

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

آموزش پروگرم کردن AVR

و

شرح کامل فیزیت های آن

گردآورنده:

مهدی کمان گری

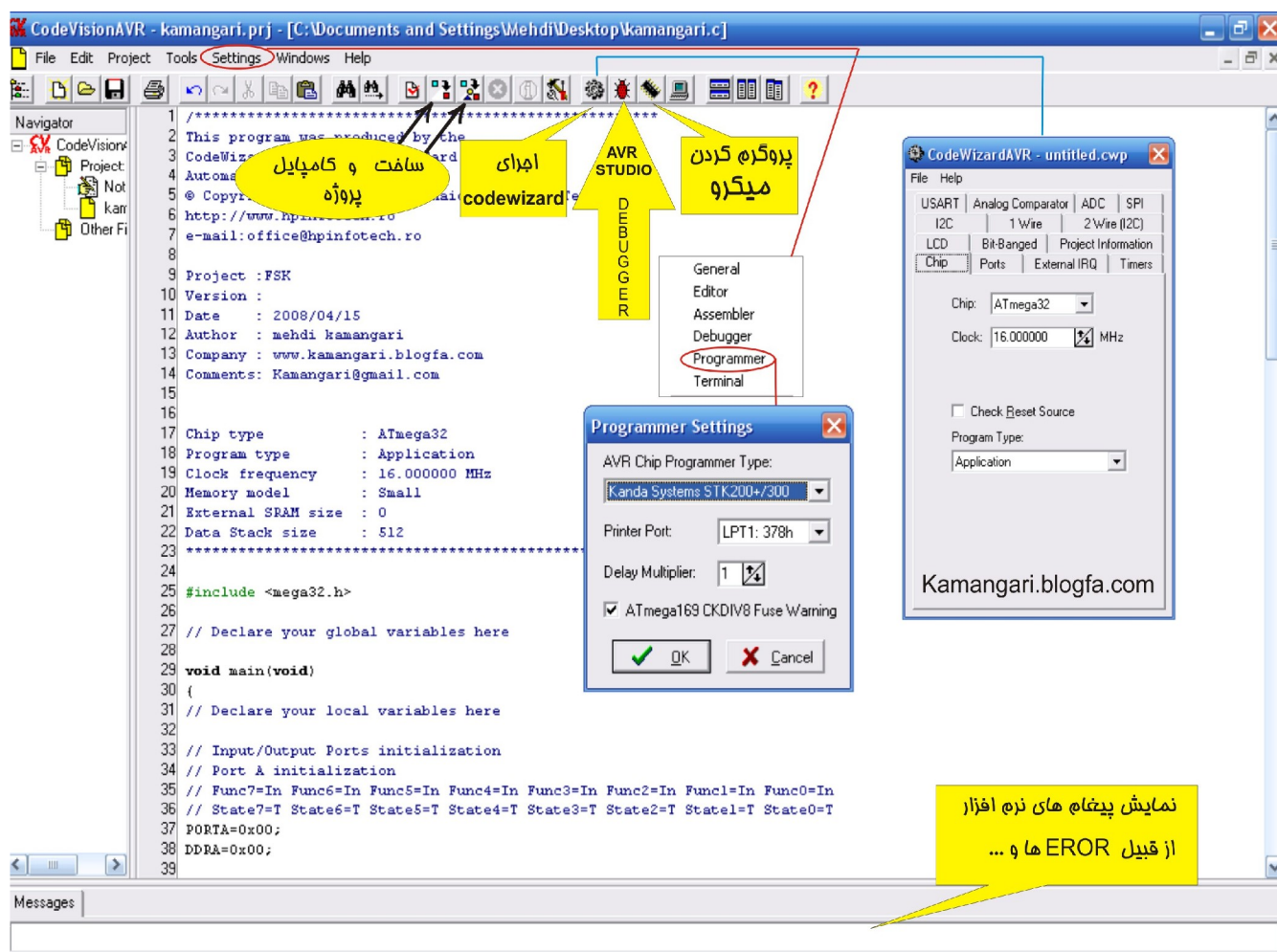
www.kamangari.blogfa.com

www.ir-micro.com



مقدمه :

در این مقاله قصد داریم نحوه ی ایجاد پروژه ، تولید فایل هگز و پروگرام کردن میکروکنترلر AVR را با نرم افزار Codevision آموزش دهیم . سپس بطور کامل ، فیزیوتی های AVR را شرح خواهیم داد . هنگامی که وارد محیط نرم افزار Codevision می شویم ، با صفحه ای به شکل زیر روبرو هستیم . البته در شکل زیر بعضی از قسمتهای مهم و پرکاربرد و حتی پنجره های زیر منوی آنها نیز نمایش داده شده است که در ادامه تک تک آنها را شرح می دهیم .

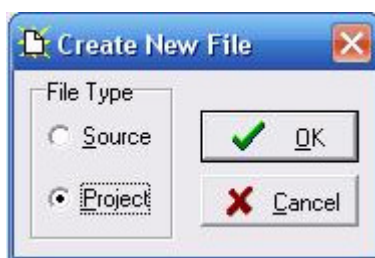


پس از آشنایی اجمالی با محیط نرم افزار ، نوبت به ایجاد یک پروژه جدید می رسد که مراحل آن در صفحات بعدی توضیح داده شده است .

در ابتدا وارد محیط نرم افزار CodeVision AVR شده و از منوی File ، گزینه ی New را انتخاب می کنیم .
(یا از ابزار استاندارد Windows که به شکل زیر است ، گزینه ی New را انتخاب می کنیم .)



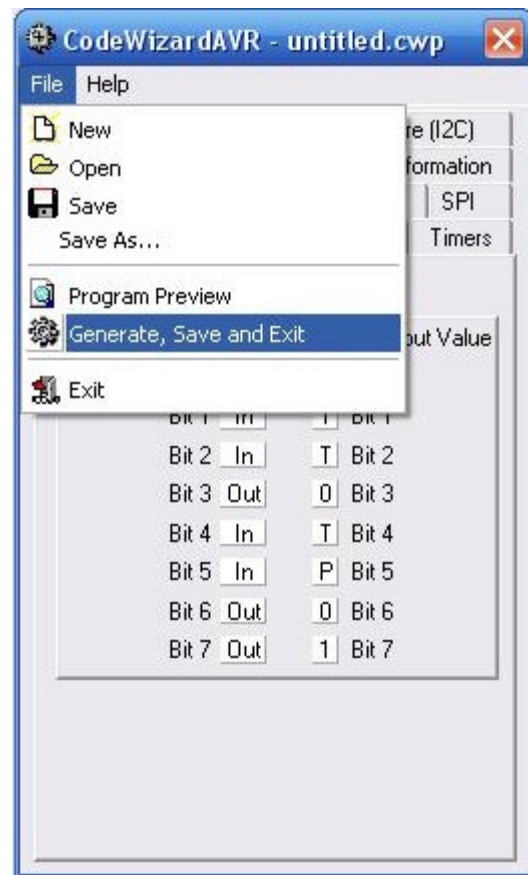
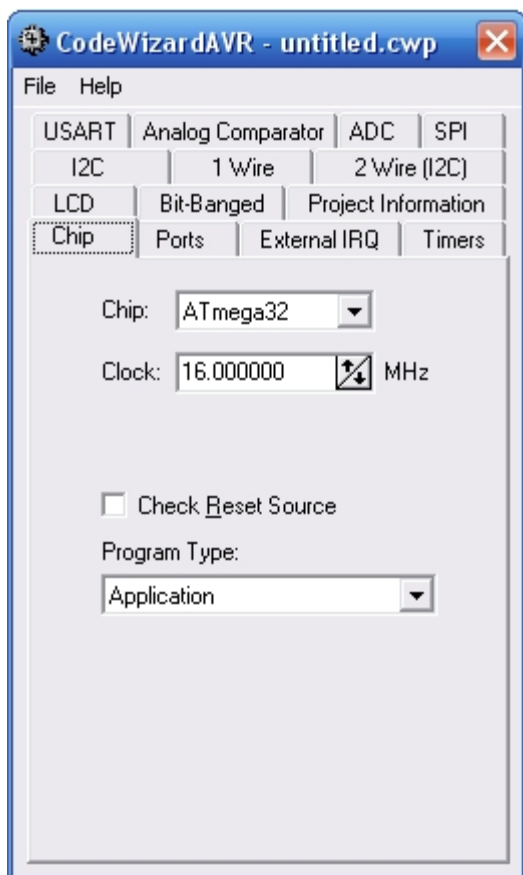
پس از انتخاب گزینه ی New ، پنجره ای به شکل زیر باز می شود که ما در اینجا گزینه ی Project را انتخاب می کنیم .



در این جا پنجره ی دیگری باز می شود و سوالی را مبنی بر استفاده و یا عدم استفاده از قابلیت ویژه ی نرم افزار که Codewizard نام دارد ، مطرح می کند که معمولاً اکثر کاربران به آن پاسخ مثبت می دهند . زیرا برنامه نویسی را ساده تر کرده و حجم کدنویسی را کاهش می دهد . پس ، در پنجره ی زیر گزینه ی Yes را کلیک می کنیم .




در پنجره ی جدید ایجاد شده ، که همان پنجره ی Codewizard است ، تنظیمات لازم را انجام می دهیم . در این پنجره می توانیم ورودی و یا خروجی بودن پورتهای ، نوع چیپ و فرکانس کاری آن را تعیین کرده و در صورت نیاز ، تایمرها یا کانترها ، وقفه ها ، LCD ، SPI ، ADC ، پورت سریال و ... را فعال کنیم .
در اینجا لازم میدانم که در مورد پورتهای توضیح مختصری را ارائه کنم . می دانیم که پورتهای میکرو در دو حالت ورودی (IN) یا خروجی (OUT) به کار می روند . ما هم برحسب نیاز هرکدام از پورتهای را در حالت IN یا OUT قرار می دهیم . اما در جلوی هر کدام از آنها ، گزینه ی دیگری نیز وجود دارد . مثلاً اگر پورت را در حالت IN قرار دهیم ، گزینه های T یعنی Tri-state (سه حالت) و P یعنی Pull up (مقاومت داخلی بالا کش) وجود دارند که ما می توانیم یکی از آنها را برحسب نیاز انتخاب کنیم . و یا اگر پورت را در حالت OUT قرار دهیم ، گزینه های 1 و 0 را می توانیم به عنوان مقدار اولیه ی پورتهای برگزینیم . این قابلیت (مثلاً استفاده از مقاومت داخلی Pull up) می تواند سخت افزار پروژه را کاهش دهد .






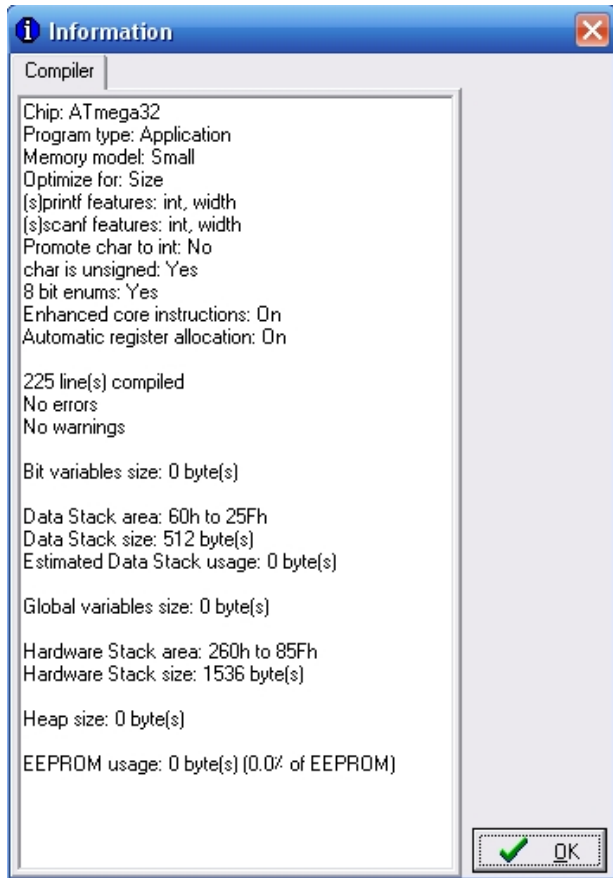
پس از اتمام تنظیمات Codewizard، گزینه ی File خود پنجره ی Codewizard را کلیک کرده و گزینه ی Generate, Save and Exit را انتخاب می کنیم. در این جا، نام پروژه را در آدرس دلخواه و در سه مرحله وارد می کنیم تا پروژه ی ما Save گردد.


منظور از سه مرحله، این است که پنجره ی اول نام C Compiler files (*.c) را از کاربر می گیرد و پس از ذخیره کردن آن، بلافاصله پنجره ی دیگری باز شده و نام Project files (*.prj) و در مرحله ی آخر، نام CodewizardAVR project files (*.cwp) را از کاربر سوال می کند. لازم به ذکر است که برای جلوگیری از هرگونه مشکل احتمالی بهتر است نام هر سه فایل یکسان باشد.

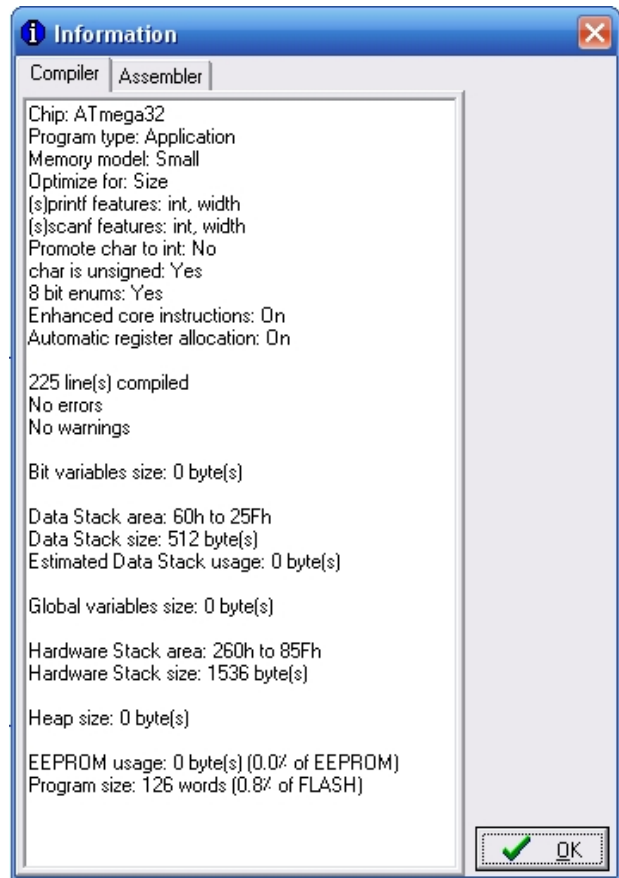
با انجام مراحل فوق، Codewizard AVR کدهای برنامه را تولید کرده و در برنامه ی اصلی قرار می دهد و آن را باز می نماید. علاوه بر کدها، توضیحاتی (Comment) نیز به متن برنامه اضافه می کند تا فهم برنامه برای کاربر ساده تر گردد. حال می بایست برنامه ی خود را در قسمتی که با عنوان //Place your code here مشخص شده است، وارد کنیم.


پس از وارد کردن و تایپ برنامه، نوبت به کامپایل برنامه و تولید کد هگز آن می رسد. برای این منظور ابتدا باید از صحت و درستی برنامه مطمئن شویم. چون اگر برنامه مشکل ساختاری داشته باشد، نرم افزار برای آگاهی ما، EROR می دهد. این EROR ها و پیغام های مشابه دیگر، در پایین صفحه قابل رویت هستند که در شکل کلی صفحه ی نرم افزار مشخص شده است. لذا گزینه ی  را کلیک می کنیم تا برنامه کامپایل شود. در پنجره ای که باز می شود، اطلاعاتی در مورد چیپ، مقدار فضای حافظه نظیر Stack، و از همه مهمتر تعداد EROR ها و Warning ها داده می شود. در صورت داشتن خطا در برنامه، با کمک گرفتن از پنجره ی پیغام ها در پایین صفحه - که منشأ خطا را نشان می دهد- برنامه را اصلاح می کنیم.

پس از اصلاح برنامه و یا در صورت نداشتن هرگونه خطا ، گزینه ی  را در بالای صفحه کلیک می کنیم تا فایل کد هگز ایجاد شود . به این ترتیب ، مهمترین مرحله از یک پروژه که همان تولید فایل هگز است ، بطور کامل انجام شد . دو شکل زیر ، پنجره هایی را که در زمان کلیک کردن دو گزینه ی  و  ، نمایش داده می شوند را نشان می دهد .

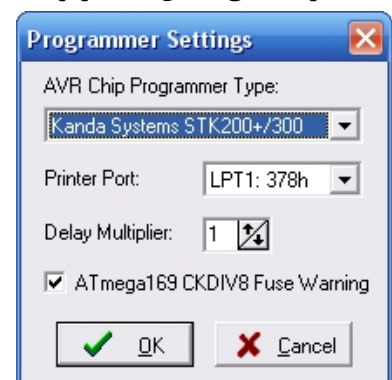
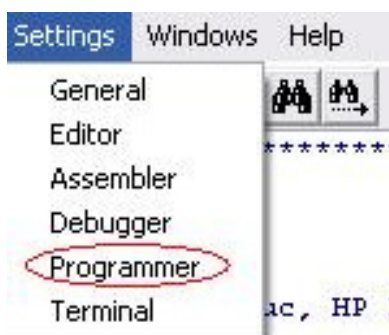



 پنجره ی مربوط به

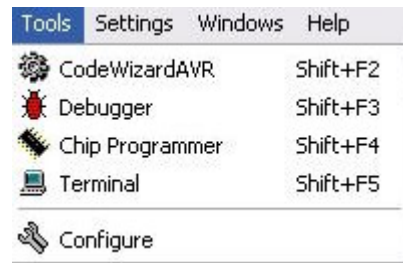


 پنجره ی مربوط به

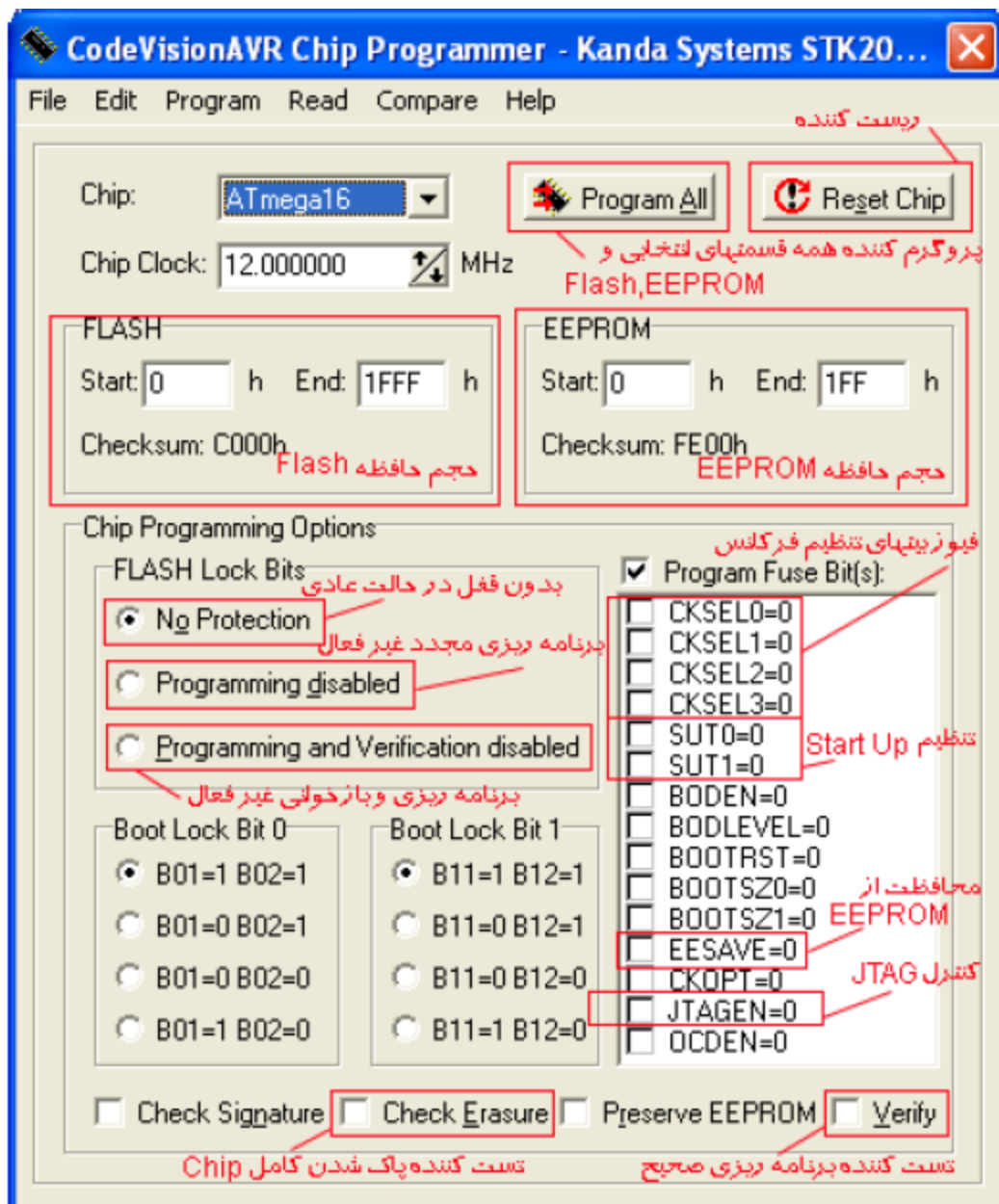
هم اکنون باید فایل کد هگز را در آی سی بریزیم . به عبارتی باید میکرو را پروگرام کنیم . برای این منظور ، ضمن اتصال پروگرامر به پورت پرینتر (پارالل یا موازی) ، تغذیه ی ۵ ولت را نیز به آی سی میکرو اعمال می کنیم . سپس از منوی Setting گزینه ی Programmer را انتخاب می کنیم . در پنجره ای که ایجاد می شود ، نوع پروگرامر (STK200+/300 یا STK500/AVRISP یا ...) و شماره ی پورت پاراللی را که به پروگرامر متصل است ، را مطابق شکل های زیر انتخاب می کنیم .



پس از OK کردن پنجره ی مربوط به Programmer setting ، گزینه ی  را انتخاب می کنیم . (یا مطابق شکل زیر، به منوی Tools رفته و گزینه ی Chip programmer را انتخاب می کنیم .)



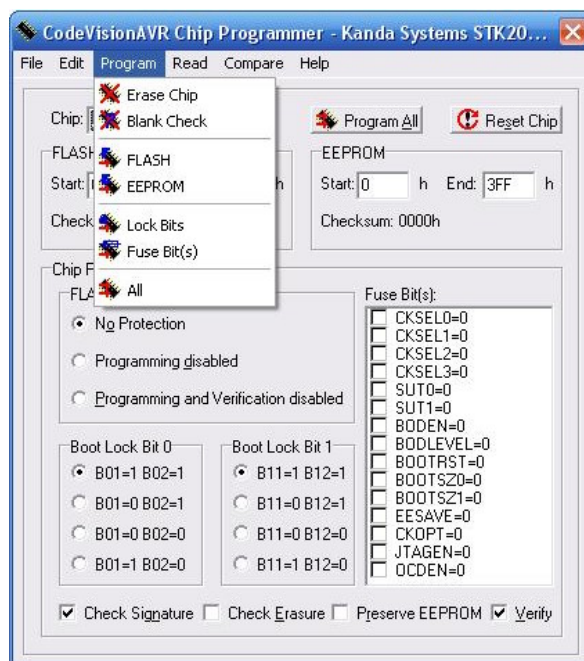
در این هنگام پنجره ای باز می شود که در شکل زیر، اجزای آن مشخص شده است . لازم به ذکر است که نوع چیپ یا فرکانس کریستال باید با مقادیر انتخابی در برنامه (Codewizard) یکسان باشد .



WWW.KAMANGARI.BLOGFA.COM

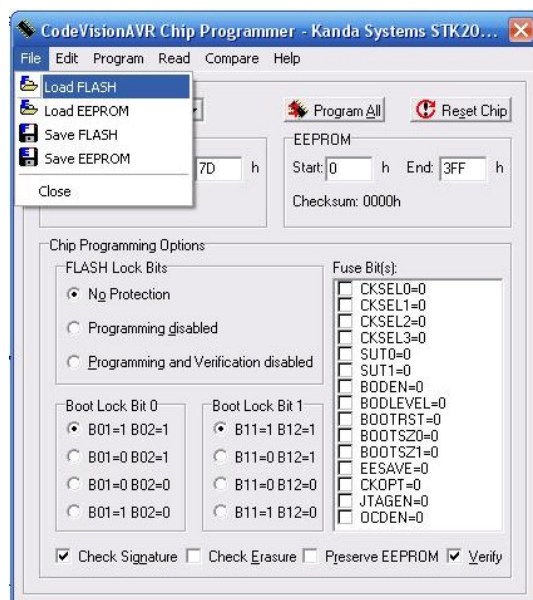
این پنجره دارای منوهای File ، Edit ، Program و ... می باشد که در شکل های جداگانه ای ، گزینه های آن مشخص شده اند .

حال از منوی Program همین پنجره ، گزینه ی Erase chip را انتخاب می کنیم . عملکرد این گزینه این است که برنامه ی قبلی را از میکرو پاک کرده و آن را برای پروگرام کردن مجدد آماده می کند .

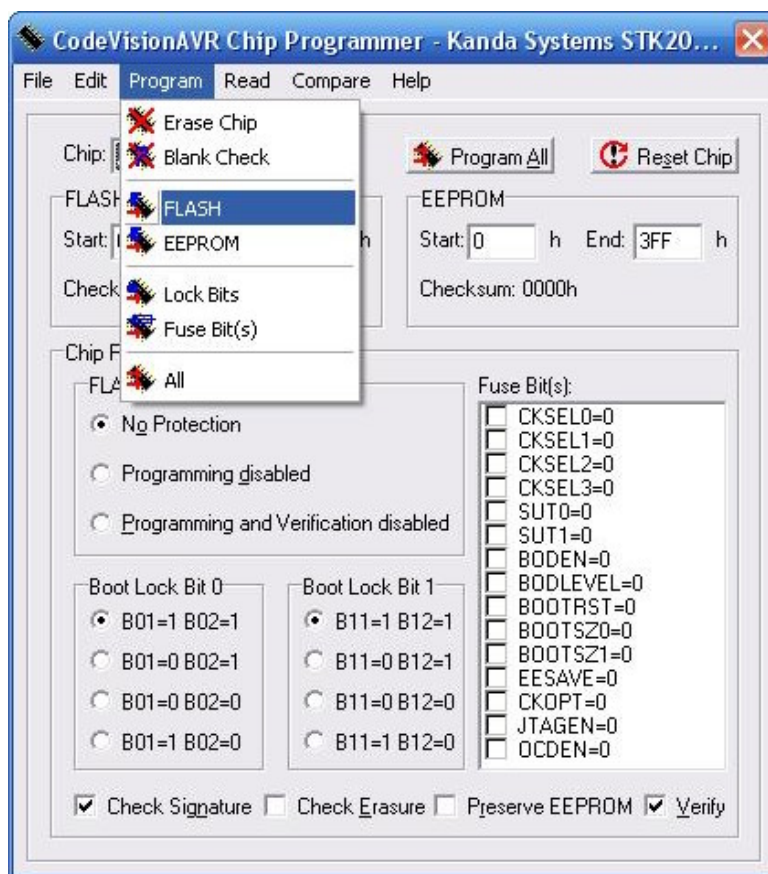


WWW.KAMANGARI.BLOGFA.COM

سپس از منوی File گزینه ی Load flash را انتخاب نموده و آدرس فایل هگز را به آن می دهیم . (لازم به ذکر است که اگر برنامه را همین الان در محیط برنامه نویسی Codevision بنویسیم و بخواهیم آن را در میکرو پروگرام کنیم ، نرم افزار بطور اتوماتیک آدرس فایل هگز برنامه را انتخاب می کند و نیازی به انجام این مرحله نیست؛ ولی اگر بخواهیم فایل هگز دیگری را در میکرو پروگرام کنیم ، باید از همین قسمت ، فایل هگز را آدرس دهی نمائیم .)



در مرحله ی آخر ، از منوی Program گزینه ی Flash را انتخاب می کنیم تا میکرو پروگرم شود .



WWW.KAMANGARI.BLOGFA.COM

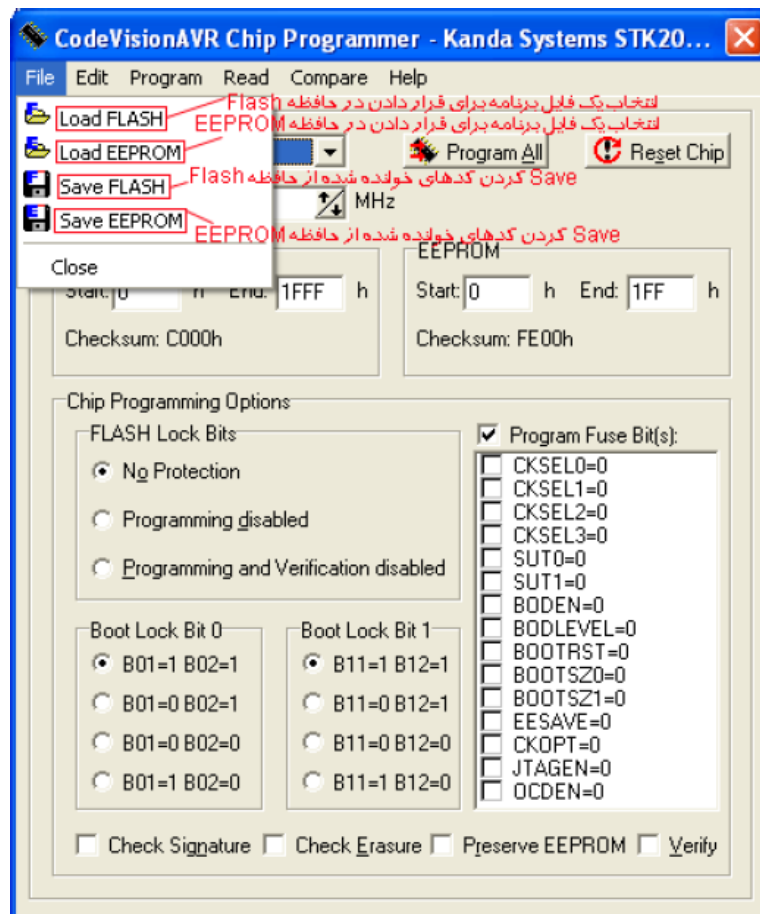
تذکره :

اگر در حین مراحل پروگرم کردن ، متوجه EROR ی شدید ، روی گزینه ی NO کلیک کرده و مراحل گفته شده را مجدداً تکرار نمایید تا میکرو بدون هیچ EROR ی پروگرم شود .
در انتها پروگرامر را از مدار جدا می کنیم تا به عملکرد میکرو در مدار خللی وارد نشود . هم اکنون میکرو برای کار در مدار آماده می باشد .

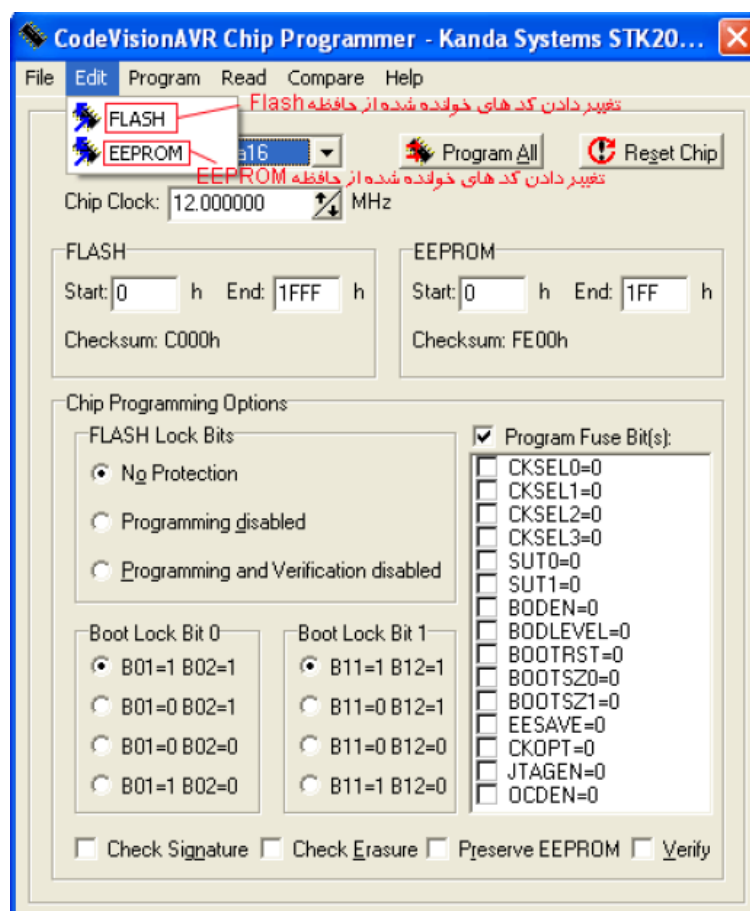
توجه :

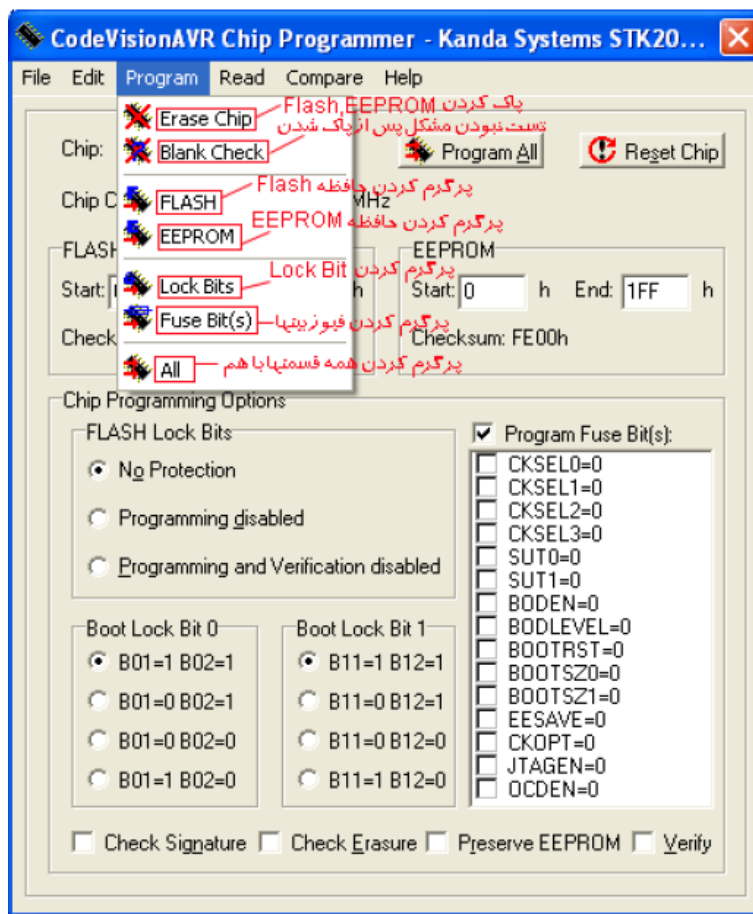
در صفحات بعدی ، زیرمنوهای پنجره ی Chip Programmer در ۵ شکل کاملاً مجزا بصورت تصویری (متن در تصویر) آورده شده است تا کاربران با گزینه های آن آشنا شده و بتوانند در هنگام پروگرم کردن ، بر حسب نیاز خود از آنها استفاده کنند .

WWW.KAMANGARI.BLOGFA.COM

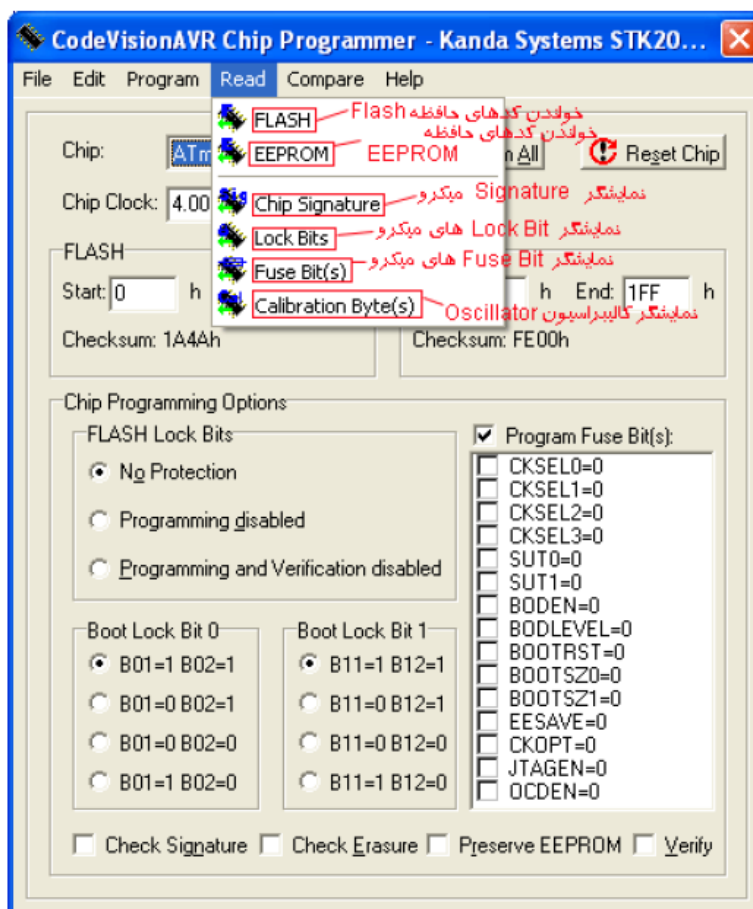


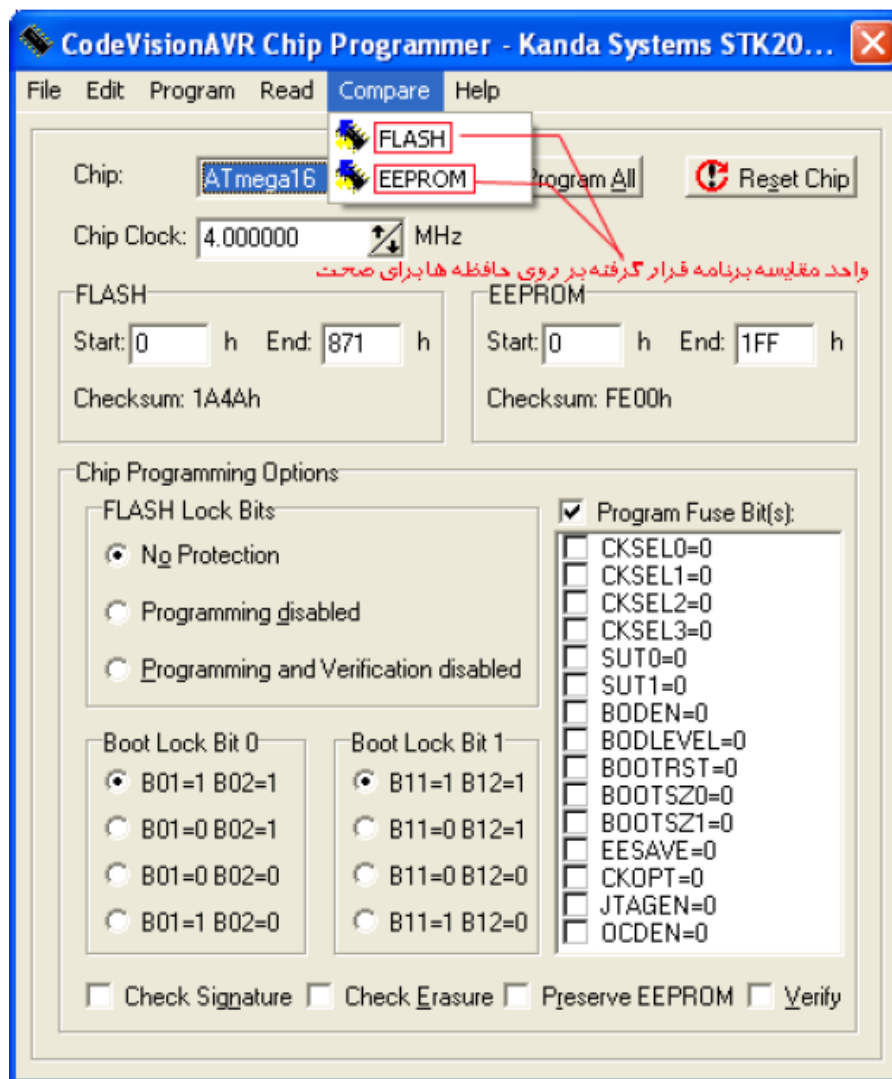
WWW.KAMANGARI.BLOGFA.COM





WWW.KAMANGARI.BLOGFA.COM





WWW.KAMANGARI.BLOGFA.COM

فیوز بیت ها :

فیوز بیت ها بخشی از حافظه ی Flash می باشند که با برنامه ریزی کردن آن ها یک سری امکانات در اختیار کاربر قرار می گیرد و علاوه بر این که بهره گیری از آنها بسیار مفید و قابل توجه است، می تواند بسیار دزدساز هم باشد. به این دلیل که برخی از کاربران، بویژه آنهایی که مبتدی و تازه کار هستند، با تنظیمات فیوزبیت ها مشکل دارند و از عملکرد آنها اطلاع درستی ندارند.

در AVR حداکثر سه بایت برای فیوز بیت ها در نظر گرفته شده است، که عبارتند از :

۱. بایت بالای فیوزبیت
۲. بایت پایین فیوزبیت
۳. فیوزبیت های توسعه یافته

اسامی این فیوزبیت ها در جدول زیر گردآوری شده است :

فیوزبیت	فیوزبیت	فیوزبیت	فیوزبیت
DWEN	STU	EESAVE	CKSEL0..3
M103C	RSTDISBL	JTAGEN	BODEN
M161C	WDTON	OCDEN	BODLEVEL
S8535C	CKDIV8	SPIEN	BOOTRST
S8515C	CKOUT	CKOPT	BOOTSZ0
	SELFPRGEN	SUT0..1	BOOTSZ1

نکته :

فیوزبیت ها با Erase کردن حافظه ی Flash (یعنی Erase chip) از بین نمی روند ؛ در حالی که لاک بیت ها (Lock bit) با Erase کردن ، پاک می شوند .

* همانطور که گفته شد تنظیمات اصلی AVR توسط فیوز بیت ها انجام می شود . تعداد و نام فیوزبیت ها در سری های مختلف AVR تقریباً با هم برابر است (البته با کمی تغییرات جزئی) . در این بخش قصد داریم تمام فیوزبیت های موجود در AVR را توضیح دهیم .

جدول فوق تقریباً تمامی فیوزبیت های موجود در AVR ها را نشان می دهد . لازم به ذکر است که با توجه به میکروکنترلر مورد نظر ، ممکن است فقط تعدادی از این فیوزبیت ها در آن بکار رفته باشد .

نکته مهم :

در فیوزبیت ها ، " 0 " به معنای برنامه ریزی شدن و " 1 " به معنای برنامه ریزی نشدن فیوزبیت می باشد . مثلاً اگر بخواهیم یک فیوزبیت را در محیط CodeVision فعال کنیم ، باید مربع کوچک کنار آن فیوزبیت را تیک بزنیم . لازم به یادآوری است که تیک زدن فیوزبیت به معنای صفر کردن (فعال کردن) آن می باشد .

فیوزبیت های CKSEL3..0 :

این فیوزبیت ها برای انتخاب منبع تولید پالس ساعت استفاده می شوند که کاربر می تواند به کمک همین فیوزبیت ها ، منبع تولید پالس ساعت مورد نیاز خود را انتخاب کند . جدول زیر ، نحوه ی این انتخاب را نشان می دهد :

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111-1010
External Low-frequency Crystal	1001
External RC Oscillator	1000-0101
Calibrated Internal RC Oscillator	0100-0001
External Clock	0000

نکته ی مهم و کاربردی :

اگر به فیوزبیت های یک میکروی نو و سالم ، کاری نداشته باشیم ، با کلاک داخلی خود که به مقدار 1 MHZ می باشد ، در مدار کار می کند . ولی اگر بخواهیم میکرو با کریستال خارجی یا ... کار کند ، باید مطابق جدول فوق ، فیوزبیت های CKSEL3..2 را تنظیم کنیم . لازم به ذکر است که چون اکثر کاربران مبتدی با پروگرامر STK200+/300 کار می کنند ، با تنظیم این فیوزبیتها مشکل دارند . فرض کنید فیوزبیت های یک میکرو را در حالت کریستال خارجی تنظیم کرده ایم . حال اگر بخواهیم فیوزبیت های آن را تغییر دهیم یا برنامه ی دیگری را در میکرو پروگرام کنیم ، باید در هنگام پروگرام کردن ، یک کریستال خارجی نیز علاوه بر تغذیه به میکرو اعمال کنیم تا نرم افزار ، میکرو را بشناسد و سپس برنامه یا فیوزبیتها را تغییر دهیم . وگرنه نه تنها میکرو پروگرام نمی شود ، بلکه حتی نرم افزار نیز نمی تواند میکرو را بشناسد ! این مهمترین و شایعترین مشکل کاربران AVR با فیوزبیت ها است .

فیوزبیت BODEN :

این فیوزبیت برای فعال کردن واحد BROWN-OUT استفاده می شود که مربوط به بازنشانی (Reset) می باشد . به عبارت دیگر در برخی از AVR ها یک مدار آشکارساز BROWN-OUT داخلی وجود دارد . این آشکارساز در صورت فعال بودن در طول عملکرد AVR ، سطح ولتاژ منبع تغذیه (VCC) را با یک ولتاژ مرجع داخلی (VBOT) مقایسه می کند و در صورتی که سطح VCC به بیش از سطح ولتاژ مرجع (VBOT+) افزایش یابد ، تایمر تأخیر وارد عمل شده و تأخیری به اندازه ی Time Out ایجاد می کند و پس از آن ، میکروکنترلر از آدرس بردار RESET شروع به اجرای برنامه می کند . برای فعال کردن آشکارساز BROWN-OUT از فیوزبیت BODEN استفاده می گردد .

فیوز بیت BODLEVEL :

این فیوزبیت اگر برنامه ریزی نشده باشد ، در صورتی که ولتاژ تغذیه پایه VCC از مقدار ۲٫۷ ولت کمتر شود ، میکروکنترلر در حالت RESET قرار می گیرد . و در صورت برنامه ریزی شدن ، میکروکنترلر با ولتاژ کمتر از ۴ ولت در حالت RESET قرار می گیرد . لازم به ذکر است که در برخی از میکروکنترلرها مانند ATMEGA2560 به جای این فیوزبیت ، فیوزبیت های BODLEVEL0..3 بکار رفته است که سه سطح ولتاژ ، ۱٫۸ ، ۲٫۷ ، و ۴٫۳ ولت را برای مقایسه در اختیار کاربر قرار می دهد .

فیوزبیت BOOTRST :

این فیوزبیت برای انتخاب بردار RESET استفاده می شود و در حالت پیش فرض برنامه ریزی نشده است .

فیوزبیت BOOTSZ0 & BOOTSZ1 :

این دو فیوزبیت برای انتخاب مقدار فضای حافظه BOOT مورد استفاده قرار می گیرند . می دانیم که حافظه ی FLASH به دو بخش برنامه ی کاربردی و BOOT تقسیم می شود که البته بعضی از AVR ها فقط دارای حافظه ی کاربردی می باشند . در میکروکنترلر هایی که فقط دارای حافظه ی کاربردی می

باشند ، برنامه از طریق پروگرامر بر روی حافظه قرار می گیرد و در طول اجرای برنامه به هیچ عنوان نمی توان در حافظه برنامه اطلاعاتی نوشت . حال به کمک تکه برنامه ای به نام BOOT LOADER که در بخش BOOT از برنامه قرار می گیرد ، می توان برنامه ی جدید یا اطلاعاتی را از طریق رابط های SPI, USART, 2WIRE و ... دریافت نمود و در بخش حافظه برنامه کاربردی قرار داد .

فیوزبیت EESAVE :

در صورت برنامه ریزی شدن این فیوزبیت به هنگام ERASE کردن میکروکنترلر ، حافظه ی EEPROM داخلی پاک نمی شود .

فیوزبیت JTAG :

این فیوزبیت برای فعال کردن رابط JTAG می باشد و به صورت پیش فرض برنامه ریزی شده است . لازم به ذکر است که در صورت برنامه ریزی شدن این فیوزبیت ، پایه های پورت JTAG دیگر به عنوان I/O معمولی به کار گرفته نمی شوند . رابط JTAG که مخفف Join Test Access Group می باشد ، برای تست ، برنامه ریزی و عیب یابی آی سی های دیجیتال (مطابق استاندارد IEEE 1149.1) به کار می رود . از قابلیت های این رابط می توان به موارد زیر اشاره کرد :

۱. برنامه ریزی حافظه EEPROM, Flash ، فیوزبیت ها ، بیت های قفل با سرعت بالا
۲. دارای قابلیت اشکال زدایی کلیه قسمت های داخلی شامل تمام واحدهای جانبی داخلی ، RAM داخلی و خارجی ، رجیستر های داخلی ، شمارنده ی برنامه ، حافظه های Flash و EEPROM
۳. دستورالعمل توقف (BREAK) AVR روی جریان اجرای برنامه و اجرای مرحله به مرحله برنامه

نکته ی مهم و کاربردی :

اگر از پورت C بخواهیم به عنوان I/O استفاده کنیم ، باید فیوزبیت JTAG را که بطور پیش فرض فعال است ، غیر فعال کنیم . چون ۴ پین TMS ، TCK ، TDI و TDO که مثلاً در ATMEGA16 و ATMEGA32 به ترتیب شماره های ۲۴ الی ۲۷ میکرو هستند ، مربوط به JTAG بوده و برای استفاده های دیگر (مثلاً اتصال به LCD) باید بصورت I/O برنامه ریزی شوند .

فیوزبیت OCDEN :

این فیوزبیت در صورت برنامه ریزی شدن (OCDEN=0) به همراه فیوز JTAGEN برای سیستم عیب یابی داخل مداری استفاده می شود .

فیوزبیت SPIEN :

این فیوزبیت برای فعال کردن قابلیت برنامه ریزی از طریق رابط SPI قابل استفاده و در حالت پیش فرض برنامه ریزی شده است .

فیوزبیت CKOPT :

این فیوزبیت برای انتخاب پالس ساعت نوسان ساز استفاده می شود . هنگامی که فیوزبیت CKOPT برنامه ریزی می شود ، فرکانس خروجی نوسان ساز دارای محدوده ی وسیع و دامنه ثابت ولتاژ در خروجی خواهد بود . این حالت برای زمانی که از میکرو کنترلر در محیط پر نویز استفاده می شود ، مناسب است . همچنین در این حالت می توان از پایه ی خروجی XTAL2 برای فعال کردن بافر پالس ساعت میکروکنترلرهای دیگر نیز استفاده کرد . هنگامی که فیوزبیت CKOPT برنامه ریزی نشده باشد ، دامنه فرکانس نوسان ساز محدود تر خواهد شد و دیگر نمی توان از پایه ی XTAL2 به منظور فعال کردن بافر پالس ساعت میکروکنترلرهای دیگر استفاده نمود . در صورت استفاده از رزوناتور سرامیکی ، اگر فیوزبیت CKOPT برنامه ریزی شود ، فرکانس نوسان ساز حداکثر 16MHZ و در صورت برنامه ریزی نشدن حداکثر 8MHZ خواهد بود . خازن های C1 و C2 که به عنوان خازن های بالانس یا نویز گیر شناخته می شوند ، دارای مقدار یکسانی بوده و مقدار آن به فرکانس کریستال مورد استفاده بستگی دارد و مطابق جدول زیر انتخاب می شود .

CKOPT	CKSEL3..1	Frequency Range (MHZ)	Recommended Range for c1 & c2
1	101	0.4 - 0.9	—
1	110	0.9 – 3.0	12 – 22
1	111	0.3 – 8.0	12 – 22
0	101,110,111	1.0≤	12 – 22

فیوزبیت SUT0..1 :

این دو فیوزبیت برای تعیین زمان Start Up بکار می روند . در برخی از میکروکنترلرها نام این فیوزبیت STU می باشد .

فیوزبیت RSTDISBL :

با برنامه ریزی این فیوزبیت پایه Reset خارجی میکرو غیر فعال می شود و دیگر میکرو بازنشانی خارجی نخواهد شد و می توان از آن پایه به عنوان ورودی معمولی استفاده کرد . لازم به ذکر است که در صورت برنامه ریزی این فیوزبیت دیگر نمی توان میکرو را توسط پروگرامر ISP که مخفف In System Programming است ، برنامه ریزی کرد .

فیوزبیت WDTON :

با برنامه ریزی کردن این فیوزبیت ، تایمر نگهبان همیشه روشن می ماند .

فیوزبیت CKDIV8 :

این فیوزبیت که در بعضی از میکروکنترلرها مانند ATMEGA 162 وجود دارد ، در صورت برنامه ریزی شدن فرکانس پالس ساعت سیستم را بر هشت تقسیم می کند .

فیوزبیت CKOUT :

این فیوزبیت که در بعضی از میکروکنترلرها مانند ATMEGA 162 وجود دارد ، در صورت برنامه ریزی شدن پالس ساعت سیستم را بر روی پایه PORTB.0 فعال خواهد کرد و می توان از آن به عنوان پالس ساعت برای دیگر قسمتهای مدار استفاده کرد .

فیوزبیت SELFPRGEN :

این فیوزبیت که در میکروکنترلرهای ATMEGA 48/88/168 وجود دارد ، در صورت برنامه ریزی شدن ، میکرو می تواند به بخش BOOT از حافظه برنامه رفته و خودش را برنامه ریزی کند .

فیوزبیت DWEN :

با برنامه ریزی این فیوزبیت سیستم عیب یابی داخل مداری فعال می شود .

فیوزبیت M103C :

این فیوزبیت در میکروکنترلرهایی که مشابه با میکروکنترلر ATMEGA 103 هستند ، وجود دارد و در صورت برنامه ریزی شدن آن ، میکرو مشابه ATMEGA 103 فعالیت خواهد کرد .

فیوزبیت M161C ، S8535C و S8515C :

این فیوزبیت ها مشابه فیوزبیت M103C می باشند با این تفاوت که میکروکنترلر به ترتیب مشابه ، ATMEGA 16 ، AT90S8535 و AT90S8515 فعالیت خواهد کرد .

کاری از :

مهدی کمان گری ؛ kamangari@gmail.com

WWW.KAMANGARI.BLOGFA.COM

WWW.IR-MICRO.COM