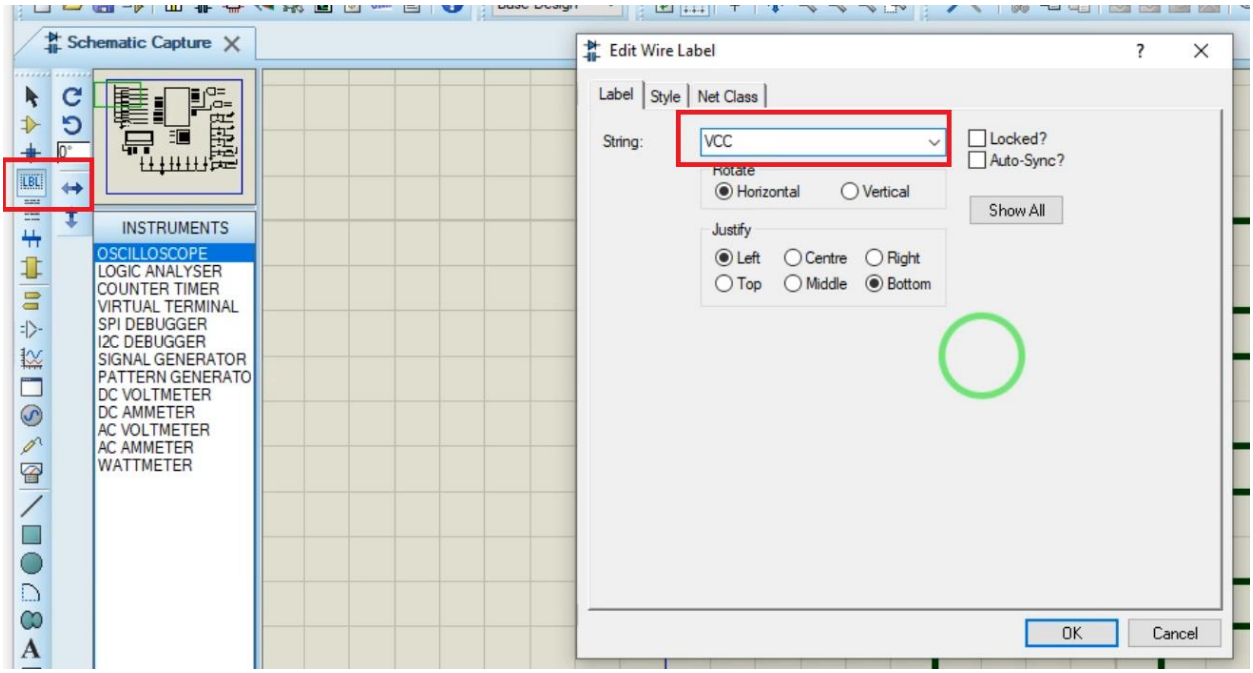
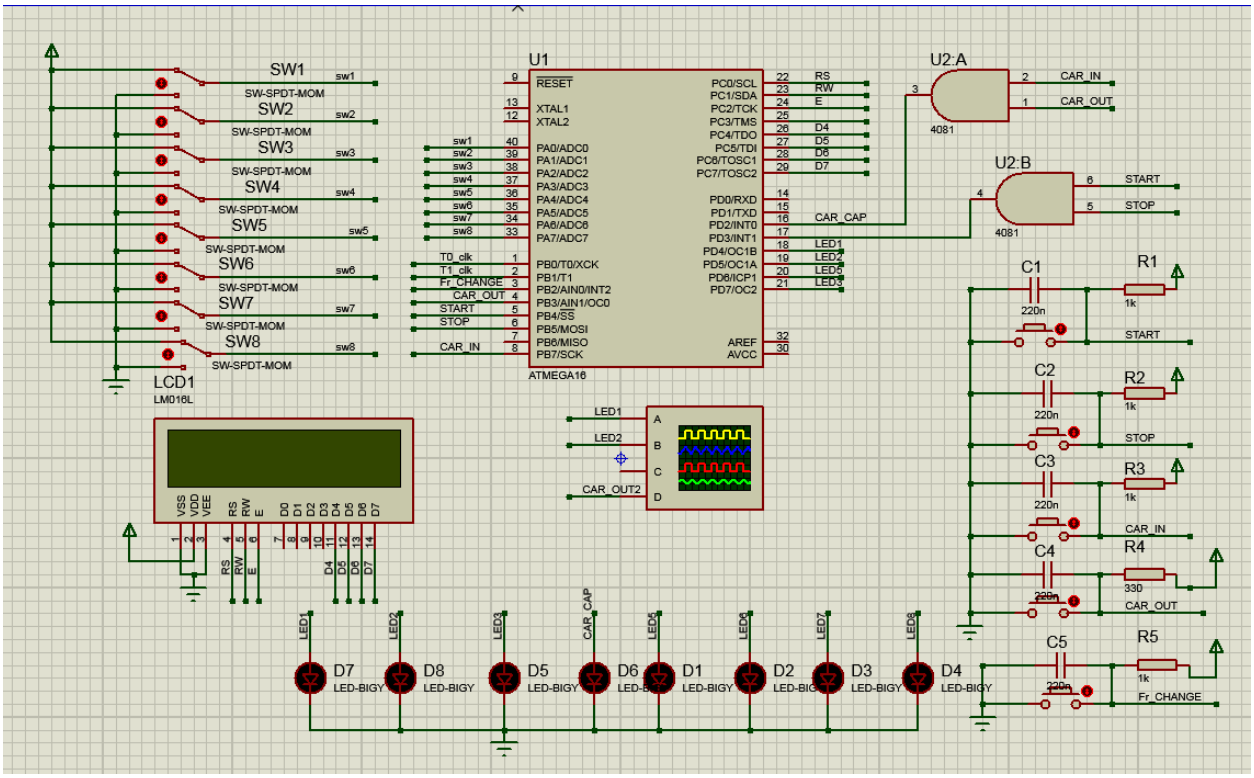
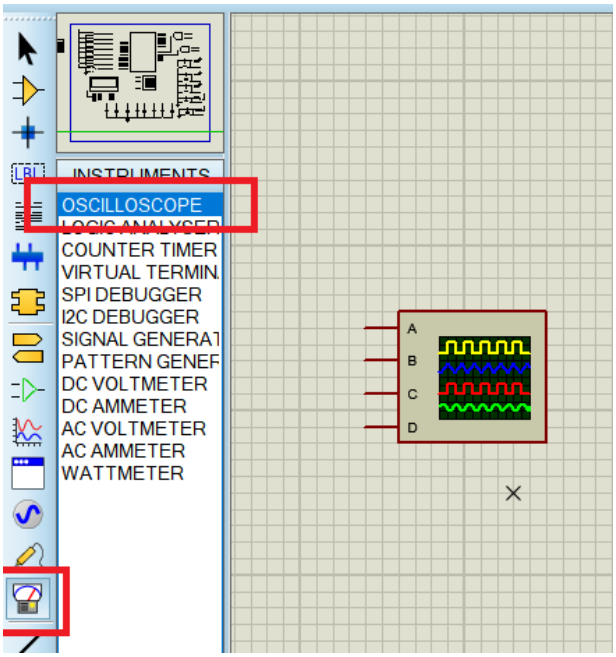
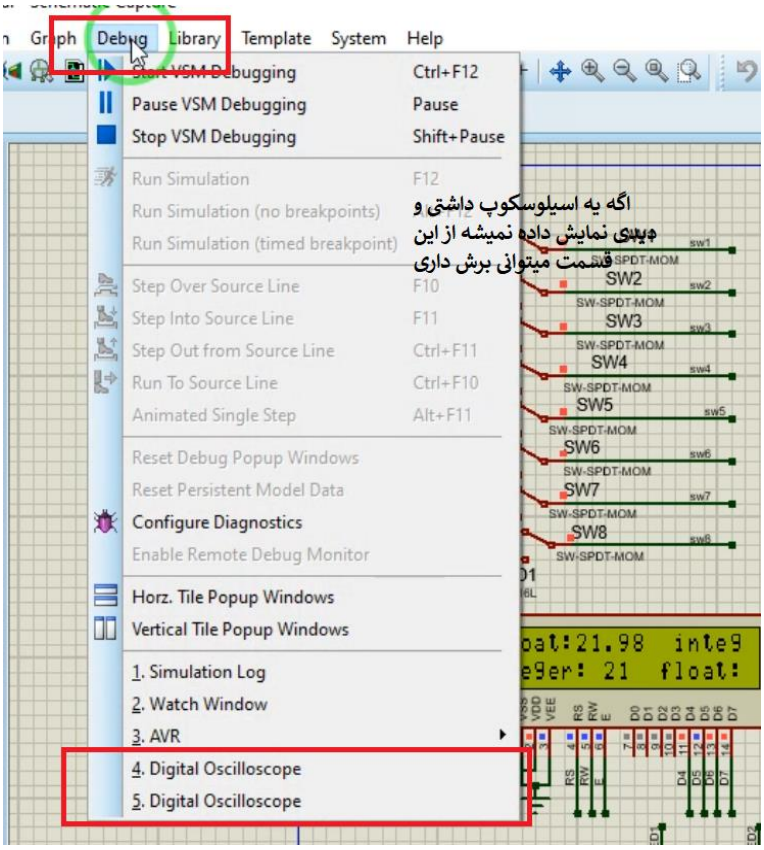
Session4 : TIMERS , INTERRUPTS IN TIMERS.



سخت افزار این جلسه

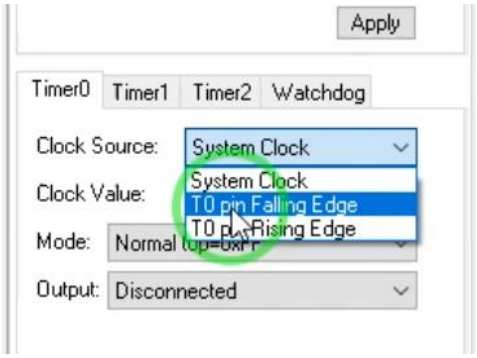






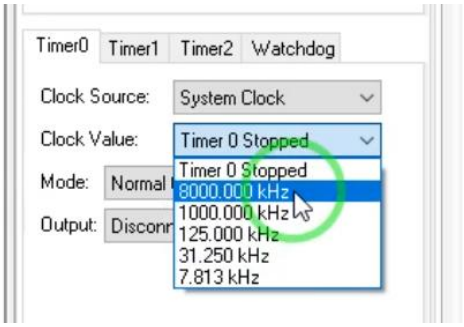
Timer0 , timer 2 ، ۸بیتی هستند.

Timer1 => ۱۶ بیتی است.

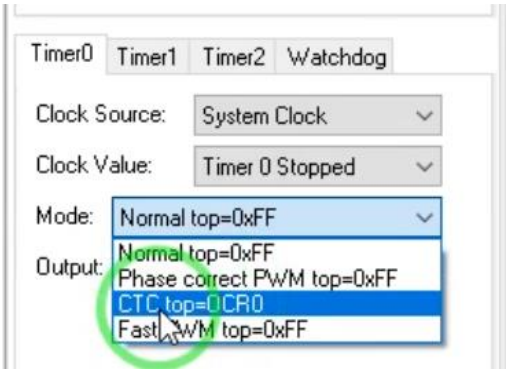


کلاک تایمر
۱. میتواند کلاک داخلی میکرو (همان ۸ مگاهرتز) باشه و یا ضریبی از اون ۸مگاهرتز باشه
۲. از سیگنالی که به پایه ی T0 متصل است، استفاده کنیم. (یه منبع خارجی داشته باشیم که به میکرو وصلش کنیم و از اون کلاک، به عنوان تایمر استفاده کنیم.)
اگه از این پایه T0 استفاده کنیم => به عنوان شمارنده میتوانیم از تایمر، استفاده کنیم.
اگه دوره تناوب این سیگنال را ندانیم => میتوانیم این تایمر را به عنوان شمارنده درنظر بگیریم.
ولی اگه مطمئن هستیم که دوره تناوبش چی هست => میشه به عنوان تایمر یا زمان سنج ازش استفاده کرد.

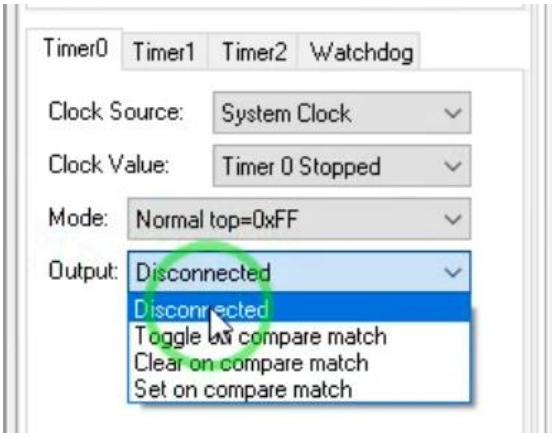
اگه از کلاک داخلی استفاده میکنیم => یه مضربی از ۸ مگاهرتز میتوانه کلاک سیستم بشه:
ضریب ها:۱ و ۱/۸ و ...



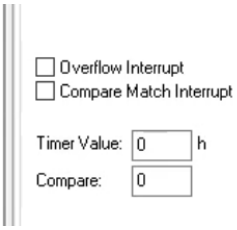
مدهای مختلف در تایمر



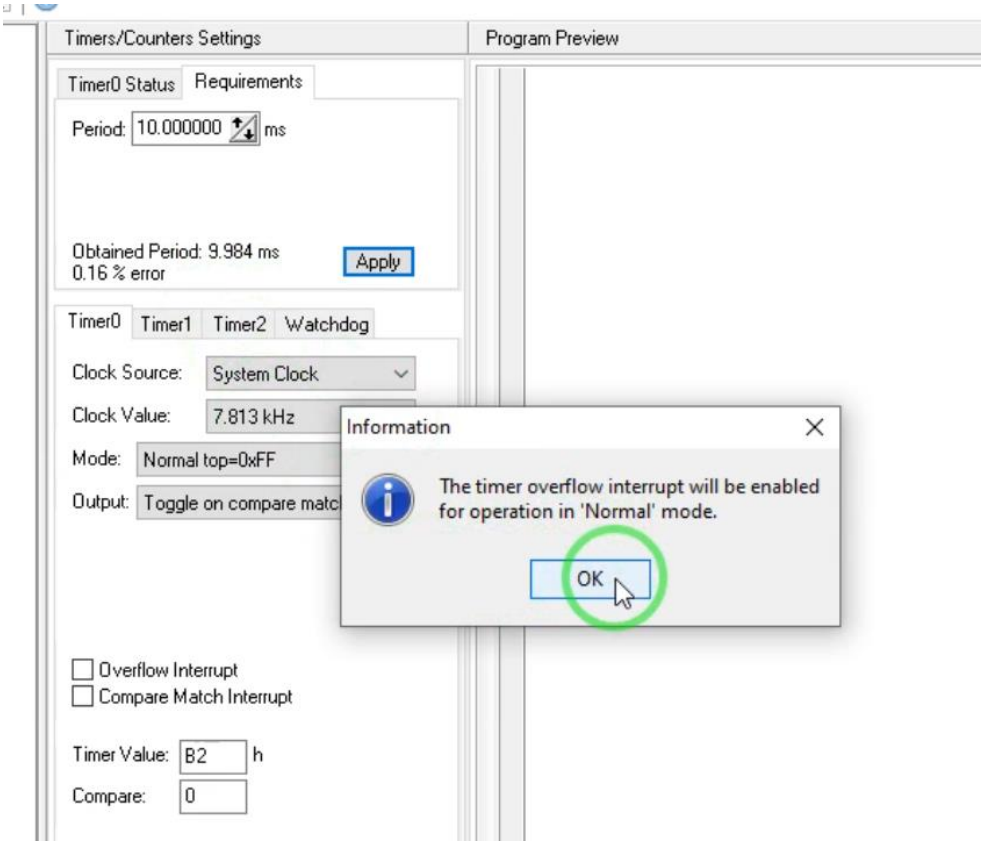
خروجی تایمر
اگر تاگل باشد=> در خروجی یک شکل موج خواهیم داشت.



فعال کردن اینتراپت در تایمر و یا حالت compare match
تعیین مقدار اولیه تایمر
تعیین مقداری که باید مقایسه شود با مقدار تایمر که اگر مساوی این مقدار بشه => در زیربرنامه ی مناسب آن، میشه یه کاری انجام داد.



اگه بخاهیم یه فرکانسی با دوره تناوب 10 میلی ثانیه ایجاد کنیم =>



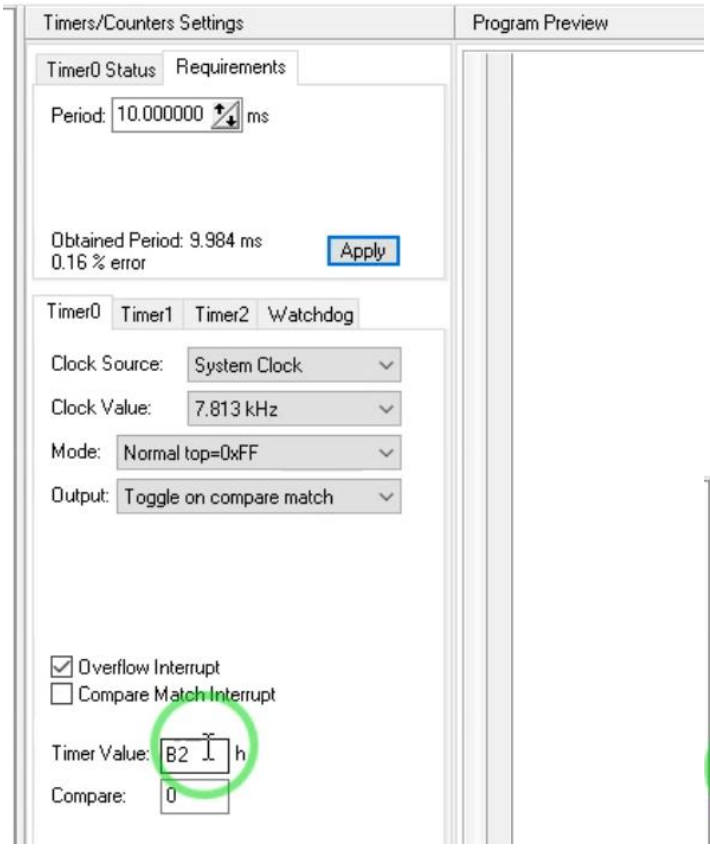
تغییراتی که خودکار رخ داد:

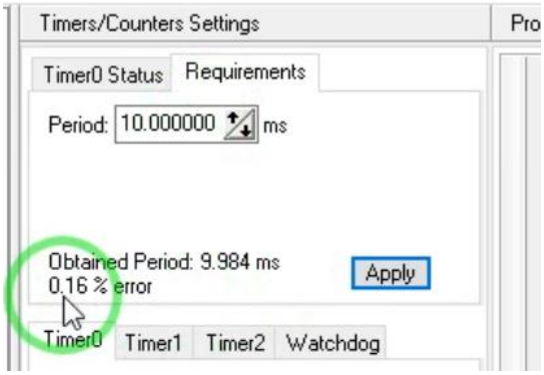
۱. فرکانس تایمر مقدار 7.813khz انتخاب میشه

۲. بیت overflow interrupt را فعال میکنه

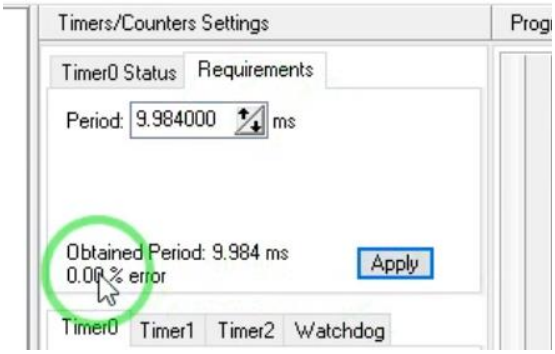
۳. مقدار اولیه ای که به تایمر اختصاص داده : B2 است. که ما در زیربرنامه ی اورفلو، ما باید مرتب این مقدار را بهش اختصاص بدیم.

اگه این کار را نکنیم => در لحظه ی اول مقدار B2 را داره، وقتی که اورفلو رخ میده مقدارش صفر میشه و دوباره تا ۲۵۵ میره تا اورفلو کنه (از عدد B2 باید شروع کنه تا 0xff یا همان ۲۵۵ بره و بعدش اورفلو کنه) تا بتواند دوره تناوب ۱۰میلی ثانیه را ایجاد کنه.

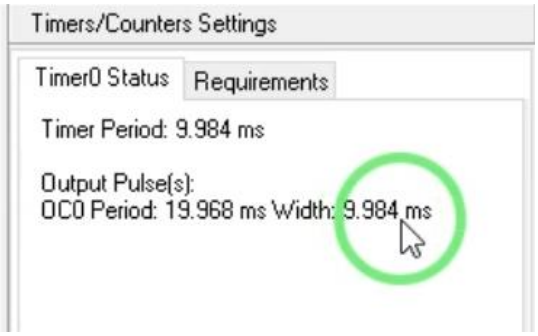




میکرو میتواند تا عدد 9.984 را به صورت دقیق ایجاد کنه که نسبت به عدد 10 میلی ثانیه، 16 درصد خطا داره. برای صفر کردن خطا => همان عددی که داره را میگذاریم.



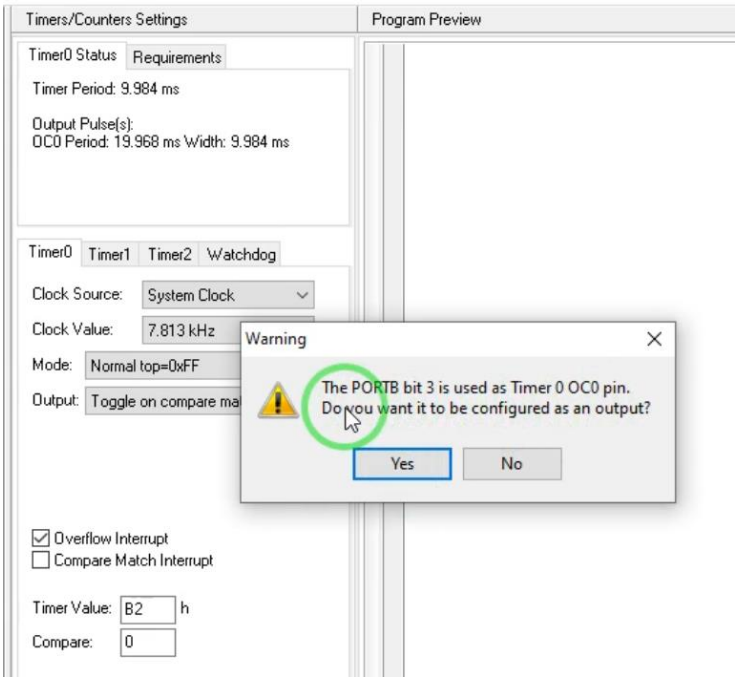
پس تایمر ما، در بازه های تقریبا ۱۰ میلی ثانیه ای، اینتراپت میده



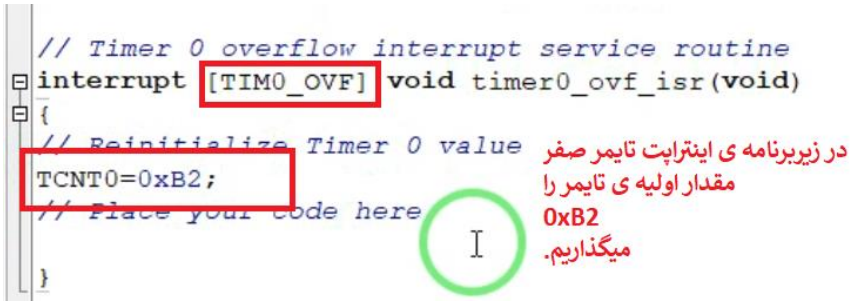
پس ما یک شکل موجی را در پایه خروجی تایمر صفر خواهیم داشت که دوره تناوبش : 19ms است و عرضش هم 9.984میلی ثانیه.

Duty cycleش هم، 50% است. یعنی از صفر تا 9.984میلی ثانیه در حالت high است و از 9.984میلی ثانیه تا 19ms در حالت low است.

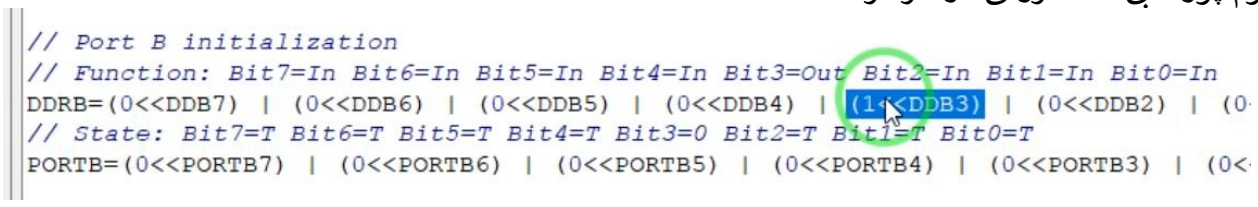
نکته ی مهم:
اگر میخاهیم از پایه ی output یک تایمر استفاده کنیم باید اون پایه را در میکرو، به صورت خروجی تعریف کنیم.
چون OCR0 در بیت PORTB.3 است => باید خروجی بشه.



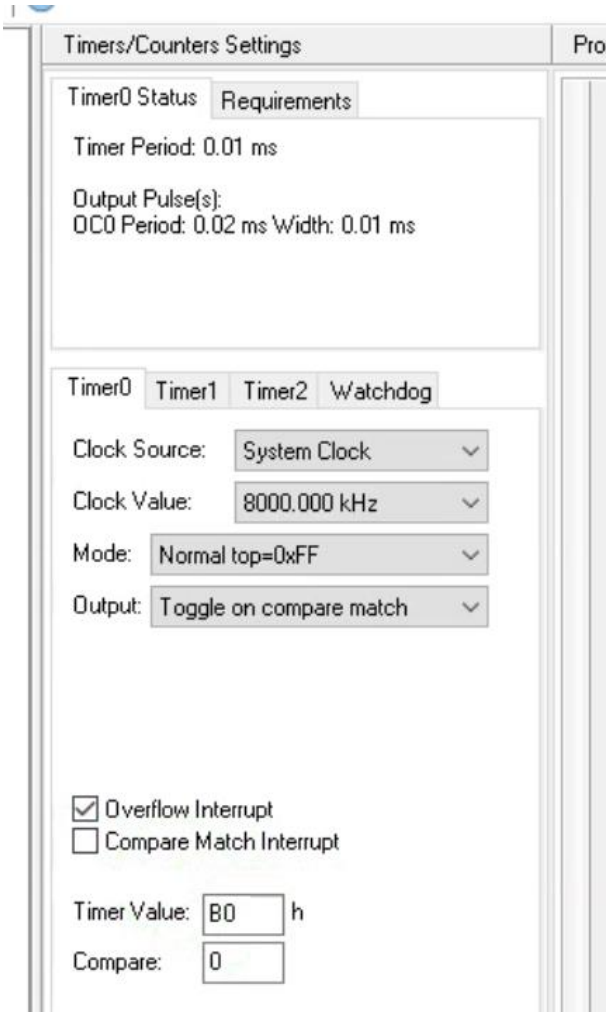
این برنامه در فضایی از میکرو ذخیره میشه که مربوط به TIMER0_OVERFLOW است.
میدانیم که به ازای هر وقفه ای، یه فضایی از میکرو درنظر گرفته میشه که کدها در ان موقعیت قرار میگیرند و هرموقع اینتراپت اتفاق افتاد، کدها از آن نقطه شروع به اجرا میکنند.



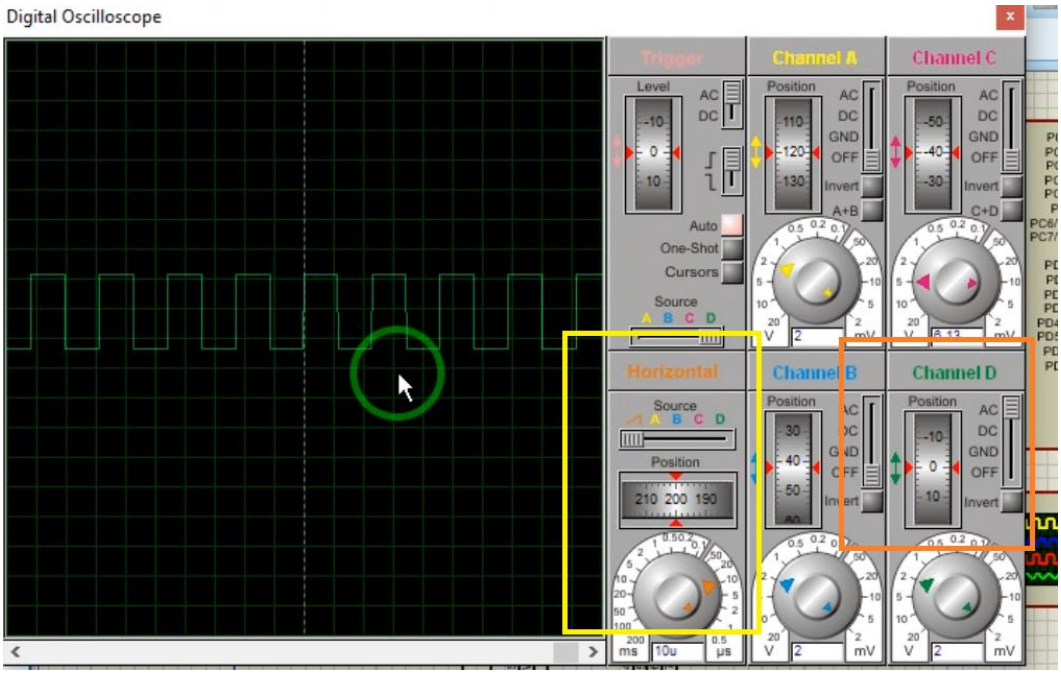
بیت سوم پورت بی => خروجی درنظر گرفته شده.



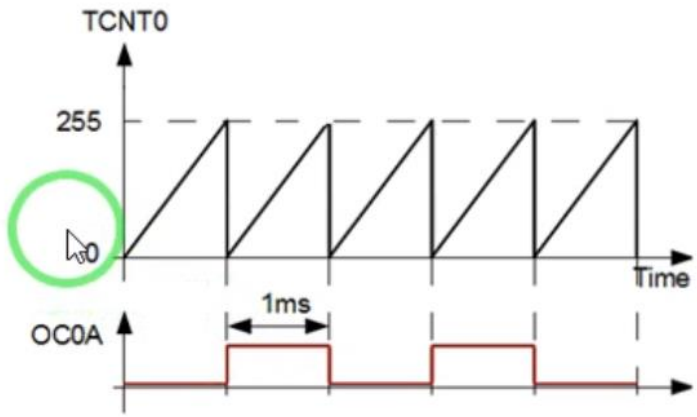
در پروتئوس نمیشه به جز فرکانس های ۸مگا و ۱مگا، از فرکانس دیگری استفاده کرد.
اگه دوره تناوب را 0.01ms درنظر بگیریم:



نمایش موجود تولید شده توسط پایه خروجی تایمرصفر



در مدنرمال



شکل 5-3- نحوه‌ی شمارش و فعال شدن بیت OC0A در حالت شمارش Normal

از عدد صفر تا ۲۵۵ را میشماره به عدد ۲۵۵ که برسه => اینتراپت اورفلو میده و سرریز میشه => سیگنال خروجی تاگل میشه (اگه صفر بوده، یک میشه و اگه یک بوده، صفر میشه)

مقداراولیه را میشه به جز صفر درنظر گرفت.
در مد نرمال، شکل موجی که در خروجی ایجاد میشه ، همواره duty cycle برابر با 50% است.
همواره تایمر از یک مقدار اولیه ای، تا ۲۵۵ (یا همان 0XFF) را میشماره و بعد، اورفلو میشه.

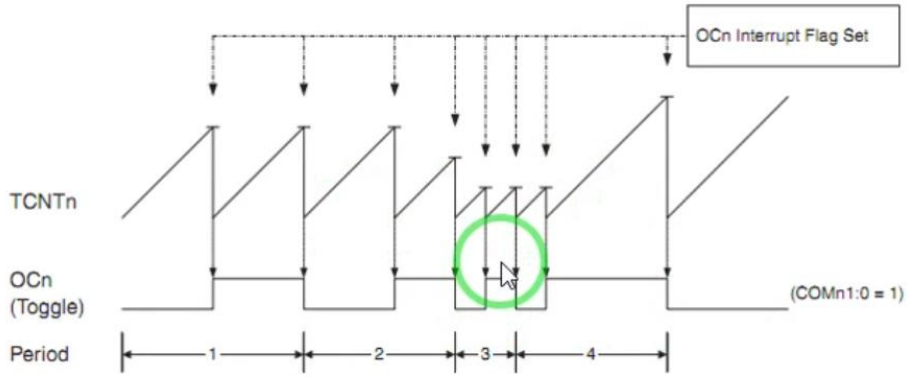
در حالت CTC

5.3.2 حالت Clear Timer on Compare Match Mode (CTC)

در این حالت محتوای شمارنده تایمر(TCNTn) با محتوای ثبات OCRnx (n صفر، یک یا دو- x: A یا B)مقایسه می‌شود و در صورتی که برابر باشند مقدار شمارنده برابر صفر می‌شود. در این حالت مقدار مقدار OCRnx به عنوان حد ماکزیمم شمارش در نظر گرفته می‌شود. لذا پس از برابری TCNTn با OCRnx سرریز اتفاق می‌افتد و پایه خروجی OCn مانند شکل 5-4تغییر 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:

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

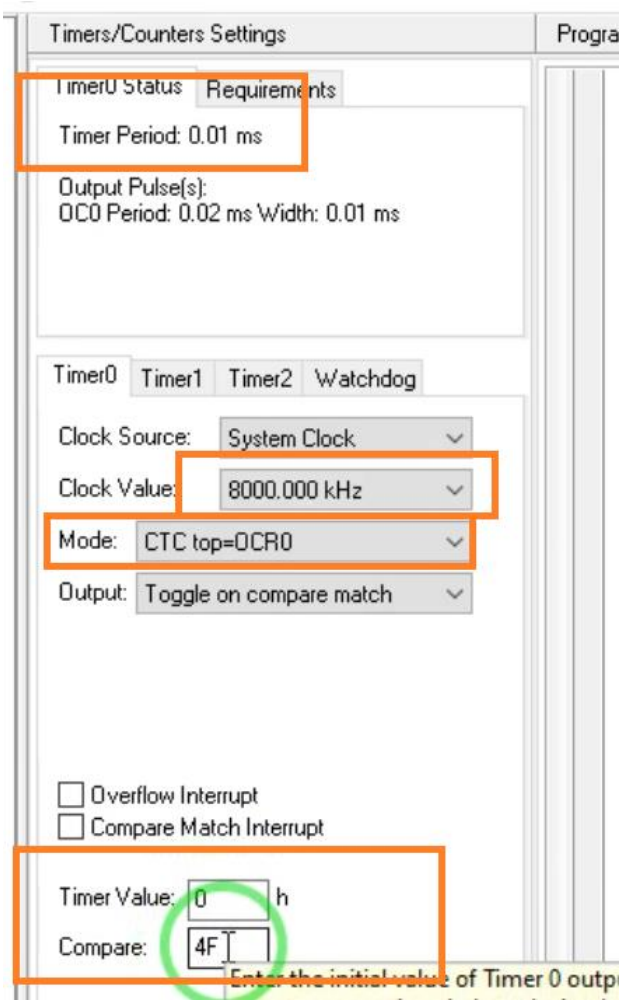
عملیات شمارش را از صفر تا مقداری که در رجیستر OCRnx هست (که اینجا میشه OCR0) میشماره و بعدش اینتراپت رخ میده.



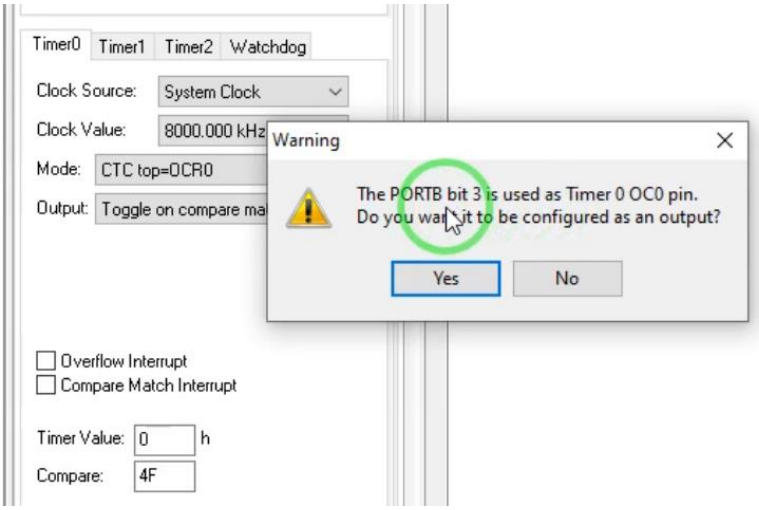
شکل 5-4: تعیین حد بالای شمارش با استفاده از ثبات OCRxy

شکل موج ایجاد شده در خروجی
مدت زمانی که در وضعیت high است با مدت زمانی که در وضعیت low است، برابر است.
شکل موج ایجاد شده در بالا، چون مدام داره مقدار OCRn را تغییر میده، شکل موج هم تغییر میکنه اگه نه duty cycle همان 50% است.

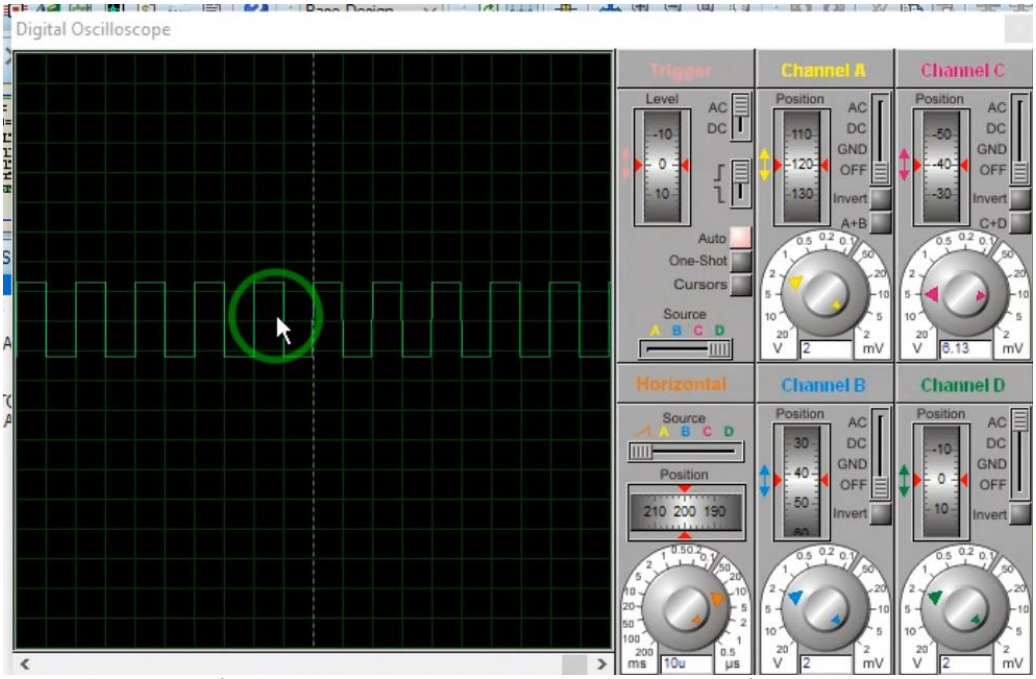
اگر دوره تناوب را 0.01ms بگذاریم و از مد CTC بخاهیم استفاده کنیم:
خود کدویزارد، فرکانس ۸مگاهرتز را انتخاب میکنه و مقدار اولیه ی 4F = OCR0 میگذاره.
از صفر تا 4F میشماره و بعدش تاگل میکنه سیگنال خروجی را.



بیت سوم از پورت بی را یک میکنه (بیت خروجی).



در اسیلوسکوپ، نتیجه بازهم یک شکل موج میشه.
اینجا باز هم duty cycle برابر 50% است.
اینجا دیگه نیاز نیست از وقفه های تایمر استفاده کنیم.



چون اینجا مقدار اولیه ی OCR0 را برابر 0X4F میگذاریم خودش چون در مد CTC قرار داره دیگه نیازی به فعال کردن وقفه نداره.

```
// Clock source: System Clock
// Clock value: 8000.000 kHz
// Mode: CTC top=OCR0
// OC0 output: Toggle on compare match
// Timer Period: 0.01 ms
// Output Pulse(s):
// OC0 Period: 0.02 ms Width: 0.01 ms
TCCR0=(0<<WGM00) | (0<<COM01) | (1<<COM00) | (1<<WGM01) | (0<<CS02) | (0<<CS01) | (1<<CS00);
TCNT0=0x00;
OCR0=0x4F;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCI
```

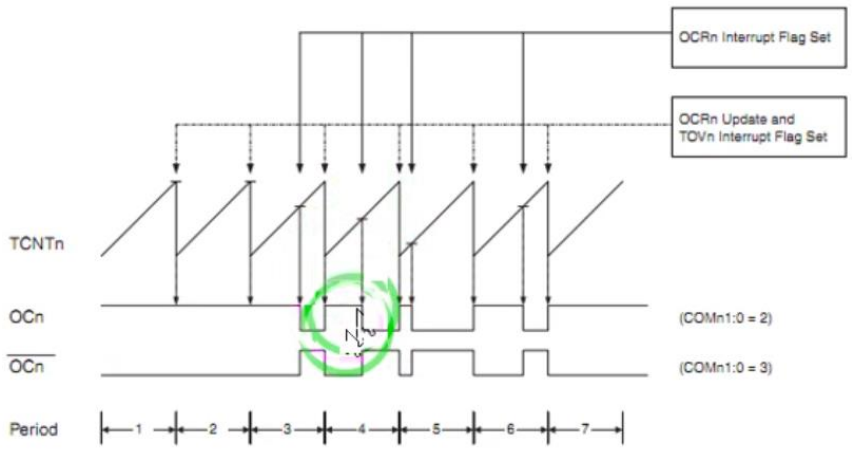
مد FAST PWM

5.3.3 حالت Fast PWM

حالت Fast PWM به عنوان ترکیبی از حالت Normal و CTC در نظر گرفته می‌شود. زیرا مانند حالت Normal شمارنده از صفر تا حد ماکزیمم آن 0xff و یا 0xffff می‌شمارد و مانند حالت CTC ، مقدار ثبات TCNTn همواره با ثبات OCRn مقایسه می‌شود و پایه خروجی OCn در هنگام سرریز تایمر و برابری ثباتهای TCNT و OCR مانند شکل 5-5 تغییر وضعیت می‌دهد. در این حالت امکان استفاده از دو وقفه تایمر شامل وقفه سرریز و وقفه Compare Match وجود دارد.

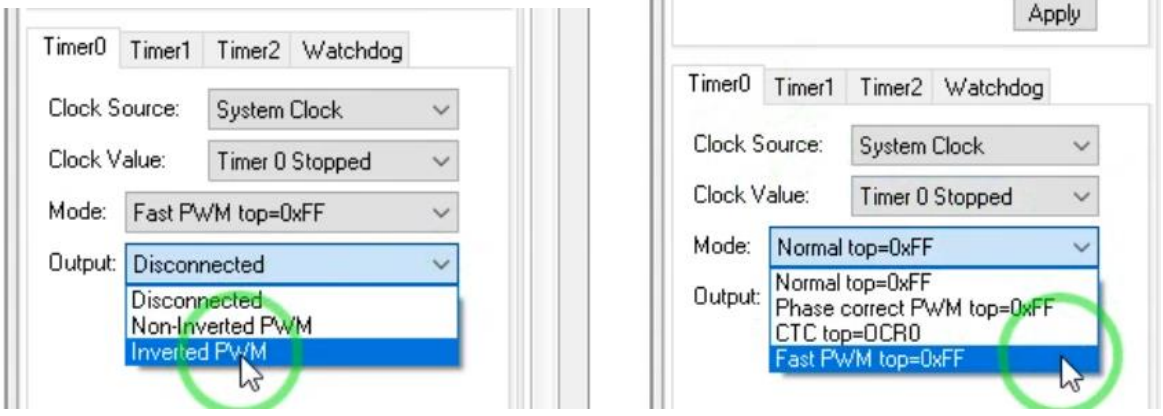
این مد، ترکیبی از مدهای نرمال و CTC است. چون مثل مد نرمال، از صفر تا 0XFF میشماره و مثل مد CTC مقدار ثبات TCNT0 با رجیستر OCR0 مقایسه میشه و اگه برابر باشند => وضعیت خروجی را تاگل میکنه.
یعنی تاگل کردن سیگنال خروجی، در دو لحظه اتفاق میفته:
۱. مقدار رجیستر TCNT0 به OCR0 میرسه
۲. زمانی که به 0XFF میرسه

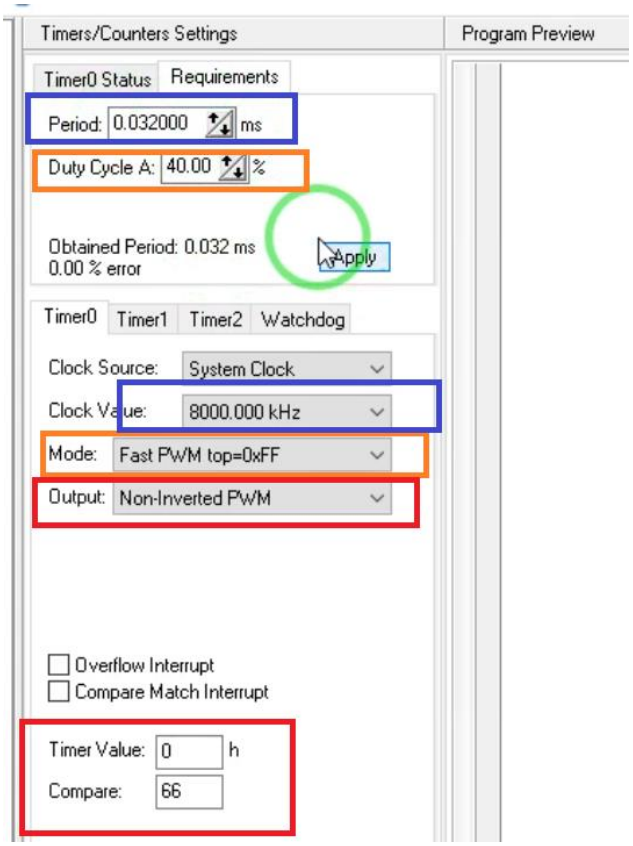
پس دوتا تغییر وضعیت خروجی در شمارش از صفر تا 0XFF داریم. این کار باعث میشه سیگنال هایی با فرکانس بالاتر تولید کنیم.
برای همین بهش FAST PWM میگن



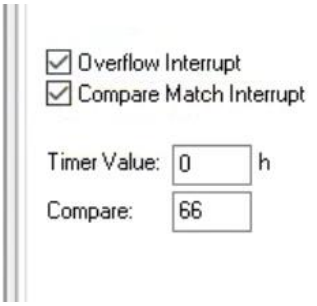
شکل 5-5: نحوه‌ی فعال شدن OCn در حالت Fast PWM

تنظیمات مربوط به FAST PWM در کدویژن
۱. انتخاب مد FAST PWM
۲. سیگنال خروجی را inverted یا non-inverted انتخاب میکنیم.
۳. باید دوره تناوب پالس را انتخاب کنیم
۴. انتخاب duty cycle پالس خروجی





در فرکانس ۸ مگاهرتز، با مقداراولیه ی OCR0 = 0X66
تایمر کانتر از مقدار صفر تا 0XFF میشمارد. و لحظه ای که به 0X66 برسه، خروجی را تاگل میکنه و زمانی که به 0XFF هم، خروجی را تاگل میکنه.
میتوانیم وقفه های اورفلو و COMPARE MATCH را هم فعال کنیم و یه عملیاتی ک میخاهیم را در زیربرنامه هاشون، انجام بدیم.



زیربرنامه ی مربوط به TIMER0_OVERFLOW

```
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here

}
```

زیربرنامه ی مربوط به TIMER0_COMP

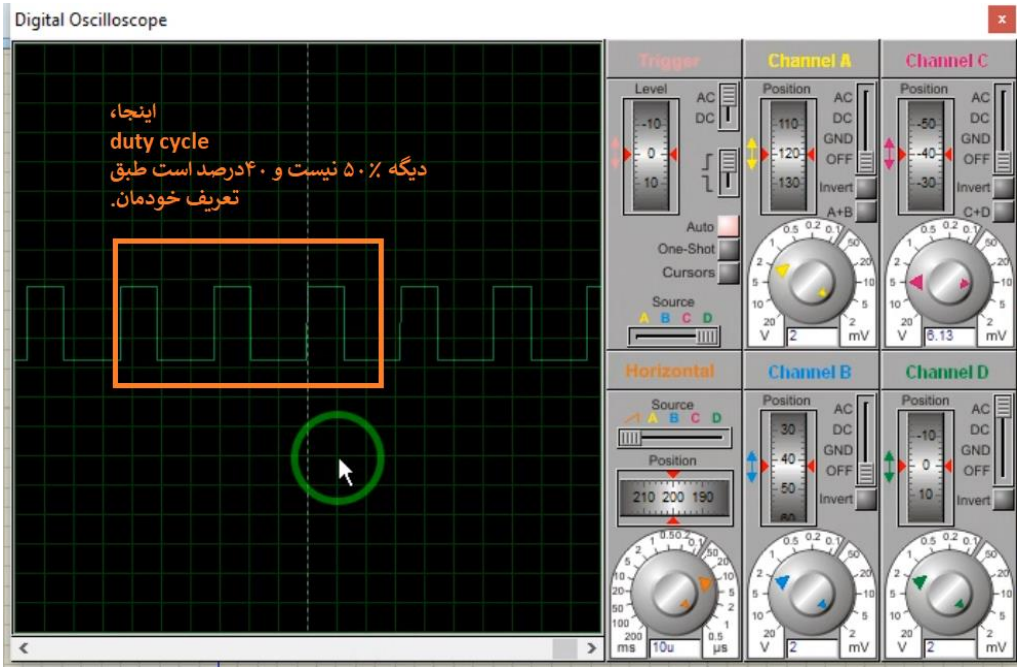
```
// Timer 0 output compare interrupt service routine
interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
// Place your code here

}
```

```
TCCR0=(1<<WGM00) | (1<<COM01) | (0<<COM00) | (1<<WGM01) | (0<<CS02) | (0<<CS01) | (1<<CS00);
TCNT0=0x00;
OCR0=0x66;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (1<<OCIE0);

// Global enable interrupts
#asm("sei")
```



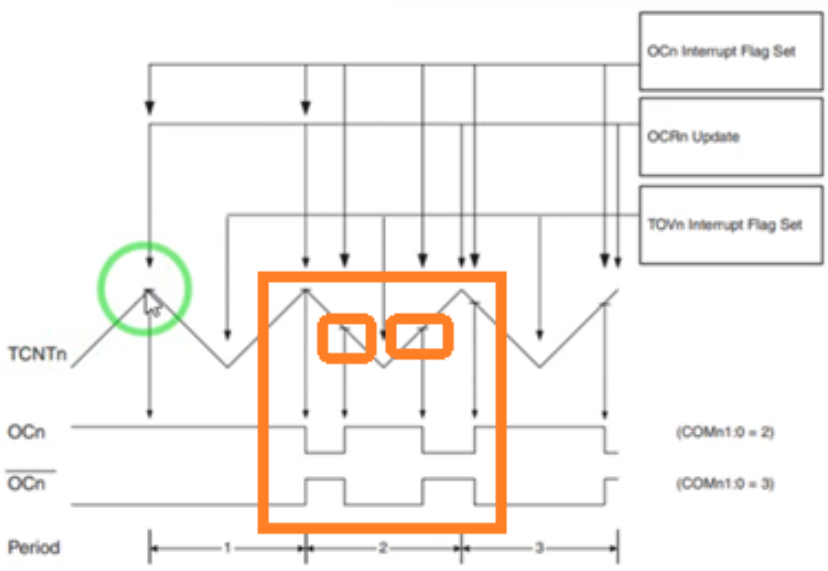
یکی از تفاوت هایی که بین مد fast pwm با مدهای نرمال و ctc وجود داره:
۱. در حالت نرمال و ctc، duty cycle برابر 50% است.
ولی در روش fast pwm، میتوانیم مقدار duty cycle را در رنج های مختلفی، تغییر بدیم.
۲. در مد fast pwm، میتوانیم فرکانس های بالاتری را ایجاد کنیم (افزایش سرعت) و شبیه سازی کنیم.

مد phase correct pwm

5.3.4 حالت Phase Correct PWM

در حالت Phase Correct PWM قابلیت ایجاد موج PWM با تصحیح فاز انجام می‌شود. 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 با صفر شدن TCNTn رخ می‌دهد و وقفه CompareMach با برابری مقدار TCNTn و OCRn اتفاق می‌افتد. از طرفی همواره هنگامی‌که TCNTn برابر با حد ماکزیمم می‌باشد مقدار OCRn به روز می‌گردد.

شکل موج با تصحیح فاز انجام میشه.



شکل 5-6: نحوه فعال شدن پایه‌ی OCn در حالت Phase Correct PWM

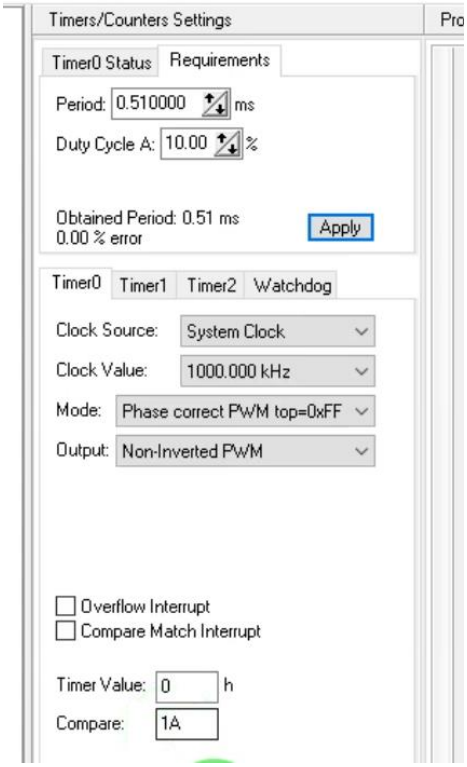
تایمر کانتر از مقدار صفر تا 0xff به صورت صعودی میشماره و از مقدار0xff تا صفر تغییر وضعیت میده و به صورت نزولی میشماره. در این بین، همواره مقدار تایمر کانتر با مقدار OCR مقایسه میشه.
وقتی که مساوی با مقدار OCR بشه، سیگنال خروجی، تاگل میشه. (تغییر وضعیت میده)

در این روش میشه بازه های زمانی طولانی تری را پیاده سازی کنیم.

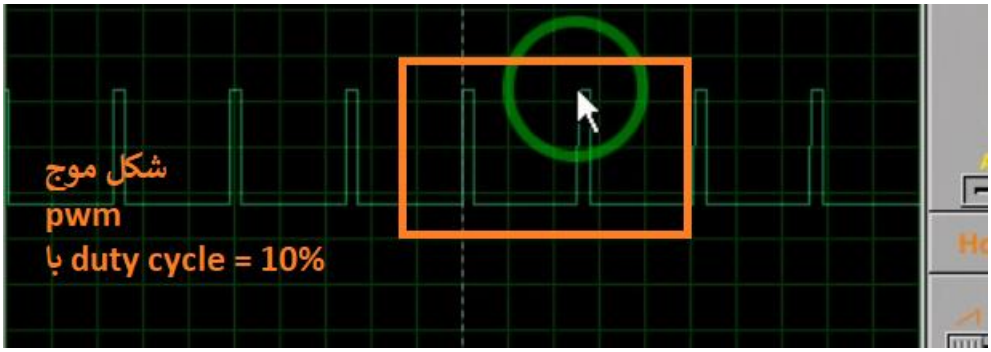
تنظیمات برای مد phase correct pwm

اگر دوره تناوب را 0.051ms و duty cycle را 10% درنظر بگیریم:

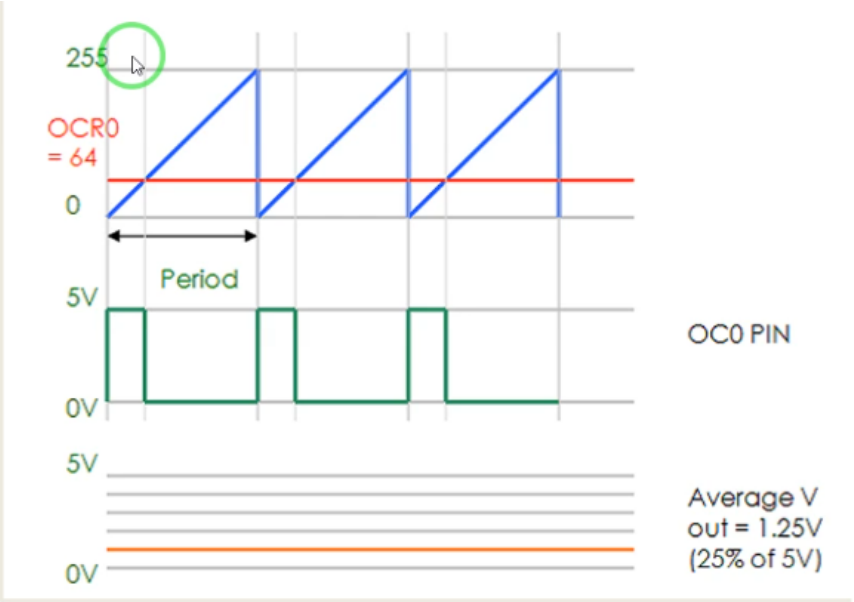
۱. کلاک یا فرکانس میکرو به صورت خودکار مقدار 1MHz انتخاب میشه
۲. خروجی را non-inverted pwm گذاشتیم.
۳. مقدار اولیه رجیستر OCR0 = 0X1A میشه



اینجا دیگه وقفه های تایمر را فعال نکردیم و شکل موجی که در خروجی ایجاد میشه به کمک همان مقایسه با مقدار 0x1A انجام میشه.

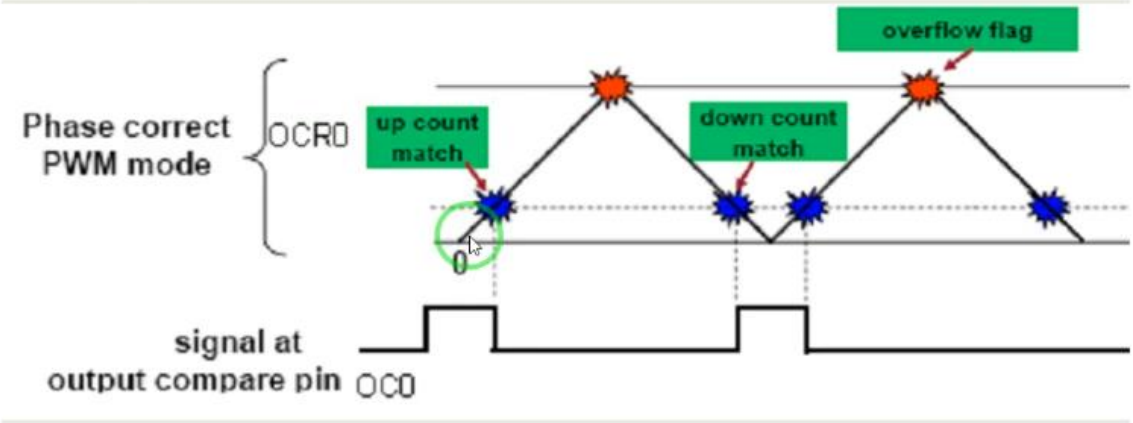


Fast PWM

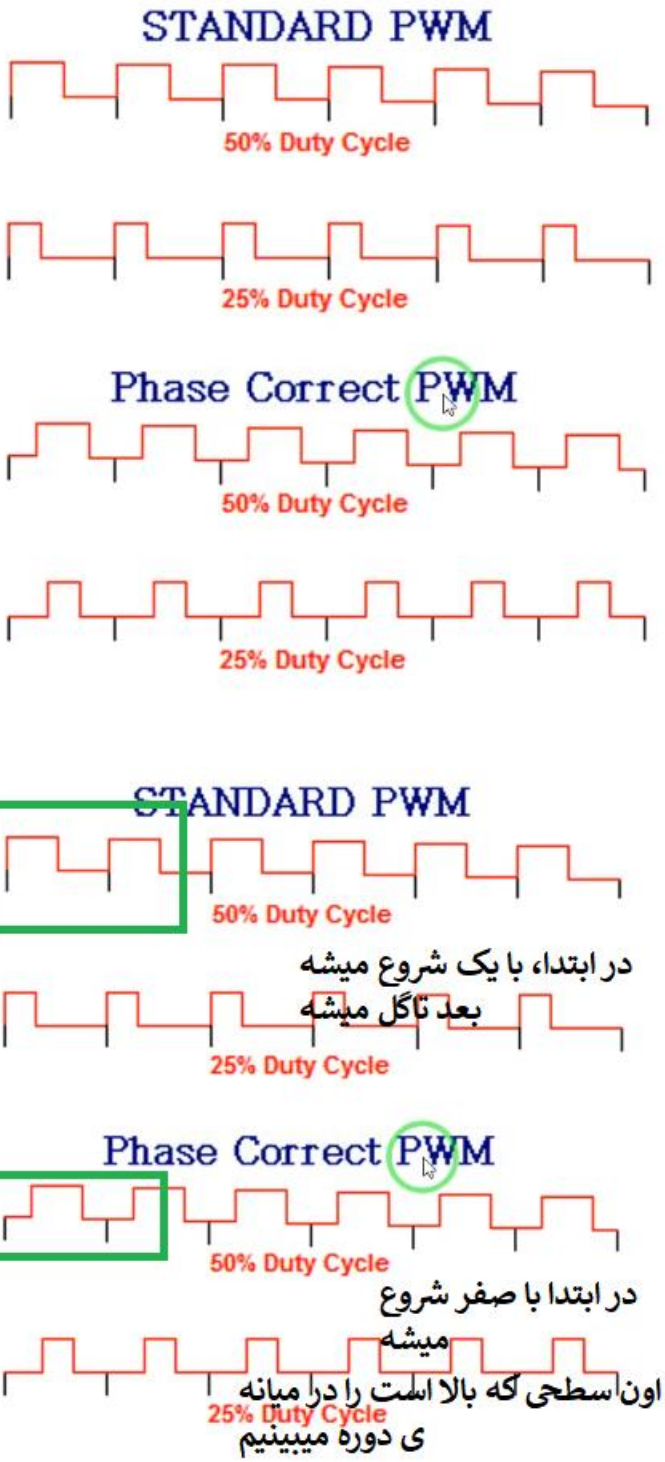


اگر مقدار OCR0 = 64 باشه که یک چهارم مقدار ۲۵۵ است انتخاب کنیم میتوانیم در خروجی یک شکل موجی را داشته باشیم که duty cycle = 25% است.
ولتاژی که در موج خروجی تولید میشه: ۲۵٪ * ۵ولت میشه که ۱/۲۵ ولت ایجاد میشه
پس با امواج pwm میتوانیم سطح موج dcای که در خروجی ایجاد میشه را تنظیم کنیم.
از امواج pwm برای کنترل دور موتور های dc استفاده میشه. (افزایش ولتاژ Dc => موتور با دور بیشتری حرکت میکنه)

Phase Correct PWM



از 0 تا 0xff به صورت صعودی
از 0xff تا 0 به صورت نزولی
اگه به مقدار OCR0 برسه => تاگل میکنه خروجی را.
پس سیگنال خروجی، فقط با مساوی شدن TCNT0,OCR0 تاگل میشه.
پس میتوانیم فاصله زمانی های بیشتری را ایجاد کنیم => یه سیگنال با فرکانس پایین ایجاد کنیم.



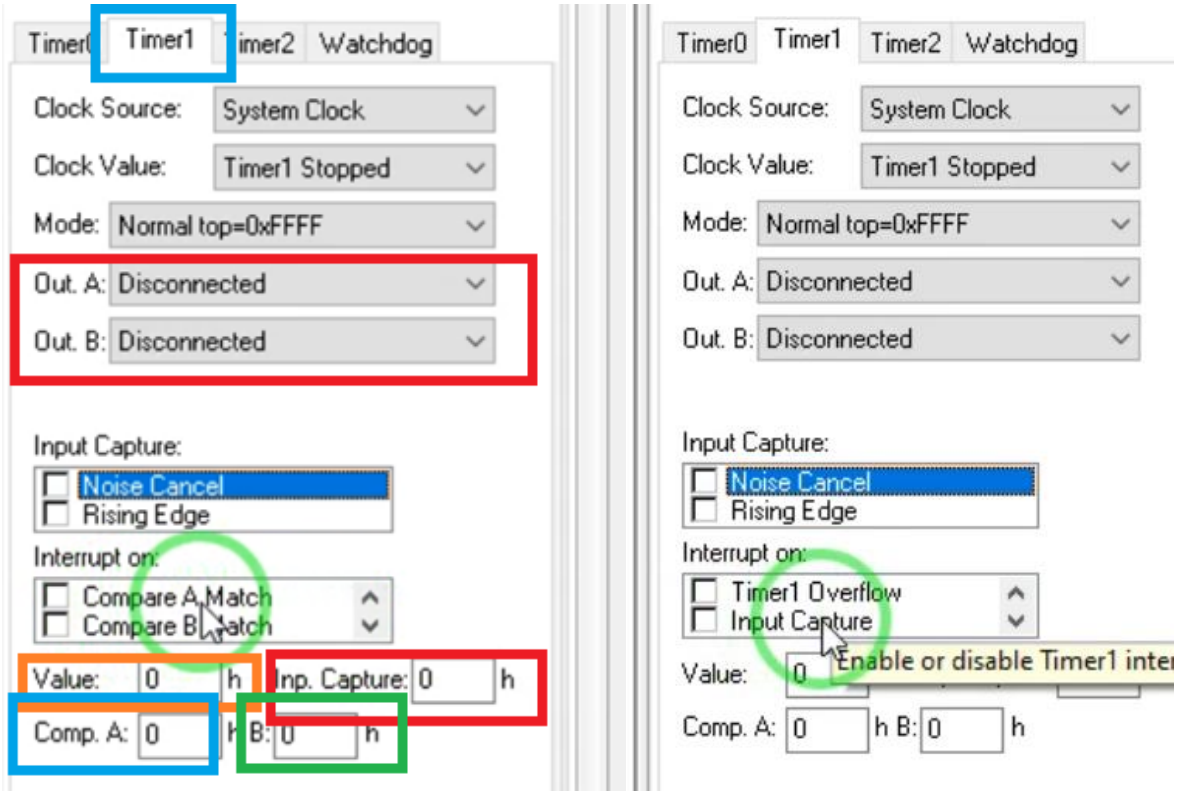
مقایسه بین fast pwm , phase correct pwm
در fast pwm ، ابتدا سطح یک سیگنال را میبینیم و بعدش صفر میشه.

در phase correct pwm ,شروع و پایان دوره تناوب، صفر است و در میانه به صورت یک قرار داره.

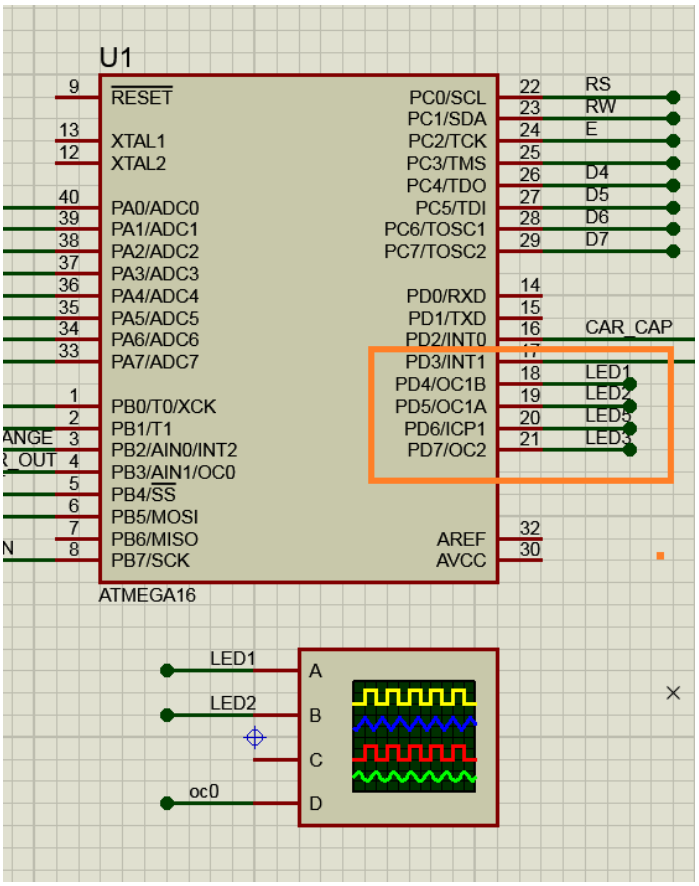
در تایمر1 امکان استفاده از input capture وجود داره. که برای اندازه گیری دور موتور Dc کاربرد داره.
تایمریک، چهارتا interrupt داره.

۱. timer1 overflow
1. input capture
۳. compare A match

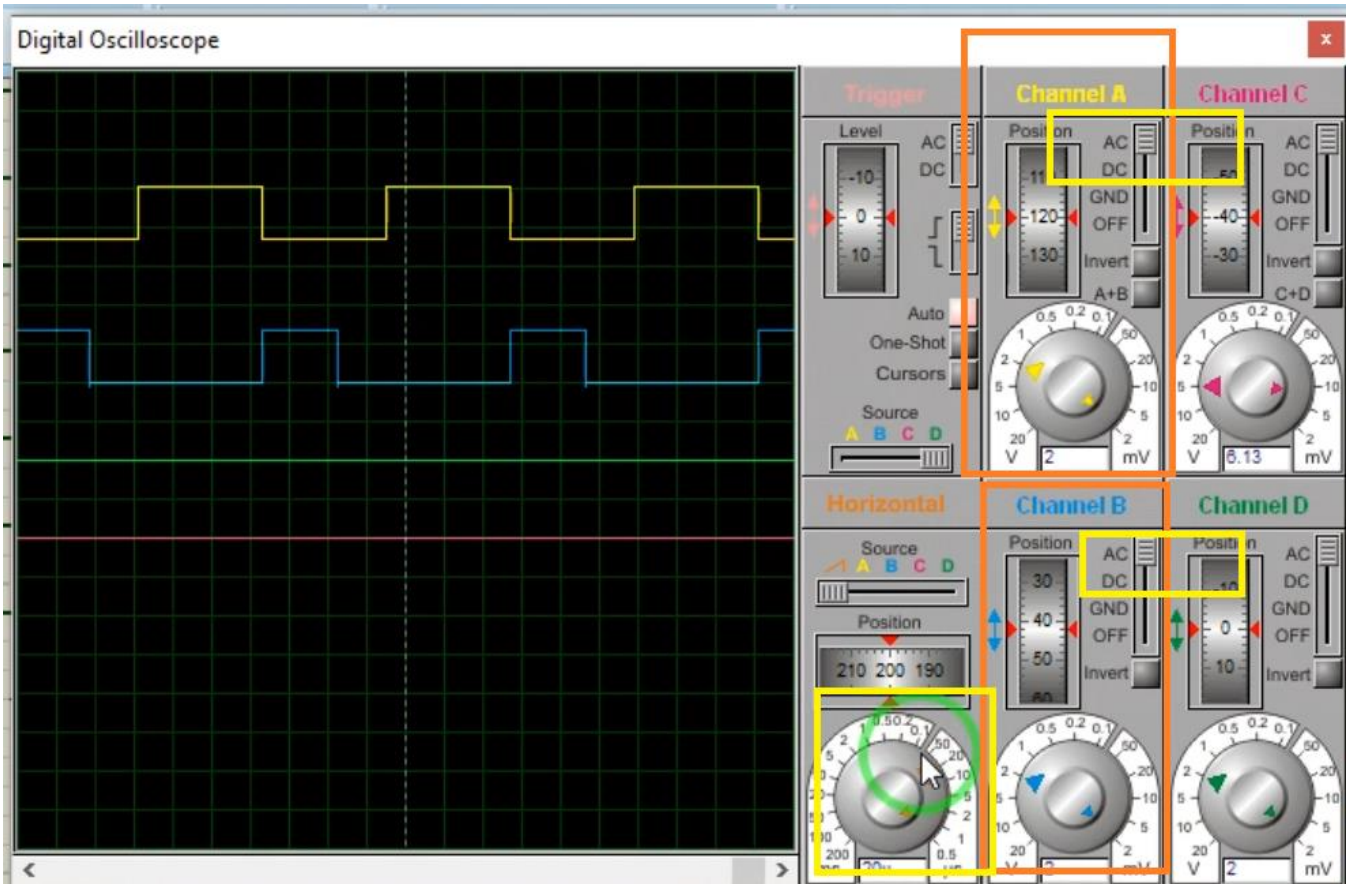
compare B match .۴

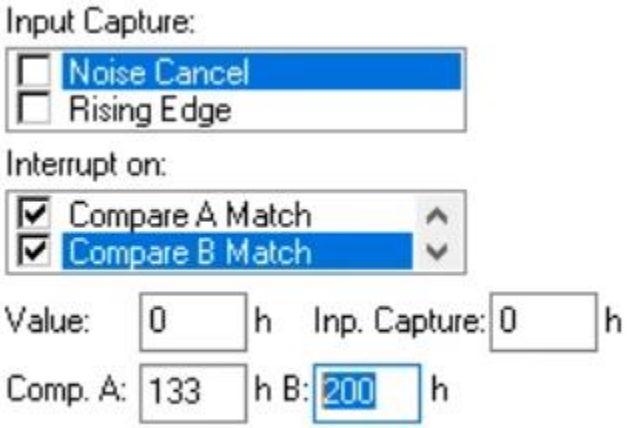


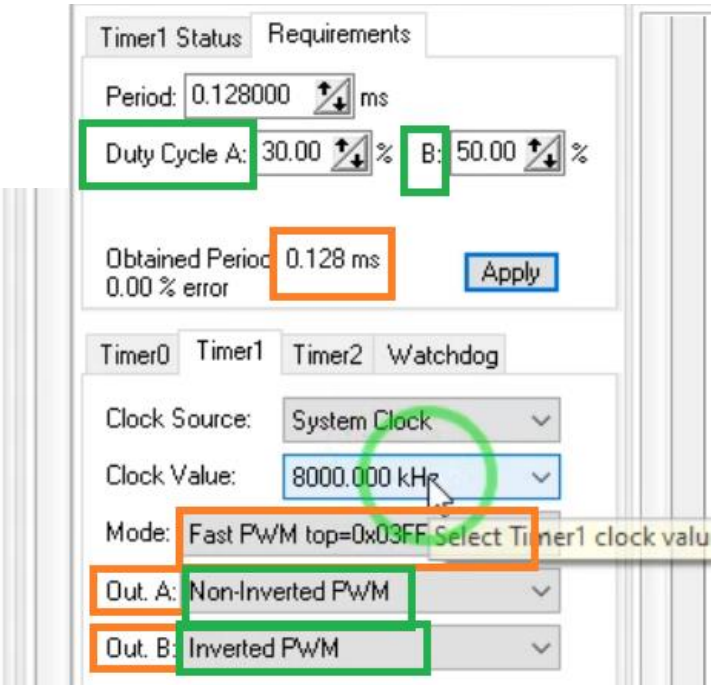
با دوتا رجیستر میتوانیم دو تا مقدار را مقایسه کنیم با مقدار تایمر کانتر
یکی با رجیستر,OCRA دیگری باOCRB



پورت های خروجی OC1A,OC1B از تایمریک ایجاد میشوند.
که پالس خروجی شان را به اسکیلوسکوپ نمایش میدهیم.

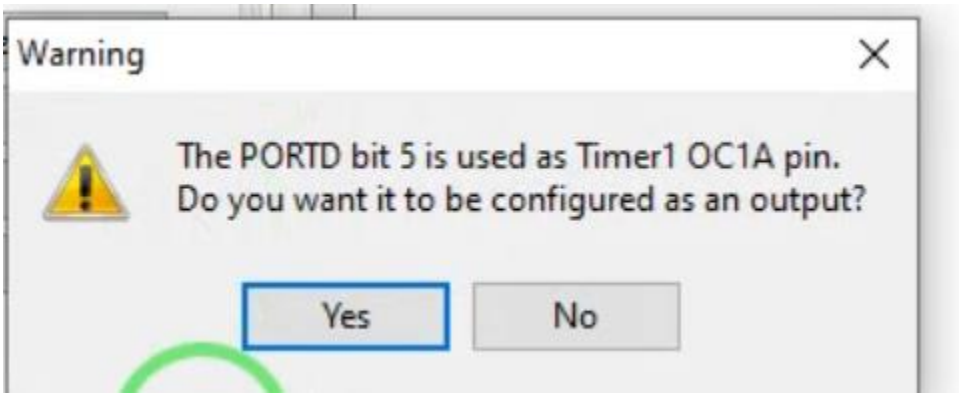


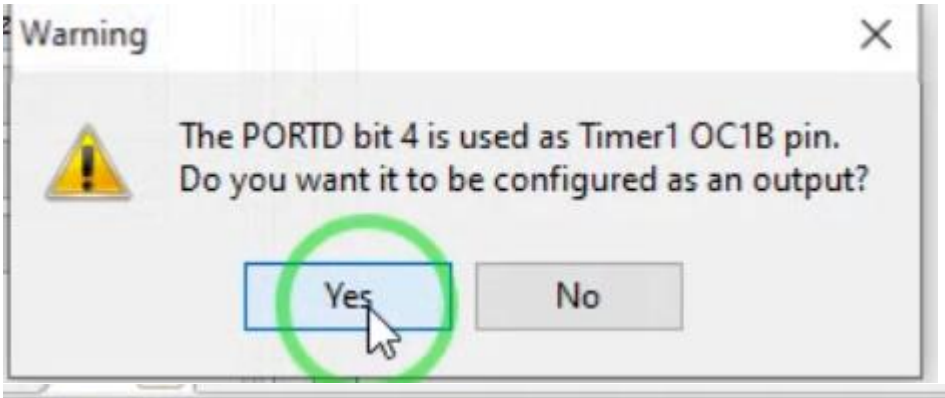




مقدار اولیه ی رجیستر OCRA,OCRB را میبینیم. (۱۳۳و۲۰۰)

باید بیت هایی که مربوط به خروجی تایمریک هستند را یک کنیم (خروجی تعریف کنیم)






```
// Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
// Place your code here

}
```

```
// Timer1 input capture interrupt service routine
interrupt [TIM1_CAPT] void timer1_capt_isr(void)
{
// Place your code here

}
```

```
// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here

}

// Timer1 output compare B interrupt service routine
interrupt [TIM1_COMPB] void timer1_compb_isr(void)
{
// Place your code here

}
```