دانشگاه حکیم سبزواری

دانشکده مهندسی برق و کامپیوتر

دستورکار آزمایشگاه میکروپروسسور

به انضمام راهنمای CodevisionAVR

تهیه و تنظیم:

معظم توکلیان

زمستان ۱۳۹۲

مقدمه ای بر زبان C :

در این مقدمه تنها سعی می شود به طور مختصر به تفصیل و تشریح بعضی دستورات مهم و کاربردی زبان برنامه نویسی C که در برنامه نویسی میکروکنترلرها استفاده می شود, بپردازیم و لذا برای آشنایی بیشتر با این دستورات دانشجو می تواند به کتابهای مرجعی که در آخر معرفی می شود و این کتاب نیز از آنها نشئت گرفته است, مراجعه کند .

انواع داده در محیط Atmel Studio برای AVR :

نوع داده	طول داده (بیت)	محدوده رقمی	مثال
Unsigned char	۸-bit	[۰،۲۵۵]	تعریف متغییر A به عنوان Char Char A ;
Char	۸-bit	[-۱۲۸،۱۲۷]	
Unsigned int	۱۶-bit	[۰،۶۵۵۳۵]	
Int	۱۶-bit	[-۳۲۷۶۸،۳۲۷۶۷]	
Unsigned long	۳۲-bit	[۰،۴۲۹۴۹۶۷۲۹۵]	
Long	۳۲-bit	[-۲۱۴۷۴۸۳۶۴۸،۲۱۴۷۴۸۳۶۴۸]	
Float	۳۲-bit	[۱,۱۷۵e-۳۸,۳,۴۰۲e۳۸]	
Double	۳۲-bit	[۱,۱۷۵e-۳۸,۳,۴۰۲e۳۸]	

انواع داده در محیط CodeVision برای AVR :

Type	Size (Bits)	Range
bit	۱	۰ , ۱
bool, _Bool	۸	۰ , ۱
char	۸	-۱۲۸ to ۱۲۷
unsigned char	۸	۰ to ۲۵۵
signed char	۸	-۱۲۸ to ۱۲۷
int	۱۶	-۳۲۷۶۸ to ۳۲۷۶۷
short int	۱۶	-۳۲۷۶۸ to ۳۲۷۶۷
unsigned int	۱۶	۰ to ۶۵۵۳۵
signed int	۱۶	-۳۲۷۶۸ to ۳۲۷۶۷
long int	۳۲	-۲۱۴۷۴۸۳۶۴۸ to ۲۱۴۷۴۸۳۶۴۷
unsigned long int	۳۲	۰ to ۴۲۹۴۹۶۷۲۹۵
signed long int	۳۲	-۲۱۴۷۴۸۳۶۴۸ to ۲۱۴۷۴۸۳۶۴۷
float	۳۲	۱,۱۷۵e-۳۸ to ۳,۴۰۲e۳۸
double	۳۲	۱,۱۷۵e-۳۸ to ۳,۴۰۲e۳۸

✓ ***نکته :*** هر متغییری را باید قبل از استفاده تعریف کنید , و مقدار اولیه آن را نیز می توانید در همان محلی که تعریف می کنید , مقدار دهی کنید . به عنوان مثال برای متغییر A داریم :

Char A=۱۰;

- ✓ ***نکته :*** حتما قبل از استفاده از متغییر ها , آنها را مقدار دهی کنید . که در صورت عدم مقدار دهی اولیه و استفاده از یک متغییر , اشکالی در کمپایل برنامه پیش نخواهد آمد , اما مقدار آن متغییر که بعد از تعریف , مقدار دهی نشده باشد , مشخص نیست و هر مقداری می تواند داشته باشد .
- ✓ ***نکته :*** باید دقت داشته باشید که ممکن است بسته به محل تعریف متغییر , متغییر را می توان محلی یا عمومی (Global) تعریف کرد . به طوریکه خارج از آن محل تعریف متغییر دیگر نتوانید به متغییر دسترسی داشته باشید , به عنوان مثال اگر متغییر شمارنده حلقه را در خود حلقه تعریف کنید , دیگر این متغییر خارج از حلقه قابل دسترسی نخواهد بود .

آرایه ها :

آرایه ها در حقیقت ماتریسهایی از متغییر ها می باشند . همانطور که یک ماتریس به طور کلی دارای یک نام می باشد اما مقادیر درایه های هر کدام مستقل از دیگری است , یک آرایه هم مجموعه ای از متغییرهای هم نام است . آرایه ها همانند ماتریس ها می توانند یک بعدی یا چند بعدی باشند که بسته به نیاز تعریف می شوند . در ذیل نمونه ای از تعریف آرایه ی یک بعدی با نام x و اندازه ۵ تعریف شده است :

```
unsigned char x[۵]={۱, ۲, ۳, ۴, ۵};
```

همانند آنچه نشان داده شده است , برای مقدار دهی یک آرایه از آکولاد {} استفاده می شود و می توان ارایه را مقدار دهی اولیه نکرد . برای دستیابی به متغییرها می توان از طریق x[n] آنها را فراخوانی کرد که در مثال بالا n از صفر شروع می شود و تا ۴ خواهد بود تا مجموع آنها پنج خانه بشود . مثلا x[۰] برابر عدد ۱ هست یا x[۴] برابر عدد ۵ می باشد .

نحوه نمایش اعداد در زبان C :

در محیط کدویژن برای نوشتن اعداد می توان به نحوه های مختلفی عمل کرد , در ادامه به چند مورد آن اشاره می کنیم :

مبنای ۱۰ : اگر عددی بدون هیچ پیش نویسی باشد , به عنوان یک عدد در مبنای ۱۰ تلقی می گردد .

مبنای ۱۶ : به منظور نوشتن اعداد در مبنای ۱۶ یا Hex باید از پیش نویس ۰x قبل از عدد استفاده کرد , مثلا اگر بخواهیم با عدد ۲۰ که در مبنای ۱۶ است (یعنی این عدد در مبنای ۱۰ برابر ۳۲ می باشد) کار کنیم باید آن را بصورت ۰x۲۰ استفاده کنیم .

نکته : دقت شود که چون ثباتهای میکروکنترلر ۸ بیتی می باشند , لذا برای مقدار دهی ثباتها تنها از اعداد دو رقمی در مبنای ۱۶ استفاده می شود .مانند: ۰x۲۰, ۰xAC, ۰xAD

مبنای دودویی : بمنظور دودویی نشان دادن اعداد یا همان باینری از پیش نویس ۰b استفاده می شود . مثلا بفرض اگر بخواهیم عدد ۲۰ را که در مبنای ۱۶ است , در مبنای دودویی نشان دهیم خواهیم داشت : ۰b۰۰۱۰۰۰۰۰

نکته : در این مورد هم باید دقت شود که برای مقدار دهی ثباتها تنها باید ۸ بیت مقدار دهی بشوند , نه بیشتر .

به صورت کد اسکی : گاهی ممکن است کاربر اعداد اسکی حروف را نیاز داشته باشد , مثلا عدد اسکی حرف A را بخواهد و دقیقا نداند عدد آن چند است , به این منظور اگر بخواهد از عدد کد اسکی A استفاده کند , کافیست آن را بصورت 'A' بنویسیم که مثلا در عبارت i='A' , متغییر i مقدار عددی اسکی حرف A را در خود ذخیره کرده است .

دستورات بیتی در زبان C :

عملگر (Operator)	نحوه عملکرد
&	و (AND)
|	یا (OR)
^	یای انحصاری (XOR)
~	نقیض (Not)
>>	شیفت به راست (right shift)
<<	شیفت به چپ (left shift)

نمونه ای از نحوه کاربرد دستورات بیتی :

A	B	A&B	A|B	A^B	~B
۰	۰	۰	۰	۰	۱
۰	۱	۰	۱	۱	۰
۱	۰	۰	۱	۱	۱
۱	۱	۱	۱	۰	۰

عملگرهای منطقی و رابطه ای در زبان C :

عملگر (Operator)	نحوه عملکرد
&&	و (AND)
||	یا (OR)
!	نقیض (Not)
< <= >= >	بررسی کوچکتر یا بزرگتری
==	بررسی تساوی
!=	بررسی عدم تساوی

نمونه ای از نحوه کاربرد عملگرهای منطقی :

A	B	A&&B	A||B	A==B	!B
۰	۰	۰	۰	۱	۱
۰	غیر صفر	۰	۱	۰	۰
غیر صفر	۰	۰	۱	۰	۱
غیر صفر	غیر صفر	۱	۱	۱	۰

نمونه ای از نحوه کاربرد عملگرهای رابطه ای :

نوع عملگر	معنای عملگر	مثال	خروجی در صورتی که متغییرage دارای مقدار ۴۲ باشد
==	تساوی	age == ۱۰۰	**False (۰)**
!=	نامساوی	age != ۰	**True (۱)**
<	کوچکتری	age < ۲۱	**False (۰)**
<=	کوچکتری یا تساوی	age <= ۱۸	**False (۰)**
>	بزرگتری	age > ۱۶	**True (۱)**
>=	بزرگتری یا تساوی	age >= ۳۰	**True (۱)**

عملگرهای جابجایی و انتساب ترکیبی بیتی در زبان C :

نحوه استفاده	دستور	عملکرد دستور جابجایی	دستور معادل	دستورانتساب ساده شده
مقدار انتقال به راست >> داده	>>	انتقال به سمت راست	A=A&B ;	A&=B ;
مقدار انتقال به چپ << داده	<<	انتقال به سمت چپ	A= A|B ;	A|=B ;

تفاوت دستورات بیتی با دستورات منطقی :

باید همیشه به این نکته توجه داشت که خروجی عمگرهای منطقی در زبان C مقداری برابر درست یا نادرست دارد که همیشه مقدار نادرست را با صفر و مقدار درست را با مقداری غیر از صفر و عموما با یک نشان می دهد .

دستورات انتساب ترکیبی در زبان C :

دستور معادل :	دستور ساده شده :
variable = variable * number;	**variable *= number;**
variable = variable / number;	**variable /= number;**
variable = variable % number;	**variable %= number;**
variable = variable + number;	**variable += number;**
variable = variable - number;	**variable -= number;**

✓ ***نکته :*** همانطوری که در جدول بالا نیز دیده می شود خروجی دستورات رابطه ای که در بالا نشان داده شده است تنها یک عبارت بولین و به عبارت دیگر یک مقدار true یا false و یا یک مقدار صفر یا یک است که به اینگونه عبارات که تنها یکی از دو مقدار درست یا غلط را می توانند بگیرند ,عبارات بولین یا Boolean expression می گویند که این عبارات بسیار پر کاربرد هستند و نمونه های آن را در تعارف دستورات بعدی خواهید دید .

✓ ***نکته :*** دقت داشته باشید که اینگونه عبارات تنها شرطی را بررسی می کنند و در صورت درستی مقدار صفر و در صورت نادرستی مقداری غیر صفر (نه الزاما یک) را باز می گردانند و برخلاف عملگرهای انتساب هیچ تغییری در خود متغییر ها نمی دهند . توجه به این نکته بسیار مهم است که این عملگرها را با عملگرهای انتساب در برنامه نویسی اشتباه نکنیم , چرا که این عملگرها اساسا با هم متفاوت می باشند و اشتباه نوشتن آن ها نیز نوعا خطایی در کامپایل برنامه برای شما بوجود نمی آورد و لذا خطایابی برنامه در صورتی که از اشتباه نوشتن و خلط کردن این عملگرها با هم باشد کار دشواری خواهد بود .

اولویت در عملگرها :

دو نوع اولویت در بخش برنامه نویسی در زبان C مطرح می شود :

الف) اولویت در بین عملگرها : که اینگونه اولویت از تقدّم و تاخّر بین عملگرها حاصل می شود , به عنوان نمونه در مثال زیر مقدار نهایی آن می تواند دو مقدار داشته باشد :

۲+۳*۶

۱ – ابتدا عدد ۲ را با ۳ جمع کنیم و بعد در عدد ۶ ضرب کنیم که حاصل می شود ۳۰

۲ – ابتدا ۳ را در شش ضرب کنیم و بعد آن را با ۲ جمع کنیم که حاصل می شود ۲۰

با توجه به جدول زیر که تقدم عملگرها را بیان کرده است , تقدم ضرب از جمع بیشتر است و لذا ابتدا عمل ضرب انجام می شود و سپس عمل جمع و در نتیجه حاصل این عبارت مقدار ۲۰ می شود .

جدول اولویت عملگرها :

	تقدم عملگرهای محاسباتی	تقدم عملگرهای منطقی
بیشترین تقدم	- - ++ % / * - +	! => > >= < <= =! == && ||
کمترین تقدم		

ب) اولویت براساس قاعده شرکت پذیری : که اینگونه اولویت زمانی مدّ نظر قرار می گیرد که از عملگرهایی در کنار هم استفاده شود که دارای اولویت یکسانی هستند . به عنوان نمونه در عملیات مقابل, مقدار نهایی عبارت می تواند به دو صورت به دست بیاید : ۴/۲*۶

۱ – ابتدا عمل تقسیم انجام بشود و بعد عمل ضرب که یعنی ۴ بر ۲ تقسیم بشود و در ۶ ضرب بشود که حاصل ۱۲ است .

۲ – ابتدا عمل ضرب و سپس عمل تقسیم انجام بشود که حاصل می شود ۴/۱۲ .

باید گفت که در زبان C اینگونه دستورات از سمت چپ به راست اجرا خواهند شد , یعنی در این مثال بخصوص که دارای اولویت یکسانی بین عملگرها است , ابتدا عمل تقسیم انجام می شود و سپس عمل ضرب و در نتیجه حاصل برابر ۱۲ خواهد بود . که حتما باید به این نکته دقت کرد که در بین عملگرها , زمانی اولویت براساس قاعده شرکت پذیری منظور می شود که عملگرها دارای تقدم یکسانی باشند , در غیر این صورت مانند مثال بالا حتی بر خلاف این قاعده عمل می شود .

✓ **نکته :** همچنین می توان با استفاده از پرانتز , اولویت یک عملگر را بالاتر از بقیه عملگرها قرار داد , به عنوان نمونه در همان مثال بالا , اگر عبارت را به صورت ۶*(۳+۲) بنویسیم , ابتدا عمل داخل پرانتز انجام می شود و سپس عمل ضرب و در نتیجه حاصل این عبارت به ۳۰ تغییر پیدا می کند . پرانتز ها در بیشترین اولویت نسبت به همه عملگرها می باشند و در واقع اگر بخواهیم پرانتز را به جدول اولویت عملگر ها اضافه کنیم, در صدر جدول اولویت قرار خواهد گرفت .

ساختار کلی دستور if :

ساختار کلی یک دستور if به صورت زیر است :

```
if ( booleanExpression-۱ )
statement-۱;
else if (booleanExpression-۲)
statement-۲;
...
else
statement-۳;
```

که به جای عبارات Boolean expression باید حتما از یک عبارت بولین استفاده کنید یا به معنای دیگر باید از دستورات رابطه ای که خروجی آنها یک مقدار بولین است و یا دستوراتی که خروجی آنها یک عدد است به جای این عبارتهای داخل پرانتز های if استفاده کنیم که اگر از عدد استفاده می کنیم همانطوری که گفته شد باید دقت داشته باشیم که عدد صفر به منزله اجرا نشدن if و مقدار false است و هر عددی غیر از صفر به عنوان مقدار یک منطقی یا true برای سیستم تلقی می شود لذا باید دستورات را نیز طوری نوشت که اگر خروجی ما قرار است یک عدد باشد , در صورتی مقدار صفر بدهد که ما نمی خواهیم دستور if اجرا شود .

و اما به جای عبارات statement-n باید دستورات برنامه را در حالاتی که مورد نیاز است تعیین کنیم , که اگر بخواهیم بیش از یک دستور استفاده کنیم باید دستورات را در کروشه {} قرار دهیم که در غیر این صورت معمولا با مشکل در کامپایل برنامه دچار خواهیم شد . دو مثال زیر بخشی از یک برنامه است که نمونه ای از طریقه استفاده دستور if می باشد .

مثال ۲ :	مثال ۱ :
و این مثال مقدار متغییر day را بررسی می کند که اگر هر کدام از مقادیر ۰ تا ۶ را داشت به ترتیب روز های هفته از جمعه (روز ۰) تا پنجشنبه (روز ۶) را در متغییر dayName که معرف روز هفته است قرار می دهد . و در غیر همه این موارد مقدار unknown را به متغییر dayName اختصاص می دهد .	که این مثال به عنوان یک ثانیه و دقیقه شمار عمل می کند که در صورتی که متغییر ثانیه (seconds) به مقدار ۵۹ برسد به متغییری که نشان دهنده دقیقه است (minutes) یکی اضافه می کند و ثانیه را برابر صفر قرار می دهد . در غیر این صورت به ثانیه یک مقدار اضافه می کند .

if (day == ۰)
dayName = "Friday";
else if (day == ۱)
dayName = "Saturday";
else if (day == ۲)
dayName = "Sunday";
else if (day == ۳)
dayName = "Monday";
else if (day == ۴)
dayName = "Tuesday";
else if (day == ۵)
dayName = "wednesday";
else if (day == ۶)
dayName = "Thursday" ;
else
dayName = "unknown" ;
	int seconds = ۰;
	int minutes = ۰;
	...
	if (seconds == ۵۹)
	{
	seconds = ۰;
	minutes++;
	}
	else
	seconds++ ;

ساختار کلی دستور switch – case :

در مواردی شبیه موارد مثال ۲ در قبل ساختار else if تقریبا در همه عبارات به غیر موارد اندکی شبیه به هم می شوند و لذا این دستورات را معمولا می توان با ساختار switch-case نیز نوشت . ساختار کلی یک دستور switch-case به صورت زیر است :

```
switch ( controllingExpression )
{
case constantExpression :
        statements
        break;
case constantExpression :
        statements
        break;
...
default :
        statements
        break;
}
```

در این ساختار به جای عبارت controllingExpression متغییری را که می خواهیم بررسی کنیم را قرار می دهیم (تنها یک متغییر می توانیم قرار دهیم) و بعد از اجرای خط اول برنامه عبارات constantExpression را بررسی می کند که اگر مقدار متغییر ما با هر کدام از این ها (که به عنوان case label یا برچسب case شناخته می شوند) برابری کرد , دستورات آن را اجرا می کند . که اگر هیچ کدام از مقادیری

که در Case ها قرار داده شده است با متغییر ما تطبیق نداشته باشد بخش default و دستورات مربوط به آن اجرا می شود که اگر در این شرایط ما از default استفاده نکرده باشیم (چرا که نوشتن این بخش اختیاری است) برنامه بدون اجرای هیچ کدام از Case ها , از ساختار switch-case خارج می شود . مثال زیر , نمونه از نحوه به کارگیری این ساختار است .

این مثال همان مثال شماره ۲ است که در ساختار if بیان شد و اکنون به صورت ساختار switch-case نوشته شده است . دقت کنید که استفاده از دستور break بعد از هر case در این شرایط الزامی است , در غیر این صورت اگر از دستور break استفاده نشود برنامه , برچسب بعدی و دستورات مربوط به آن را نیز اجرا خواهد کرد .	switch (day)
	{
	case ۰ :
	dayName = "Sunday";
	break;
	case ۱ :
	dayName = "Monday";
	break;
	case ۲ :
	dayName = "Tuesday";
	break;
	...
	default :
	dayName = "Unknown";
	break;
	}

ساختار switch-case در خیلی موارد بسیار مفید است اما متاسفانه این ساختار دارای محدودیت هاو قوانینی است که موجب می شود نتوان همیشه از آن استفاده کرد این محدودیتها و قوانین شامل موارد زیر است :

- از ساختار switch-case تنها می توان برای داده های اصلی مانند int یا string استفاده کرد . درصورتی که بخواهیم داده های دیگری مانند float یا double را به کار ببریم , باید از ساختار else if استفاده کنیم .
- برچسبهای case ها (یعنی constantExpression ها) باید یک مقدار ثابت مانند ۴۲ یا "۴۲" باشند . و نمی توان در آنها از اعمال محاسباتی مثلا i*۲ که مقدار i برابر ۲۱ می باشد به جای عدد ۴۲ استفاده کرد و در صورتی که نیاز به این محاسبات داریم باید از ساختار else if استفاده کنیم .
- برچسبهای case ها (یعنی constantExpression ها) باید دارای مقدار بی نظیر و یکتایی باشند , به عبارت دیگر دو برچسب از case های متفاوت نمی توانند مقدار یکسانی داشته باشند .
- در این ساختار ممکن است شما بخواهید یک عبارت را برای چندین برچسب case اجرا کنید , در این صورت باید دستورات case را پشت سر هم بنویسید بدون اینکه بین هر کدام از case ها دستوری قرار دهید و در واقع تمام دستورات را در آخرین case قرار می دهید که آن دستورات به ازای تمام برچسبهای قبلی نیز اجرا می شود . در حقیقت اگر دستور break را ننویسیم تا زمانی که برنامه به این دستور نرسد تمام دستورات case های پایین تر از خود را نیز اجرا می کند . به عنوان نمونه در زیر داریم :

در این ساختار برچسبهای Heart و Diamonds در صورتی که مقدار trums با هر کدام از اینها برابری کند دستورات زیر ساختار برچسب diamonds را اجرا می کند یعنی مقدار Red را در متغییر color قرار می دهد . و اگر متغییر trumbs برابر با برچسب clubs باشد دستور color =”black”; دو بار اجرا می شود , یک بار مربوط به خود این برچسب و یک بار به دلیل برچسب بعدی که بین این دو از دستور break استفاده نشده است , در نتیجه نوشتن دستور به روش برچسبهای Hearts و Diamonds بسیار بهتر و	switch (trumps)
	{
	case Hearts :
	case Diamonds :
	color = "Red";
	break;
	case Clubs :
	color = "Black";
	case Spades :
	color = "Black";
	break;
	}

مقرون به صرفه تر از این روش است .	

ساختار کلی حلقه while :

این ساختار دستور یا دستوراتی را به ازای برقراری یک شرط به صورت مکرر تکرار می کند . پس در واقع ما در اینجا به یک عبارت شرطی یا BooleanExpression نیاز داریم . ساختار کلی دستور While به صورت زیر است :

```
initialization
while (Boolean expression)
{
statement
update control variable
}
```

همانطوری که در این دستورات می بینید و در دستور for نیز خواهید دید , وجود سه عبارت initialization و booleanExpression و update controlvariable در یک ساختار حلقه الزامی است که عبارت initialization به مقدار دهی اولیه متغییر کنترل حلقه بر می گردد که اگر این متغییر مقدار دهی نشده باشد الزاما باید این مقدار دهی اولیه انجام شود تا در گام های بعدی حلقه دچارمشکل نشویم , چراکه اگر ما مقداری را در متغییر تعریف شده به عنوان مقدار اولیه قرار ندهیم , ممکن است ان متغییر هر مقداری داشته باشد . و عبارت update controlvariable نیز در واقع شرط حلقه را به روز می کند که مثلا اگر بخواهیم یک حلقه ۱۰ بار اجرا بشود , باید در هر بار اجرای حلقه یک واحد به متغییر اضافه کنیم که این کار در حقیقت به روز رسانی شرط حلقه است .و همچنین عبارت statement بیان گر محل نوشتن دستورات است . از جمله نکاتی که در نوشتن دستور while باید رعایت شود می توان به موارد زیر اشاره کرد :

- عبارت قرار داده شده حتما باید یک عبارت بولی باشد که دو مقدار درست یا نادرست را برگرداند . (قبلا گفته شد که عبارتهای بولی به ازای مقدار درست مقدار غیر صفر و به ازای یک مقدار نادرست مقدار صفر را بر می گردانند , در نتیجه اگر عبارتی باشد که یک عدد را به عنوان شرط حلقه برگرداند اگر عدد صفر باشد حلقه اجرا نخواهد شد و اگر عدد غیر صفر باشد حلقه اجرا خواهد شد . در برنامه نویسی AVR نوعا مرسوم است که کل دستورات را در یک حلقه While قرار می دهند و به جای عبارت booleanExpression مقدار یک را جایگزین می کنند تا همیشه این حلقه اجرا شود .)
- عبارت booleanExpression حتما باید داخل پرانتز نوشته شود .
- در صورتی که مقدار عبارت بولی نادرست باشد , دستورات حلقه اصلا اجرا نخواهد شد .
- در صورتی که می خواهید بیشتر از یک دستور را داخل حلقه قرار دهید , باید دستورات را داخل آکولاد قرار دهید .

نمونه ای از طریقه استفاده از ساختار حلقه while به صورت زیر می باشد :

```
int i = ۰;
while (i < ۱۰)
{
PORTB = i ;
i++;
}
```

که در اینجا حلقه تا زمانی که مقدار متغییر i کمتر از ۱۰ باشد تکرار خواهد شد و در هر بار تکرار نیز مقدار i (با فرض خروجی بودن پورت) به پورت B ارسال می شود . و در این مثال booleanExpression همان عبارت جلوی دستور while است (یعنی i<۱۰) که شرط حلقه می باشد , و عبارت initialization در حقیقت همان دستور اول می باشد یعنی int i=۰; که هم متغییر را تعریف کرده ایم و هم مقدار صفر را به عنوان مقدار اولیه در ان قرار داده ایم , و عبارت i++; نیز همان update controlvariable می باشد که بعد از هر بار اجرای دستورات (که در اینجا فقط یک دستور اصلی داریم که مقدار متغییر را به پورت می فرستد) یک واحد به متغییر i اضافه می کند و در حقیقت متغییر را به روز رسانی می کند .

✓ ***نکته :*** حتما الزامی نیست که مقدار اولیه متغییر را برابر با صفر قرار دهیم , بلکه انتخاب مقدار اولیه بستگی به نحوه کاربرد و دستورات ما دارد , به عنوان مثال اگر بخواهیم مقدار پورت B از ۱۰ تا صفر کاهش یابد باید دستورات را به صورت زیر تغییر دهیم :

```
Int i = ۱۰;

While ( i<۰)

{

PORTB = i;

i--;

}
```

که در اینجا از شماره ۱۰ مقدار i شروع شده و یک واحد یک واحد از ان کسر می شود و به پورت ارسال می شود که این نکته بسیار مهم است که برای هر حلقه سه عبارت initialization و booleanExpression و update controlvariable را مطابق با برنامه و نوع کاربرد مورد نیاز باید تعیین کنیم , که اشتباه در تعیین هر کدام از اینها باعث بروز خطاهایی می شود که عیب یابی آنها گاها بسیار دشوار است .

ساختار کلی حلقه for :

این حلقه نیز از همان سه عبارتی که در حلقه while بود استفاده می کند , منتها طریقه نوشتن این حلقه به صورت زیر است :

for (initialization; Boolean expression; update control variable)
statement

که سه عبارت داخل پرانتز همان عبارات مقدار دهی اولیه و شرط حلقه و به روز رسانی متغییر کنترل حلقه می باشد که توسط کاما از یک دیگر جدا شده اند .
به عنوان نمونه , مثال قبل که با حلقه while نوشته شده بود اکنون با حلقه for به صورت زیر خواهد :

```
for (int i = ۰; i < ۱۰; i++)
{
PORTB = i ;
}
```

✓ ***نکته :*** دقت کنید که شما می توانید داخل حلقه for همانطوری که نشان داده شده است متغییری را تعریف کنید , اما در این صورت دیگر در خارج از حلقه به ان متغییر دسترسی نخواهید داشت , یعنی اگر بخواهید بعد از آکولاد بسته حلقه for از متغییر i استفاده کنید , کامپایلر از شما خطا خواهد گرفت . به عنوان مثال دستور زیر کامپایل نخواهد شد :

```
for (int i = ۰; i < ۱۰; i++)
{
PORTB = i ;
}

PORTA = i ;   //. این دستور به دلیل اینکه متغییر خارج از حلقه تعریف نشده است کامپایل نخواهد شد
```

برای رفع این اشکال در صورتی که به متغییر i خارج از حلقه نیز نیازمند هستید باید متغییر را خارج از حلقه مانند زیر تعریف کنید :

```
Int i ;
for ( i = ۰; i < ۱۰; i++)
{
PORTB = i ;
}
```

✓ ***نکته :*** در برنامه کدویژن شما نمی توانید متغییر حلقه را در خود حلقه for تعریف کنید , یعنی نمی توانید از دستور زیر استفاده کنید :
for (int i=۰ ; i<=۱۰ ; i++) که در اینجا متغییر i در خود حلقه تعریف شده است و کدویژن در کامپایل اینطور دستوری اشکال می گیرد .

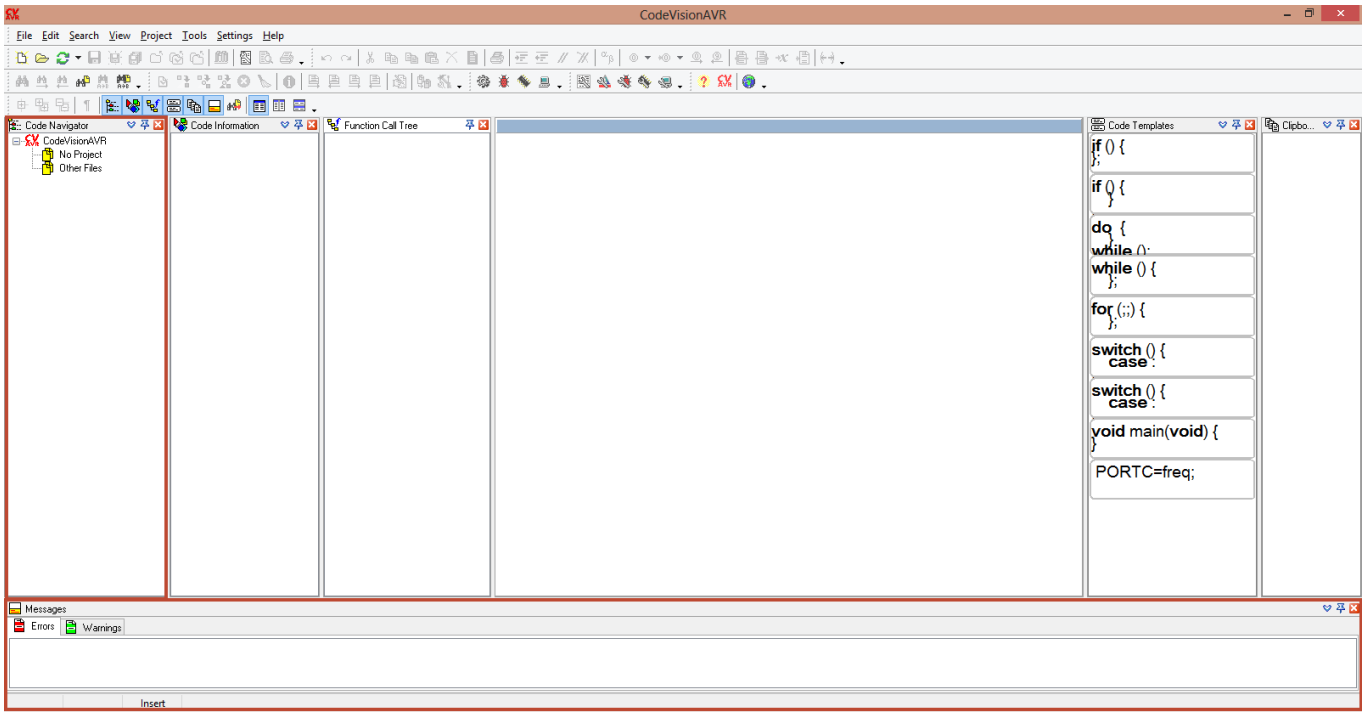
✓ ***نکته :*** حلقه for این قابلیت را دارد که مانند حلقه while بتوانید سه عبارت اساسی حلقه را به صورت جدا شده از هم استفاده کنید , یعنی می توانید حلقه for را به صورت زیر به کار ببرید :

```
Initialization ;
for (; Boolean expression;)
{
Statement
update control variable
}
```

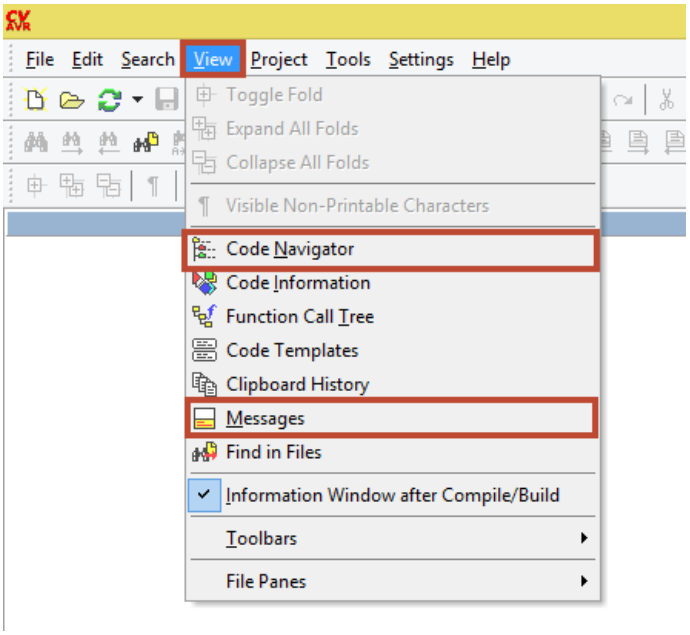
که حتی می توانید عبارت booleanExpression را نیز به داخل حلقه ببرید و در آن صورت حلقه for به شکل for(; ;) در خواهد آمد .

کار با نرم افزار CodeVision AVR :

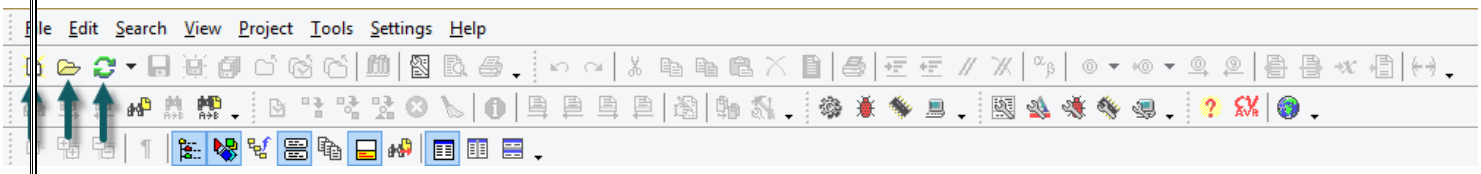
پس از نصب برنامه و باز کردن آن با پنجره ای شبیه پنجره بالا روبرو می شوید . در قسمت بالا نوار ابزار هایی قرار دارد که معمولا کاربرد بیشتری در برنامه نویسی دارند .



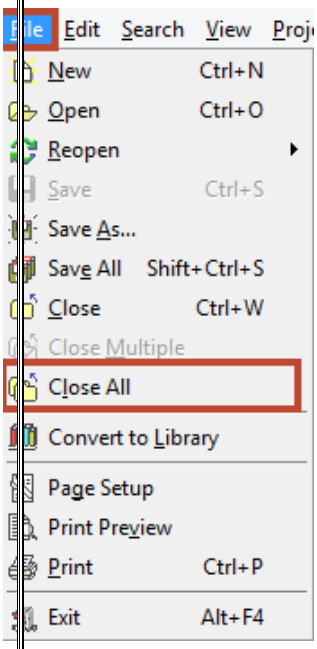
برای بستن پنجره های اضافی که در ابتدا (بعد از نصب) با آن مواجه می شوید می توانید از گزینه ضرب-در ( ) در کنار هر پنجره استفاده کنید. پنجره هایی که معمولا در آزمایشگاه بیشتر به آن نیاز دارید پنجره های Code Navigator (در سمت چپ) و پنجره Messages (در پایین صفحه) است . اگر به اشتباه پنجره ای را بستید می توانید از منوی View در بالا نرم افزار پنجره مورد نظر را انتخاب و باز کنید , به عنوان مثال اگر پنجره های نام برده پس از باز کردن محیط نرم افزار مشاهده نشد , می توانید به صورت زیر به راحتی آنها را نمایش دهید .



در ادامه به توضیح سه منوی نشان داده شده برای باز کردن پروژه جدید یا پروژه های قبلی می پردازیم :

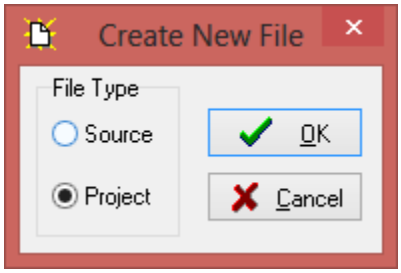


سه منوی نشان داده شده در شکل , برای باز کردن و یا ساخت پروژه جدید استفاده می شوند . به این صورت که اولین منو (از سمت چپ) که تصویر یک کاغذ سفید است , برای ساخت یک پروژه جدید می باشد . و دومین گزینه که همان Open هست , برای باز کردن پروژه ای است که قبلا نوشته شده است و گزینه سبز رنگ بعدی Reopen هست که برای باز کردن پروژه هایی است که قبلا باز کرده اید و آنها را بسته اید و با فلش کنار این گزینه می توانید آنها را تنها با انتخاب نامشان باز کنید . همچنین تفاوت اساسی گزینه Open با Reopen در این است که اگر در حال کار بر روی یک پروژه هستید و بخواهید پروژه دیگری را باز کنید , اگر بخواهید از گزینه Open استفاده کنید , باید پروژه قبلی را کاملا ببندید در غیر این صورت تنها فایل *.c پروژه انتخابی را باز خواهید کرد و اگر آن را تغییر دهید , برنامه چون داخل پروژه قبلی است , با زدن گزینه کامپایل همان پروژه قبلی را کامپایل می کند و فایل باز شده جدید را تغییری نمی دهد. که البته در بعضی موارد تنها می خواهید فایل برنامه پروژه قبلی را ببینید و نمی خواهید به طور کامل آن پروژه را باز کنید که در این صورت این گزینه مورد خوبی خواهد بود , اما اگر بخواهید پروژه جدید باز کنید باید پروژه قبلی را ببندید و برای بستن پروژه قبلی باید از داخل منوی File گزینه Close All را انتخاب بفرمایید .



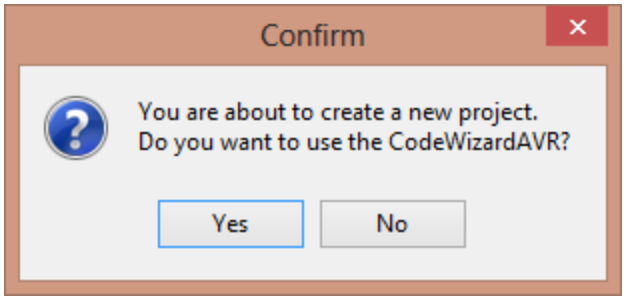
اما گزینه Reopen در حقیقت همه این کارها را با هم انجام می دهد , یعنی هم پروژه قبلی را می بندد , و هم پروژه جدید را باز می کند .که در این صورت برای بعضی موارد بسیار کاربردی تر و ساده تر خواهد بود .

اما چون برای اولین بار است که محیط کدویژن را باز می کنیم به ناچار , سراغ گزینه New می رویم .گزینه New شامل تنظیماتی است که در ادامه به توضیح مفصل تر آن می پردازیم :



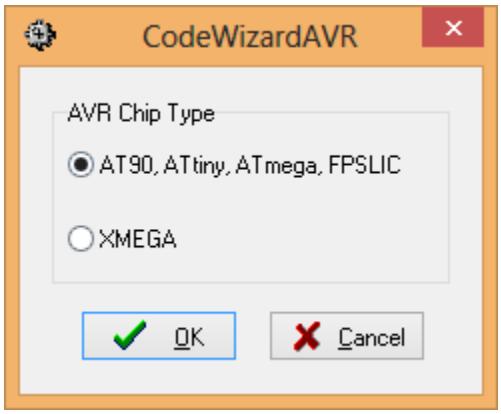
با زدن گزینه New پنجره مقابل باز می شود.گزینه source بنظر برای ساخت فایلهای کتابخانه ای استفاده می شود و ما در محیط آزمایشگاه با این گزینه کاری نخواهیم داشت .

گزینه Project برای ساخت پروژه جدید که مد نظر ما می باشد است .که با انتخاب آن و زدن دکمه ok به مرحله بعد می رویم .



در این پنجره از شما پرسیده می شود که آیا می خواهید تنظیمات سخت افزاری میکروکنترلر را با CodeWizard انجام دهید یا خیر . محیط CodeWizard تنظیماتی را به صورت دیداری (visual) برای راحتی کاربر در مقدار دهی ثبات ها فراهم می کند .

با انتخاب گزینه No مستقیما از شما آدرسی که می خواهید فایلهای پروژه در آن ذخیره شود , پرسیده می شود و پس از آن وارد محیط برنامه نویسی می شوید . (که البته در این بخش نیازمند تنظیمات مختلفی است که در این مقاله به توضیح آنها نخواهیم پرداخت .)

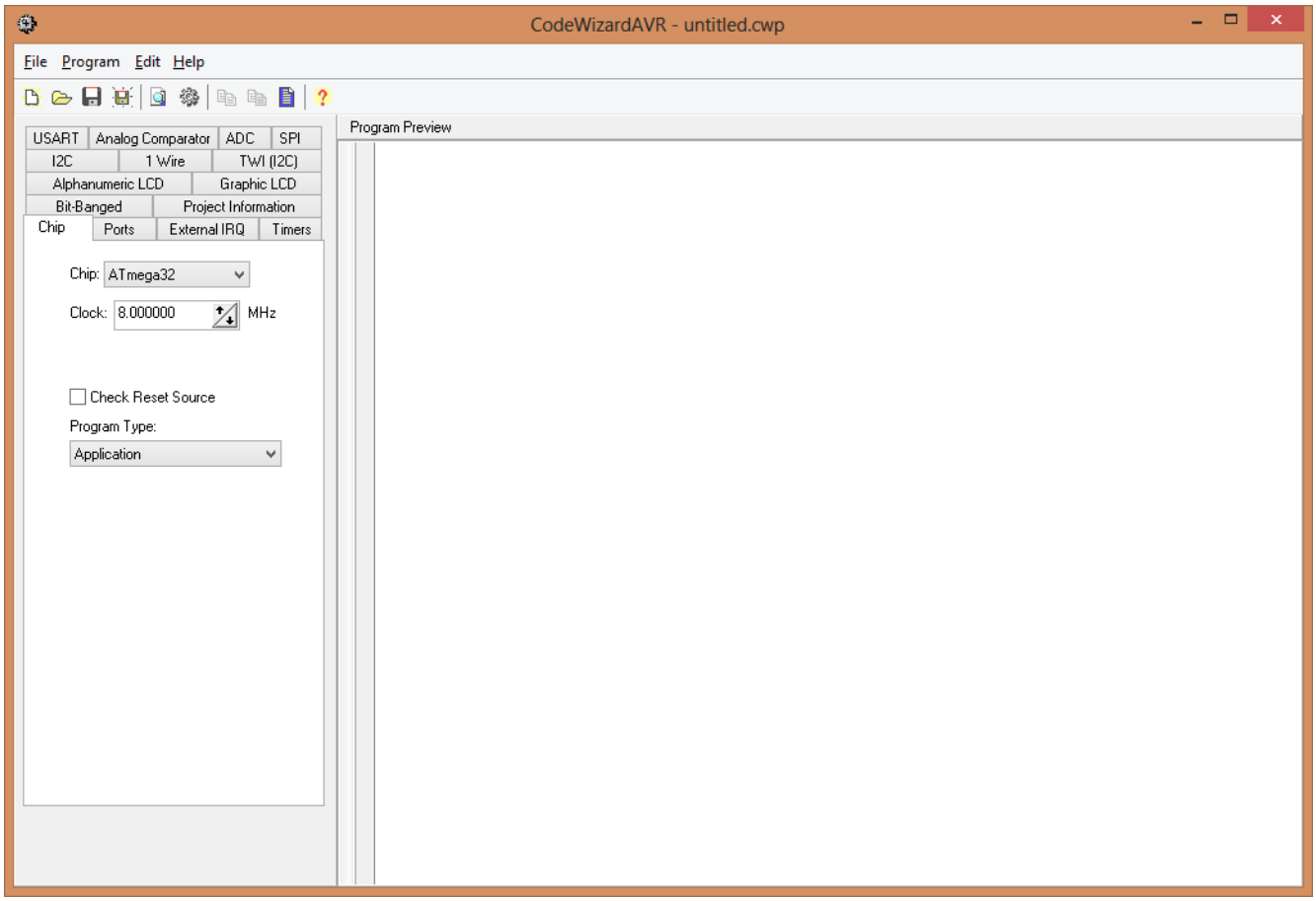


با انتخاب گزینه Yes پنجره زیر باز می شود که از شما خانواده میکروکنترلری که می خواهید برای آن برنامه نویسی کنید را می پرسد .

چون ما نوعا با سری Atmega کار می کنیم , لذا گزینه اول را انتخاب می کنیم و بر روی OK کلیک می کنیم تا پنجره اصلی CodeWizard باز شود .

تنظیمات پنجره CodeWizard :

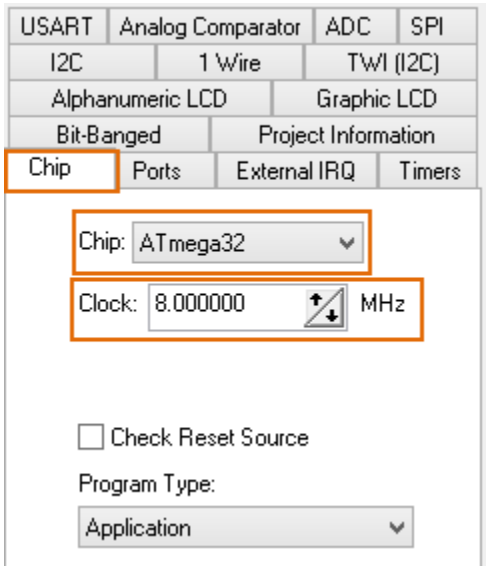
در ذیل پنجره CodeWizard به صورت کامل نشان داده شده است . در ادامه بخش های مهمی که معمولا کاربرد بیشتری دارد را توضیح مختصری می دهیم .



سربرگ Chip :

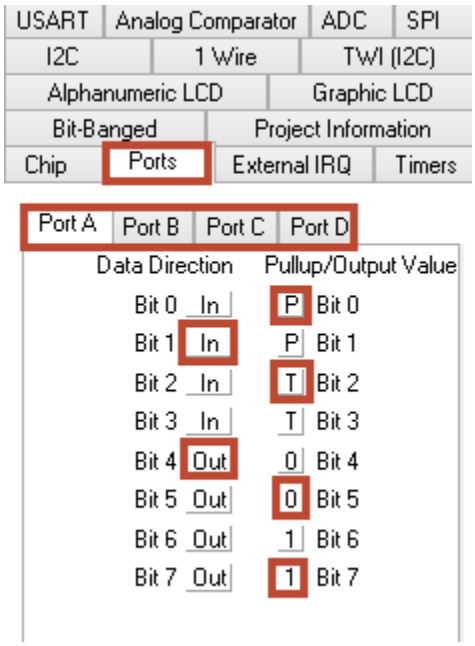
معمولا سربرگی* (Tab) که به صورت پیش فرض پس از مرحله بعدی انتخاب شده است , همان سربرگ Chip است که این سربرگ برای انتخاب میکروکنترلری است که می خواهیم برای آن برنامه نویسی کنیم . همچنین با انتخاب میکروکنترلر مورد نظر , تنظیمات سازگار مربوط به آن نیز در سربرگهای دیگر این پنجره نشان داده خواهد شد . مثلا اگر می بینید که تنظیمات پورتها صحیح نیستند یا اصلا سربرگی مانند Timers یا ADC یا LCD وجود ندارد حتما دوباره دقت کنید که میکروکنترلر درست را انتخاب کرده اید , چرا که بعضی از میکروکنترلر ها این سخت افزارها را (مانند ADC و ....) پشتیبانی نمی کنند .

*: تب , یا Tab یا سربرگ به هر کدام از صفحات کوچکی که درون پنجره اصلی قرار دارد گفته می شود. مثلا در شکل زیر هر کدام از قسمتهای سمت چپ (Ports , Chip , External IRQ , Timers و...) یک سربرگ یا تب حساب می شوند .



در شکل مقابل سربرگ Chip بهتر نشان داده شده است . همانطور که دیده می شود تنها کافیست از سربرگ Chip , میکروکنترلر ATmega۳۲ را انتخاب کنید .و در بخش Clock فرکانس کاری میکروکنترلر را که قرار است با آن برنامه ریزی شود را تعیین کنید. این فرکانس بنظر تنها برای بدست آوردن زمان تاخیر ها و بعضی توابع دیگر در محیط کدویژن استفاده می شود . واگرنه تعیین فرکانس خود میکروکنترلر در عمل توسط فیوزبیتها (نام تعدادی از بیتهای مهم و خاص که در هنگام برنامه ریزی کردن (Program) میکروکنترلر می توان آنها را تغییر داد .) و در محیط پروتئوس توسط تنظیمات میکروکنترلر که در بخش بعدی توضیح داده می شود , تعیین می گردد.

سربرگ Ports :



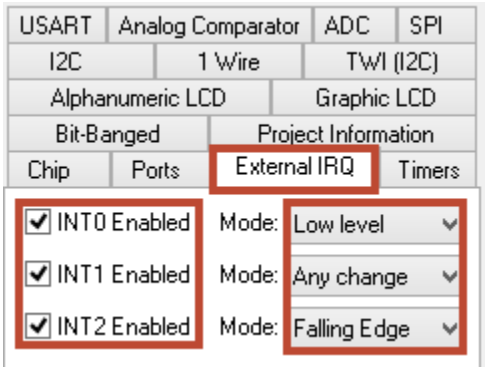
در این سربرگ شما می توانید تنظیمات مربوط به پورتها را اعم از خروجی یا ورودی بودن آنها و تنظیم مقاومت بالاکش (PullUp) انجام دهید . به عنوان مثال در شکل بیت صفر به عنوان ورودی (In) و با مقاومت بالاکش (P) تنظیم شده است . همچنین بیت یک به عنوان ورودی و بدون مقاومت بالاکش (T مخفف Tri-state) تنظیم شده است . و بیت چهار به عنوان خروجی با مقدار اولیه صفر و بیت ۷ به عنوان خروجی با مقدار اولیه یک تنظیم شده است . در حقیقت تنظیم این موارد چیزی جز مقدار دهی ثباتهای DDRx و PORTx نمی باشد (البته برای مقاومت بالاکش بیت دیگری نیز هست که فعلا از آن بحثی نخواهیم کرد .) و شما در حقیقت با خروجی یا ورودی کردن بیت ها , مقدار ثبات DDRx را یک یا صفر می کنید و به همین صورت برای مقادیر دیگر .

در قسمت بالای سربرگ , سربرگهای دیگری برای انتخاب پورت مورد نظر می باشد .

مقاومت بالا کش یا PullUp و حالت Tri-state : در حقیقت زمانی که شما یک پورت را به عنوان ورودی تعریف می کنید , این سوال مطرح می شود که اگر هیچ پایه ای به آن پورت متصل نشود , مقدار خوانده شده ان (مقدار ثبات PINx) چقدر خواهد بود ؟ پاسخ این است که بستگی دارد ! در واقع می توان تعیین کرد که اگر به پایه میکروکنترلر که به عنوان ورودی تعریف شده است , هیچ ولتاژی داده نشود , مقدار آن چقدر باشد . که مقاومت بالاکش وظیفه آن این است که اگر به ورودی مستقیما صفر یا یک داده نشود , مقدار ولتاژ ورودی را همانطور که از اسمش بر می آید بالا می کشد و برابر یک قرار می دهد . پس در این حالت باید دستگاه ورودی مقدار پورتها را صفر کند تا میکروکنترلر متوجه بشود که ورودی وارد شده است .

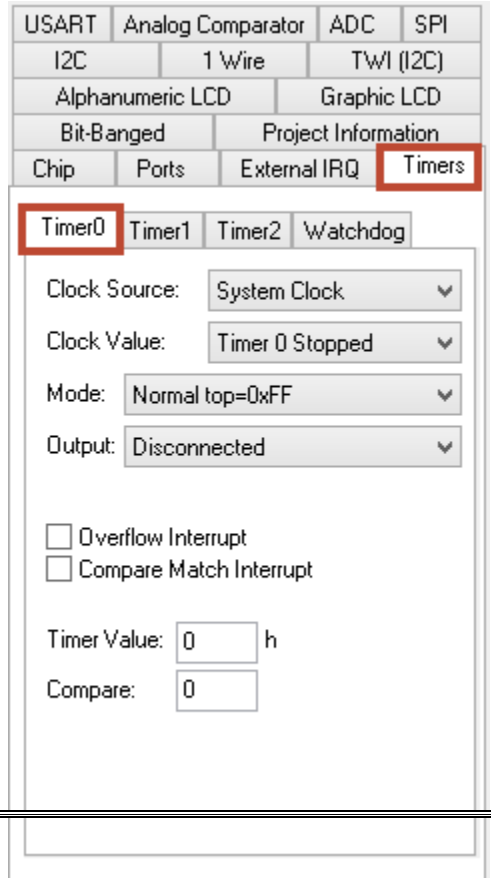
در غیر این صورت اگر مقاومت بالاکش تنظیم نشود , سه حالت وجود خواهد داشت , مقدار یک , مقدار صفر و مقدار شناور (Sink or Float) که اگر ورودی صفر داده شود , PINx صفر می شود و اگر یک داده شود , یک می شود و اگر هیچ مقداری داده نشود , دیگر معلوم نیست مقدار PINx چقدر خواهد بود , یعنی بسته به شرایط محیط ممکن است یک , یا صفر باشد .

سربرگ External IRQ :



در این بخش شما می توانید وقفه های خارجی را فعال کنید که پس از فعال کردن هر وقفه نوع حساسیت آن را نیز می توانید تعیین کنید . که وقفه های خارجی شماره صفر و یک می توانند حساس به سطح ,لبه , یا هر تغییری (یعنی حساس به لبه بالارونده و پایین رونده باهم) باشند و وقفه خارجی شماره ۲ تنها می تواند حساس به لبه بالارونده یا لبه پایین رونده باشد .

سربرگ Timers :



در این بخش شما می توانید تنظیمات مربوط به تایمرها را به راحتی انجام دهید . در بالای این سربرگ می توانید شماره تایمر را انتخاب کنید . گزینه دیگری به نام Watchdog نیز در انتهای این موارد دیده می شود که برای خودکار راه اندازی کردن میکروکنترلر پس از گیر کردن آن استفاده می شود و در این بخش به توضیح آن نمی پردازیم .

Clock Source : در این بخش نوع تایمر یا کانتر بودن Timer را مشخص می کنیم . در حالتی که در شکل نشان داده شده است گزینه انتخابی (System Clock) نشان می دهد که در وضعیت تایمر قرار داریم . گزینه های دیگر این بخش مربوط به پایه ورودی است که به ازای هر کلاک از آن پایه TCNT۰ یک شماره افزایش می یابد که در حقیقت کار یک شمارشگر را انجام می دهد . که می توانید تعیین کنید حساس به لبه بالارونده باشد یا لبه پایین رونده . در ادامه با فرض تایمر بودن پیش می رویم .

Clock Value : در این بخش شما می توانید فرکانسی که به تایمر می دهید را تنظیم کنید . دقت کنید که در اینجا حرفی از مقسم فرکانسی زده نشده است , بلکه تنها

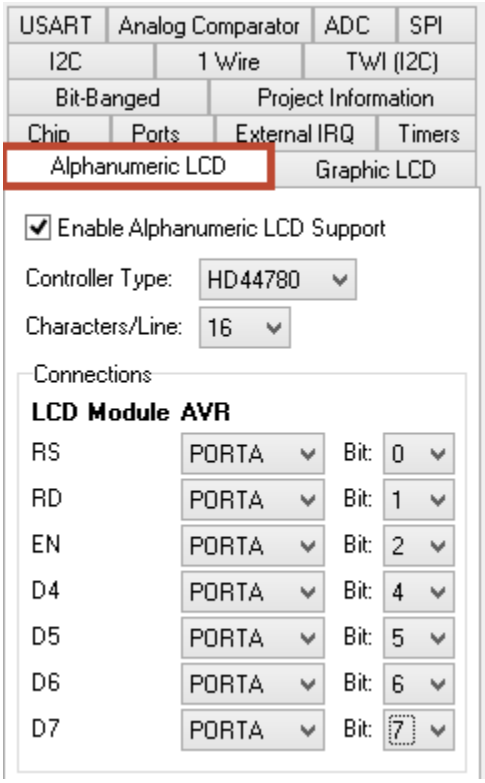
فرکانسها نوشته شده است . و این فرکانس ها در حقیقت همان فرکانس میکروکنترلر است که بر مقسم تقسیم شده است و مقدارآن نوشته شده است , لذا از لحاظ محاسباتی کمی عملیات را برای کاربر ساده کرده است .

Mode : در این بخش می توانید حالات تایمر خود را اعم از حالت نرمال , CTC , Fast PWM و PWM Phase Correct تعیین کنید . مقدار top=۰xFF که در شکل نیز نشان داده شده است بیشترین مقداری است که TCNT۰ می تواند در این حالت به آن برسد که در حالت نشان داده شده مقدار فعلی ۲۵۵ می باشد .

Output : این بخش در حقیقت تنظیمات مربوط به COM۰۰ و COM۰۱ را انجام می دهد که تعیین می کند پایه PB۳ یا همان OC۰ در حالت انتخابی چگونه عمل کند . که ۴ حالت برای آن وجود دارد که بسته به مد انتخابی عمل می کند .

وقفه: در پایین صفحه دو ناحیه وجود دارد که وقفه های مربوط به سرریز و تطبیق مقایسه را می توانید از روی آنها فعال کنید .

مقدار اولیه : در بخشهای Timer Value و Compare می توانید مقدار اولیه ثباتهای TCNT۰ و OCR۰ را تعیین کنید . علامت h نشان داده شده یعنی اینکه اعداد را باید به صورت هگز و در مبنای ۱۶ وارد کنید .



سربرگ Alphanumeric LCD :

این سربرگ مخصوص تنظیمات LCD های کاراکتری می باشد . منظوراز LCD های کاراکتری آن دسته از LCDهایی می باشند که در اصل برای فرستادن و چاپ کاراکترهایی خاص (شامل حروف الفبای انگلیسی و تعداد اشکال دیگر همانند '+' و '-' و '&' و ...) استفاده می شوند . که در مقابل این LCD ها , LCDهای گرافیکی می باشند که می توان گفت که توانایی چاپ هر تصویر دلخواهی را دارند .

برای نمایش تنظیمات در این بخش باید گزینه Enable Alphanumeric LCD Support را فعال کنید .

Controller Type : در این بخش نوع تراشه ای که در LCD موجود است را تعیین می کنید .چرا که در حقیقت درون LCD نیز یک میکروکنترلر وجود دارد که وظیفه نمایش و پیدا کردن کاراکترهای مشخص را دارد . در حالت معمول انتخاب گزینه HD۴۴۷۸۰ مورد مناسبی برای نمایش در محیط پروتئوس می باشد اگرچه در عمل باید یا نوع کنترلر LCD را پیدا کنید , یا از روش آزمایش و خطا بفهمید که کدام گزینه برای LCD شما مناسب است .

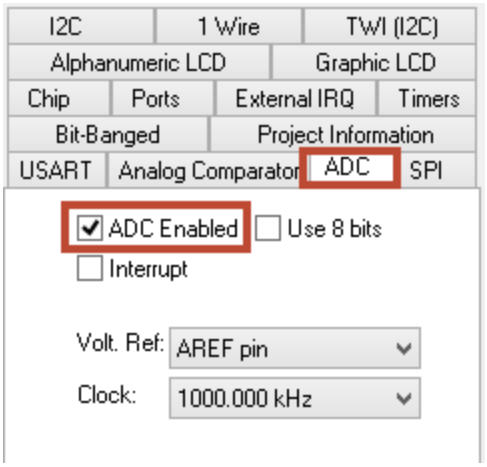
Characters/Line: در این بخش تعداد ستون LCD خود را تعیین می کنید که در موارد آزمایشگاهی از LCD های دارای ۱۶ ستون استفاده می شود .

Connections :

LCD Module AVR :

در این بخش پایه های LCD را تعیین می کنید که هر به کدام پورت متصل اند . دقت کنید که می توانید هر پایه LCD را تعیین کنید که به کدام پورت و به پین چندم آن متصل باشد که این طریقه انتخاب گرچه بسیار می تواند مفید باشد , اما در عین حال می تواند باعث اشتباه نیز بشود. شاید بهتر باشد به صورت پیش فرض مقادیر را برای کارهایی معمولی تغییر ندهیم و تنها پورت موردنظر را انتخاب کنیم . حتما دقت می شود که برای انتخاب پورت باید پورت تک تک بیتها را به پورت دلخواه تغییر دهید (در شکل تمام پایه های LCD به پورت A متصل هستند .)

سربرگ ADC :



این بخش مربوط به تنظیمات مبدل انالوگ به دیجیتال است . در صورتی که گزینه ADC Enabled غیر فعال باشد , هیچ کدام از تنظیمات نشان داده شده در شکل دیده نخواهد شد .

با فعال کردن این گزینه در حقیقت کدویژن بیت ۷ از ثبات ADCSRA را که همان بیت ADEN (ADC Enable) می باشد , یک می کند .

گزینه Use ۸ bits : با تنظیم این گزینه در برنامه باید از ثبات ADCH به جای ADCW استفاده کرد . و کدویژن بیت ADLAR را به طور خودکار یک قرار می دهد .

گزینه Interrupt : این گزینه برای فعال کردن وقفه تمام شدن تبدیل است . با فعال کردن این گزینه کدویژن زیر روال برنامه وقفه را به طور پیش فرض در بالای تابع اصلی ((Void main(void)) می سازد و همچنین مقدار بیت ۴ از ثبات ADCSRA را نیز که همان ADIE (ADC Interrupt Enable) است را یک می کند . (با انتخاب این گزینه تنظیمات دیگری نیز علاوه بر تنظیمات بالا نشان داده خواهند شد , که فعلا از توضیح آنها صرفنظر می کنیم .)

Volt. Ref : این بخش برای انتخاب مرجع ADC می باشد و دارای سه گزینه AREF pin , AVCC pin و Int.,cap. On AREF می باشد .

گزینه اول : منبع متصل شده به AREF را به عنوان مرجع در نظر می گیرد . (دقت داشته باشید که ولتاژ AREF باید کمتر از ۲ ولت و به طور تقریبی بیشتر از VCC نباشد.) .

گزینه دوم : ولتاژ AVCC را (که باید همان ولتاژ VCC متصل شده به پایه ۱۰ را با یک فیلتر پایین گذر به آن متصل کرد) به عنوان مرجع در نظر می گیرد .

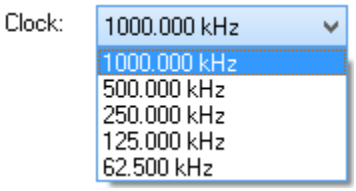
گزینه سوم : ولتاژ ۲,۵۶ ولت داخلی را به عنوان مرجع در نظر می گیرد . که عبارت Int به معنای Internal یعنی داخلی می باشد . و منظور از cap. on AREF بنظر این باشدکه در صورت استفاده از این حالت بهتر است به صورت خارجی پایه AREF را که به منبعی متصل نیست , با یک خازن به پایه AVCC متصل کنید . تا با این کار خطای در محاسبات ADC کمتر بشود . اگرچه این روش برای حالت گزینه دوم نیز در عمل بهتر است صورت گیرد .

درحقیقت با انتخاب این گزینه ها , کدویژن بر مبنای جدول زیر بیتهای REFx را در ثبات ADMUX مقدار دهی می کند.

Table 83. Voltage Reference Selections for ADC

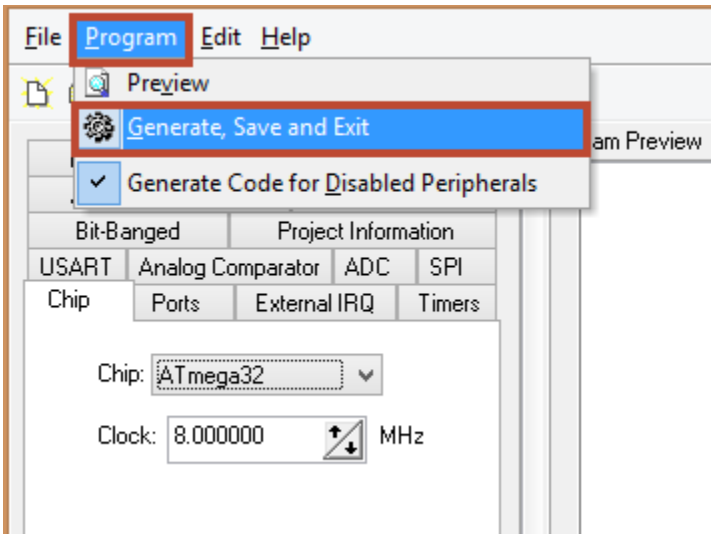
REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Clock : در این بخش شما می توانید فرکانس ورودی ADC را تنظیم کنید. دقت کنید که مقادیر نشان داده شده در شکل به ازای فرکانس میکروکنترلر برابر ۸مگاهرتز محاسبه شده است , اگر مقدار فرکانس را تغییر دهید, این اعداد نیز تغییر خواهند کرد . با تنظیم این گزینه , کدویژن مقادیر بیتهای ADPSx را که سه بیت اول ثبات ADCSRA می باشند را مقدار دهی می کند .



نکته : دقت کنید که تنظیم خاصی برای انتخاب کانال در این بخش موجود نیست و احتمالا تنها راه انتخاب کانال , با مقدار دهی دستی ثبات ADMUX صورت می پذیرد .

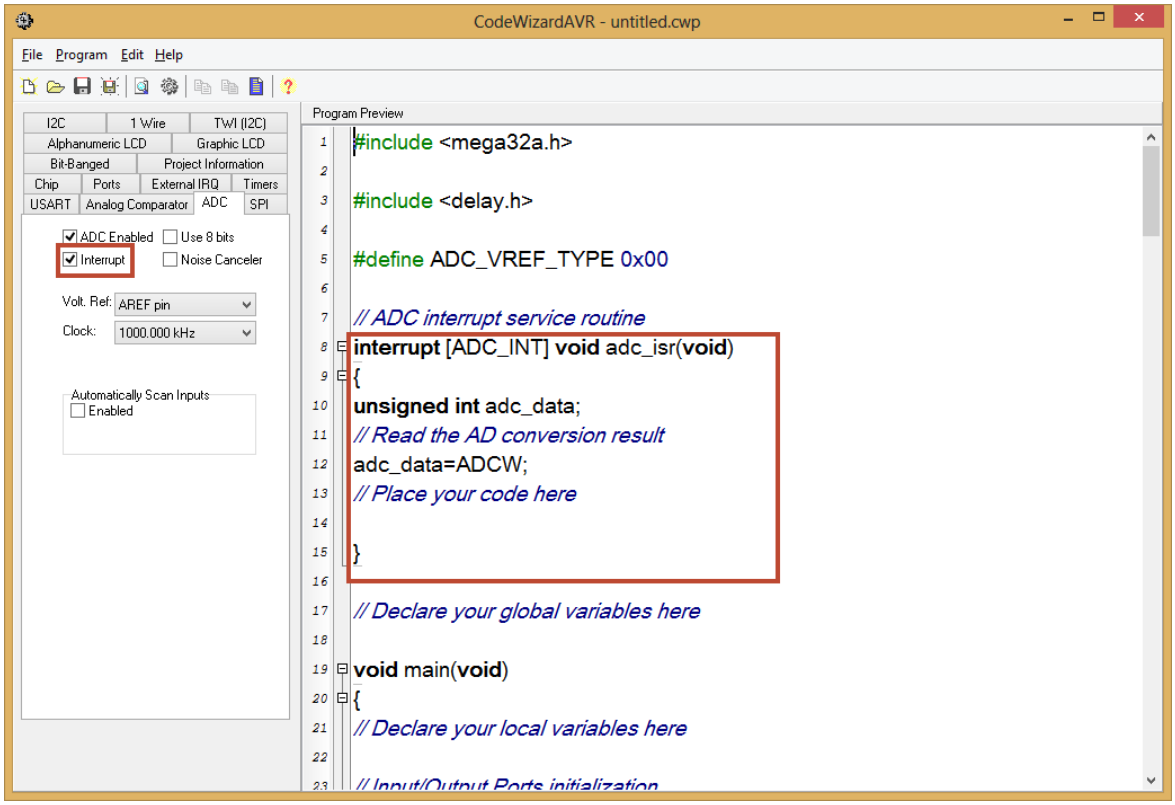
ذخیره سازی پروژه ساخته شده و شروع برنامه نویسی :



پس از آنکه تنظیمات پروژه کاری خود را به طور کامل انجام دادید , باید این تنظیمات در ثبات های مربوطه قرار گیرند , لذا باید در ابتدا پروژه کاری خود را در محل مورد نظر ذخیره کنید . برای این کار در همان پنجره CodeWizard از منوهایی که در بالای صفحه قرار دارد , در منوی Program بر روی گزینه Generate, Save and Exit کلیک کنید . در پنجره باز شده , به آدرسی که تمایل دارید در آن پروژه خود را ذخیره کنید بروید و نام پروژه خود را در قسمت File Name: بدهید و بر روی گزینه Save کلیک کنید. دقت کنید نام فایل سه بار از شما خواسته می شود, یک بار برای فایل *.c که حاوی برنامه شما است , بار دیگر برای فایل *.prj که فایل پروژه شما است . و دفعه سوم برای فایل *.cwp است که بنظر فایلی است که محیط CodeWizard برای خود می سازد .

گزینه Preview برای این است که قبل از ذخیره کردن پروژه یک پیش نمایشی از آن را در سمت راست سربرگها (قسمتی که فعلا خالی است) ببنید . این کار برای زمانی مناسب است که مثلا شما پروژه ای ساخته اید و در زمان ساخت آن وقفه ADC را فعال نکرده اید . در نتیجه برنامه زیر روال آن نیز برای شما ساخته نشده است . اکنون اگر بخواهید از وقفه ADC استفاده کنید چه کار خواهیدکرد؟ همانطور که می دانید نام زیر روالها بسیار طولانی و تا حدی پیچیده اند و لذا حفظ کردن آن ها نیز چندان کار مناسبی به نظر نمی رسد , بخصوص اینکه طریقه نوشتن زیر روالها در محیطهای برنامه نویسی مختلف (مثلا Atmel Studio) متفاوت است . یک روش مناسب شاید این باشد که مانند روش قبل برای ساخت یک پروژه جدید اقدام کنیم , منتها در اینجا هدف ساخت پروژه جدید نیست , بلکه این بار تنظیمات را طوری تنظیم می کنیم که محیط CodeWizard زیر روال وقفه را خودش بسازد , به این صورت که پس از انجام تنظیمات , این بار به جای انتخاب گزینه نشان داده شده در شکل بالا , گزینه Preview را بزنید . مشاهده می کنید که در سمت راست پیش نمایشی از محیط برنامه نویسی نمایش داده می شود . از این بخش می توانید به راحتی زیر روال وقفه ADC یا هر وقفه دیگری همانند وقفه سر ریز یا تطبیق مقایسه را کپی کرده و پنجره را ببندید , اکنون کافی است این زیر روال را بالاتر از تابع اصلی در محیط پروژه قبلی خود کپی کنید .

در تصویر زیر زیر روال وقفه ADC مشخص شده است . وتنها در این حالت کافیست گزینه Interrupt را که در شکل معلوم شده است تیک بزنید و به بقیه تنظیمات کاری ندارید (چرا که شما فقط اسم تابع زیر روال را , یعنی interrupt [ADC_INT] void adc_isr(void) را نیاز دارید و به بقیه موارد نیازی ندارید.


CodeWizardAVR - untitled.cwp
File Program Edit Help
I2C
1 Wire
TWI (I2C)
Alphanumeric LCD
Graphic LCD
Bit-Banged
Project Information
Chip
Ports
External IRQ
Timers
USART
Analog Comparator
ADC
SPI
ADC Enabled
Use 8 bits
Interrupt
Noise Canceler
Volt. Ref:
AREF pin
Clock:
1000.000 kHz
Automatically Scan Inputs
Enabled
Program Preview
#include <mega32a.h>
#include <delay.h>
#define ADC_VREF_TYPE 0x00
// ADC interrupt service routine
interrupt [ADC_INT] void adc_isr(void)
{
unsigned int adc_data;
// Read the AD conversion result
adc_data=ADCW;
// Place your code here
}
// Declare your global variables here
void main(void)
{
// Declare your local variables here