دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

دستورکار آزمایشگاه ریزپردازنده
دستورکار آزمایشگاه طراحی سیستم‌های دیجیتال2

(مبتنی بر ریزپردازنده ATMEGA16/32)

تهیه کننده:

زهرا محمدزاده

بررسی کننده:

دکتر امیر خورسندی

شهریور 1401

4 جلسه چهارم

آشنایی با وقفه‌ها و تایمرها

4.1 هدف

معمولاً تعداد زیادی از کارهای کنترلی، نیازمند اندازه‌گیری زمان یا شمارش یک اتفاق هستند. برای پیاده‌سازی چنین عملیات‌هایی، در ریزپردازنده‌ها یک یا چند تایمر[1] تعبیه شده است. در این جلسه نحوه کار با تایمرها، حالت‌های کاری مختلف آن‌ها و وقفه‌های مرتبط مورد بحث قرار خواهند گرفت.

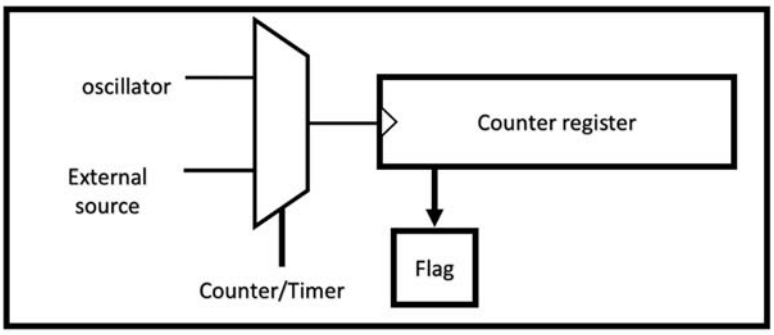
4.2 مقدمه

تایمرها در حقیقت شمارنده‌های سخت‌افزاری مجزایی هستند که در صورت فعال بودن به طور موازی با پردازش‌های CPU عمل شمارش را انجام داده و مقدار آن‌ها افزایش یا کاهش می‌یابد. از این رو تایمرها مهم‌ترین ابزار برای سنجش زمان در ریزپردازنده‌ها می‌باشند.

واحد تایمر مطابق شکل 4-1 با هر پالس ساعت یک واحد می‌شمارد. به طور مثال یک ریزپردازنده با فرکانس پالس ساعت 1MHz را در نظر بگیرید. اگر همین پالس ساعت ریزپردازنده را مستقیماً به تایمر وصل کنیم، محتوای ثبات Counter در هر یک میکروثانیه یک عدد افزایش می‌یابد. لذا برای تاخیر به اندازه 100 میکروثانیه بایستی منتظر ماند تا مقدار شمارنده تایمر از صفر به 100 افزایش یابد. لازم به ذکر است که منبع شمارش تایمر می‌تواند پالس ساعت داخلیCPU یا یک پالس خارجی باشد. چنان‌چه منبع پالس خارجی به کار رود، می‌توان از تایمرها به عنوان شمارنده[2] پالس‌های خارجی ریزپردازنده نیز استفاده نمود.

ریزپردازنده Atmega16/32 سه تایمر دارد که تایمرهای صفر و 2، هشت بیتی و تایمر 1 شانزده بیتی است. تعداد بیت‌های ذکرشده مشخص‌کننده بازه قابل شمارش و یا در واقع ح د بالای شمارنده تایمر هستند.

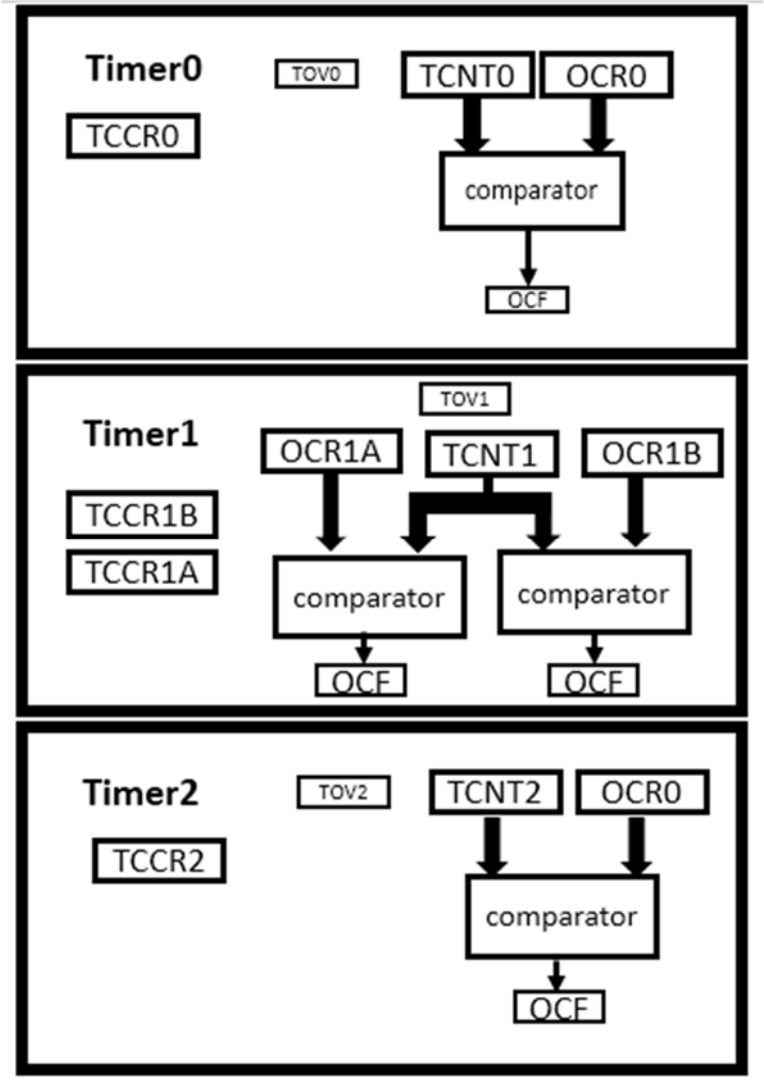
[1] Timer
[2] Counter



شکل 4-1: نمایی از ساختار کلی تایمر

4.3 ثبات‌های تایمر

ثبات‌های هر یک از تایمرها در شکل 4-2 نشان داده شده است.



شکل 4-2: ثباتهای هر یک از تایمرها

تنظیم تایمرها و استفاده از آن‌ها به وسیله‌ی ثبات‌های مربوطه انجام می‌شود. این ثبات‌ها در زیر معرفی شده‌اند. حرف n شماره‌ی تایمر را مشخص می‌کند و در ریزپردازنده Atmega16/32 می‌تواند صفر، یک یا دو باشد.

ثبات TCCRn[1]: در این ثبات پیکربندی تایمر انجام می‌شود. بیت‌های کنترلی تایمر صفر درشکل 4-3 نشان داده شده است.

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

شکل 4-3 : معرفی بیت‌های ثبات کنترلی Timer 0

[1] Timer/Counter Control Register

نحوه‌ی کار با بیت‌های این ثبات در جدول‌های زیر آمده است (برای جزییات بیشتر به Datasheet مراجعه نمایید).

جدول 4-1 : تنظیم حالت کاری تایمر صفر با استفاده از بیت‌های WGM00 و WGM01

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	BOTTOM	MAX

جدول 4-2 : تنظیم نحوه‌ی فعال شدن پایه‌ی OC0 در حالت غیر PWM با استفاده از بیت‌های COM00 و COM01

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 4-3 : نظیم نحوه‌ی فعال شدن پایه‌ی OC0 در حالت Fast PWM با استفاده از بیت‌های COM00 و COM01

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at BOTTOM, (non-inverting mode)
1	1	Set OC0 on compare match, clear OC0 at BOTTOM, (inverting mode)

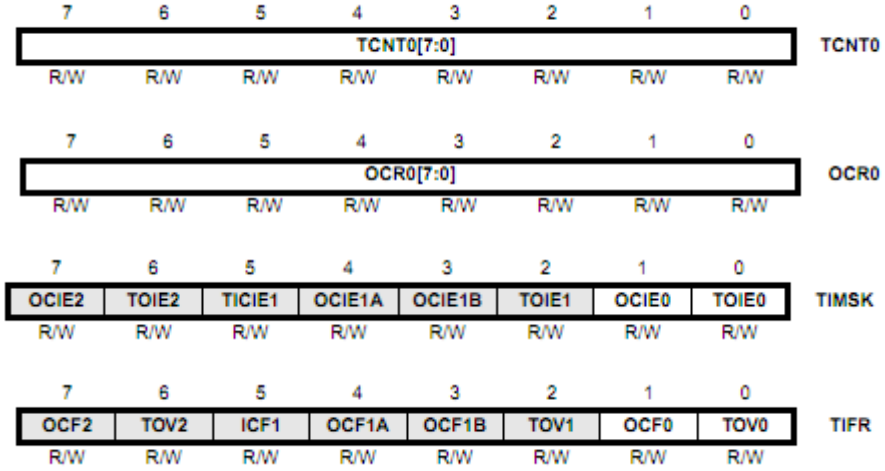
جدول 4-4 : تنظیم نحوه‌ی فعال شدن پایه‌ی OC0 در حالت Phase Correct PWM با استفاده از بیت‌های COM00 و COM01

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 4-5 : تنظیم فرکانس شمارنده با استفاده از بیت‌های CS00 و CS01 و CS02

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.

سایر ثبات‌های مربوط به تایمر صفر در شکل 4-4 نشان داده شده است.



شکل 4-4: ساختار ثبات‌های TCNT0، OCR0، TIMSK، TIFR

ثبات TCNTn[1]: این ثبات مقدار اصلی شمارنده یا تایمر را در هر لحظه درون خود نگه می‌دارد. به محض راه‌اندازی مجدد، محتوای آن صفر می‌شود و از آن پس با هر پالسی که به آن وارد می‌شود یکی می‌شمارد. هم‌چنین می‌توان مقداری را در آن ذخیره کرده یا از روی آن خواند.

ثبات OCRn[2]: محتوای این ثبات با محتوای TCNTn مقایسه می‌شود. اگر با هم برابر باشند پرچم OCFn یک خواهد شد.

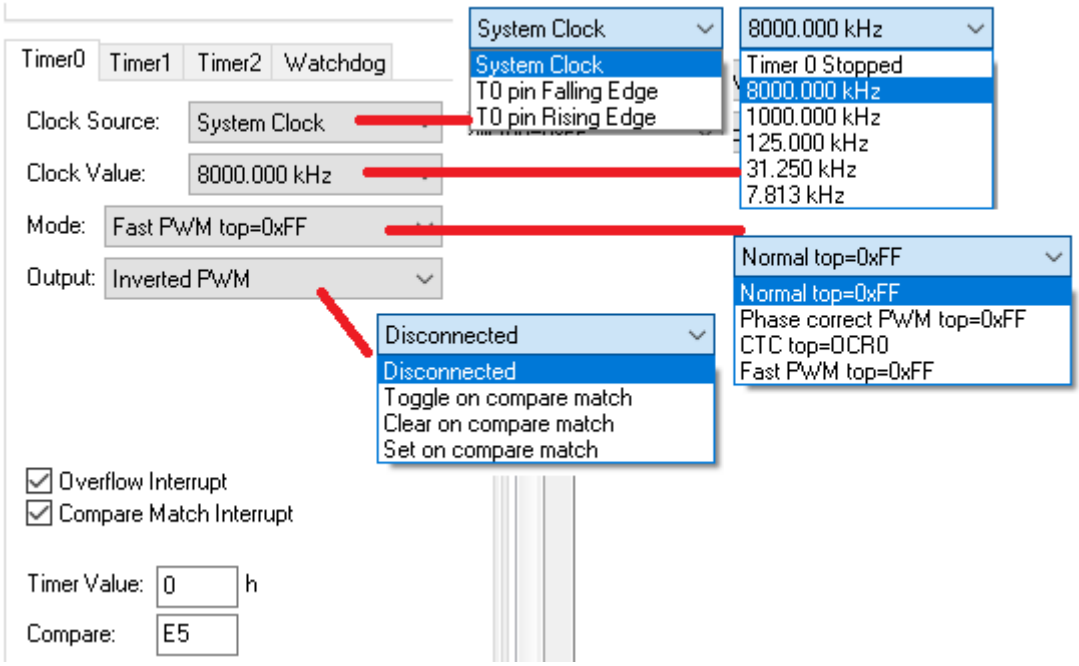
[1] Timer/Counter Register
[2] Output Compare Register

ثبات TIMSK[1]: وقفه‌های تایمر صفر، یک و دو را در شرایط سرریز کردن و یا برابر شدن محتوای ثبات‌های TCNTn و OCRn، به ترتیب با بیت‌های TOIEn و OCIEn فعال می‌کند.

ثبات TIFR[2]: هنگامی‌که وقفه‌های مورد نظر از تایمر فعال باشد، رخداد این وقفه‌ها در ثبات TIFR قابل مشاهده است. هرگاه سرریز اتفاق بیافتد پرچم سرریز (TOVn) برابر با یک می‌گردد و با برابر شدن ثبات‌های TCNn و OCRn، پرچم مقایسه‌ی خروجی (OCFn) یک خواهد شد.

4.4 تنظیمات CodeWizard برای تایمر

پارامترهایی که در تنظیم تایمر در محیط Codevision قابل پیکربندی هستند در شکل 4-5 نشان داده شده است.



شکل 4-5: تنظیمات تایمر در CodeWizard

[1] Timer/Counter Interrupt Mask Register
[2] Timer/Counter Interrupt Flag Register

1- **Clock Source**: همان گونه که گفته شد، منبع پالس ساعت تایمر می‌تواند پالس ساعت داخلی ریزپردازنده باشد یا این یکی از پایه‌های ریزپردازنده برای شمارش استفاده شود و تایمر به یک شمارنده تبدیل گردد (در این حالت هر زمان یک پالس به پایه‌ی مشخصی از ریزپردازنده اعمال شود، مقدار شمارنده یک واحد زیاد می‌شود و این زمان می‌تواند دوره تناوب مشخصی نداشته باشد).

2- **Clock Value**: در صورتی که برای تایمرها، Clock Source از نوع System Clock انتخاب شود، فرکانس تایمر توسط Clock Value تعیین می‌شود. با این کار فرکانس تایمر، مضربی از فرکانس ریزپردازنده خواهد بود (با ضریب کمتر از 1). این فرآیند Prescale نامیده می‌شود. در حالتی که در تنظیمات Clock Source تایمر را به Counter تبدیل کنیم، تنها Timer2 قابلیت Prescale کردن را خواهد داشت. برای به دست آوردن فرکانس تایمر در حالت Prescale (با مضرب N) از رابطه زیر استفاده می‌شود:

$$\text{Prescale: N=1,8,64,256,1024} \rightarrow F_{Timer-Clock} = \frac{F_{Osc}}{N}$$

3- **حالت کاری**: تایمرها می‌توانند شمارش را به انواع مختلف و تا مقادیر متفاوتی انجام دهند. چهار حالت قابل استفاده شامل Normal، CTC، Fast PWM و Phase Correct PWM می‌باشند که هر کدام از این مدها در بخش‌های بعدی توضیح داده خواهند شد.

4- **Output**: تایمرها می‌توانند بر روی پایه‌های مشخصی از تراشه با عناوین OCn (که n در آن شماره‌ی تایمر است) پالس‌هایی را به طور خودکار ایجاد کنند. این قابلیت برای زمانی که پالس‌های دقیق و منظم مورد نیاز است استفاده می‌شود.

5- **Input Capt**: امکان ثبت زمان رخدادهای خارجی با استفاده از تایمر 16 بیتی شماره یک فراهم می‌شود.

6- **Interrupt on**: نوع وقفه مورد نظر از تایمر انتخاب می‌شود.

7- **Value**: در این بخش مقدار اولیه برخی ثبات‌ها ثبت می‌شود.

4.5 حالت های کاری تایمر

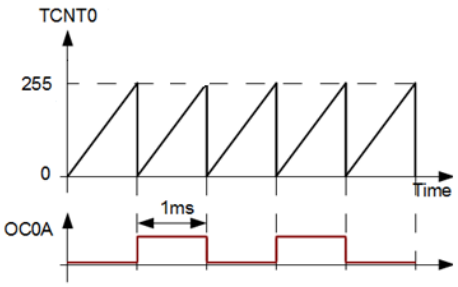
همان گونه که توضیح داده شد چهار حالت کاری مختلف برای تایمرها قابل اتخاب می‌باشد که در ادامه هر کدام از این حالت‌ها توضیح داده شده‌اند.

4.5.1 حالت Normal

در حالت Normal شمارش تایمر تا بالاترین مقداری که تایمر می‌تواند بشمارد ادامه می‌یابد (برای تایمر هشت بیتی تا 255 و برای تایمر شانزده بیتی تا 65535) و بعد از آن دوباره تایمر از صفر شروع به شمارش می‌کند. هر بار

که این سرریز اتفاق می‌افتد، خروجی بر روی پایه OCn تغییر وضعیت داده و یک پالس مربعی ایجاد می‌کند که فرکانس آن برای شمارنده m بیتی از فرمول زیر به دست می‌آید:

$$F_{Generatedwave} = \frac{F_{TimerClock}}{2^{m+1}}$$



شکل 4-6- نحوه‌ی شمارش و فعال شدن بیت OC0A در حالت شمارش Normal

همان طور که در شکل 4-6 مشاهده می‌شود مقدار OCR0A با هر سرریز تایمر 8 بیتی شماره 1، تغییر وضعیت می‌دهد.

در شکل فوق چرخه‌ی کار (Duty Cycle) برابر با 50 درصد است، چون مدت زمان یک و صفر بودن خروجی در یک دوره تناوب یکسان است. در حالت کلی چرخه کار طبق رابطه‌ی زیر محاسبه می‌شود:

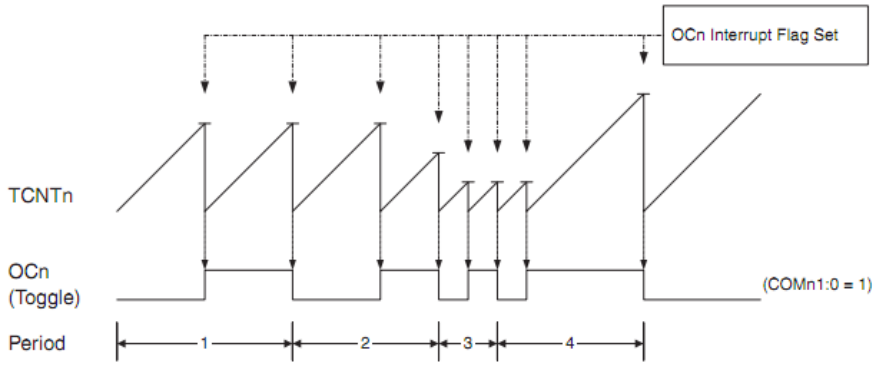
$$\text{چرخه کار} = \frac{\text{زمان یک بودن پالس}}{\text{زمان کل یک دوره تناوب}}$$

4.5.2 حالت CTC[1]

در این حالت محتوای شمارنده تایمر (محتوای ثبات TCNTn) با محتوای ثبات OCRnx (n صفر، یک یا دو و x برابر A یا B) مقایسه می‌شود. در صورتی که این دو برابر باشند مقدار شمارنده صفر قرار داده می‌شود. در حقیقت در این حالت مقدار OCRnx به عنوان حد ماکزیمم شمارش در نظر گرفته می‌شود. لذا پس از برابری TCNTn با OCRnx سرریز اتفاق می‌افتد و پایه خروجی PD7 (OC2) یا PB3 (OC0/AIN1) و یا PD5 (OC1A) یا PD4 (OC1B) بنا به تنظیمات انجام شده در کدویزارد مانند شکل 4-7 تغییر می‌کند. مزیت این حالت قابلیت تغییر ثبات OCRnx در حین برنامه است که می‌توان پالس‌هایی با دوره تناوب متغیر ایجاد نمود. فرکانس موج PWM در خروجی OCn برابر است با:

[1] Clear Timer on Compare Match

$$f_{OCnPWM} = \frac{f_{osc}}{2 * N * (1 + OCRnx)} \qquad N = 1,8,64,256,1024$$



شکل 4-7: تعیین حد بالای شمارش با استفاده از ثبات OCRnx در این شکل بعد از هر وقفه وضعیت پایه OCn تغییر می‌نماید.

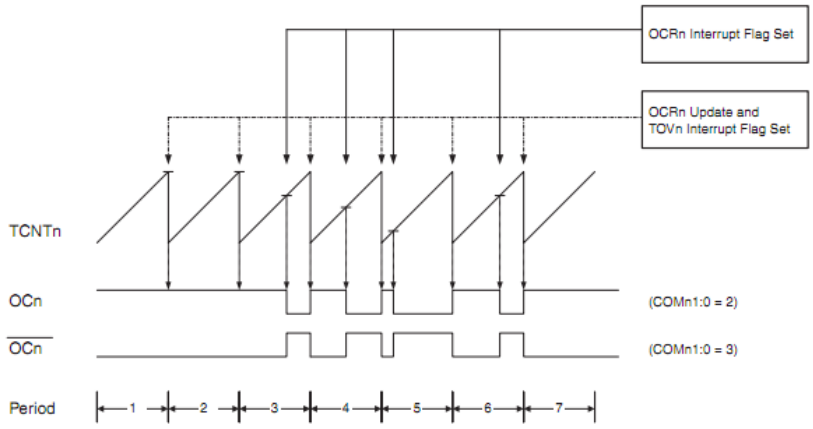
4.5.3 حالت Fast PWM

این حالت به عنوان ترکیبی از حالت‌های Normal و CTC در نظر گرفته می‌شود. در این حالت از یک سو شمارش مانند حالت Normal از صفر تا حد ماکزیمم (0xff و یا 0xffff) صورت می‌پذیرد و و از سوی دیگر مانند حالت CTC، مقدار ثبات TCNTn همواره با ثبات OCRn مقایسه می‌شود. پایه خروجی OCn در هنگام برابری این ثبات‌ها و هم‌چنین در زمان سرریز شدن تایمر مانند شکل 4-8 تغییر وضعیت می‌دهد. در این حالت امکان استفاده از دو وقفه تایمر شامل وقفه سرریز و وقفه Compare Match وجود دارد.

بدین ترتیب با تغییر فرکانس تایمر می‌توان دوره تناوب و با تغییر مقدار رجیستر OCRn چرخه‌ی کار را تغییر داد. لازم به ذکر است که مقدار OCRn در روتین وقفه سرریز تایمر به روز می‌گردد. فرکانس و چرخه کار موج PWM تولید شده در خروجی OCn با فرض شمارنده 8 بیتی برابر است با:

$$f_{OCnPWM} = \frac{f_{osc}}{N * 256} \qquad N = 1,8,64,256,1024$$

$$Duty\ Cycle(\%) = \frac{OCRnx}{256} * 100$$



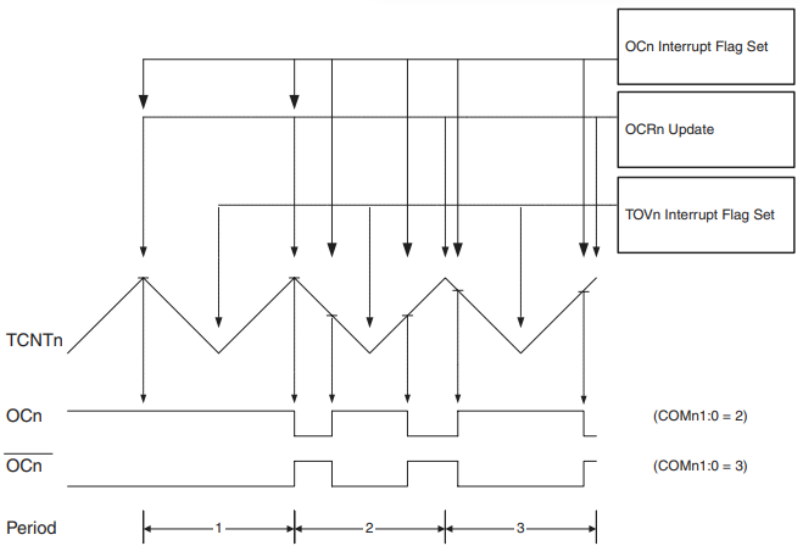
شکل 4-8: شکل موج تولیده شده بر روی پایه OCx در حالت Fast PWM

4.5.4 حالت Phase Correct PWM

در حالت Phase Correct PWM قابلیت ایجاد موج PWM با تصحیح فاز وجود خواهد داشت. در این حالت تایمر ابتدا از مقدار اولیه تا مقدار ماکزیمم به صورت افزایشی می‌شمارد. سپس از مقدار ماکزیمم تا مقدار اولیه به صورت کاهشی به شمارش ادامه می‌دهد. در حین شمارش مقدار TCNTn با OCRn مقایسه می‌شود و در صورتی که برابر باشند خروجی OCn تغییر وضعیت می‌دهد.

در این حالت امکان استفاده از دو وقفه تایمر شامل وقفه سرریز و وقفه Compare Match وجود دارد.

وقفه سرریز تایمر بر خلاف حالتهای قبلی با صفر شدن TCNTn رخ می‌دهد و وقفه CompareMach در صورت برابری مقدار TCNTn و OCRn اتفاق می‌افتد در هریک از وقفه ها میتوان مقدار ORCn را تغییر داد ولی فقط هنگامی که TCNTn برابر با حد ماکزیمم می‌باشد مقدار OCRn به روز می‌گردد.



شکل 4-9: نحوه فعال شدن پایه‌ی OCn در حالت Phase Correct PWM

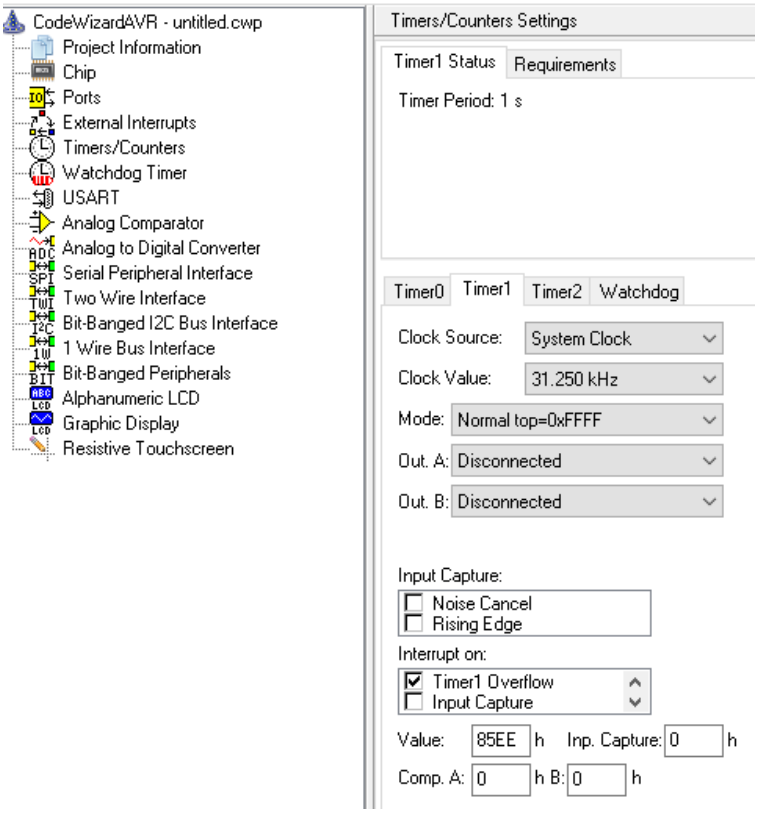
در این حالت، فرکانس موج PWM در پایه خروجی OCn از رابطه‌ی زیر محاسبه می‌شود:

$$F_{\mathrm{PWM}} = \frac{\mathrm{F}_{osc}}{\mathrm{N} * 510} \qquad N = 1,8,64,256,1024$$

عدد 510 بر اساس شمارش از 0 تا 255 و از 255 تا 0 به دست آمده است.

4.6 مثال کاربردی- طراحی ثانیه شمار

ثانیه شماری را در نظر بگیرید که قابلیت شمارش تا 60 ثانیه را دارد و در هر لحظه زمان را روی LCD نیز نشان می‌دهد. در شکل 4-10 نمایی از تنظیمات تایمر یک برای این منظور نشان داده شده است.



شکل 4-10: نمایی از تنظیمات تایمر یک برای طراحی ثانیه شمار

کد ایجاد شده توسط CodeWizard به همراه تغییرات لازم در برنامه 4-1 نشان داده شده است. برای مختصر شدن حجم برنامه، توضیحات حذف شده است.

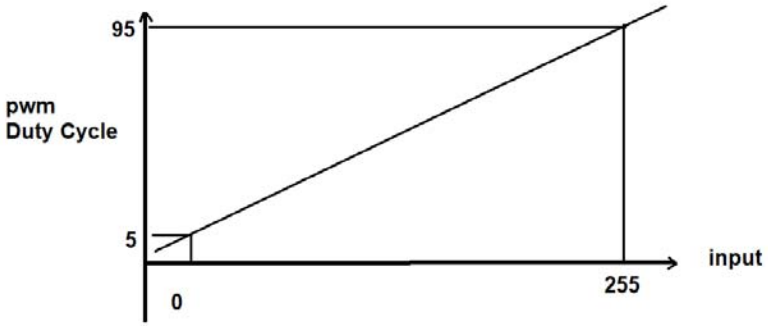
```
#include <mega16.h>
#include <alcd.h>
#include <stdio.h>

int i=0;
char str[10];

interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{

TCNT1H=0x85EE >> 8;
TCNT1L=0x85EE & 0xff;
i++;
if( i==60 )
{
 i=0;
}

sprintf(str,"%d ",i);
lcd_clear;()
lcd_puts(str);
}
```

برنامه 4-1

```
void main(void)
{
//NOTE: define port init for your application

TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) |
(0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (1<<CS12) |
(0<<CS11) | (0<<CS10);
TCNT1H=0x85;
TCNT1L=0xEE;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) |
(1<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

// Global enable interrupts
#asm("sei")
lcd_init(16);
lcd_clear();

while(1);
}
```

4.7 محاسبه خطا

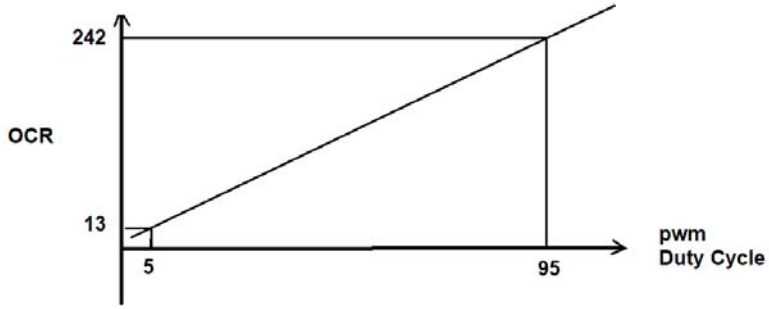
در برخی سیستم‌ها گاهی لازم خواهد شد که رفتار سیستم بنا به ورودی‌های دریافت شده تغییر کند و با شرایط جدید تطبیق یابد. به عنوان مثال فرض کنید طراحی یک پالس PWM با دوره تناوب 8.192 ms و با چرخه‌ی کار متغیر (بر اساس داده دریافتی از ورودی‌های متصل به یکی از درگاه‌ها) مد نظر باشد، در این شرایط بنا بر شکل 4-11 رابطه‌ی ذیل برقرار است.



شکل 4-11: رابطه خطی بین ورودی و Duty_Cycle

$$DutyCycle_{PWM} = \frac{95-5}{255-0} * input + 5$$

بعد از تعیین DutyCycle بر اساس ورودی، با فرض حالت کاری Fast PWM و وضعیت پایه خروجی به صورت Non _inverted PWM، مقادیر مختلف OCR محاسبه می‌شود.



$$OCR = \frac{242-13}{95-5} * (DutyCycle_{PWM} - 5) + 13$$

از تلفیق روابط بالا، رابطه‌ی زیر به دست می‌آید.

$$OCR = (\frac{229}{255} * input) + 13$$

بدین ترتیب مقدار ثبات OCR برای ایجاد موج PWM محاسبه می‌گردد. حال دستورات زیر را برای پیاده سازی در نظر بگیرید:

```
1    OCR_reg=(229/255)*input+13;
2    OCR _reg=((229*input)/255)+13 ;
3    OCR _reg=(((229*input)+13*255)/255);
```

با توجه به این که ریزپردازنده Atmega16/32 یک ریزپردازنده ۸ بیتی است و نوع داده floating point را پشتیبانی نمی‌کند، مقدار خروجی رابطه‌ی یک همواره برابر با ۱۳ خواهد بود . ولی روابط 2 و 3 به درستی اجرا می‌شوند. با استفاده از برنامه 2-4 می‌توان حاصل عملیات روابط 1 تا 3 را مقایسه نمود.

برنامه 2-4

```
for(input=0;input<255;input++)
{
      ans1_8bit=(229/255)*input+13;
      ans2_8bit=((229*input)/255)+13;
      ans3_8bit=(((229*input)+13*255)/255);

sprintf(scr,"%2d,%2d,%2d,%2d\n\r",input,ans1_8bit,ans2_8bit,ans3_8bit);
 puts(scr);
  delay_ms(50);
}
```

input	ans1	ans2	ans3	real
0	13	13	13	13
1	13	13	13	13.8
2	13	14	14	14.7
3	13	15	15	15.7
28	13	38	38	38.1
188	13	181	181	181.8
254	13	241	241	241.1

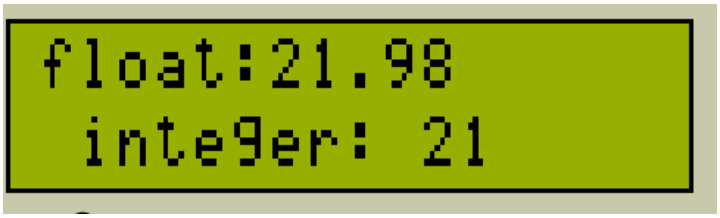
لذا می‌توان درصد خطای دوره تناوب‌های مورد نظر را محاسبه کرد که برای این منظور از رابطه زیر استفاده می‌شود.

$$Error(\%) = \frac{Real\ value\ (floating\ point) - Calculated\ value(integer)}{Real\ value\ (floating\ point)} * 100$$

برای محاسبه رقم‌های اعشاری می توان از تکنیک‌های دیگری استفاده کرد. در یکی از تکنیک‌ها، می‌توان مقدار متغیر را 100 برابر نمود و بعد از پایان محاسبات، نتیجه نهایی را با بر 100 تقسیم نمود تا قسمت حقیقی محاسبه گردد و با محاسبه باقیمانده تقسیم متغیر بر100 قسمت اعشاری نیز محاسبه می‌شود. در این حالت بهتر است متغیر از نوع unsigned long int انتخاب شود. با استفاده از این روش می‌توان مانند برنامه 4-3 بسیاری از پارامترهای اعشاری را محاسبه و روی صفحه نمایش مانند شکل 4-12 نشان داد.

برنامه 4-3

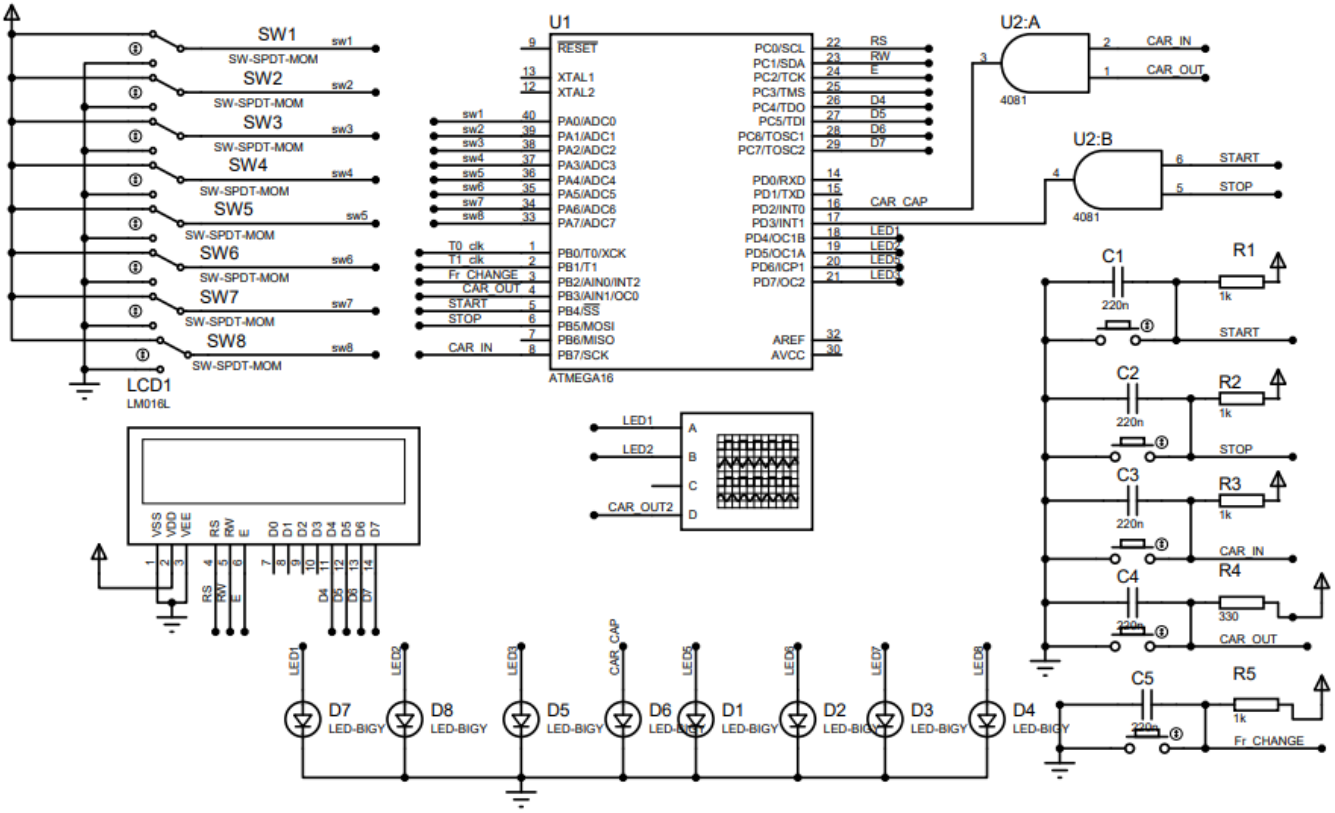
```
data3=10;
data2=(((229*data3)+13*255)/255);
data=(100*((229*data3)+13*255)/255);
sprintf(scr,"float:%d.%d \n\r",data/100,data%100);
lcd_puts(scr);
sprintf(scr,"integer: %d \n\r",data2);
lcd_puts(scr);
```



شکل 4-12: قالیت ایجاد اعداد ممیز شناور در ریزپردازنده هشت بیتی

4.8 برنامه‌های اجرایی مبحث تایمرها

با در نظر گرفتن سخت‌افزار نشان داده شده در شکل 4-13، برنامه‌های زیر را در ارتباط با مبحث تایمر و وقفه‌های آن‌ها بنویسید.



شکل 4-13: نمای سخت‌افزاری مبحث تایمر

1- یک تایمر(کرنومتر) با دقت 0.01 ثانیه طراحی و پیاده‌سازی کنید که با فشردن کلید start شروع به شمارش نماید و زمان آن همواره روی خط اول LCD نمایش داده شودو این تایمر باید با زدن کلید stop متوقف شود و زمان به صورت ثابت روی LCD باقی بماند. اگر بعد از متوقف شدن، مجدداً کلید stop زده شود، زمان نشان داده شده روی LCD صفر گردد. (راهنمایی: ابتدا یکی از تایمرها را در بازه زمانی مناسب فعال نمایید و

زیربرنامه محاسبه زمان را در وقفه تایمر فراخوانی نمایید. سپس در روتین وقفه خارجی یک، وضعیت پایه‌های start و stop را بررسی کرده و بنا به شرایط، تصمیم لازم اتخاذ گردد.)

2- پارکینگی را در نظر بگیرید که به اندازه 1000 ماشین ظرفیت دارد. مسئول پارکینگ به محض ورود ماشین، کلید CAR_IN و در زمان خروج ماشین، کلید CAR_OUT را فشار می‌دهد و ظرفیت فضای خالی پارکینگ روی LCD نمایش داده می‌شود و بعد از پر شدن عبارت FULL پخش می‌شود. اگر کلیدها به پایه‌های وقفه‌ی خارجی متصل باشند، زیربرنامه‌ای بنویسید که ظرفیت خالی پارکینگ روی LCD نمایش داده شود. (راهنمایی در زیربرنامه وقفه خارجی، وضعیت کلیدهای مورد نظر بررسی نمایید و از آنجا که ساعت در خط اول نمایش داده می‌شود برای نمایش ظرفیت پارکینگ از خط دوم استفاده نمایید.)

3- زیربرنامه‌ای بنویسید که یک شکل موج مربعی را بر روی پایه‌های خروجی تایمرها (PD4/OC1B و PD5/OC1A) ایجاد نماید. فرکانس ورودی باید از سوییچ‌های متصل به ریزپردازنده دریافت شده و دوره تناوب روی LCD نمایش داده شود. (دوره تناوب را در محدوده 1 میکروثانیه تا 10 میلی ثانیه در نظر بگیرید و با توجه به محدودیت شبیه‌ساز پروتئوس، فقط از فرکانس‌های 8 مگاهرتز و 1 مگاهرتز استفاده نمایید. دوره تناوب شکل موج را می‌توانید در خط دوم در کنار ظرفیت خالی پارکینگ نمایش دهید.)