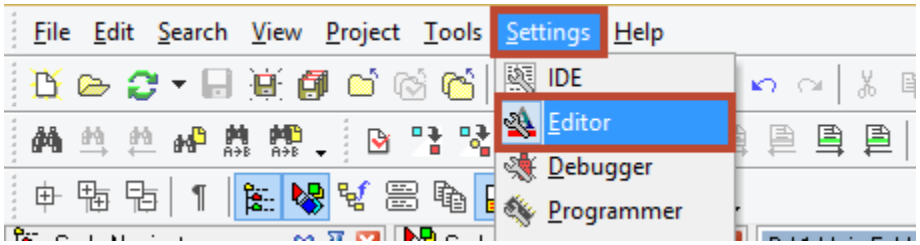
تغییر اندازه فونت محیط برنامه نویسی :

اگرچه شاید موضوع این بخش کمی نامربوط به بخش های دیگر باشد , اما معمولا یکی از مواردی که در محیط کدویژن نیاز می شود تغییر اندازه فونت در محیط برنامه نویسی (این محیط با نام Editor شناخته می شود) است . لذا کمی به توضیح این بخش می پردازیم. تنظیمات کلی مربوط به محیط نرم افزار در منوی بالای صفحه نرم افزار, بخش Settings می باشد . از این بخش گزینه Editor برای تنظیمات محیط برنامه نویسی از جمله تغییر رنگ نوشته ها , اندازه فونت , نوع فونت و .... می باشد . این منوها در شکل زیر نشان داده شده اند .

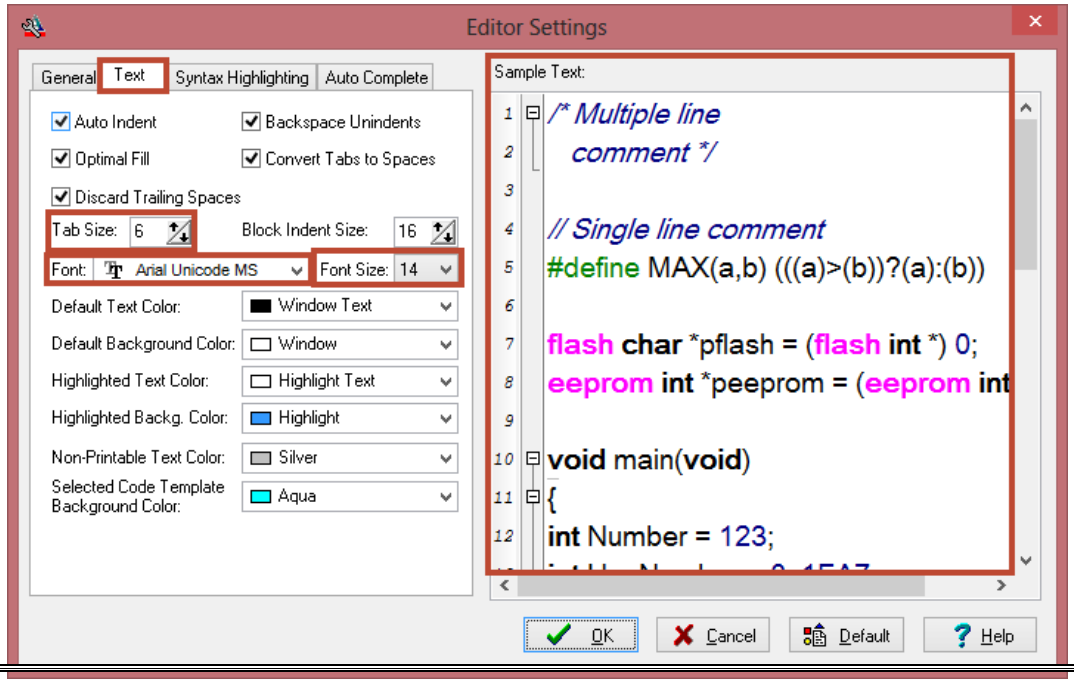


با کلیک بر روی گزینه Editor پنجره زیر باز می شود . از این رو که تنظیمات در این بخش نیز بسیار زیاد هستند , تنها به توضیح تغییر اندازه فونت می پردازیم و بقیه را به خود دانشجو می سپاریم . به این منظور با کلیک بر روی تب دوم (Test) , صفحه مانند شکل زیر می شود . که به ترتیب به بخشهای نشان داده شده در شکل می پردازیم :

Tab Size : زمانی که در برنامه نویسی نیاز داریم تا چند فاصله ایجاد بشود , به جای چند بار فشار دادن کلید Space بر روی کیبورد باید از کلید Tab برروی صفحه کلید استفاده کرد . اما اینکه با هر بار زدن Tab چند فاصله ایجاد بشود بسته به مقدار این گزینه دارد . در اینجا که عدد ۶ وارد شده است , یعنی با هر باز زدن Tab مکان نما به اندازه ۶ بار فشردن کلید Space به سمت جلو می رود .

Font : این بخش که در زیر گزینه Tab Size می باشد بمنظور انتخاب نوع فونت است , که مثلا ممکن است برنامه نویسی بخواهد نوع فونت خود را به دلخواه برای بهتر شدن محیط تغییر دهد که برای این منظور باید نوع فونت را از این بخش انتخاب کند .

Font Size : این بخش نیز بمنظور تغییر اندازه فونت می باشد . شما با تغییر نوع و اندازه فونت در پنجره سمت راست می توانید تغییرات را بر روی کدهای نمونه نوشته شده مشاهده کنید و ببینید که آیا اندازه و نوع فونت انتخابی مناسب است یا نه .



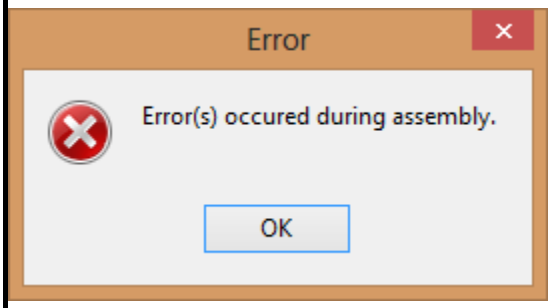
بقیه تنظیمات این بخش مربوط به تغییر رنگها و یا تنظیمات تخصصی دیگر است که شاید در برخی کاربردها برای برنامه نویسان حرفه ای بسیار مفید باشد .

عیب یابی (Debugging):

عیبهای یک برنامه نوشته شده را شاید بتوان در حالت کلی به دو دسته تقسیم کرد . ۱- اشکالات در کامپایل برنامه ۲- اشکالات در حین اجرای برنامه (منظور اشکالاتی است که در حین شبیه سازی یا در عمل با آن مواجه خواهید شد). در این بخش تا حد مختصری در باب خطاهای معمولی که ممکن است در حین کامپایل برنامه با آن برخورد کنید و همینطور بعضی راه حل های آن ها بحث خواهیم کرد و اشکالات در اجرای برنامه را به بعد موکول می کنیم .در جدول زیر در سمت چپ , لیستی از بعضی از خطاهایی است که در حین کامپایل ممکن است با آن مواجه شوید و در سمت راست بعضی راههای رفع آنها .

خطای کامپایل	بعضی راههای رفع خطا
۱) -invalid #include directive -can't open #include file: mega۳۲	این خطاها معمولا به دلیل اشتباه نوشتن کتابخانه ها نشان داده می شود . مثلا ممکن است آخر نام کتابخانه mega۳۲, پسوند h. را نگذاشته باشید لذا با پیغام خطای مبنی بر ناتوانی کامپایلر در باز کردن کتابخانه mega۳۲ (خطای دوم) مواجه خواهید شد . که اگر نام کتابخانه ای را وارد کرده باشید که محیط CodeVision آن را نداشته باشد نیز با این خطا مواجه خواهید شد . همچنین ممکن است کامپایلر به شما به ازای تک تک ثبات ها خطایی بدهد و بگوید که آنها را نمی شناسد و شما همه دستورات را درست نوشته باشید , در این مواقع نیز باید مطمئن شوید که فایل کتابخانه را به درستی به کامپایلر معرفی کرده باشید . (در بخش ۵ یک وجه دیگر این مورد بررسی شده است .) علاوه بر این ممکن است در ابتدای نصب برنامه , همه چیز را درست وارد کرده باشید و باز پیغام خطایی (به فرم و شکلی غیر از دو مورد در سمت راست) مبنی بر اشکال در کتابخانه ها و یا include ها ببنید , در این صورت باید مطمئن شوید که برنامه CodeVision شما درست نصب شده باشد , برای این منظور همیشه سعی کنید یک پروژه سالم را در این مواقع همراه خود داشته باشید . منظور از پروژه سالم پروژه ای است که در یک سیستم دیگر قبلا کاملا تست شده است و هیچ خطای کامپایلی نداشته است و سپس در این سیستم جدید بعد از نصب نرم افزار آن را اجرا کنید و دوباره کامپایل کنید . اگر کامپایلر نتوانست این پروژه سالم را کامپایل کند , معلوم می شود اشکال از نصب نرم افزار است و باید دوباره آن را به صورت صحیح نصب کنید (طریقه نصب نرم افزار خارج از بحث این جزوه می باشد .)
۲) ';' expected	همانطوری که از نوع خطا معلوم است , معمولا این خطا زمانی رخ می دهد که در آخر دستورات زبان C , نقطه-ویرگول را فراموش کرده باشید . دقت کنید که با کلیک بر روی خطا , نوعا محل اولین دستور بعدی نشان داده می شود , که یعنی از انتهای دستور قبلی تا دستور بعدی ,کامپایلر نقطه-ویرگول پیدا نکرده است . اگر با بررسی برنامه باز این خطا رفع نشد باید بدانید که گاهی اشتباه نوشتن بعضی دستورات و یا اشتباه کردن در تعریف توابع (مثلا در یک تابع آکولاد کم یا زیاد گذاشته بشود و ...) نیز گاهی منجر به این خطا می شود که در این صورت خطای نشان داده شده , با خطای در برنامه تفاوت دارد , یعنی در این حالات باید دقت کنید که برنامه شما نقط-ویرگول کم ندارد , بلکه دستورات را در جایی دیگر اشتباه کرده اید . پس بهتر است برنامه را با دقت بیشتری بررسی کنید . مثلا ممکن است دستوری شبیه دستور زیر بنویسید :

	unsigned char i k; اما باید بدانید که دقیقا این دستور این خطا را ایجاد می کند , و این به این دلیل است که شما بین i و k فاصله گذاشته اید و در زبان C در نام هیچ متغییر و یا تابعی نمی توان فاصله گذاشت ,یعنی اگر بخواهیم متغییر با نام my var بسازید نمی توانید به این صورت آن را تعریف کنید بلکه باید آن را به صورتهای myVar یا my_var یا شبیه اینها بنویسید . همچنین دقت کنید که در زبان C علامت line نیز مانند فاصله نمی توانید استفاده کنید .یعنی مثلا به جای my_var نمی توانید بنویسید my-var .در این صورت ممکن است کامپایلر خطاهایی به شما نشان دهد که هیچ به این موضوع مربوط نمی شود .پس شکل درست دستور بالا به صورت زیر خواهد بود : unsigned char i k;
۳) must declare first in block	در تعریف متغییر ها به صورت محلی (یعنی متغییر هایی که درون توابع یا درون حلقه ها تعریف می شوند , مانند تعریف متغییر ها در تابع اصلی void main(void) که تنها در تابع اصلی می توان به این متغییر ها دسترسی پیدا کرد .) اگر نیاز به تعریف متغییری باشد , حتما باید بعد از باز کردن آکولاد مربوط به تابع (یا حلقه یا ...) متغییر ها را تعریف کنید . که ترجمه خطای نشان داده شده نیز همان را می گوید که در هر بلاک باید اولین باشد و هنگامی که روی خطا کلیک کنید , شما را به محل تعریف متغییرها می برد و به زبان ساده بین دو آکولاد را یک بلاک گویند . پس در حقیقت یک تابع که یک آکولاد باز و بسته دارد , یک حلقه for یا while یا یک دستور شرطی if که دارای آکولاد باز و بسته هستند , همه یک بلاک حساب می شوند و در هر کدام از اینها اگر بخواهیم یک متغییر تعریف کنیم , باید در ابتدای بلاک به تعریف آن بپردازیم , یعنی دقیقا اولین دستوری که بعد از باز شدن آکولاد آن بخش می باشد , تعریف متغییر ها باشد. **نکته :** در تعریف متغییر ها به صورت global (یعنی متغییر هایی که خارج از توابع تعریف می شوند) چون خاصیت این متغییر ها وابسته نبودن آن ها به یک بلاک خاص می باشد , لذا تعریف آنها در ابتدا یا انتهای یک بلاک معنی ندارد , چرا که اصلا آنها داخل هیچ بلاکی تعریف نمی شوند .
۴) undefined symbol '0xE0' undefined symbol 'I' undefined symbol 'Lcd_init'	این خطاها معمولا به دلیل اشتباه نوشتن توابع یا متغییرها و یا دستورات دیگر می باشد . **۱)** یک خطایی که در بعضی موارد ممکن است برای برنامه نویس پیش بیاید و در رفع آن به مشکل بخورد خطایی شبیه خطای اول است , فرض کنید عدد 0xE0 را بخواهید در یک ثبات قرار دهید و کامپایلر خطای اول را به شما نشان دهد, فکر میکنید خطای کار شما در چیست ؟ به ظاهر همه چیز درست است و هیچ مشکلی نباید باشد , اما اگر دقت بیشتری بکنید ,خواهید دید که در این دستور به جای صفر , از حرف O (حرف او) استفاده کرده اید . چرا که حرف او (O) با صفر (۰) شباهت بسیار زیادی دارند و تمایز این دو گاهی می شود گفت غیر ممکن است . لذا در این موارد اگر با این خطا برخورد کردید , حتما بررسی کنید که عدد صفر را درست وارد کرده باشید . **۲)** خطای دوم در اثر اشتباه نوشتن و استفاده کردن متغییر I رخ داده است . در حقیقت در برنامه اگر متغییر را به صورت i تعریف کنید و بخواهید از آن به صورت I استفاده کنید , با این خطا مواجه می شوید . **۳)** اگر به این دستور نیز کمی دقت داشته باشید , کامپایلر می گویدکه علامت Lcd_init را نمی شناسد , و حرف کاملا درستی است , چرا که قبلا گفته شده است که همه دستورات زبان C با حروف کوچک نوشته می شوند . لذا بزرگ نوشتن حرف اول Lcd باعث می شود که کامپایلر این دستور را به طور کلی نشناسد . و همچنین در صورت برخورد با خطاهای اینچنینی , اولین موردی که بررسی می کنید درستی دستورات به صورت نوشتاری باشد . اگر درستی را بررسی کردید و مطمئن شدید که هیچ خطایی را در تایپ مرتکب نشده اید و باز با این خطا روبرو شدید , به خطای بعدی مراجعه کنید .
۵)	در اصل یک برنامه نویس باید بداند که هر تابع و دستوری در کدام کتابخانه وجود دارد . تفاوت

undefined symbol 'TIM۲_COMP' undefined symbol 'PORTB' undefined symbol 'ADCSRA' undefined symbol 'ADC_INT' undefined symbol 'DDRA' undefined symbol 'DDRB' undefined symbol 'DDRC' undefined symbol 'ADMUX' undefined symbol 'ADCSRA' undefined symbol 'TCCR۲' undefined symbol 'OCR۲' undefined symbol 'TIMSK' undefined symbol 'ADCH' undefined symbol 'ADCH' undefined symbol 'ADCH' ...(Error in mega۳۲.h) یا undefined symbol 'lcd_init' undefined symbol 'lcd_gotoxy' undefined symbol 'lcd_gotoxy' .... (Error in lcd.h)	خطاهای در این بخش با بخش قبلی در این است که اینجا خطای شما مربوط به اشتباه تایپ کردن دستورات نیست و علاوه بر آن یک خطا نیز دریافت نخواهید کرد . یعنی مثلا با خطاهایی شبیه چند خطای اول (ممکن است شما از ثباتهای دیگری در برنامه خود استفاده کرده باشید و دقیقا این خطاها را مشاهده نکنید) برخورد کنید . که اولین چیزی که شاید به ذهن متبادر بشود , این است که احتمالا کامپایلر در پیدا کردن کتابخانه مربوطه به مشکل خورده است . و در این موارد حتما نگاه کنید که همه کتابخانه ها را وارد کرده باشید . حتی ممکن است نام کتابخانه ای را به اشتباه وارد کرده باشید , و با خطای بخش اول (در ابتدای جدول) نیز مواجه نشوید . همچنین مشاهده می کنید که ممکن است چندین بار یک خطا نشان داده شود که این نیز مربوط به چند بار استفاده کردن از آن دستور است . (خطاهای بخش دوم مربوط به اشتباه وارد کردن و یا اصلا وارد نکردن کتابخانه lcd.h می باشد .) در خطاهای در این بخش , خطاهای اول که مربوط به ثباتهای AVR می باشند , معلوم می شود که کامپایلر کتابخانه mega۳۲.h (یا هر AVR مورد استفاده) را پیدا نکرده است . خطاهای بعدی که مربوط به صفحه نمایش می باشند , در کتابخانه lcd.h موجود اند . توابع itoa و atoi و شبیه اینها در کتابخانه stdlib.h می باشند و تابع sprint در کتابخانه stdio.h می باشد . توابع ریاضی همچون sin , cos و ... در کتابخانه math.h می باشند . و به همین صورت کتابخانه های بسیار دیگر .
خطای در دستورات اسمبلی	 خطایی به صورت مقابل معمولا نشان دهنده این است که دستورات اسمبلی را یا وارد نکرده اید یا اشتباه وارده کرده اید (مثلا در تعریف LCD بجای equ. نقطه ابتدای آن را اشتباها نگذاشته اید یا مثلا دستور #asm("sei") را اشتباها طور دیگری نوشته اید. و به همین صورت, دیگر دستورات اسمبلی که معمولا استفاده زیادی دارند .

نکته ۱ : دقت کنید که ممکن است هیچ کدام از راههای گفته شده برای رفع اشکال خطاها در بعضی موارد کمکی به رفع خطای برنامه شما نکند . در این موراد گاها پیش می آیدکه خطای کامپایلر حتی گمراه کننده نیز هست . و برای رفع خطا باید نکات گفته شده در برنامه نویسی را به دقت مراعات کنید . (مثلا هرگز علامت underline (_) را با line (-) یا همان منها اشتباه نکنید , در توابع و دستورات هرگز فاصله و line وجود ندارد یا نقطه ویرگولها و تعداد آکولادها و ...). جدول زیر چند نمونه از استناها را که قبلا توضیحات در باب آنها ارائه شده است را نشان می دهد.

خطای کامپایلر	علت خطا
';' expected	در یک حالت خاص مثلا کامپایلر به دستور زیر خطای مقابل را می دهد : PORTD=PORTC * i _۵; که اگر دقت شود به جای منها در عبارات محاسباتی ریاضی از underline استفاده شده است و در

نتیجه کامپایلر بعد از underline به دنبال (;) می گردد و آن را پیدا نمی کند و این خطا را می دهد . عبارت درست به صورت زیر است . PORTD=PORTC * i -۵;	
همچنین خطای روبرو به دستور زیر داده شده است : lcd-init(۱۶); که همانطوری که دیده می شود به جای نوشتن underline از line در تابع استفاده شده است . یعنی به جای نوشتن ;(۱۶)lcd-init باید نوشته شود ;(۱۶)lcd_init . در اینجا چون عبارت منها وارد شده است , و این عبارت دو عملوند می گیرد , لذا کامپایلر به دنبال دستور یا متغییری با نام lcd می گردد و چون آن را پیدا نمی کند این خطا را نشان می دهد . یعنی اگر در برنامه شما متغییری با نام lcd وجود داشته باشد دیگر کامپایلر پیغام خطای زیر را می دهد : undefined symbol 'init' که همان اشکال در مورد متغییر init پیدا می شود که می گوید آن را به عنوان عملوند نمی تواند پیدا کند اما lcd را که در برنامه فرضی ما یک متغییر است پیدا کرده است و دیگر از آن خطایی نمی گیرد و این در حالی است که به طور کلی lcd که در اینجا مد نظر گرفته شده است با تابع lcd_init(۱۶) متفاوت است . لذا اگر طوری بشود که این دستور اجرا نیز بشود مسلما آن کاری را که مد نظر برنامه نویس بوده است , انجام نخواهد داد . این اشکالات در بحثهای بعدی دنبال خواهند شد ان شاالله تعالی .	**undefined symbol 'lcd'**

نکته ۲: دقت کنید که می شود گفت در همه موارد کامپایلر از بالای برنامه شروع به کامپایل می کند تا به پایین برنامه برسد . پس اگر چندین خطا مشاهده کردید , همیشه سعی کنید اول خطایی را که بالاتر است برطرف کنید و سپس به سراغ بقیه خطاها بروید .وهمچنین گاهی (مانند مورد دوم در جدول بالا) کامپایلر تنها به اولین خطایی که می رسد بقیه برنامه را بررسی نمی کند و تنها همان یک خطا را نشان می دهد , لذا ممکن است با برطرف کردن آن خطا خطاهای بیشتری به شما نشان داده بشوند و این دلیل بر این نیست که دستور قبلی درست بوده است و شما نباید آن را تغییری بدهید .

توصیه هایی برای بهتر برنامه نوشتن :

۱. قبل از شروع برنامه نویسی, در ذهن خود به طور کلی یک الگوریتمی از روند انجام برنامه تصور کنید که برنامه شما قرار است دقیقا چه کاری انجام دهد و بعد الگوریتم تصور شده در ذهن را باید در قالب برنامه نویسی پیاده کنید . به عنوان مثال در کار با DotMatrix ها در مرحله تصور یک الگوریتم , باید بدانیم که چطور یک حرف را در DotMarix نشان می دهند که مثلا در این مورد بخصوص باید ستون به ستون صفحه DotMatrix را مطابق آن حرفی که می خواهیم چاپ بشود , با سرعت زیاد روشن کنیم تا آن حرف مورد نظر ما در صفحه نشان داده شود . و بعد با این تصور از ساختار کلی روند کار , سعی می کنیم همین ها را با برنامه نویسی میکروکنترلر در عمل پیاده سازی کنیم .

۲. اگر چه در محیط CodeWizard کاربر می تواند در هنگام ساخت پروژه جدید , بسیاری از تنظیمات را به راحتی انجام دهدو از محاسبات دوری کند , اما معمولا برای تغییر پروژه ها بعد از ساخت بخصوص در بعضی موارد یا راههای طولانی وجود دارد , یا راههای آن را کمتر کسی می داند , به طوری که می شود گفت برای تغییر آنها راهی نیست . علاوه بر آن بهتر است همه تنظیمات میکرو را برنامه نویس به صورت دستی انجام دهد تا علاوه بر اطلاع دقیق از ثبات ها و تنظیمات آن , تسلط کافی بر برنامه نویسی و مقدار دهی ثبات ها پیدا کند . چرا که در شرایط بسیاری لازم می آید برنامه نویس بر طبق نوع مسئله مقدار دهی های برنامه را تغییر دهد . در حالی که این مقدار دهی ها در تنظیمات CodeWizard تنها مقدار دهی های اولیه می باشند لذا به عنوان یک تمرین خیلی خوب ,

همیشه سعی کنید بعد از ساختن پروژه تمام محتویات داخل فایل را پاک کنیدو برنامه را از صفر شروع به نوشتن کنید. و تنها در محیط CodeWizard فرکانس کاری میکروکنترلر و همچنین در صورت نیاز به نام بردار وقفه ها , وقفه ها را فعال کنید .

۳. همیشه سعی کنید برنامه را مرتب بنویسید تا در صورت بروز مشکل بتوانید راحت تر آن را عیب یابی کنید و همچنین در صورت نیاز به تغییر برنامه , بتوانید بدون هیچ سختی آن را تغییر دهید , به عنوان مثال به دو تصویر زیر دقت کنید :

```
while (1)
{
for(i ; i<=9 ;)
{
PORTA = x[i];
for (j ;j<=9 ;)
{
PORTB = x[j] ;
delay_ms(75);
if (PINC & 0x01)
{
if (j==1)
{
PORTB=x[j];
j=9;
break;
}
else
{
if (j%2 & j!=0 )
{
j-=2;
}
else if (j==0)
{
j=9;
delay_ms(75);
break;
}
else
{
j--;
}
}
}
```

```
while (1)
{
for(i ; i<=9 ;)
{
PORTA = x[i];
for (j ;j<=9 ;)
{
PORTB = x[j] ;
delay_ms(75);
if (PINC & 0x01)
{
if (j==1)
{
PORTB=x[j];
j=9;
break;
}
else
{
if (j%2 & j!=0 )
{
j-=2;
}
else if (j==0)
{
j=9;
delay_ms(75);
break;
}
else
{
j--;
}
}
}
```

هر دو این تصاویر از لحاظ دستوری با هم برابری می کنند و هر کدام از این دستورات یک کار را انجام می دهند , اما همانطوری که به وضوح دیده می شود دستوراتی که به صورت شکل سمت چپ نوشته شده اند , بسیار منظم تر و راحت تر برای خواندن و فهمیدن و در نتیجه عیب یابی هستند به طوری که حتی یک کم گذاشتن آکولاد در آن به وضوح دیده می شود و دقیقا بالعکس , در طرف سمت چپ اگر مثلا یک آکولاد کم گذاشته شده باشد پیدا کردن آن بسیار دشوار خواهد بود . به طور کلی در محیطهای برنامه نویسی حرفه ای مانند Atmel Studio یا Visual Studio در حین برنامه نویسی , خود محیط برنامه سعی می کند این شکل از برنامه نویسی را رعایت کند و مثلا بعد از یک آکولاد دستور حلقه for اگر کلید Enter را بزنید , برنامه خود به خود به خط بعدی , اما با فاصله ای بیشتر از خط قبلی نسبت به ابتدای سطر می رود تا معلوم بشود که دستوراتی که از اینجا به بعد نوشته می شوند مربوط به حلقه for می باشند , اما در محیطهای برنامه نویسی مانند کدویژن هنوز از این قابلیت پشتیبانی نمی شود و لذا خود برنامه نویس باید به این طور نوشتن عادت کند . در واقع روش کلی نوشتن به این صورت این است که برای دستوراتی مانند – for , while , if-else ,..... switch-case بعد از نوشتن دستور و رفتن به خط بعدی با یک بار زدن دکمه Tab که باعث می شود مکان نما کمی جلوتر

برود , شروع به برنامه نویسی کنیم . دقت کنید که آکولادهای این دستورات را دقیقا زیر دستورات باز و بسته می کنیم و تنها دستورات را با فاصله می نویسیم . مانند آن چیزی که بالا در شکل سمت چپ می بینید . که همانطوری که دیده می شود , در این صورت خود برنامه کدویژن , ابتدا و انتهای آکولادها را نیز نشان می دهد .

۴. همیشه سعی کنید که اگر پرانتز یا آکولادی را باز می کنید , همان جا آن را ببندید و بعد دستورات را بین آن بنویسید , چراکه بسیار راحت می شود که پرانتز یا آکولادی را باز کنید و تا زمانی که به انتهای آن و تکمیل آن برسید فراموش کنید که آن را ببندید و بعد با خطای در کامپایل برنامه مواجه شوید . و همیشه پیدا کردن یک آکولاد به سادگی ترجمه خطای کامپایل شده نیست .

۵. همیشه الزاما محل خطا و نوع خطا آنی نیست که بعد از کامپایل به شما نشان داده می شود , ممکن است با کم یا زیاد گذاشتن یک آکولاد و یا پرانتز ,کامپایلر خطاهای زیادی از برنامه و دستورات شما بگیرد , در حالی که آن دستورات هیچ اشکالی نیز نداشته باشند ,و حتی محلی که برنامه شما اشکال دارد یا نوع اشکال را نیز به درستی به شما نشان ندهد , لذا اهمیت نکته شماره ۵ در اینجا خودش را نشان می دهد که خود را عادت دهیم که با باز کردن هر آکولاد یا پرانتز , آن را ببندیم و بعد دستورات را داخل آن بنویسیم , چرا که در صورت اشتباه , بعضی اوقات پیدا کردن محل اشتباه و نوع اشتباه بسیار دشوار خواهد بود .

۶. سعی کنید در صورت برخورد با Error در کامپایل برنامه ,ابتدا به دنبال پیدا کردن ساده ترین اشتباهات همانند کم بودند تعداد پرانتزها و آکولادها و یا نگذاشتن سمی کالون (;) در انتهای دستورات باشید . چرا که زیاد پیش می آید که در برنامه نویسی , اشتباهات بسیار ساده , خطاهای بزرگ و البته نامربوط به نوع اشتباه پیش می آورد .

۷. ممکن است در برنامه نویسی خود , با وجود درست بودن دستورات و کامپایل شدن برنامه ببینید که برنامه شما در عمل آن خروجی که می خواهید را به شما نمی دهد . در این موارد سعی کنید از فاکتورهایی کنترلی برای تشخیص محل خطا استفاده کنید . به عنوان مثال اگر یک شمارنده دارید که قرار است از یک تا ۱۰ بشمارد و بر روی LCD نمایش دهد , اما برنامه فقط تا عدد ۵ می شمارد , می توانید عدد خود آن شمارنده را به یک پورت AVR بفرستید (مثلا شمارنده i می باشد) مانند :

```
PORTA =i;
```

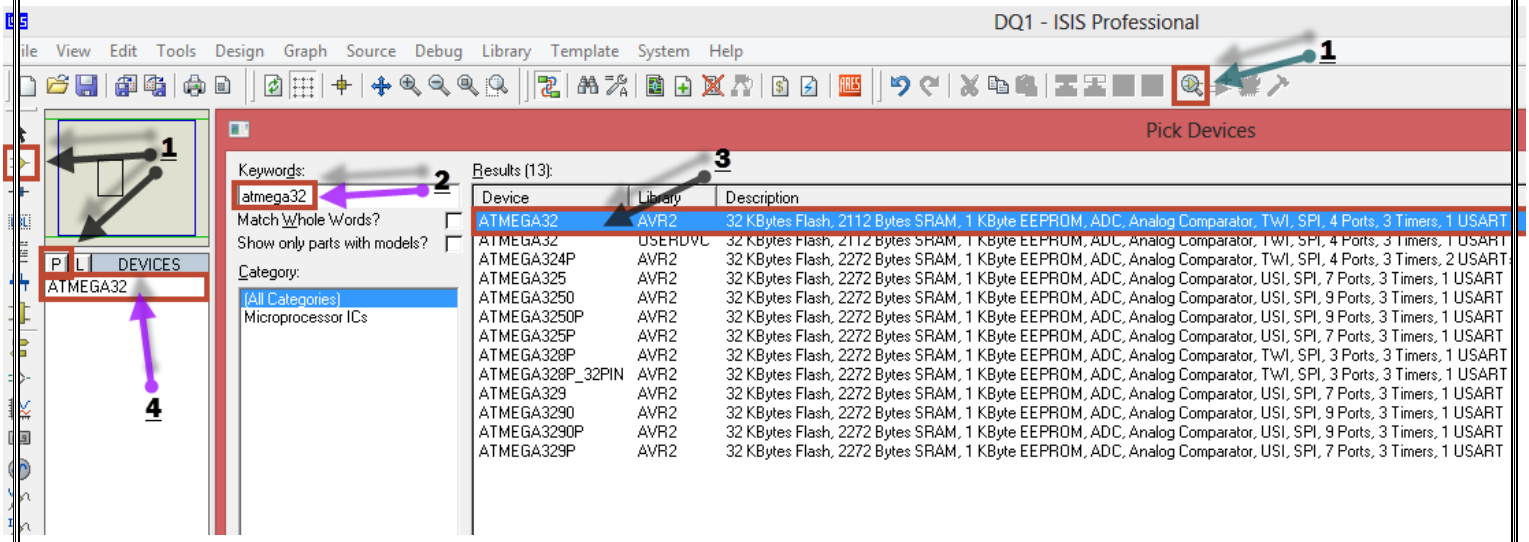
و بعد مثلا بفهمیم که آیا اصلا عدد شمارشگر ما از مقدار ۵ بیشتر می شود یا مشکل از جای دیگر است , اگر بیشتر از ۵ شد , معلوم می شود که لااقل شمارشگر ما درست می شمارد و مشکل در بخشهای دیگری است که باید برویم آنها را نگاه کنیم و باز با قرار دادن فاکتورهای کنترلی دیگری مطابق با آن دستورات , تشخیص بدهیم که این عیب از کجا می آید و چطوری آن را حل کنیم . و یا حتی مثلا در کار با DotMatrix ممکن است وقتی می خواهیم یک حرف را نشان بدهیم , شکلی دیگر به ما نشان بدهد که مثلا می توانیم با قرار دادن تاخیر در بین دستوراتی که ستون ها و سطرها را تغییر می دهند مرحله به مرحله ببینیم که خروجی ما چگونه است که مثلا ممکن است یک سطر یا ستون را اشتباهی نشان دهد که می فهمیم اشکال ممکن است در آرایه هایی که تعریف کرده ایم باشد . به طور کلی شاید بشود گفت در برنامه هایی که دارای شمارنده های متفاوت است و با مشکل در شمارش برخورد کرده ایم که یک عدد خاص را نمی شمارد و ... بهتر است مقدار شمارشگر را مرحله به مرحله به یکی از پورتهای خروجی بفرستیم تا مقدار لحظه ای آن را داشته باشیم (دقت کنید که ممکن است لازم باشد برای اشکال زدایی بعضی برنامه ها , بعضی دستورات برنامه را تغییر دهید) . و برای برنامه هایی مانند کار با صفحه کلید و یا DotMatrix که برنامه با سرعت بالا اجرا می شود استفاده از دستورات delay بسیار ضروری می باشد . اما باز هم تاکید می شود که همیشه الزاما اینطور نیست و ممکن است در خیلی موارد نتوانید از فاکتورهای کنترلی استفاده کنید و عیب یابی برنامه و قرار دادن فاکتورهایی برای شناسایی برنامه به نوع برنامه و نوع سخت افزاری که قرار است با آن کار کند , مربوط می شود .

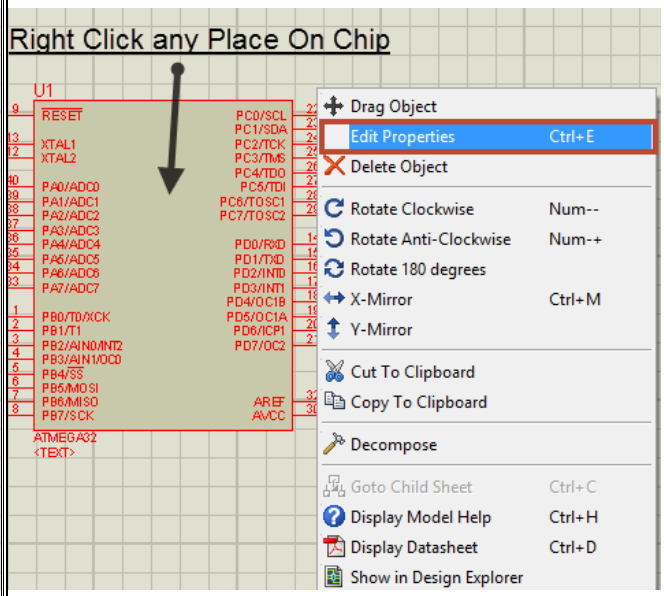
شبیه سازی در نرم افزار پروتئوس (Proteus) :

پس از اتمام کار برنامه نویسی خود بهتر است قبل از پیاده سازی , نتایج را در شبیه سازی ها مشاهده کنید .برای این منظور شاید یکی از بهترین محیطها فعلا محیط نرم افزار پروتئوس می باشد . چون این نرم افزار یکی از نرم افزار های پرکاربرد برای شبیه سازی مدارات است , لذا فرض را بر این می گیریم که دانشجو با کلیت محیط این نرم افزار آشنایی لازم را دارد و تنها به برخی موارد مربوط به شبیه سازی AVR در این محیط از جمله تنظیم فرکانس میکروکنترلر , بارگذاری فایل , اجرای برنامه به صورت Step by Step و ... می پردازیم .

قرار دادن Atmega۳۲ در محیط نرم افزار از کتابخانه :

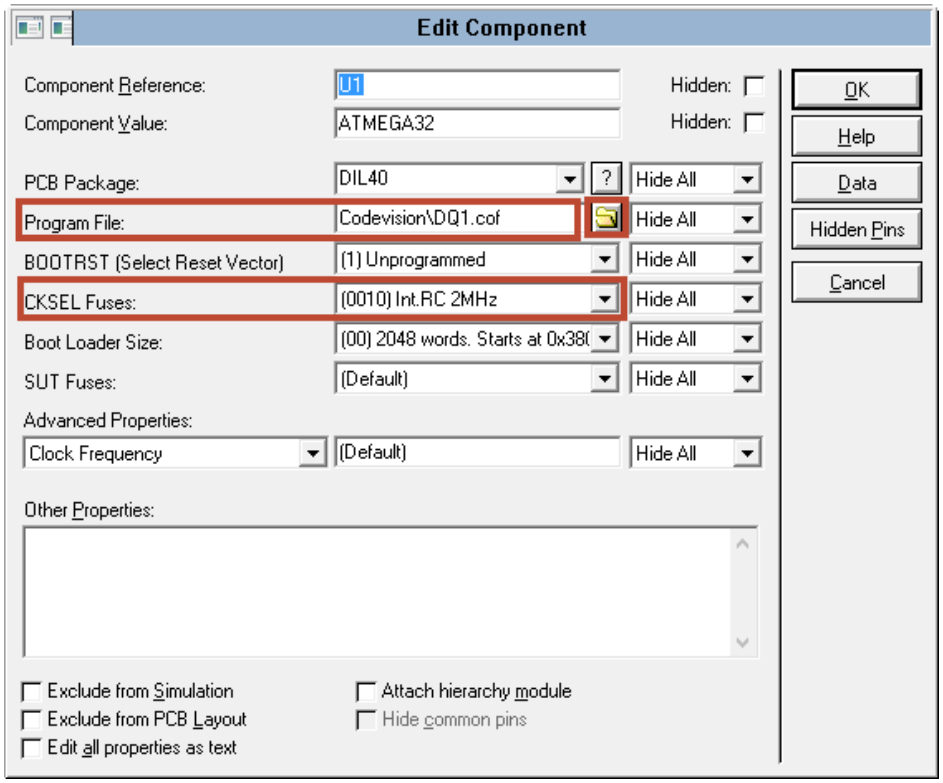
برای باز کردن کتابخانه قطعات , می توان از یکی از راههای نشان داده شده در شکل - از طریق آیکون در نوار ابزار بالا یا نوار ابزار سمت چپ محیط پروتئوس (با دو مرحله) - که با نام شماره یک نامگذاری شده است , استفاده کرد . پس از باز شدن پنجره (**Pick Devices**) با نوشتن عبارت Atmega۳۲ (شماره ۲ در شکل) لیستی از میکروکنترلرهایی که دارای عبارت Atmega۳۲ در نامگذاری خود می باشند , در سمت راست این پنجره نشان داده می شوند . معمولا اولین گزینه , همان Atmega۳۲ می باشد که تمام برنامه های ما برای آن نوشته می شوند که با دو بار کلیک بر روی آن (شماره ۳ در شکل) در بخش قطعات (شماره ۴ در شکل) می توان آن را مشاهده کرد و در محیط شبیه سازی قرار داد.





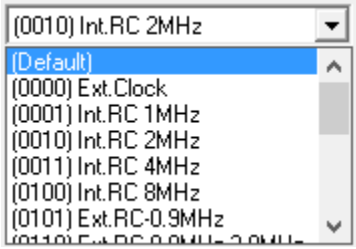
تنظیمات AVR :

پس از انجام مراحل بالا, اکنون دیگر می توانید پنجره کتابخانه قطعات را بسته و Atmega۳۲ را در محیط شبیه سازی قرار دهید . حال نیاز دارید تا میکروکنترلر خود را به دلخواه تنظیم کنید و فایل برنامه نویسی شده در آن را بارگزاری (Program) کنید . به این منظور بر روی میکروکنترلر راست کلیک کرده و گزینه دوم (ممکن است در بعضی ورژنهای پروتئوس گزینه دوم نباشد) یعنی Edit Properties را انتخاب کنید . پنجره جدیدی که باز می شود , محل تنظیمات آی سی میکروکنترلر شما می باشد که در ادامه به توضیح برخی از بخش های آن می پردازیم .



تنها بخشهایی که در آزمایشگاه به آنها نیاز پیدا خواهید کرد دو بخش Program File و CKSEL Fuses می باشد که درشکل نشان داده شده اند . بخش Program File برای دادن آدرس برنامه ای است که نوشته و کامپایل کرده اید و بخش CKSEL Fuses برای انتخاب فیوز بیتهای کلاک یا به زبان ساده تر انتخاب فرکانس میکروکنترلر می باشد .

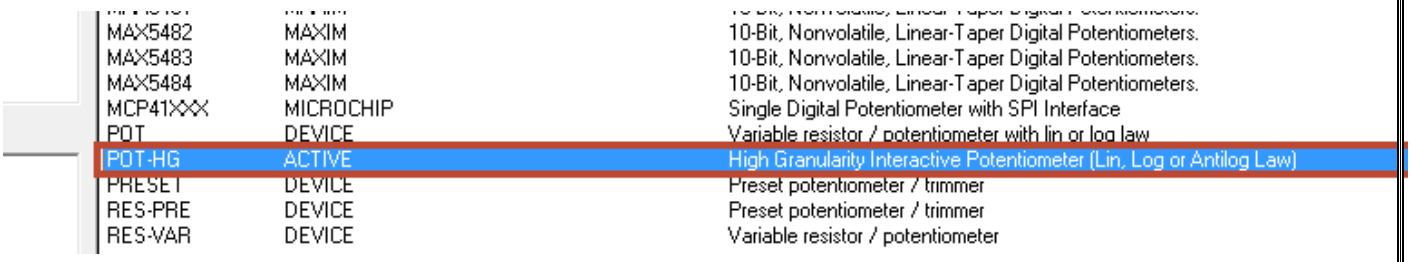
در کنار بخش Program File یک آیکون "پوشه" دیده می شود که با کلیک کردن بر روی آن در پنجره باز شده , باید آدرس فایل کامپایل شده در محیط کدویژن را (که معمولا فایلی هم نام با پروژه و با پسوند *.cof می باشد) وارد نمایید .



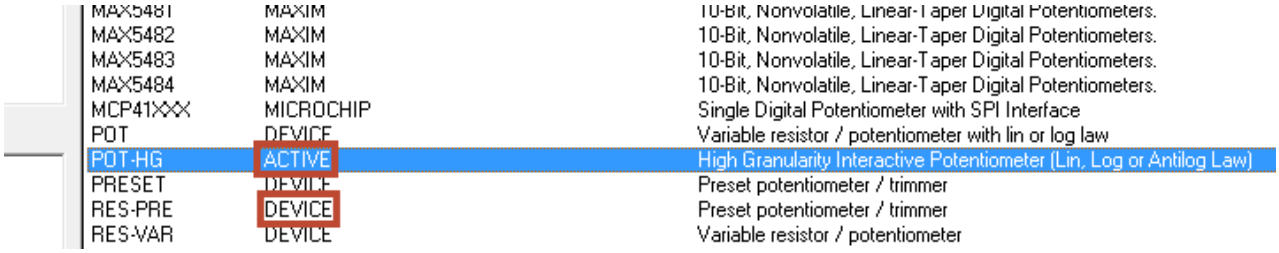
در بخش CKSEL Fuses با باز کردن منوی کرکره ای شما گزینه هایی بصورت مقابل را مشاهده خواهید کرد که در آزمایشگاه شما تنها کافیست از فرکانسهای ۱،۲،۴ و ۸ مگاهرتز استفاده کنید و به بقیه موارد نیازی ندارید و اگر بخواهید آنها را انتخاب کنید باید کریستال مورد نظر را نیز در محیط پروتئوس برای میکروکنترلر خود قرار دهید .

قرار دادن بقیه قطعات مورد نیاز در محیط پروتئوس :

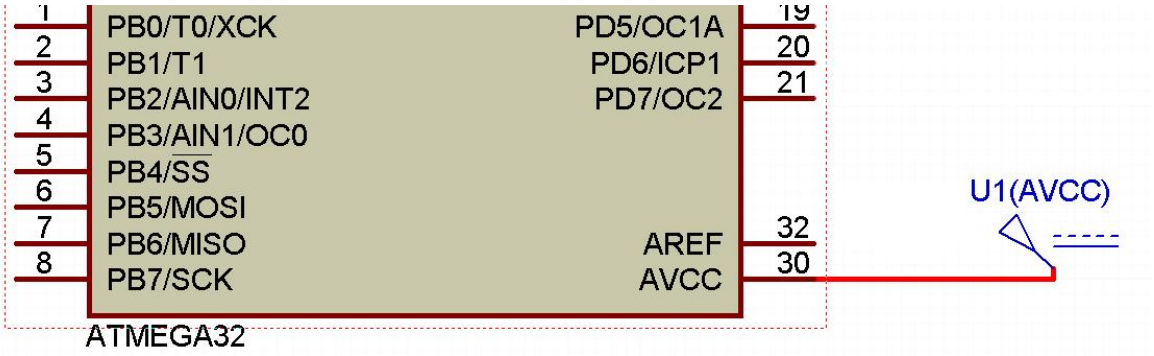
علاوه بر میکروکنترلر شما به مواردی همچون پتانسیومتر و ... در محیط شبیه سازی نیازمند هستید , برای یافتن این قطعات کافیست عبارت کلیدی را که باید جستجو بکنید بدانید , به عنوان مثال برای قرار دادن پتانسیومتر کافیست عبارت POT را تایپ کنید و از لیست همانند شکل زیر عبارت POT-HG را انتخاب کنید .



نکته ۱ : در کنار قطعات گاهی عبارت DEVICE و گاهی عبارت ACTIVE و عباراتی شبیه اینها نوشته می شود . منظور از عبارت ACTIVE این است که این قطعه در حین شبیه سازی - یعنی زمانی که بر روی دکمه  کلیک می کنید و این دکمه به رنگ سبز در می آید – می تواند تغییر کند , یعنی مثلا اگر پتانسیومتر شما ACTIVE باشد (همانند بالا که در کنار قطعه POT-HG عبارت ACTIVE نوشته شده است) می توانید در حین شبیه سازی مقدار پتانسیومتر را تغییر دهید . در غیر این صورت اگر از قطعاتی که ACTIVE نیستند استفاده کنید , در حین شبیه سازی نمی توانید مقدار آن را تغییر دهید و لذا عملا یک پتانسیومتر با مقاومت معمولی برایتان فرقی نخواهد داشت .

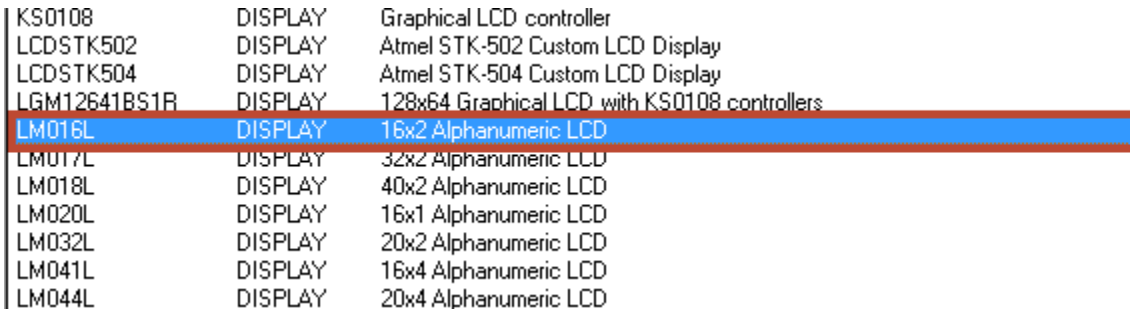


نکته ۲ : معمولا از پتانسیومتر در هنگام کار با AVR برای ADC استفاده می شود , همیشه برای کار با ADC در محیط پروتئوس باید به خاطر داشته باشید که پایه AVCC را به یک منبع ولتاژ ۵ ولتی متصل کرده باشید . یعنی همیشه میکروکنترلر شما در کار با ADC به صورت زیر باشد:

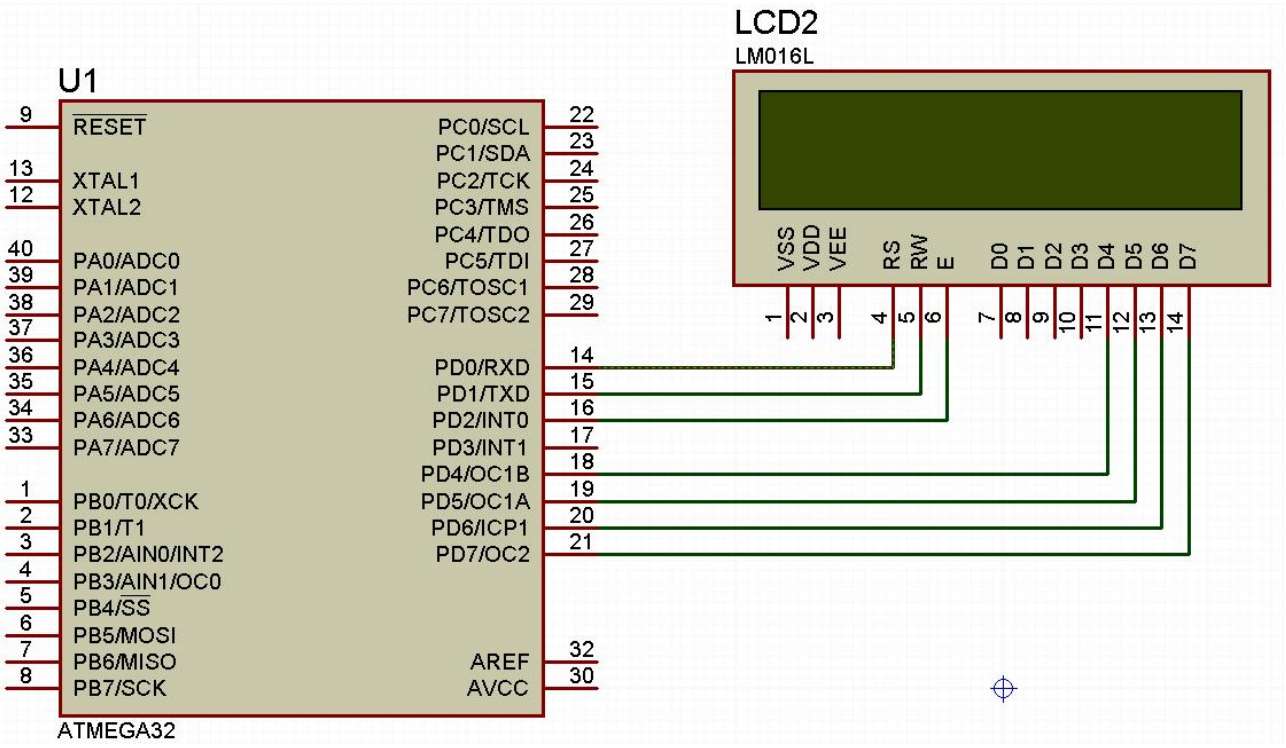


LCD :

برای قرار دادن LCD همین عبارت را وارد می کنید و در لیست سمت راست گزینه نشان داده شده در شکل را انتخاب می کنید . دلیل این انتخاب نیز این است که اگر در بخش توضیحات این قطعه در جلوی آن نگاه کنید نوشته است : Alphanumeric LCD ۱۶*۲ که یعنی یک LCD با ۱۶ ستون و دو سطر است که همان LCD است که ما در محیط کدویژن برای آن برنامه نویسی کرده ایم .



برای نحوه سیم بندی LCD نیز اگر تنظیمات پیش فرض را تغییر ندهید کافیست همانند شکل عمل کنید که مثلا فرض کنید LCD شما به پورت D قرار است متصل باشد .



نکته : دقت کنید که پایه PORTD.۳ خالی است و در حقیقت همیشه پین ۳ از هر پورت دراتصال با LCD باید خالی باشد .

دستور کارآزمایشگاه:

آزمایشهایی که در این دستور کار طراحی گردیده اند بسته به امکانات موجود در آزمایشگاه به طرق مختلف می توانند پیاده سازی و اجرا گردند. تعدادی از آنها توسط محیط شبیه ساز AVR studio به راحتی قابل تست و اجرا بوده اما در برخی از آزمایشات که نیاز به استفاده از نمایشگر هفت قسمتی و یا کلیدهای فشاری ویا سنسور دما و..... می باشد می توان از محیط شبیه ساز نرم افزار Proteusبرای این منظور استفاده نمود. اما پیشنهاد می گردد کلیه ازمایشات علاوه بر اینکه در یک محیط شبیه ساز تست واجرا میشوند توسط پرو گرامر در میکرو کنترل AVR نیز کپی گردیده و عملا برروی برد بورد و یا در صورت وجود ترینر در آزمایشگاه مورد تست عملی قرار گیرند.

۱)نمایشگر LED

الف) یک LED را به پایه Portb.۰ متصل نموده سپس برنامه ای بنویسید که LED را بافواصل زمانی ۱ثانیه روشن وخاموش نماید.

ب) ۸عدد LED را به پورت B متصل نموده سپس برنامه ای بنویسید که LED ها را یک در میان با فواصل زمانی ۱ ثانیه روشن وخاموش نماید.

ج) برنامه قسمت ب را بگونه ا ی تغییر دهید که LEDها بصورت متقارن ،دوتا دو تا از وسط روشن شده و به دو طرف حرکت نموده واین کار تکرار گردد.

د) برای ۸عددLED متصل شده به پورتBیک برانامه رقص نور با حداقل ۳۰حالت غیر تکراری به سلیقه خودتان بنویسید.

۲)نمایشگر هفت قسمتی(۷SEG) و کلید فشاری

الف) یک نمایشگر هفت قسمتی کاتد مشترک را از طریق تراشه ۷۴۴۸(مبدل BCD به هفت قسمتی)به ۴بیت کم ارزش پورت A متصل نموده سپس برنامه ای بنویسید که یک شمارش ۰تا۹ ثانیه برروی ۷seg مشاهده گردد.

ب) همان آزمایش الف را بدون استفاده از تراشه ۷۴۴۸تکرار نمایید .در این حالت تمامی پایه های پورت Aمورد استفاده قرار میگیرند.

ج) دو عدد نمایشگر هفت قسمتی کاتد مشترک را به AVR متصل نموده و یک شمارنده دو رقمی ۰۰تا۹۹ با فواصل زمانی یک ثانیه طراحی نمایید.

د) می خواهیم نحوه شمارش شمارنده دو رقمی آزمایش ج توسط یک کلید فشاری که به پایه PortC.۰ متصل نموده ایم کنترل گردد بگونه ای که در مدتیکه کلید رها می باشد شمارش بصورت افزایشی اما در مورد فشرده بودن کلید شمارش بصورت کاهشی انجام گردد.

ه) همان آزمایش د را بنحوی تغییر دهید که شمارنده بصورت زوج یا فرد عمل شمارش را انجام دهد. در مدتیکه کلید رها می باشد شمارش بصورت زوج اما در مدت فشرده بودن کلید شمارش بصورت فرد انجام گیرد. ضمنا هنگام تغییر وضعیت شمارش از زوج به فرد یا بالعکس به نزدیکترین عدد زوج یا فرد برود.

۳)نمایشگر LCD متنی

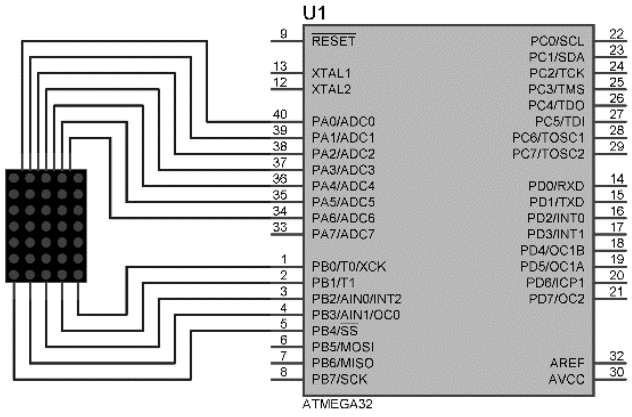
الف) یک LCD ۲*۱۶را به پورت B متصل نموده سپس برنامه ای بنویسید که پیغام HELLO را در وسط سطر اول به نمایش در اورده و بعد از سه ثانیه ان را به سطر دوم منتقل و سطر اول را پاک نماید و اینکار مرتبا تکرار گردد.

ب) برنا مه ای بنویسید که پیغام HELLO را کاراکتر به کاراکتربا فواصل زمانی ۰,۵ثانیه از سمت راست صفحه LCD به تدریج وارد نموده و به سمت چپ حرکت داده تا زمانی که به طور کامل از سمت چپ محو گردد.

ج) چهار عدد کلید فشاری به نامهای a,b,c,d را به پایه های پورت A وصل کرده سپس می خواهیم یک شمارنده سه رقمی برروی LCD با فواصل زمانی ۰,۵ثانیه طراحی نموده بگونه ای که فشردن کلید aشمارش را بصورت زوج و افزایشی و فشردن کلید b شمارش را بصورت فرد و افزایشی و فشردن کلید c شمارش را بصورت زوج و کاهشی و فشردن کلیدdشمارش را بصورت فرد و کاهشی تعیین نماید.ضمنا در ابتدای کار که هنوز کلیدی فشار داده نشده است شمارش بصورت تک وحدی و افزایشی انجام گردد.

۴)نمایشگر ماتریس نقطه ای (Dot matrix)

در این نمایشگر که برای نمایش اعداد و حروف از نقاط بیشتری استفاده میشود کیفیت نمایش و تعداد کاراکتر هایی که قابل نمایش می باشند افزایش می یابد.



اصول کار در نوع کاتد مشترک آن به این صورت است که سطرهایی که یک شوند قابلیت روشن شدن پیدا میکنند و اگر ستون مربوط به انها صفر شود چون مسیر جریان بر قرار میشود روشن میشوند.

برای نمایش یک کاراکتر باید ابتدا سطر اول را فعال کرده ("۱")ولی بقیه سطرها غیر فعال باشند سپس اطلاعات مر بوط به ستون ان را برروی پایه های مربوط به ستون قرار داده و تاخیر کوتاهی ایجاد نموده سپس سطر بعدی را فعال و اطلاعات مربوط به ستون را داده و تاخیر کوتاه و به همین صورت تا انتها ادامه داده ودر یک حلقه این کا را تکرارمی نماییم .

الف) سطرهای نمایشگر را به پورت Aو ستونها را به پورت Bمطابق شکل متصل نموده سپس برنامه ای بنویسید که حرف Aرا برروی ان نمایش دهد.

ب) مجددا برنامه را به گونه ای تغییر دهید که حروف آو ب و پ وترا با فواصل زمانی یک ثانیه به ترتیب برروی نمایشگر فوق نشان داده و این کار را تکرار نماید.

۵)صفحه کلید ماتریسی

الف) یک LCD را به پورت B و یک صفحه کلید ۴*۴را به پورت C متصل نموده سپس برنامه ای بنویسید که در صورت فشردن یک کلید شماره سطر وستونی که کلید در ان قرار گرفته است را برروی LCD نمایش دهد.

ب) ابتدا۱۶عدد کلید موجود در صفحه کلید را به ترتیب از ۰ تا F نامگذاری کرده وسپس برنامه ای بنویسید که در صورت فشردن هر کلید مقدار نامگذاری شده برای ان برروی صفحه LCD به نمایش در اید.

ج) برنامه ای بنویسید که ابندا ۵عدد دو رقمی را از صفحه کلید دریافت و برروی LCD نمایش داده سپس اعداد زوج را تشخیص و تعداد انرا بروی LCD نمایش دهد.

د) پنج عدد از کلیدهای موجود بر روی صفحه کلید را به ترتیب برای علامتهای +و- /و*و=تعریف نموده و سایر کلیدها را برای ارقام ۰تا۹ در نظر بگیرید.حال برنامه ای بنویسید که ابتدا یک عدد۴رقمی را از صفحه کلید دریافت و برروی LCD نمایش داده سپس منتظر تعیین نوع عمل محاسباتی (+ و - * /)گردیده و در مرحله بعد عدد دوم نیز وارد شده و به محض فشردن کلید = نتیجه عمل ریاضی برروی LCD به نمایش در آید.

۶)تایمر/کانترها

الف) به کمک تایمر /کانتر صفر و قرار دادن ان در حالت تایمر برنامه ای بنویسید که یک موج مربعی با فرکانس ۱KHz برروی پایه PortB۳,۰ تولید گردد.

ب) دو عدد کلید فشاری به نامهای Upو down به پایه های PortA.۰ و PortA.۱ متصل نموده و برنامه قبلی را بگونه ای تغییر داده که امکان کم وزیاد نمودن فرکانس موج مربعی فراهم گردد.

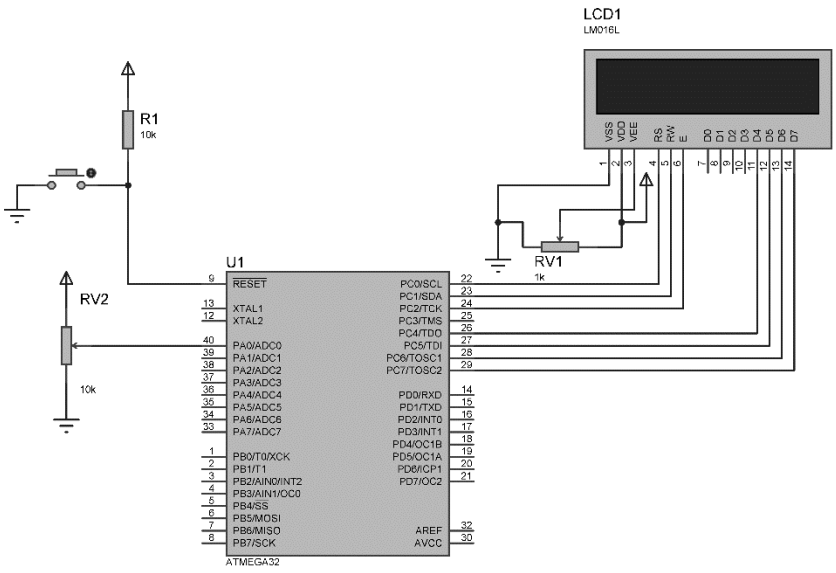
ج) توسط تایمر/کانتر ۱، یک موج PWM با دوره کارکرد ۲۰ درصد برروی پایه های PortD.۴ تولید نمایید.فرکانس موج PWM را ۲۵۰Hz در نظر بگیرید.

د) از یک سیگنال ژنراتور یک موج مربعی با فرکانس ۱۵۰Hz گرفته و ان را به پایه Icp (پایه PortD.۶) متصل کرده سپس برنامه ای بنویسید که فرکانس موج را به کمک خاصیت capture در تایمر/کانتر یک اندازه گیری نموده و برروی LCD نمایش دهد.

۷)مبدل انالوگ به دیجیتال(ADC)

مطابق شکل یک پتانسیومتر را به یکی از کانالهای ADC داخلی متصل نموده سپس می خواهیم مقدار ولتاژ انالوگ وارد شده به پایه PortA.۰ را مرتبا برروی LCD نمایش دهیم برنامه مربوط به این کار را بنویسید.

ب) می خواهیم به کمک سنسور دمای LM۳۵ یک دماسنج دیجیتالی طراحی نماییم.برای این کار خروجی LM۳۵ را به یکی از کانالهای ADC متصل نموده (سنسور LM۳۵ از سنسورهای دقیق حرارتی بوده که ولتاژخروجی ان به طور خطی متناسب با دما تغییر میکند) و درجه حرارت محیط را برحسب سانتی گراد برروی LCD نمایش دهد.رینج دمایی قابل اندازه گیری توسط LM۳۵ از ۵۵- درجه سانتی گراد تا ۱۵۰درجه سانتی گراد می باشد و به ازای هر ۱,۵ درجه سانتی گراد افزایش ولتاژخروجی ان ۱۰ میلی ولت زیاد خواهد شد.



ج) یک موج سینوسی با فرکانس ۵۰Hz و حداکثر دامنه ۵ ولت را توسط سیگنال ژنراتور به یکی از کانالهای ADCمتصل نموده و با فرکانس نمونه برداری مناسب انرا به دیجیتال تبدیل و مقادیر بدست امده توسط ADC را برروی LCD نمایش دهید.

۸)ارتباطات سریال

الف) می خواهیم عدد۷۵Hرا بصورت سریال و اسنکرون با نرخ انتقال ۹۶۰۰ بیت در ثانیه از طریق پایه TDXارسال نماییم. برنامه مربوط به این کار را نوشته سپس ان را در یک حلقه تکرار قرار داده و خروجی TDXرا برروی اسیکولوسکوپ مشاهده نمایید.سعی کنید بیتهای شروع وپایان و بیتهای داده را برروی اسیلوسکوپ پیدا نمایید.

ب) دو عدد میکرو کنترل AVR را از طریق پروتکول USART به یکدیگر متصل کرده سپس به هر میکرو کنترلر یک صفحه کلید ۴*۴ و یک LCD متصل نمایید.برنا مه ای بنویسید که هر کاراکتری که توسط صفحه کلید زده میشود این کاراکتر برای میکرو کنترلر مقابل بصورت سریال فرستاده شده و برروی LCD آن نمایش داده شود و این کار بصورت دو طرفه باشد.

یعنی هر میکروکنترلری بتواند هم فرستنده و هم گیرنده باشد. برای اتصال دو میکروکنترلر به یکدیگر می بایست پایه های TXD و RXD متقابلا به یکدیگر متصل گردند.

ج) آزمایش ب را توسط پروتکل SPI مجددا تکرار نمایید.

د) آزمایش الف را توسط پروتکل I۲C تکرار نمایید.

ه) میکرو کنترلر AVR را از طریق تراشه MAX۲۳۲ به پورت RS۲۳۲یک کامپیوتر متصل نموده سپس برنامه ای بنویسید که هر عددی که توسط صفحه کلید ۴*۴ در قسمت مربوط به میکرو کنترلر فشرده میشود برروی مانیتور کامپیوتر نمایش داده شده و هر کلیدی که بر روی صفحه کلید کامپیوتر فشرده میگردد برروی صفحه LCDمتصل به میکرو کنترلر به نمایش در آید.