرهای AVR دارای یک تایمر موسوم به Watchdog یا سگ نگهبان هستند که وظیفه این تایمر مراقبت از میکرو در برابر هنگ کردن است . میکرو به هنگام کار ممکن است به دلیل نویز یا هر چیز دیگری وارد آدرس نامشخصی از حافظه شود و در آنجا گیرد کند که در چنین مواقعی Watchdog می تواند به طور اتوماتیک میکرو را Reset کند . نحوه کار بدین صورت می باشد که تایمر شروع به شمارش کرده و هرگاه شمارش آن کامل شود و سرریز کند میکرو Reset می شود . پس باید در برنامه در فواصل زمانی مناسب تایمر را روشن و قبل از سرریز آن را خاموش یا مقدار آن را Reset کنید . Reset کردن تایمر توسط دستور اسمبلی زیر انجام می پذیرد:

```
#asm  (“WDR”)
```

فقط توجه داشته باشید که زمان اجرای برنامه کمتر از زمان پر شدن تایمر باشد و اگر برنامه شما طولانی است می توانید این کار را در چند جا و به طور مکرر انجام دهید .

مدار داخلی این تایمر در شکل زیر آمده است :



شکل7-9

رجیستر WDTCR (Watch Dog Timer Control Register)

Bit	7	6	5	4	3	2	1	0	
	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت WDP1,2,3

توسط این سه بیت و مطابق جدول زیر می توان زمان سرریز و Reset میکرو را توسط تایمر تعیین کرد .

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 3.0V	Typical Time-out at V_{CC} = 5.0V
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

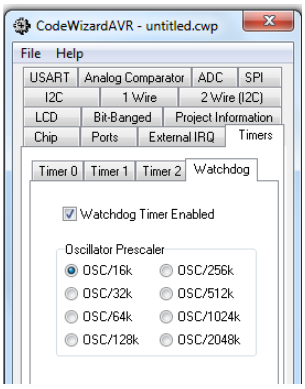
جدول11-7

بیت WDE

با یک شدن این بیت watchdog فعال می شود اما توجه کنید که برای خاموش کردن تایمر باید علاوه بر صفر کردن این بیت باید بیت WTDE نیز یک شود .

تنظیمات CodeWizard

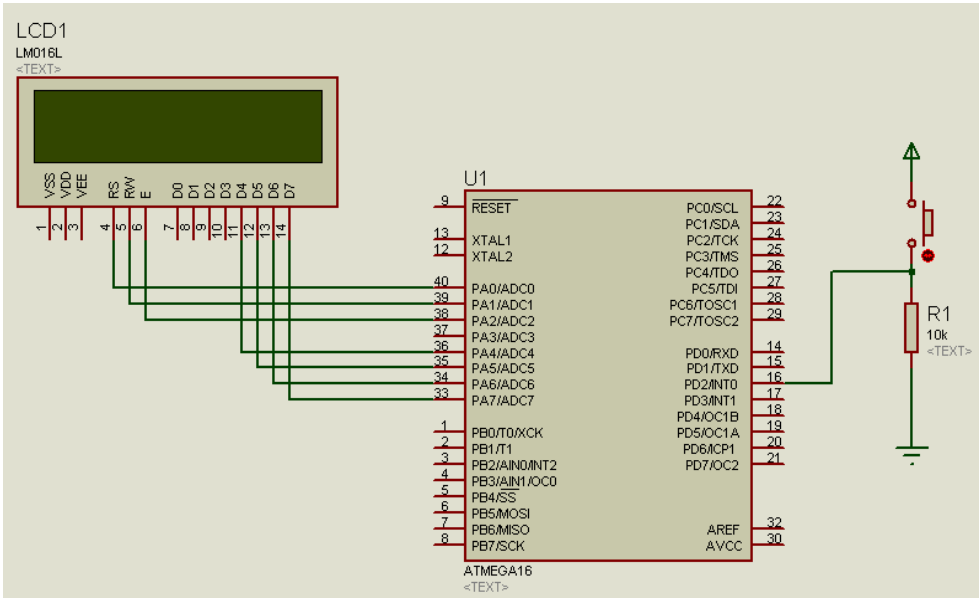
در کد ویزارد بر روی برگه Timers کلید کنید و در آنجا Watchdog را انتخاب کنید با فعال کردن آن می توانید کلاک تایمر و زمان Reset را تعیین کنید .



شکل10-7

مثال : در این مثال میکرو در حال نمایش کاراکتر "Salam" بر روی نمایشگر می باشد و مقدار Watchdog دائما در حال Reset شدن می باشد . حال اگر کلید را فشار دهیم وقفه رخ داده و از آنجا که در داخل تابع وقفه یک حلقه بینهایت می باشد مانند یک هنگ عمل می کند و تایمر پر شده و Reset رخ می دهد .

```
#include <mega16.h>                                    //معرفی میکرو مورد استفاده
#asm                                                   //شروع برنامه اسمبلی
.equ __lcd_port=0x1B ;PORTA                            //معرفی پورت A به عنوان پرت نمایشگر
#endasm                                                // پایان برنامه اسمبلی
#include <lcd.h>                                       //فراخوانی کتابخانه lcd
interrupt [EXT_INT0] void ext_int0_isr(void)  {        // تابع وقفه صفر
lcd_clear();                                           //پاک کردن نمایشگر
while(1);     }                                        //حلقه بینهایت
void main(void)  {                                     // تابع اصلی
while (1)  {                                           // حلقه بینهایت
   lcd_gotoxy(0,0);                                    // رفتن به سطر و ستون مورد نظر
   lcd_putsf("Salam")                                  //نمایش یک کاراکتر بر روی نمایشگر
   lcd_putsf("      "                                  //پاک کردن کاراکترهای باقیمانده از نمایش قبل
   #asm("WDR")          } }                           //Reset کردن تایمر
```


LCD1
LM016L
<TEXT>
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA16
<TEXT>
R1
10k
<TEXT>

شکل11-7