

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

آشنایی با میکروکنترلرهای

AVR و نرم افزار CodevisionAVR

امیر ره افروز

در صورتی که علاقه مندید تا اطلاعات جامعتری را در مورد میکروکنترلرهای AVR بدست آورید می توانید کتاب زیر را تهیه کنید:



برای اطلاعات بیشتر می توانید به سایتهای زیر مراجعه کنید:

http://profiles.yahoo.com/am_rahafrooz
<http://bme.aut.ac.ir/arahafrooz>

علاوه بر این برای تماس با اینجانب می توانید از آدرسهای پست الکترونیکی زیر استفاده کنید:

arahafooz@cic.aut.ac.ir

am_rahafrooz@yahoo.com

موفق باشید.

فصل اول

آشنایی با AVR

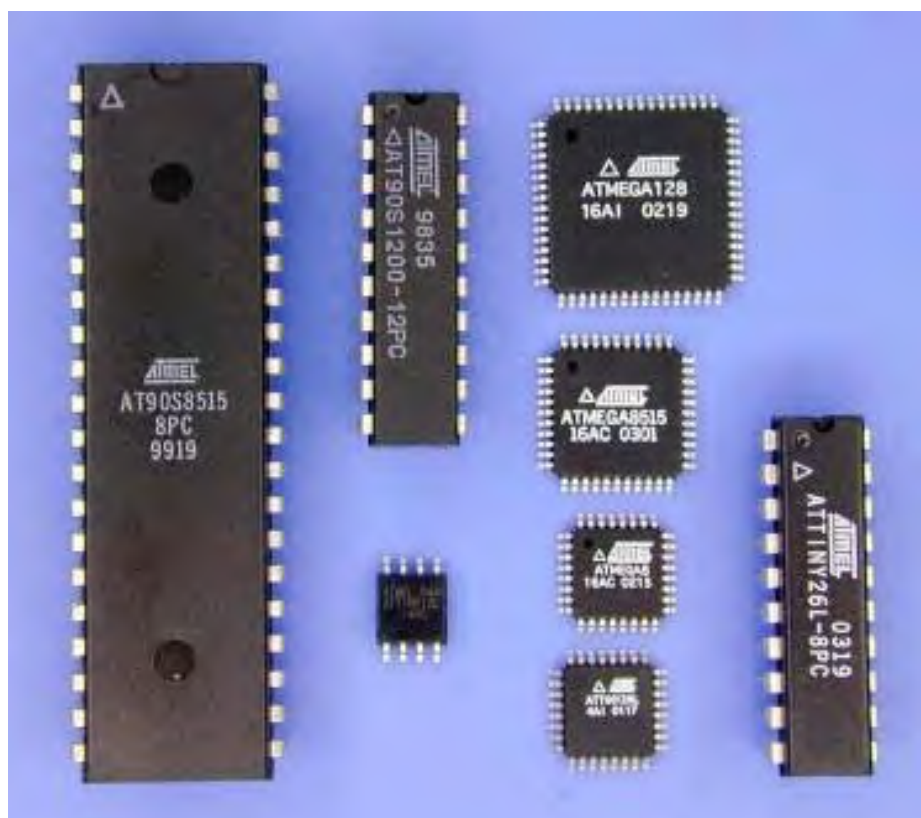
AVRها میکروکنترلرهای ۸ بیتی از نوع CMOS با توان مصرفی پایین هستند که بر اساس ساختار پیشرفته RISC ساخته شده‌اند. پس از ساخت اولین نسخه‌های AVR در سال ۱۹۹۶، این سری از میکروکنترلرها توانست نظر علاقمندان الکترونیک را به خود جذب کند به طوری که امروزه یکی از پرمصرفترین انواع میکروکنترلرها به حساب می‌آید. همان طور که می‌دانید نمی‌توان هیچ نوع میکروکنترلری را به عنوان بهترین معرفی کرد چرا که هر میکروکنترلر، کاربرهای

خاص خود را دارد و بر اساس خصوصیات داخلی، می‌تواند تنها برای موارد ویژه‌ای به عنوان بهترین انتخاب گردد، ولی با این حال با مطالعه صفحات بعدی و آشنایی با امکانات و نرم افزارهای جانبی AVR متوجه خواهید شد که در کل استفاده از AVR بر بقیه ترجیح دارد.

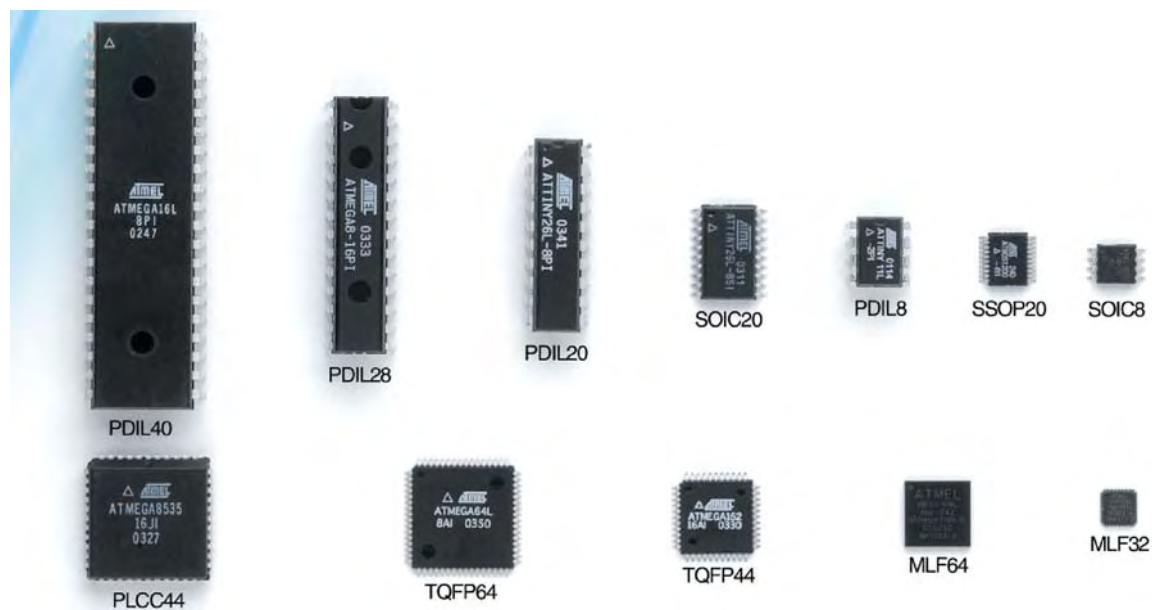
AVRها با ساختار RISC، دستورات را تنها در یک پالس ساعت اجرا می‌نمایند و به این ترتیب می‌توانیم تا به ازای هر یک مگاهرتز، یک مگادستور را در ثانیه (MIPS) اجرا کرده و برنامه را از لحاظ سرعت پردازش و نیز مصرف توان بهینه کنیم. AVRها، ۳۲ رجیستر همه منظوره (R0..R31) و مجموعه دستورات قدرتمندی را شامل می‌گردند. تمام این ۳۲ رجیستر مستقیماً به ALU متصل شده‌اند، بنابراین دسترسی به دو رجیستر در یک سیکل ساعت هم امکان‌پذیر است. این ساختار موجب می‌گردد تا سرعت آنها نسبت به میکروکنترلرهای CISC بتواند تا ۱۰ برابر هم افزایش یابد.

خانواده میکروکنترلرهای AVR، تراشه‌هایی پیشرفته با امکانات جانبی کامل هستند. زمانی که شروع به یادگیری مفاهیم اصلی آنها نمایید، لذت فراگیری تمام جزئیاتشان آغاز می‌شود.

میکروکنترلرهای AVR به سه دسته تقسیم می‌شوند:



شکل ۱-۱ انواع میکروکنترلرهای AVR.



شکل ۱- ۲ انواع مختلف میکروکنترلرهای AVR از ۸ پایه تا ۱۶ پایه

Tiny AVR-

AVR(classic AVR)-

Mega AVR-

تفاوت بین این سه نوع به امکانات موجود در آنها مربوط می‌شود. Tiny AVR ها غالباً تراشه‌هایی با تعداد پین و مجموعه دستورات کمتری نسبت به Mega AVR می‌باشند و به عبارتی از لحاظ پیچیدگی حداقل امکانات را دارند. Mega AVR ها حداکثر امکانات را داشته و AVR(classic AVR) ها در بین این دو نوع قرار می‌گیرند.

تمام تراشه‌های AVR مجموعه دستورات و ساختار حافظه مشابهی دارند و از این رو تغییر از یک تراشه به تراشه دیگر کاری بسیار ساده است.

امکانات کلی یک AVR

- در حدود ۱۳۰ دستور که اکثر آنها در یک سیکل ساعت اجرا می‌شوند.
- ۳۲ رجیستر ۸ بیتی همه منظوره.
- ضرب کننده سخت افزاری با زمان اجرای ۲ سیکل ساعت.
- دارای سه نوع حافظه FLASH (برای کدهای برنامه)، SRAM, EEPROM.

- برنامه‌ریزی تراشه در داخل مدار موردنظر بدون نیاز به پروگرامر (ISP)^۱.
- حفاظت از کدهای برنامه در مقابل خواندن.
- قابلیت تنظیم نوسانگر برای کار توسط کریستال خارجی، کریستال فرکانس پایین خارجی، نوسانگر RC خارجی، نوسانگر RC داخلی و فرکانس خارجی.
- مجهز به پروتکل JTAG برای انجام عمل دیباگ، تست واسکن کردن وسایل جانبی تراشه و نیز برنامه‌ریزی حافظه‌های FLASH, EEPROM و فیوزها.
- شمارنده و تایمر ۸ بیتی.
- شمارنده و تایمر ۱۶ بیتی.
- RTC^۲ با نوسانگر جداگانه.
- کانالهای PWM (با استفاده از تایمرها به صورت ۸ بیتی و ۱۶ بیتی برای تولید PWM)
- امکان تنظیم تایمر به صورت CTC^۳.
- ADC^۴های ۱۰ بیتی با یک ورودی و یا ورودی تفاضلی با بهره قابل تنظیم ۱، ۱۰، ۲۰۰.
- مجهز به پروتکل I2C^۵ یا TWI^۶، ارتباط دو سیمه از شرکت Philips.
- ارتباط سریال USART^۷ با قابلیت برنامه‌ریزی.
- ارتباط سریال SPI^۸ به صورت Master/Slave.
- تایمر نگهبان^۹، قابل برنامه‌ریزی با نوسانگر مجزا.
- مقایسه کننده آنالوگ با امکان تعریف وقفه برای آن.
- RESET شدن در زمان اتصال به برق^{۱۰}.
- Brown-out Detector با قابلیت برنامه‌ریزی.

In System Programming
Real Time Clock
Clear Timer On Compare Match
Analog to Digital Converter
Inter-IC bus
Two wire Interface
Universal Synchronous and Asynchronous serial Receiver and Transmitter
Serial peripheral Interface
Watch Dog Timer
Power On Reset

- منابع وقفه داخلی و خارجی.

- با شش حالت مختلف برای کاهش توان مصرفی.

- کار با ولتاژهای ۵,۵-۴,۵ در مدل‌های بدون پسوندد L مثل ATmega32, و ۵,۵-۲,۷ در مدل‌های با

پسوند L مثل (L) ATmega 32.

ممکن است بعضی از امکانات ذکر شده در بعضی از انواع ساده AVR وجود نداشته باشد و حجم فضاهای Sram, Eeprom, Flash هم در آنها متفاوت باشد. لذا برای انتخاب صحیح AVR اگر شما از لحاظ مالی محدودیت دارید باید نوعی از AVR را انتخاب کنید که تنها امکانات مورد نیاز شما را داشته باشد. ولی در صورتی که محدودیتی ندارید، مسلماً باید قویترین AVR ممکن را انتخاب کنید. چرا!!...

ممکن است از خود بپرسید، چگونه باید یاد بگیریم تا کدهای AVR را بنویسیم!

یادگیری یک موضوع جدید جالب است ولی می‌تواند کمی هم مایوس کننده باشد. با این وجود امکان فراگیری AVR تنها از طریق مطالعه برگه های اطلاعاتی آن امکان پذیر می‌باشد که روشی وقتگیر و پیچیده است.

ما راه دیگری را ارائه می‌کنیم:

۱- مقداری برنامه از پیش نوشته شده پیدا کنید.

۲- چگونگی عملکرد کدها را بفهمید.

۳- آن را به گونه‌ای تغییر دهید که نیاز شما را برآورده کند.

چگونه از برگه های اطلاعاتی AVR استفاده کنیم؟

زمانی که به برگه های اطلاعاتی AVR نگاه کنید احتمالاً خواهید ترسید، برای مثال برگه اطلاعاتی مربوط به (L) ATmega128 تقریباً ۳۵۰ صفحه است و مطالعه کل آن و به یاد سپردن محتوای آن کار مشکلی است. خوشبختانه لازم نیست تا این کار را انجام دهید. برگه های اطلاعاتی، مجموعه‌ای از اطلاعات کامل و حرفه‌ای می‌باشند و شما می‌توانید در زمانی که نسبت به عملکرد امکانات AVR و یا وسایل جانبی مطمئن نیستید از آنها به عنوان مرجع استفاده کنید.

زمانی که شما یک برگه های اطلاعاتی مربوط به یکی از انواع AVR را مشاهده نمایید، می‌توانید آن را به قسمتهای زیر

تقسیم کنید.

۱- اولین صفحه شامل اطلاعات کلیدی و لیست امکانات میکروکنترلر.

۲- خلاصه‌ای از ساختار AVR.

۳- توضیح وسایل جانبی.

۴- برنامه‌ریزی حافظه.

۵- مشخصات الکتریکی.

۶- خلاصه رجیسترها.

۷- خلاصه مجموعه دستورات.

۸- اطلاعات مربوط به ابعاد و شکل ظاهری میکروکنترلر.

زبان برنامه‌نویسی C

امروزه زبان سطح بالای C هر چه بیشتر برای برنامه‌نویسی میکروکنترلرها عمومیت یافته است. مزایای استفاده از زبان C به جای زبان اسمبلی زیادند: کاهش زمان برنامه‌نویسی، نگهداری ساده‌تر و قابلیت حمل بیشتر، استفاده مجدد از کدها و ساده‌تر شدن فهم برنامه. عیب آن هم افزایش حجم کدها و در نتیجه کاهش سرعت برنامه می‌باشد. برای کاهش یا رفع این معایب ساختار AVR به گونه‌ای طراحی شده تا دستورات تولید شده توسط کامپایلر C را به صورت مؤثر دیکد و اجر نماید.

طراحی کامپایلر C قبل از تکمیل ساختار و مجموعه دستورات AVR صورت گرفته است. نتیجه این عملکرد همزمان بین تیمهای کامپایلر و AVR، میکروکنترولی با کدهای تولید شده جهت استفاده بهینه و کارایی بالا می‌باشد.

در این کتاب بر آنیم تا ابتدا با نرم افزار CodevisionAVR برای کامپایل کردن برنامه‌های C آشنا شده، و چند برنامه کاربردی را توضیح داده و اجرا کنیم. در اینجا از تراشه ATmega32 استفاده می‌کنیم. در شکل (۱-۳) کلی تراشه و عملکرد پایه‌های مختلف این میکروکنترلر را می‌بینید. در مورد سخت افزار لازم برای راه اندازی تراشه در فصل دوم توضیح خواهیم داد.

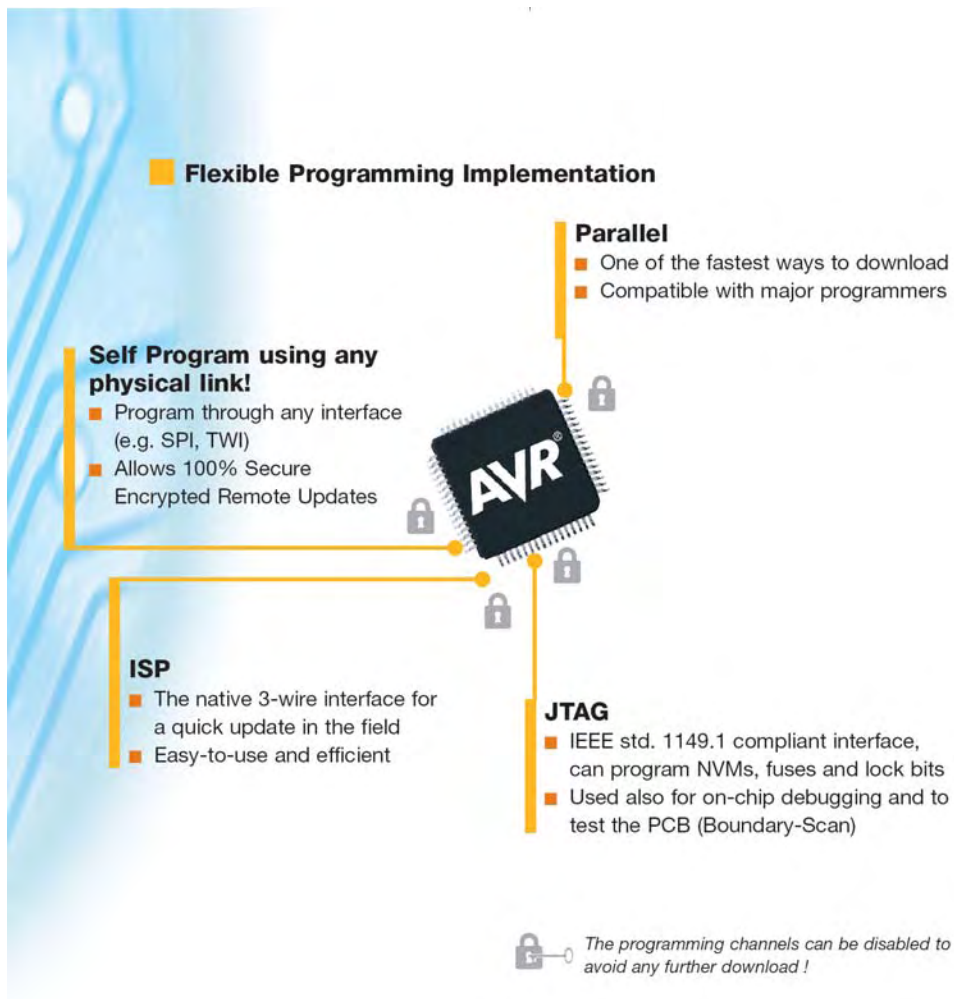
PDIP

(XCK/T0) PB0	<input type="checkbox"/>	1	40	<input type="checkbox"/>	PA0 (ADC0)
(T1) PB1	<input type="checkbox"/>	2	39	<input type="checkbox"/>	PA1 (ADC1)
(INT2/AIN0) PB2	<input type="checkbox"/>	3	38	<input type="checkbox"/>	PA2 (ADC2)
(OC0/AIN1) PB3	<input type="checkbox"/>	4	37	<input type="checkbox"/>	PA3 (ADC3)
(\overline{SS}) PB4	<input type="checkbox"/>	5	36	<input type="checkbox"/>	PA4 (ADC4)
(MOSI) PB5	<input type="checkbox"/>	6	35	<input type="checkbox"/>	PA5 (ADC5)
(MISO) PB6	<input type="checkbox"/>	7	34	<input type="checkbox"/>	PA6 (ADC6)
(SCK) PB7	<input type="checkbox"/>	8	33	<input type="checkbox"/>	PA7 (ADC7)
\overline{RESET}	<input type="checkbox"/>	9	32	<input type="checkbox"/>	AREF
VCC	<input type="checkbox"/>	10	31	<input type="checkbox"/>	GND
GND	<input type="checkbox"/>	11	30	<input type="checkbox"/>	AVCC
XTAL2	<input type="checkbox"/>	12	29	<input type="checkbox"/>	PC7 (TOSC2)
XTAL1	<input type="checkbox"/>	13	28	<input type="checkbox"/>	PC6 (TOSC1)
(RXD) PD0	<input type="checkbox"/>	14	27	<input type="checkbox"/>	PC5 (TDI)
(TXD) PD1	<input type="checkbox"/>	15	26	<input type="checkbox"/>	PC4 (TDO)
(INT0) PD2	<input type="checkbox"/>	16	25	<input type="checkbox"/>	PC3 (TMS)
(INT1) PD3	<input type="checkbox"/>	17	24	<input type="checkbox"/>	PC2 (TCK)
(OC1B) PD4	<input type="checkbox"/>	18	23	<input type="checkbox"/>	PC1 (SDA)
(OC1A) PD5	<input type="checkbox"/>	19	22	<input type="checkbox"/>	PC0 (SCL)
(ICP) PD6	<input type="checkbox"/>	20	21	<input type="checkbox"/>	PD7 (OC2)

فصل دوم

مقدمه

دو روش مختلف جهت برنامه‌ریزی تراشه‌های AVR وجود دارد که برنامه‌ریزی موازی و سریال (حالت ISP) نامیده می‌شوند. در حالت موازی تراشه موردنظر در سوکت پروگرامر قرار داده می‌شود. در این روش لازم است تا ولتاژ ۱۲+ ولت به پایه RESET اعمال گردد. ارتباط بین پروگرامر و تراشه موردنظر نیز همان طور که از اسمش پیداست به کمک دستورات برنامه‌نویسی موازی برقرار می‌شود و به همین دلیل سرعت برنامه‌ریزی در این روش دو برابر سریعتر از حالت ISP است. برخلاف روش ISP که بعضی از انواع AVR از آن پشتیبانی نمی‌کنند، روش برنامه‌ریزی موازی می‌تواند برای انواع AVR به کار برده شود. بنابراین به طور کلی از این روش در مواردی که تولید انبوه موردنظر است استفاده می‌گردد، اما آنچه در این بخش موردنظر ماست، برنامه‌ریزی سریال است.



شکل ۱-۲ روشهای مختلف برنامه ریزی میکروکنترلرهای AVR

برنامه ریزی سریال (ISP):

از آنجایی که برنامه‌ریزی سریال زمانی انجام می‌شود که تراشه قبلاً بر روی برد موردنظر قرار گرفته است به این روش ISP می‌گویند. قبل از اینکه به توضیح قسمتهای مختلف در این روش بپردازیم لازم است قسمتی از حافظه AVR، به

نام Boot Loader را معرفی نماییم. در انواعی از AVR که از روش ISP حمایت می‌کنند. حافظه FLASH، به دو قسمت تقسیم می‌شود که اولین قسمت همان Boot Loader است. کدهای موجود در این قسمت امکان ISP را برای ما ایجاد می‌نمایند. به عبارتی ساده‌تر به کمک برنامه موجود در این قسمت، AVR کدهای برنامه را از PC گرفته و خودش را برنامه‌ریزی می‌کند. جالب است بدانید که این قسمت (Boot Loader) می‌تواند خودش را هم برنامه‌ریزی مجدد نماید.

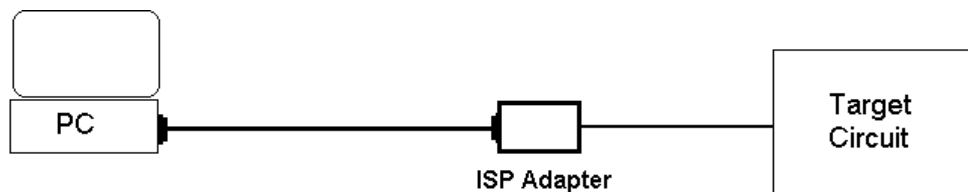
هدف ما در این قسمت توضیح و ساخت سخت افزار واسطی است که امکان برقراری ارتباط بین PC و برنامه‌Boot Loader بر روی AVR را ایجاد می‌کند. این وسیله Programming adapter نامیده می‌شود.

به طور کلی در این بخش دو پروتوکل مختلف وجود دارند، که جهت برنامه‌ریزی به کار می‌روند. اولین پروتوکل SPI^۱ و دومی JTAG نام دارند. در عمل آنچه بیشتر به کار می‌رود همان SPI است. البته به کمک پروتوکل JTAG نه تنها قادریم تا تراشه موردنظر را برنامه‌ریزی کنیم، بلکه با تأمین سخت‌افزار و نرم‌افزار کافی می‌توانیم عمل Debugging را هم انجام دهیم و بنابراین به وسایل برنامه‌ریزی پیچیده‌تر با قیمتی بسیار بیشتر از یک ارتباط ساده SPI نیاز خواهد بود، به این ترتیب به دلیل قیمت بالای آن در مقایسه با روش SPI، اصولاً از JTAG برای Debugging استفاده می‌شود.

علاوه بر دو روش قبل، روش برنامه‌ریزی سریالی با استفاده از ولتاژ ۱۲ ولت هم وجود دارد که روش برنامه‌ریزی سریال ولتاژ بالا نامیده می‌شود. از آنجایی که این روش تنها برای تراشه‌های ۸ پایه به کار می‌رود، در اینجا به آن اشاره نمی‌کنیم.

در روش ISP، با استفاده از پروتوکل SPI می‌توانیم علاوه بر برنامه‌ریزی تراشه، صحت برنامه نوشته شده را هم تشخیص دهیم (verify). در این روش تنها به سه سیگنال احتیاج داریم، یک خط ورودی، یک خط خروجی و یک خط Clock. علاوه بر این، پین RESET در تراشه هم باید در حین برنامه‌ریزی پایین (صفر منطقی) نگه داشته شود ضمناً به ولتاژ ۱۲ ولت هم نیازی نداریم. با این حال در بعضی از تراشه‌ها امکان تنظیم فیوزها را نداریم و حتی بعضی از آنها اصلاً از قابلیت SPI حمایت هم نمی‌کنند. چنین تراشه‌هایی باید به روش موازی برنامه‌ریزی شوند.

واسطه‌های ISP



ISP -

عباراتی مثل “واسطه‌های برنامه‌ریزی سریال و موازی” که اغلب به کار می‌روند ممکن است معنی نادرستی برای ما داشته باشد، چرا که بعضی از مولدهای واسطه ISP، از پورت موازی کامپیوتر استفاده می‌کنند در صورتی که بقیه به پورتهای سریال کامپیوتر (COM) متصل می‌شوند، ولی هر دوی آنها از روش برنامه‌ریزی سریال استفاده می‌کنند. (شکل ۲-۲)

واسطه‌های ISP برای پورت موازی بسیار ساده‌تر هستند. بعضی از انواع آنها حتی هیچ احتیاجی به عناصر فعال هم ندارند ولی شما می‌توانید تصور کنید که اتصال مستقیم مدار به پورت موازی ممکن است باعث ایجاد مشکل شود. بنابراین اکثر واسطه‌ها از یک درایو خط استفاده می‌کنند. ولی با این حال هنوز هم چندان ایمن نیست. پورت موازی در مقابل اتصال کوتاه و یا اضافه بار چندان مقاوم نیست و ممکن است با قطع و وصل تغذیه مورد، پورت آسیب ببیند.

در مقابل پورت موازی، پورت سریال کامپیوتر به خوبی حفاظت می‌گردد. دو نوع مدار واسطه برای پورت COM وجود دارد. نوع ساده‌تر با استفاده مستقیم از خطوط پورت، پالسهای برنامه‌ریزی لازم را ایجاد می‌کند. نوع دوم، دستورات برنامه‌ریزی و اطلاعات برنامه را به مدار واسطه‌ای می‌فرستد، که مدار واسطه آنها را تغییر داده و پالسهای لازم را تولید می‌کند. این نوع از مدار واسطه به نوعی توانایی پردازش هم نیاز دارد که اصولاً به کمک یک میکروکنترلر تأمین می‌گردد.

از آنجایی که صنعت کامپیوتر به سمت USB^۱ حرکت می‌کند با کامپیوترهای کیفی که حتی به پورتهای سریال و موازی مجهز نشده‌اند، این ارتباط بسیار مهم خواهد بود. سخت‌افزار آماده برای برنامه‌ریزی به کمک USB نیز وجود دارد که راجع به آن صحبتی نخواهیم کرد.

شرکت Atmel، پس از تولید AVR، کیت‌های آماده‌ای را برای افراد تازه کاری که قصد دارند کار با AVR را فرا گیرند طراحی نمود. این کیت‌ها به گونه‌ای طراحی شده‌اند تا فرد به کمک این کیت‌ها، امکان به کارگیری تمام امکانات AVR را داشته باشد. ولی آنچه در مورد این کیت‌ها برای ما مهم است، تنها شیوه اتصال آنها به کامپیوتر است. این کیت‌ها به چند دسته تقسیم می‌شوند که هر یک تنها برای کار با تعدادی از انواع تراشه‌ها طراحی شده‌اند. در ادامه چند مورد از آنها را ذکر می‌کنیم:

Kanda systems STK200+/300

Atmel STK500/AVRISP

Dentronics DT006

Vogel Elektronik VTEC-ISP

Futurlec JRAVR

Microtronics ATCPU/Mega2000

آنچه در اینجا مدنظر ماست مدار واسط برای برقراری ارتباط ISP، در نوع STK200/STK300 است. STK200

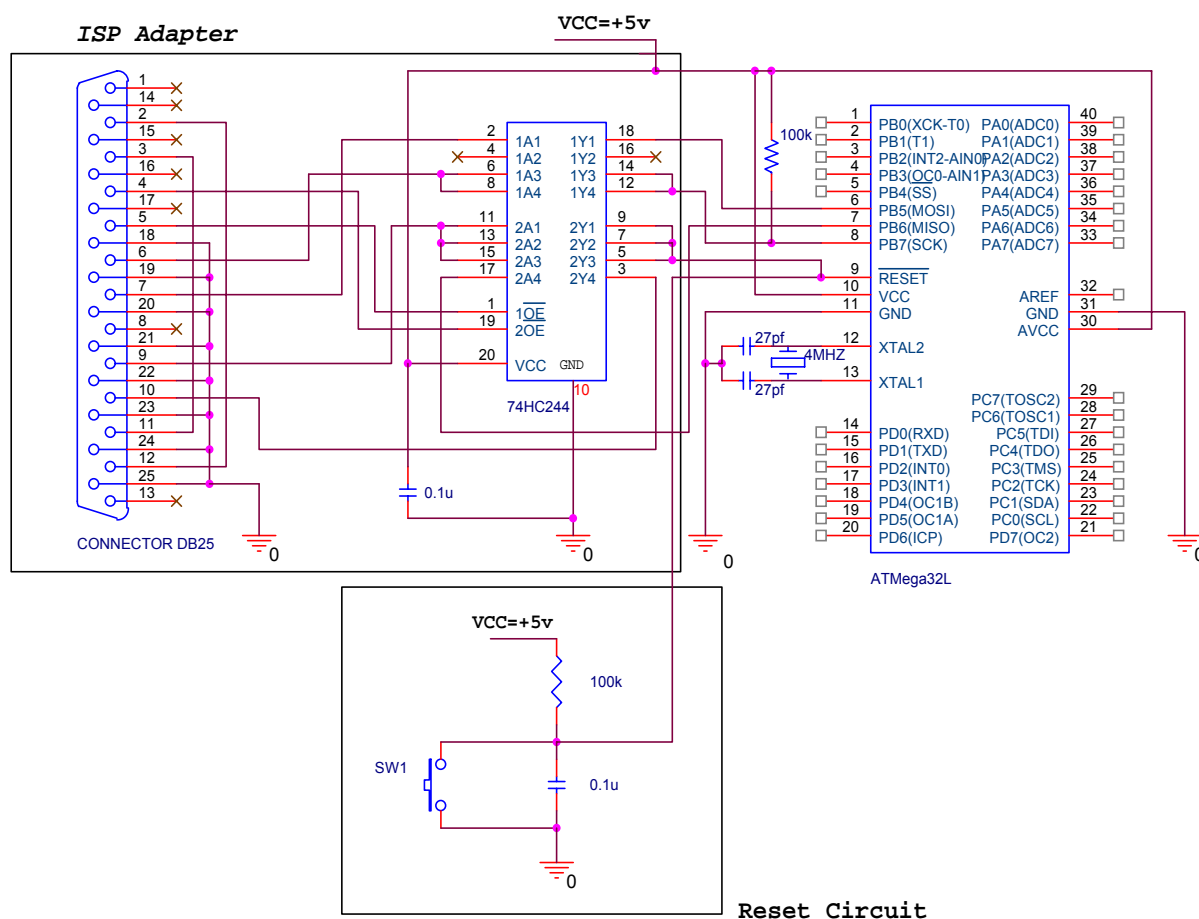
برای کار با پردازنده‌های AVR و STK300 برای کار با پردازنده‌های ATmega طراحی شده‌اند. این دو برد ساختار بسیار

مشابهی دارند و تنها تفاوت بین آنها در محل قرارگیری سیم مربوط به شناسایی برد می‌باشد. مدار مربوط به این مدلها در شکل

(۲-۳) نشان داده شده است که البته چندان هم پیچیده نیست. ممکن است در اولین نگاه فریبده باشد، چرا که شکل بیش از

حد نیاز پیچیده شده است.

Minimum Hardware for working with AVR



AVR

این واسطه حتی می‌تواند بدون هیچ قطعه الکترونیکی هم ساخته شود، ولی اگر آی سی بافر وجود نداشته باشد باید

مراقب باشیم تا پینهای مربوط به برنامه‌ریزی را به مدار دیگری متصل نکنیم. البته در صورت استفاده از این مدار، هنوز هم باید

در مورد پینهای $MOSI^1$ ، $MISO^2$ و SCK^3 احتیاط نماییم. واضح است که ما باید در زمان برنامه‌ریزی مطمئن شویم که این پینها توسط وسیله‌های خارجی دیگر به GND یا Vcc متصل نشده باشند.

در زمانی که AVR در حالت برنامه‌ریزی نیست، 74HC244 به عنوان یک مدار ایزوله کننده عمل می‌کند. در این حالت تمام خروجی‌ها امپدانس بالا می‌شوند، به طوری که بر عملکرد تراشه تأثیر نخواهند گذاشت.

زمانی که حالت ISP انتخاب گردد، نیمه پایین 74HC244 فعال می‌شود و خط RESET از سیستم موردنظر را زمین می‌کند. این عمل توسط سه بافر صورت می‌گیرد و از آنجایی که باید خازن مربوط به مدار RESET در تراشه دشارژ گردد، این کار به خوبی انجام می‌شود. زمانی که تراشه در حالت RESET قرار گرفت، با فرض این که خطوط SCK ، $MOSI$ و $MISO$ در تراشه، تحت تأثیر هیچ مدار جانبی قرار نداشته باشند، زمان آن است تا نیمه بالایی 74HC244 فعال گردد و خطوط $MOSI$ ، $MOSI$ را درایو نماید.

دو اتصال کوتاه موجود در کانکتور پورت پرینتر توسط کامپیوتر، جهت شناسایی پروگرامر استفاده می‌شوند. اتصال کوتاه پینهای ۲ و ۱۲ به عنوان خط تشخیص برای STK200 و اتصال کوتاه پینهای ۳ و ۱۳ به عنوان خط تشخیص برای STK300 به کار می‌روند و هر دوی آنها می‌توانند به طور همزمان با یکدیگر هم وجود داشته باشند.

لازم به ذکر است، در صورتی که قصد دارید تا به جای Atmega32L از تراشه دیگری استفاده کنید، برای برنامه ریزی آن تنها کافی است تا پایه های $MOSI$ ، $MISO$ ، SCK ، $RESET$ را همانند مدار قبل به کابل ISP متصل نمایید.

همان طور که قبلاً هم گفته شد، زمانی که قرار است از AVR در مد ISP استفاده شود، مورد مورد نظر باید به گونه‌ای طراحی شود تا پینهای ISP ($MOSI$ ، $MISO$ ، SCK ، $RESET$) برای این عمل ذخیره گردند، با این حال ممکن است در عمل تعداد پورتها کافی نباشد.

در صورتی که موارد لازم رعایت گردد پینهای ISP می‌توانند به صورت مشترک برای عمل ISP و نیز به عنوان I/O به کار روند. برای این منظور در نظر گرفتن موارد زیر ضروری است. (شکل ۲-۴)

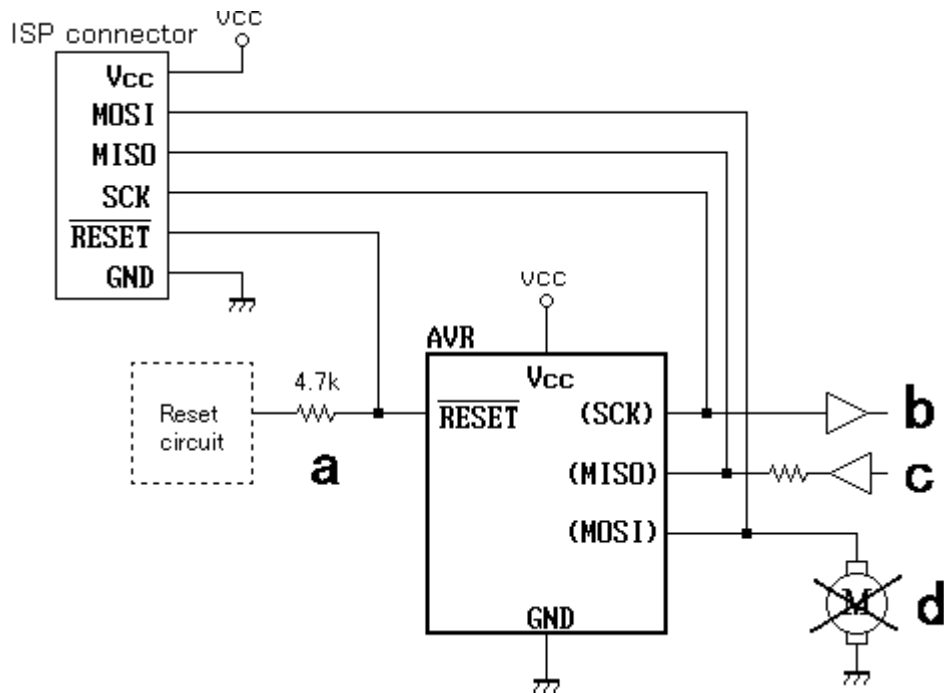
۱- قرار دادن یک مقاومت بین پین $RESET$ و مدار $RESET$. در این صورت مدار $RESET$ اختلالی

ایجاد نخواهد کرد.

Master Out Slave In

Master In Slave Out

Serial Clock



شکل ۲-۴ ملاحظات لازم برای مدار ISP

۲- مطمئن شوید که عمل ISP، هیچ عمل دیگری را تحت تأثیر قرار نمی‌دهد.

۳- در حین عمل ISP، از درایو کردن مدارهای خروجی خودداری کنید و یا باید از یک مقاومت استفاده

نمایید.

۴- از اتصال بارهای سنگین که بر روی عمل ISP تأثیر می‌گذارند خودداری کنید.

لازم به ذکر است، در صورتی که قصد دارید تا ATmega128 را به روش ISP برنامه‌ریزی نمایید از آنجایی که خطوط ورودی و خروجی SPI با خطوط ارسال و دریافت اولین UART در تراشه مشترک هستند. برای جلوگیری از بروز مشکل از سخت‌افزار بیشتری استفاده می‌شود و پینهای آن هم تفاوت می‌کند که این مطلب به خواننده واگذار می‌شود.

در پایان لازم است به این نکته هم توجه کنید؛ در صورتی که شما فیوزهای AVR را به طور غیرصحیح برنامه‌ریزی نمایید ممکن است کریستال و نوسانگر داخلی از کار بیفتد. یکی از بدترین کارهایی که شما می‌توانید انجام دهید برنامه‌ریزی غیرصحیح فیوزها و غیرفعال کردن امکان ISP, JTAG است. در این حالت هیچ راهی جز تعویض تراشه AVR وجود ندارد، البته شما هنوز هم می‌توانید آن را به حالت اول بازگردانید ولی این کار تنها از طریق برنامه‌ریزی موازی میسر خواهد بود که اصولاً کار مشکلی است و امکانات خاص خود را می‌طلبد.

فصل سوم

آشنایی با Codevision AVR

Codevision AVR یک کامپایلر C، محیط توسعه یافته یکپارچه (IDE)¹ و تولیدکننده خودکار کدهای برنامه است که برای کار با میکروکنترلرهای AVR ساخت شرکت ATMEL طراحی شده است. برنامه طوری طراحی شده که در ویندوزهای 95, 98, Me, NT, 2000, Xp قابل اجرا باشد. این برنامه یک پروگرامر ISP را هم شامل می‌شود که امکان انتقال کدهای برنامه به میکروکنترلرها را بعد از انجام موفق عمل کامپایل فراهم می‌کند. این نرم افزار علاوه بر کتابخانه‌های C استاندارد، دارای کتابخانه‌های دقیقی برای کار با LCDهای کارکتری، تولید وقفه، تنظیم انرژی مصرفی تراشه، قابلیت SPI، قابلیت I²C، ارتباط یک سیمه، کار با سنسورهای دمای LM75، DS1820 و EEPROMهای سریال است.



شکل ۳-۱ نرم افزار CodevisionAVR

علاوه بر این CodewizardAVR, CodevisionAVR، تولیدکننده خودکار برنامه را هم شامل می‌شود. این برنامه این امکان را به ما می‌دهد تا برای تنظیم امکانات مختلف تراشه از قبیل تایمرها، ADC، SPI، TWI و ... بدون نیاز به نوشتن کد برای آنها به صورت گرافیکی تنظیمات اولیه مورد نیاز را انجام دهیم در این صورت Codewizard کدهای لازم را برای ما تولید می‌کند و به این صورت می‌توانیم در کمترین زمان ممکن برنامه‌های خود را بنویسیم.

در این بخش قصد داریم تا اولین برنامه خود را بنویسیم و آن را اجرا کنیم، تا اینجا شما باید کابل ذکر شده در بخش قبل را ساخته و حداقل مدار لازم را نیز آماده کرده باشید. (شکل ۲-۳) علاوه بر این باید نرم افزار CodevisionAVR و AVRstudio را هم نصب کرده باشید. ضمناً می‌توانید نسخه رایگان نرم افزار CodevisionAVR را از آدرس اینترنتی زیر دریافت کنید:

<http://www.hpinfotech.ro>

برای دریافت نرم افزار AVRStudio هم به سایت شرکت Atmel مراجعه نمایید:

<http://www.atmel.com>

در این قسمت می‌خواهیم تا کار را با نوشتن یک برنامه ساده آغاز کنیم و یک LED را با فرکانس 2Hz

خاموش و روشن کنیم.

ایجاد یک پروژه جدید

اولین کار ایجاد پروژه جدیدی برای برنامه موردنظر است. می‌توانید این کار را با استفاده از منوی File|New یا فشار دادن دکمه Create New File در نوار ابزار انجام دهید.

با انجام این کار پنجره‌ای باز می‌شود (شکل ۳-۲)، شما باید File type|Project را انتخاب نموده و دکمه OK را فشار دهید.



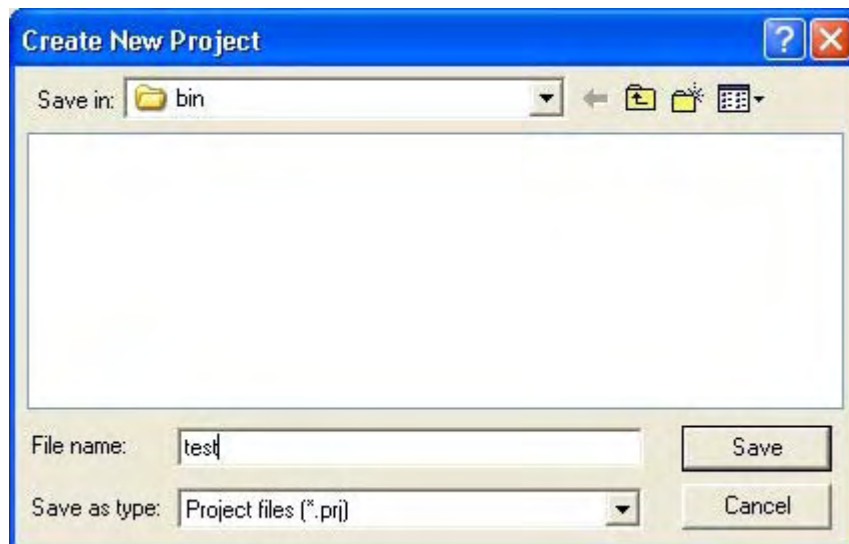
شکل ۳-۲ پنجره Create New File

حالا پنجره دیگری باز می‌شود (شکل ۳-۳) و از شما سؤال می‌گردد که آیا می‌خواهید پروژه جدید را به کمک CodewizardAVR انجام دهید.



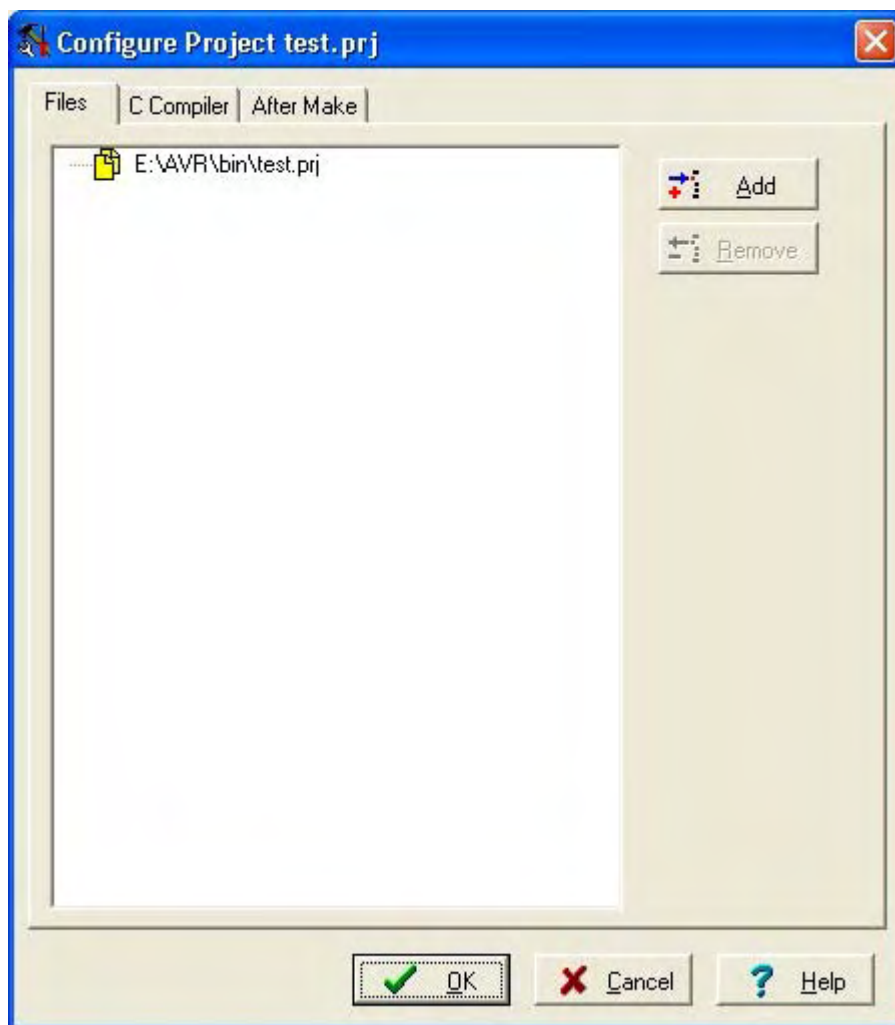
شکل ۳-۳ پنجره Confirm

در این قسمت دکمه No را انتخاب کنید. با این کار پنجره Create New Project باز خواهد شد (شکل ۳-۴) که باید در آن نام فایل پروژه و محل آن را مشخص نمایید. فایل پروژه با پسوند .ptj ذخیره خواهد شد.

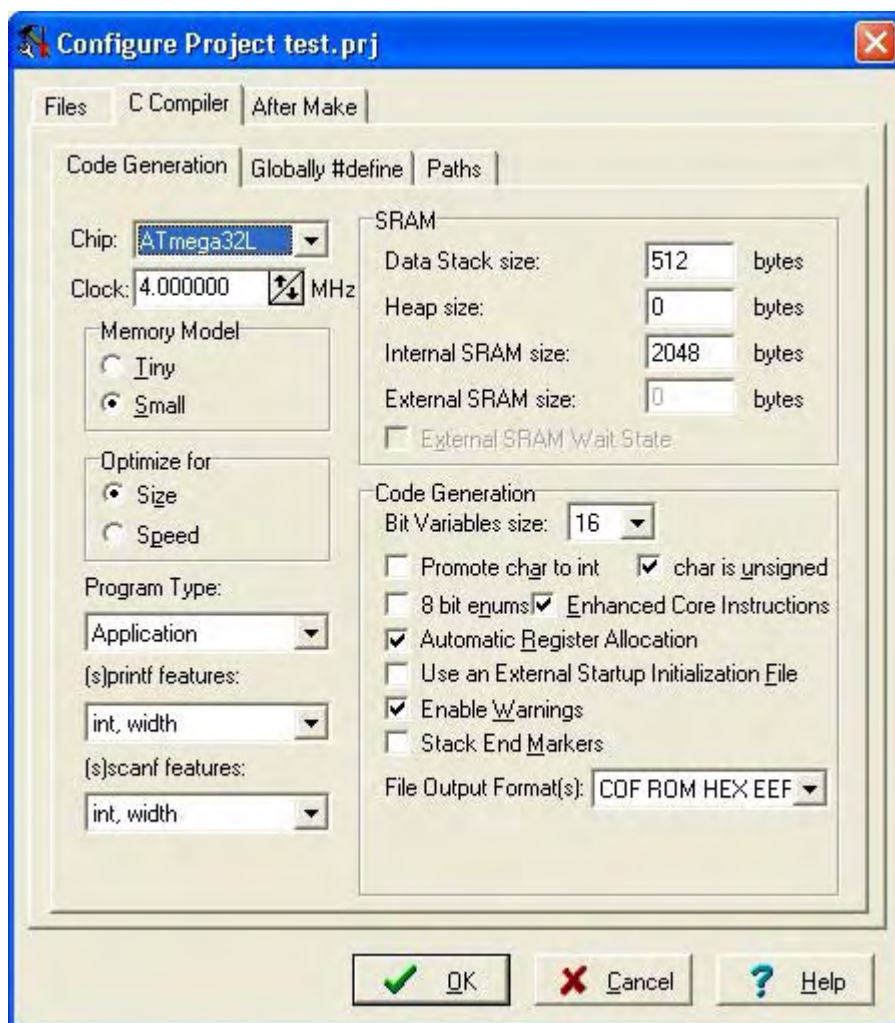


شکل ۳-۴ پنجره Create New Project

حالا پنجره جدیدی باز می شود که مربوط به تنظیمات کامپایلر است (شکل ۳-۵)، ابتدا در قسمت C Compiler|Code Generation، گزینه Chip، نوع تراشه AVR را برابر Atmega32L قرار داده، سپس در گزینه Clock مقدار کریستال متصل به تراشه را برابر 4MHz انتخاب نمایید و دکمه OK را فشار دهید. (شکل ۳-۶)



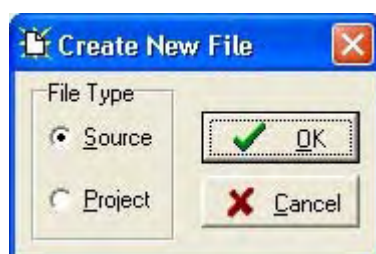
شکل ۳-۵ پنجره Configure Project قسمت Files



- پنجره Configure Project قسمت C Compiler

حالا باید فایل اصلی پروژه که کدهای برنامه در آن قرار دارد را ایجاد کنید. (شکل ۳-۷) برای این کار دوباره منوی

File|New و یا دکمه Create New File در نوار ابزار را انتخاب نموده و گزینه File Type|Source را انتخاب و دکمه OK را فشار دهید.



شکل ۳-۷ پنجره Create New File

برای فایل جدید ایجاد شده پنجره جدیدی ظاهر می‌گردد. نام فایل جدید untitled.c است. با استفاده از منوی

File، گزینه Save As نام فایل را به main.c تغییر دهید.

حالا باید کدهای برنامه زیر را در آن وارد نمایید. بعد از انجام این کار محتویات آن را ذخیره کنید.

```
#include <mega32.h>
#include <delay.h>
/*This is a test program written to flashes
the LED on PORTA.0 with a period of 0.5 second*/
main()
{
while(1){/*loop forever*/
PORTA.0=1;
delay_ms(250);
PORTA.0=0;
delay_ms(250);
}
}
```

دستور `#include <mega32.h>` فایل سرآمد `mega32.h` را که شامل تعریف رجیسترهای ATMega32L

است، به پروژه اضافه می‌کند. زمانی که بخواهیم در برنامه تاخیری ایجاد کنیم. ابتدا باید فایل `delay.h` را به کمک دستور

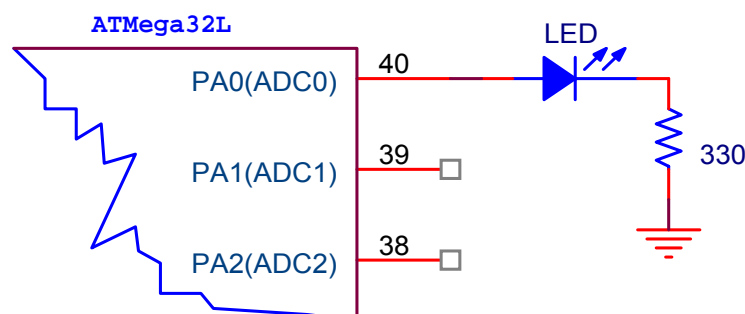
`#include <delay.h>` به پروژه اضافه نماییم. قسمت اصلی هر برنامه با `main()` آغاز می‌شود. دستور `while(1)` حلقه ای را

تعریف می‌کند که شرط آن همواره درست است و دستورات داخل آن مدام تکرار می‌شود. دستور `delay_ms(250)` نیز

تاخیری برابر ۲۵۰ میلی ثانیه تولید می‌کند.

در صورتی که سخت افزار موجود در فصل قبل را آماده کرده اید تنها کافی است المانهای زیر را به آن اضافه

کنید. (شکل ۳-۸)

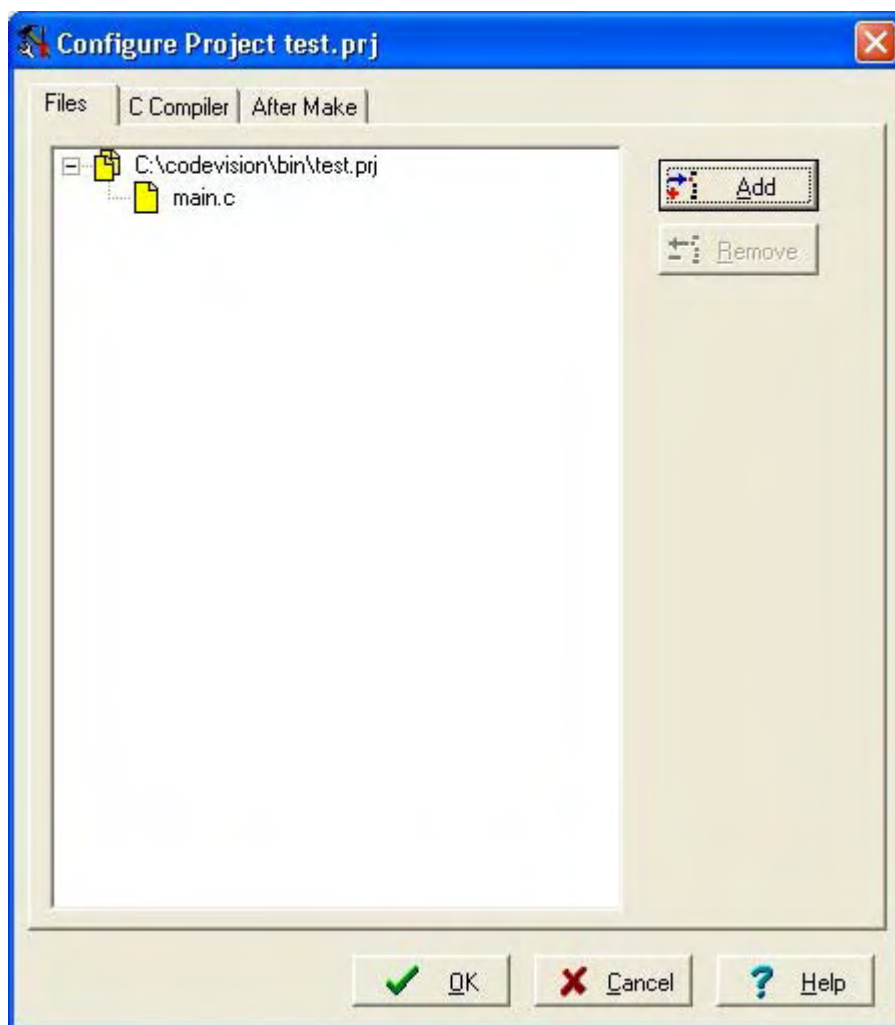


شکل ۳-۸ مدار مربوط برای روشن و خاموش کردن LED در PA0

حالا باید این فایل را به پروژه اضافه کنید. برای این کار از منوی Project، گزینه Configure را انتخاب کنید. در

قسمت Files، دکمه Add را فشار دهید و از پنجره باز شده فایل main.c که کدهای برنامه را قبلاً در آن وارد کرده‌اید،

انتخاب نموده و دکمه Open را فشار دهید. حالا فایل main.c به پروژه اضافه شده است. (شکل ۳-۹)



شکل ۳-۹ پنجره Configure Project بعد از اضافه کردن فایل main.c

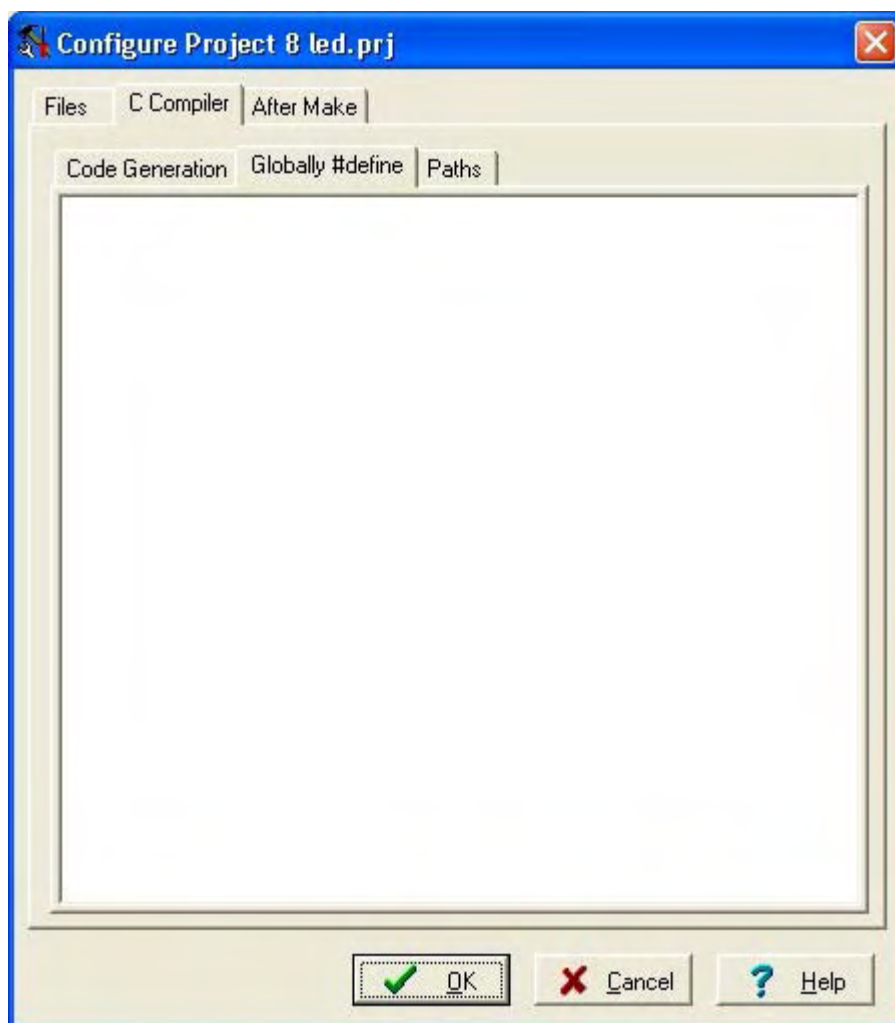


- قبل از اضافه کردن فایل به پروژه

- قبل از اضافه کردن فایل به پروژه

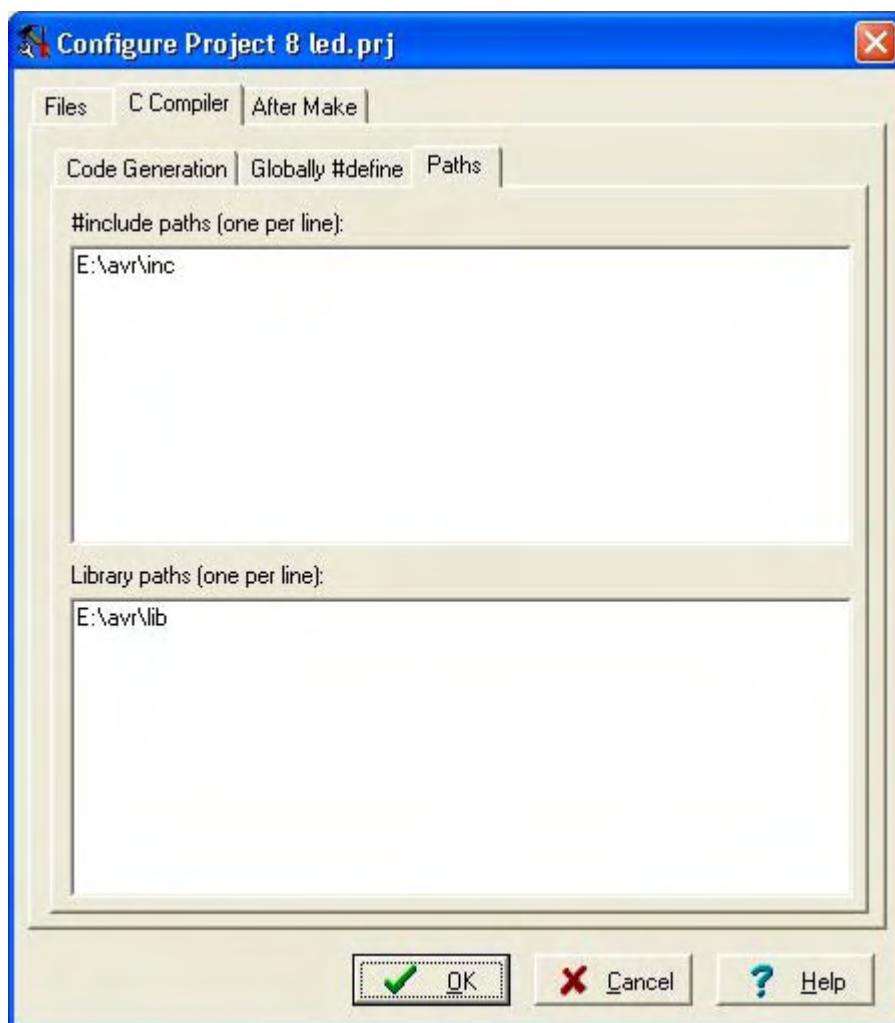
تنظیم کامپایلر

تنظیم پارامترهای کامپایلر C برای برنامه جاری توسط منوی Project، گزینه Configure، قسمت C Compiler انجام می‌شود. ابتدا در قسمت Code Generation، گزینه Chip، نوع تراشه AVR را برابر Atmega32L قرار داده، سپس در گزینه Clock مقدار کریستال متصل به تراشه را برابر 4MHz انتخاب کنید. لازم به ذکر است در صورتی که مقدار آن درست تنظیم نشود توابع تأخیر کار خود را درست انجام نخواهد داد. احتیاجی به تغییر بقیه مدار نیست. بد نیست بدانید در صورتی که پروژه شما از بیش از یک فایل تشکیل شده، می‌توانید تعاریف سراسری^۱ را در قسمت Globally#define وارد نمایید. (شکل ۳-۱۲)



شکل ۳-۱۲ پنجره Configure Project قسمت Globally#define

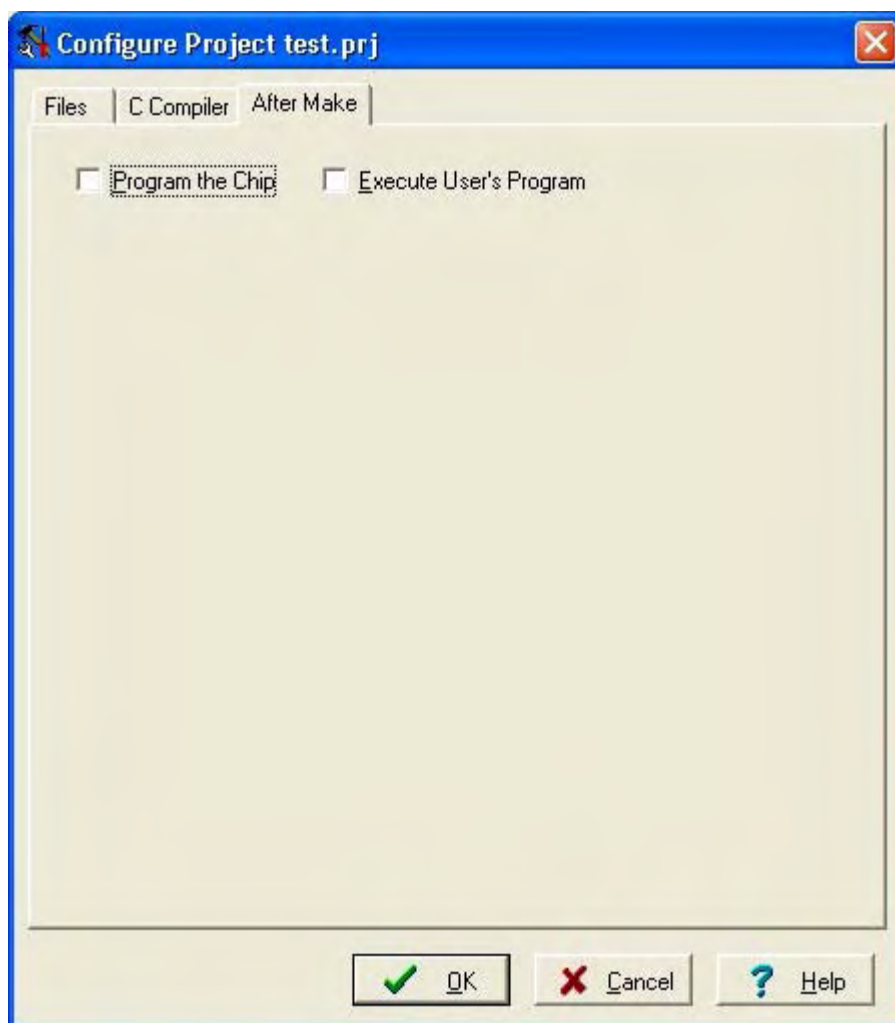
در قسمت paths هم می‌توانید مسیرهای جدیدی را برای فایل‌های سرآمد و کتابخانه‌ای تعریف کنید. (شکل ۳-۱۳)



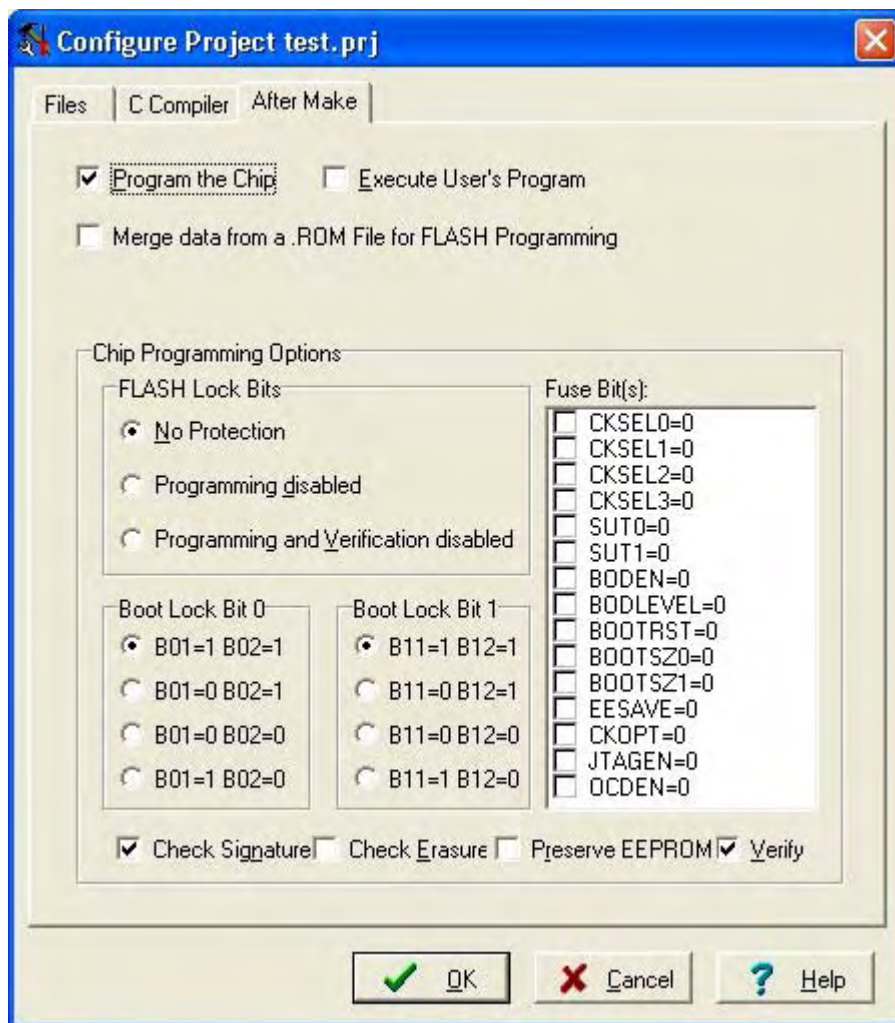
شکل ۳-۱۳ پنجره Configure Project قسمت Paths

در اینجا تنظیم کامپایلر هم به پایان رسیده است، ولی قبل از این که پنجره را ببندید قسمت After Make را

انتخاب کنید (شکل ۳-۱۴)



شکل ۳-۱۴ پنجره Configure Project قسمت After Make قبل از فعال کردن برنامه ریزی تراشه و در آن گزینه Program the Chip را علامت بزنید. این کار باعث می شود تا پس از کامپایل و اسمبل کردن برنامه بتوانید، مستقیماً کدها را به تراشه بفرستید و آن را برنامه ریزی کنید. (شکل ۳-۱۵) به کمک گزینه های دیگر هم می توانید حافظه ها را قفل نموده و فیوزهای موردنظران در روی تراشه را هم تنظیم کنید، در پایان با فشار دکمه OK پنجره را ببندید.



شکل ۳-۱۵ پنجره Configure Project قسمت After Make بعد از فعال کردن برنامه ریزی تراشه

تنظیمات نرم افزار

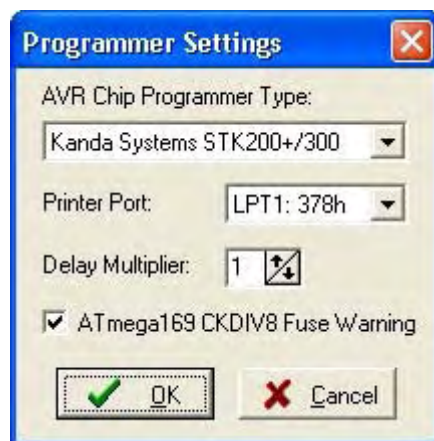
از آنجایی که این اولین برنامه شما با Codevision می باشد باید قسمت های مختلف آن را نیز به درستی تنظیم کنید. تنظیمات لازم شامل تعیین Debugger و آدرس آن، برای شبیه سازی برنامه و تنظیم نوع پروگرامر و آدرس پورت مربوط به آن می باشد.

برای تنظیم دیباگر از منوی Settings، گزینه Debugger را انتخاب کنید. در پنجره باز شده (شکل ۳-۱۶) در قسمت Debugger، Atmel AVR Studio4 را انتخاب کنید و در قسمت Directory and Filename آدرس نرم افزار AVR Studio که قبلاً نصب نموده اید را وارد کرده و دکمه OK را فشار دهید.



شکل ۳-۱۶ پنجره Debugger Settings

برای تنظیم پروگرامر، از منوی Settings، گزینه Programmer را انتخاب کنید. در این پنجره (شکل ۳-۱۷) می‌توانید نوع پروگرامر ISP و آدرس پورتی را که پروگرامر از طریق آن به کامپیوتر متصل شده است را مشخص نمایید. در قسمت AVR Chip Programmer Type، Kanda Systems STK200+/300 و در قسمت Printer Port، LPT1:378h، را مشخص کنید و دکمه OK را فشار دهید.



شکل ۳-۱۷ پنجره Programmer Settings

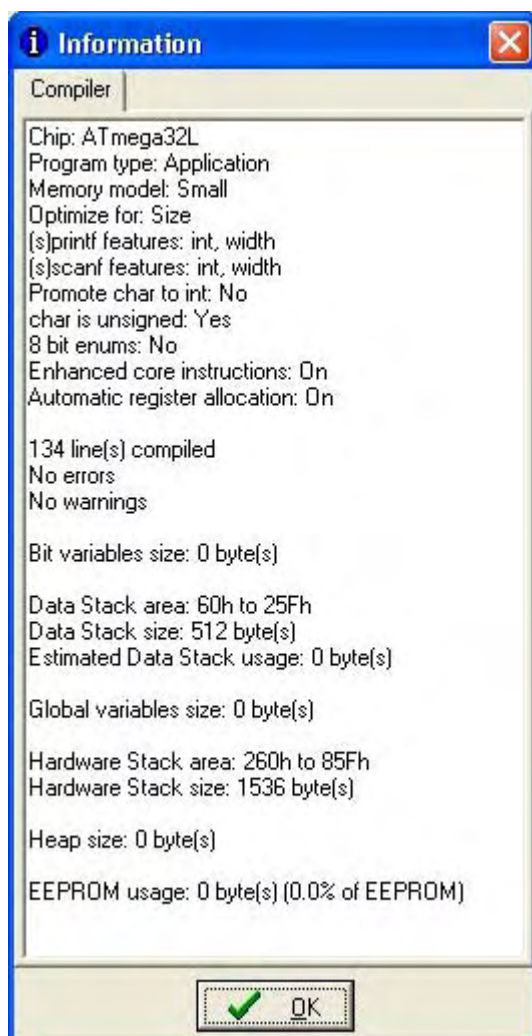
ایجاد برنامه قابل اجرا

برای قابل اجرا نمودن برنامه موارد زیر باید انجام گردد:

۱- کامپایل کردن فایل اصلی پروژه با پسوند c. و تولید فایل اصلی به زبان اسمبلی.

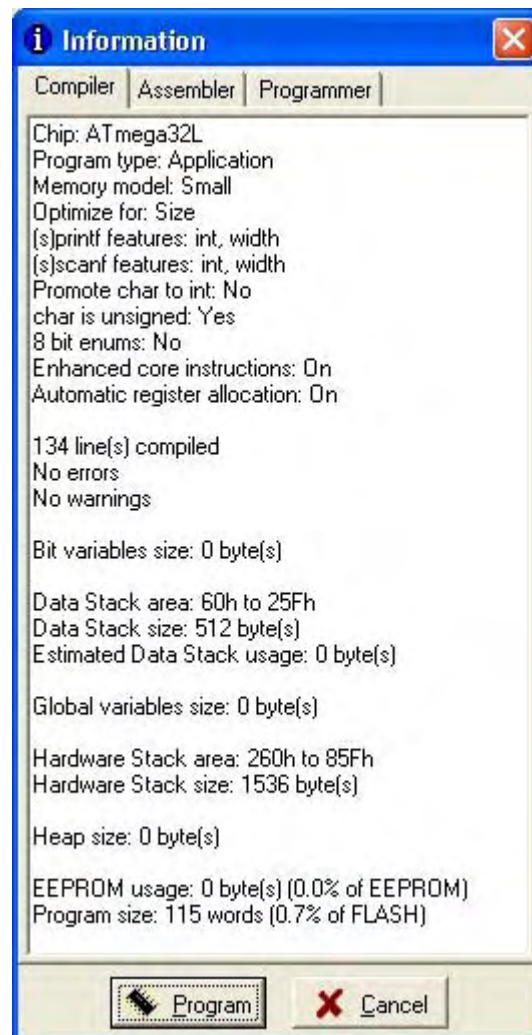
۲- اسمبل کردن فایل اسمبلی ایجاد شده به کمک اسمبلر AVRASM32.

برای انجام مورد ۱ می‌توانید از منوی Project گزینه Compile File را انتخاب نموده، کلید F9 را فشار داده و یا دکمه Compile در نوار ابزار را فشار دهید. بعد از کامپایل، فایل با پسوند .asm تولید می‌شود و پنجره Information که نتایج کامپایل را نشان می‌دهد ظاهر می‌شود. (شکل ۳-۱۸)

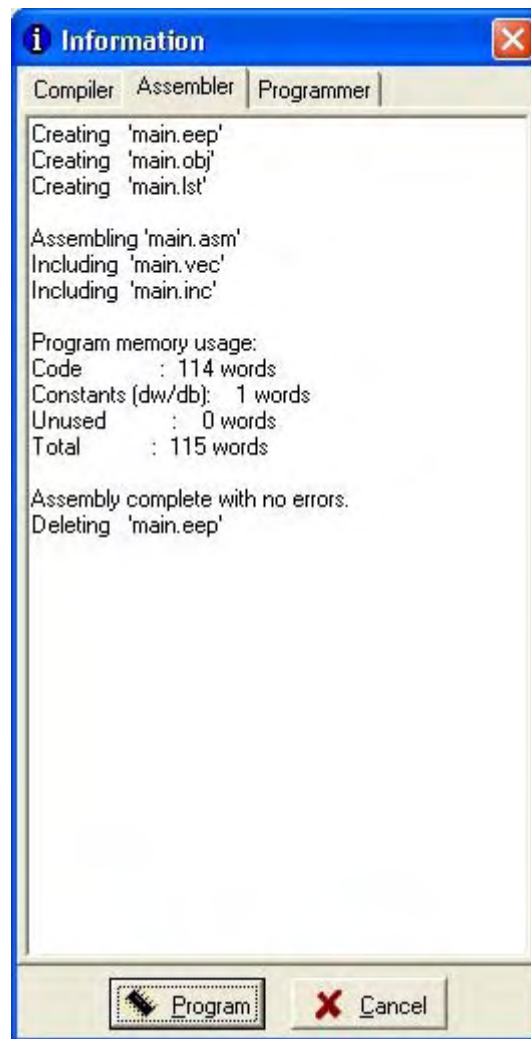


شکل ۳-۱۸ پنجره Information حاوی نتایج حاصل از کامپایل

در صورتی که بخواهیم موارد ۱ و ۲ را انجام دهیم باید از منوی Project، گزینه Make File را انتخاب نموده، کلید F9 را فشار داده و یا دکمه Make در نوار ابزار را فشار دهید. پس از اتمام این مرحله پنجره Information باز می‌شود که این بار از سه قسمت تشکیل شده است. در قسمت Compile (شکل ۳-۱۹)، تنظیمات مربوط به کامپایلر و میزان استفاده برنامه از حافظه‌های تراشه مشخص شده است. در قسمت Assembler (شکل ۳-۲۰) فایل‌های ایجاد شده و میزان حافظه کدها نشان داده شده و قسمت Programmer (شکل ۳-۲۱) تنها از یک شمارنده تشکیل شده که تعداد دفعات برنامه‌ریزی تراشه موردنظر را نشان می‌دهد که می‌توانید آن را صفر نمایید.



Compile Information -

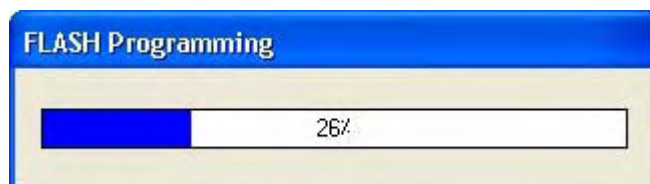


شکل ۳-۲۰ پنجره Information قسمت Assembler



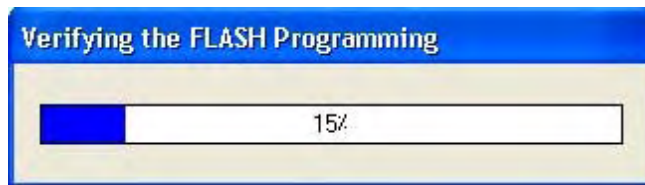
شکل ۳- ۲۱ پنجره Information قسمت Programmer

در صورتی که پروژه بدون ایجاد خطا کامپایل و اسمبل شده باشد، کابل ISP و منبع تغذیه آن هم وصل باشند، می‌توانید دکمه Program را فشار دهید. با انجام این کار برنامه‌ریزی تراشه به سرعت آغاز می‌گردد. ابتدا حافظه‌های تراشه برنامه‌ریزی می‌شود. (شکل ۳-۲۲)



شکل ۳- ۲۲ پنجره Flash Programming در زمان برنامه‌ریزی تراشه

و سپس صحت اطلاعات نوشته شده بررسی می‌گردد. (شکل ۳-۲۳)



شکل ۳-۲۳ پنجره Verify برای اطمینان از صحت کدهای نوشته شده

و اجرای کدها در تراشه نیز بلافاصله بعد از آن شروع می شود.

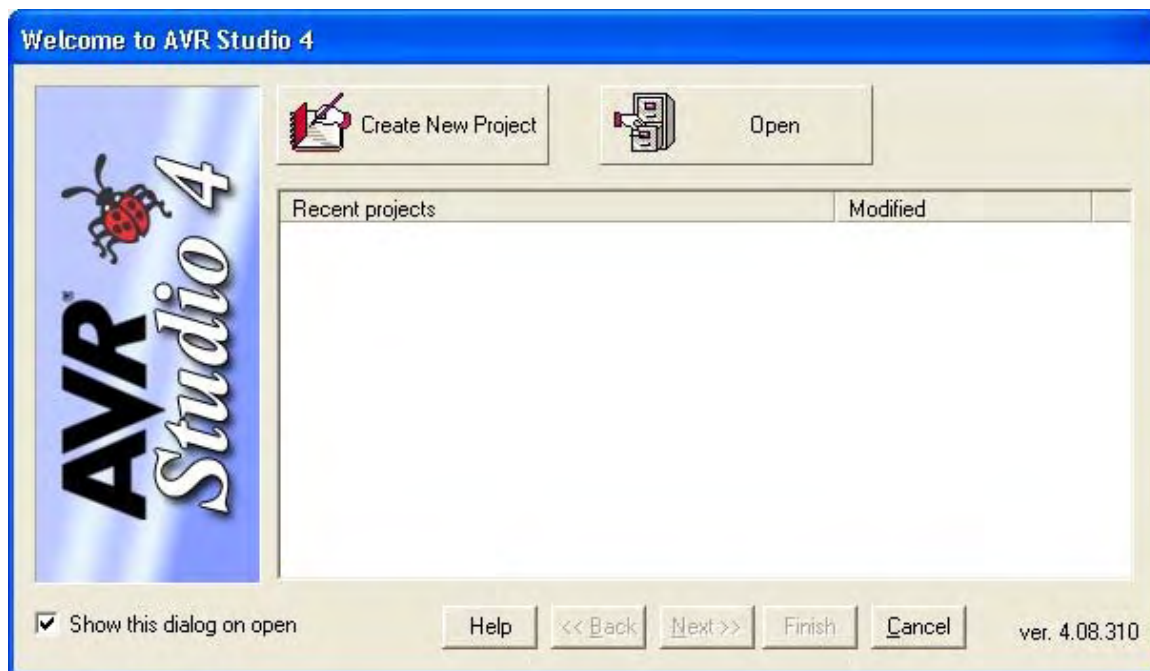
در صورتی که بخواهیم تا قبل از برنامه ریزی تراشه به کمک شبیه سازی برنامه از صحت عملکرد آن مطمئن شویم از نرم افزار AVRStudio استفاده می کنیم. در حقیقت پس از انجام اعمال کامپایل و اسمبل کردن برنامه Codevision یک فایل object با پسوند COFF تولید می کند. نسخه AVRStudio 4.06 یا نسخه های بعد از آن با استفاده از این فایل امکان شبیه سازی برنامه را برای ما فراهم می آورند.

برای اجرای این نرم افزار در منوی Tools , Debugger را انتخاب کنید و یا در نوار ابزار دکمه Run the debbuger را فشار دهید.



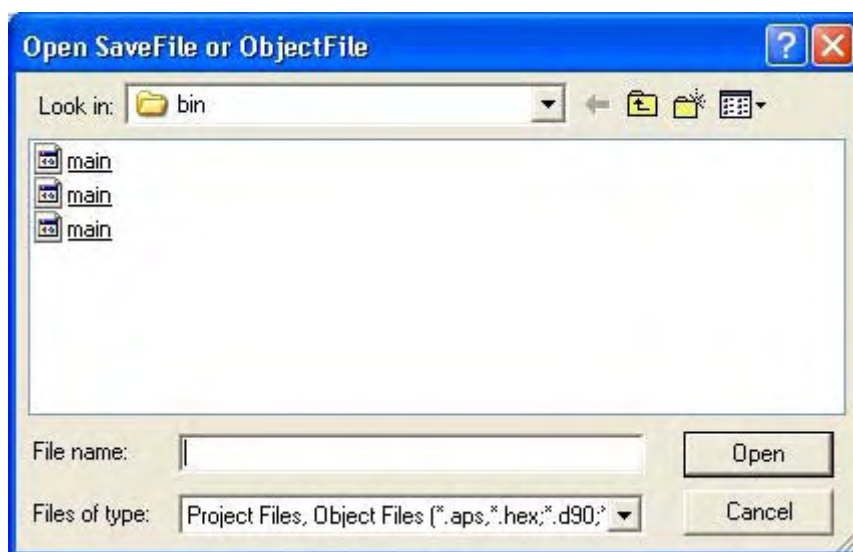
AVRStudio -

با اجرای نرم افزار AVRStudio , پنجره Welcome to AVR Studio 4 مطابق شکل (۳-۲۵) زیر باز می شود.



شکل ۳- ۲۵ پنجره Welcom toAVR Studio 4 برای باز کردن فایل object

در این پنجره بر روی دکمه Open کلیک نمایید. با انجام این کار پنجره دیگری باز می شود. (شکل ۳-۲۶)

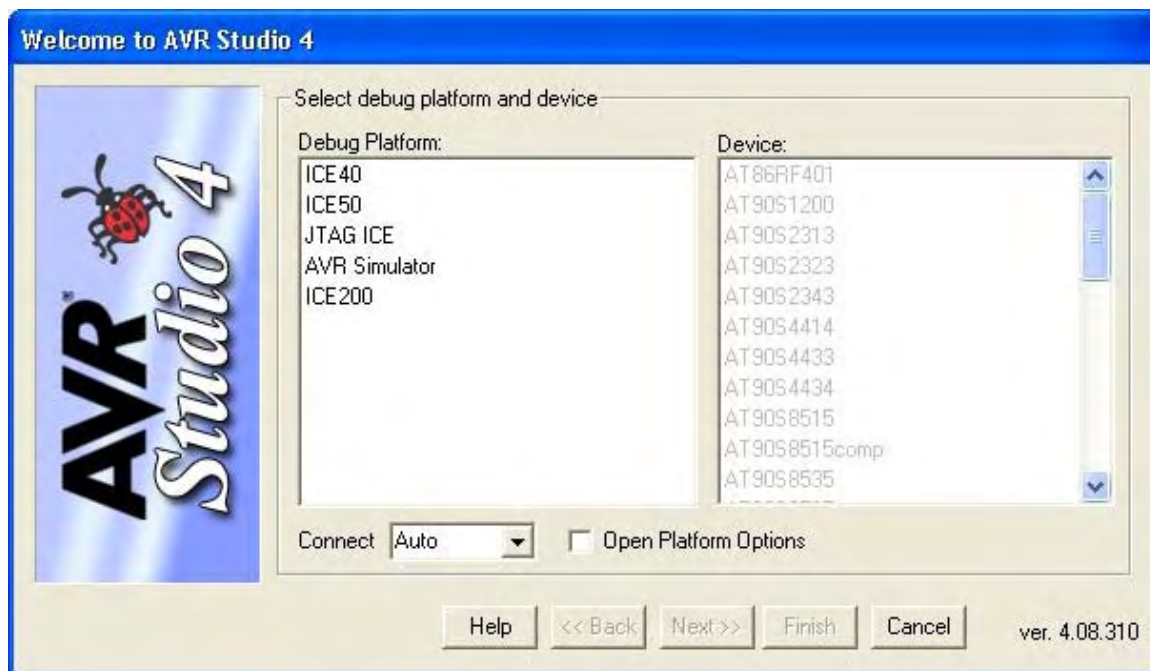


شکل ۳- ۲۶ پنجره Open ObjectFile

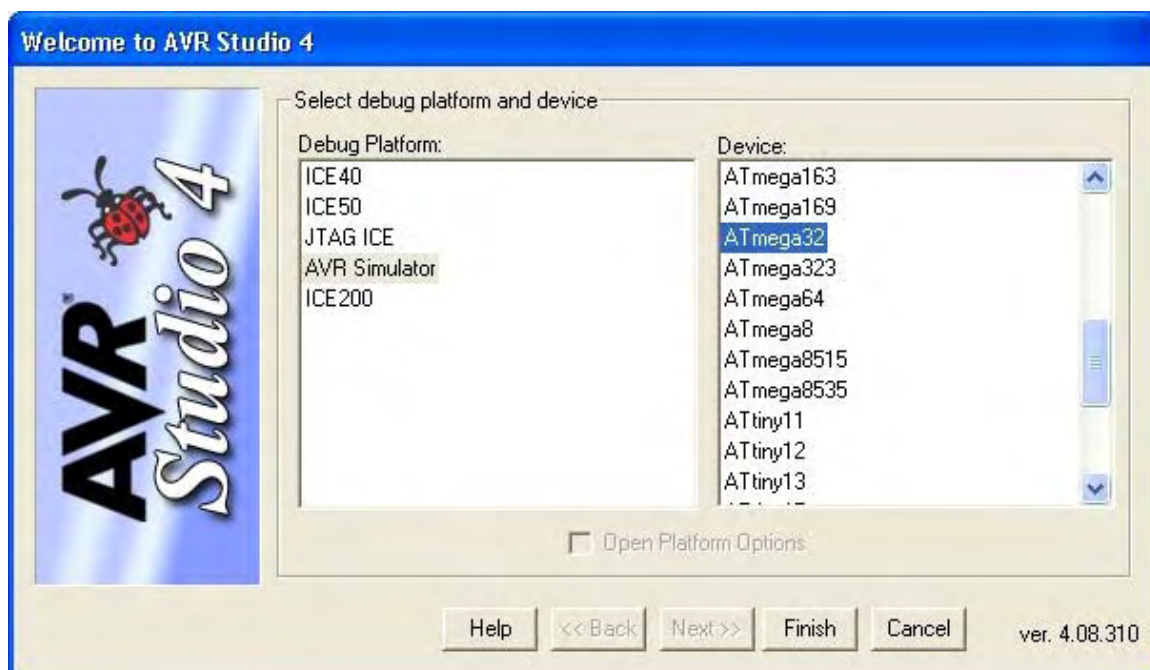
شما باید در این قسمت فایل با پسوند COFF که به آن اشاره شد را باز نمایید. بنابراین کافی است تا روی اولین فایل

کلیک کنید.

پس از انتخاب فایل COFF حالا وقت آن است که روش شبیه سازی و نوع تراشه را انتخاب کنیم.



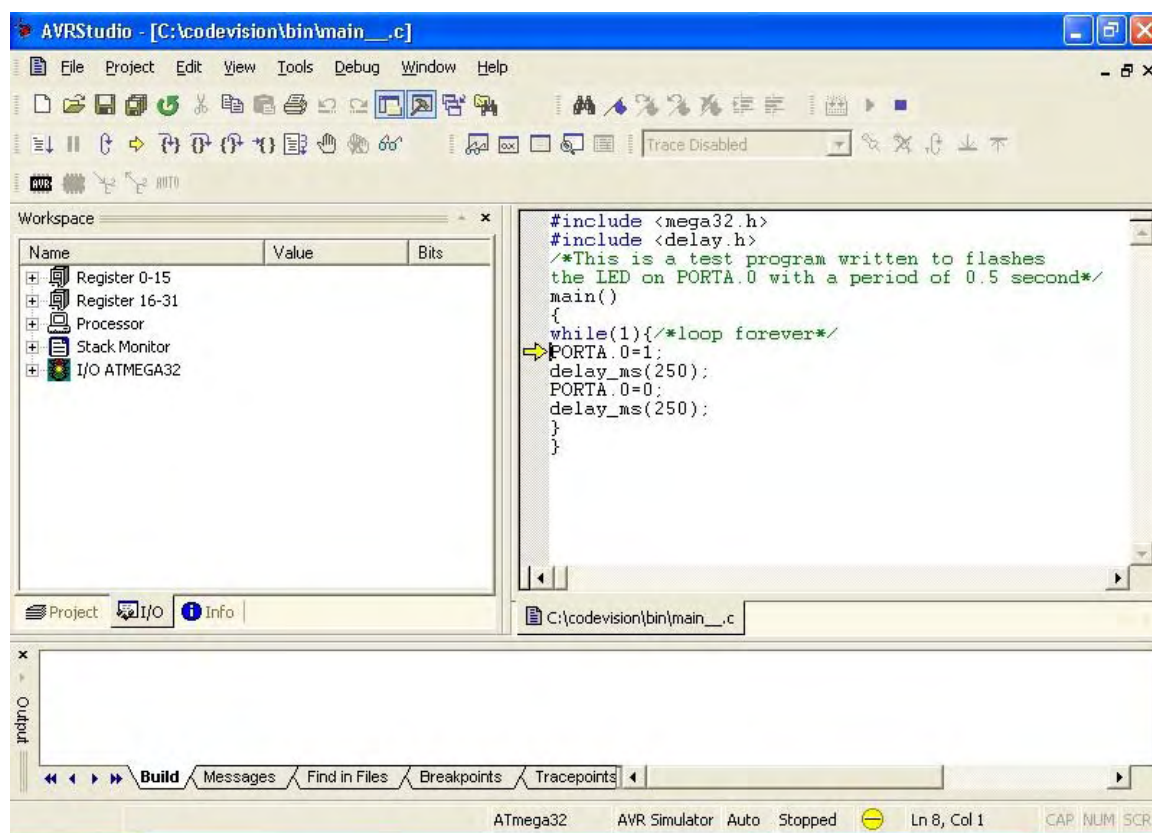
شکل ۳-۲۷ پنجره Welcom to AVR Studio 4 برای انتخاب روش شبیه سازی و نوع تراشه
 در (شکل ۳-۲۷) قسمت Debug Platform , AVR Simulator و در قسمت Device , Atmega32 را
 انتخاب نموده (شکل ۳-۲۸) و دکمه Finish را فشار دهید.



- پنجره Welcom to AVR Studio 4 بعد از انتخاب روش شبیه سازی و نوع تراشه

در این مرحله AVRStudio ، فایل COFF را باز می کند و برای انجام عمل شبیه سازی آماده می شود.(شکل

۲۹-۳)



شکل ۲۹-۳ محیط نرم افزار AVRStudio بعد از باز کردن فایل object

حالا می توانید با قرار دادن نقاط شکست (Break Points) در قسمتهای مورد نظر برنامه و مشاهده رجیسترهای

مختلف در پنجره Workspace و تغییر آنها از صحت برنامه خود مطمئن شوید که به دلیل سادگی این کار را به شما واگذار

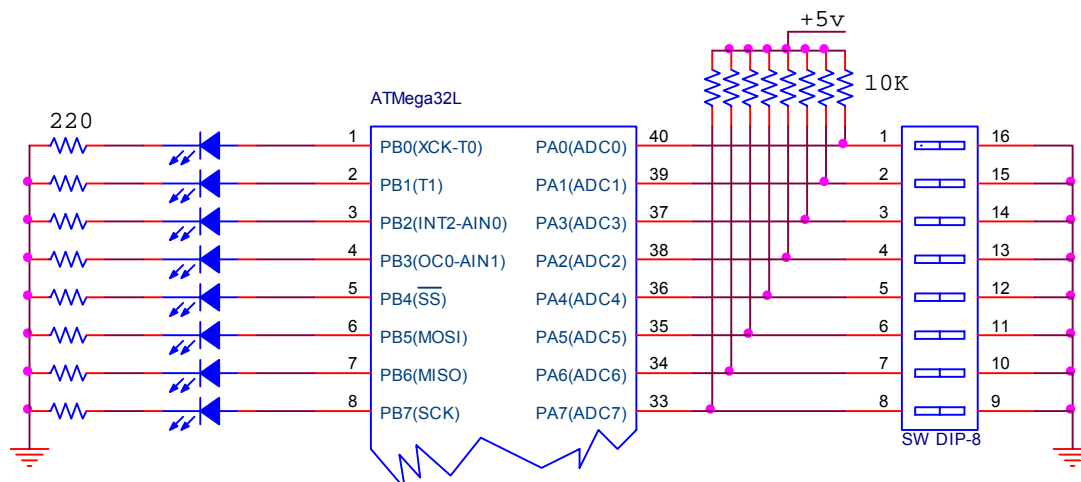
می کنیم.

فصل چهارم

آشنایی با Codewizard AVR

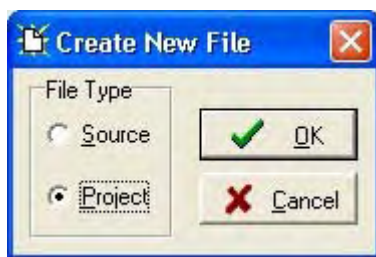
به دلیل تنوع امکانات و وسایل جانبی AVR، تنظیمات مربوط به آنها ممکن است کار بسیار مشکلی به نظر برسد. نرم افزار Codewizard این امکان را به ما می دهد تا در ابتدای برنامه تنظیمات اولیه قسمتهای مختلف را به صورت گرافیکی انجام دهیم . پس از آن، این نرم افزار به صورت خود کار کدهای مربوط را برای ما ایجاد می کند . به این ترتیب علاوه بر صرفه جویی در وقت، کار با AVR ها نیز ساده تر می شود .

برای این که در عمل با این نرم افزار بیشتر آشنا شوید . کار را با یک مثال ادامه می دهیم . می خواهیم برنامه ای بنویسیم که موقعیت کلیدها، در پورت A را خوانده و LED های متناظر در پورت B را تنظیم نماید.(شکل ۴-۱)



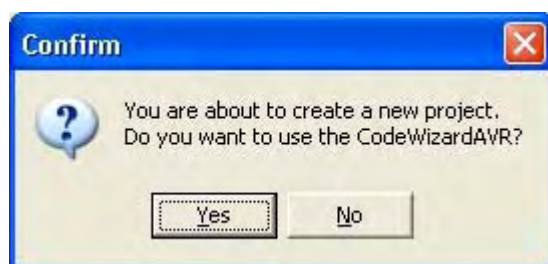
شکل ۴-۱ چگونگی اتصال هشت کلید به پورت A و هشت LED به پورت B

برای ایجاد یک پروژه جدید همانند قبل از منوی File قسمت New را کلیک کرده و در پنجره Create.Project New File را انتخاب کرده و دکمه OK را فشار دهید.



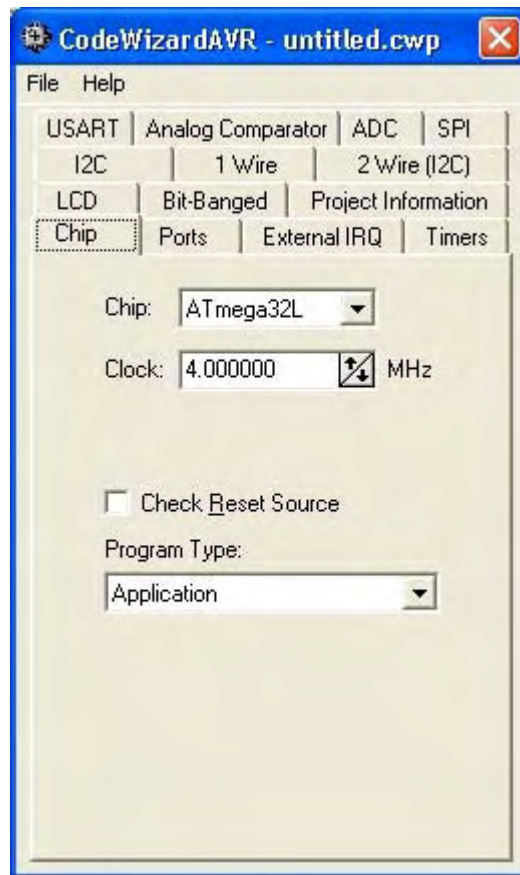
شکل ۴-۲ پنجره Create New File

از آنجایی که قصد داریم تا پروژه را به کمک CodewizardAVR انجام دهیم، در پنجره Confirm دکمه Yes را فشار دهید. (شکل ۴-۳) تا نرم افزار CodewizardAVR اجرا گردد.



-

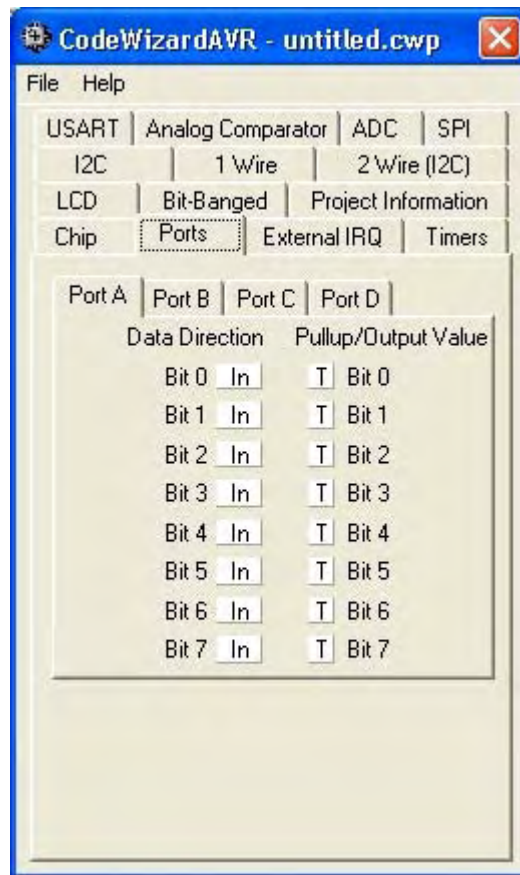
قبل از هر چیز در قسمت Chip ، نوع تراشه (Atmega32L) و فرکانس کریستال مورد استفاده (4 MHz) را تنظیم نمایید. (شکل ۴-۴)



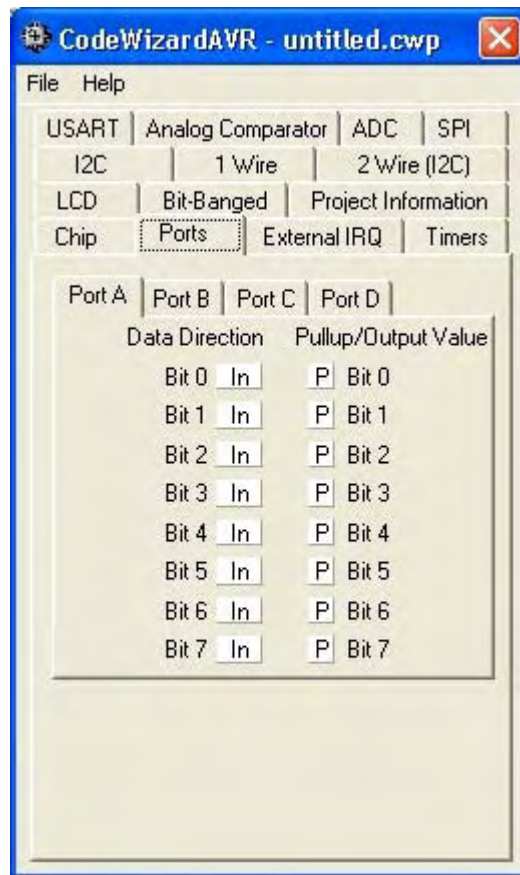
شکل ۴-۴ پنجره CodewizardAVR قسمت Chip

حالا می توانیم به کمک قسمت Ports، وضعیت اولیه تمامی پورتها به صورت ورودی – خروجی و مقدار اولیه آنها را تنظیم نماییم.

ابتدا باید پورت A را به صورت ورودی در آوریم . همان طور که در قسمت Port A می بینید تمام بیتها ورودی اند . در جلوی هر یک از بیتها حرف T نوشته شده است.(شکل ۴-۵) باید بر روی آنها کلیک کنید، تا به حرف P (Pull up) تبدیل شوند.(شکل ۴-۶)

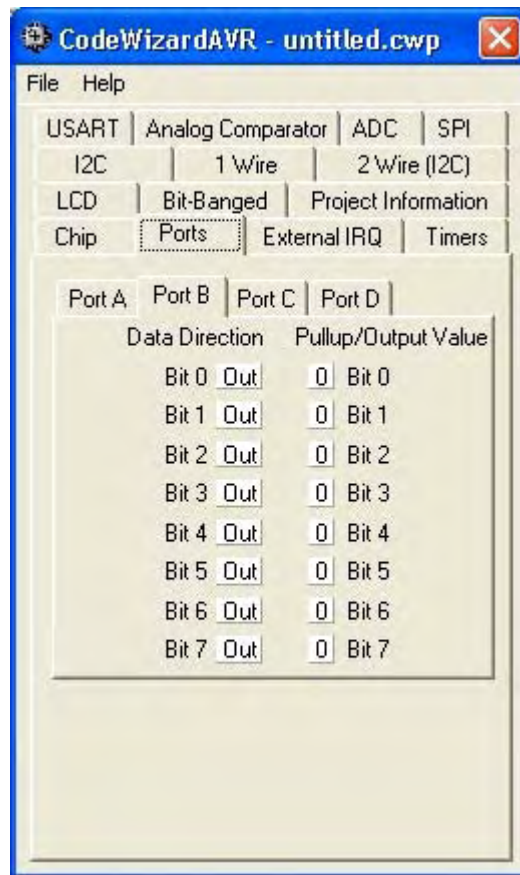


شکل ۴-۵ پنجره Codewizard، پورت A در حالت پیش فرض



شکل ۴-۶ پنجره Codewizard، پورت A بعد از تنظیم

حالا باید پورت B را به صورت خروجی درآوریم. با انتخاب قسمت Port B و کلیک بر روی حرف In، در مقابل هر یک از بیتها این عبارت به Out تبدیل می شود. ضمناً می توانید مقدار اولیه پورت را به صورت 0 و 1 تنظیم کنید. (شکل ۴-۷)



- پنجره Codewizard، تنظیمات پورت B

در اینجا تنظیمات اولیه به پایان رسیده است و باید کدهای مربوطه تولید شوند. برای این کار از منوی File قسمت

Generate, Save and Exit را انتخاب می کنید. (شکل ۴-۸)



شکل ۴-۸ منوی File در برنامه CodewizardAVR

با انجام این عمل پنجره ای باز می شود و از شما می خواهد تا فایل اصلی پروژه خود را نام گذاری کرده و آن را ذخیره

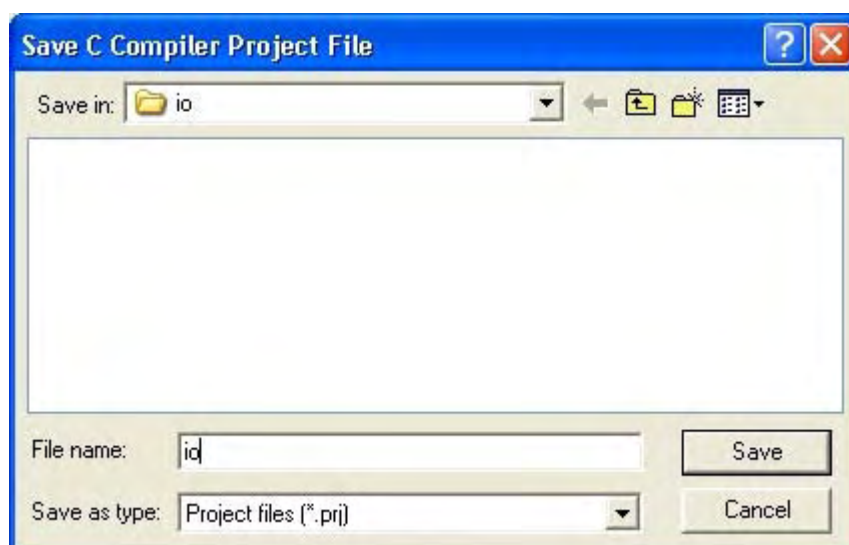
کنید. نام آن را io انتخاب کنید دکمه Save را فشار دهید. (شکل ۴-۹)



شکل ۴-۹ ذخیره کردن فایل اصلی پروژه به زبان C

حالا پنجره دیگری باز می شود که فایل پروژه را ذخیره خواهد کرد نام آن را هم io انتخاب کنید و دکمه Save را

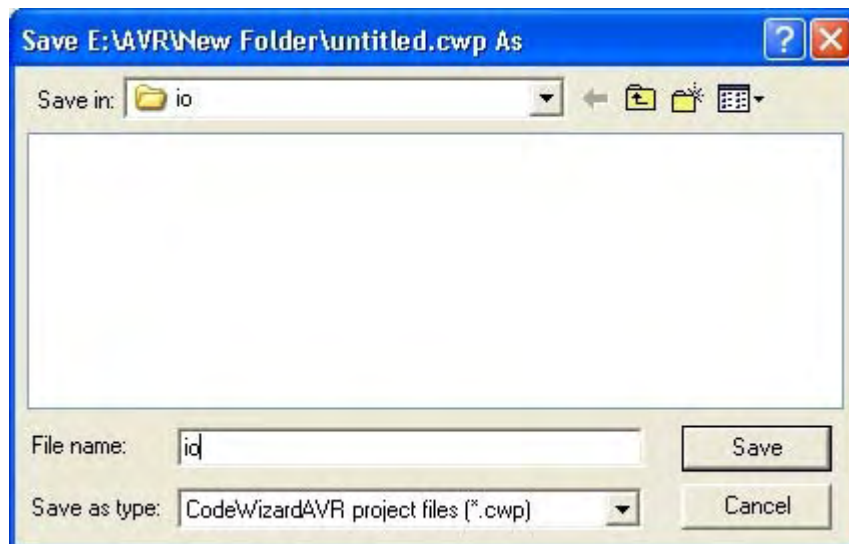
فشار دهید(شکل ۴-۱۰)



شکل ۴-۱۰ ذخیره کردن فایل پروژه

پنجره بعدی فایل پروژه مربوط به CodewizardAVR را ذخیره می کند. نام آن را نیز io قرار دهید و آن را ذخیره

کنید.(شکل ۴-۱۱)



شکل ۴- ۱۱ ذخیره کردن فایل پروژه مربوط به CodewizardAVR

با انجام این کار CodewizardAVR کدهای برنامه را تولید کرده و آنها را در فایل اصلی پروژه قرار می دهد و آن را باز می نماید.

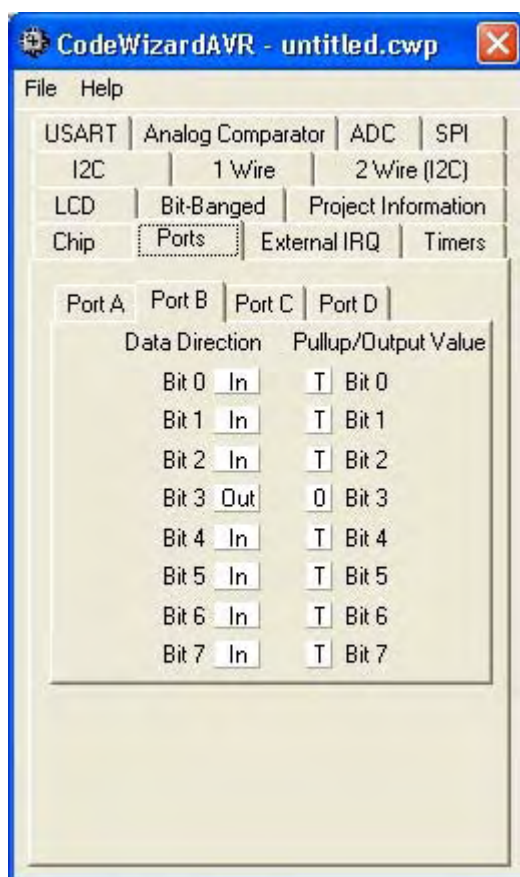
همان طور که می بینید علاوه بر کدها، در هر قسمت توضیحاتی هم اضافه شده اند که باعث ساده تر شدن فهم برنامه می شوند . حالا کافی است تا کدهای خود را بعد از قسمت `//Place your code here` اضافه کنید . از آنجایی که این برنامه ساده است کافی است تا در این قسمت دستور زیر را اضافه نمایید .

```
PORTB = PINA;
```

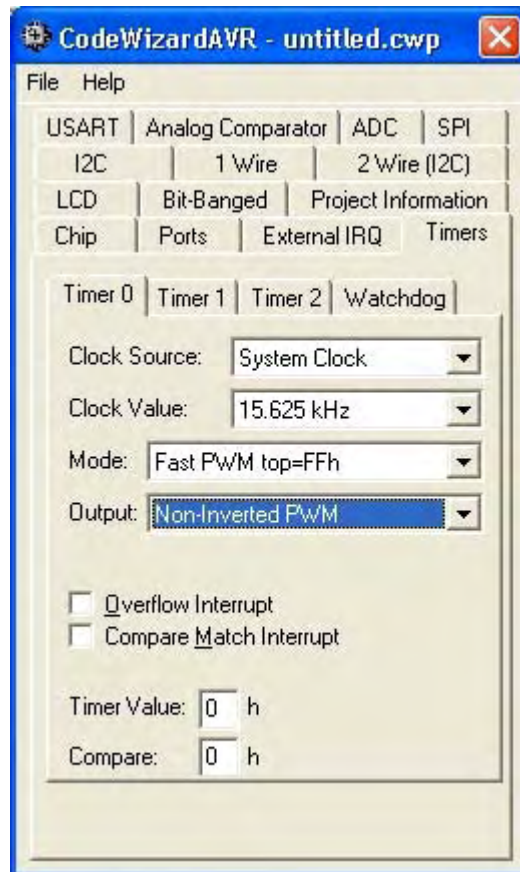
حالا فایل را ذخیره و کامپایل نموده و تراشه را پروگرام کنید و نتیجه را ببینید.

PWM

می خواهیم با استفاده از تایمر صفر در مد Fast mode PWM یک موج PWM تولید کنیم . پهنای این موج ۲۵۵ است و ما می خواهیم تا طول قسمت موثر به صورت متناوب از ۰ تا ۱۰۰ و از ۱۰۰ تا ۰ با تاخیر ۲۰ میلی ثانیه بین هر دو عدد متوالی تغییر کند. در این صورت اگر موج PWM را به یک LED بدهیم نور آن به صورت تدریجی کم و زیاد می شود و برای نوشتن این برنامه همانند مثال قبل یک پروژه ایجاد کرده و نرم افزار CodewizardAVR را اجرا کنید پس از تنظیم نوع تراشه و فرکانس کریستال مورد استفاده (همانند قبل)، در قسمت Ports , بیت چهارم از پورت B را به صورت خروجی در آورید.(شکل ۴-۱۲)



این بیت همان OC0¹ است که برای تولید PWM توسط تایمر صفر استفاده می شود. سپس در قسمت Timers، بخش Timer 0، قسمت clock source، System clock را انتخاب کنید و بقیه قسمت‌ها را هم مطابق شکل (۴-۱۳) تنظیم نمایید.



شکل ۴-۱۳ تنظیم تایمر صفر برای تولید PWM

در این تنظیمات CodewizardAVR به پایان می رسد و باید آن را به صورتی که در مثال قبل گفته شد به کد تبدیل نمایید.

در این قسمت باید ابتدا بعد از دستور `#include <mega32.h>`، دستور `#include <delay.h>` را اضافه کنید سپس بعد از خط توضیح `//place your code hear` دستورات زیر را وارد نمایید.

```
unsigned char i;
for(i=0;i<100;i++)
{
    OCR0=i;
```

Output Compare 0

```

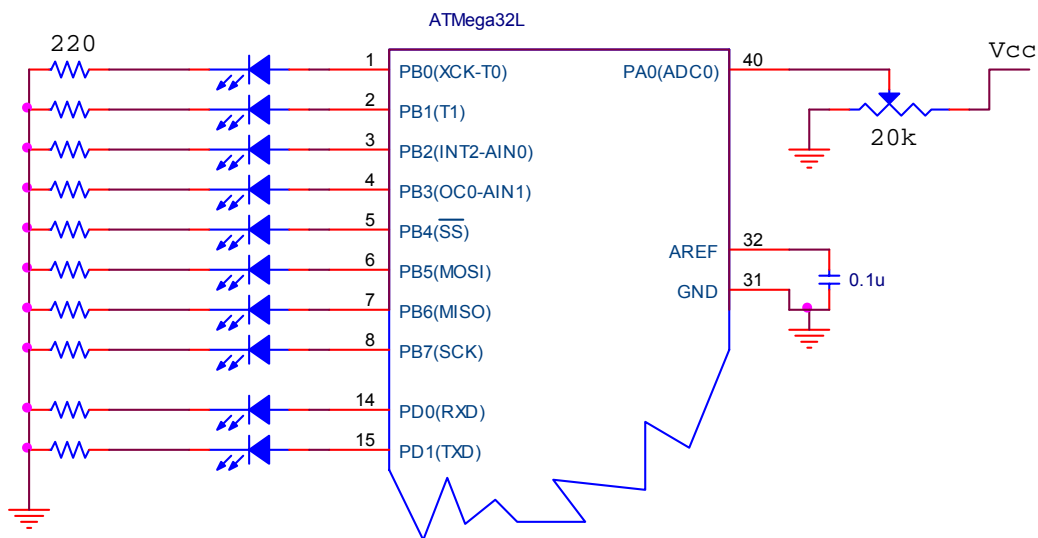
        delay_ms(20);
    }
    for(i=100;i>1;i--)
    {
        OCR0=i;
        delay_ms(20);
    }
}

```

حالا برنامه را کامپایل نموده و تراشه را پروگرام کنید . موج PWM در خروجی OC0 را می توانید به کمک یک اسیلوسکوپ حافظه دار هم مشاهده نمایید .

ADC

در این قسمت قصد داریم، برنامه ای بنویسیم تا ورودی آنالوگ در پایه (ADC0) PORTA را خوانده و آن را به صورت ۱۶ بیتی در پورت B و دو بیت کم ارزش پورت D به کمک LED نشان دهد .

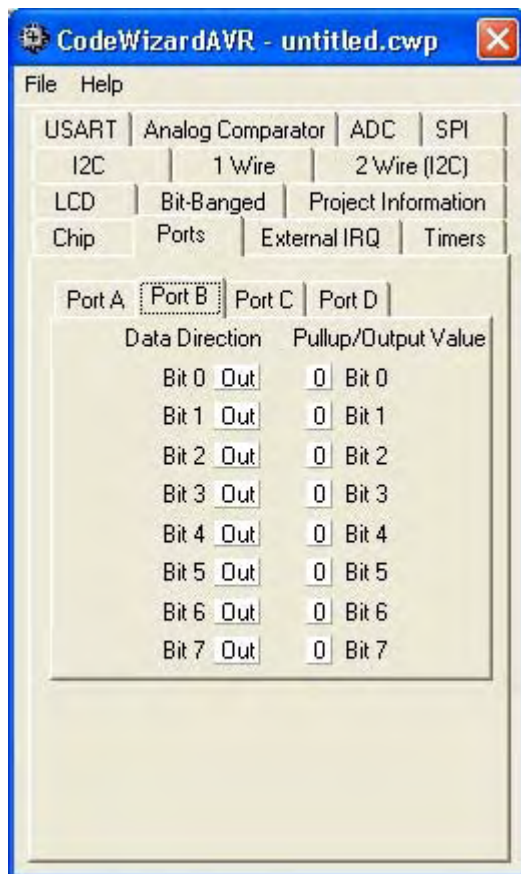


سخت افزاری که برای اجرای برنامه باید به سخت افزار اصلی اضافه گردد به صورت زیر است: یک خازن سرامیکی ۰٫۱ میکروفاراد بین پایه AREF و زمین، یک پتانسیومتر ۲۰ کیلو اهم در ADC0 و ۱۰ عدد LED و مقاومت ۲۲۰ اهم که در شکل (۴-۱۴) زیر نشان داده شده اند

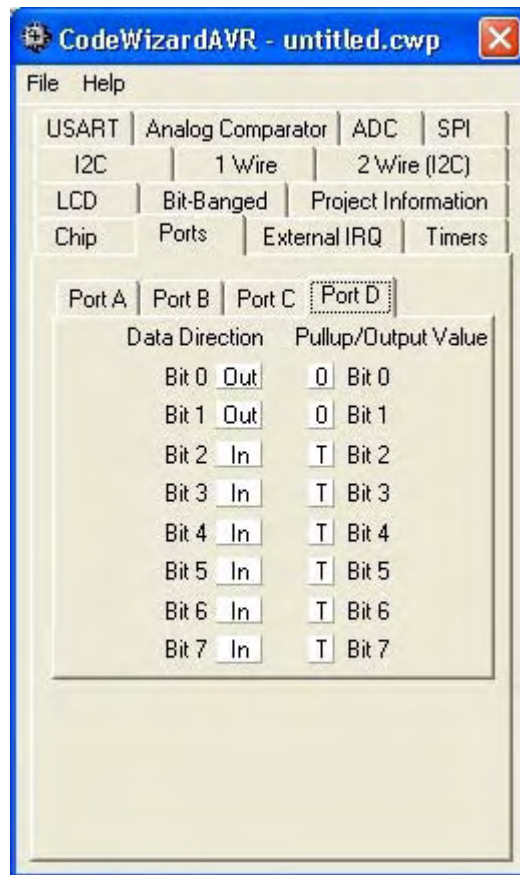
در ابتدا باید یک پروژه جدید ایجاد کنید و همانند مثالهای قبل نرم افزار CodewizardAVR را اجرا نمایید . بعد

از تنظیم نوع تراشه و فرکانس کریستال، در قسمت Ports ، پورت B و دو بیت کم ارزش پورت D را به صورت خروجی در

آورید. (شکلهای ۴-۱۵ و ۴-۱۶)



شکل ۴-۱۵ تنظیم پورت B به صورت خروجی

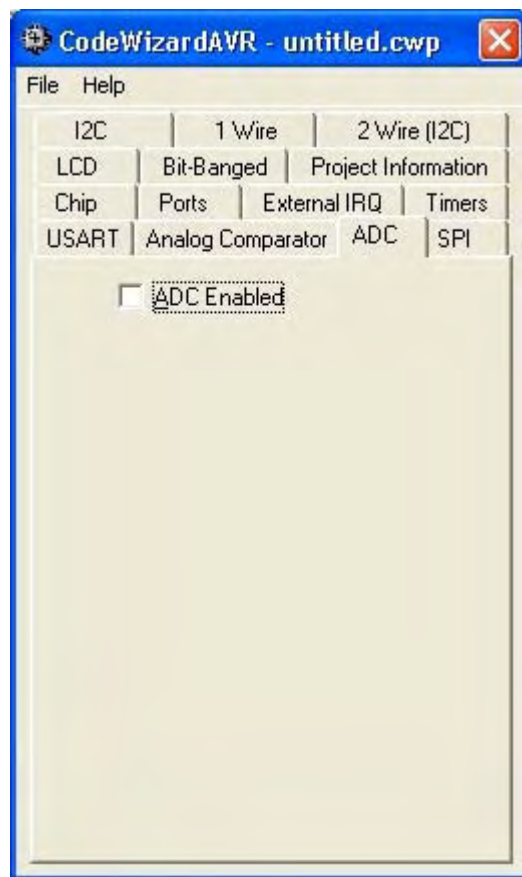


شکل ۴-۱۶ تنظیم پینهای صفر و یک از پورت D به عنوان خروجی

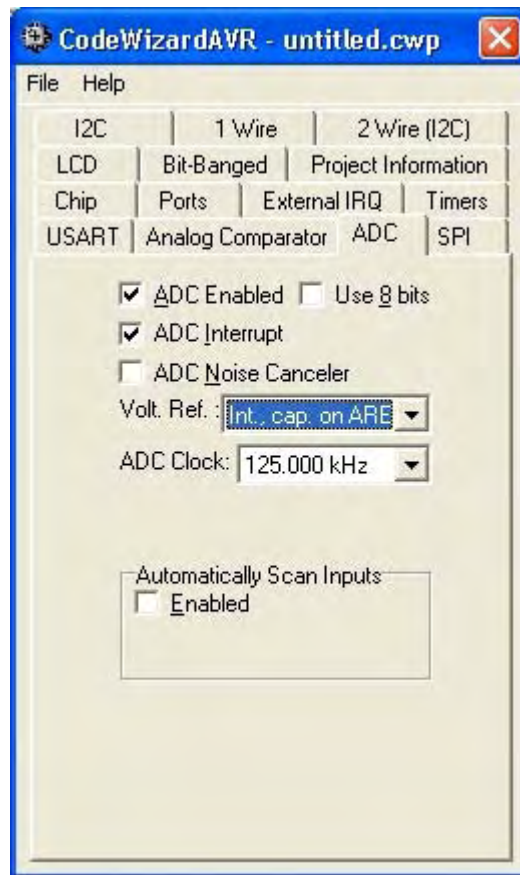
سپس در قسمت ADC، (شکل ۴-۱۷) ابتدا ADC Enabled را علامت بزنید. با انجام این کار تنظیمات مربوط به

ADC نمایان می شوند. (شکل ۴-۱۸) سپس Interrupt ADC را نیز علامت زده، Volt. Ref را برابر Int, Cap.on

و AREF و ADC Clock را برابر 125.000KHz انتخاب کنید.



شکل ۴-۱۷ قسمت ADC قبل از فعال کردن آن



شکل ۴-۱۸ قسمت ADC و تنظیمات آن

در این موقع کل تنظیمات اولیه به پایان رسیده است و باید کدهای برنامه را تولید نمایید .

حالا باید دستوراتی را به فایل تولید شده اضافه کنید . ابتدا در تابع interrupt بعد از قسمت // place your code

hear دستورات زیر را وارد کنید .

```
PORTB=ADCL;
PORTD.0=ADCH.0;
PORTD.1=ADCH.1;
ADCSRA|=0x40;
```

سپس بعد از دستور #asm("sei") ، دستور ADCSRA|= 0x40 را بنویسید . حالا می توانید ، پروژه را ذخیره

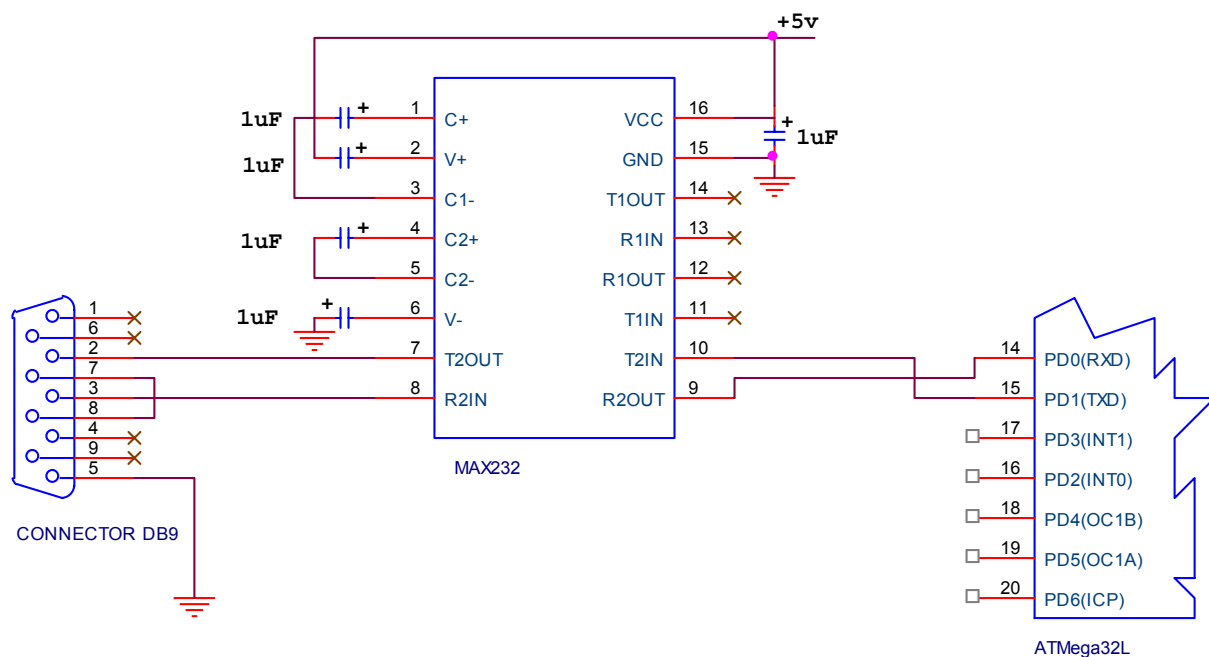
و کامپایل کرده و تراشه را پروگرام کنید.

فصل پنجم

ارتباط سریال و LCD

ارتباط سریال با کامپیوتر

در این قسمت می‌خواهیم برنامه‌ای بنویسیم که اطلاعات را به صورت سریال از تراشه به کامپیوتر بفرستد. می‌خواهیم عبارت "Hello World" با نرخ ارسال 9600 bit/sec، بدون parity، با یک بیت توقف و هشت بیت دیتا فرستاده شود. ترجیحاً بهتر است، ابتدا در مورد سخت‌افزار لازم برای برقراری ارتباط سریال صحبت کنیم. مدار آن در شکل (۵-۱) نشان داده شده است.

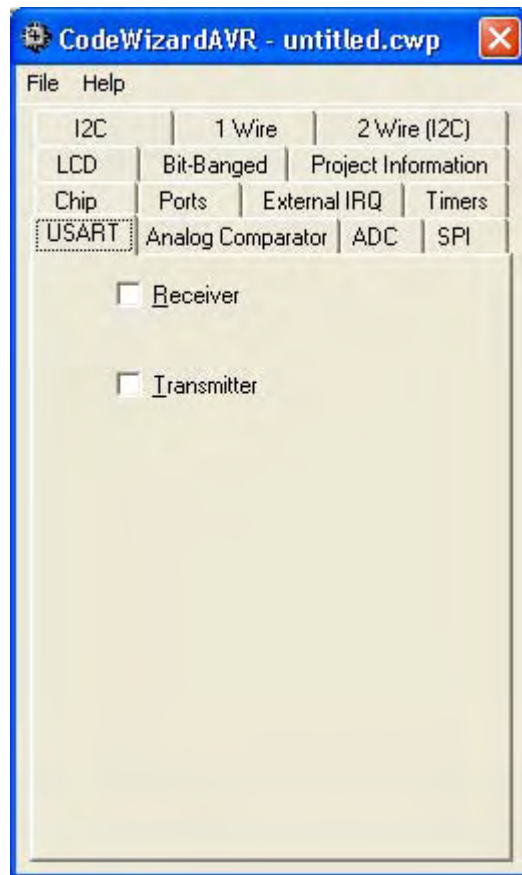


شکل ۵-۱ برقراری ارتباط سریال

حالا باید به سراغ نوشتن برنامه برویم. ابتدا پروژه جاری را با انتخاب گزینه Close Project از منوی File ببندید.

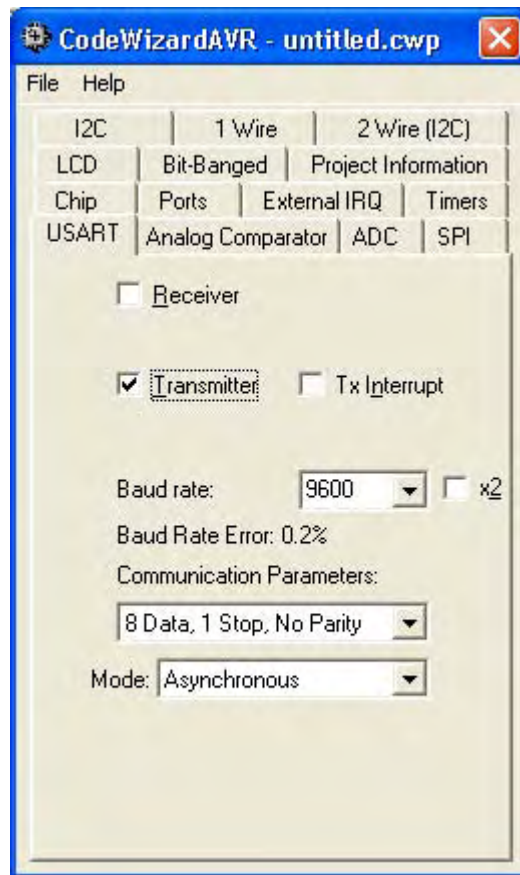
حالا باید پروژه جدیدی ایجاد کنید. برای انجام این کار از منوی Tools, CodewizardAVR را انتخاب کنید. همانند قبل

در اولین مرحله نوع تراشه و فرکانس کریستال مورد استفاده را مشخص کرده، سپس قسمت USART را انتخاب کنید.(شکل



شکل ۵-۲ پنجره CodewizardAVR قسمت USART

از آنجایی که قصد داریم در این برنامه از تراشه به عنوان فرستنده استفاده کنیم، گزینه Transmitter را علامت بزنید. با انجام این کار موارد دیگری به صفحه اضافه می‌شوند، کافی است تا Baudrate را برابر 9600، Communication Parameters را برابر Mode 8 Data, 1 Stop, No Parity و Mode را برابر Asynchronous انتخاب نمایید. (شکل ۵-۳)



شکل ۵-۳ تنظیم UART به صورت فرستنده

در این جا تنظیمات اولیه برنامه به پایان رسیده است و باید پروژه ایجاد و کدهای موردنظر تولید شوند.

حالا باید دستورات زیر را به این فایل اصلی پروژه اضافه کنید. بعد از عبارت `///Declar your local Variables`

here دستورات زیر را اضافه کنید:

```
char message[]="Hello World!";
```

```
int i;
```

این دو دستور، دو متغیر را تعریف می کنند، اولی متغیر `message` که یک آرایه رشته ای است و همان پیغامی است

که می خواهیم به کامپیوتر بفرستیم و متغیر دوم، `i`، یک شمارنده است. حالا باید قبل از حلقه `while(1)`، دستورات زیر را

اضافه کنید. این دستورات پیغام "Hello World" را به کامپیوتر می فرستند.

```
for (i=0;i<12;i++)
```

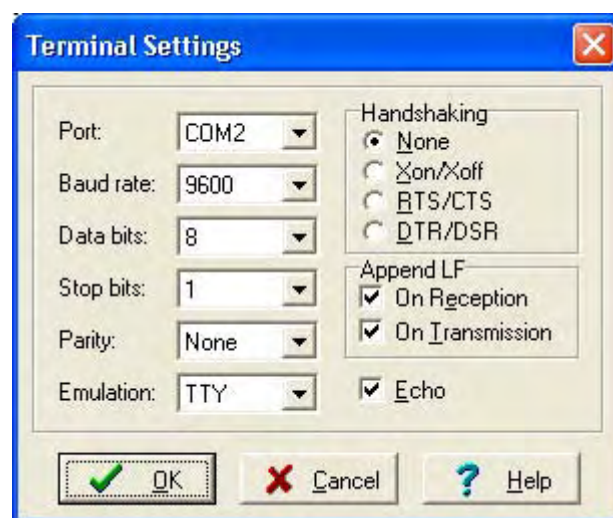
```
{
```

```
putchar(message[i]);
```

```
}
```


در اینجا نوشتن کدهای برنامه به پایان رسیده است. حالا در منوی Project، قسمت Configure را انتخاب کرده و از قسمت After make گزینه Program the Chip را انتخاب کنید و با فشار دادن دکمه OK پنجره را ببندید. در این قسمت باید تنظیمات مربوط به برنامه Terminal را انجام دهید. این برنامه امکان مشاهده اطلاعات رسیده از پورت سریال توسط تراشه و یا امکان فرستادن اطلاعات برای آن را فراهم می‌آورد.

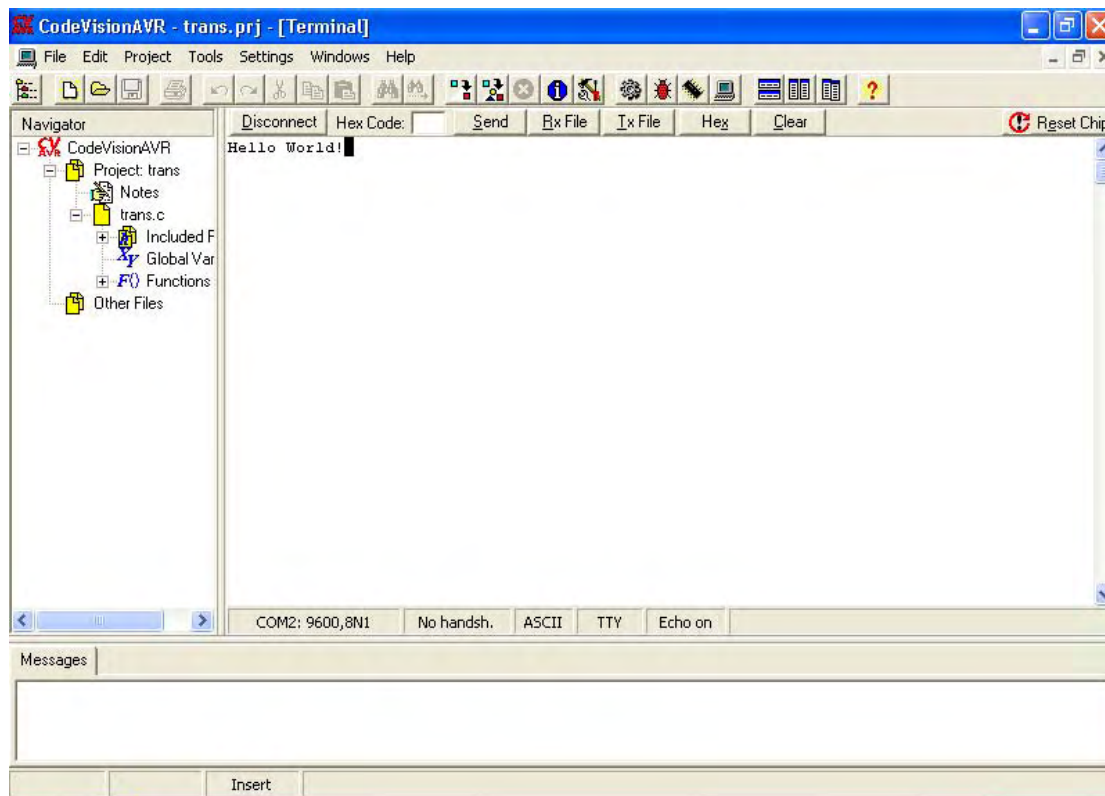
از منوی Settings قسمت Terminal را انتخاب کنید. با انجام این کار پنجره Terminal Settings باز می‌شود. (شکل ۵-۴)



شکل ۵-۴ پنجره Terminal Settings

در قسمت Port باید نام پورت سریالی را که تراشه به آن متصل است انتخاب کنید. در قسمت Baud rate، 9600. در Data bits، عدد 8 و در Stopbits، عدد 1 را انتخاب نمایید و دکمه OK را فشار دهید. در ادامه برنامه را کامپایل کرده و تراشه را برنامه‌ریزی کنید.

حالا باید برنامه Terminal را اجرا کرده و چگونگی اجرا شدن برنامه را مشاهده نمایید. برای انجام این کار از منوی Tools، Terminal را انتخاب کنید و یا از نوار ابزار دکمه Run the Terminal را فشار دهید. همان طور که انتظار داشتیم عبارت "Hello World" یک بار روی صفحه نوشته می‌شود. (شکل ۵-۵) توسط این برنامه می‌توانید صفحه را پاک کرده و تراشه را RESET کنید.



شکل ۵-۵ برنامه Terminal

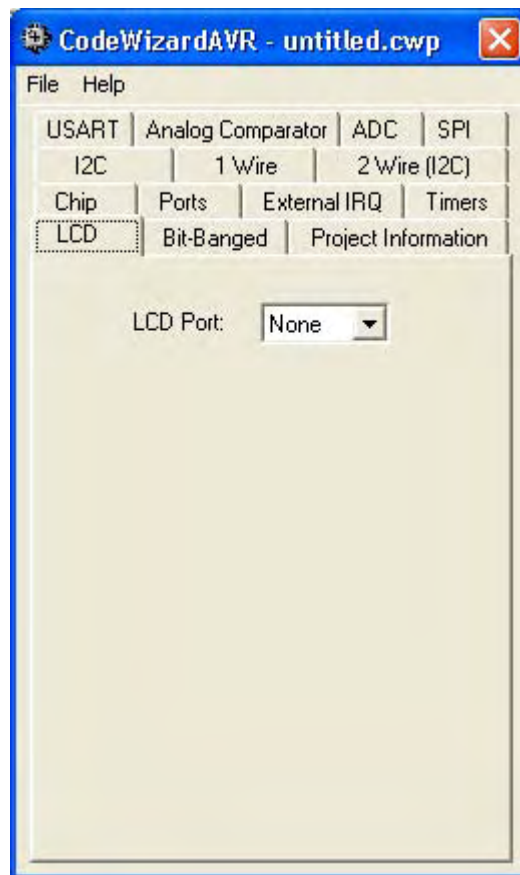
حالا اگر بخواهیم این عبارت به صورت متوالی به کامپیوتر فرستاده شود باید حلقه (For) نوشته شده در قسمت قبل را در داخل حلقه `while` بنویسیم. این کار را انجام دهید و نتیجه را ببینید.

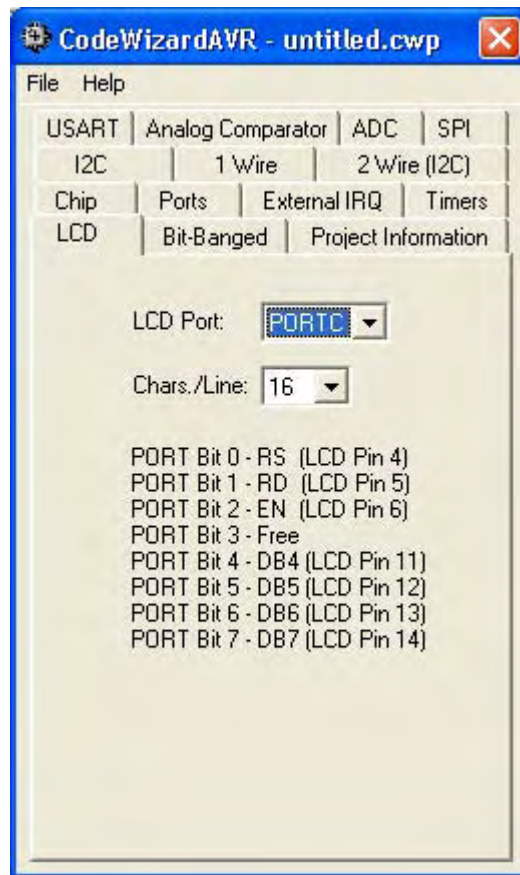
همانطور که دیدید برقراری ارتباط سریال در نرم افزار CodevisionAVR بسیار ساده است.

LCD

در این قسمت قصد داریم تا کار با LCD را بیاموزیم. می‌خواهیم برنامه‌ای بنویسیم که به صورت مداوم از شماره ۱ تا ۱۰۰ را با فاصله ۰,۲ ثانیه بشمارد و هر بار با رسیدن به مقدار ۱۰۰، ۲ ثانیه صبر کرده و دوباره از عدد ۱ شروع کند. در این برنامه از یک LCD کارکتری ۱۶×۲ استفاده می‌کنیم.

ابتدا پروژه جاری را ببندید و نرم افزار CodewizardAVR را اجرا کنید. بعد از تنظیم نوع تراشه و کریستال مورد استفاده، قسمت LCD را انتخاب نمایید (شکل ۵-۶) و در لیست انتخابی LCD Port، PORTC را انتخاب کنید.

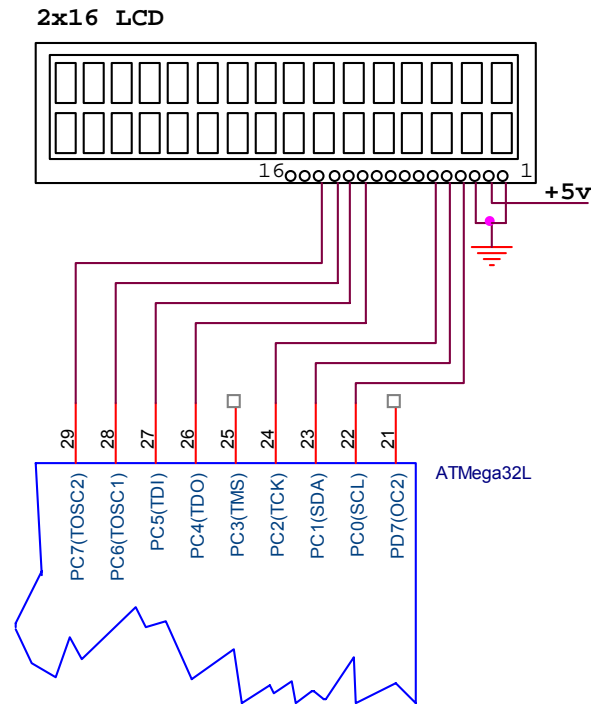




شکل ۵-۷ تنظیمات مربوط به LCD

در قسمت Chars./Line هم عدد ۱۶ را انتخاب کنید. عدد ۱۶ به این معنا است که LCD مورد استفاده یک LCD کارکتری ۱۶×۲ است. همان طور که در قسمت پایین پنجره می‌بینید، (شکل ۵-۷) طریقه اتصال پایه‌های مختلف LCD به پینهای پورت C مشخص شده‌اند. البته علاوه بر این پایه‌ها، در عمل حداقل سه پایه دیگر از LCD هم باید تنظیم شوند. این پایه‌ها به قرار زیرند:

پایه اول به زمین، پایه دوم به منبع تغذیه +۵ ولت و پایه سوم هم که برای کنترل کنتراست استفاده می‌شود باید به زمین متصل گردد.



شکل ۵-۸ اتصال LCD به AVR

در اینجا تنظیمات اولیه به پایان رسیده است. بنابراین باید پروژه جدیدی ایجاد و کدهای موردنظر را تولید کنید.

حالا کافی است تا کدهایی را برای انجام عمل شمارش و نمایش آنها بر روی LCD به برنامه اضافه کنیم. بعد از دستور، `<mega32.h>`، دستورات `<stdio.h>` و `<delay.h>` را اضافه کنید. سپس در اول تابع `main`، بعد از عبارت `//Declare your local variables here` دو متغیر را به صورت زیر تعریف کنید.

```
int i=0;
```

```
char lcd_buffer[20];
```

متغیر `i` یک شمارنده است و متغیر `lcd_buffer` عبارتی را که باید در LCD نشان داده شود در بر می گیرد.

کدهای اصلی برنامه را در درون حلقه `while` و بعد از عبارت `//place your code here` بنویسید:

```
for(i=0;i<101;i++)
```

```
{
```

```
    sprintf(lcd_buffer,"Count from 0-100=%u",i);
```

```
    lcd_clear();
```

```
    lcd_puts(lcd_buffer);
```

```
    delay_ms(200);
```

```
}
```

```
delay_ms(2000);
```

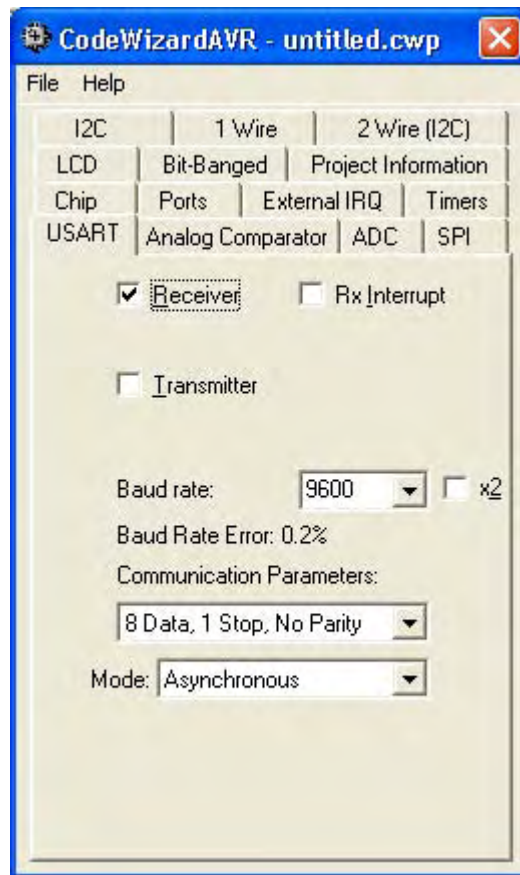
حالا برنامه را کامپایل کرده، تراشه را برنامه‌ریزی کنید و نتیجه را مشاهده نمایید.

لازم به ذکر است که در برنامه بالا، هر بار توسط دستور (`lcd_clear`) LCD پاک شده و به کمک دستور (`lcd_puts`)، عبارتی را در آن می‌نویسیم.

LCD و ارتباط سریال

در این بخش می‌خواهیم برنامه‌ای بنویسیم که در آن تراشه اطلاعات را از پورت سریال بگیرد و آنها را بر روی LCD متصل شده به پورت C نشان دهد. سخت افزار لازم برای این برنامه در دو برنامه قبل مورد توجه قرار گرفت، (شکلهای ۵-۱ و ۵-۸) بنابراین در اینجا تنها به قسمت نرم افزاری آن اشاره می‌کنیم.

همانند برنامه‌های قبل ابتدا برنامهٔ CodewizardAVR را اجرا کنید، پس از تنظیم نوع تراشه و مقدار کریستال مورد استفاده در قسمت Chip، قسمت LCD را انتخاب کرده و در لیست انتخاب LCD Port، PORTC را انتخاب کنید. سپس قسمت USART را انتخاب نمایید. از آنجایی که می‌خواهیم تا از تراشه به عنوان گیرنده استفاده کنیم، قسمت Receiver را علامت بزنید. با انجام این کار موارد دیگری به صفحه اضافه می‌شوند که می‌توانید در آن پارامترهای مربوط به ارتباط سریال را تنظیم کنید. (شکل ۵-۹) این پارامترها را همانند مثال قبلی در مورد ارتباط سریال تنظیم کنید.



شکل ۵-۹ تنظیم USART به صورت گیرنده

بعد از انجام این مراحل، دیگر کار تنظیمات اولیه به پایان رسیده و باید پروژه جدیدی ایجاد و کدهای موردنظر را تولید کنید.

حالا باید متغیرهای مورد نیاز در برنامه را در قسمت `//Declare your local Variables here`، در ابتدای تابع `main` به صورت زیر بنویسید.

```
int i;
char code;
char lcd_buffer[38];
```

که در آن `i` یک شمارنده است. `code`، کارکتری است که هر بار از طریق پورت سریال دریافت می‌شود و آرایه `lcd_buffer`، کارکترهایی است که باید در LCD نوشته شوند.

حالا باید خطوط اصلی را در داخل حلقه `while(1)` به صورت زیر اضافه نماییم.

```
code=getchar();
```

```

if (code!="")
{
for (i=0;i==36;i++)
{
lcd_buffer[i+1]=lcd_buffer[i];
}
lcd_buffer[0]=code;
lcd_puts(lcd_buffer);
}

```

دستور () `getchar` در هر بار اجرای حلقه، کارکتری را از پورت سریال می‌گیرد و آن را در متغیر `code` قرار می‌دهد. دستور شرطی `if` موجب می‌شود تا اگر واقعاً کارکتری از طریق پورت سریال برای تراشه فرستاده شده آن کارکتر را بر روی LCD به همراه کارکترهای قبلی نشان دهد و در غیر این صورت کاری انجام ندهد. دستورات موجود در داخل حلقه `for` هم موجب می‌شوند تا با رسیدن هر کارکتر، کارکترهای ماقبل یک واحد به سمت راست شیفت پیدا کرده و به همراه کارکتر جدید بر روی صفحه LCD نشان داده شوند. در اینجا نوشتن برنامه به پایان رسیده و می‌توانید آن را کامپایل نموده و تراشه را برنامه‌ریزی کنید.

می‌توانیم صحت برنامه نوشته شده را در هم عمل ببینید. برای این منظور ابتدا نرم افزار Notepad را اجرا کرده و

عبارت "Hello World" را در آن تایپ کنید و آن را با نام دلخواه ذخیره نمایند. (شکل ۵-۱۰)