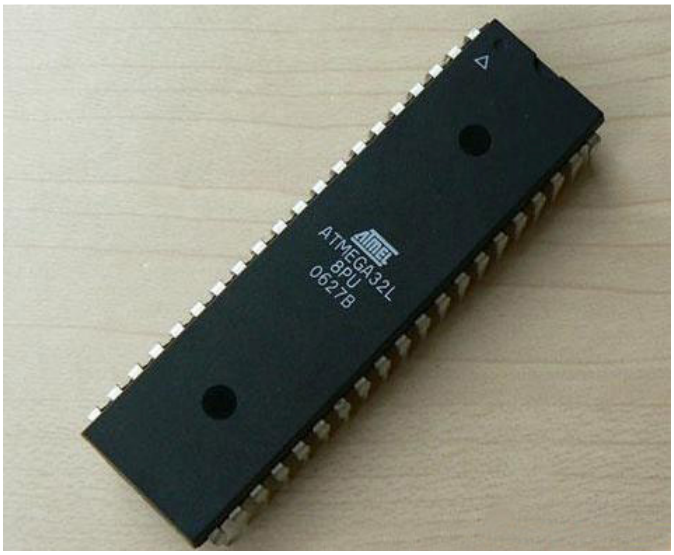
به نام خدا

آموزش میکروکنترلر AVR به زبان C (WinAVR)

(جلسه اول)



میکروکنترلر چیست؟

میکروکنترلرها آی سی های برنامه پذیر می باشند که می توان از آنها برای ساخت وسایل دیجیتال و کنترل کننده مانند ماشین حساب، تایمر، ساعت، سیستم های سخن گو، MP3 Player، فرکانس متر، بیسیم دیجیتال، قفل رمز دیجیتال، روبات و سایر ابزار آلات دیجیتالی و کنترلی استفاده نمود. میکرو کنترلرها از بخش های مختلفی مانند CPU، SRAM، FLASH تشکیل شده و شامل امکاناتی نظیر تایمر، مبدل آنالوگ به دیجیتال، مبدل دیجیتال به آنالوگ (در بعضی از مدلها) و بخش های دیگر می باشند. میکروکنترلرها با مجموعه ای از پایه ها که اصطلاحاً آنها را پورت می نامند با دنیای خارج در ارتباط اند. برای کار با میکروکنترلرها و استفاده از امکانات آنها باید توسط برنامه های خاص کامپیوتری به نام کامپایلر برنامه ای برای میکرو نوشته شود؛ سپس برنامه توسط دستگاهی به نام پروگرمر به میکروکنترلر انتقال داده می شود. برنامه در حافظه فلش میکرو ذخیره می شود و به طور دائم در آن باقی می ماند. میکروکنترلرها باعث کوچک شدن مدارهای الکترونیکی شده و قابلیت بروز آوری نرم افزار را بدون دستکاری سخت افزار و تنها با تعویض برنامه به دستگاه می دهند.

انواع میکروکنترلرها:

از جمله میکروکنترلرها می توان به میکروکنترلرهای خانواده PIC از شرکت میکروچیپ و سری 8051 از اینتل و خانواده AVR از اتمل اشاره کرد. همچنین سری ARM نیز از جمله میکروکنترلرهای پیشرفته تر می باشند که در ساخت تجهیزات حرفه ای و صنعتی از قبیل کارت های توسعه PC تلفن های هوشمند کاربرد دارند. در این سری مقالات به معرفی و نحوه کار با میکروکنترلرهای خانواده AVR به همراه آموزش برنامه نویسی برای این میکروکنترلرها به زبان C به کمک کامپایلر رایگان WinAVR می پردازیم.

پیش نیازها و موارد لازم:

برای شروع کار با میکروکنترلرها لازم است با مباحثی مانند الکترونیک، دیجیتال، برنامه نویسی C، نحوه بستن مدارهای الکترونیکی و اصول کار با کامپیوتر آشنایی کافی داشته باشید از جمله نیازمندیهای سخت افزاری نیز دستگاهی به نام پروگرمر AVR می باشد که در مدل های مختلفی تولید می شود. همچنین به وسایل آزمایشگاهی مانند منبع تغذیه، تخته آزمایش یا بِرِد بُرد، یکی از میکروکنترلرهای خانواده AVR مثل ATmega8 یا ATmega16 ، سیم، کلید فشاری، دیود نورانی LED، اهم متر و سایر تجهیزاتی که در یک آزمایشگاه شخصی الکترونیک مشاهده می شود نیاز خواهید داشت. همچنین یک کامپیوتر برای برنامه نویسی و انتقال برنامه به میکرو مورد نیاز است. در این سری مقالات به تدریج با نحوه کار و برنامه نویسی با کامپایلر رایگان WinAVR و نحوه انتقال برنامه به میکروکنترلر آشنا خواهیم شد. همچنین در هر جلسه بخش تازه ای از میکرو و یا نحوه ارتباط میکرو با یک ابزار جدید مورد بحث قرار می گیرد.

شروع:

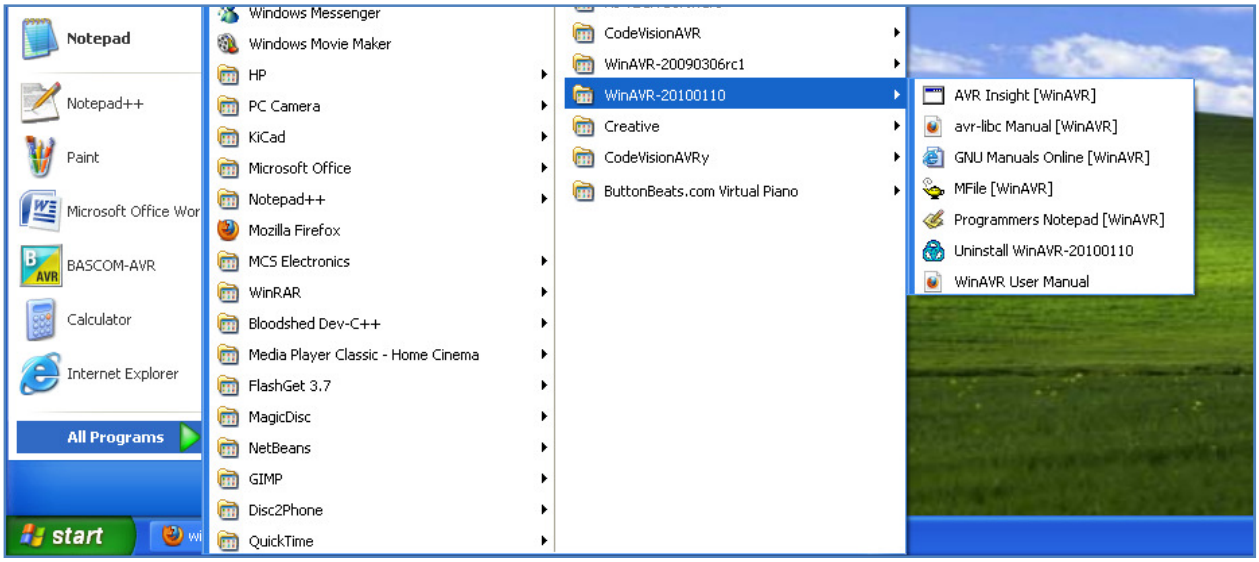
در اینجا به معرفی کامپایلر GNU و رایگان WinAVR می پردازیم که زبان آن C بوده و از هسته GCC استفاده می نماید. این کامپایلر بر روی سیستم عامل های ویندوز و لینوکس قابل اجرا بوده و به همین دلیل سورس ها و پروژه های رایگان زیادی در رابطه با آن وجود دارد. یکی از کتابخانه های قدرتمند موجود کتابخانه avrlib می باشد که حاوی هدرفایل های ارتباط با ادوات جانبی متداول از قبیل LCD و MMC و... است. در این سری مقالات به طور مفصل با این کامپایلر و زبان C آشنا می شویم.

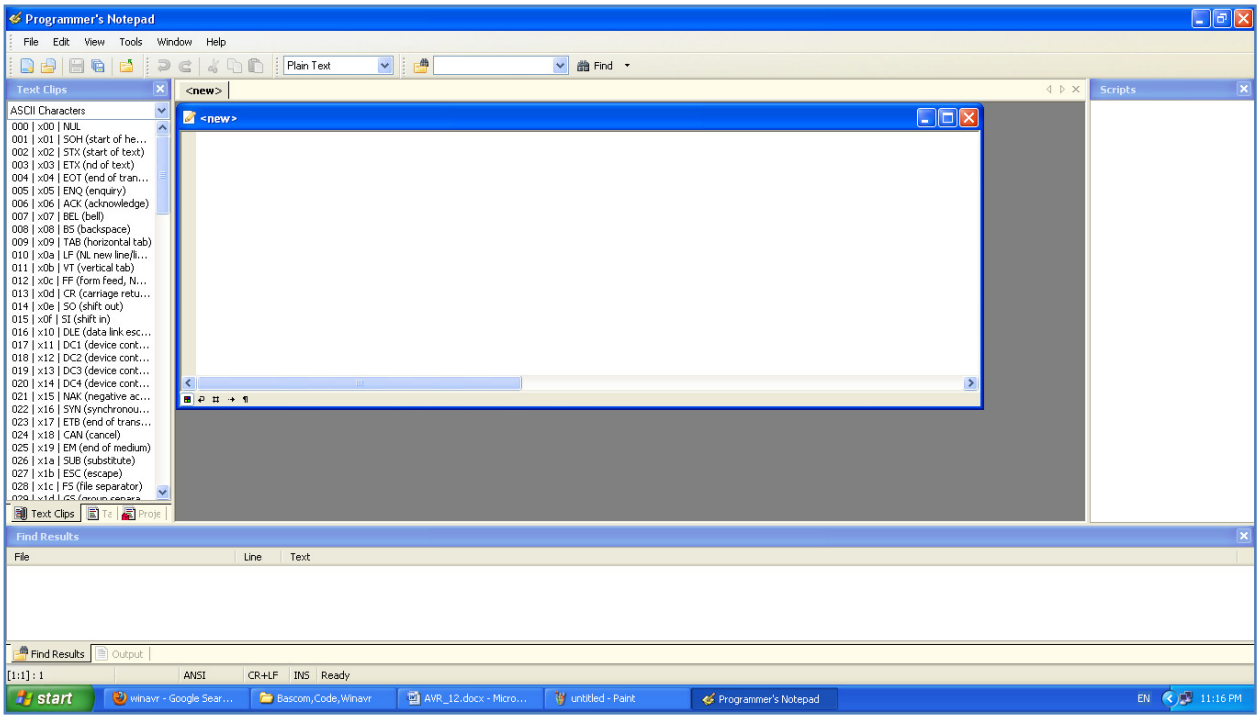
دانلود، نصب و راه اندازی WinAVR

برای دانلود WinAVR می توانید به آدرس صفحه خانگی این کامپایلر که در زیر به آن اشاره شده است مراجعه و آن را بارگیری نمایید:

http://winavr.sourceforge.net

همچنین می توانید با سرچ عبارت WinAVR در اینترنت آدرس جدید دانلود آن را پیدا کنید. پس از دانلود و نصب برنامه به منوی استارت مراجعه کرده و بر روی All Programs کلیک کنید. مطابق شکل زیر پوشه برنامه مذکور حاوی چندین فایل مختلف می باشد. نظر به اینکه وین ای وی آر یک کامپایلر خط فرمان می باشد در این نسخه تحت ویندوز، ادیتور رایگان Programmers Notepad برای محیط کد نویسی آن پیکره بندی شده است. با کلیک بر روی آن تصویر زیر ظاهر می شود.

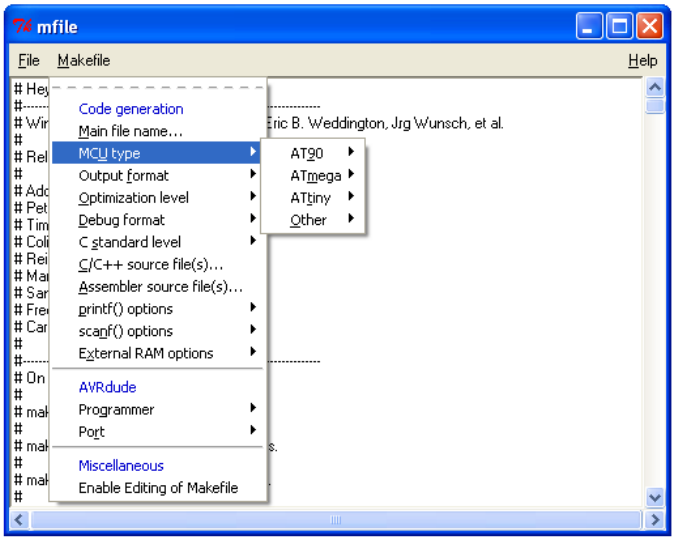




کدنویسی در این محیط انجام می شود. البته بهتر است از نوار ابزار بالا از قسمت Plain Text بر روی گزینه C / C++ کلیک شود تا کدهای نوشته شده حالت رنگی و فرمت بندی شده به خود بگیرند. این ادیتور برای مشاهده سورسهای زبان های مختلف گزینه خوبی به شمار می رود و یک ویرایشگر همه منظوره می باشد.

برنامه بعدی که در پوشه WinAVR قرار گرفته است MFile نام دارد که فایل پیکره بندی کامپایلر را تولید می کند. همانطوریکه می دانید برنامه های خط فرمان مانند اسمبلر اتمل برای پیکر بندی، پارامترهایی را به صورت کاراکتر و عدد دریافت می کنند. مثلاً در اسمبلر اتمل بایستی کاراکترهای f و l را که هر یک تنظیم خاصی برای اسمبلر هستند به صورت مقابل وارد کنیم: avrasm -fl Prog1.asm

کامپایلر GCC نیز از این قاعده مستثنی نبوده و البته پارامترهای آن بسیار زیاد می باشد؛ پارامترهایی از قبیل نوع میکرو، نوع پروگرمر، فرکانس کاری، نام فایل سورس اصلی، نام هدر فایلها و ... . برنامه MFile به ما کمک می کند تا با استفاده از تنظیمات گرافیکی، فایلی به نام Makefile حاوی تمام تنظیمات لازم برای کامپایلر تولید کنیم. پنجره این برنامه مطابق با شکل صفحه بعد می باشد.

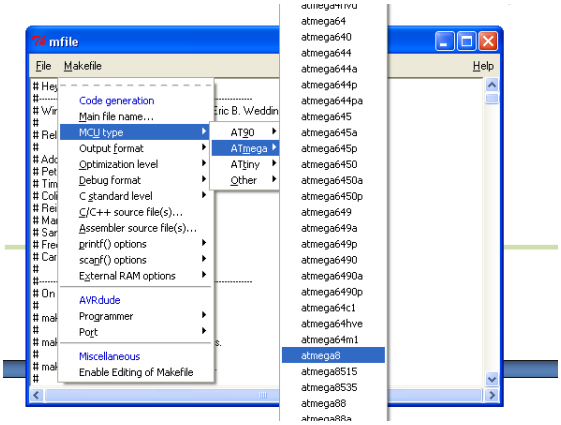


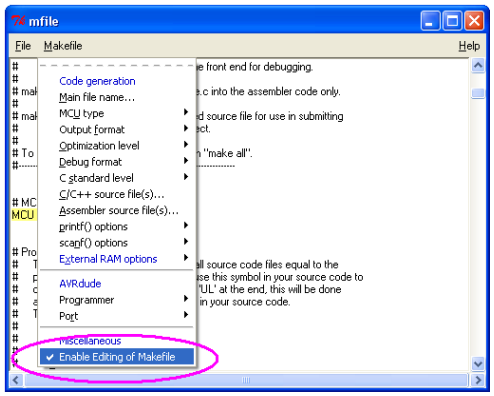
در این برنامه پس از تعیین نوع میکرو و نام فایل اصلی که به صورت پیش فرض main می باشد، در صورتی که برنامه ما شامل هیچ فایل و کتابخانه اضافه در اطراف سورس نباشد از منوی فایل بر روی Save As... کلیک کرده و فایل ایجاد شده را بدون تغییر نام در پوشه ای که قصد نوشتن برنامه را داریم ذخیره می نماییم. فایل C ایجاد شده توسط Programmers Notepad نیز بایستی در همان پوشه ذخیره گردد.

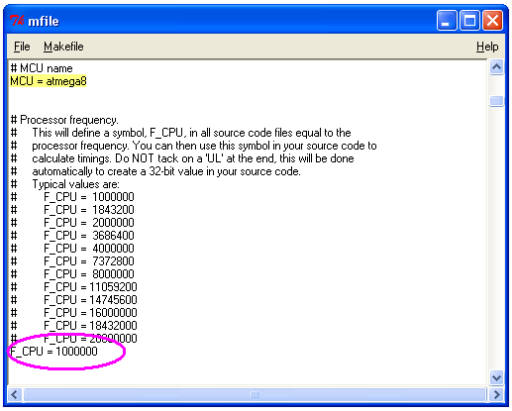
پس از آشنایی با محیط این کامپایلر رایگان نوبت به ایجاد یک پروژه ساده می رسد تا با نحوه عملکرد این محیط آشنا شویم و به آموزش ساختار زبان C بپردازیم. در قسمت بعد در ابتدا یک برنامه ساده چشمکزن با LED و میکروی ATmega8 می نویسیم و به تدریج با مفاهیم فایل های کتابخانه ای و نوشتن Header فایلها و توابع آشنا می شویم.

برنامه نویسی اولین پروژه با WinAVR:

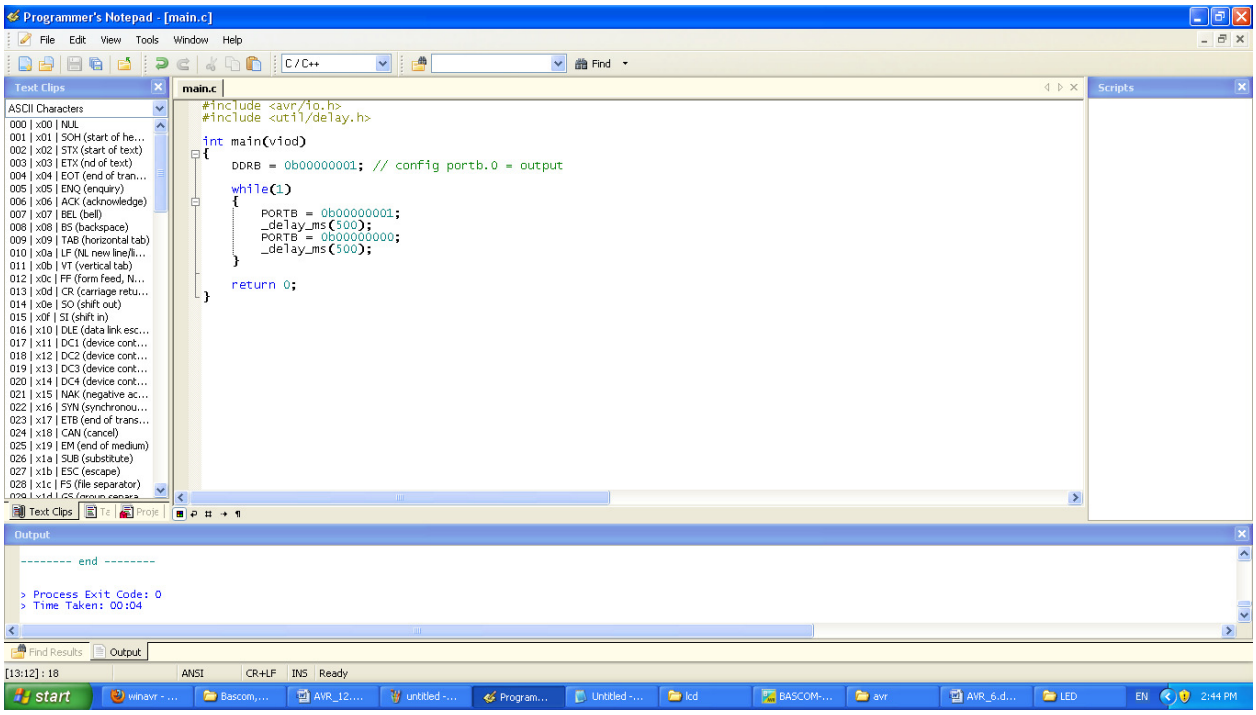
اولین پروژه ما یک چشمکزن ساده با ATmega8 می باشد که دیود LED قرار گرفته بر روی پورت B.0 را با فرکانس یک هرتز روشن و خاموش می نماید. در این پروژه از اسیلاتور داخلی 1 مگاهرتز میکرو (گزینه پیش فرض تنظیمات کارخانه) استفاده شده است. برای شروع کد نویسی در ابتدا پوشه ای به نام LED در درایو C ایجاد می کنیم. سپس از منوی استارت بر روی All Programs و WinAVR کلیک کرده، سپس برنامه MFile را بر می گزینیم. مطابق شکل پایین ابتدا نوع میکرو را انتخاب می کنیم. سپس همانند شکل بعدی از منوی Makefile بر روی آخرین گزینه یعنی Enable Editing of Makefile کلیک می کنیم تا قادر به ویرایش محتوای فایل شویم. در مرحله بعد مانند شکل سوم این صفحه مقدار F_CPU را از 8000000 به 1000000 تغییر می دهیم تا فرکانس کاری CPU را تنظیم کنیم.



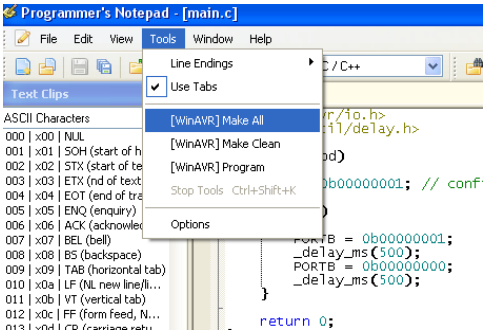




پس از انجام تنظیمات از منوی File > Save As... فایل ایجاد شده را بدون تغییر نام در پوشه LED واقع در درایو C ذخیره می کنیم و پنجره برنامه را می بندیم. سپس برنامه Programmers Notepad را باز کرده و از نوار ابزار بالا C / C++ را انتخاب میکنیم و کد زیر را در محیط مذکور می نویسیم و در نهایت آن را با نام main.c در پوشه LED ذخیره می کنیم.

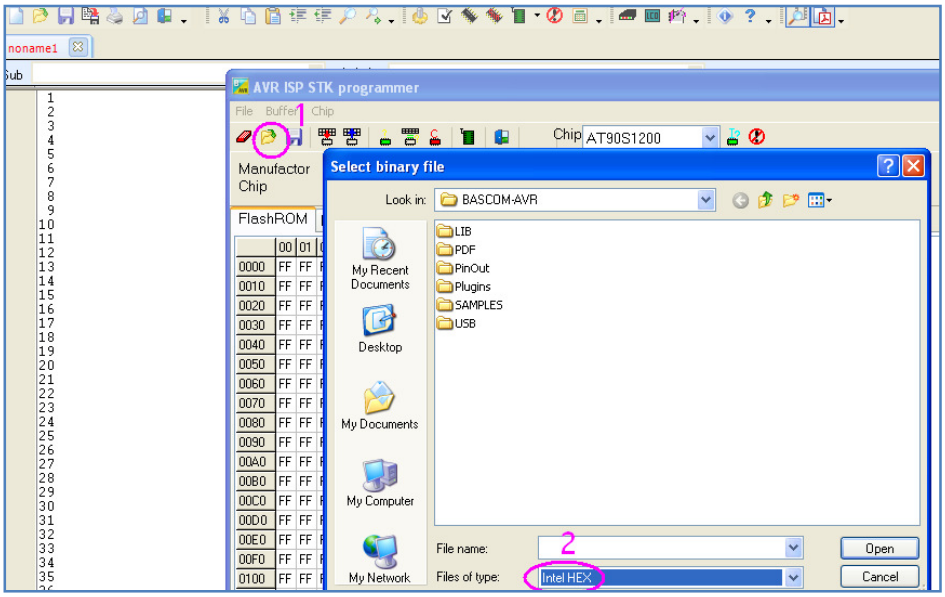


برای کامپایل برنامه از منوی Tools بر روی Make All و یا Make Clean کلیک کنید. پس از انجام عملیات در صورتی که مرتکب خطایی نشده باشید در قسمت پایین پنجره عبارت Process Exit Code: 0 نمایش داده می شود. در این حالت اگر به پوشه LED مراجعه کنیم فایل Hex را مشاهده خواهیم کرد و می توانیم آن را در میکرو بارگزاری نماییم.

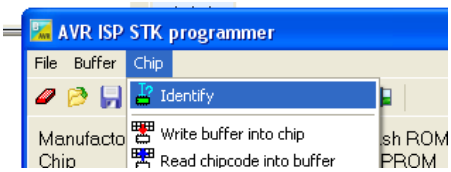


پروگرم کردن میکرو:

در این قسمت نوبت به پروگرم کردن میکرو (انتقال فایل هگز به حافظه فلش میکرو) می رسد. لازم به ذکر است برای پروگرم کردن میکرو از نسخه دموی کامپایلر Bascom استفاده می کنیم. برای دانلود نسخه آزمایش این کامپایلر که در اینجا فقط از بخش پروگرمر آن استفاده می کنیم می توانیدبه سایت www.mcselec.com مراجعه کنید. در صورتی که پروگرمر شما نرم افزاری خاصی برای پروگرم کردن میکرو داشته باشد نیازی به دانلود استفاده از بسکام نیست و می توانید طبق توضیحات دفترچه پروگرمر خود برای انتقال فایل Hex تولید شده به میکرو استفاده نمایید. در صورت نیاز به پروگرمر Bascom کامپایلر بسکام را باز کرده، روی File > New کلیک کنید تا یک فایل خالی باز شود، سپس دکمه F4 را بزنید محیط پروگرمر STK نمایش داده شود. مطابق شکل صفحه بعد روی شکل پوشه Open واقع در نوار ابزار پنجره پروگرمر کلیک کنید تا پنجره انتخاب فایل هگز ظاهر شود. نظر به اینکه این پنجره به صورت پیش فرض به دنبال فایل باینری میگردد و فایل های با پسوند Hex را نشان نمی دهد بایستی از قسمت پایین Files of type گزینه Intel HEX را انتخاب نمایید. با استفاده از این روش به راحتی می توان توسط کامپایلر بسکام هر نوع فایل Hex را در میکرو پروگرم نمود. دلیل استفاده از این روش عملکرد نامناسب پروگرمر داخلی WinAVR می باشد که معمولاً قادر به شناسایی پورت LPT نیست.

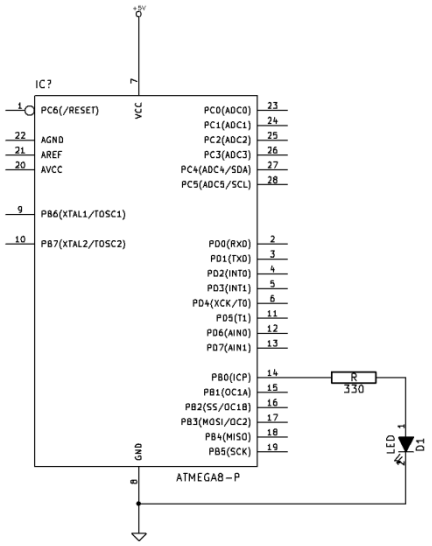


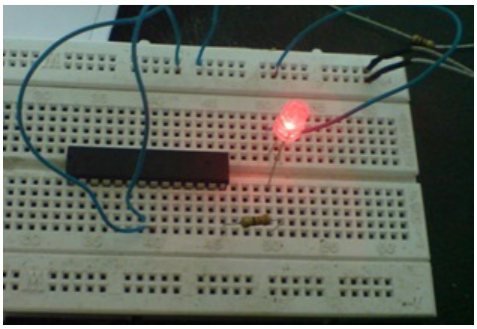
نکته: پس از انتخاب فایل هگز بایستی از منوی Chip یک بار بر روی Identify کلیک کنیم تا دکمه های این برنامه فعال شوند. (ضمناً در این برنامه نیازی به تنظیم فیوزبیت ها نمی باشد، نحوه تنظیمات در مقالات بعدی شرح داده خواهد شد).



بستن مدار به صورت فیزیکی و تست:

در این بخش نوبت به تست پروژه به صورت فیزیکی می رسد. پس از پروگرم کردن میکرو مداری مطابق با شکل زیر بر روی برد بورد بسته و پس از وصل تغذیه مشاهده می کنیم که دیود نورانی LED به تناوب چشمک می زند: (تغذیه مورد نیاز برای این میکرو ۵ ولت مستقیم می باشد).





تحلیل برنامه چشمکزن به زبان C

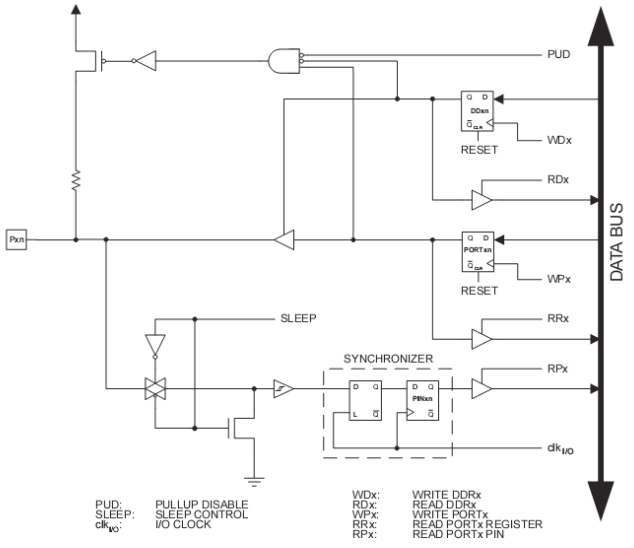
پس از تست پروژه یاد شده نوبت به تحلیل برنامه و چگونگی ساختار زبان C می رسد. لازمه برنامه نویسی با Win AVR آشنایی مقدماتی با زبان C می باشد. همانطوریکه می دانید در زبان C از یک سری ساختارهای خاصی برای برنامه نویسی استفاده می شود؛ به طور مثال برای ایجاد حلقه از دستور While و برای انتساب از دستور = بهره گرفته می شود. واقعیت اینست که برنامه نویسان اسمبلی در محیط C احساس راحتی بیشتری می کنند و فرصت کمتری را برای نوشتن کد های تکراری صرف می کنند. به طور مثال در زبان اسمبلی برای ایجاد تاخیر بایستی ده ها خط کد نوشته و به تناوب چندین رجیستر را پر و خالی کنیم تا مدتی تاخیر ایجاد شود. ولی در زبان C با استفاده از تابع (delay_ms_ به راحتی می توانیم مقدار زمان تاخیر خود را بدون توجه به تغییر فرکانس CPU ایجاد کنیم.

توابعی که در بالا نام برده شد توسط همین کامپایلر و ادیتور تهیه شده اند و در آینده با نحوه ایجاد آنها آشنا خواهیم شد. توابع می توانند در داخل سورس پروژه قرار بگیرند و یا در فایلهای دیگری به نام هدر فایل. ما در این پروژه از دو هدر فایل استفاده کرده ایم. هدر فایل اول به نام io.h شامل نام پورت ها می باشد. با include کردن این هدر فایل می توانیم از نام های PORTB و DDRB و... به جای آدرس پورت ها استفاده نماییم. Header file بعدی بنام delay.h شامل دستوراتی برای ایجاد تاخیر در برنامه می باشد که هر دو هدرفایل فوق به عنوان بخشی از کتابخانه استاندارد WinAVR محسوب می شوند.

نکته دیگر در برنامه های C اینست که هر برنامه از توابع تشکیل شده است. ما می توانیم بخش های مختلف برنامه را به صورت تابع به دنبال هم بنویسیم و با فراخوانی هر یک عمل مورد نظر را انجام دهیم. مثلاً در این برنامه با فراخوانی (delay_ms(500_ تابع ایجاد تاخیر را با مقدار آرگومان 500 فراخوانی کرده ایم. برای فراخوانی توابع بایستی نام تابع را بنویسیم. تنها تابعی که پس از شروع برنامه به طور خودکار اجرا می گردد تابع main نام دارد. به همین روی ما موظفیم تابعی با همین نام نوشته و کد خود را داخل آن قرار دهیم. این تابع نیازی به فراخوانی نداشته و حتی بهتر است که هرگز آنرا به صورت دستی فراخوانی نکنیم. این تابع در صورتی که با موفقیت اجرا شود (که البته همیشه هم همینطور است) بایستی مقدار 0 را بر گرداند. به همین روی بایستی در انتهای این تابع عبارت return 0; را بنویسیم. این تابع در شرایط عادی هیچ مقداری را به عنوان آرگومان ورودی نمی گیرد پس از جلوی آن عبارت void به معنای هیچ را می نویسیم. نکته دیگر در خصوص حیطه عملکرد توابع و ساختارها می باشد. محدوده قلمرو هر تابع با { شروع و به } ختم می شود. نکته مهم آخر استفاده از ; در پایان هر دستور می باشد. کامپایلر C به گونه ای طراحی شده که دستور ها را با علامت ; از یکدیگر جدا می کند. در کامپایلر بیسیک دستورها با کاراکتر انتهای خط از یکدیگر جدا می شوند و به همین دلیل در هنگام کامپایل سورسهای بیسیک که توسط سیستم های مختلف از قبیل مک و لینوکس تولید شده باشند مشکل ایجاد می شود چرا که ویندوز از دو کاراکتر CR-LF برای انتهای خط استفاده می کند ولی در سیستم های دیگر از یکی از این کاراکترها استفاده می شود. زبان C به دلیل اینکه یک زبان بین المللی می باشد و سورسهای آن ممکن است توسط هر سیستمی ایجاد شود از کاراکتر آشکار ; برای جداسازی دستورات از یکدیگر استفاده می کند که بایستی مدنظر قرار داده شود.

یکی از ساختارهای به کار برده شده در این برنامه حلقه شرطی While می باشد. این حلقه تا زمانی که دستور مقابل آن درست باشد و مقدار 1 را برگرداند دستورات محدوده قلمرو خود را تکرار می کند. نظر به اینکه ما در این برنامه به یک حلقه بینهایت احتیاج داریم در جلوی این دستور عدد 1 را می نویسیم تا این حلقه برای همیشه اجرا گردد. 1 به معنی True است.

سایر دستورات نیز دستورات انتساب هستند که پین شماره صفر پورت B را روشن رو خاموش می کنند و نیز پین مربوطه را در حالت خرو جی قرار می دهند. به طور دقیق تر دستور ;DDRB = 0b00000001 برای تنظیم پایه شماره 0 پورت B به عنوان خروجی به کار می رود. در میکروکنترلرها معمولاً هر 8 پایه را با نام پورت شناخته و برای آن نام A، B، C، D و... قرار می دهند. ضمناً هر کدام از پایه های پورت را نیز از 0 تا 7 شماره گزاری می کنند. مثلا B.0 تا B7. از طرفی کمی توان هر پورت را به صورت ورودی یا خروجی تنظیم کرد. برای این منظور بایستی در رجیستر DDR که به معنای Data Direction Register یا رجیستر جهت داده می باشد برای تنظیم پایه مربوطه به صورت خروجی عدد 1 و تنظیم به صورت ورودی عدد 0 را بنوسیم. نظر به اینکه برای هر پورت یک رجیستر DDR خاص مثل DDRB برای پورت B وجود دارد و هر پورت نیز از 8 پایه تشکیل شده است بایستی تمام پایه های یک پورت را یکجا مقدار دهی کنیم. برای این منظور باید یک بایت اطلاعات را در رجیستر مربوطه بنویسیم و در زبان C برای نوشتن یک مقدار در یک رجیستر می توانیم از علامت = استفاده نماییم. ضمناً از مزیت های زبان C اینست که می توانیم با قرار دادن علامت 0b قبل از عدد، آن عدد را به صورت باینری بنویسیم تا خواندن آن آسان تر باشد. در این صورت سمت چپ ترین رقم بیت شماره 7 یا همان پایه شماره 7 پورت و سمت راست ترین بیت نیز بیت شماره 0 یا پایه شماره صفر پورت خواهد بود. بطور مثال در این برنامه با نوشتن دستور DDRB = 0b00000001 پایه شماره صفر پورت B را به صورت خروجی تنظیم کرده و بقیه پایه ها را به صورت ورودی تنظیم نموده ایم. در این حالت با دستور PORTB = 0b00000001 می توان پایه شماره صفر پورت B را روشن کرد و با دستور PORTB = 0b00000000 آن را خاموش نمود. در شکل زیر که از برگه اطلاعاتی میکروکنترلر ATmega8 کپی شده است مدارات داخلی یکی از پینها را مشاهده می کنید. همانطوریکه در شکل مشاهده می شود بر روی هر پایه یک مقاومت با ترانزیستور MOSFET قرار داده شده است. این ترانزیستور در صورتی که DDR برابر با 0 بوده و PORT برابر با 1 باشد فعال شده و مقاومت پول آپ (Pull-up) را روی پایه مربوطه فعال می نماید. در این حالت پایه مربوطه به صورت ورودی تنظیم شده و در حالت عادی 1 می باشد، با اتصال پایه به زمین می توان مقدار 0 را در رجیستر PIN قرار داد و از آن در برنامه استفاده نمود. این روش برای اتصال ورودی سیگنال یا کلید به میکرو استفاده می شود و کاربر را از نصب مقاومت پول آپ خارجی بی نیاز می کند. در جلسات آینده بیشتر در خصوص نحوه تنظیم پایه به صورت ورودی بحث خوهیم کرد.



Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

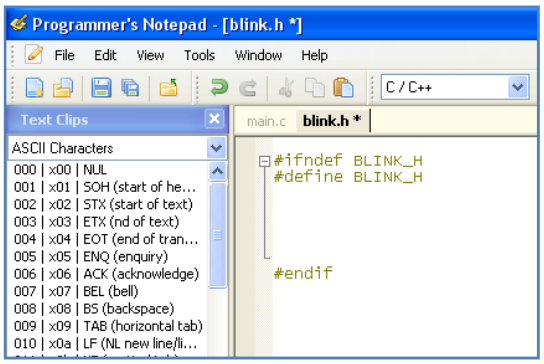
در این قسمت با ساختارهای توابع آشنا شدیم. در بخش بعدی تصمیم داریم کل برنامه ای را که نوشته ایم به صورت یک کتابخانه و هدر فایل عمومی تبدیل کرده و به نام blink.h (چشمکزن) به صورت عمومی منتشر کنیم. در بخش بعدی با نحوه نوشتن توابع به صورت Header file و نحوه استفاده از آنها آشنا خواهیم شد.

طریقه نوشتن Header file

برای نوشتن هدر فایل یک فایل خالی به صورت C / C++ باز می کنیم:

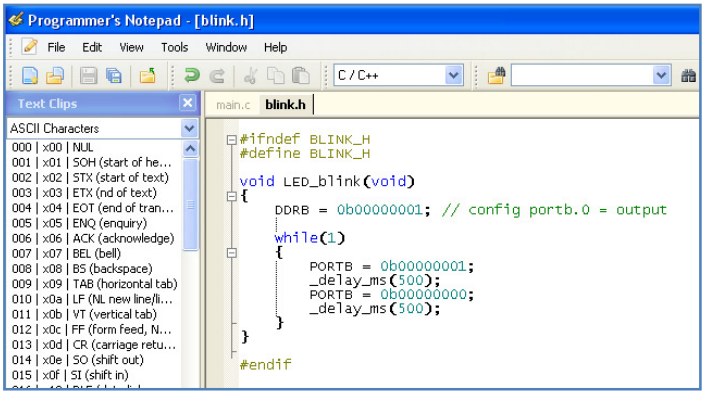
File > New > C / C++

سپس دستورات زیر را در آن می نویسیم تا قالب اصلی هدر فایل ایجاد شود. این دستورات به ما کمک می کند که کتابخانه مربوطه بیش از یک بار تعریف نشود. در پروژه های بزرگ که هرکدام از بخش ها توسط یک برنامه نویس مجزا نوشته می شود ممکن است چندین نفر به طور همزمان یک هدر فایل را include کنند که در این صورت خطاهایی در موقع کامپایل اتفاق می افتد. دستورات زیر در ابتدا چک می کنند که اگر فایل جاری تعریف نشده است آن را تعریف کنند و در غیر این صورت با فایل مربوطه به عنوان یک فایل خالی رفتار شود.

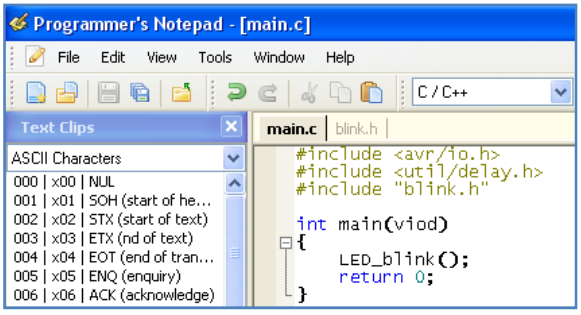


پس از نوشتن دستورات مقابل، فایل مربوطه را در پوشه LED به نام blink.h ذخیره می کنیم. ضمناً عبارت نوشته شده در کد مربوطه بایستی همان نام فایل ولی با حروف بزرگ باشد و به جای نقطه از آندرلاین استفاده شود: BLINK_H

پس از نوشتن قالب هدر فایل، کد برنامه مورد نظر را در داخل این قالب می نویسیم. همانطوریکه می بینید تابعی به نام LED_blink نوشته ایم که هیچ مقداری را نمی گیرد و هیچ مقداری را نیز بر نمی گرداند (پس نیازی به نوشتن return 0; در انتهای آن نیست)



در قسمت بعدی به برنامه main باز می گردیم و خط "#include "blink.h را در ابتدای آن اضافه می کنیم. می بینیم که برنامه ما به شکل بسیار ساده و قابل فهم روبرو تبدیل شده است و تنها لازم است که تابع LED_blink() را فراخوانی کنیم تا برنامه اجرا شود. به همین ترتیب می توانیم کتابخانه های مختلفی را برای خود بنویسیم و بارها و بارها در پروژه های مختلف از آنها استفاده نماییم. توجه داشته باشید که در این برنامه به دلیل اینکه کتابخانه مذکور در داخل پوشه پروژه جاری قرار داشت از ترکیب "blink.h" به جای <blink.h> استفاده کردیم. در واقع ترکیب دوم فقط برای کتابخانه های استاندارد که در پوشه include داخل کامپایلر قرار دارند استفاده می شود و ترکیب اول برای کتابخانه های محلی که در پوشه پروژه قرار دارند.



در جلسه بعدی با کتابخانه بزرگ و رایگان avrlib نوشته Pascal Stang آشنا می شویم که شامل صدها هدرفایل در زمینه ارتباط با LCD های کاراکتری و گرافیکی، کارتهای حافظه MMC و SD، ارتباط با پورت سریال، راه اندازی مبدل آنالوگ به دیجیتال، TCP/IP، FAT، ماژول GPS و... می باشد. در جلسه بعد با کتابخانه آماده lcd.h که برای راه اندازی LCD کاراکتری طراحی شده کار می کنیم. لازم به ذکر است که کتابخانه های رایگان بسیار زیادی از افراد مختلف برای دانلود رایگان بر روی اینترنت قرار داده شده است که به راحتی می توانید از آنها استفاده نمایید.

ادامه دارد...