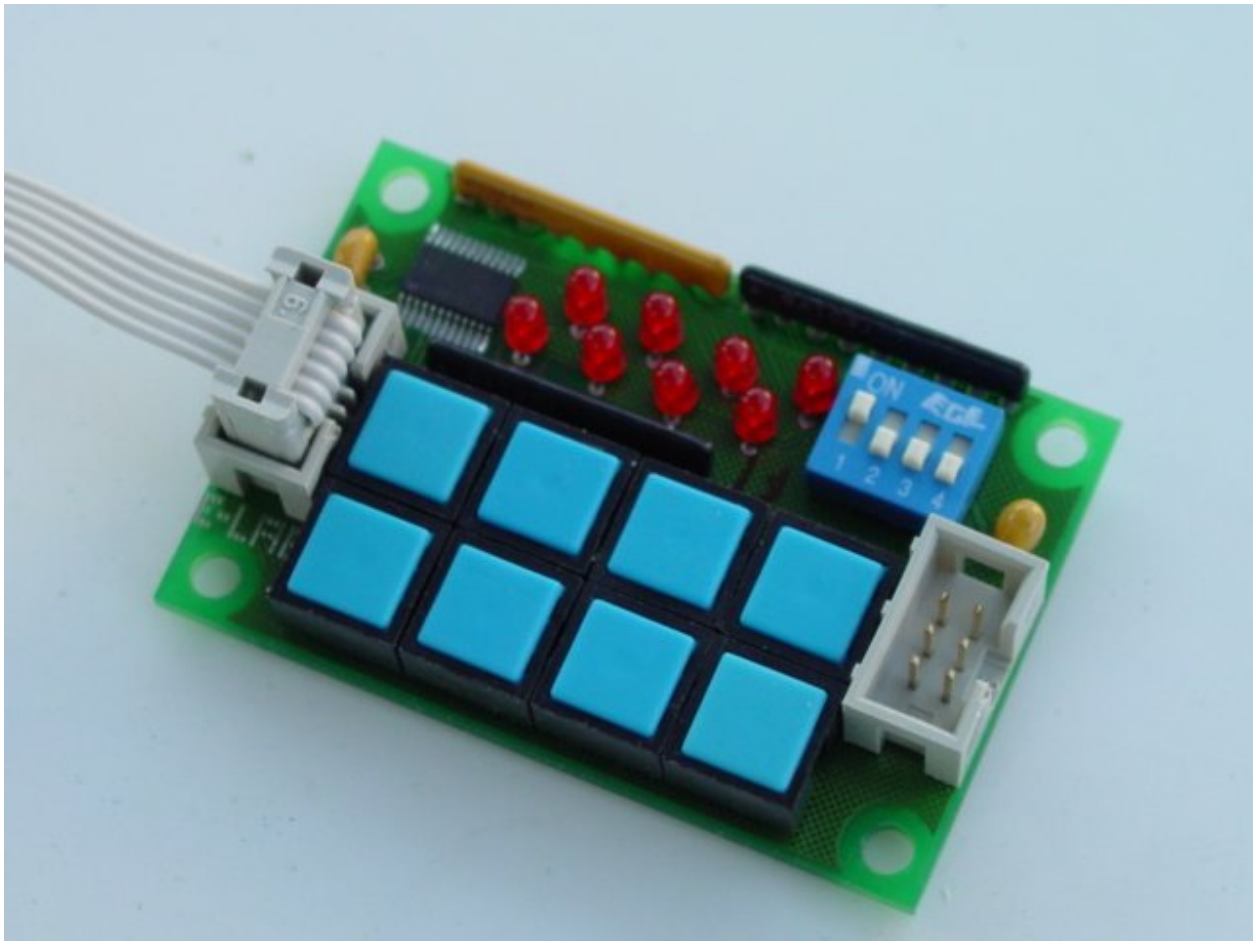
به نام خدا

(آموزش AVR جلسه دوم)

(Bascom)



مقدمه:

در جلسه قبل به معرفی میکروکنترلرهای AVR و توضیح پورت های ورودی/ خروجی پرداختیم. سپس قطعات مورد نیاز برای پروژه اول را معرفی کرده و یک مدار نمونه با حداقل قطعات بر روی برد بورد بستیم و در نهایت به معرفی محیط های برنامه نویسی AVR پرداختیم. در این جلسه تصمیم داریم اولین برنامه برای مدار

بسته شده را که یک چشمکزن ساده می باشد شرح دهیم و در نهایت برنامه نحوه خواندن ورودی (کلید) را بنویسیم.

Bascom AVR

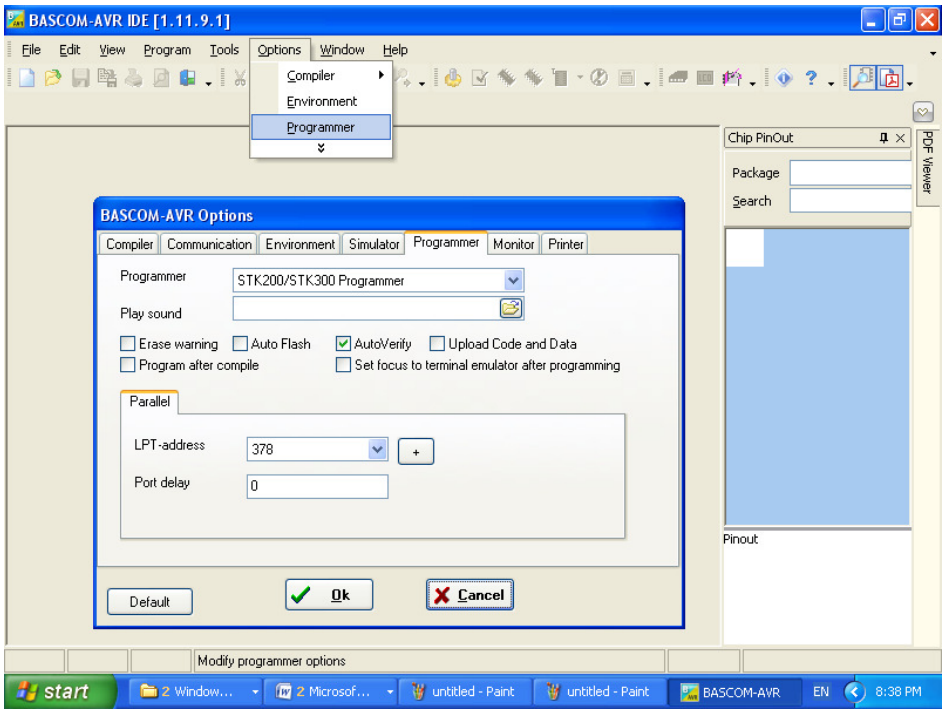
همانطوریکه در جلسه قبل ذکر کردیم به دلیل اینکه در این سری مقالات سطح تمام خوانندگان در نظر گرفته می شود نحوه کد نویسی نیز با برنامه های ساده شروع شده و به برنامه های حرفه ای ختم می گردد. اولین برنامه ای که برای پروژه چشمکزن می نویسیم در محیط بسکام (بیسکام) کد نویسی می شود. زبان این برنامه BASIC استاندارد بوده و شباهت بسیار زیادی به QBASIC و VB دارد. این برنامه رایگان نبوده و نسخه اصلی آن حدود 134 هزار تومان قیمت دارد. ما در اینجا برای آموزش از نسخه DEMOی این برنامه استفاده می کنیم که تا 4 کیلوبایت کد را پشتیبانی می کند و تمام ویژگیهای نسخه اصلی را دارا می باشد. بسکام یک کامپایلر سطح بالا (HLL) بوده و دارای دستورات و توابع متنوعی می باشد که کار برنامه نویسی را راحت تر کرده و کاربر را از جزئیات سخت افزار و رجیسترهای داخلی میکرو دور می کند.

بسکام توسط سایت www.mcselec.com ارائه و پشتیبانی می شود. می توانید آخرین نسخه آزمایشی آن را از سایت مربوطه دانلود کرده و یا از بخش دانلود سایت AVR64 به نشانی زیر نسخه آزمایشی BASCOM 1.11.9.1 DEMO را دریافت نمایید: www.avr64.com

پس از دانلود برنامه که یک فایل 20 مگابایتی می باشد آن را نصب کنید. مراحل نصب برنامه ساده بوده و پس از چند بار زدن کلید Next برنامه نصب می شود. این برنامه معمولا در آدرس زیر نصب می شود:
C:\Program Files\MCS Electronics\BASCOM-AVR

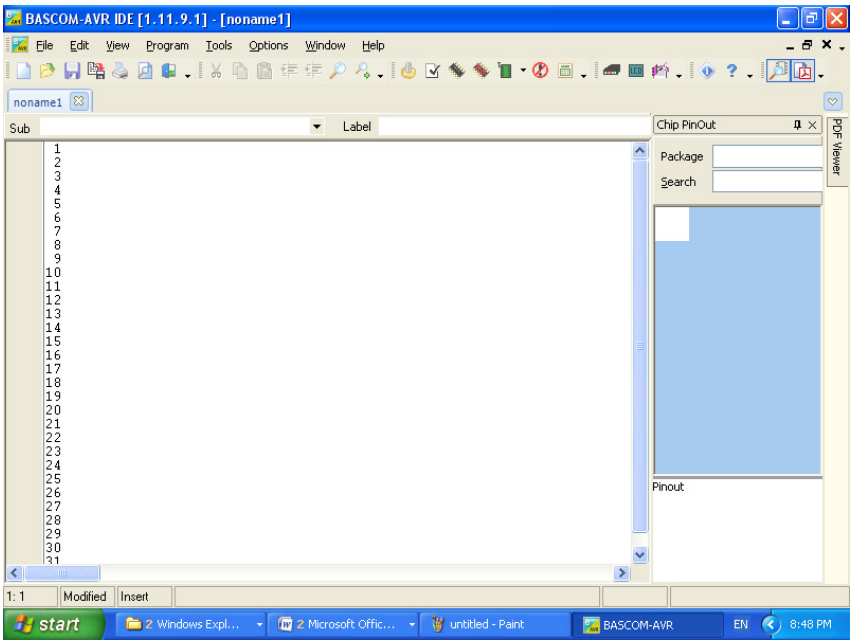
پس از نصب برنامه روی 

کلیک کنید تا محیط برنامه مطابق شکل صفحه بعد باز شود. در این محیط یک سری تنظیمات اولیه را بایستی انجام دهید که اولین تنظیم مربوط به تعریف نوع پروگرمر می باشد. در صورتی که پروگرمر STK200/300 یا Poverty Prog را خریداری کرده اید مطابق شکل صفحه بعد از منوی Options روی Programmer کلید کرده و در پنجره جدید باز شده در سربرگ Programmer گزینه STK200/STK300 Programmer را انتخاب نمایید.



شکل ۱- معرفی پروگرمر به محیط Bascom

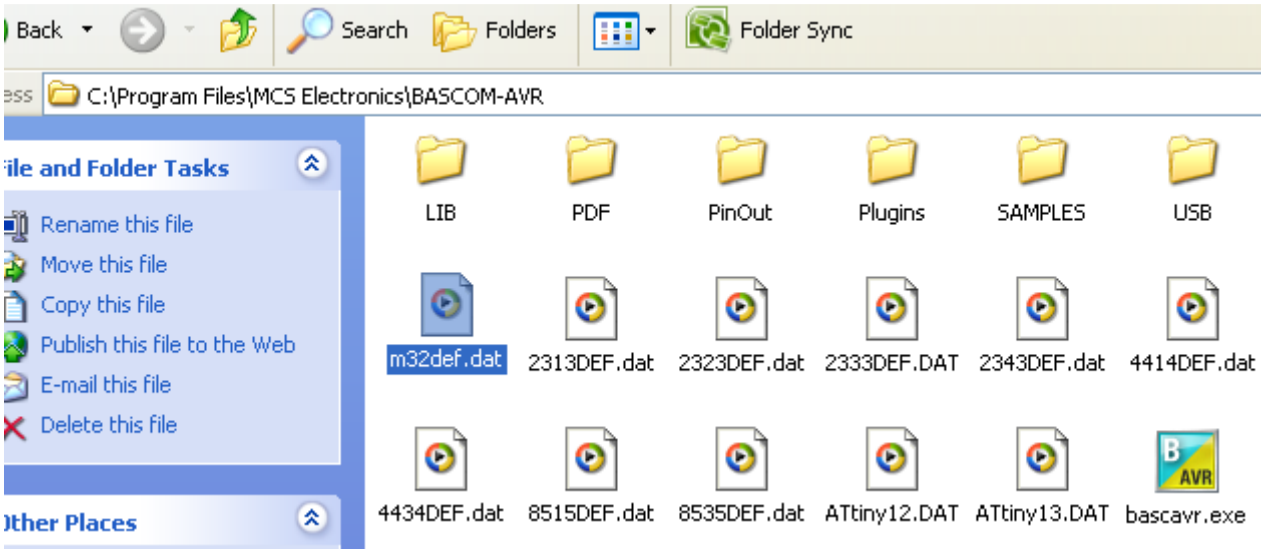
پس از معرفی پروگرمر روی OK کلیک کنید تا انتخاب انجام شده ثبت شود. سپس از منوی File روی New کلیک کرده و یا روی  کلیک نمایید تا یک پنجره جدید باز شود و بتوانید کدها را در آن وارد کنید:



شکل ۲- ایجاد فایل جدید

در محیط بیسکام یک سری از دستورات که با علامت $ شروع می شوند فقط به عنوان راهنمای استفاده از منابع و تنظیمات داخلی به شمار می روند و به اصطلاح رهنمودهای کامپایلر هستند. دو نمونه از این دستورها $regfile و $crystal می باشند که اولی برای معرفی نوع میکرو و دومی برای معرفی فرکانس کاری CPU استفاده می شود. با توجه به اینکه ما در اینجا از میکروی ATmega32 استفاده می کنیم مقدار اولی را برابر با m32def.dat قرار می دهیم و مقدار فرکانس CPU میکرو را هم به صورت پیش فرض 1000000 هرتز برابر با یک مگاهرتز تنظیم می کنیم. (این مقدار به صورت پیش فرض توسط فیوزبیت داخلی در کارخانه تنظیم می شود و نوسانساز RC داخلی 1MHz فعال بوده و نیازی به نصب کریستال در خارج از میکرو نمی باشد. نکته قابل توجه یکسان بودن عدد تنظیم شده در مقابل دستور $crystal وعدد تنظیم شده در بخش فیوزبیت ها می باشد که در صورت عدم تطابق مدت زمان اجرای تمام دستورات بهم می خورد. این تنظیمات در بخش بعدی به طور مفصل شرح داده خواهد شد). اما عبارت m32def.dat برای ATmega32 از کجا آمد؟ و برای سایر میکروها از چه عبارتی باید استفاده کنیم؟ اگر به محل نصب برنامه که عموماً در آدرس زیر است نگاه کنید فایل هایی تحت عنوان m32def، 2313DEF و... را مشاهده می کنید که هر یک حاوی آدرس های رجیسترها، پورت ها و سایر بخش های هر میکرو هستند و نام آن فایل شباهت زیادی به نام درج شده روی میکرو دارد. می توانید نام فایل را از پوشه محل نصب برنامه بیابید و در ابتدای برنامه در دستور $regfile به صورت $regfile="m32def.dat" درج کنید.

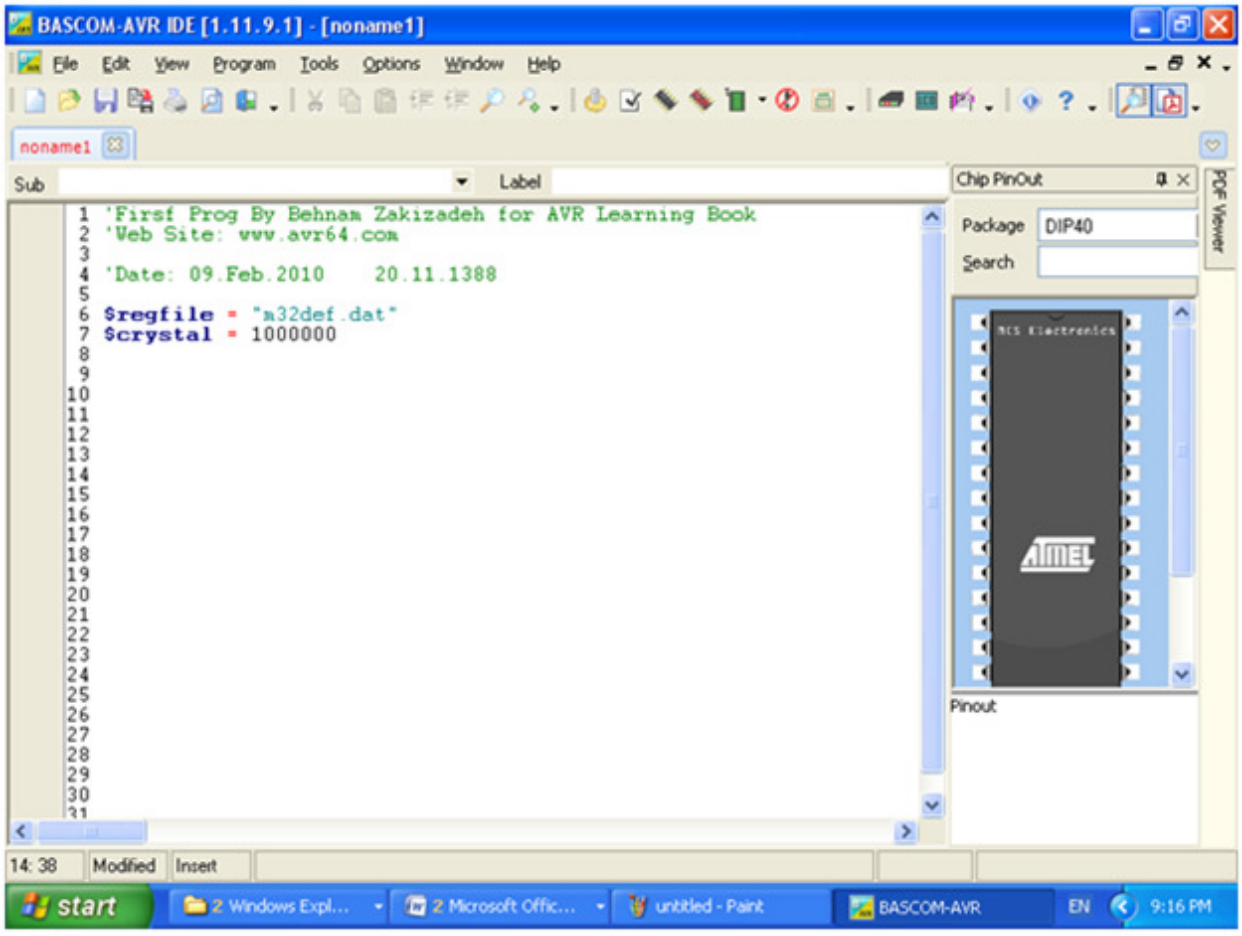
C:\Program Files\MCS Electronics\BASCOM-AVR



شکل ۳- فایل های مشخص کننده نوع میکرو

پس تا اینجا دو دستور مهم و حیاتی در یک برنامه نوشته شده به زبان بیسیک را آموختیم. این دستورات را مطابق شکل زیر در پنجره کد نویسی وارد کنید. به محض وارد کردن دستور اول شکل ظاهری میکرو در سمت راست پنجره نمایان می شود و با انتخاب هر کدام از پایه ها نام و کاربرد آن پایه در زیر شکل نوشته می شود. این قسمت در بسکام سری قدیم موجود نبود ولی در ورژن جدید اضافه شده و کار طراح را ساده تر می کند. اگر نتوانستید این پنجره را ببینید از منوی View روی Pin Layout کلیک کنید.

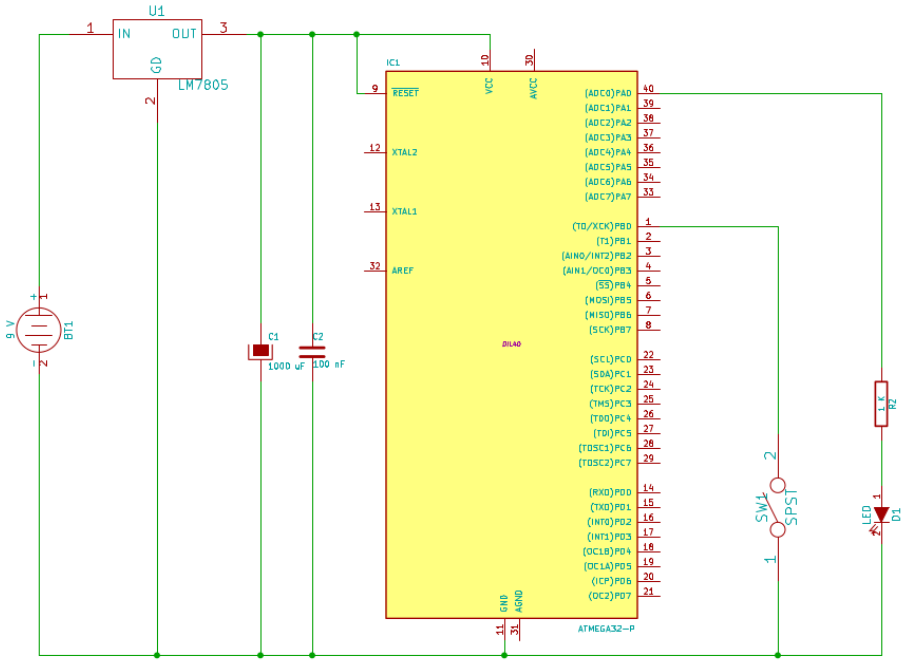
نکته مهم دیگر نوشتن توضیحات در ابتدای برنامه می باشد. با قرار دادن یک ' در ابتدای خط، کل خط به رنگ سبز درآمده و به عنوان توضیح به حساب می آید. همیشه در هر برنامه ای نام خود و اطلاعات تماس و نیز تارخ ساخت آن را بنویسید. این اطلاعات بهتر است به صورت بین المللی باشد تا اگر کد نوشته شده توسط شما به هر صورت پخش شد کاربر جدید بتواند با شما تماس برقرار کند و سوالات خود را بپرسد.



شکل ۴- شروع کد نویسی

قبل از تایپ سایر دستورات، برنامه نوشته شده را Save کنید. برای این منظور از منوی File روی Save As... کلیک کرده و برنامه را با نام BlinkLED در یک پوشه دلخواه ذخیره کنید. حتماً برنامه را در یک پوشه ذخیره کنید چرا که تعداد زیادی فایل تولید می شود که برای جابجایی برنامه بهتر است همه فایل ها را با هم جابجا کنید. اگر به پیغامی در مورد CFG File برخوردید Yes را انتخاب کنید.

در ادامه بایستی پورت های ورودی/ خروجی را تعریف کنیم. نگاهی به نقشه جلسه پیش می اندازیم:



شکل ۵ - نقشه پروژه چشمکزن

در این نقشه مشاهده می کنیم که LED به پایه شماره ۴۰ یعنی PA0 متصل شده است. قرار است که با 0 و 1 کردن این پایه از طریق برنامه میکرو، LED را روشن و خاموش کنیم پس این باید به عنوان خروجی پیکره بندی شود. پورت ها می توانند به صورت ورودی یا خروجی پیکره بندی شوند. هر پورت دارای سه رجیستر به نام های DDR، PORT و PIN می باشد. رجیستر DDR برای تنظیم ورودی و خروجی است. رجیستر PIN برای خواندن مقدار پین ها در حالت ورودی و در نهایت رجیستر PORT برای نوشتن در پایه ها در حالت تنظیم خروجی به کار می رود. (اگر پورتی را به عنوان ورودی تعریف کنیم و در رجیستر PORT عدد 1 را قرار دهیم به معنای اتصال مقاومت PullUp داخلی می باشد). برای تنظیم پایه شماره 0 پورت A به عنوان خروجی باید رجیستر DDR شماره 0 پورت A را 1 کنیم. 1 به معنای تنظیم به صورت خروجی و 0 به معنای تنظیم به صورت ورودی می باشد.

```
ddra = &B00000001
```

&Bبه معنای باینری می باشد و چون پورت های AVR هشت است عدد مقابل آن هم بایستی 8 رقم باشد. سمت راست ترین رقم همان پایه شماره 0 یا کم ارزش ترین بین محسوب می شود. سمت چپ ترین بیت نیز پایه شماره 7 یا پرارزش ترین بیت به حساب می آید. از علامت H& نیز می توانید برای نوشتن عدد به صورت هگزادسیمال استفاده کنید .

در صورتی که فقط مایلید تا یکی از پایه های پورت موردنظر را به عنوان ورودی یا خروجی تنظیم کنید و به پایه های دیگر کاری نداشته باشید می توانید با یک نقطه ایندکس پایه مورد نظر را تنظیم کنید. در این صورت عدد سمت راست علامت مساوی بایستی فقط 0 یا 1 و بدون هیچ علامتی باشد. برای مثال در دستور زیر پایه شماره 0 پورت A را به عنوان خروجی تعریف می کنیم :

```
ddra.0 = 1
```

در بسکام یک دستور مشابه نیز برای تنظیم ورودی/ خروجی به کار می رود که متداول تر است، ولی معمولاً باعث اشتباه مبتدیان می شود. نام این دستور configبوده و در صورتی که مثلاً به صورت config porta = output یا config pina.0 = output به کار رود می تواند دستورهایی تنظیم DDR را شبیه سازی کند. در این دستور نکات ظریفی نهفته است که در صورت رعایت نکردن می تواند منجر به اشباه شود. اشتباهاتی که در هنگام کامپایل مشاهده نمی شود ولی در عمل می بینیم که برنامه کار نمی کند و یا نتایج غلط می دهد .

یکی از نکات مهم دستور config اینست که موقع ترکیب با نام یک پورت مثلاً config porta :منظور از portaهمان رجیستر PORT نیست و منظور DDRA می باشد و دستوری مشابه config porta = output برابر با چیزی شبیه DDRA = &B11111111 :است! (یا DDRA = 255 یا .(DDRA = &HFF و نیز با دستور Config portaفقط می توانیم کل پورت را به صورت ورودی و خروجی تعریف کنیم و قادر نیستیم با همین دستور پایه ها را به صورت مختلف ورودی و خروجی کنیم .با این حال برای تنظیم پایه ها فقط به صورت تکی می توانیم از دستور دیگری مثل config pina.0 = output :بهره بگیریم. در ادامه بیشتر با دستورات Configکار می کنیم. فقط توجه داشته باشید که برای تنظیم کل یک پورت به ورودی و خروجی باید از نام PORTاستفاده کنیم و برای پایه ها از نام PIN و این PORT و PIN هیچ ارتباطی با رجیسترهای PORT و PINندارند و اشاره ای DDR و DDR.X هستند .

config pina.0 = **output**

نکته قابل توجه در تنظیم پورت ها اینست که برای تعریف هر پایه به صورت جدا بایستی از عبارت Pin استفاده کنیم ولی در صورتی که بخواهیم کل پورت را به عنوان ورودی/ خروجی تعریف کنیم از عبارت Port استفاده می کنیم. مثلاً دستور زیر کل پورت A را به صورت خروجی تعریف می کند (پایه های ۳۳ تا ۴۰):

```
Config PORTA = Output
```

پس از تنظیم پورت ها نوبت به بدنه اصلی برنامه می رسد. در بدنه برنامه بایستی پین شماره صفر پورت A را با فرکانس ۱ هرتز روشن و خاموش کنیم. منظور از یک هرتز یعنی LED در هر ثانیه یک بار روشن و خاموش شود و به عبارتی دوره تناوب روشن و خاموش شدن LED یک ثانیه طول بکشد. این بدین معنی است که LED نیم ثانیه خاموش باشد و نیم ثانیه روشن. تصمیم داریم این ماجرا تا ابد تکرار شود پس کل برنامه را در یک حلقه بی نهایت قرار می دهیم. این حلقه با Do و Loop محصور می شود. در آخر برنامه هم دستور End را قرار می دهیم.

دستور زیر پورت متصل شده به LED را ۱ می کند و سبب روشن شدن LED می شود:

```
Set Porta.0
```

دستور زیر هم همین کار را انجام می دهد:

```
Porta.0 = 1
```

دستور زیر پورت متصل شده به LED را 0 می کند و سبب خاموش شدن LED می شود:

```
Reset Porta.0
```

دستور زیر هم همین کار را انجام می دهد:

```
Porta.0 = 0
```

دستور زیر ۵۰۰ میلی ثانیه (نیم ثانیه) تاخیر ایجاد می کند:

```
Waitms 500
```

سایر دستورات ایجاد تاخیر عبارتند از Wait، Waitms وWaitus که اولی بر حسب ثانیه، دومی بر حسب میلی ثانیه و سومی بر حسب میکرو ثانیه تاخیر ایجاد می کنند. این دستورات زیاد دقیق نیستند و برای مواردی که زمان دقیق در آنها اهمیت چندانی ندارد (مثل چشمکزن) مورد استفاده قرار می گیرند. برای ایجاد زمان دقیق از تایمرها استفاده می شود که بعداً شرح داده می شود. لازم به ذکر است که می توان جلوی عبارت های Wait و Waitms و Waitus متغیر قرار داد و مقدار آنرا در حین اجرای برنامه تغییر داد.

نکته قابل توجه در مورد عملکرد پورت ها اینست که برای ۰ و ۱ کردن یک پایه باید در رجیستر PORT آن بنویسیم هر چند که PIN را به عنوان ورودی/ خروجی تعریف کرده باشیم. ضمناً هرگز مجاز نیستیم که مثلاً PORTA.0 را به عنوان ورودی یا خروجی تعریف کنیم. عبارتPORT فقط زمانی به عنوان ورودی خروجی تعریف می شود که هر ۸ پایه مد نظر باشند. در مثال بعد مشاهده می کنید که برای خواندن کلید باید از رجیستر PIN بخوانیم و اگر مقدار PORT را بخوانیم مقدار ذخیره شده در داخل میکرو را تحویل می دهد و هر چقدر پایه را از خارج ۰ و ۱ کنیم تغییری مشاهده نمی کنیم. پس هنگام تعریف ورودی یا خروجی اگر پایه تکی مد نظر بود Pin را تعریف می کنیم و اگر کل پورت مد نظر بود PORT را تعریف می کنیم. ولی در برنامه موقع نوشتن در پورت چه تکی، چه کل پورت باید در رجیستر PORT بنویسیم و موقع خواندن از پورت چه تکی چه کل پورت باید از رجیستر PIN بخوانیم. در مثال های آینده با عملکرد پورتها بیشتر آشنا می شوید.

دستورات زیر همگی درست اند:

```
'Output نوشتن
Config Pina.0 = Output
Config Portb = Output

Set Porta.0
Porta.0 = 1
Reset Porta.0
Porta.0 = 0
Portb = &B11001110مقدار باینری
```

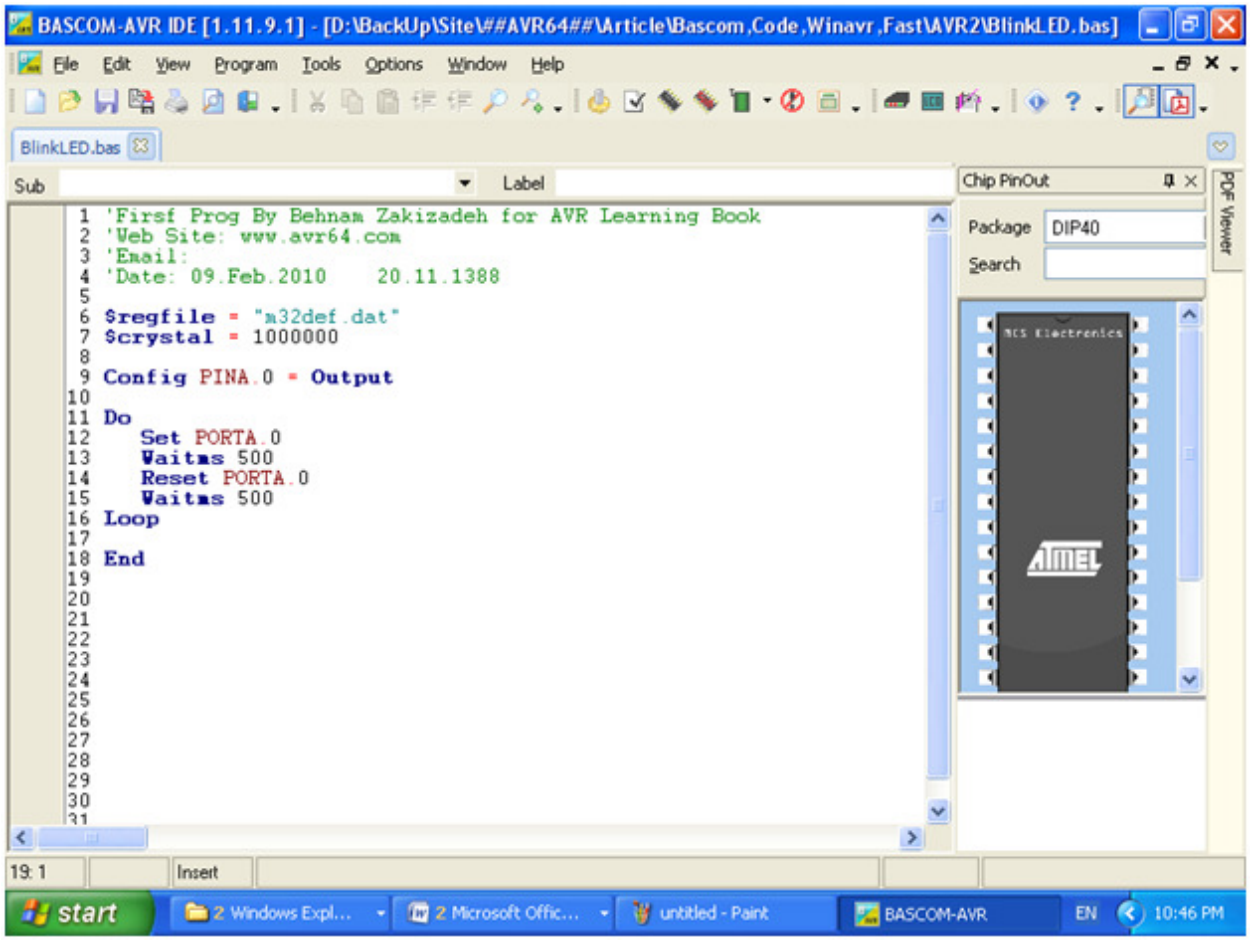
```
Portb.0 = 1
Portb.5 = 0
Portb = 206 مقدار دهدهی
Portb = &HCE مقدار هگز

'Input خواندن
Config Pinc.0 = Input
Config Portd = Input
X = Pinc.0 یک متغیر از نوع بایت می باشد x
X = Pind
X = Pind.5
```

دستورات زیر کاملاً اشتباه اند:

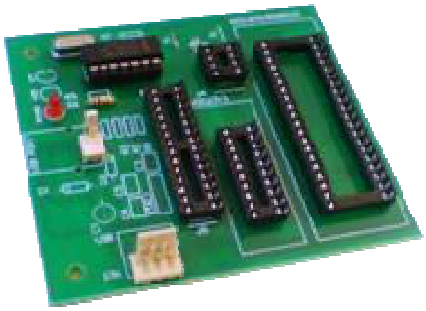
```
Config Porta.0 = Output
Config Porta.0 = Input
X = Porta.1
X = Porta
Set Pina.0
Set Pina
```

برنامه کامل چشمکزن در شکل زیر آورده شده است. در این برنامه دستورات روشن و خاموش کردن LED داخل حلقه Do . . Loop آورده شده اند و به طور مرتب تا ابد تکرار می شوند.



شکل ۶ – برنامه کامل پروژه چشمکزن

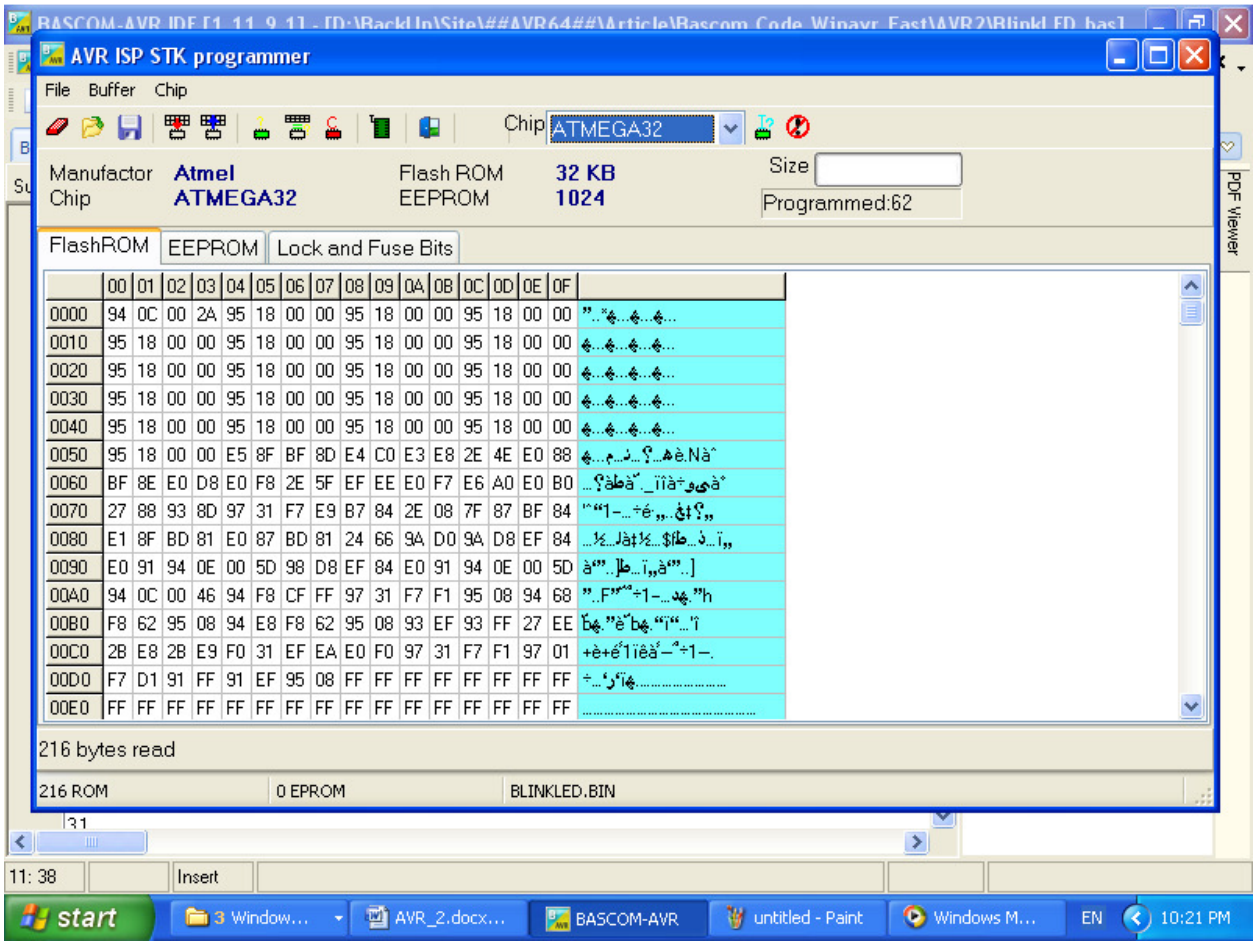
برای کامپایل کردن برنامه کلید F7 را زده و یا بر روی  کلیک کنید. در صورتی که برنامه خطا نداشته باشد با موفقیت کامپایل می شود و درصد حافظه ای را که اشغال کرده است به شما نشان خواهد داد. با توجه به اینکه میکروکنترلر ATmega32 به میزان 32 کیلوبایت حافظه دارد، مقدار درصد مصرف شده از 32KB محاسبه می شود.



برای ارسال برنامه به میکرو ابتدا آی سی را داخل سوکت مربوطه قرار داده و هر دو پورت LPT و USB را به کامپیوتر وصل کنید سپس کلید F4 را زده یا اینکه بر روی علامت  کلیک کنید؛ در این حالت پنجره Programmer باز می شود که کد Hex تولید شده را نمایش می دهد. در صورتی که قبل از مشاهده این پنجره پیغام های زیر را مشاهده کردید پورت LPT پروگرمر به کامپیوتر وصل نشده و یا آن را برای بسکام تعریف نکرده اید. همچنین ممکن است آدرس پورت پرینتر را درست نداده باشید (378) و یا میکرو معیوب باشد یا تغذیه USB قطع باشد.

شکل ۷- پیغام های خطای پروگرمر

در صورتی که مشکلی نباشد پنجره پروگرمر به صورت زیر باز می شود:

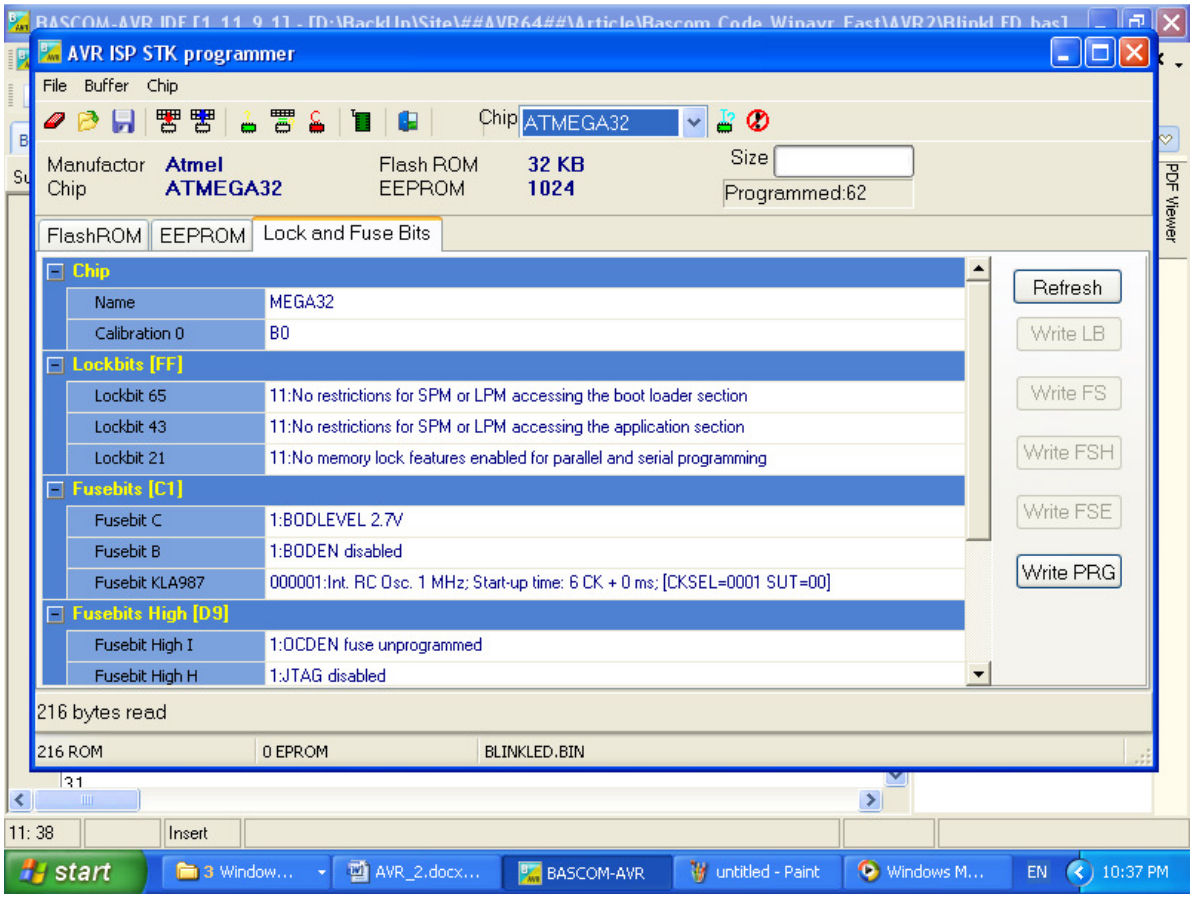


شکل ۸- پنجره پروگرمر

در این پنجره با زدن کلید می توانید برنامه را روی آی سی پروگرم کنید. توجه داشته باشید که در حال حاضر داخل این پنجره کد هگز حاصل از پروژه نوشته شده با بسکام قرار دارد ولی با زدن کلید می توانید این محتویات را پاک کنید و با زدن کلید کد هگز دلخواه خود را باز کنید و با زدن کلید برنامه را داخل میکرو Upload نمایید.

این مطالب مربوط به حافظه Flash میکرو بود که برنامه نوشته شده به زبان ماشین تبدیل شده و در آن قرار می گیرد. این حافظه از نظر میکرو فقط خواندنی خواهد بود و میکرو قدرت نوشتن روی آن را ندارد (مگر با Boot Loader که فعلاً شرح داده نمی شود). سربرگ بعدی مربوط به حافظه EEPROM است که یک حافظه پایدار بوده و میکرو قدرت خواندن و نوشتن آن را دارد و با قطع برق پاک نمی شود. می توان در سربرگ EEPROM در این حافظه اطلاعات ثابتی را نوشت و یا مقدار بایت های آن را تغییر داد. معمولا در پروژه های حرفه ای مثل قفل رمز دیجیتال رمز پیش فرض را در این بخش می نویسند.

اما سربرگ آخر مربوط به فیوز بیت ها و بیت های قفل میکرو بوده و با تنظم آن ها می توان بخش هایی از میکرو را فعال یا غیر فعال کرد، همچنین می توان میکرو را قفل کرد تا برنامه نوشته شده توسط دیگران کپی نشود. بخش مهم این قسمت فیوربیت کلاک میکرو است که می توان با تنظیم آن سرعت کار میکرو را کم و زیاد کرد و نوسان ساز داخلی RC و یا نوسان ساز کریستال خارجی و.. را تنظیم نمود. در اینجا مهمترین فیوزبیت ها را به ترتیب شرح می دهیم.



شکل ۹- فیوزبیت ها

سه فیوزبیت اول که به عنوان Lock bits نام گزاری شده اند و هر سه به صورت پیش فرض روی 11 تنظیم شده اند مربوط به قفل حافظه فلش میکرو هستند. در صورتی که روی هر کدام کلیک کرده و مقدار هر سه را به 00 تغییر دهید و سپس روی کلید Write LB (که فعال خواهد شد) کلیک کنید حافظه میکرو را از کپی دیگران محفوظ کرده اید. البته در صورتی که فقط سومی را هم روی 00 قرار دهید کفایت می کند ولی بهتر است که دو مورد بالا را هم روی 00 قررا دهید تا از طریق Boot Loader هم نتوان به حافظه فلش دست پیدا کرد. (برای باز کردن این قفل کافیست تا یک فایل جدید را روی میکرو فلش کنید، البته فایل قبلی از بین می رود ولی میکرو قابل استفاده می شود)

از سه فیوز بیت بعدی که تحت عنوان Fuse Bits آمده اند، دو فیوز بیت اول مربوط به عملکرد ریست خودکار میکرو پس از رسیدن به حداقل ولتاژ خاص می باشد. در صورتی که فیوز اولی که BODLEVEL نام دارد روی 2.7v تنظیم شده باشد و ولتاژ تغذیه از ۲/۷ ولت کمتر شود میکرو ریست می شود و اگر روی ۴ ولت تنظیم شده باشد با کمتر شدن ولتاژ از ۴ ولت میکرو ریست خواهد شد. البته این در صورتی است که فیوزبیت بعدی یعنی BODEN روی Enabled باشد. این عملکرد مثل قضیه Low Battery دستگاه های الکترونیکی است که با کم شدن ولتاژ خاموش می شوند. دلیل آن هم اینست که مثلاً شاید یک دوربین دیجیتال با ولتاژ 3.5 ولت هم کار کند و بخش پردازش آن هیچ مشکلی نداشته باشد ولی ماژول دوربین ماکسیمم کیفیت خود را در این ولتاژ نداشته باشد و عکس های با کیفیت بد تهیه کند. یا GSM MODEM یک گوشی موبایل در ارسال و دریافت SMS دچار خطا شود و کاراکترها را اشتباهی دریافت کند؛ پس بهتر است که خاموش شود و اصلاً کار نکند!

آخرین فیوز بیت گروه بالا که فوق العاده مهم است و عموماً به آن توجهی نمی شود مربوط به انتخاب منبع کلاک میکرو و میزان آن است. این فیوز بیت به طور پیش فرض روی Int RC 1MHz قرار دارد که به معنی اسیلاتور داخلی RC (مقاومت خازن) یک مگاهرتز است. در این صورت باید در برنامه عبارت `$crystal=1000000` را بنویسیم. در نمونه میکروی m32 چند گروه از فیوز بیت های فرکانس داخلی وجود دارند که گروه زیر مد نظر است:

شکل ۱۰- فیوزبیت های کلاک

در این گروه در جلوی انتخاب پیش فرض عبارت default value نوشته شده است و اگر دقت کنید زمان start-up هر ۴ مقدار آن ۶۴ میلی ثانیه است. معنی آن اینست که پس از وصل تغذیه به میکرو ۶۴ میلی ثانیه تاخیر ایجاد می شود تا نویز های احتمالی برق گرفته شوند و سپس CPU کار خود را شروع می کند. دقیقاً مثل محافظ یخچال که با وصل برق حدود ۴ دقیقه صبر می کند تا ولتاژ برق شهر تثبیت شود. مقادیر معتبر بعدی ۲، ۴ و ۸ مگاهرتز هستند. در این صورت می توانید دستورات `$crystal=1000000` یا `$crystal=2000000` یا `$crystal=4000000` و `$crystal=8000000` را بدون هیچ مشکلی در آغاز برنامه به کار ببرید به شرطی که متناسب با آن فیوز بیت را تنظیم کرده و روی کلید Write FS کلیک کنید.

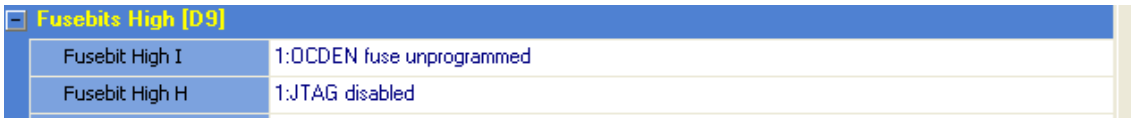
انتخاب بعدی عبارتست از کلاک خارجی که با Ext Clock نشان داده شده است که در صورت استفاده از آنها بایستی کلاک خود را به پایه XTAL1 میکرو وصل کرده و متناسب با میزان کلاک مقدار آن را از این منو انتخاب کنید. این منبع بیشتر در کارهای حرفه ای کاربرد دارد؛ مثلاً ارتباط میکرو با یک ماژول ویدئویی که با فرکانس `17.734475MHz` کار می کند و خروجی فرکانس می دهد.

انتخاب بعدی مربوط به نوسان ساز خارجی مقاومت و خازن است که با Ext RC نمایش داده می شود، این انتخاب محاسبات خاص خود را دارد و در صورتی که بخواهید محصولی با قیمت پایین طراحی کنید که با فرکانس خاصی کار کند و در آن محدوده کریستال وجود نداشته باشد (مثلا ۵ مگاهرتز) می توانید از این انتخاب استفاده کنید. در کل استفاده چندانی از این بخش نمی شود و معمولا در اکثر پروژه ها از RC داخلی استفاده می شود.

انتخاب آخر که فوق العاده مهم و کاربردی است گزینه کریستال خارجی می باشد. این گزینه با Ext Crystal نمایش داده می شود و به چند گروه تقسیم شده است. معمولاً گزینه آخر که تحت عنوان `111111:Ext. Crystal` نام گزاری شده است برای کریستال های فرکانس بالا از ۴ تا ۱۶ مگاهرتز به خوبی جواب می دهد. البته در موارد حساس که مثلاً برای ارتباط سریال آسنکرون از کریستال 11.0592 استفاده می کنید به گزینه های دیگر هم توجه داشته باشد و در کل حتماً به توصیه های دیتا شیت میکروی مربوطه مراجعه کنید. در این صورت در آغاز برنامه باید از دستور زیر برای فرکانس 11.0592 استفاده کنید: `$crystal=11059200` (فرکانس برحسب هرتز و با ۶ رقم اعشار در نظر گرفته می شود)

فیوز بیت ها بخش حساس و مهم هر میکرو هستند و مطالعه کاربردهای آنها در پروژه های صنعتی و حرفه ای به شدت توصیه می شود. به دلیل همه جانبه بودن این سری مقالات، زیاد وارد جزئیات هر مبحث نمی شویم، فقط به

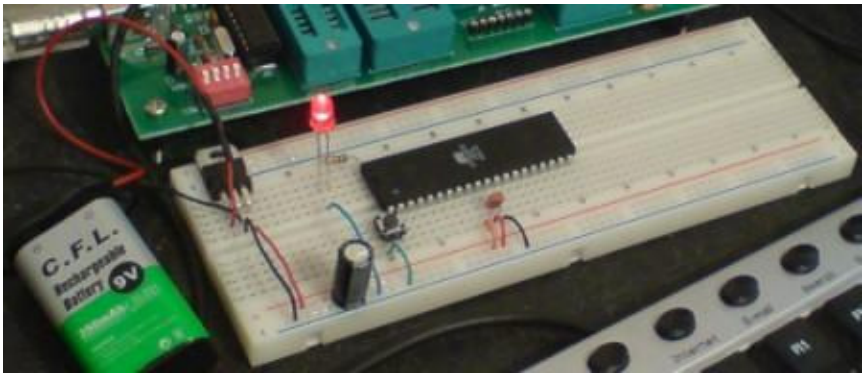
طور خلاصه باید ذکر شود که دو فیوز بیت بعدی که در شکل زیر نشان داده شده است را حتماً در حالت disable یا unprogrammed قرار دهید. این دو فیوز مربوط به ارتباط JTAG (که یک نوع پروتکل ارتباطی است) می باشند و به صورت پیش فرض فعال هستند و در صورتی که آنها را غیر فعال نکنید پورت JTAG میکرو (عموماً پورت C در Mega16 و Mega32) به درستی کار نخواهد کرد و در صورت استفاده از کی پد در این پورت مدام عدد ۷ برگردانده می شود! فراموش نکنید که پس از غیر فعال کردن این بخش حتماً روی کلید Write FSH که در سمت راست پنجره فعال می شود کلیک کنید تا این تغییرا ت در فیوز های میکرو نوشته شوند.



شکل ۱۱- فیوزبیت های JTAG

اصولا پس از هر تغییری در فیوز بیت ها کلید مربوط به آنها در سمت راست فعال می شود که با کلیک کردن آن تغییرات حاصله در میکرو ذخیره خواهند شد.

در صورتی که همه چیز به خوبی پیش رفته باشد با قرار دادن میکرو روی برد بورد و متصل کردن تغذیه می بینید که LED با فرکانس ۱ هرتز شروع به چشمکزدن می نماید. می توانید به عنوان تمرین مقدار دستور `waitms 500` را تغییر داده و یا از دستور `wait 1` استفاده کنید تا زمان تاخیر کمتر و بیشتری به دست آورید و سرعت چشمکزدن LED را کم و زیاد کنید و یا با تغییر میزان روشن بودن به میزان خاموش بودن به LED حالت فلاش زدن بدهید. همچنین می توانید از LED های زیادی روی 8 پایه پورت A استفاده کرده و با روشن و خاموش کردن آنها به صورت متناوب پروژه های تزئینی مختلفی طراحی کنید.

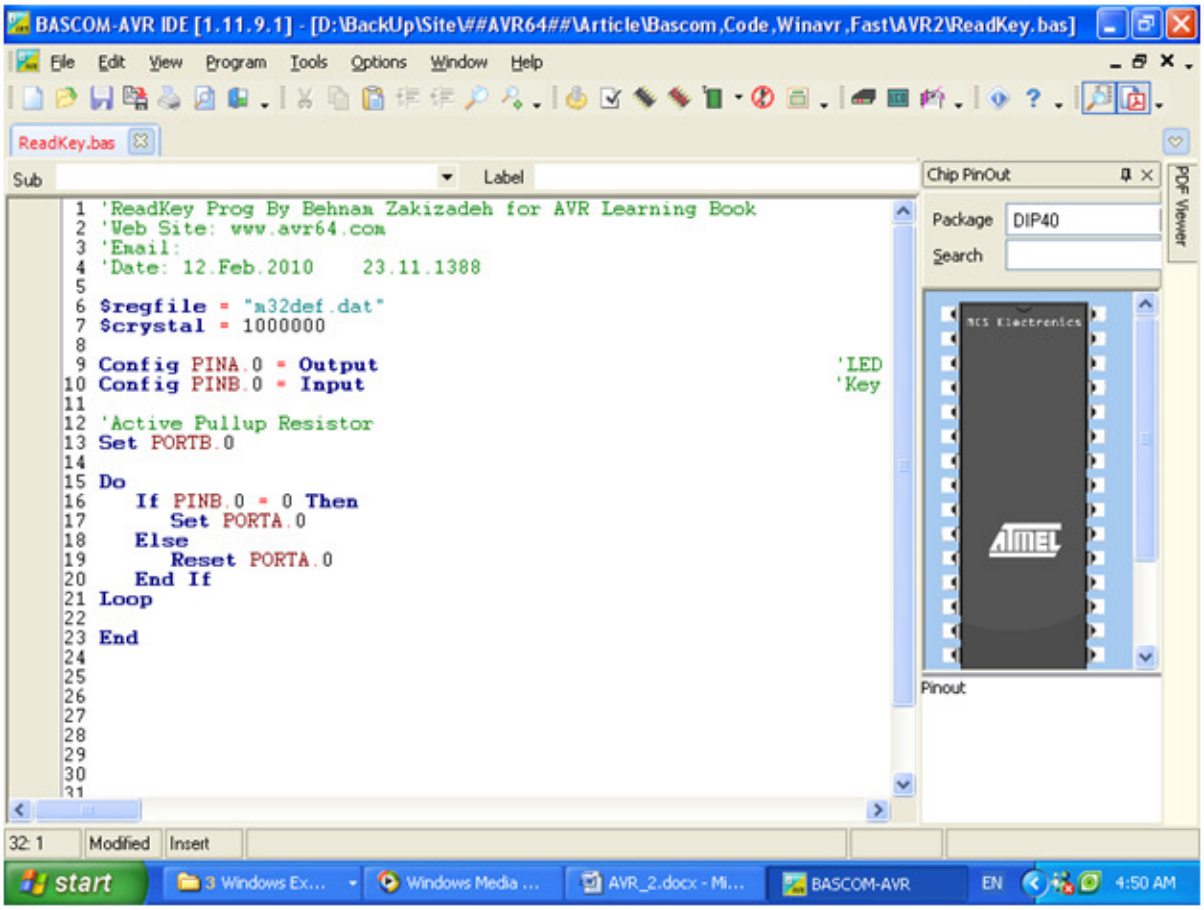


شکل ۱۲- آزمایش پروژه چشمکزن

برنامه خواندن از ورودی

تا اینجا نحوه نوشتن در پورت را یاد گرفتیم و موفق شدیم پایه متصل به LED را روشن و خاموش کنیم. در این قسمت تصمیم داریم نحوه خواندن یک پایه را که عموماً به یک کلید متصل می شود آموزش دهیم. از جلسه پیش به یاد داریم که برای خواندن کلید راه های متفاوتی وجود دارد. یکی از ساده ترین این راه ها این بود که کلید را بین یک پایه میکرو و زمین (قطب منفی) قرار داده و مقاومت Pull-up داخلی را به صورت نرم افزاری فعال کنیم. در این برنامه ترتیبی می دهیم که با زدن کلید LED روشن شود.

برنامه قبل (BlinkLED) را باز کرده و آن را به صورت زیر تغییر می دهیم و با عنوان (Readkey) در یک پوشه جدید ذخیره می کنیم (همیشه پیغام داده شده در مورد CFG فایل را Yes بزنید) . سعی کنید که همیشه نام تمام فایل ها و پوشه های کامپیوتر و موبایل خود را حداکثر ۸ کاراکتر و به صورت لاتین تایپ کنید تا تحت هیچ سیستمی به مشکل بر نخورید.



شکل ۱۴- برنامه پروژه خواندن کلید

در اینجا به توضیحات برنامه نوشته شده می پردازیم. در این برنامه تا خط 9 همه چیز مطابق برنامه قبل پیش رفته است؛ یعنی در خطوط ۱ تا ۴ توضیحات را تایپ کرده ایم، یک خط را خالی گذاشته ایم و در خطوط ۶ و ۷ نام میکرو و فرکانس کاری ان را مشخص نموده ایم. در خط ۹ هم پایه A0 را به عنوان خروجی تعریف کرده ایم. توجه داشته باشید که خطوط و فضاهای خالی تاثیری در عملکرد برنامه ندارند و فقط به زیبایی و خوانایی کد کمک می کنند.

در خط ۱۰ پایه B0 را که به کلید وصل شده است (نگاهی به شماتیک پروژه چشمکزن بیندازید) به عنوان ورودی تعریف می کنیم. شاید خط ۱۳ کمی گیج کننده باشد؟ ما پایه B0 را به عنوان ورودی تعریف کرده ایم ولی آن را یک کرده ایم. این کار چه معنایی دارد؟ با توجه به اینکه ما پایه را به عنوان ورودی تعریف کرده ایم و آن را به یک کلید متصل کرده ایم یک کردن این پایه سبب می شود تا به جای یک شدن پایه در خارج از میکرو، مقاومت Pull-up داخلی آن پایه فعال شود. اصولا دستور Set portx.y دو کار را در شرایط مختلف انجام می دهد: اگر پایه مورد نظر خروجی تعریف شده باشد با این دستور می توانیم منطق آن را عوض کرده و آن پایه را ۰ و ۱ کنیم، ولی در صورتی که پایه مورد نظر به عنوان ورودی تعریف شده باشد به کمک این دستور می توانیم مقاومت پول آپ داخلی آن را فعال و با دستور Reset portx.x آن را غیر فعال کنیم. همانطوریکه در جلسه اول ذکر شد وجود مقاومت پول آپ باعث می شود که در شرایط عادی پایه B0 روی منطق ۱ باشد و نویز نگیرد و به صورت ناگهانی ۰ نشود و در واقع ۱ بودن پایه متصل به کلید را در شرایط عادی تضمین می کند.

برنامه بررسی وضعیت کلید در داخل یک حلقه `Do..Loop` محصور شده تا برای همیشه وضعیت کلید را چک کند و متناسب با آن کار محول شده را انجام دهد. در خط ۱۶ یک دستور شرطی نوشته شده است. دستور `IF` وضعیت پین B0 را بررسی می کند، در صورتی که این پایه 0 شود یعنی کلید فشرده شده است پس دستور زیر `IF` اجرا شده و دیود LED روشن می شود. اما در غیر این صورت (`Else`) دستور زیر `Else` اجرا شده و LED خاموش می شود. در حالت عادی که کلید را فشار نداده ایم دستور `Else` در هر ثانیه حدود یک میلیون بار پورت متصل به LED را صفر می کند. دلیل آن هم اینست که فرکانس کاری CPUی میکرو را روی ۱ مگاهرتز بسته ایم.

تا اینجا یاد گرفتیم که چگونه بسکام را راه اندازی کرده و اولین پروژه خود را با آن بنویسیم. به عنوان تمرین می توانید از کلیدها و LED های بیشتری استفاده کنید و یا با ترکیب منطق های شرطی و دستورات `Wait` یک تایمر راه پله طراحی کنید. البته در این صورت باید پورت خروجی را به وسیله یک ترانزیستور

تقویت کرده و توسط آن یک رله را راه اندازی کنید تا بتوانید بار 220 ولت را کنترل نمایید. این موارد در پروژه های بعدی شرح داده می شوند.

این جلسه را همین جا تمام می کنیم و در جلسه بعد به معرفی سایر دستگاه های ورودی/خروجی از قبیل LCD و Keypad و نحوه کد نویسی آنها در محیط بسکام می پردازیم.

ادامه دارد ...

مولف: بهنام زکی زاده

www.avr64.com

۱ اسفند ۱۳۸۸

آخرین ویرایش ۱۳ مهر ۱۳۹۲ ✓