

HP InfoTech

<https://ceil.ir>

شروع به کار با CodeVisionAVR برای 6.1 Atmel Studio

ترجمه توسط گروه مهندسی رامتین الکترونیک

<https://ceil.ir>

انتشار و باز نشر فقط با درج لینک این وبسایت مجاز است.

فهرست

3.....	1. مقدمه
3.....	2. آماده سازی
4.....	3. ایجاد جدید پروژه
16.....	4. ویرایش کد منبع
17.....	5. پیگردینی پروژه
20.....	6. ساخت پروژه و برنامه نویسی تراشه
21.....	7. خطایابی برنامه
26.....	8. نتیجه
27.....	9. ضمیمه آ – منبع کد

1. معرفی

هدف از این آموزش راهنمایی کاربر در تهیه، ساخت و رفع اشکال یک برنامه نمونه سی "C" ورژن 3.07 یا کامپایلر جدیدتر با استفاده از CodeVisionAVR برای Atmel Studio ورژن 6.1 یا بالاتر است. در این آموزش برای مثال یک برنامه ساده برای میکروکنترلر Atmel ATmega328 می نویسیم که از برد UNO آردوینو استفاده شده است.

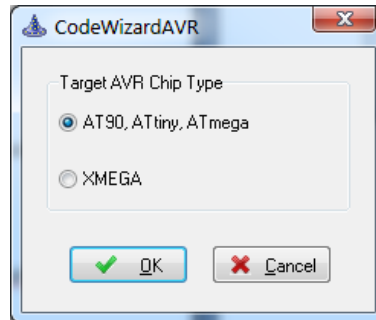
2. آماده سازی

Atmel Studio را از www.atmel.com دانلود و نصب کنید
کامپایلر سی : CodeVisionAVR را با اجرای فایل CodeVisionAVR.msi نصب کنید.
هنگامی که از شما خواسته شد، از دایرکتوری نصب پیش فرض پیشنهاد شده توسط نصب کننده استفاده کنید.
لطفاً توجه داشته باشید که برای نصب و استفاده از CodeVisionAVR در ویندوز به دسترسی Administrator نیاز است.
آماده سازی سخت افزاری زیر را انجام دهید:

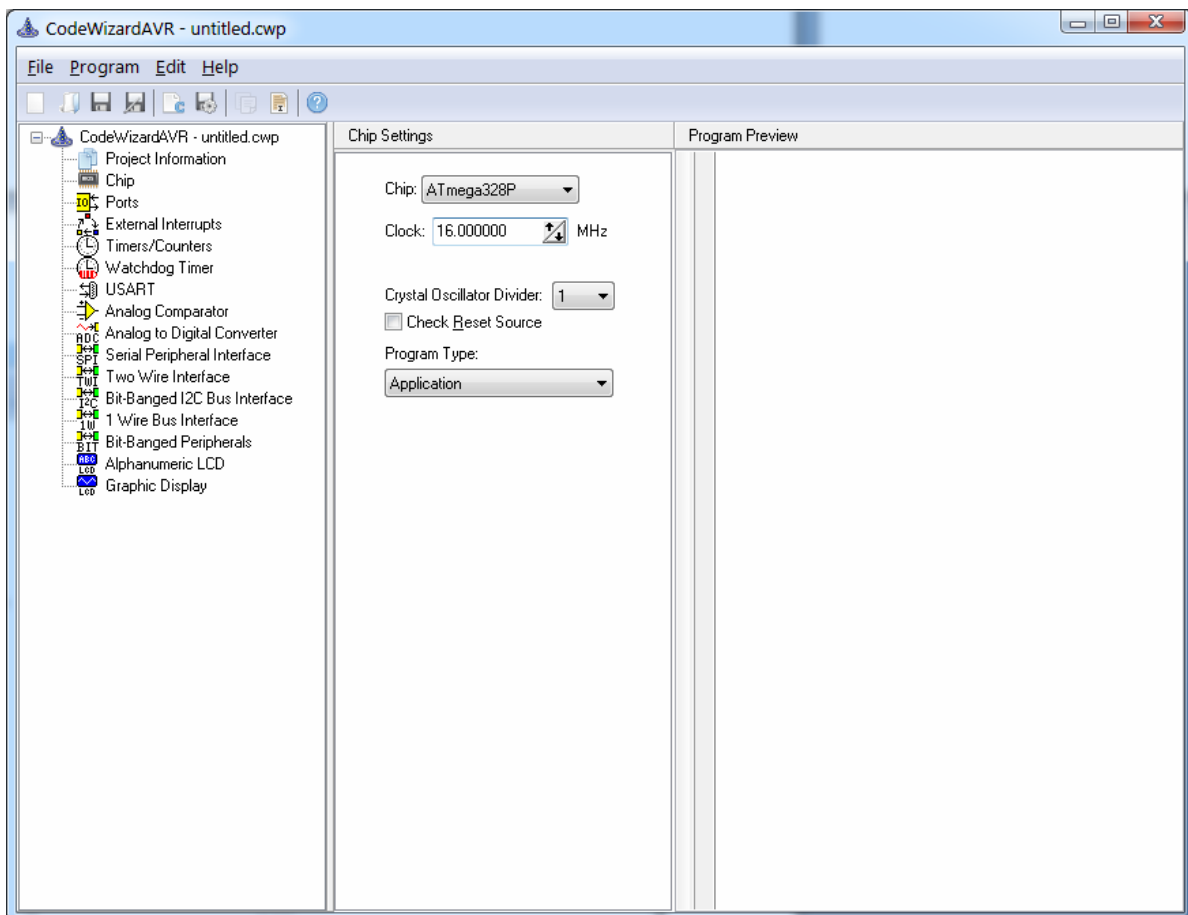
- اگر قبلاً نصب نشده است، یک هدر 6 پین را به محل مشخص شده ICSP در برد آردوینو UNO لحیم کنید.
- کاتدهای 8 ال ای دی را به خروجی هایی که روی برد DIGITAL 0..7 مشخص شده است وصل کنید. این خروجی ها با پین های PORTD PD0..PD7 میکروکنترلر مطابقت دارند.
- آند هر LED را با استفاده از یک مقاومت 1k به پین 5 ولت هدر کانکتور پاور برد وصل کنید.
- کانکتور USB برد آردوینو را به پورت USB کامپیوتر خود وصل کنید. این کار منبع تغذیه و ارتباط را برای برد فراهم می کند.

3. ایجاد پروژه جدید

اتمل استودیو IDE را اجرا کنید.
پروژه جدیدی را به وسیله منو File|New|Project ایجاد کنید.
در پنجره باز شده اجازه دهید تا کدویزارد اجرا شود تا پروژه جدید برای خانواده تراشه AVR ایجاد شود.

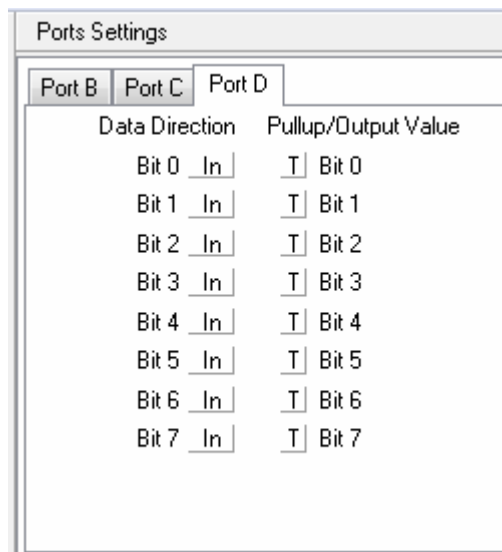
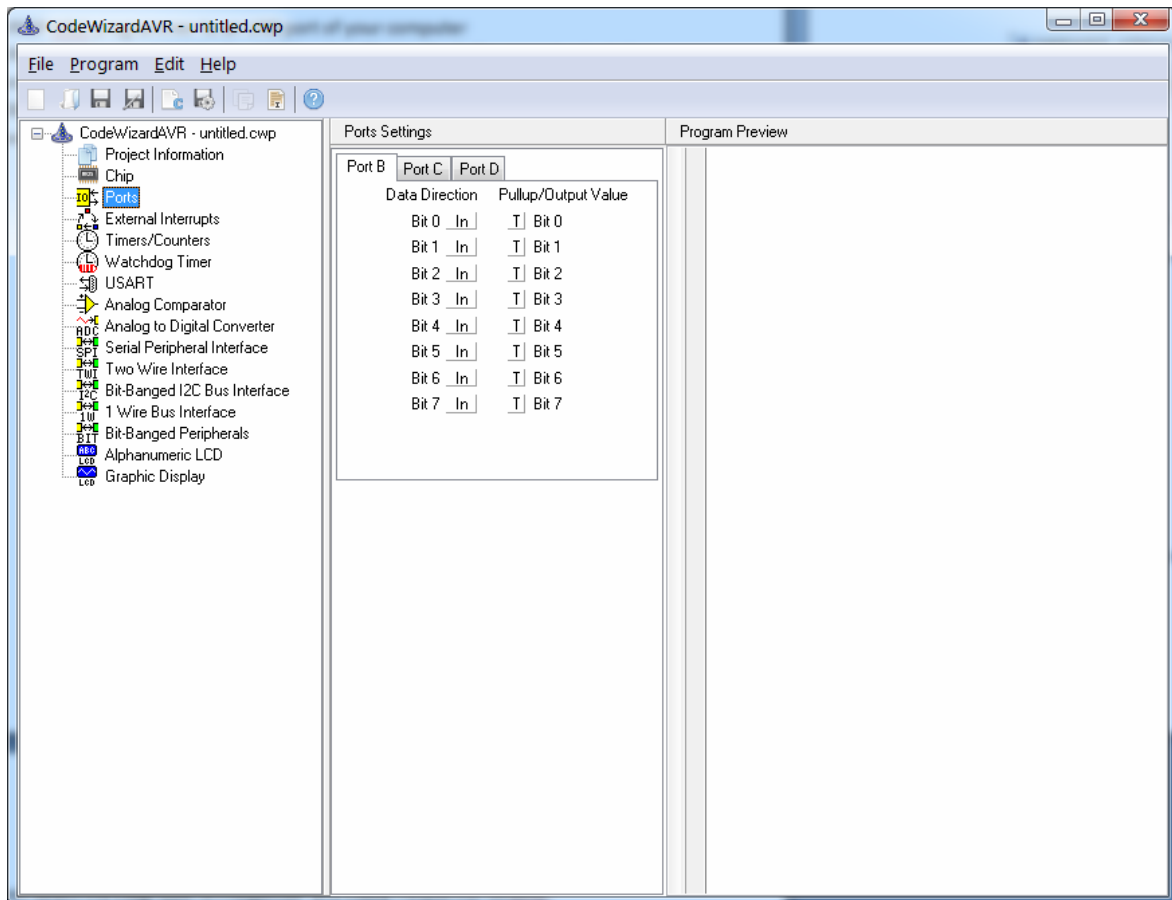


گزینه AT90, ATtiny, ATmega را انتخاب کنید و OK را بزنید.
CodeWizardAVR اجرا می شود و پنجره زیر نمایش داده می شود.



در پنل تنظیمات چیپ، نوع چیپ: ATmega328P و فرکانس ساعت: 16 مگاهرتز را انتخاب کنید.

مرحله بعدی پیکربندی پین های PORTD PD0 تا PD7 به عنوان خروجی است. برای رسیدن به این هدف، روی گره **Ports** درخت CodeWizard کلیک کنید. یک پانل پیکربندی جدید برای **Port Setting** نمایش داده می شود:



برای انتخاب پیکربندی PORTD روی تب **Port D** کلیک کنید:

همانطور که مشاهده می شود، **Port D Data Direction** برای تمام پین های I/O به طور پیش فرض به عنوان ورودی (In) تنظیم شده است.

Ports Settings		
Port B	Port C	Port D
Data Direction	Pullup/Output Value	
Bit 0 Out	0	Bit 0
Bit 1 Out	0	Bit 1
Bit 2 Out	0	Bit 2
Bit 3 Out	0	Bit 3
Bit 4 Out	0	Bit 4
Bit 5 Out	0	Bit 5
Bit 6 Out	0	Bit 6
Bit 7 Out	0	Bit 7

روی هر دکمه بیت 0 تا بیت 7 کلیک کنید تا بین های ورودی/خروجی را به عنوان خروجی تنظیم کنید:

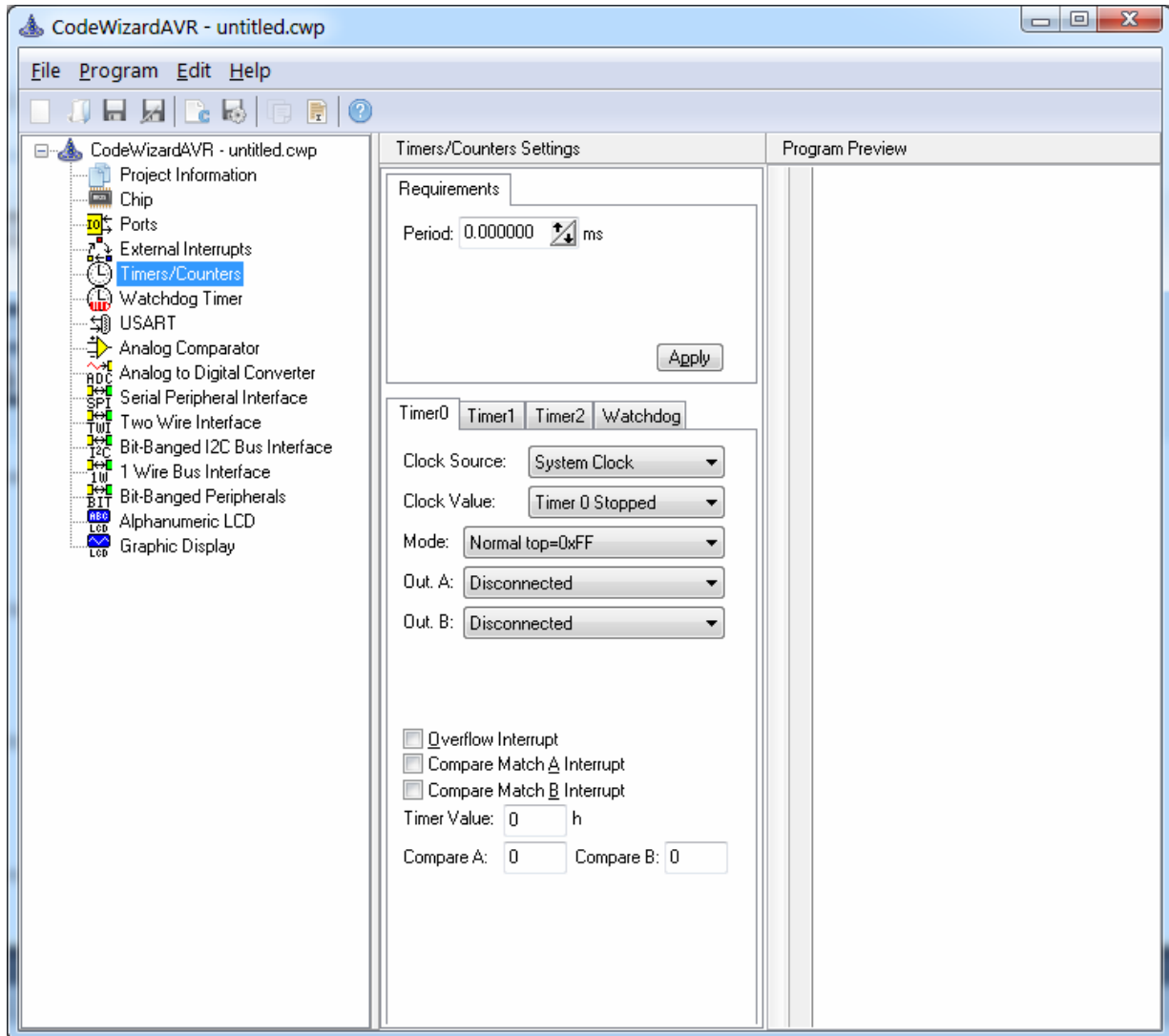
از آنجایی که LED ها پس از تنظیم مجدد چیپ باید خاموش باشند، پتانسیل کاتدهای آنها باید +5 ولت باشد، بنابراین مقادیر خروجی

Ports Settings		
Port B	Port C	Port D
Data Direction	Pullup/Output Value	
Bit 0 Out	1	Bit 0
Bit 1 Out	1	Bit 1
Bit 2 Out	1	Bit 2
Bit 3 Out	1	Bit 3
Bit 4 Out	1	Bit 4
Bit 5 Out	1	Bit 5
Bit 6 Out	1	Bit 6
Bit 7 Out	1	Bit 7

برای پورت D بیت 0 تا بیت 7 باید با کلیک بر روی دکمه های مربوطه روی 1 تنظیم شود:

مرحله بعدی پیکربندی یک تایمر / شمارنده برای ایجاد وقفه پس از هر 200 میلی ثانیه است. روی گره **Timers/Counters** درخت CodeWizard کلیک کنید.

یک پانل **Timers/Counters Settings** نمایش داده می شود:



Timers/Counters Settings

Requirements

Period: 0.000000 ms

Apply

Timer0 Timer1 Timer2 Watchdog

Clock Source: System Clock

Clock Value: Timer1 Stopped

Mode: Normal top=0xFFFF

Out. A: Disconnected

Out. B: Disconnected

Input Capture:

Noise Cancel

Rising Edge

Interrupt on:

Timer1 Overflow

Input Capture

Value: 0 h Inp. Capture: 0 h

Comp. A: 0 h B: 0 h

Timer1 استفاده خواهد شد، بنابراین روی تب مربوطه کلیک کنید:

از آنجایی که پس از هر 200 میلی ثانیه به یک وقفه مچ مقایسه تایمر 1 نیاز داریم، باید حالت عملیاتی **Mode** را به این صورت انتخاب کنیم :

Period: 200 ms مقدار **CTC top=OCR1A** را در پنل **Requirements** مشخص کرده و چک باکس **Interrupt on Compare A** را علامت بزینید:

در این حالت عملیاتی، تایمر 1 پالس های ساعت سیستم را از Pre Scaled می شمارد تا زمانی که رجیستر TCNT1 با مقدار رجیستر OCR1A برابر شود. هنگامی که این اتفاق می افتد، رجیستر TCNT1 به طور خودکار به 0 بازنشانی می شود و تایمر 1 در مقایسه با CR1A وقفه مطابقت ایجاد میشود.

با کلیک بر روی دکمه **Apply** در پنل **CodeWizardAVR .Requirements** مقادیر مورد نیاز را برای ثبات های پیکربندی Timer 1 ایجاد می کند:

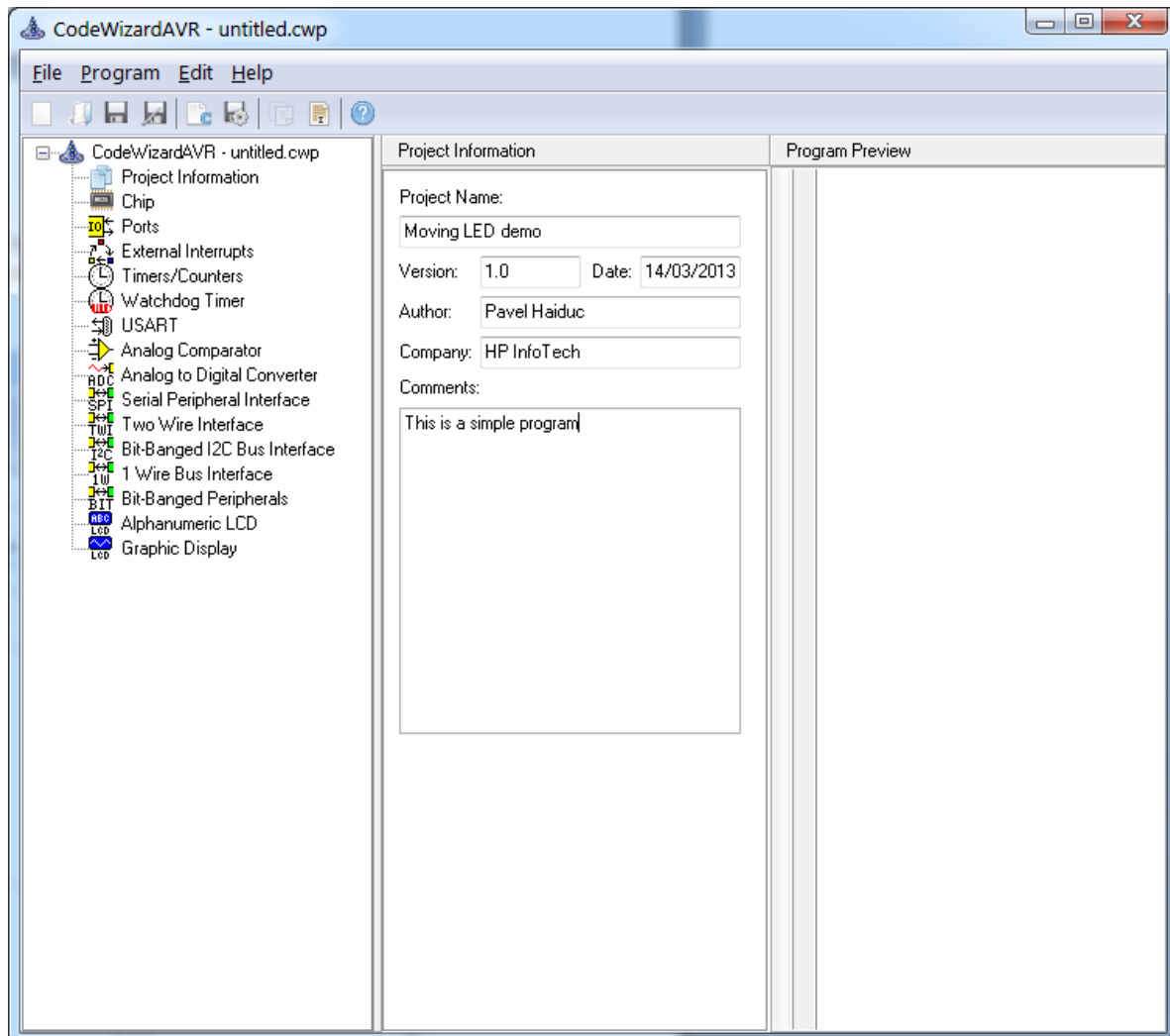
The screenshot shows the 'Timers/Counters Settings' dialog box. The 'Requirements' tab is selected, displaying a 'Period' of 200.000000 ms and an 'Obtained Period' of 0.2 s with 0.00% error. Below, the 'Timer1' tab is active, showing 'Clock Source' as System Clock, 'Clock Value' as 250.000 kHz, 'Mode' as CTC top=OCR1A, and both 'Out. A' and 'Out. B' as Disconnected. Under 'Input Capture', both 'Noise Cancel' and 'Rising Edge' are unchecked. Under 'Interrupt on:', 'Compare A Match' is checked. The 'Value' and 'Inp. Capture' fields are both set to 0. The 'Comp. A' field is set to C34F.


همانطور که در پنجره بالا مشاهده می شود، ساعت سیستم 16 مگاهرتز بر 64 تقسیم می شود تا مقدار **Clock Value** 1 برابر با 250 کیلوهرتز به دست آید و رجیستر OCR1A با مقدار 0xC34F مقداردهی اولیه می شود. بازه زمانی به دست آمده بین دو وقفه 0.2 ثانیه خواهد بود که با خطای 0% مطابقت دارد.

توجه: پیکربندی تایمر خودکار در نسخه CodeVisionAVR Evaluation غیرفعال است. بنابراین با کلیک روی دکمه **Apply**، یک پیغام خطا توسط نسخه Evaluation صادر می شود.

کاربر باید به صورت دستی **Clock Value: 250.000 kHz** را انتخاب کند و مقدار C34F را در **Comp** وارد کند.

مرحله بعدی، قبل از ایجاد کد برنامه واقعی، این است که با کلیک بر روی گره **Project Information** و تکمیل **comments** در پنل مربوطه، نظرانی را در مورد برنامه خود مشخص کنیم:



با استفاده از منوی **Program|Generate** یا کلیک بر روی دکمه  toolbar، برنامه C ایجاد می شود که در پنجره **Program Preview** برنامه قابل مشاهده است:

```

Program Preview
1  #include <mega328p.h>
2
3  // Declare your global variables here
4
5  // Timer1 output compare A interrupt service routine
6  interrupt [TIM1_COMPA] void timer1_compa_isr(void)
7  {
8  // Place your code here
9  |
10 }
11
12 void main(void)
13 {
14 // Declare your local variables here
15
16 // Crystal Oscillator division factor: 1
17 #pragma optimize-
18 CLKPR=(1<<CLKPCE);
19 CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
20 #ifdef _OPTIMIZE_SIZE_
21 #pragma optimize+
22 #endif
23
24 // Input/Output Ports initialization
25 // Port B initialization
26 // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
27 DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (
28 // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
29 PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) |
30
31 // Port C initialization
32 // Function: Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
33 DDRC=(0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) | (0<<DDC0);
34 // State: Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
35 PORTC=(0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) |

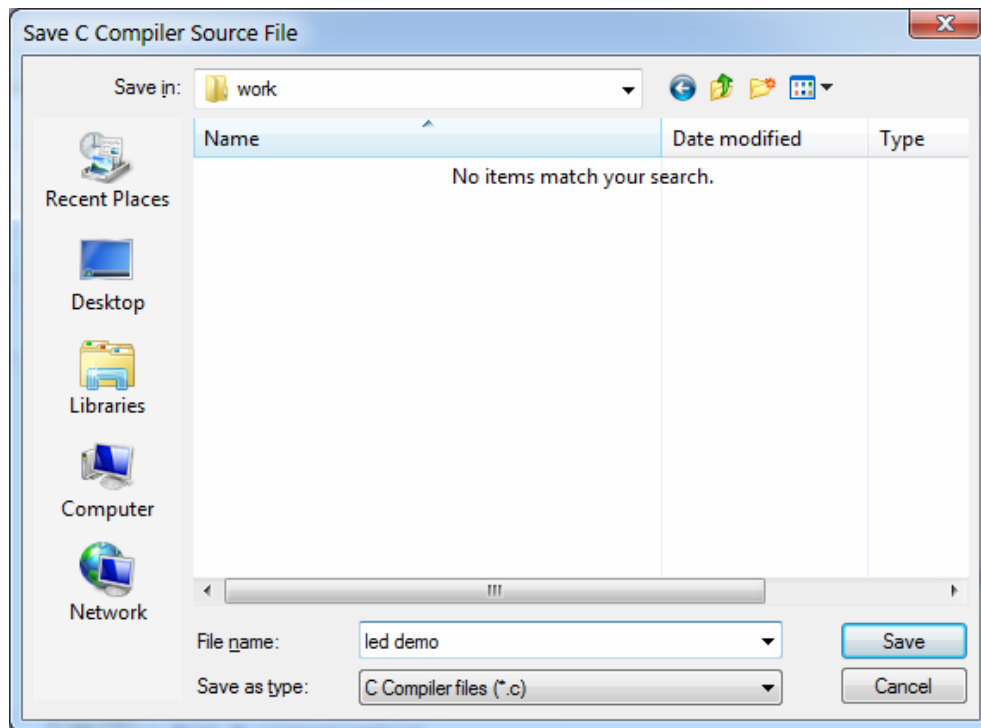
```

با کلیک بر روی یک گره محیطی در درخت CodeWizard، مکان نما در پنجره **Program Preview** برنامه در دنباله کد اولیه مربوط به آن ابزار جانبی قرار می گیرد.

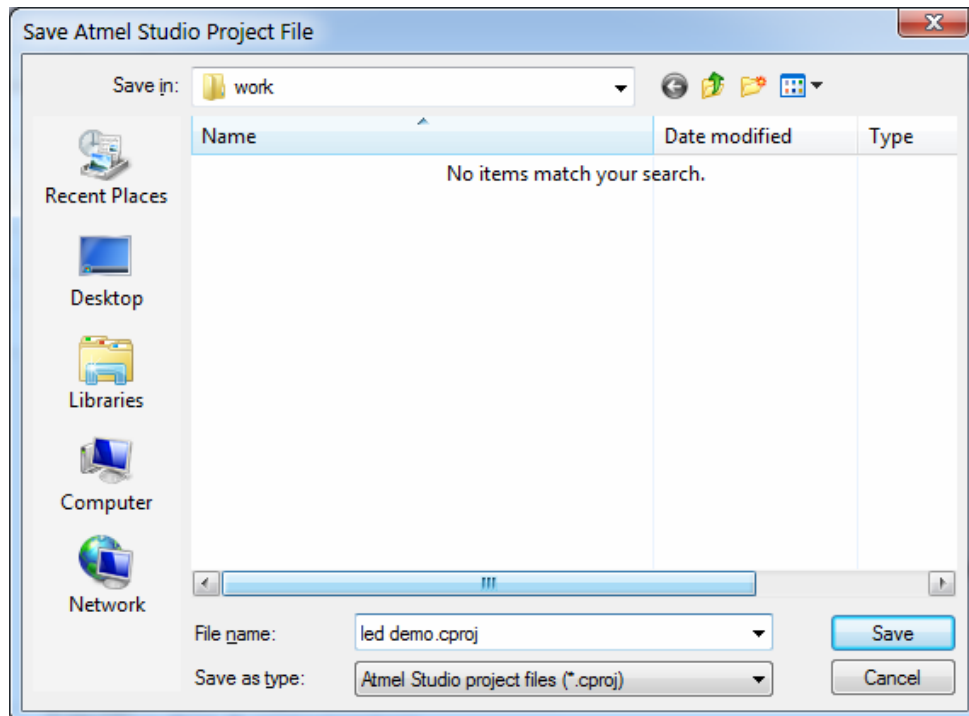
توجه: به طور پیش فرض CodeWizardAVR حتی برای وسایل جانبی که استفاده نمی شوند (غیرفعال) کد اولیه تولید می کند. اگر بازنشانی نرم افزار با پرسش به آدرس 0 انجام شود، این یک اقدام ایمنی برای پیکربندی صحیح تراشه است. به منظور کاهش اندازه برنامه تولید شده، می توان با برداشتن تیک گزینه منوی **Program|Generate Code for Disabled Peripherals** غیرفعال کرد.

وقتی از کد اولیه تولید شد، باید برنامه جدید را با استفاده از منو **Program|Generate, Save and Exit** یا در تولبار  ذخیره کنیم.

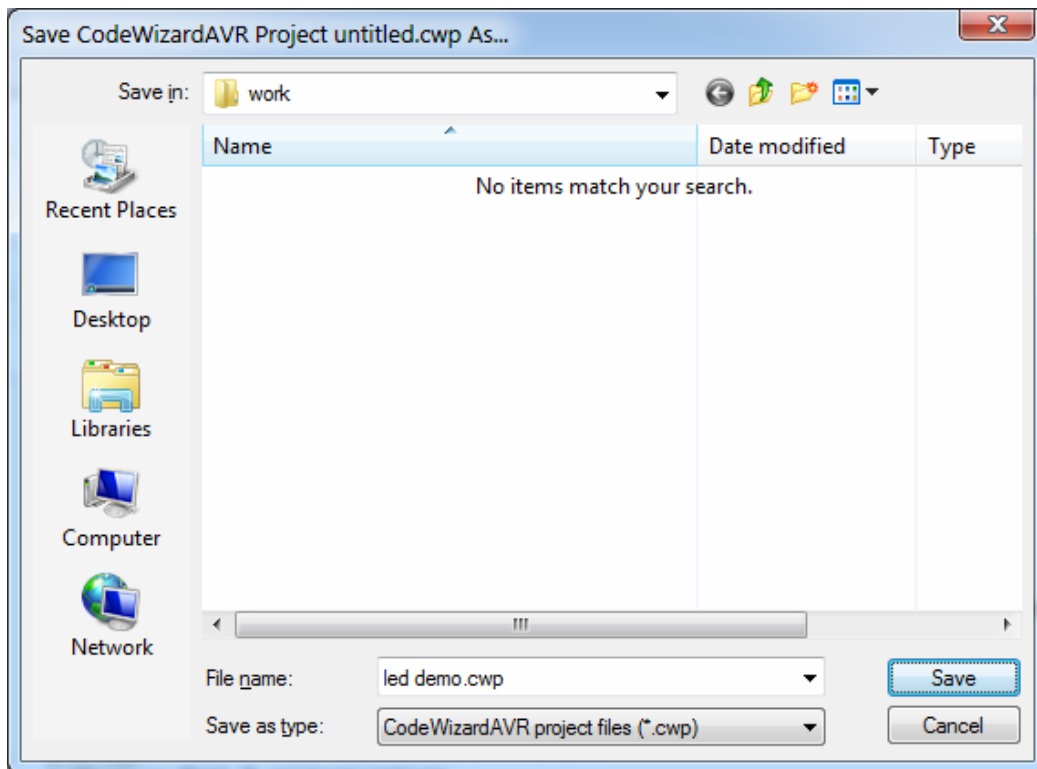
ابتدا نام اولین فایل منبع C. پروژه از ما خواسته می شود:



در ادامه باید نام پروژه Atmel Studio را مشخص کنیم:

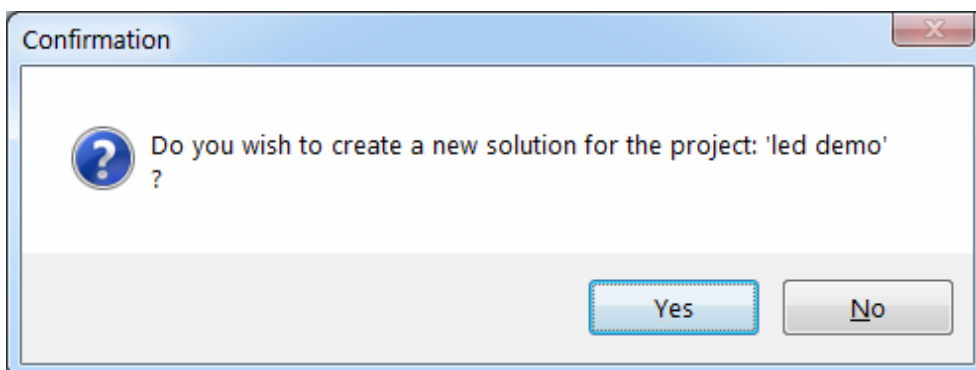


و در نهایت باید پیکربندی محیطی پروژه خود را در فایل **cwp** پروژه CodeWizardAVR ذخیره کنیم:

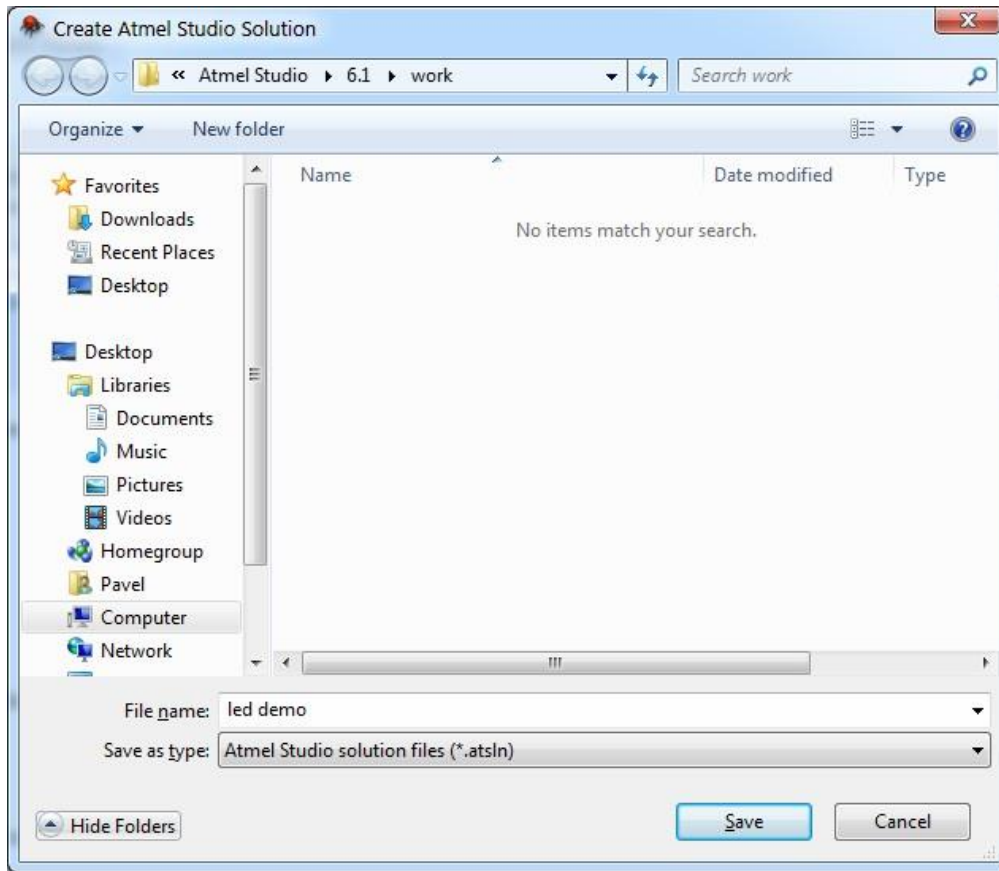


این به ما این امکان را می دهد که فقط با بارگذاری مجدد فایل **cwp** در CodeWizardAVR از همان پیکربندی محیطی برای پروژه های دیگر استفاده کنیم.

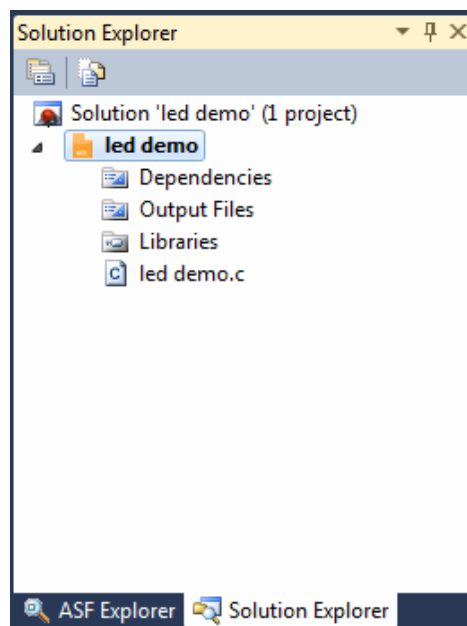
هنگامی که همه این فایل ها ذخیره شدند، Atmel Studio از ما می خواهد solution جدیدی برای پروژه نمایشی "led demo" ایجاد کنیم:



ما روی دکمه Yes کلیک می کنیم و نام سلوشن از شما خواسته می شود:



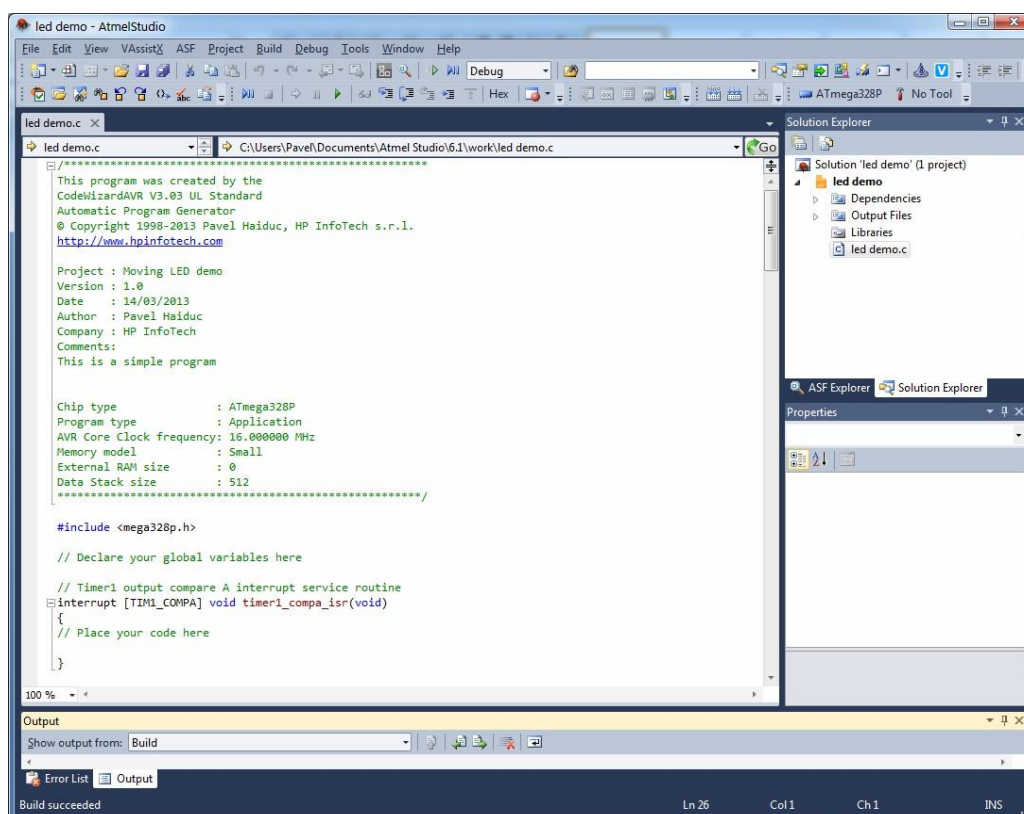
پس از انجام این کار، سلوشن و پروژه جدید در Atmel Studio بارگذاری می شود و اطلاعات مربوطه در Solution Explorer نمایش داده می شود:



1. ویرایش کد منبع

CodeWizardAVR تمام کدهای مورد نیاز برای مقداردهی اولیه ابزارهای جانبی را برای برنامه ما ایجاد کرده است، اکنون باید بخشی از کد مورد نیاز برای اجرای وظیفه خود را اضافه کنیم: به طور متوالی با تأخیر 200 میلی ثانیه، هر یک از 8 LED متصل به PORTD روشن می شود.

برای باز کردن فایل منبع C پروژه در ویرایشگر، باید روی گره led demo.c در Solution Explorer دوبار کلیک کنیم. هنگامی که فایل در پنجره ویرایشگر بارگذاری شد، می توانیم تغییراتی را در آن اعمال کنیم:



تمام عملکرد برنامه توسط تایمر 1 در مقایسه با روال سرویس وقفه `timer1_compa_isr` (Timer 1 compare with OCR1A) انجام می شود که هر 200 میلی ثانیه فراخوانی می شود. کد مورد نیاز با متن پررنگ در آنجا اضافه می شود:

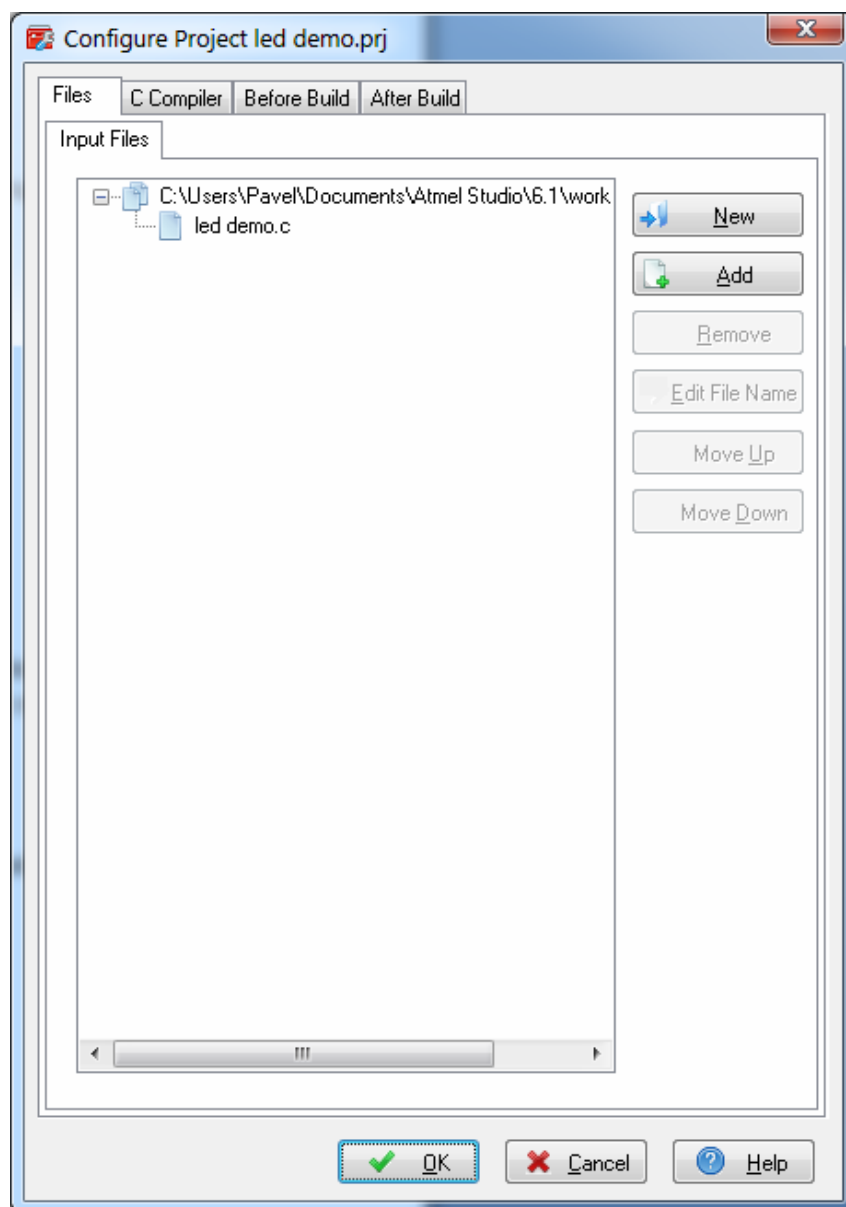
```
// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here

// If all LEDs are off, light the first one
if (PORTD == 0xFF) PORTD = 0xFE;
// One of the LEDs is already lighted, turn it off and light the next one
else PORTD = (PORTD << 1) | 1;
}
```

توجه: از آنجایی که آندهای LED توسط یک مقاومت به +5 ولت و کاتدها به خروجی های PORTD متصل می شوند، برای روشن شدن آنها باید خروجی مربوطه را روی سطح منطقی 0 تنظیم کرد.

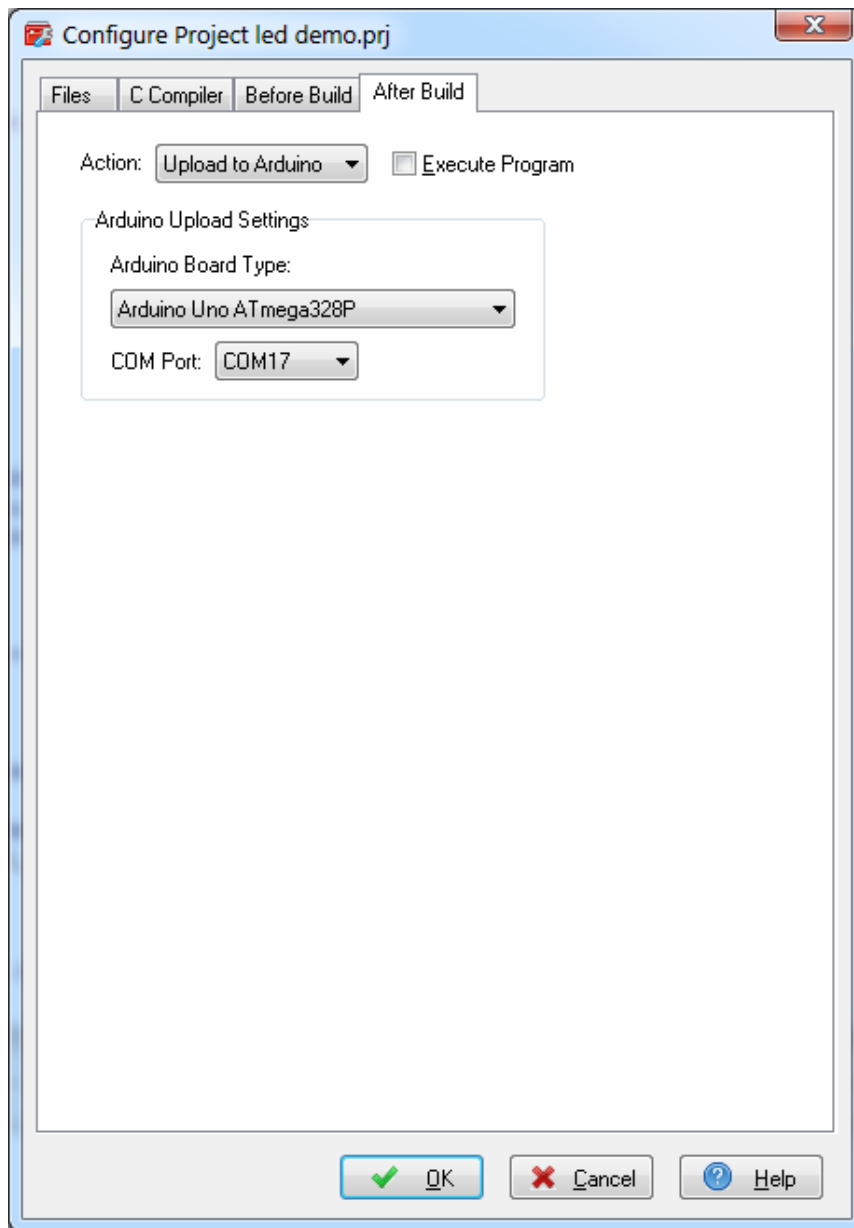
1. پیکربندی پروژه

2. هنگامی که تغییرات مورد نیاز در کد منبع برنامه ایجاد شد، مرحله بعدی پیکربندی گزینه های ساخت پروژه با استفاده از منوی **Project|Configure** است.
3. پنجره معاوره ای نمایش داده می شود:



ما باید تراشه را پس از بیلد موفقیت آمیز به طور خودکار برنامه ریزی کنیم.

این کار را می توان با انتخاب تب **After Build** و فعال کردن گزینه **Action: Upload to Arduino** انجام داد:



در تنظیمات آپلود آردوینو "Arduino Upload Settings"، گزینه های زیر باید تنظیم شوند:

- نوع برد آردوینو "Arduino Board Type" : Arduino Uno ATmega328P
- **COM Port** : پورت سریالی که برای ارتباط با برد توسعه استفاده می شود.

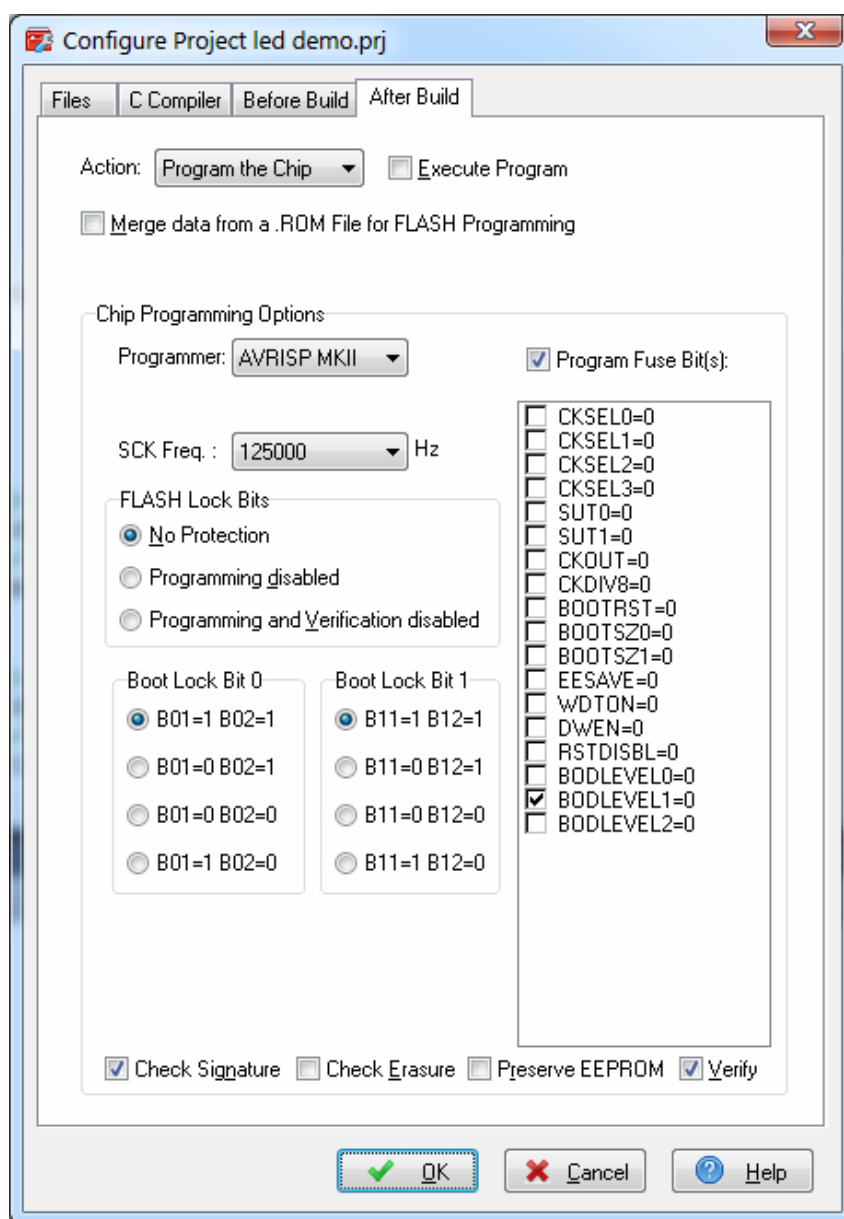
نکته: پورت ارتباط سریال مجازی است و در هنگام اتصال برد آردوینو UNO به کامپیوتر توسط درایور USB FTDI ارائه می شود. توسط افزونه CodeVisionAVR شناسایی می شود و در لیست COM Port ظاهر می شود. (اگر برد Arduino UNO به رایانه متصل باشد)

اگر رایانه شما دارای چندین پورت سریال است، مطمئن شوید که یک پورت صحیح مرتبط با برد Arduino UNO را انتخاب کنید. شماره پورت سریال را می توان با استفاده از منو زیر فهمید:

Windows Control Panel > Hardware and Sound > DeviceManager > Ports (COM & LPT) > Arduino UNO.

همچنین تراشه را می توان با استفاده از برنامه نویسی USB Atmel AVRISP MkII به هدر ISP برد آردوینو UNO متصل است، برنامه ریزی کرد.

در این حالت Action: Program گزینه Chip باید انتخاب شود:



گزینه های زیر باید تنظیم شوند:

- **Programmer:** AVRISP MKII
- **SCK Freq:** 125000 Hz
- **Program Fuse Bit(s) :** enabled
- همه فیوز بیت ها = 0 حالت برنامه ریزی نشده (not checked) به جز **BODLEVEL1=0** که باید در حالت برنامه ریزی شده باشد (checked).

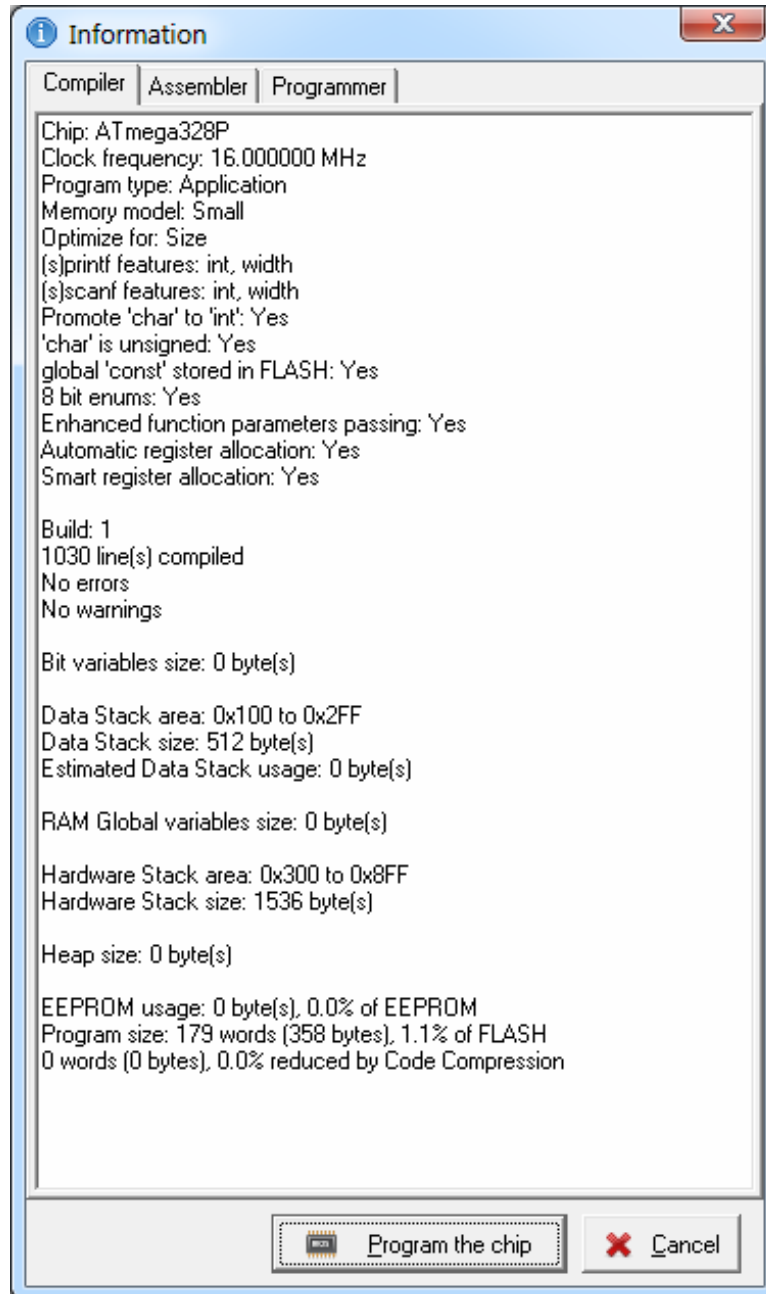
توجه : اگر AVRISP MkII برای برنامه نویسی برد Arduino UNO استفاده شود، بوت لودر از پیش برنامه ریزی شده توسط سازنده در تراشه ATmega328P پاک می شود.

این از هر گونه آپلود آینده با استفاده از درگاه COM مجازی در گذرگاه USB جلوگیری می کند، مگر اینکه بوت لودر و فیوز بیت ها دوباره به درستی برنامه ریزی شوند.

تغییرات پیکربندی پروژه باید با کلیک بر روی دکمه OK تایید شود.

1. ساخت پروژه و برنامه نویسی تراشه

مرحله آخر کامپایل و پیوند دادن برنامه ما با استفاده از دستور **Build|Build led demo menu** است. پس از ساخت موفق، پنجره Information زیر نمایش داده می شود:



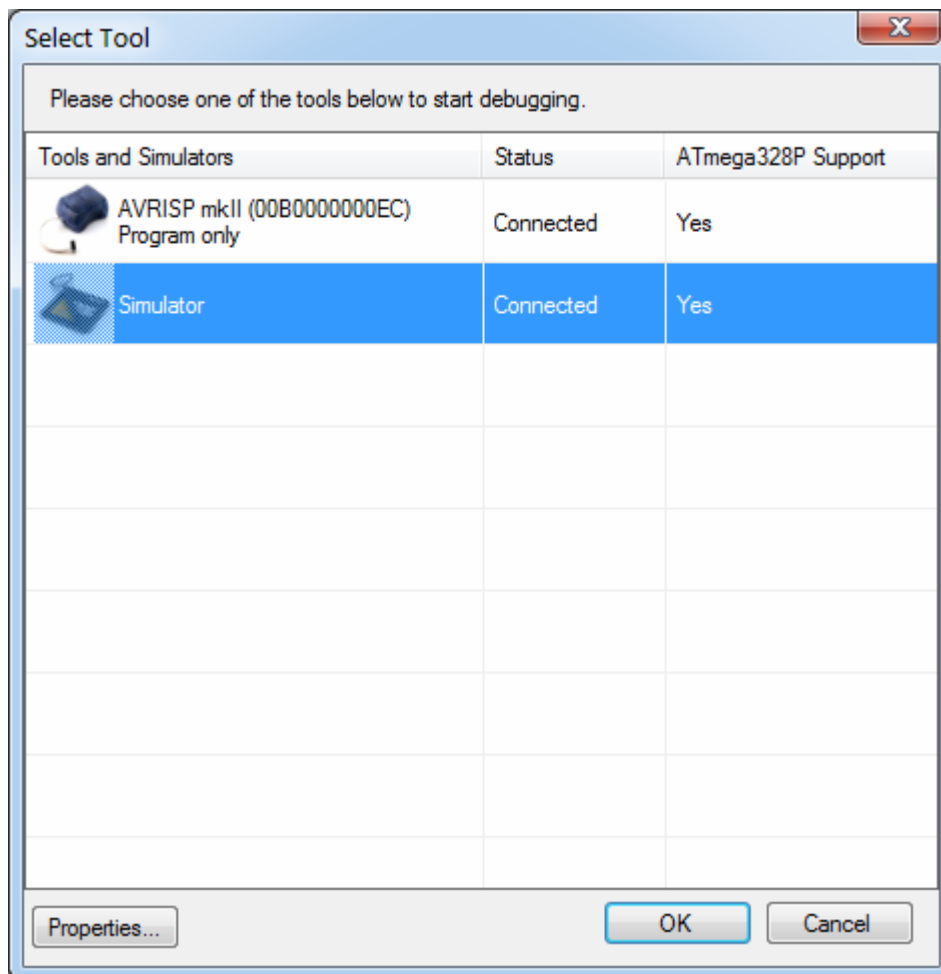
با کلیک بر روی دکمه **Program the chip** برنامه کامپایل شده را به FLASH تراشه منتقل می کند و فیوز بیت ها را برنامه ریزی می کند. پس از انجام این عملیات، برنامه شروع به اجرا می کند.

1. اشکال زدایی برنامه

هنگامی که برنامه با موفقیت ساخته شد، می توان آن را در فرم سورس لول با استفاده از شبیه ساز Atmel Studio اشکال زدایی کرد. یک سشن اشکال زدایی را می توان با استفاده از دستور **Debug|Start Debugging and Break** در منو، دکمه نوار ابزار یا با فشار دادن کلیدهای **Alt+F5** شروع کرد.

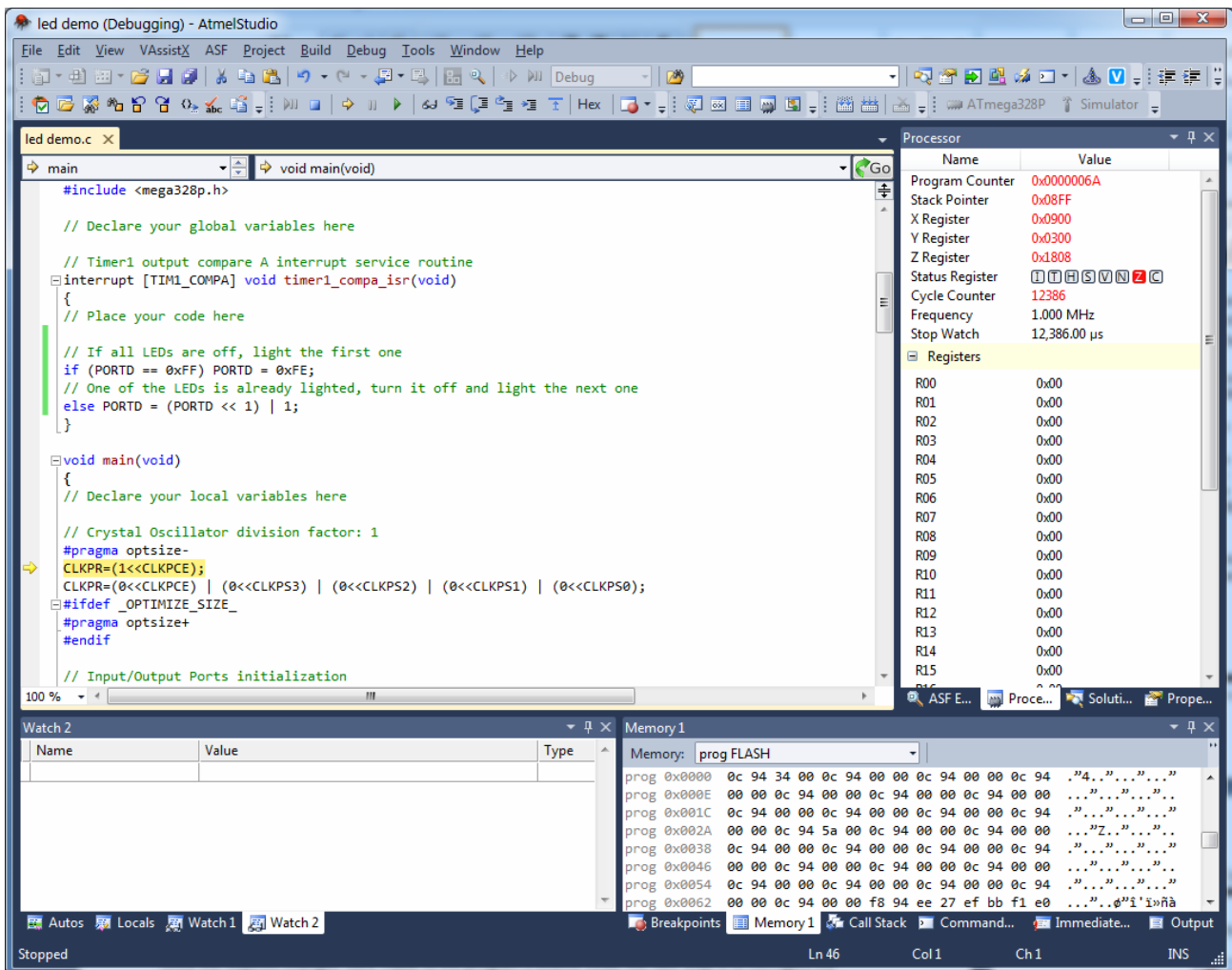
اگر فایل های منبع از آخرین **Build** تغییر کرده باشند، قبل از شروع اشکال زدایی، یک **Rebuild** به طور خودکار انجام می شود. Atmel Studio از فایل شی **.cof** تولید شده توسط CodeVisionAVR برای اشکال زدایی استفاده می کند.

هنگامی که سشن اشکال زدایی برای اولین بار شروع می شود، از کاربر خواسته می شود تا ابزاری را که برای ردیابی اجرای برنامه استفاده می کند انتخاب کند:

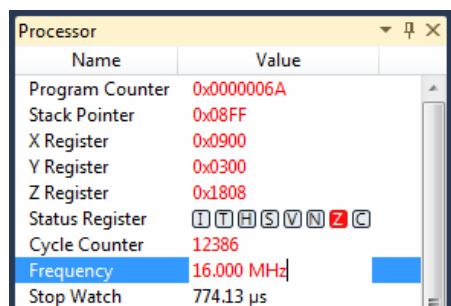


برای مثال ما شبیه ساز **Simulator** را انتخاب کرده و روی دکمه **OK** کلیک می کنیم.

پس از انتخاب ابزار اشکال زدایی، سشن اشکال زدایی شروع می شود:



شبه‌ساز ابتدا کد اولیه راه‌اندازی سطح پایین را اجرا می‌کند که شامل پر کردن تمام مکان‌های RAM با 0، مقداردهی اولیه متغیرهای سراسری، داده‌ها و نشانگرهای پشته سخت‌افزاری است. پس از اتمام این عملیات، اجرای برنامه به عملکرد اصلی برنامه منتقل می‌شود. دیباگر اجرای برنامه را در اولین خط منبع C اصلی متوقف می‌کند و به کاربر این امکان را می‌دهد که برنامه را از آنجا تک مرحله ای کند. رجیسترهای چیب AVR در پنجره Processor نمایش داده می‌شوند. به‌طور پیش‌فرض شبه‌ساز از فرکانس ساعت 1000 مگاهرتز استفاده می‌کند، اما در مثال ما، تراشه با فرکانس 16.000 مگاهرتز کار می‌کند، بنابراین باید با کلیک بر روی قسمت Frequency و وارد کردن مقدار صحیح در آنجا، این فرکانس را تغییر دهید:



and pressing **Enter**.

گام بعدی در اشکال زدایی برنامه ما این است که یک نقطه شکست (breakpoint) در ابتدای خروجی تایمر 1 مقایسه روتین سرویس وقفه timer1_compa_isr است.

این کار با قرار دادن مکان نما در اول خط کد در تابع فوق و انتخاب دستور **Debug|Toggle Breakpoint** یا فشار دادن کلید **F9** به دست می آید:

```

led demo.c x
timer1_compa_isr interrupt [TIM1_COMPA] void timer1_compa_isr(void)
#include <mega328p.h>

// Declare your global variables here

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


// If all LEDs are off, light the first one
if (PORTD == 0xFF) PORTD = 0xFE;
// One of the LEDs is already lighted, turn it off and light the next one
else PORTD = (PORTD << 1) | 1;
}

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

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=(1<<CLKPCE);
CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif
    
```

خط با نقطه شکست (breakpoint) های لایت خواهد شد.

پس از گذاشتن بریک پوینت می توانیم با استفاده از **Debug|Step Into** در منو یا دکمه  در تولبار یا کلید **F11** به اندازه یک مرحله در برنامه به جلو برویم.

با انتخاب دستور **Debug|Continue** در منو، فشردن کلید **F5** یا دکمه  در تولبار می توانیم اجرای برنامه خود را تا رسیدن به نقطه شکست شروع کنیم.

نکته مهم: شبیه ساز Atmel Studio برنامه را با سرعتی بسیار کمتر از تراشه AVR واقعی اجرا می کند، بنابراین رسیدن به نقطه شکست تنظیم شده در تابع `timer1_compa_isr` در زمان واقعی پس از 200 میلی ثانیه رخ نمی دهد. به نظر می رسد بسته به سرعت کامپیوتر میزبان، شبیه ساز به مدت 1 ... 2 دقیقه متوقف می شود، اما در نهایت اجرای برنامه در نقطه بریک پوینت متوقف می شود:

```

led demo.c x
timer1_compa_isr
External RAM size : 0
Data Stack size : 512
*****/

#include <mega328p.h>

// Declare your global variables here

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

// If all LEDs are off, light the first one
if (PORTD == 0xFF) PORTD = 0xFE;
// One of the LEDs is already lighted, turn it off and light the next one
else PORTD = (PORTD << 1) | 1;
}

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

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=(1<<CLKPCE);
CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+

```

هنگامی که اجرای برنامه در تابع `timer1_compa_isr` متوقف شد، ما مقدار `Stop Watch` را $200776.44 \mu s$ در پنجره Processor خواهیم دید.

Name	Value
Program Counter	0x0000005D
Stack Pointer	0x08FD
X Register	0x0900
Y Register	0x02FE
Z Register	0x1802
Status Register	I T H S V N Z C
Cycle Counter	3212423
Frequency	16.000 MHz
Stop Watch	200,776.44 μs

این مقدار کمی بزرگتر از 200 میلی ثانیه است که به عنوان دوره وقفه مقایسه خروجی تایمر 1 تعیین کرده ایم. $776.44 \mu s$ اضافی توسط کد راه اندازی اولیه، پس از تنظیم مجدد تراشه و لحظه ای که رجیسترهای تایمر 1 برای اولین بار مقداردهی اولیه می شوند و شمارش شروع می شود، مورد نیاز است.

برای ادامه اجرای برنامه دوباره کلید F5 را فشار می دهیم.
 بسته به سرعت کامپیوتر میزبان، شبیه ساز پس از 1...2 دقیقه دوباره در نقطه شکست timer1_compa_isr متوقف می شود. این بار مقدار Stop Watch در پنجره Processor مقدار 400776.38 μs خواهد بود:

Name	Value
Program Counter	0x0000005D
Stack Pointer	0x08FD
X Register	0x0900
Y Register	0x02FE
Z Register	0x1802
Status Register	I T H S V N Z C
Cycle Counter	6412422
Frequency	16.000 MHz
Stop Watch	400,776.38 μs

بنابراین فاصله زمانی بین دو وقفه خروجی مقایسه A تایمر 1 مقدار زیر خواهد بود:

$$400776.38 \mu s - 200776.44 \mu s = 199999.94 \mu s = 199.99994 ms$$

که برابر با 200 میلی ثانیه با خطای ناچیز است.

هنگام اشکال زدایی می توان از دستورات اضافی زیر استفاده کرد:

- **Debug|Step Over**، کلید F10 یا دکمه  در نوار ابزار برای اجرای یک دستور.
- اگر دستور العمل حاوی یک فراخوانی تابع باشد، تابع نیز اجرا می شود.
- **Debug|Step Out**، کلیدهای Shift+F11 یا دکمه  در نوار ابزار برای ادامه اجرا تا تابع فعلی کامل شود.
- **Debug|Run To Cursor**، کلیدهای Ctrl+F10 یا دکمه  در نوار ابزار برای ادامه اجرا تا رسیدن به موقعیت مکان نما فعلی در فایل منبع یا رسیدن به دیسبلی ویو
- **Debug|Reset**، کلیدهای Shift+F5 یا دکمه نوار ابزار برای شروع مجدد اجرای برنامه از ابتدا
- **Debug|Restart** یا دکمه  در نوار ابزار برای راه اندازی مجدد دیباگر و بارگیری مجدد برنامه رفع اشکال
- **Debug|Toggle Breakpoint** یا کلید F9 برای تنظیم بریک پوینت در موقعیت مکان نما فعلی در فایل منبع C یا دیسبلی ویو
- **Debug|New Breakpoint|Break at Function** برای تنظیم نقطه بریک پوینت در ابتدای یک تابع خاص
- **Debug|Delete All Breakpoints** یا کلیدهای Ctrl+Shift+F9 برای حذف تمام نقاط شکست تنظیم شده
- **Debug|Disable All Breakpoints** غیرفعال کردن همه نقاط شکست برای غیرفعال کردن موقت تمام نقاط شکست تنظیم شده
- **Debug|All Breakpoints** را فعال کنید تا تمام نقاط شکست تنظیم شده را دوباره فعال کنید
- **Debug|Break All**، کلیدهای Ctrl+F5 یا دکمه  نوار ابزار برای توقف اجرای برنامه
- **Debug|Windows** امکان نمایش پنجره های خاص برای دیدن متغیرها، ثبات های پردازنده، رجیسترهای ورودی/خروجی و محیطی، محتویات حافظه، کد دیسبلی و غیره را می دهد.
- **Debug|Stop Debugging**، کلیدهای Ctrl+Shift+F5 یا دکمه  نوار ابزار سشن دیباگینگ را متوقف می کند.

برای کسب اطلاعات بیشتر در مورد استفاده از دیباگر، لطفاً به راهنمای Atmel Studio مراجعه کنید.

توجه: کامپایلر برخی از تکنیک های بهینه سازی را اعمال می کند که ممکن است از دیباگ کردن صحیح برنامه در حال اجرا جلوگیری کند.

بنابراین توصیه می شود برای اشکال زدایی کد گزینه **Project|Configure|C Compiler|Code Generation|Optimize for: Speed** را انتخاب کنید.

اگر برنامه در FLASH تراشه قرار می گیرد، این گزینه باید برای **Release** نیز فعال بماند، در این حالت برنامه هم سریعتر اجرا می شود.

1. نتیجه گیری

این مقاله دستورالعمل های لازم را برای شروع سریع توسعه و اشکال زدایی یک برنامه با استفاده از افزونه CodeVisionAVR برای Studio Atmel ارائه می دهد. CodeVisionAVR همراه با راهنمای جامع و راهنمای کاربر ارائه شده است که باید به طور کامل مورد مطالعه قرار گیرد تا بتوان از تمام ویژگی های کامپایلر C ، کتابخانه های جانبی مرتبط، CodeWizardAVR و ویرایشگر/تبدیل تصویر/فونت LCD Vision برای نمایشگرهای گرافیکی استفاده کرد. کامپایلر با تعداد زیادی برنامه نمونه ارائه شده است که شامل موارد زیر است:

- Alphanumeric LCD
- Graphic LCD, TFT and OLED displays
- Resistive touch screen
- SD Memory Cards
- USART
- TWI, I²C
- SPI
- ADC
- Boot loaders
- USB
- Web Server
- XMEGA EBI
- XMEGA Quadrature Encoder
- XMEGA DAC
- DS1820, DS18B20, LM75, DS1621 temperature sensors

این برنامه ها در زیر شاخه های `\Examples ATxmega` و `\Examples` دایرکتوری نصب CodeVisionAVR قرار دارند.

Appendix A – The Source Code

```

/*****
This program was created by the
CodeWizardAVR V3.03 Standard
Automatic Program Generator
© Copyright 1998-2013 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Moving LED demo
Version : 1.0
Date    : 14/03/2013
Author  : Pavel Haiduc
Company : HP InfoTech
Comments:
This is a simple program

Chip type           : ATmega328P
Program type        : Application
AVR Core Clock frequency: 16.000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 512
*****/

#include <mega328p.h>

// Declare your global variables here

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

// If all LEDs are off, light the first one
if (PORTD == 0xFF) PORTD = 0xFE;
// One of the LEDs is already lighted, turn it off and light the next one
else PORTD = (PORTD << 1) | 1;
}

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

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=(1<<CLKPCE);
CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

```

```
// Input/Output Ports initialization
// Port B initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) |
(0<<DDB0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) |
(0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRC=(0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) | (0<<DDC0);
// State: Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) |
(0<<PORTC0);

// Port D initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) | (1<<DDD1) |
(1<<DDD0);
// State: Bit7=1 Bit6=1 Bit5=1 Bit4=1 Bit3=1 Bit2=1 Bit1=1 Bit0=1
PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) | (1<<PORTD2) |
(1<<PORTD1) | (1<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A=(0<<COM0A1) | (0<<COM0A0) | (0<<COM0B1) | (0<<COM0B0) | (0<<WGM01) | (0<<WGM00);
TCCR0B=(0<<WGM02) | (0<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 250.000 kHz
// Mode: CTC top=OCR1A
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer Period: 0.2 s
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (1<<WGM12) | (0<<CS12) | (1<<CS11) |
(1<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0xC3;
OCR1AL=0x4F;
```

```

OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2A output: Disconnected
// OC2B output: Disconnected
ASSR=(0<<EXCLK) | (0<<AS2);
TCCR2A=(0<<COM2A1) | (0<<COM2A0) | (0<<COM2B1) | (0<<COM2B0) | (0<<WGM21) | (0<<WGM20);
TCCR2B=(0<<WGM22) | (0<<CS22) | (0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2A=0x01;
OCR2B=0x00;

// Timer/Counter 0 Interrupt(s) initialization
TIMSK0=(0<<OCIE0B) | (0<<OCIE0A) | (0<<TOIE0);

// Timer/Counter 1 Interrupt(s) initialization
TIMSK1=(0<<ICIE1) | (0<<OCIE1B) | (1<<OCIE1A) | (0<<TOIE1);

// Timer/Counter 2 Interrupt(s) initialization
TIMSK2=(0<<OCIE2B) | (0<<OCIE2A) | (0<<TOIE2);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// Interrupt on any change on pins PCINT0-7: Off
// Interrupt on any change on pins PCINT8-14: Off
// Interrupt on any change on pins PCINT16-23: Off
EICRA=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
EIMSK=(0<<INT1) | (0<<INT0);
PCICR=(0<<PCIE2) | (0<<PCIE1) | (0<<PCIE0);

// USART initialization
// USART disabled
UCSR0B=(0<<RXCIEN0) | (0<<TXCIEN0) | (0<<UDRIEN0) | (0<<RXEN0) | (0<<TXEN0) | (0<<UCSZ02) |
(0<<RXB80) | (0<<TXB80);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);
ADCSRB=(0<<ACME);
// Digital input buffer on AIN0: On
// Digital input buffer on AIN1: On
DIDR1=(0<<AIN0D) | (0<<AIN1D);

// ADC initialization
// ADC disabled
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) |
(0<<ADPS1) | (0<<ADPS0);

```

```
// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) |
    (0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

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

while (1)
{
    // Place your code here

}
}
```