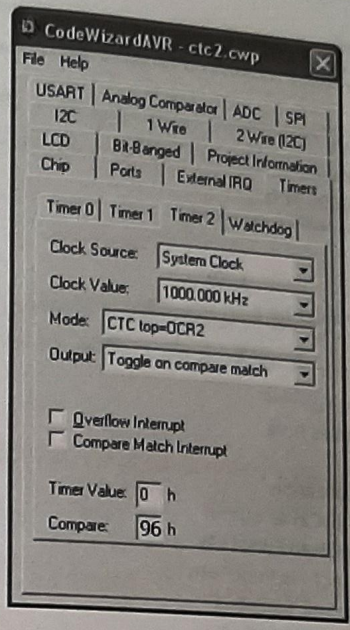


تنظیمات ابزار CodeWizardAVR برای برنامه ctc2.c

- پالس ساعت تایمر همان پالس ساعت میکروکنترلر باشد (مطابق جدول (۶-۲)).
- چون بیت WGM21 برابر 1 است، پس مطابق جدول (۶-۴) تایمر به‌صورت CTC کار می‌کند.
- چون بیت COM20 مساوی 1 است، پس مطابق جدول (۶-۵) پالس مربعی در خروجی OC2 تولید می‌شود.

◆دستور (۴): مقدار اولیه تایمر TCNT2 را برابر 0 می‌کند.

◆دستور(۵): مقدار اولیه ثبات OCR2 را برابر 150 قرار می‌دهد.

- **بخش C:**

پالس مربعی در خروجی OC2 تولید می‌شود و عملیات تکرار می‌گردد.

فرکانس پالس CTC در پایه OC2 مطابق فرمول (۳) برابر است با:

$$f_{OC2} = \frac{\text{فرکانس پالس تایمرمیکروکنترلر}}{2N(1+OCR2)} = \frac{1000\,KHz}{2\times1(1+150)} \approx 3.3\,KHz$$

فایل با پسوند HEX برنامه توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.

◆ ۶-۷: انتخاب پالس ساعت تایمر 2 و دیاگرام زمانبندی[1] در میکروکنترلرهای AVR

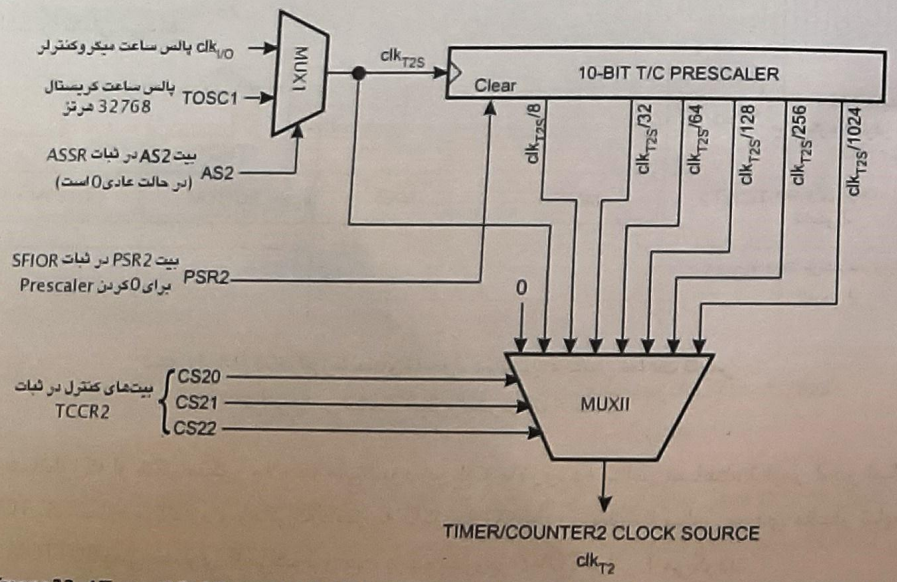
همانطورکه از قسمت انتخاب پالس ساعت تایمر 2 (شکل (۶-۱)) و شکل (۶-۱۴) مشاهده می‌شود، پالس ساعت تایمر 2 (clk_{T2}) ممکن است توسط مولتی پلکسر MUX1:

- از خارج میکروکنترلر با اتصال کریستال 32768 هرتز در پایه‌های TOSCI و TOSC2 برای تولید پالس ساعت RTC[2] تأمین شود.
- یا به‌طور پیش‌فرض توسط پالس ساعت میکروکنترلر تأمین گردد.

خروجی قسمت انتخاب پالس ساعت به‌عنوان پالس ساعت تایمر می‌باشد که تایمر آن را می‌شمارد.

همانطورکه از جدول (۶-۲) و شکل (۶-۱۴) مشاهده می‌شود، با انتخاب مقادیر CS22, CS21, CS20 در ثبات کنترل TCCR2 ، پالس ساعت تایمر 2 مستقیماً خروجی مولتی‌پلکسر MUX1 ، یعنی پالس ساعت میکروکنترلر یا از کریستال TOSC1 گرفته می‌شود و یا پالس ساعت تایمر 2 ، توسط Prescaler و MUXII ، برابر پالس ساعت خروجی مولتی‌پلکسر MUX1 تقسیم بر 8, 32, 64, 128, 256 یا 1024 می‌شود.

شکل (۶-۱۴) ساختار اصولی Prescaler و چگونگی انتخاب پالس ساعت تایمر 2 توسط بیت‌های CS22, CS21, CS20 و مولتی پلکسر MUX1 و MUXII را نشان می‌دهد.



شکل (۶-۱۴) ساختار اصولی Prescaler تایمر 2 در میکروکنترلر AVR: ATmega8، ATmega16، ATmega32...

1- Timing Diagram

2- Real Time Counter (RTC)

نکته: زمانی‌که CS22, CS21, CS20 در ثبات کنترل تایمر 2 ، (TCCR2) مساوی صفر هستند، تایمر 2 متوقف است و نمی‌شمارد.

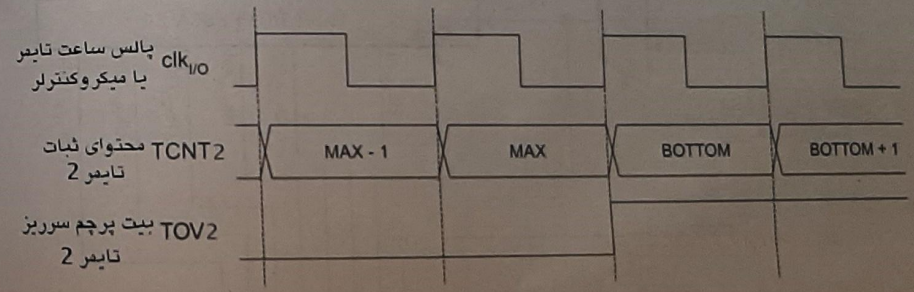
با 1 کردن بیت AS2 در ثبات ASSR* مولتی‌پلکسر 1 (MUX1)، پالس ساعت تایمر 2 را، از کریستال 32768 هرتز که به پایه TOSC2, TOSC1 متصل شده است، تأمین می‌کند، در این حالت پالس ساعت تایمر 2 برابر فرکانس 32768 هرتز تقسیم بر عدد 8, 32, 64, 128, 256 یا 1024 می‌گردد که می‌توان از آن‌ها پالس مبنای زمان 1 ثانیه دقیق برای ساعت را تأمین نمود.**

شماره بیت	7	6	5	4	3	2	1	0
ثبات ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB
	R	R	R	R	R/W	R	R	R
مقدار اولیه	0	0	0	0	0	0	0	0

شکل(۶-۱۵) ثبات ASSR

نکته: کریستال 32768 هرتز فقط برای تایمر 2 است و قسمت‌های دیگر میکروکنترلر از آن استفاده نمی‌کنند. لذا میکروکنترلر باید با اسلاید تور داخلی مثلاً 1MHz و یا... کار کند تا بتواند برای قسمت‌های مختلف میکروکنترلر پالس ساعت مورد لزوم را تأمین کند.

شکل (۶-۱۶) دیاگرام زمانبندی تایمر 2 موقعی‌که پالس ساعت تایمر برابر پالس ساعت میکروکنترلر باشد را نشان می‌دهد.

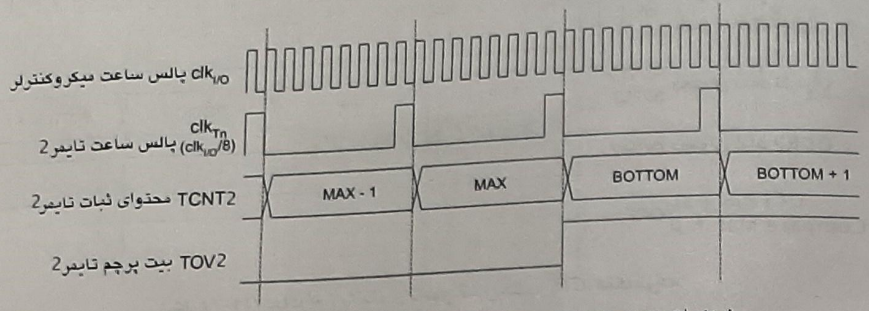


شکل (۶-۱۶) دیاگرام زمانبندی تایمر2 هنگامی‌که پالس ساعت تایمر برابر پالس ساعت میکروکنترلر AVR باشد.

همانطورکه از شکل مذکور ملاحظه می‌شود، در لبه بالارونده پالس ساعت، تایمر 2 در ثبات TCNT2 یک شماره می‌اندازد و زمانی‌که تایمر به ماکزیمم (MAX) رسید، با پالس بعدی مقدار تایمر برابر BOTTOM یعنی مساوی 0 می‌شود و بیت پرچم سرریز TOV2 برابر 1 می‌گردد.

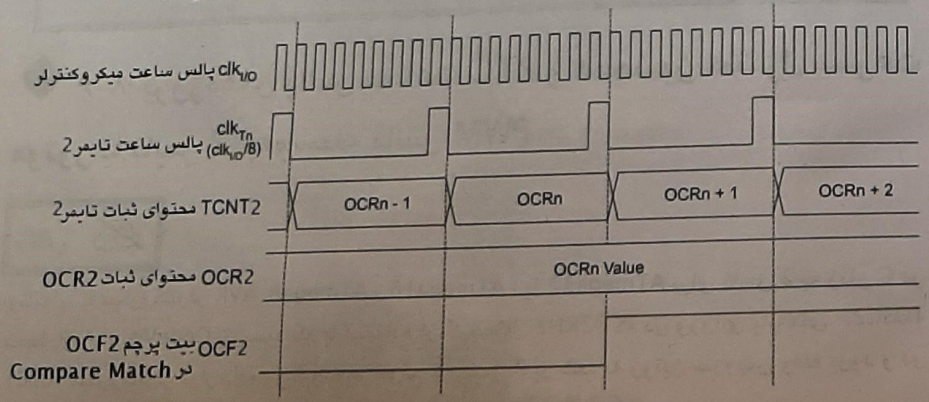
* بقیه بیت‌های ASSR برای تغییر مقادیر ثبات‌های TCNT2, TCCR2, OCR2 در حالت آسنکرون باشد.
** در ادامه مثال‌هایی دراین مورد خواهیم داشت.

شکل (۶-۱۷) همان دیاگرام شکل (۶-۱۶) است فقط پالس ساعت تایمر 2 ، توسط Prescaler برابر 1/8 پالس ساعت میکروکنترلر AVR شده است.



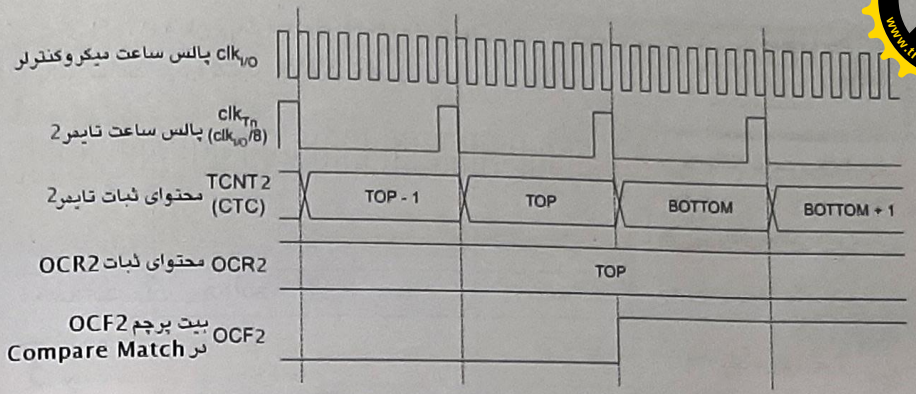
شکل (۶-۱۷) دیاگرام زمانبندی تایمر 2، هنگامی‌که پالس ساعت تایمر برابر 1/8 پالس ساعت میکروکنترلر AVR است.

شکل (۶-۱۸) دیاگرام زمانبندی تایمر 2 در حالت PWM زمانی‌که پالس ساعت تایمر 1/8 پالس ساعت میکروکنترلر AVR باشد را نشان می‌دهد. زمانی‌که مقدار تایمر TCNT2 برابر مقدار ثبات OCR2 شد، یا Compare Match رخ داد، بیت پرچم OCF1 در ثبات پرچم TIFR برابر 1 می‌شود.



شکل (۶-۱۸) دیاگرام زمانبندی تایمر 2، هنگامی‌که پالس ساعت تایمر برابر 1/8 پالس میکروکنترلر AVR است و Compare Match رُخ می‌دهد.

شکل (۶-۱۹) همان دیاگرام زمانبندی تایمر 2 در حالت CTC می‌باشد. موقعی که مقدار تایمر 2 یعنی TCNT2 برابر TOP باشد، بیت پرچم OCF2 برابر 1 می‌شود.



شکل (۶-۱۹) دیاگرام زمانبندی تایمر 2 در حالت CTC، هنگامی‌که پالس ساعت تایمر 2 برابر 1/8 پالس میکروکنترلر است.

نکته: *قبل از شروع کار تایمر2، باید به ثبات‌های آن مقدار اولیه داد. مقدار اولیه دادن به ثبات‌ها معمولاً شامل مقدار اولیه دادن به ثبات تایمر 2، TCNT2، مقدار ثبات OCR2، انتخاب پالس ساعت برای تایمر 2 و انتخاب نوع پالس PWM در ثبات کنترل TCCR2، فعال کردن یا غیرفعال کردن وقفه .... در ثبات TIMSK... می‌باشد.*

◆ ۶-۸: پروژه‌های پالس ساعت RTC و همچنین کنترل سرعت موتور با تایمر 2، به‌وسیله پالس PWM

مثال

برنامه‌ای با میکروکنترلر AVR: ATmega8 ، ATmega16 ، یا ATmega32 برای تایمر 2 به زبان C در محیط CodeVisionAVR بنویسید که با استفاده از کریستال 32KHz که در ورودی پایه‌های TOSC2, TOSC1 میکروکنترلر قرار داده می‌شود، هر موقع که تایمر 2 پر شد، به روتین سرویس وقفه برود و در بیت 0 پورت D، پالس 1 ثانیه برای مبنای زمان RTC تولید کند.

حل

برنامه آن مطابق شکل (۶-۲۰) می‌باشد. در برنامه مذکور متناسب با گزینه‌هایی که در ابزار CodeVisionAVR (شکل (۶-۲۰-ج)) در نرم‌افزار CodeVisionAVR انتخاب کردیم، نرم‌افزار مذکور به ثبات‌های تایمر 2 و... مقدار اولیه داده است. به این ترتیب در:

● بخش A:

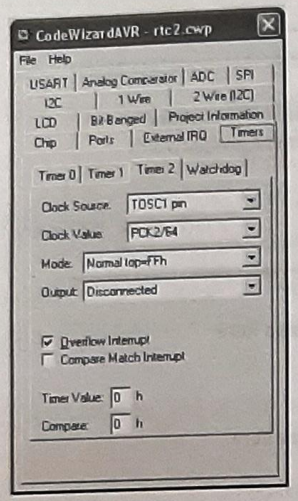
نوع میکروکنترلر و فرکانس آن مشخص شده است.

● بخش B:

توسط عبارت (۱)، Header file میکروکنترلر AVR: ATmega16 است که اگر میکروکنترلر AVR: ATmega8 یا ATmega32 باشد به ترتیب <mega8.h> یا <mega32.h> می‌باشد.

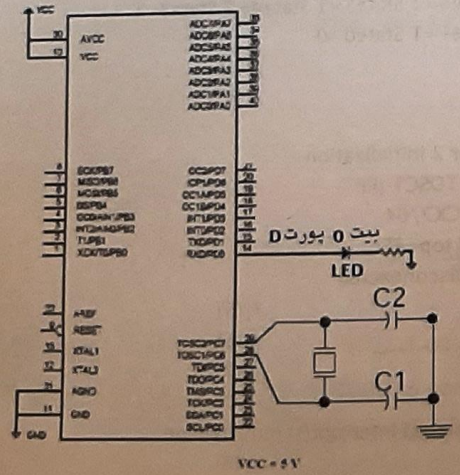
● بخش D*:

بیت 0 پورت D را به‌صورت خروجی تعریف می‌کند و به ثبات‌های تایمر 2 مقدار اولیه می‌دهد. در این بخش:



ج: تنظیمات ابزار میکروکنترلر CodeWizardAVR برابر برنامه rtc2

میکروکنترلر AVR: ATmega8 ، ATmega16 ، ATmega32**



* بخش C بعداً بحث می‌شود.

** شماره پایه‌های آن مطابق شکل (۳-۱۰) فصل (۳) می‌باشد.

نام فایل: rtc2.c

```
/*****************************************************
A Section
Chip type           : ATmega16
Program type        : Application
Clock frequency     : 1.000000 MHz
*****************************************************/
//B Section
#include <mega16.h>                    //(1)
//*****************************************************
//C Section
// Timer 2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
// Place your code here
PORTD.0 ^=1;                           //(2) blink  PD0 every 1 second
}
//*****************************************************
//D Section
// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In
// Func2=In Func1=In Func0=Out
// State7=T State6=T State5=T State4=T State3=T
// State2=T State1=T State0=0
PORTD=0x00;                            //(3)
DDRD=0x01;                             //(4)

// Timer/Counter 2 initialization
// Clock source: TOSC1 pin
// Clock value: PCK2/64
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x08;                             //(5)
TCCR2=0x04;                            //(6)
TCNT2=0x00;                            //(7)
OCR2=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x40;                            //(8)

// Global enable interrupts
#asm("sei")                            //(9)
```



```
while (1)                              //(10)
    {
    // Place your code here

    };
}
```

شکل (۶-۲۰) برنامه به زبان C در محیط CodeVisionAVR برای میکروکنترلر AVR، ATmega8، ATmega16 * *ATmega32*. *برای تایمر2 با استفاده از کریستال خارجی 32 KHz که هر 1 ثانیه، بیت 0 پورت D را برای مبنای زمان ساعت RTC معکوس کند.*

◆ دستورات (۳) و (۴): بیت 0 پورت D را خروجی می‌کنند.

◆ دستور (۵): بیت AS2 در ثبات ASSR را برابر 1 می‌کند تا تایمر 2، پالس خود را از طریق Prescaler از کریستال 32KHz که به پایه‌های TOSC2, TOSC1 متصل شده است بگیرد.

◆ دستور (۶): پالس ساعت کریستال را توسط Prescaler تقسیم بر 64 می‌کند و به تایمر 2 می‌دهد، یعنی فرکانس پالس تایمر 2 برابر 512 Hz =32768/64 می‌شود.

◆ دستور (۷): مقدار تایمر2 را برابر 0 قرار می‌دهد.

◆ دستور (۸): بیت فعال‌ساز وقفه تایمر 2، یعنی TOIE2 در ثبات TIMSK را برابر 1 می‌کند.

◆ دستور (۹): بیت فعال‌ساز کلی I در ثبات SREG را برابر 1 می‌کند.

به این ترتیب با دستورات (۸) و (۹) وقفه سرریز تایمر 2 فعال می‌شود.

◆ عبارت (۱۰): منتظر می‌ماند تا تایمر2، پُرشود، به محض اینکه تایمر 2، پُر شد، میکروکنترلر به روتین سرویس وقفه در بخش C می‌رود.

همانطور که می‌دانیم تایمر 2 به‌ازاء هر 256 پالس ورودی پر می‌شود، لذا روتین سرویس وقفه با فرکانس:

$$\frac{512}{256}=2\ Hz$$

یا با پریود:

$$T=\frac{1}{f}=\frac{1}{2}=0.5$$

فعال می‌شود و بیت 0 پورتD را معکوس می‌کند.

● بخش C:

روتین سرویس وقفه است که در آن عبارت (۲) باعث می‌شود، بیت 0 پورت D با عدد 1 XOR** شود، در نتیجه بیت 0 پورت D معکوس می‌شود و سپس به عبارت (۱۰) برمی‌گردد و منتظر می‌ماند تا مجدداً تایمر 2 پُر شود و عملیات فوق تکرار گردد.

* اختلاف آنها در توضیح برنامه بحث شده است.

** در گیت XOR اگر یک بیت ورودی برابر 1 باشد، خروجی XOR معکوس ورود دیگر می‌باشد.

این ترتیب 0.5 ثانیه بیت 0 پورت D برابر 0 و 0.5 ثانیه مساوی 1 می‌شود، لذا پریود آن 1 ثانیه می‌گردد که از آن می‌توان برای پالس دقیق مبنای زمان برای ساعت RTC استفاده نمود.

فایل با پسوند HEX برنامه مذکور توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.

مثال

برنامه‌ای به زبان C در محیط CodeVisionAVR برای میکروکنترلر AVR : ATmega16, ATmega8 یا ATmega32 ... بنویسید که پالس PWM به‌صورت Phase Correct در خروجی OC2 تولید کند. به‌طوریکه با فشار دادن کلید Up در پایه2 پورت A عرض پالس زیاد شود تا سرعت موتور بالا رود و با فشار دادن کلید Down در پایه 3 پورت A عرض پالس تقلیل یابد تا سرعت کم شود.

حل

برنامه آن مطابق شکل (۶-۲۱- ب) می‌باشد. در برنامه مذکور:

- **بخش A:**

  مشخصات میکروکنترلر AVR نوشته شده است.

- **بخش B:**

  - عبارت (۱): فایل Header file میکروکنترلر AVR : ATmega16 است که اگر میکروکنترلرهای AVR : ATmega8 یا ATmega32 باشند فایل مذکور به ترتیب «mega8.h» یا «mega32.h» است.
  - عبارت (۲): چون در برنامه به تأخیر نیاز داریم، لذا با عبارت (۲) فایل تابع تأخیر را در برنامه قرار داده‌ایم تا در عبارت (۱۴) از تأخیر ۳۰ میلی‌ثانیه استفاده کنیم.

- چون در میکروکنترلر AVR : ATmega16 بیت‌های 2 و 3 پورت A به ترتیب برای افزایش (Up) عرض پالس PWM و کاهش (Down) عرض پالس PWM استفاده شده‌اند. لذا پورت A ورودی شده است و به تایمر 2 مقدار اولیه داده شده است تا به‌صورت Phace Correct PWM کار کند.
  - عبارات (۳) و (۴): پورت A را ورودی کرده‌اند تا بتوان از کلیدهای Up و Down به‌عنوان ورودی استفاده کرد.
  - عبارات (۵) و (۶): بیت (۷) پورت D یا OC2 را خروجی می‌کنند.
  - عبارت (۷): ثبات کنترل تایمر 2، TCCR2 را برابر 61 هگزادسیمال یعنی مساوی 01100001 قرار می‌دهد.

در نتیجه:

- مقدار WGM21, WGM20 را برابر 01 قرار می‌دهد تا مطابق جدول (۶-۴) پالس Phase Correct PWM در خروجی OC2 تولید شود.

- مقدار COM20 و COM21 را به ترتیب برابر 10 قرار می‌دهد تا مطابق جدول (۶-۳) پالس ... به‌صورت Non-inverted تولید شود.
- بیت‌های CS22, CS21, CS20 را برابر 001 قرار داده است تا مطابق جدول (۶-۲) پالس ساعت تایمر، همان پالس ساعت میکروکنترلر و برابر 1000 KHz باشد.
  - دستور (۸): ثبات Compare Match یعنی ثبات OCR2 را برابر 70 هگزادسیمال قرار می‌دهد.

به این ترتیب در این بخش مقدار اولیه به تایمر2، ثبات‌ها و پورت‌ها داده شده است.

- **بخش C:**
  - عبارت (۹): باعث می‌شود تا حلقه While یعنی دستورات بین (۱۰) تا (۱۴) مرتباً تکرار گردند.
  - عبارت (۱۰): اگر بیت 2 پورت A یعنی PINA.2 برابر 1 شود (کلیه Up فشار داده شود) و همچنین مقدار ثبات OCR2 کمتر از 240 باشد*، عبارت (۱۱) اجرا می‌شود، درغیراین‌صورت عبارت (۱۲) اجرا می‌گردد.
  - عبارت (۱۱): به ثبات Compare Match، یعنی به OCR2 عدد 10 را اضافه می‌کند تا عرض پالس PWM افزایش یابد.
  - عبارت (۱۲): اگر بیت 3 پورت A یعنی PINA.3 برابر 1 شود (یا کلیه Down فشار داده شود) و همچنین مقدار ثبات OCR2 بیشتر از 10 باشد** عبارت (۱۳) اجرا می‌شود درغیراین‌صورت عبارت (۱۴) اجرا می‌گردد.
  - عبارت (۱۳): از ثبات Compare Match یعنی از OCR2 مقدار 10 واحد کم می‌کند تا عرض پالس PWM تقلیل یابد.
  - عبارت (۱۴): باعث می‌شود تأخیری حدود ۳۰ میلی‌ثانیه در حلقه While ایجاد شود که اگر فشار دادن کلید Up یا کلید Down طولانی شدند عرض پالس خیلی سریع تغییر نکند و قابل کنترل باشد.

با توجه به مطالب فوق دستورات (۱۰) تا (۱۴) مرتباً تکرار می‌شوند و باعث می‌شوند با فشار دادن کلید Up عرض پالس PWM افزایش یابد و سرعت موتور زیاد شود و با فشار دادن کلید Down عرض پالس PWM کم شود تا سرعت موتور تقلیل یابد.

در مدار مذکور از بافر ULN2003 استفاده شده*** تا جریان لازم برای موتور را تأمین کند. فایل HEX برنامه مذکور توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.

---

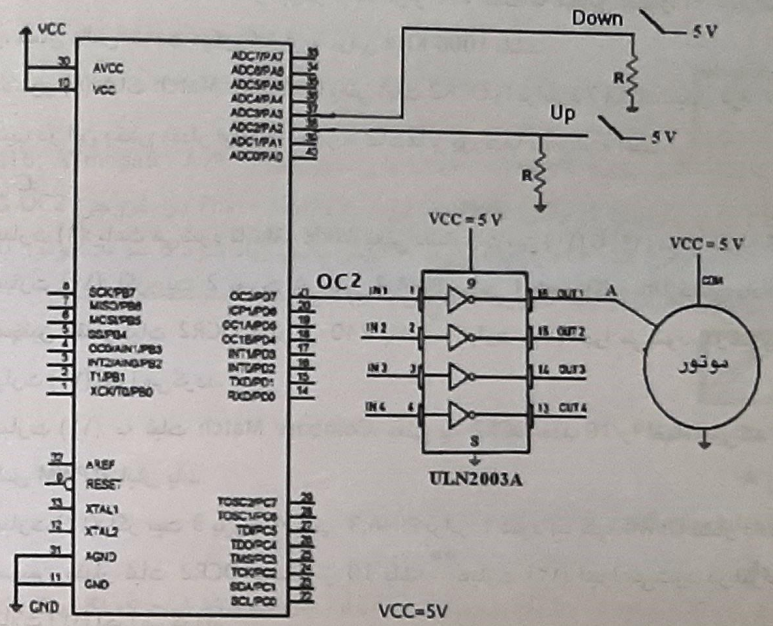
* این شرط باعث می‌شود مقدار ثبات OCR2 از 255 بیشتر نشود تا دوباره به 0 برنگردد.

** این شرط باعث می‌شود تا حداقل مقدار ثبات OCR2 از 10 کمتر نشود تا مجدداً به ماکزیمم برنگردد.

*** این بافر تا ۵۰۰ میلی‌آمپر می‌تواند جریان بدهد. برای اطلاعات بیشتر به Sata Sheet آن در CD مراجعه فرمایید.

الف : شکل میکروکنترلر

ATmega32, Atmega16, ATmega8:AVR ❋



نام فایل variable_phase_correct_atmega16.c

```
/*****************************************************
A Section
Chip type           : ATmega16L
Program type        : Application
Clock frequency     : 1.000000 MHz
*****************************************************/
//B Section
#include <mega16.h>                    //(1)
#include <delay.h>                     //(2)
// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
```

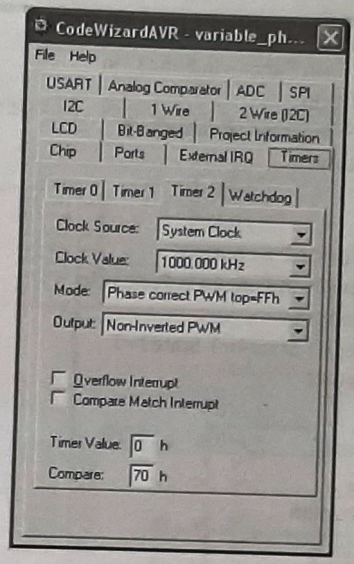
❋ شماره پایه‌های آن مطابق شکل (۳-۱۰) فصل (۳) می‌باشد.


```
// Port A initialization,as input
// Func7=In Func6=In Func5=In Func4=In Func3=In
// Func2=In Func1=In Func0=In
// State7=P State6=P State5=P State4=P State3=P
// State2=P State1=P State0=P
PORTA=0xFF;                            //(3)
DDRA=0x00;                             //(4)
// Port D initialization,PD7 or OC2 as output
// Func7=Out Func6=In Func5=In Func4=In Func3=In
// Func2=In Func1=In Func0=In
// State7=0 State6=T State5=T State4=T State3=T
// State2=T State1=T State0=T
PORTD=0x00;                            //(5)
DDRD=0x80;                             //(6)

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 1000.000 kHz
// Mode: Phase correct PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x61;                            //(7)
TCNT2=0x00;
OCR2=0x70;                             //(8)
//C Section
while (1)                              //(9)
    {
    // Place your code here
    if (PINA.2==1 && OCR2<240)         //(10)
    OCR2=OCR2+10;                      //(11)
    if (PINA.3==1 && OCR2>10)          //(12)
    OCR2=OCR2-10;                      //(13)
    delay_ms(30);                      //(14)
    };
}
```

ب: برنامه

شکل (۶-۲۱) برنامه به زبان C در محیط CodeVisionAVR در میکروکنترلر AVR: ATmega8 ، ATmega16 یا ATmega32 برای تولید پالس PWM به روش Phase Correct به‌طوریکه اگر پایه 2 پورت A بنام Up برابر 1 شود، عرض پالس زیاد گردد و اگر بیت 3 پورت A بنام Down مساوی 1 شود عرض پالس کم گردد.



ج: تنظیمات ابزار CodeWizardAVR برای برنامه variable_ phase_ correct_ atmega16.c

◆ ۶-۹: خلاصه

در این فصل طرز کار تایمر 2، حالت‌های مختلف آن و تولید پالس PWM بحث شده است. علاوه‌براین ثبات‌های OCR2, TCNT2, TIFR, TIMSK, TCCR2 و کاربردهای آن‌ها در تولید پالس PWM به‌صورت CTC, Phase Correct PWM, Fast PWM و همچنین تولید پالس مبنای زمان 1 ثانیه برای ساعت RTC، با ذکر مثال‌ها و پروژه‌هایی از تایمر 2، با دستورات اسمبلی میکروکنترلرها و زبان C در میکروکنترلرهای AVR: ATmega8 و ATmega16 و ATmega32 ... مورد بحث قرار گرفت.

۶-۱۰ : تمرین

۶-۱- تایمر 2 در میکروکنترلرهای AVR، چند بیتی است و پالس ساعت آن چگونه تأمین می‌شود؟

۶-۲- تایمر 2 در میکروکنترلرهای AVR را چگونه می‌توان استفاده نمود.

۶-۳- دستوراتی در میکروکنترلرهای AVR: ATmega32, ATmega16, ATmega8 ... بنویسید که:

الف: تایمر 2 در حالت معمولی کار کند و پالس ساعت آن برابر 1/1024 پالس ساعت میکروکنترلر باشد.

ب: چه موقع تایمر شروع به شمارش می‌کند.

۶-۴- برنامه‌ای بنویسید که در میکروکنترلرهای AVR: ATmega8، ATmega16، ATmega32 تایمر2 به‌صورت معمولی کار کند و در هر لحظه مقدار تایمر 2 را در پورت D قرار دهد.

۶-۵- برنامه تمرین (۶-۴) را در میکروکنترلر AVR : ATmega32, ATmega16 به زبان C در محیط CodeVisionAVR بنویسید.

۶-۶- دستوراتی بنویسید که تایمر2 در حالت:

الف: Fast PWM به‌صورت Non-inverted با فرکانس پالس ساعت تایمر، برابر پالس ساعت میکروکنترلر AVR باشد.

ب: مقدار Compare Match رابرابر 120 قرار دهد.

۶-۷- برنامه‌ای بنویسید که با تایمر 2، پالس PWM با روش Fast PWM در خروجی OC2 به‌صورت Non-Inverted تولید کند (در میکروکنترلر AVR : ATmega8 یا ATmega16 یا ATmega32 ...). پالس ساعت تایمر 1/8 پالس ساعت میکروکنترلر و مقدار ثبات OCR2 نیز برابر 50 باشد.

فرکانس پالس PWM چقدر است؟

۶-۸- برنامه مثال قبل را با زبان C در محیط CodeVisionAVR بنویسید.

۶-۹- دستوراتی در میکروکنترلرهای AVR : ATmega8 ، ATmega16 یا ATmega32 ... بنویسید که تایمر 2 :

الف: درحالت Non-inverted, Phase Correct PWM کار کند و پالس ساعت تایمر برابر 1/8 پالس ساعت میکروکنترلر باشد.

ب: مقدار Compare Match یا ثبات OCR2 مساوی 50 باشد.

۶-۱۰- برنامه‌ای به زبان C در محیط CodeVisionAVR برای میکروکنترلر AVR : ATmega8 ، ATmega16 ، ATmega32 بنویسید که پالس PWM به‌صورت Phase Correct در خروجی OC2 تولید کند، به‌طوریکه با فشار دادن کلید up در پایه 2 پورت C، عرض پالس زیاد شود و با فشار دادن کلید Down در پایه 3 پورت C عرض پالس تقلیل یابد.

۶-۱۱- چگونه می‌توان با تایمر 2، پالس مربعی متقارن در حالت CTC تولید کرد و فرکانس آن چقدر است؟

۶-۱۲- دستوراتی بنویسید که:

الف: میکروکنترلر AVR را در حالت CTC با خروجی OC2 به‌صورت Toggle قرار دهد و پالس ساعت تایمر 2، برابر 1/8 پالس ساعت میکروکنترلر باشد.

ب: مقدار ثبات OCR2 یعنی Compare Match را برابر 80 بگیرید.

ج: پایه OC2 را خروجی کنید.

*** *** ***