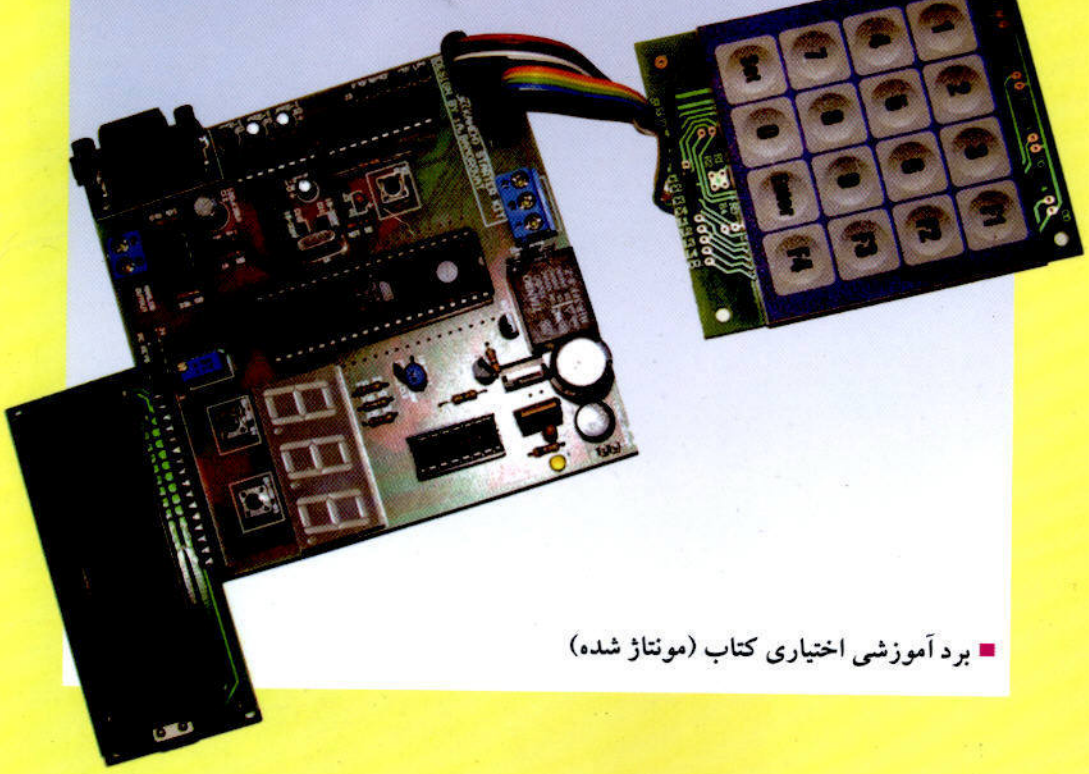
میکروکنترلرهای AVR

مهندس علی کاهه

چاپ سیزدهم (ویراست ۲)



■ برد آموزشی اختیاری کتاب (مونتاژ شده)

مقدمه

مطالعه و کار با یک میکروکنترلر غالباً برای دانشجویان لازم و ضروری است و چه بهتر که این یادگیری به‌روز باشد و دانشجو زمان خود را برای مطالعه درباره میکروکنترلری جدید صرف کند. یکی از جدیدترین میکروکنترلرهای قوی عرضه شده به بازار الکترونیک متعلق به شرکت ATMEL به نام میکروکنترلرهای AVR است. این میکروکنترلرهای 8 بیتی به علت وجود کامپایلرهای قوی به زبان HLL (HIGH LEVEL LANGUAGES) مورد استقبال و استفاده دانش‌پژوهان قرار گرفته است. در راستای آموزش و یادگیری، کتب و نشریات آموزشی نقش موثری را ایفا می‌کنند. کتابی که در حال خواندن آن هستید در مورد انواع میکروکنترلرهای 8 بیتیAVR است که چاپ اول آن در بهار سال ۸۳ بوده و سعی شده نگارش و توضیحات آن به گونه‌ای باشد که دانشجویان عزیز به راحتی بتوانند حتی با کمی آگاهی داشتن از مبانی دیجیتال و زبان برنامه‌نویسی آن را درک و به راحتی از آن ، ستفاده نمایند. حال پس از ۶ چاپ، با توجه به نظرات ارسالی خوانندگان و همچنین تغییرات و . شرفتهای رخ داده میکروکنترلرهای AVR طی این مدت، ضرورت ویراست جدید قطعی شد که به انجام آن همت گماردم.

علل ویراست دوم

از مهمترین دلایل ویراست کتاب:

۱- بهینه‌سازی و به روز کردن اطلاعات کتاب با توجه به نظرات خوانندگان و پیشرفتهای رخ داده در میکروکنترلرهای AVR

۲- ارائه **برد مدار چاپی** به همراه کتاب در راستای افزایش علاقمندی خواننده و استفاده عملی از مطالب این ویراست (برای اطلاع بیشتر فصل ۸ و پیوست ب را ببینید)

۳- رفع خطاهای نوشتاری و اشکالات بوجود آمده در شکلها

فصل‌های کتاب

در فصل یک میکروهای نوع TINYAVR و فیوز بیت‌های هر یک، در فصل دو میکروهای نوع AT90S و فیوز بیت‌های هر یک و در فصل سه میکروهای نوع MEGAAVR و فیوز بیت‌های هر یک مورد بررسی قرار گرفته است. در فصل سوم، دو بخش کلاک سیستم (۱) و (۲) به معرفی انواع کلاک سیستم میکروهای MEGAAVR پرداخته است.

یکی از کامپایلرهای قوی میکروهای AVR ، BASCOM است که در این کتاب قصد داریم به معرفی آن بپردازیم. به همین دلیل فصل چهارم را اختصاص به معرفی محیط BASCOM و منوهای مربوطه داده‌ایم و در انتهای فصل مدارهای مختلفPROGRAMMER ارائه شده است. برای برنامه‌نویسی در محیط BASCOM لازم است با دستورات آن آشنا شویم. فصل پنجم به معرفی تمام دستورات این محیط پرداخته است.برای استفاده از امکانات AVR بایستی وسیله مورد نظر در محیط BASCOM پیکره‌بندی (CONFIG) شود. فصل ششم به معرفی نحوه پیکره‌بندی تمام امکانات میکروهای AVR پرداخته است. در فصل هفتم آشنایی و کار با حافظه های سریال 2-WIRE ارائه شده است. در نهایت کار با برد همراه کتاب و برنامه چند پروژه عملی نیز در فصل هشتم آمده است. خطاهای میکروکنترلرهای AVR در ویرایشهای مختلف به همراه کُدهای خطا در محیط BASCOM و جدول INSTRUCTION SET میکروهای AVR و اطلاعات دیگری از محیط BASCOM در پیوست آمده است.

مختصری راجع به AVR

زبانهای سطح بالا یا همان HLL (HIGH LEVEL LANGUAGES) به سرعت در حال تبدیل شدن به زبان برنامه‌نویسی استاندارد برای میکروکنترلرها (MCU) حتی برای میکروهای 8 بیتی کوچک هستند. زبان برنامه‌نویسی BASIC و C بیشترین استفاده را در برنامه‌نویسی میکروها دارند ولی در اکثر کاربردها کدهای بیشتری را نسبت به زبان برنامه‌نویسی اسمبلی تولید می‌کنند. ATMEL ایجاد تحولی در معماری ، جهت کاهش کد به مقدار مینیمم را درک کرد که نتیجه این تحول میکروکنترلرهای AVR هستند که علاوه بر کاهش و بهینه‌سازی مقدار کدها به طور واقع عملیات را تنها در یک کلاک سیکل توسط معماری RISC (REDUCED INSTRUCTION SET COMPUTER) انجام می‌دهند و از 32 رجیستر همه منظوره (ACCUMULATORS) استفاده می‌کنند که باعث شده 4 تا 12 بار سریعتر از میکروهای مورد استفاده کنونی باشند.

تکنولوژی حافظه کم مصرف غیرفرّار شرکت ATMEL برای برنامه‌ریزی AVR ها مورد استفاده قرار گرفته است در نتیجه حافظه‌های FLASH و EEPROM در داخل مدار قابل

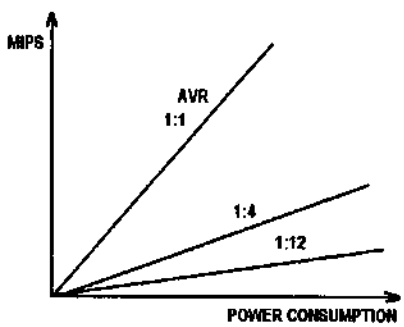
برنامه‌ریزی (ISP) هستند. میکروکنترلرهای اولیه AVR دارای 1 ، 2 و 8 کیلوبایت حافظه FLASH و به صورت کلمات 16 بیتی سازماندهی شده بودند.

AVR ها به عنوان میکروهای RISC با دستورات فراوان طراحی شده‌اند که باعث می‌شود حجم کد تولید شده کم و سرعت بالاتری بدست آید.

عملیات تک سیکل

با انجام تک سیکل دستورات، کلاک اسیلاتور با کلاک داخلی سیستم یکی می‌شود. هیچ تقسیم کننده‌ای در داخل AVR قرار ندارد که ایجاد اختلاف فاز کلاک کند. اکثر میکروها کلاک اسیلاتور به سیستم را با نسبت 1:4 یا 1:12 تقسیم می‌کنند که خود باعث کاهش سرعت می‌شود. بنابراین AVR ها 4 تا 12 بار سریعتر و مصرف آنها نیز 12 - 4 بار نسبت به میکروکنترلرهای مصرفی کنونی کمتر است .

نمودار زیر افزایش MIPS (MILLION INSTRUCTION PER SECONDS) را به علت انجام عملیات تک سیکل AVR (نسبت 1:1) در مقایسه با نسبت‌های 1:4 و 1:12 در دیگر میکروها را نشان می‌دهد.



نمودار مقایسه افزایشMIPS/POWER Consumption در AVR با دیگر میکروکنترلرها

طراحی برای زبان‌های BASIC و C

با آنکه زبانهای BASIC و C بیشترین استفاده را در زبانهای HLL دارند. اما تا امروز معماری بیشتر میکروها برای زبان اسمبلی طراحی شده و کمتر از زبانهای HLL حمایت کرده‌اند.

هدف ATMEL طراحی معماری بود که هم برای زبان اسمبلی و هم زبانهای HLL مفید باشد. به طور مثال در زبانهای C و BASIC می‌توان یک متغییر محلی به جای متغییر سراسری در داخل زیربرنامه تعریف کرد، در این صورت فقط در زمان اجرای زیربرنامه مکانی از حافظه RAM برای متغییر اشغال می‌شود در صورتی که اگر متغییری به عنوان سراسری تعریف گردد در تمام وقت مکانی از حافظه FLASH ROM را اشغال کرده است.

برای دسترسی سریعتر به متغییرهای محلی و کاهش کد، نیاز به افزایش رجیسترهای همه

منظوره است. AVR ها دارای 32 رجیستر هستند که مستقیماً به ALU (ARITHMETIC LOGIC UNIT) متصل شده‌اند، و تنها در یک کلاک سیکل به این واحد دسترسی پیدا می‌کنند. سه جفت از این رجیسترها می‌توانند بعنوان رجیسترهای 16 بیتی استفاده شوند.

برنامه زیر نشان می‌دهد که چگونه تعداد مناسب رجیسترهای همه منظوره (در AVR ها) می‌توانند با معماری CISC با یک ACCUMULATOR مقایسه گردند. برای این منظور می خواهیم از معادله (1)، A را بدست بیاوریم. می بینیم که با کدهای AVR این محاسبه در عرض 4 کلاک سیکل و با کدهای CISC در عرض 48-96 کلاک سیکل انجام می‌گیرد.

Function(1): A = ((A .and. 84h) + (B.eor.C).or.80h

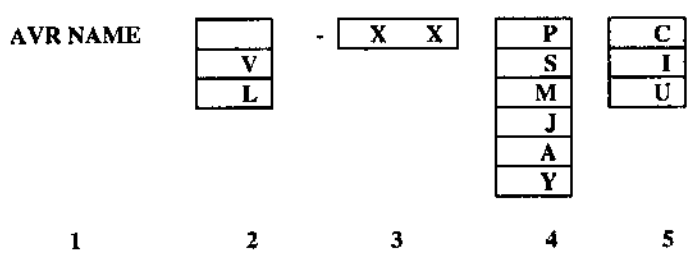
AVR CODE	CISC CODE
EOR B,C	MOV ACC,C
ANDI A,#84h	EOR ACC,B
ADD A,B	MOV TMP,ACC
ORI B,#80h	MOV ACC,A
	AND ACC,#84h
	ADD ACC,TMP
	OR ACC,#80h
	MOV A,ACC
4 clocks	48-96 clocks
8 bytes	12-16 bytes

نتیجه تمام موارد بحث شده، میکروکنترلرهای AVR با سرعت بالا و سازماندهی RISC هستند و به سه نوع AT90S یا AVR ، TINYAVR و MEGAAVR تقسیم‌بندی شده‌اند.

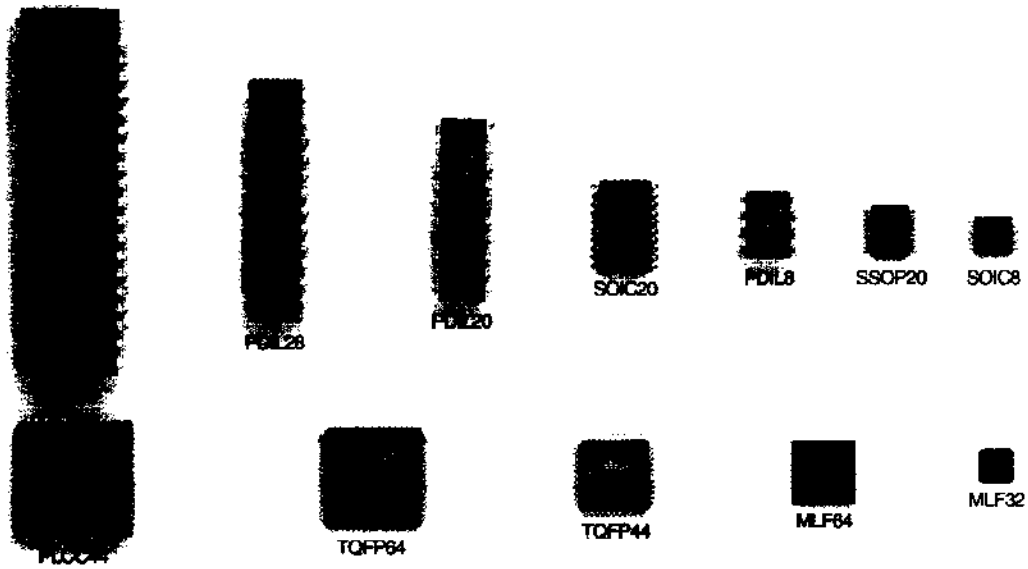
بسته بندی میکروکنترلرهای AVR

AVR ها دارای بسته بندی های متنوعی از جمله DIP,PLCC,TQFP,MLF, SOIC,SSOP هستند. شکل ۰-۱ نمایی گرافیکی از مدلهای مختلف از بسته بندیها AVR را نشان میدهد.

میکروهای AVR دارای پسوندهایی هستند. شکل ۰-۲ انواع پسوند را بر روی بسته بندی AVR ها نشان میدهد که در ادامه به تعریف هر یک می پردازیم.



شکل ۰-۲ انواع پسوندهای AVR بر روی بسته بندی



شکل ۰-۱ نمای گرافیکی بسته بندیهای AVR

1 . نام میکروکنترلر به طور مثال ATTINY2313,ATMEGA128 که معمولاً اعداد آنها مقدار حافظه FLASH را مشخص می کنند.

2 . طبق جدول ۰-۱ مشخص کننده فرکانس و ولتاژ کاری میکرو است.

3 . مشخص کننده بیشترین فرکانس کاری AVR است.به طور مثال 8 به معنای فرکانس کاری ماکسیمم 8MHZ است.

4 . طبق جدول ۰-۲ مشخص کننده نوع بسته بندی (PACKAGE) است.

5 . طبق جدول ۰-۳ مشخص کننده محدوده دمای کاری است.

جدول ۰-۱ فرکانس و ولتاژ کاری AVR ها

مثال	میکروکنترلر	محدوده تغذیه	فرکانس کاری
ATTINY2313V	AVR با پسوند V	1.8-5.5V	0-4MHZ
ATMEGA8L	AVR با پسوند L	2.7-5.5V	0-8MHZ
ATMEGA32	AVR بدون پسوند	4.0-5.5V	0-16MHZ

جدول ۰-۲ نوع بسته بندی AVR ها

علامت	نوع بسته بندی
P	Plastic Dual Inline Package (PDIP)
S	Plastic Gull Wing Small Outline Package (SOIC)
M	Micro Lead Frame Package (MLF)
J	Plastic J-leaded Chip Carrier (PLCC)
A	Thin Profile Plastic Quad Flat Package (TQFP)
Y	Plastic Shrink Small Outline Package (SSOP)

جدول ۰-۳ محدوده دمای کاری AVR ها

علامت	نوع بسته بندی	محدوده دمای کاری
C	COMMERCIAL	(0°C TO 70°C)
I	INDUSTRIAL	(-40°C TO 85°C)
U	INDUSTRIAL (Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances)	(-40°C TO 85°C)

سپاسگزاری

- در ادامه لازم است از همسرم، خانم مهندس بامروّت که زحمت ویراست و تهیه برد کتاب را عهده گرفتند کمال تشکر را داشته باشم.
- همچنین از مدیریت محترم انتشارات نص، آقای مهندس زارع جهت چاپ کتاب نیز قدردانی می نمایم.

با تمام تلاشی که صورت گرفته تا اشکلات احتمالی کتاب به حداقل برسد اما به هیچ عنوان کتاب خالی از اشکال نیست. امید است دانش‌پژوهان و اساتید محترم باز هم مرا از انتقادات و نظرات خود جهت ارائه کاری بهتر بهره‌مند سازند. لطفاً نظرات خود را به آدرس الکترونیکی مؤلف یا ناشر ارسال کنید. در عنوان کلمه [AVR-KAHEH] را قید کنید.

آدرس الکترونیکی ناشر Nasspublish.co@gmail.com
آدرس الکترونیکی مؤلف alikaheh@yahoo.com

علی کاهه – بهار ۸۶

فهرست مطالب

فصل ۱ : میکروکنترلرهای TINYAVR

۱-۱ خصوصیات ATTINY10، ATTINY11 ، ATTINY12 ..... ۲۰
فیوز بیت‌های ATTINY11 و ATTINY12 ..... ۲۱
منابع کلاک ATTINY11/12 ..... ۲۴
۱-۲ خصوصیات ATTINY15L ۲۵
فیوز بیت‌های ATTINY15L ۲۷
۱-۳ خصوصیات ATTINY26 , ATTINY26L ۲۸
فیوز بیت‌های ATTINY26 ..... ۳۰
کلاک سیستم ATTINY26 ..... ۳۲
۱-۴ خصوصیات ATTINY28V ، ATTINY28L ۳۹
فیوز بیت‌های ATTINY28 ..... ۴۰
منابع کلاک ATTINY28 ..... ۴۰
۱-۵ خصوصیات ATTINY13 ، ATTINY13L ۴۲
فیوز بیت‌های ATTINY13 ..... ۴۴
۱-۶ خصوصیات ATTINY22 ، ATTINY22L ۴۶
فیوز بیت‌های ATTINY22 ..... ۴۷
۱-۷ خصوصیات ATTINY2313 ، ATTINY2313L ۴۷
فیوز بیت‌های ATTINY2313 ..... ۴۹

فصل ۲ : میکروکنترلرهای AVR

۲-۱ خصوصیات AT90S1200 ..... ۵۴
فیوز بیت‌های AT90S1200 ..... ۵۵
۲-۲خصوصیات AT90S2313 ..... ۵۵
فیوز بیت‌های AT90S2313 ..... ۵۷
۲-۲ خصوصیات AT90S2323/LS2323/S2343/LS2343 ..... ۵۷
فیوز بیت‌های AT90S/LS2323 ..... ۵۹
فیوز بیت‌های AT90S/LS2343 ..... ۵۹
۲-۳ خصوصیات AT90S2333/LS2333/S4433/LS4433 ..... ۶۰
فیوز بیت‌های AT90S2333/4433 ..... ۶۱
۲-۴ خصوصیات AT90S8515 ..... ۶۳
فیوز بیت‌های AT90S8515 ..... ۶۴
۲-۵ خصوصیات AT90S8535/ LS8535 ..... ۶۵
فیوز بیت‌های AT90S8535 ..... ۶۰

فصل ۳ : میکروکنترلرهای MEGAAVR

۳-۱ خصوصیات ATMEGA323 , ATMEGA323L ۷۰
فیوز بیت‌های ATMEGA323 ..... ۷۱
۳-۲ خصوصیات ATMEGA32 ,ATMEGA32L ۷۴
فیوز بیت‌های ATMEGA32 ..... ۷۶
۳-۳ خصوصیات ATMEGA128 , ATMEGA128L ۷۹
فیوز بیت‌های ATMEGA128 ..... ۸۱
۳-۴ خصوصیات ATMEGA163 , ATMEGA163L ۸۳
فیوز بیت‌های ATMEGA163 ..... ۸۵
۳-۵ خصوصیات ATMEGA8 , ATMEGA8L ۸۷
فیوز بیت‌های ATMEGA8 ..... ۸۹
۳-۶ خصوصیات ATMEGA8515 , ATMEGA8515L ۹۲
فیوز بیت‌های ATMEGA8515 ..... ۹۳
۳-۷ خصوصیات ATMEGA8535 , ATMEGA8535L ۹۵
فیوز بیت‌های ATMEGA8535 ..... ۹۷
۳-۸ خصوصیات ATMEGA161 , ATMEGA161L ۹۹
فیوز بیت‌های ATMEGA161 ..... ۱۰۱
۳-۹ خصوصیات ATMEGA162 , ATMEGA162V ۱۰۲
فیوز بیت‌های ATMEGA162 ..... ۱۰۴
۳-۱۰ خصوصیات ATMEGA16 , ATMEGA16L ۱۰۶
فیوز بیت‌های ATMEGA16 ..... ۱۰۸
۳-۱۱ خصوصیات ATMEGA103 , ATMEGA103L ۱۱۱
فیوز بیت‌های ATMEGA103 ..... ۱۱۳
۳-۱۲ خصوصیات ATMEGA169 , ATMEGA169L , ATMEGA169V ۱۱۳
فیوز بیت‌های ATMEGA169 ..... ۱۱۶
۳-۱۳ خصوصیات ATMEGA64 , ATMEGA64L ۱۱۸
فیوز بیت‌های ATMEGA64 ..... ۱۱۹
۳-۱۴ خصوصیات ATMEGA48/88/168 , ATMEGA48L/88L/168L ۱۲۲
فیوز بیت‌های ATMEGA48/88/168 ..... ۱۲۴
۳-۱۵ خصوصیات ATMEGA325/3250/645/6450 ..... ۱۲۷
فیوز بیت‌های ATMEGA325/3250/645/6450 ..... ۱۲۹
۳-۱۶ خصوصیات ATMEGA329/3290/649/6490 ..... ۱۳۱
فیوز بیت‌های ATMEGA329/3290/649/6490 ..... ۱۳۲
۳-۱۷ خصوصیات ATMEGA165 , ATMEGA165V ۱۳۴
فیوز بیت‌های ATMEGA64 ..... ۱۳۶
۳-۱۸ کلاک سیستم (۱) ۱۳۸
۳-۱۹ کلاک سیستم (۲) ۱۴۴

فصل ۴ : محیط برنامه نویسی BASCOM AVR

۴-۱ معرفی منوهای محیط BASCOM ۱۵۲
۴-۲ معرفی محیط شبیه‌سازی (SIMULATOR) ۱۵۷
۴-۳ معرفی محیط برنامه‌ریزی ۱۶۰
۴-۴ معرفی محیط TERMINAL EMULATORE ۱۶۲
۴-۵ ساخت ISP PROGRAMMER ۱۶۳

فصل ۵ : دستورات و توابع محیط برنامه نویسی BASCOM

۵-۱ بدنه یک برنامه در محیط BASCOM ۱۶۶
معرفی میکرو، کریستال ۱۶۶
اسمبلی و بیسیک، یاداشت ۱۶۷
آدرس شروع برنامه‌ریزی حافظه FLASH ،تعیین کلاک و پایان برنامه ۱۶۸
۵-۲ اعداد و متغییرها و جداول LOOKUP ۱۶۹
دیمانسیون متغییر ۱۶۹
دستورات CONST و ALIAS ۱۷۰
دستورات INCR ،CHR، INSTR ۱۷۱
دستورات LOW ، HIGT ، DECR و CHECKSUM ۱۷۲
دستورات LEFT ، RIGHT، UCASE، LCASE ۱۷۳
دستورات SWAP ، LTRIM و LEN ۱۷۴
دستورات SPACE و ROTATE و تابع MID ۱۷۵
توابع FUSING و FORMAT ۱۷۶
جداول LOOKUP و LOOKUPSTR ۱۷۷
۵-۳ توابع ریاضی و محاسباتی ۱۷۷
عملگرهای ریاضی و منطقی و تابع ۱۷۷
توابع EXP ، ABS ۱۷۸
توابع LOG10، LOG، RND ۱۷۹
توابع TAN، SIN، COS ۱۸۰
توابع SINH ، TANH، COSH و ASIN ۱۸۱
توابع ATN2،ATN و ACOS ۱۸۲
توابع RAD2DEG و DEG2RAD ۱۸۳
تابع ROUND ۱۸۴
۵-۴ توابع تبدیل کدها و متغییرها ۱۸۴
توابع ASC و HEX ۱۸۴
توابع MAKEDEC ، HEXVAL ، MAKEBCD و MAKEINT ۱۸۵
توابع STRING ، STR ، VAL و BIN2GREY ۱۸۶
تابع GRAY2BIN ۱۸۷
۵-۵ رجیسترها و آدرس‌های حافظه ۱۸۷
دستورات SET و TOGGLE ۱۸۷

دستورات RESET، BITWAIT، CPEEKH و CPEEK ۱۸۸
دستورات OUT، LOADADR و INP ۱۸۹
دستورات POKE، PEEK و VARPTR ۱۹۰
۵-۶ دستورالعمل‌های حلقه و پرش ۱۹۰
دستورالعملهای GOTO و JMP ۱۹۰
دستورالعملهای DO-LOOP و FOR-NEXT ۱۹۱
دستورالعملهای WHILE-WEND و IF ۱۹۲
دستورالعمل CASE ۱۹۳
دستورالعمل EXIT و ON VALUE ۱۹۴
۵-۷ ایجاد تاخیر در برنامه ۱۹۵
دستورات DELAY و WAITUS ۱۹۵
دستورات WAIT و WAITms ۱۹۶
۵-۸ زیربرنامه و تابع ۱۹۶
معرفی تابع ۱۹۶
معرفی زیربرنامه ۱۹۷
فراخوانی با CALL ۱۹۸
بکارگیری متغییر محلی با LOCAL ۱۹۹
دستور GOSUB ۲۰۰

فصل ۶ : پیکره‌بندی و کار با امکانات AVR در BASCOM

۶-۱ پیکره‌بندی پورت‌ها ۲۰۲
بررسی پورت‌های میکرو ATMEGA32 ..... ۲۰۳
پورت A ۲۰۳
پورت B ۲۰۴
پورت C ۲۰۶
پورت D ۲۰۹
۶-۲ پیکره‌بندی صفحه‌کلید 4×4 ..... ۲۱۱
دستور GETKBD() ۲۱۲
۶-۳ پیکره‌بندی صفحه‌کلید کامپیوتر ۲۱۲
دستور GETATKBD ۲۱۴
۶-۴ پیکره‌بندی LCD ۲۱۶
تعیین نوع LCD ۲۱۶
تعیین آدرس باس LCD و دستور LCD ۲۱۷
دستورات CLS، DISPLAY و CURSOR ۲۱۸
دستورات HOME،LOCATE، SHIFT CURSOR و SHIFTLCD ۲۱۹
دستورات FOURTH LINE،UPPERLINE، LOWERLINE و THIRDLINE و تابع DEFLCDCHAR ۲۲۰
۶-۵ پیکره‌بندی تایمر/کانترها ۲۲۱
معرفی تایمر/ کانتر صفر و رجیسترها ۲۲۱

پیکره‌بندی تایمر/کانتر صفر در محیط BASCOM ۲۲۲
معرفی تایمر/ کانتر یک و رجیسترها ۲۲۴
پیکره‌بندی تایمر/کانتر یک در محیط BASCOM ۲۳۰
معرفی تایمر/ کانتر دو رجیسترها ۲۳۶
پیکره‌بندی تایمر/کانتر دو در محیط BASCOM ۲۳۹
پیکره‌بندی تایمر/کانتر سه در محیط BASCOM ۲۴۳
۶-۶ ارتباط با پورت سریال ۲۴۸
UART سخت‌افزاری ۲۴۸
تعیین و تغییر میزان باود ۲۴۹
ارسال داده در حالت UART سخت‌افزاری ۲۴۹
پیکره‌بندی SERIALOUT ۲۴۹
دستورات PRINTBIN و PRINT ۲۵۰
دریافت داده در حالت UART سخت‌افزاری(بدون وقفه) ۲۵۱
پیکره‌بندی SERIALIN و دستورات INKEY و WAITKEY ۲۵۱
دستورات INPUTBIN و INPUT ۲۵۲
دستور INPUTHEX ۲۵۳
دریافت داده در حالت UART سخت‌افزاری(با وقفه) ۲۵۴
دستورات ENABLE URXC,URXC1 ..... ۲۵۴
تعریف برچسب اینتراپت ۲۵۴
UART نرم‌افزاری ۲۵۵
تعیین و تغییر میزان باود ۲۵۶
ارسال داده در حالت UART نرم‌افزاری ۲۵۷
دستورات PRINT و PRINTBIN ۲۵۷
دریافت داده در حالت UART نرم‌افزاری ۲۵۸
دستورات WAITKEY ، INKEY و INPUT ۲۵۸
دستورات INPUTBIN و INPUTHEX ۲۵۹
اتصال AVR به RS232 ..... ۲۶۰
تراشه MAX232 ، MAX233 ..... ۲۶۰
۶-۷ مبدل آنالوگ به دیجیتال (ANALOG TO DIGITAL CONVERTOR) ۲۶۱
پیکره‌بندی ADC در محیط BASCOM ۲۶۳
دستور GETADC ۲۶۴
کار با وقفه ADC ۲۶۵
۶-۸ مقایسه‌کننده آنالوگ ۲۶۶
پیکره‌بندی مقایسه‌کننده آنالوگ در BASCOM ۲۶۶
۶-۹ پیکره‌بندی GRAPHICAL LCD DISPLAY ۲۶۷
پیکره‌بندی T6963 GRAPHICAL LCD DISPLAY ۲۶۷
دستورات CLS ،CLS G ،CLS TEXT ،LCD و PSET ،LOACTE ۲۶۹
دستورات LINE ، CIRCLE ،CURSOR ، SHOWPIC و برچسب $BGF ۲۷۰

پیکره‌بندی KS108(SED) GRAPHICAL LCD DISPLAY ۲۷۱
دستورات LCDAT ،CLS TEXT ،CLS GRAPH، CLS ۲۷۲
دستورات LINE ، CURSOR ،PSET ،LOACTE ، CIRCLE ، SHOWPIC و برچسب $BGF ۲۷۳
۶-۱۰ ارتباط سریال SPI ۲۷۴
ارتباط SPI و رجیسترهای مربوطه ۲۷۶
پیکره‌بندی SPI در محیط BASCOM ۲۷۹
دستورات مربوط به ارتباط SPI ۲۸۰
۶-۱۱ پیکره‌بندی ارتباط سریال I2C (2-WIRE) ۲۸۱
تعیین کلاک I2C ۲۸۱
تعیین پایه SDA و SCL و دستور I2C RECEIVE ۲۸۲
دستورات I2CSTART , I2CSTOP , I2CRBYTE , I2CWBYTE , I2C SEND ۲۸۳
۶-۱۲ پیکره‌بندی WATCHDOG ۲۸۴
۶-۱۳ وقفه‌ها ۲۸۵
دستورات ENABLE , DISABLE ۲۸۵
دستور ON INTERRUPT و پیکره‌بندی وقفه‌های خارجی ۲۸۵
۶-۱۴ حافظه EEPROM داخلی میکروهای AVR ۲۸۷
دستورات WRITEEEPROM و READEEPROM ۲۸۸
۶-۱۵ مُدهای SLEEP ۲۸۸
مُدهای IDLE و ADC NOISE REDUCTION ۲۸۸
مُدهای POWER-DOWN ،POWER- SAVE ، STANDBY و EXTENDED -STANDBY ۲۸۹
دستورات اجرای مُدهای SLEEP در BASCOM ۲۹۰
۶-۱۶ کار با حافظه BOOT (BOOT LOADER FLASH SECTION) ۲۹۱
وارد شدن به برنامه BOOTLOADER ۲۹۱
کار با BOOTLOADER در محیط BASCOM ۲۹۱
۶-۱۷ دستورات و پیکره‌بندی‌های جانبی ۲۹۳
دستور DEBOUNCE و PULSEOUT ۲۹۳
دستور PULSEIN ۲۹۴
دستور SOUND ۲۹۵

فصل ۷ : حافظه‌های EEPROM سریال 2-WIRE

۷-۱ معرفی انواع حافظه‌های سری..AT24 ..... ۲۹۷
۷-۲ معرفی پایه‌ها ۲۹۹
۷-۳ طرز کار حافظه ۳۰۰
۷-۴ آدرس‌دهی سخت‌افزاری حافظه ۳۰۱
۷-۵ انواع عملیات نوشتن حافظه ۳۰۲
۷-۶ انواع عملیات خواندن حافظه ۳۰۳

فصل ۸ : پروژه های عملی

۸-۱ مدار و PCB برد کتاب ۳۰۶
۸-۲ اتصال کلید به میکرو توسط دستور DEBOUNCE ۳۱۰
۸-۳ اسکن صفحه‌کلید ۴×۴ ..... ۳۱۲
۸-۴ اسکن صفحه‌کلید کامپیوتر ۳۱۴
۸-۵ برنامه ساعت ۳۱۶
۸-۶ تابلو روان توسط LCD ۳۱۷
۸-۷ نمایش دما بر روی LCD توسط سنسور دمای LM335 ..... ۳۱۸
۸-۸ خروجی آنالوگ توسط پالس PWM و VISUAL BASIC ۳۲۱
۸-۹کار با 7Segment سه تایی ۳۲۵
۸-۱۰ مانیتورینگ دما توسط نرم افزار VISUAL BASIC 6.0 ..... ۳۲۷
۸-۱۱ فرکانس‌متر دیجیتال ۳۳۱
۸-۱۲ اسکن صفحه‌کلید ۴×۴ توسط انکدر MM74C922 ..... ۳۳۲
۸-۱۳ موتور پله‌ای STEPER MOTOR ۳۳۵
۸-۱۴ ارتباط سریال دو میکرو از طریق باس SPI ۳۳۸
۸-۱۵ ارتباط با حافظه EEPROM سریال AT24C256 ..... ۳۴۲
۸-۱۶ مقایسه‌کننده آنالوگ و CAPTURE تایمر یک ۳۴۶
۸-۱۷ RTC (REAL TIME CLOCK) ۳۴۶
۸-۱۸ LCD گرافیکی ۳۴۹
۸-۱۹ کار با محیط TERMINAL EMULATOR ۳۵۰

پیوست الف

خطاهای میکروکنترلرهای AVR ۳۵۴
AVR Instruction Set Table ۳۷۱
BASCOM Editor Keys ۳۷۵
reserved BASCOM statements or character ۳۷۶
Error Codes In bascom AVR ۳۷۷
Bascom PC Keyboard Key Code ۳۸۰

پیوست ب

برد مدار چاپی آموزشی

۱

میکروکنترلرهای TINYAVR

در این فصل به معرفی میکروکنترلرهای نوع TINYAVR از سری میکروکنترلرهای AVR شرکت ATMEL می‌پردازیم. در این فصل خصوصیات و قابلیت‌های هر یک از میکروهای نوع TINYAVR تشریح و در ادامه فیوز بیت‌های هر یک به طور کامل بررسی شده‌اند. فیوز بیت‌ها قسمتی از حافظه FLASH هستند که امکاناتی را در اختیار کاربر قرار می‌دهند. فیوز بیت‌ها با ERASE میکرو از بین نمی‌روند و می‌توانند توسط بیت‌های قفل مربوطه، قفل شوند. کلاک سیستم هر یک از میکروها در صورت نیاز به توضیح بیشتر بلافاصله بعد از فیوز بیت‌ها گفته شده است. خانواده TINYAVR جهت کاهش قیمت و صرف وقت برای پروژه های کاربران بهینه سازی شده‌اند.

در زیر به طور نمونه تعدادی از کاربردهای انواع خانواده TINYAVR را می بینید.

ATtiny11 : External Logic, Mechanical Switch Replacement, Frequency Controller
ATtiny12 : Security Surveillance, Remote Keyless Entry, Gas Engine Controller
ATtiny15 : Refrigerator Control, Sensors, Emergency Lighting
ATtiny26 : Light Ballast, Battery Chargers, Laptop Mouse

اهداف

۱. آشنایی کامل با انواع میکروهای TINYAVR
۲. آشنایی کامل با فیوز بیت‌های هر یک از میکروها
۳. توانایی برنامه‌ریزی فیوز بیت‌های هر یک از میکروها
۴. توانایی برنامه‌ریزی فیوز بیت‌ها برای تعیین کلاک سیستم دلخواه

۱-۱ خصوصیات ATtiny10، ATtiny11 ، ATtiny12

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 90 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 8MIPS در فرکانس 8MHZ
- **حافظه ، برنامه و داده غیر فرار**
  - 1K بایت حافظه FLASH قابل برنامه‌ریزی داخلی

    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 64 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

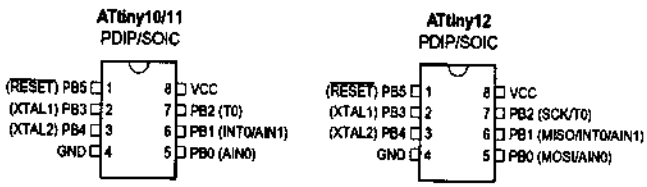
    پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - یک تایمر – کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا
  - یک مقایسه‌گر آنالوگ داخلی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - وقفه در اثر تغییر وضعیت پایه
- **خصوصیات ویژه میکروکنترلر**
  - تغذیه کم در مُدهای IDLE وPOWERDOWN
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - ارتباط سریال SPI برای برنامه‌ریزی ATtiny12 در داخل مدار IN SYSTEM PROGRAMMING
  - POWER – ON RESET CIRCUIT برای ATtiny12
  - قابل انتخاب بودن اسیلاتور RC داخلی جهت کاهش قسمتهای خارجی برای ATtiny12
  - عملکرد کاملاً ثابت
  - توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS
- **توان مصرفی در 4MHZ ، 3V ، 25^0C**
  - حالت فعال 2.2mA (ACTIVE MODE)
  - درحالت بی‌کاری 0.5mA (IDLE MODE)
  - در حالت POWER – DOWN : < 1μA
- **ولتاژهای عملیاتی (کاری)**
  - 1.5V تا 5.5V برای (ATtiny12V-1)

— 2.7V تا 5.5V برای (ATtiny11L-2 و ATtiny12L-4)
— 4V تا 5.5V برای (ATtiny11-6 و ATtiny12-8)

• **فرکانسهای کاری**
— 0MHZ تا 1.2MHZ برای (ATtiny12V-1)
— 0MHZ تا 2MHZ برای (ATtiny11L-2)
— 0MHZ تا 4MHZ برای (ATtiny12L-4)
— 0MHZ تا 6MHZ برای (ATtiny11-6)
— 0MHZ تا 8MHZ برای (ATtiny12-8)

• **انواع بسته‌بندی**
— 8 پایه در انواع PDIP و SOIC

• **ترکیب بسته‌بندی**



شکل ۱-۱ ترکیب بسته بندی ATtiny12 و ATtiny11

فیوز بیت‌های ATtiny11 و ATtiny12

فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات فیوز بیتها ، (0) به معنای برنامه‌ریزی شدن و (1) به معنای برنامه‌ریزی نشدن بیت است.

فیوز بیت‌های ATtiny11

این میکرو دارای 5 فیوز بیت (FSTRT , RSTDISBL , CKSEL2..0) به شرح زیر است :

FSTRT : این بیت با توجه به جدول ۱-۱ مشخص کننده زمان شروع (START-UP) از ری‌ست یا مُدهای SLEEP است. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده (1) است.

RSTDISBL : به معنای RESET DISABLE است. با برنامه‌ریزی این بیت می‌توان از پایه ری‌ست خارجی (PB5) به عنوان I/O عمومی استفاده کرد. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

CKSEL2..0 : با برنامه‌ریزی کردن این بیت‌ها کلاک سیستم را می‌توان با توجه به جدول ۱-۲ در مُدهای مختلف تغییر داد. این سه بیت به صورت پیش‌فرض "100" برنامه ریزی شده اند و میکرو با اسیلاتور RC داخلی و فرکانس ثابت 1.0MHZ کار می‌کند.

جدول ۱-۱ تعیین زمان START-UP برای ATTINY11 به ازاء VCC=2.7v

SELECTED CLOCK OPTION	START UP TIME	
	FSTRT UNPROGRAMMED	FSTRT PROGRAMMED
EXTERNAL CRYSTAL	67ms	4.2ms
EXTERNAL CERAMIC RESONATOR	67ms	4.2ms
EXTERNAL LOW-FREQUENCY CRYSTAL	4.2s	4.2s
EXTERNAL RC OSCILLATOR	4.2ms	67µs
INTERNAL RC OSCILLATOR	4.2ms	67µs
EXTERNAL CLOCK	4.2ms	5 CLOCK FROM RESET , 2 CLOCK FROM POWER-DOWN

جدول ۱-۲ انتخاب کلاک سیستم برای ATTINY11

DEVICE CLOCKING OPTIONS	CKSEL2..0
EXTERNAL CRYSTAL/CERAMIC RESONATOR	111
EXTERNAL LOW-FREQUENCY CRYSTAL	110
EXTERNAL RC OSCILLATOR	101
INTERNAL RC OSCILLATOR(FIX 1.0MHZ)	100
EXTERNAL CLOCK	000
RESERVED	OTHER OPTIONS

فیوز بیت‌های ATtiny12

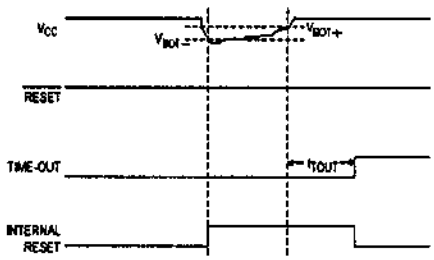
این میکرو دارای 8 فیوز بیت (SPIEN, BODLEVEL ,BODEN, RSTDISBL ,CKSEL3..0) است.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 1.8V پایین‌تر شود (شکل۱-۲)ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند ولی زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. لازم به تذکر است که این بیت به همراه بیت‌های CKSEL3..0 زمان شروع (START UP) میکرو را نیز تعیین می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN - OUT این بیت بایستی طبق جدول ۱-۳ برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

RSTDISBL : به معنای RESET DISABLE است. با برنامه‌ریزی این بیت (0 نوشتن) می‌توان از پایه RESET خارجی به عنوان I/O عمومی استفاده کرد. RSTDISBL به صورت پیش‌فرض برنامه‌ریزی نشده (1) است.



شکل ۱-۲ زمان بندی ری‌ست BROWN-OUT

جدول ۱-۳ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=1.8V
00	AT VCC=2.7V

CKSEL3..0 : با برنامه‌ریزی کردن این بیت‌ها و بیت BODLEVEL کلاک سیستم وزمان Start-up را می‌توان با توجه به جدول ۱-۴ در مُدهای مختلف انتخاب کرد. این سه بیت به صورت پیش‌فرض "0010" و میکرو با اسیلاتور RC داخلی با فرکانس ثابت 1.2MHZ کار می‌کند. کریستال خارجی بین پایه‌های XTAL1 و XTAL2 از میکرو متصل می‌شود.

جدول ۱-۴ انتخاب کلاک سیستم و زمان START-UP برای ATTTINY12

CKSEL3..0	Clock Source	Start-up Time, VCC = 1.8V,BODLEVEL Unprogrammed	Start-up Time, VCC=2.7V, BODLEVEL Programmed
1111	Ext. Crystal/Ceramic Resonator	1K CK	1K CK
1110	Ext. Crystal/Ceramic Resonator	3.6 ms + 1K CK	4.2 ms + 1K CK
1101	Ext. Crystal/Ceramic Resonator	57 ms 1K CK	67 ms + 1K CK
1100	Ext. Crystal/Ceramic Resonator	16K CK	16K CK
1011	Ext. Crystal/Ceramic Resonator	3.6 ms + 16K CK	4.2 ms + 16K CK
1010	Ext. Crystal/Ceramic Resonator	57 ms + 16K CK	67 ms + 16K CK
1001	Ext. Low-frequency Crystal	57 ms + 1K CK	67 ms + 1K CK
1000	Ext. Low-frequency Crystal	57 ms + 32K CK	67 ms + 32K CK
0111	Ext. RC Oscillator	6 CK	6 CK
0110	Ext. RC Oscillator	3.6 ms + 6 CK	4.2 ms + 6 CK
0101	Ext. RC Oscillator	57 ms + 6 CK	67 ms + 6 CK
0100	Int. RC Oscillator	6 CK	6 CK

CKSEL3..0	Clock Source	Start-up Time, VCC = 1.8V,BODLEVEL Unprogrammed	Start-up Time, VCC=2.7V, BODLEVEL Programmed
0011	Int. RC Oscillator	3.6 ms + 6 CK	4.2 ms + 6 CK
0010	Int. RC Oscillator	57 ms + 6 CK	67 ms + 6 CK
0001	Ext. Clock	6 CK	6 CK
0000	Ext. Clock	3.6 ms + 6 CK	4.2 ms + 6 CK

ادامه جدول صفحه قبل

منابع کلاک ATTINY11/12

منظور از منابع کلاک انواع کلاکی است که میکرو می تواند با آن کار کند. در این بخش قصد داریم به معرفی منابع کلاک دو میکرو ATTINY11/12 بپردازیم.

اسیلاتور کریستالی خارجی (EXTERNAL CRYSTAL/CERAMIC RESONATOR)

در این حالت کریستال یا نوسانگر سرامیکی (CERAMIC RESONATOR) یا کریستال کوارتز (QUARTZ CRYSTAL) همانطور که در شکل ۱-۳ نشان داده شده است به دو پایه XTAL1 و XTAL2 میکرو وصل می‌شود. خازنهای C1 وC2 برای کریستال‌ها و نوسانگرها یک مقدار بایستی باشند. مقادیر خازنها بستگی به کریستال ، نوسانگر و نویزهای الکترومغناطیسی محیط دارند که مقدار نامی 32PF مناسب است.

اسیلاتور کریستالی فرکانس پایین خارجی(EXTERNAL LOW FREQ CRYSTAL)

برای استفاده از کریستال ساعت 32.768KHZ ، کریستال طبق شکل ۱-۳ به پایه‌های XTAL1 و XTAL2 متصل می‌شود.

اسیلاتور RC کالیبره شده داخلی (INT. RC OSCILLATOR)

اسیلاتور RC کالیبره شده داخلی ، برای ATTINY11 برابر 1MHZ (پیش فرض میکرو) و برای ATTINY12 برابر 1.2MHZ (پیش فرض میکرو)است.

اسیلاتور RC خارجی(EXT. RC OSCILLATOR)

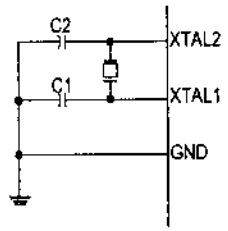
نحوه اتصال RC به پایه‌های XTAL1 در شکل ۱-۴ آمده است. مقدار خازن بایستی حداقل 20PF و مقاومت باید در رنج 100K - 3K باشد. خازن و مقاومت سه فرکانس در جدول ۱-۵ آمده است.

کلاک خارجی (EXT. CLOCK)

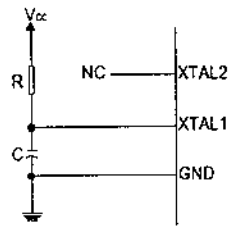
برای راه‌اندازی میکرو در این مُد، کلاک خارجی به پایه XTAL1 طبق شکل ۱-۵ وصل می‌شود.

جدول ۱-۵ مُدهای کاری اسیلاتور RC خارجی

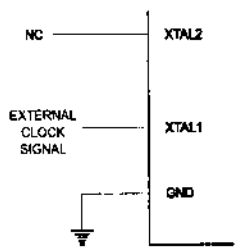
R [kΩ]	C [pF]	FREQUENCY
100	70	100KHZ
31.5	20	1.0MHZ
6.5	20	4.0MHZ



شکل ۱-۳ اتصال کریستال به میکرو در حالت اسیلاتور کریستالی



شکل ۱-۴ اتصال RC به میکرو در حالت اسیلاتور RC خارجی



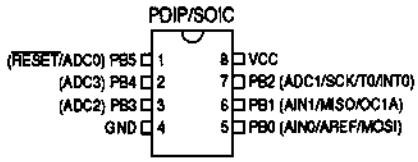
شکل ۱-۵ اتصال کلاک خارجی به پایه میکرو در حالت کلاک خارجی

۱-۲ خصوصیات ATtiny15L

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 90 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
- **حافظه ، برنامه و داده غیر فرار**
  - 1K بایت حافظه FLASH قابل برنامه‌ریزی داخلی
    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 64 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

- **خصوصیات جانبی**
  - ایجاد وقفه با تغییر وضعیت پایه
  - دو تایمر – کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا
  - خروجی PWM ، 8 بیتی با فرکانس 150KHZ
  - 4 کانال مبدل آنالوگ به دیجیتال (ADC)
    یک کانال تفاضلیADC با کنترل گین 20x
  - یک مقایسه‌گر آنالوگ داخلی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
- **خصوصیات ویژه میکروکنترلر**
  - تغذیه کم در مُدهای IDLE و POWERDOWN
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - ارتباط سریال SPI برای برنامه‌ریزی در داخل مدار(IN – SYSTEM PROGRAMMING)
  - مدار POWER – ON RESET
  - مدار BROWN –OUT DETECTION CIRCUIT
  - اسیلاتور داخلی کالیبره شده 1.6MHZ و قابل تنظیم برای کاهش قسمتهای خارجی
  - کلاک داخلی 25.6MHZ برای TIMER / COUNTER
  - عملکرد کاملاً ثابت
  - توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS
- **توان مصرفی در 1.6MHZ ، 3V ، 25^0C**
  - حالت فعال 3mA (ACTIVE MODE)
  - درحالت بی‌کاری 1mA (IDLE MODE)
  - در حالت POWER – DOWN : < 1μA
- **ولتاژهای عملیاتی (کاری)**
  - 2.7V تا 5.5V
- **فرکانسهای کاری**
  - کلاک سیستم داخلی 1.6MHZ
- **خطوط I/O و انواع بسته‌بندی**
  - 6 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 8 پایه (PIN) در انواع PDIP و SOIC

• ترکیب بسته‌بندی



شکل ۱-۶ ترکیب بسته‌بندی ATtiny15L

فیوز بیت‌های ATtiny15L

این میکرو دارای 6 فیوز بیت (SPIEN , BODLEVEL , BODEN , RSTDISBL , CKSEL1,0) است. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

BODLEVEL: زمانی که این بیت برنامه‌ریزی نشده(پیش‌فرض) باشد اگر ولتاژ پایه VCC از ولتاژ 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و میکرو را ری‌ست می‌کند ولی زمانی که بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و میکرو را ری‌ست می‌کند. لازم به تذکر است که این بیت به همراه بیت‌های CKSEL1..0 زمان شروع (START UP) میکرو را نیز تعیین می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN – OUT این بیت بایستی برنامه‌ریزی شده باشد(جدول۱-۶) . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

RSTDISBL : به معنای RESET DISABLE است. با برنامه‌ریزی این بیت (0 نوشتن) می‌توان از پایه RESET خارجی (PB5) به عنوان I/O عمومی استفاده کرد. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده (1) است.

CKSEL1,0 : با برنامه‌ریزی کردن این دو بیت زمان start-up میکرو را با توجه به جدول ۱-۷ می‌توان در مُدهای مختلف تغییر داد. این دو بیت به صورت پیش‌فرض "00" و میکرو با اسیلاتور RC داخلی 1.6MHZ و زمان شروع طولانی 64 ms + 18 CK کار می‌کند.

جدول ۱-۶ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

جدول ۱-۷ انتخاب زمان RESET DELAY (X=DON'T CARE)

BODEN	CKSEL [1:0]	Start-up Time, T TOUT at VCC = 2.7V	Start-up Time, T TOUT at VCC = 5.0V	Recommended Usage
x	00	256 ms + 18 CK	64 ms + 18 CK	BOD disabled, slowly rising power
x	01	256 ms + 18 CK	64 ms + 18 CK	BOD disabled, slowly rising power
x	10	16 ms + 18 CK	4 ms + 18 CK	BOD disabled, quickly rising power
1	11	18 CK + 32 µs	18 CK + 8 µs	BOD disabled
0	11	18 CK + 128 µs	18 CK + 32 µs	BOD enabled

۱-۳ خصوصیات ATtiny26L , ATtiny26

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 118 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 16MIPS در فرکانس 16MHZ
- **حافظه ، برنامه و دادهٔ غیر فرّار**
  - 2K بایت حافظه FLASH قابل برنامه‌ریزی داخلی
    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 128 بایت حافظه SRAM
  - 128 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - ایجاد وقفه با تغییر وضعیت بر روی 11 پایه
  - یک تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا
  - یک تایمر - کانتر 8 بیتی پرسرعت (HIGH-SPEED) با PRESCALER مجزا
  - دو خروجی PWM فرکانس بالا
  - 11 کانال مبدل آنالوگ به دیجیتال (ADC)
    11 کانال ADC با کنترل گین 1x و 20x
    8 کانال ADC تفاضلی
    7 کانال ADC تفاضلی با کنترل گین 1x و 20x

— یک مقایسه‌گر آنالوگ داخلی
— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

• **خصوصیات ویژه میکروکنترلر**
— تغذیه کم در مُدهای IDLE و POWERDOWN
— دارای مُد کاهش نویز (NOISE REDUCTION)
— منابع وقفه (INTERRUPT) داخلی و خارجی
— ارتباط سریال SPI برای برنامه‌ریزی در داخل مدار (IN – SYSTEM PROGRAMMING)
— قابلیت ارتباط سریال USI (UNIVERSAL SERIAL INTERFACE)
— مدار POWER – ON RESET CIRCUIT
— مدار BROWN –OUT DETECTION CIRCUIT
— اسیلاتور داخلی برای کاهش قسمتهای خارجی
— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

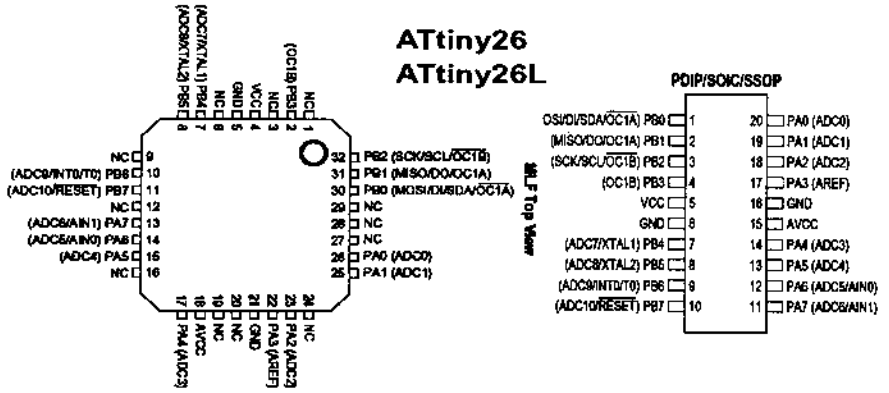
• **توان مصرفی در1MHZ 3V ، 25^0C برای (ATTINT26L)**
— حالت فعال (ACTIVE MODE)
در 16MHZ و 5V و 25^0C : 15mA
در 1MHZ و 3V و 25^0C : 0.70mA
— درحالت بی‌کاری (IDLE MODE)
در 1MHZ و 3V و 25^0C : 0.18mA
— در حالت POWER – DOWN : < 1μA

• **ولتاژهای عملیاتی (کاری)**
— 2.7V تا 5.5V برای (ATtiny26L)
— 4.5V تا 5.5V برای (ATtiny26)

• **فرکانسهای کاری**
— 0MHZ تا 8MHZ برای (ATtiny26L)
— 0MHZ تا 16MHZ برای (ATtiny26)

• **خطوط I/O و انواع بسته‌بندی**
— 16 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
— 20 پایه (PIN) در انواع PDIP و SOIC و32 پایه در نوع MLF

● ترکیب بسته‌بندی



شکل ۱-۷ ترکیب بسته‌بندیATtiny15L

فیوز بیت‌های ATtiny26

ATTINY26 دارای دو بایت فیوز بیت طبق دو جدول ۱-۸ و۱-۹ می‌باشد. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

RSTDISBL : به معنای RESET DISABLE است. با برنامه‌ریزی این بیت (0 نوشتن) می‌توان از پایه RESET خارجی (PB7) به عنوان I/O عمومی استفاده کرد. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده(1) است.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

جدول ۱-۸ بایت پر ارزش فیوز بیتهای ATtiny26

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1(UNPROGRAMMED)
-	6	-	1(UNPROGRAMMED)
-	5	-	1(UNPROGRAMMED)
RSTDISBL	4	SELECT IF PB7 IS I/O PIN OR RESET PIN	1 (UNPROGRAMMED, PB7 IS RESET PIN)
SPIEN	3	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0 (PROGRAMMED, SPI PROG.ENABLED)
EESAVE	2		
BODLEVEL	1	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	0	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)

جدول ۱-۹ بایت کم ارزش فیوز بیتهای ATtiny26

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
PLLCK	7	USE PLL FOR INTERNAL CLOCK	1(UNPROGRAMMED)
CKOPT	6	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM پاک می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ریست داخلی میکرو فعال شده و سیستم را ریست می‌کند ولی زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ریست داخلی میکرو فعال شده و میکرو را ریست می‌کند. لازم به تذکر است که این بیت به همراه بیت‌های CKSEL3..0 زمان شروع (START UP) میکرو را نیز تعیین می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN - OUT این بیت بایستی طبق جدول ۱-۱۰ برنامه‌ریزی شده باشد. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

PLLCK : این بیت برای راه‌اندازی کلاک PLL داخلی برنامه‌ریزی می‌شود و فرکانس 64MHZ کلاک PLL می‌تواند بر 4 تقسیم گردد و به عنوان کلاک سیستم استفاده شود. این موضوع در بخش کلاک سیستم ATTINY26 توضیح داده شده است.

جدول ۱-۱۰ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

CKOPT : بیتی برای انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در بخش کلاک سیستم ATTINY26 آمده است.

SUT1، SUT0 : عملکرد این دو بیت در بخش کلاک سیستم ATTINY26 کاملاً توضیح داده شده است.

CKSEL3...0 : عملکرد این بیت‌ها در بخش کلاک سیستم ATTINY26 کاملاً توضیح داده شده است.

کلاک سیستم ATtiny26

توزیع کلاک

کلاک در ATTINY26 طبق شکل ۱-۸ توزیع می‌شود.

کلاک CPU – CLK_{CPU}

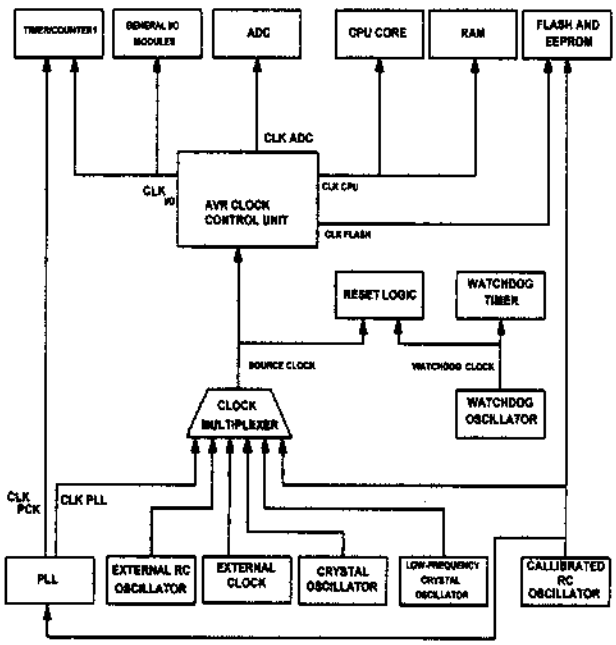
این کلاک برای انجام عملیات AVR به طور مثال رجیسترها استفاده می‌شود. متوقف شدن(HALT) این کلاک باعث انجام نگرفتن عملیات و محاسبات AVR می شود.

کلاک I/O – $CLK_{I/O}$

این کلاک توسط بسیاری از ماژول‌های I/O به طور مثال تایمرها ، کانترها و SPI استفاده می‌شود.

کلاک FLASH – CLK_{FLASH}

کلاک FLASH معمولاً با کلاک CPU فعال و عملیات ارتباطی با حافظه FLASH را کنترل می‌کند.



شکل ۱-۸ توزیع کلاک بخشهای ATTINY26 مختلف در میکرو

کلاک CLKADC – ADC

ADC از یک کلاک جداگانه حساس استفاده می‌کند که باعث می‌شود دو کلاک CPU و I/O به حالت HALT رفته تا نویز حاصل از مدار دیجیتال داخلی کاهش یافته و در نتیجه عملیات تبدیل با دقت بیشتری انجام یابد.

منابع کلاک (CLOCK SOURCE)

میکرو ATTINY26 دارای انواع منابع کلاک اختیاری است که می‌توان به وسیله فیوز بیت‌های قابل برنامه‌ریزی حافظه FLASH انواع آن را انتخاب کرد. کلاک انتخاب شده به عنوان ورودی کلاک AVR طبق جدول ۱-۱۱ در نظر گرفته شده و کلاک مناسب به هر قسمت سیستم داده می‌شود.

نکته

در تمام جداول فیوز بیت‌ها ، 0 به معنای بیت برنامه‌ریزی شده (PROGRAMMED) و 1 به معنای بیت برنامه‌ریزی نشده (UNPROGRAMMED) است.

جدول ۱-۱۱ انتخاب کلاک سیستم برای ATTINY26

DEVICE CLOCKING OPTION	CKSEL3...0	PLLCK
EXTERNAL CRYSTAL/CERAMIC RESONATOR	1111 - 1010	1
EXTERNAL – LOW FREQUENCY CRYSTAL	1001	1
EXTERNAL RC OCSILLATOR	1000 - 0101	1
CALIBRATED INTERNAL RC SCILLATOR	0100 - 0001	1
EXTERNAL CLOCK	0000	1
PLL CLOCK	0001	0

با توجه به جدول ۱-۱۲ عملکرد پایه‌های PB5 ، PB4 و همچنین کلاک سیستم کاملاً مشخص می‌شود.

جدول ۱-۱۲ انتخاب انواع کلاک سیستم و طرز کار پایه‌های PB5 ، PB4 برای ATTINY26

DEVICE CLOCKING OPTION	PLLCK	CKSEL [3:0]	PB4	PB5
EXTERNAL CLOCK	1	0000	XTAL1	I/O
INTERNAL RC OSCILLATOR	1	0001	I/O	I/O
INTERNAL RC OSCILLATOR	1	0010	I/O	I/O
INTERNAL RC OSCILLATOR	1	0011	I/O	I/O
INTERNAL RC OSCILLATOR	1	0100	I/O	I/O
EXTERNAL RC OSCILLATOR	1	0101	XTAL1	I/O
EXTERNAL RC OSCILLATOR	1	0110	XTAL1	I/O
EXTERNAL RC OSCILLATOR	1	0111	XTAL1	I/O
EXTERNAL RC OSCILLATOR	1	1000	XTAL1	I/O
EXTERNAL LOW-FREQUENCY OSCILLATOR	1	1001	XTAL1	XTAL2
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1010	XTAL1	XTAL2
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1011	XTAL1	XTAL2

DEVICE CLOCKING OPTION	PLLCK	CKSEL [3:0]	PB4	PB5
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1100	XTAL1	XTAL2
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1101	XTAL1	XTAL2
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1110	XTAL1	XTAL2
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1111	XTAL1	XTAL2
PLL	0	0001	I/O	I/O

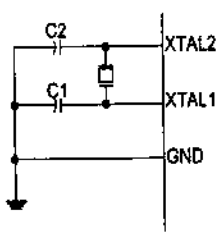
ادامه جدول صفحه قبل

هنگامیکه CPU از مُد POWER-DOWN خارج می‌شود زمانی به نام زمان شروع (START-UP) برای رسیدن کریستال به شرایط پایدار ایجاد و سپس دستورات برنامه اجرا می‌شود و زمانی که CPU از ری‌ست شروع به کار می‌کند ، تاخیری اضافه (DELAY) برای رسیدن ولتاژ به سطح پایدار ایجاد شده و سپس اجرای برنامه آغاز می‌شود. برای ایجاد زمان بندی‌های مذکور از اسیلاتور WATCHDOG استفاده می‌شود.

اسیلاتور کریستالی خارجی (EXTERNAL CRYSTAL/RESONATOR OSCILLATOR)

در این حالت کریستال یا نوسانگر سرامیکی (CERAMIC RESONATOR) یا کریستال کوارتز (QUARTZ CRYSTAL) همانطور که در شکل۱-۹ نشان داده شده است به دو پایه XTAL1 و XTAL2 وصل می‌شود.

خازنهای C1 و C2 برای کریستال‌ها و نوسانگرها بایستی یک مقدار باشند. مقادیر خازنها بستگی به کریستال ، نوسانگر و نویزهای الکترومغناطیسی محیط دارند. بعضی از خازنهای مورد استفاده برای کریستال‌های مختلف در جدول ۱-۱۳ آمده است. برای نوسانگرهای سرامیکی مقدار خازنهایی که توسط کارخانه پیشنهاد می‌گردد بایستی استفاده شود. فیوز بیت CKOPT همیشه برنامه‌ریزی شده است.



شکل ۱-۹ اتصال کریستال به میکرو در حالت اسیلاتور کریستالی

جدول ۱-۱۳ مُدهای عملیاتی اسیلاتور کریستالی

CKOPT	CKSEL 3..0	FREQUENCY RANGE (MHZ)	RECOMMENDED RANGE FOR CAPACITORS C1 AND C2 FOR USE WITH CRYSTAL (PF)
1	101[1]	0.4 – 0.9	-
1	110	0.9 – 3.0	12 – 22
1	111	3.0 – 16.0	12 – 22
1		16 ≤	12 – 15

[1] – این انتخاب برای نوسانگر سرامیکی استفاده می‌شود و نباید آن را برای کریستال به کار برد.

جدول ۱-۱۴ انتخاب زمان START-UP برای کلاک اسیلاتور کریستالی

CKSEL 0	SUT 1..0	START-UP TIME FROM POWER-DOWN	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
0	00	258 CK[1]	4.1ms	CERAMIC RESONATOR , FAST RISING POWER
0	01	258 CK[1]	65 ms	CERAMIC RESONATOR , SLOWLY RISING POWER
0	10	1K CK[2]	-	CERAMIC RESONATOR , BOD ENABLE
0	11	1K CK[2]	4.1 ms	CERAMIC RESONATOR , FAST RISING POWER
1	00	16K CK	65 ms	CERAMIC RESONATOR , SLOWLY RISING POWER
1	01	16K CK	-	CERAMIC RESONATOR , BOD ENABLE
1	10	16K CK	4.1 ms	CRYSTAL OSCILLATOR , FAST RISING POWER
1	11	16K CK	65ms	CRYSTAL OSCILLATOR , SLOWLY RISING POWER

[1] - این گزینه‌ها زمانی که سیستم در فرکانسهای بالا کار نمی‌کند استفاده می‌گردد. انتخاب این گزینه‌ها برای کریستال‌ها مناسب نیست.

[2] - این گزینه‌ها برای نوسانگرهای سرامیکی استفاده می‌شود. همچنین می‌توانند برای کریستال‌ها زمانی که در فرکانسهای پایین کار می‌کنند استفاده شوند.

توسط فیوز بیت CKSEL0 و SUT1..0 زمان آغاز (START-UP) را می‌توان طبق جدول ۱-۱۴ انتخاب کرد.

اسیلاتور کریستالی فرکانس پایین (EXTERNAL LOW -FREQUENCY OSCILLATOR)

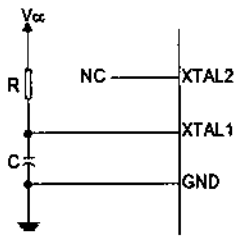
برای استفاده از کریستال ساعت 32.768KHZ ، فیوز بیتهای CKSEL با 1001 برنامه‌ریزی و کریستال طبق شکل ۱-۹ به پایه‌های XTAL1 و XTAL2 متصل می‌شود. با برنامه‌ریزی کردن CKOPT می‌توان خازنهای داخلی را فعال نمود و در نتیجه خازنهای خارجی را برداشت. مقدار نامی خازنهای داخلی 36PF است. هنگامی که این نوع کریستال انتخاب می‌شود، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۱-۱۵ قابل انتخاب است.

جدول ۱-۱۵ انتخاب زمان START-UP برای کلاک اسیلاتور کریستالی فرکانس پایین

SUT 1..0	START-UP TIME FROM POWER-DOWN	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	1K CK	4.1 ms	FAST RISING POWER OR BOD ENABLE
01	1K CK	65 ms	SLOWLY RISING POWER
10	32K CK	65 ms	STABLE FREQUENCY AT START -UP
11	RESERVED		

اسیلاتور RC خارجی (EXTERNAL RC OSCILLATOR)

اتصال RC به پایه‌های XTAL1 در شکل ۱-۱۰ آمده است. فرکانس تقریبی توسط معادله (f =1 /(3RC بدست می‌آید. مقدار خازن بایستی حداقل 22PF باشد. با برنامه‌ریزی کردن فیوز بیت CKOPT کاربر می‌تواند خازنهای داخلی36PF را بین XTAL1 وGND را راه‌اندازی کند و در نتیجه دیگر نیازی به خازن خارجی نیست.



شکل ۱-۱۰ اتصال RC به میکرو در حالت اسیلاتور RC خارجی

اسیلاتور در 4 مُد فرکانسی می‌تواند کار کند که این فرکانس‌ها طبق فیوز بیت‌های CKSEL3..0 طبق جدول ۱-۱۶ قابل انتخاب است.

هنگامی که فرکانس کاری انتخاب می‌شود، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۱-۱۷ قابل انتخاب است.

جدول ۱-۱۶ مُدهای عملیاتی اسیلاتور RC خارجی

CKSEL 3..0	FREQUENCY RANGE (MHZ)
0101	≤ 0.9
0110	0.9 – 3.0
0111	3.0 – 8.0
1000	8.0 – 12.0

جدول ۱-۱۷ انتخاب زمان START-UP برای کلاک اسیلاتور RC خارجی

SUT1..0	START-UP TIME FROM POWER-DOWN	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	18 CK	-	BOD ENABLE
01	18 CK	4.1ms	FAST RISING POWER
10	18 CK	65 ms	SLOWLY RISING POWER
11	6 CK[1]	4.1ms	FAST RISING POWER OR BOD ENABLE

[1]- این گزینه زمانی که میکرو در فرکانسهای بالا کار می‌کند نباید انتخاب شود.

اسیلاتور RC کالیبره شده داخلی(INTERNAL RC OSCILLATOR)

اسیلاتور RC کالیبره شده داخلی، کلاک‌های نامی داخلی 1 ، 2، 4 و 8MHZ را طبق جدول ۱-۱۸ در ولتاژ 5V و دمای 25^0C تولید می‌کند. با برنامه‌ریزی شدن بیت‌های CKSEL این کلاک می‌تواند به عنوان کلاک سیستم استفاده شود که در این حالت نیازی به مدار خارجی نیست. زمانی که از این مُد استفاده می‌شود فیوز بیت CKOPT همیشه بایستی برنامه‌ریزی شده باشد.

نکته حالت پیش‌فرض میکرو اسیلاتور RC داخلی 1MHZ است.

هنگامی که فرکانس کاری انتخاب می‌شود ، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۱-۱۹ قابل انتخاب است.

جدول ۱-۱۸ مُدهای عملیاتی اسیلاتور RC کالیبره شده داخلی

CKSEL 3..0	NOMINAL RANGE (MHZ)
0001(1)	1.0
0010	2.0
0011	4.0
0100	8.0

(1)- برای میکرو به صورت پیش‌فرض این گزینه انتخاب شده است.

جدول ۱-۱۹ انتخاب زمان START-UP برای کلاک اسیلاتور RC کالیبره شده داخلی

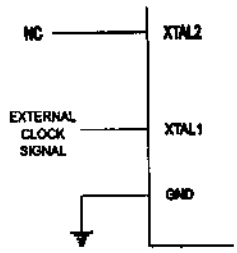
SUT 1..0	START-UP TIME FROM POWER-DOWN	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10(1)	6 CK	65 ms	SLOWLY RISING POWER
11	RESERVED		

(1)- برای میکرو به صورت پیش‌فرض این گزینه انتخاب شده است.

کلاک خارجی (EXTERNAL CLOCK)

برای راه‌اندازی میکرو در این مُد، کلاک خارجی به پایه XTAL1 بایستی طبق شکل ۱-۱۱ وصل شود. برای کار در این مُد بیت‌های CKSEL با 0000 برنامه‌ریزی می‌شوند. با برنامه‌ریزی کردن فیوز بیت CKOPT خازن داخلی 36PF بین پایه‌های XTAL1 و GND فعال می‌شود.

هنگامی که این نوع کلاک انتخاب می‌شود، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۱-۲۰ قابل انتخاب است.



شکل ۱-۱۱ اتصال کلاک خارجی به پایه میکرو در حالت کلاک خارجی

در این مُد باید از تغییرات ناگهانی فرکانس کلاک خارجی برای اطمینان از انجام پایدار و صحیح عملیات میکروکنترلر (MCU) جلوگیری کرد. تغییرات بیشتر از 2% در فرکانس کلاک خارجی ممکن است باعث رفتارهای غیرقابل انتظار میکرو شود. زمانی که قصد تغییر فرکانس کلاک را دارید بایستی میکرو در حالت RESET نگه داشته شود.

کلاک فرکانس بالای PLL

مدار PLL داخلی در این مُد، فرکانس 64MHZ را تولید می‌کند که می‌تواند به عنوان کلاک تایمرکانتر1 و تقسیمی از آن به عنوان کلاک سیستم استفاده شود. در صورت برنامه‌ریزی فیوزبیت PLLCK و فیوز بیت‌های CKSEL فرکانس PLL بر 4 تقسیم شده و به عنوان کلاک سیستم مورد استفاده قرار می‌گیرد. در این مُد بایستی ولتاژ منبع تغذیه بین 4.5V تا 5.5V باشد. در این حالت زمان START-UP طبق جدول ۱-۲۱ تنظیم می‌شود.

جدول ۱-۲۰ انتخاب زمان START-UP برای کلاک خارجی

SUT 1..0	START-UP TIME FROM POWER-DOWN	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10	6 CK	65 ms	SLOWLY RISING POWER
11	RESERVED		

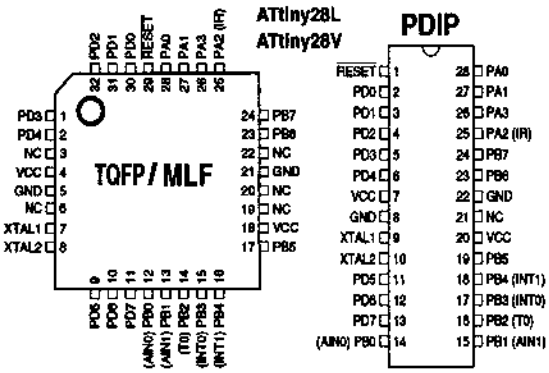
جدول ۱-۲۱ انتخاب زمان START-UP برای کلاک PLL داخلی

SUT 1..0	START-UP TIME FROM POWER-DOWN	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10	6 CK	65 ms	SLOWLY RISING POWER
11	16K CK	-	SLOWLY RISING POWER

۱-۴ خصوصیات ATtiny28V و ATtiny28L

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 90 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 4MIPS در فرکانس 4MHZ
- **حافظه ، برنامه و دادهٔ غیر فرار**
  - 2K بایت حافظه FLASH قابل برنامه‌ریزی داخلی
  - قفل برنامهٔ FLASH
- **خصوصیات جانبی**
  - یک تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا
  - یک مقایسه‌گر آنالوگ داخلی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
- **خصوصیات ویژه میکروکنترلر**
  - تغذیه کم در مُدهای IDLE و POWERDOWN
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - مدار POWER - ON RESET داخلی
  - اسیلاتور RC داخلی برای کاهش قسمتهای خارجی
  - توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS
- **توان مصرفی در 1MHZ ، 2V ، 25°C**
  - حالت فعال 3mA (ACTIVE MODE)
  - درحالت بی‌کاری 1.2mA (IDLE MODE)
  - در حالت POWER - DOWN : < 1μA
- **ولتاژهای عملیاتی (کاری)**
  - 1.8 V تا 5.5V برای (ATtiny28V)
  - 2.7V تا 5.5V برای (ATtiny28L)
- **فرکانسهای کاری**
  - 0MHZ تا 1.2MHZ برای (ATtiny28V)
  - 0MHZ تا 4MHZ برای (ATtiny28L)

- **خطوط I/O و انواع بسته‌بندی**
  - 11 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 28 پایه (PIN) نوع PDIP ، 32 پایه نوع TQFP و 32 پایه نوع MLF
- **ترکیب پایه‌ها**



شکل ۱-۱۲ ترکیب بسته بندی ATtiny28

فیوز بیت‌های ATtiny28

ATTINY28 دارای 5 فیوزبیت است. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

INTCAP : در صورت برنامه‌ریزی این بیت خازن‌های داخلی بین پایه‌های XTAL1 و XTAL2 متصل می‌شود و در صورت استفاده از کریستال خارجی نیازی به نصب این خازنها در خارج نیست. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

CKSEL3..0 : برای تعیین کلاک سیستم و زمان شروع (START UP) از این بیت‌ها طبق جدول ۱-۲۲ استفاده می‌شود. این بیت‌ها به صورت پیش‌فرض 0010 هستند و میکرو با اسیلاتور RC داخلی با مقدار نامی 1.2MHZ و زمان Start-up طولانی کار می‌کند.

منابع کلاک ATTINY28

اسیلاتور کریستالی خارجی(EXTERNAL CRYSTAL/CERAMIC RESONATOR)

در این حالت کریستال یا نوسانگر سرامیکی (CERAMIC RESONATOR) یا کریستال کوارتز (QUARTZ CRYSTAL) به دو پایه XTAL1 و XTAL2 وصل می‌شود. خازنهای C1 وC2 برای کریستال‌ها و نوسانگرها بایستی یک مقدار باشند. مقادیر خازنها بستگی به کریستال، نوسانگر و نویزهای الکترومغناطیسی محیط دارند که مقدار نامی32PF مناسب است.

جدول ۱-۲۲ انتخاب انواع کلاک سیستم و زمان START-UP برای ATTINY28

CKSEL3..0	Clock Source	Start-up Time, VCC=2.7V, BODLEVEL Programmed
1111	Ext. Crystal/Ceramic Resonator	1K CK
1110	Ext. Crystal/Ceramic Resonator	4.2 ms + 1K CK
1101	Ext. Crystal/Ceramic Resonator	67 ms + 1K CK
1100	Ext. Crystal/Ceramic Resonator	16K CK
1011	Ext. Crystal/Ceramic Resonator	4.2 ms + 16K CK
1010	Ext. Crystal/Ceramic Resonator	67 ms + 16K CK
1001	Ext. Low-frequency Crystal	67 ms + 1K CK
1000	Ext. Low-frequency Crystal	67 ms + 32K CK
0111	Ext. RC Oscillator	6 CK
0110	Ext. RC Oscillator	4.2 ms + 6 CK
0101	Ext. RC Oscillator	67 ms + 6 CK
0100	Int. RC Oscillator	6 CK
0011	Int. RC Oscillator	4.2 ms + 6 CK
0010	Int. RC Oscillator	67 ms + 6 CK
0001	Ext. Clock	6 CK
0000	Ext. Clock	4.2 ms + 6 CK

اسیلاتور کریستالی فرکانس پایین خارجی (EXTERNAL LOW FREQ CRYSTAL)

برای استفاده از کریستال ساعت 32.768KHZ ، کریستال به پایه‌های XTAL1 و XTAL2 متصل می‌شود.

اسیلاتور RC خارجی(EXT. RC OSCILLATOR)

نحوه اتصالRC به پایه‌های XTAL1 در شکل ۱- ۱۳ کشیده شده است. مقدار خازن بایستی حداقل 20PF باشد و مقاومت باید در محدوده 100K – 3K باشد. خازن و مقاومت سه فرکانس در جدول۱-۲۳ آمده است.

اسیلاتور RC کالیبره شده داخلی (INT. RC OSCILLATOR)

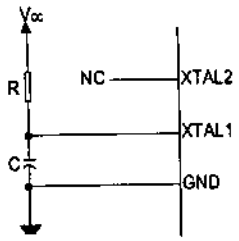
اسیلاتور RC کالیبره شده داخلی، برای ATTINY11 برابر 1MHZ (پیش فرض میکرو) و برای ATTINY12 برابر 1.2MHZ (پیش فرض میکرو)است.

کلاک خارجی (EXT. CLOCK)

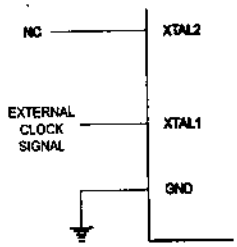
برای راه‌اندازی میکرو در این مُد، کلاک خارجی به پایه XTAL1 طبق شکل ۱-۱۴ بایستی وصل شود.

جدول ۱-۲۳ مُدهای کاری اسیلاتور RC خارجی

R [KΩ]	C [pF]	FREQUENCY
100	70	100KHZ
31.5	20	1.0MHZ
6.5	20	4.0MHZ



شکل ۱-۱۳ اتصال RC به میکرو در حالت اسیلاتور RC خارجی



شکل ۱-۱۴ اتصال کلاک خارجی به پایه میکرو در حالت کلاک خارجی

۱-۵ خصوصیات ATtiny13L , ATtiny13

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 120 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - عملکرد کاملاً ثابت
  - سرعتی تا 20MIPS در فرکانس 20MHZ
- **حافظه ، برنامه و دادهٔ غیر فرّار**
  - 1K بایت حافظه FLASH قابل برنامه‌ریزی داخلی

    پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 64 بایت حافظه SRAM
  - 64 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - یک تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER و دو کانال PWM
  - 4 کانال مبدل آنالوگ به دیجیتال 10 بیتی

— یک مقایسه‌گر آنالوگ داخلی
— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

• **خصوصیات ویژه میکروکنترلر**
— تغذیه کم در مُدهای IDLE و POWERDOWN
— دارای مُد کاهش نویز (NOISE REDUCTION)
— قابلیت ON-CHIP DEBUG
— منابع وقفه (INTERRUPT) داخلی و خارجی
— ارتباط سریال SPI برای برنامه‌ریزی در داخل مدار (IN – SYSTEM PROGRAMMING)
— مدار POWER – ON RESET CIRCUIT
— مدار BROWN –OUT DETECTION CIRCUIT
— اسیلاتور داخلی برای کاهش قسمتهای خارجی
— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

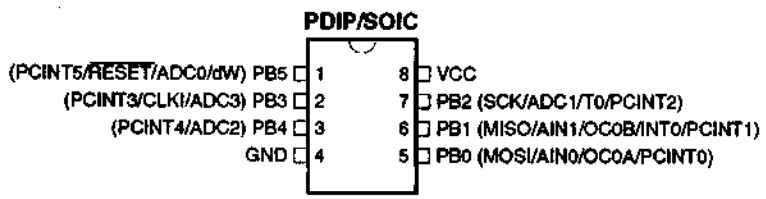
• **توان مصرفی کم**
— حالت فعال
در 1MHZ و ولتاژ 1.8V : 20μA
— حالت POWER-DOWN
در ولتاژ 1.8V < 1μA

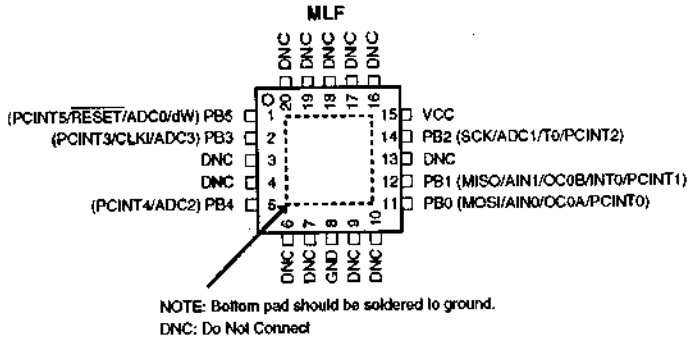
• **ولتاژهای عملیاتی (کاری)**
— 1.8V تا 5.5V برای (ATtiny13V)
— 2.7V تا 5.5V برای (ATtiny13)

• **فرکانسهای کاری**
— 0MHZ تا 4MHZ برای (ATtiny13V)
— 0MHZ تا 10MHZ برای (ATtiny13)

• **خطوط I/O و انواع بسته‌بندی**
— 6 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
— 8 پایه (PIN) در انواع PDIP و SOIC و 20 پایه در نوع MLF

•**ترکیب پایه ها**





شکل ۱-۱۵ بسته بندی ATTINY13

فیوز بیت‌های ATTINY13

ATTINY13 دارای دو بایت فیوز بیت طبق جدول‌های ۱-۲۴ و ۱-۲۵ می‌باشد .

جدول ۱-۲۴ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1(UNPROGRAMMED)
-	6	-	1(UNPROGRAMMED)
-	5	-	1(UNPROGRAMMED)
SELFPRGEN	4	SELF PROGRAMMING ENABLE	1(UNPROGRAMMED)
DWEN	3	DEBUG WIRE ENABLE	1(UNPROGRAMMED)
BODLEVEL1	2	BROWN-OUT DETECTOR TRRIGER LEVEL	1(UNPROGRAMMED)
BODLEVEL0	1	BROWN-OUT DETECTOR TRRIGER LEVEL	1(UNPROGRAMMED)
RSTDISBL	0	EXTERNAL RESET DESABLE	1(UNPROGRAMMED)

جدول ۱-۲۵ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
SPIEN	7	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	1(UNPROGRAMMED)
EESAVE	6	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED EEPROM NOT PRESERVED
WDTON	5	WDT TIMER ALWAYS ON	1(UNPROGRAMMED)
CKDIV8	4	DEVIDE CLOCK BY 8	0(PROGRAMMED)
SUT1	3	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	2	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

SELFPRGEN : با برنامه‌ریزی این بیت می‌توان از دستور SPM در برنامه استفاده نمود.

DWEN :در صورتی که این بیت برنامه‌ریزی نشده باشد، می‌توان حافظه برنامه را حتی اگر بیتهای قفل برنامه‌ریزی شده باشند توسط ارتباط DEBUGWIRE خواند. بنابرین در صورتی که بخواهید برنامه به هیچ وجهی خوانده نشود این بیت را برنامه‌ریزی کنید.

BODLEVEL1, BODLEVEL0 : عملکرد BROWN-OUT DETECTION طبق منطق این دو بیت در جدول ۱-۲۶ مشخص می‌شود .

RSTDISBL : در حالت پیش‌فرض PB5 پایه ری‌ست است. با برنامه‌ریزی این بیت، پایه PB5 به عنوان پایه I/O استفاده می‌شود.

SPINE : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM پاک می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

جدول ۱-۲۶ انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

BODLEVEL FUSES 1, 0	VBOT(VBOT- ,VBOT+)
11	BOD DISABLE
10	1.8v ± 25mv
01	2.7v ± 25mv
00	4.3v ± 25mv

WDTON : در حالت پیش‌فرض WATCHDOG غیرفعال و کاربر بایستی نرم‌افزاری WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن و کاربر بایستی نرم افزاری WATCHDOG را ری ست کند.

CKDIV8 : توسط این بیت می‌توان کلاک سیستم راجهت کاهش مصرف بر 8 تقسیم نمود.در این حالت تمام کلاک ها از جمله CLKCPU,CLKI/O,CLKFLASH و CLKADC بر 8 تقسیم می‌شوند.

SUT0,1 : زمان STARTUP با توجه به منطق این دو بیت توسط جدول ۱-۲۷ در نظر گرفته می‌شود.

جدول ۱-۲۷ انتخاب زمان START-UP برای کلاک اسیلاتور RC کالیبره شده داخلی

SUT1..0	START-UP TIME FROM POWER-DOWN	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	14 CK	BOD ENABLE
01	6 CK	14CK+4 ms	FAST RISING POWER
10[1]	6 CK	14CK+64 ms	SLOWLY RISING POWER
11	RESERVED		

[1]- پیش فرض میکرو

CLSEL1,0 : این دو بیت کلاک سیستم را با توجه به جدول ۱-۲۸ مشخص می‌کنند.

جدول ۱-۲۸ مُدهای عملیاتی کلاک

CKSEL1,0	FREQUENCY RANGE (MHZ)
10[1]	9.6
01	4.8
00	EXTERNAL CLOCK
11	128KHZ INTERNAL OSCILLATOR

[1]-پیش فرض میکرو

۱-۶ خصوصیات ATtiny22L , ATtiny22

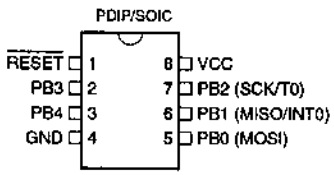
- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 118 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 1MIPS در فرکانس 1MHZ
- **حافظه ، برنامه و دادهٔ غیر فرّار**
  - 2K بایت حافظه FLASH قابل برنامه‌ریزی داخلی

    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن
  - 128 بایت حافظه SRAM
  - 128 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

    پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - یک تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - ارتباط سریال SPI برای برنامه‌ریزی در داخل مدار (IN - SYSTEM PROGRAMMING)
- **خصوصیات ویژه میکروکنترلر**
  - تغذیه کم در مُدهای IDLE و POWERDOWN
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - مدار POWER - ON RESET CIRCUIT
  - اسیلاتور داخلی برای کاهش قسمتهای خارجی

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS
— عملکرد کاملاً ثابت

- **توان مصرفی در 3V ، 25^0C**
  — حالت فعال 1.5mA (ACTIVE MODE)
  — درحالت بی‌کاری 100µA (IDLE MODE)
  — در حالت POWER – DOWN : < 1µA
- **ولتاژهای عملیاتی (کاری)**
  — 2.7V تا 6v
- **فرکانسهای کاری**
  — اسیلاتور داخلی 1MHZ
- **خطوط I/O و انواع بسته‌بندی**
  — 5 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  — 8 پایه (PIN) در انواع PDIP و SOIC
- **ترکیب پایه ها**



فیوز بیت‌های ATTINY22

این میکرو دارای یک فیوزبیت **SPIEN** می باشد.
SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود. این بیت در ولتاژ پایین مُد برنامه‌ریزی سریال قابل دسترس نمی‌باشد.

۱-۷ خصوصیات ATtiny2313V , ATtiny2313

- **از معماری AVR RISC استفاده می‌کند.**
  — کارایی بالا و توان مصرفی کم
  — دارای 120 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  — 32*8 رجیستر کاربردی
  — عملکرد کاملاً ثابت
  — سرعتی تا 20MIPS در فرکانس 20MHZ

- **حافظه ، برنامه و دادهٔ غیر فرار**

— 2K بایت حافظه FLASH قابل برنامه‌ریزی داخلی

پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن(WRITE / ERASE)

— 128 بایت حافظه SRAM

— 128 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)

— قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

- **خصوصیات جانبی**

— یک تایمر – کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا

— یک تایمر – کانتر 16 بیتی پرسرعت (HIGH-SPEED) با PRESCALER مجزا

— 4 کانال PWM

— USART دو طرفه

—یک مقایسه‌گر آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

- **خصوصیات ویژه میکروکنترلر**

— قابلیت ON-CHIP DEBUG

— تغذیه کم در مُدهای IDLE و POWERDOWN و STANDBY

— منابع وقفه (INTERRUPT) داخلی و خارجی

— ارتباط سریال SPI برای برنامه‌ریزی در داخل مدار (IN – SYSTEM PROGRAMMING)

— قابلیت ارتباط سریال USI (UNIVERSAL SERIAL INTERFACE)

— مدار POWER – ON RESET CIRCUIT

— مدار BROWN –OUT DETECTION CIRCUIT

— اسیلاتور داخلی برای کاهش قسمتهای خارجی

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

- **توان مصرفی در 3V ، 25^0C**

— حالت فعال

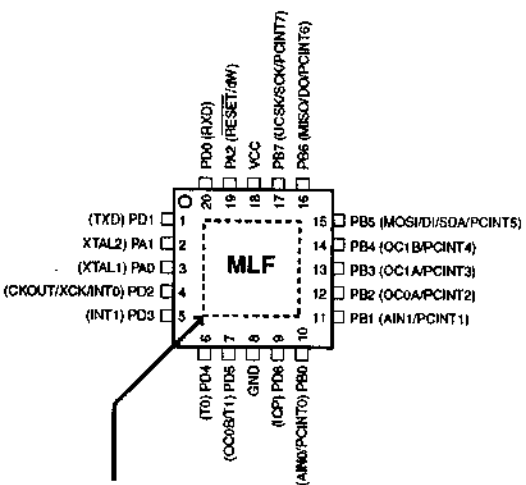
در 1MHZ و ولتاژ 1.8V : 230UA

در 32KHZ و ولتاژ 1.8V : 20UA

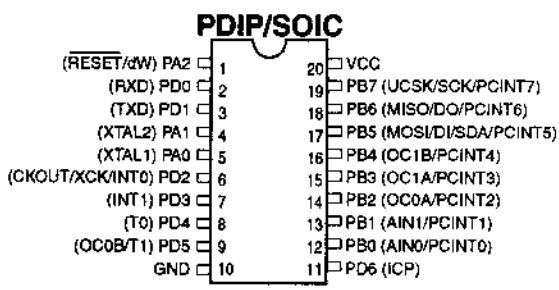
— حالت POWER-DOWN

در ولتاژ 1.8V > 0.1UA

- **ولتاژهای عملیاتی (کاری)**
  - 1.8V تا 5.5V برای (ATtiny2313V)
  - 2.7V تا 5.5V برای (ATtiny2313)
- **فرکانسهای کاری**
  - 0MHZ تا 4MHZ برای (ATtiny2313V)
  - 0MHZ تا 10MHZ برای (ATtiny2313)
- **خطوط I/O و انواع بسته‌بندی**
  - 18 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 20 پایه (PIN) در انواع PDIP و SOIC و 20 پایه MLF
- **ترکیب پایه ها**



NOTE: Bottom pad should be soldered to ground.
DNC: Do Not Connect



فیوز بیت‌های ATTINY2313

ATTINY2313 دارای سه بایت فیوز بیت طبق جدول‌های ۱-۲۹، ۱-۳۰ و ۱-۳۱ می‌باشد.

جدول ۱-۲۹ بایت پیشرفته فیوزبیتهای میکرو

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1(UNPROGRAMMED)
-	6	-	1(UNPROGRAMMED)
-	5	-	1(UNPROGRAMMED)
-	4	-	1(UNPROGRAMMED)
-	3	-	1(UNPROGRAMMED)
-	2	-	1(UNPROGRAMMED)
-	1	-	1(UNPROGRAMMED)
SELFPRGEN	0	SELF PROGRAMMING ENABLE	1(UNPROGRAMMED)

جدول ۱-۳۰ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
DWEN	7	DEBUGWIRE ENABLE	1(UNPROGRAMMED)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
EESAVE	6	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
WDTON	4	WATHDOG TIMER ALWAYS ON	1(UNPROGRAMMED)
BODLEVEL2	3	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL1	2	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL0	1	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
RSTDISBL	0	EXTERNAL RESET DESABLE	1(UNPROGRAMMED)

جدول ۱-۳۱ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
CKDIV8	7	DIVIDE CLICK BY 8	0(PROGRAMMED)
CKOUT	6	OUTPUT CLOCK ON CKOUT PIN	1(UNPROGRAMMED)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

SELFPRGEN : با برنامه‌ریزی این بیت می‌توان از دستور SPM در برنامه استفاده نمود.

DWEN :در صورتی که این بیت برنامه‌ریزی نشده باشد، می‌توان حافظه برنامه را حتی اگر بیت‌های قفل برنامه‌ریزی شده باشند توسط ارتباط DEBUGWIRE خواند. بنابرین در صورتی که بخواهید برنامه به هیچ وجهی خوانده نشود این بیت را برنامه‌ریزی کنید.

SPINE : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو، حافظه EEPROM پاک می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

WDTON : در حالت پیش‌فرض WATCHDOG غیرفعال و کاربر بایستی نرم‌افزاری WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن و کاربر بایستی نرم افزاری WATCHDOG را ری ست کند.

BODLEVEL2.. BODLEVEL0 : عملکرد BROWN-OUT DETECTION طبق منطق این دو بیت در جدول ۱-۳۲ مشخص می‌شود .

RSTDISBL : در حالت پیش‌فرض PA2 پایه ری‌ست است. با برنامه‌ریزی این بیت، پایه PA2 به عنوان پایه I/O استفاده می‌شود.

CKDIV8 : توسط این بیت می‌توان کلاک سیستم راجهت کاهش مصرف بر 8 تقسیم نمود. در این حالت تمام کلاک ها از جمله CLKCPU,CLKI/O,CLKFLASH و CLKADC بر 8 تقسیم می‌شوند.

جدول ۱-۳۲ انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

BODLEVEL FUSES 2 , 1 , 0	VBOT(VBOT- ,VBOT+)
111	BOD DISABLE
110	1.8v ± 25mv
101	2.7v ± 25mv
100	4.3v ± 25mv
011	RESERVED
010	
001	
000	

CKOUT : با برنامه‌ریزی این بیت، کلاک سیستم از پایه PD2 خارج می‌شود.
SUT0,1 : زمان STARTUP با توجه به منطق این دو بیت در نظر گرفته می‌شود.
CLSEL3,0 : این دو بیت کلاک سیستم را با توجه به جدول ۱-۳۳ مشخص می‌کنند.

جدول ۱-۳۳ انتخاب کلاک سیستم

CKSEL3..0	DEVICE CLOCKING OPTION
1111	EXTERNAL CRYSTAL/CERAMIC RESONATOR 8.0MHZ<
1110	EXTERNAL CRYSTAL/CERAMIC RESONATOR 3.0 - 8.0 MHZ(C= 12 – 22PF)
1101	EXTERNAL CRYSTAL/CERAMIC RESONATOR 0.9 - 3.0 MHZ(C= 12 – 22PF)
1100	EXTERNAL CRYSTAL/CERAMIC RESONATOR 0.4 - 0.9 MHZ(C= 12 – 22PF)
0110	WATCHDOG OSCILLATOR 128KHZ
0100-0101	INT. RC OSCILLATOR 8MHZ
0010-0011	INT. RC OSCILLATOR 4MHZ
0000	EXT. CLOCK
0001/0011/0101/0111	RESERVED

۲

میکروکنترلرهای AVR

دراین فصل به معرفی میکروکنترلرهای نوع AT90S از سری میکروکنترلرهای AVR شرکت ATMEL می‌پردازیم. خصوصیات و قابلیت‌های هر یک از میکروهای نوع AT90S تشریح و در ادامه فیوز بیت‌های هر یک به طور کامل بررسی شده‌اند. فیوز بیت‌ها قسمتی از حافظه FLASH هستند که امکاناتی را در اختیار کاربر قرار می‌دهند. فیوز بیت‌ها با ERASE میکرو از بین نمی‌روند و می‌توانند توسط بیت‌های قفل مربوطه، قفل شوند. کلاک سیستم هر یک از میکروها در صورت نیاز به توضیح بیشتر بلافاصله بعد از فیوز بیت‌ها گفته شده است.

اهداف

۱. آشنایی کامل با انواع میکروهای AT90S
۲. آشنایی کامل با فیوز بیت‌های هر یک از میکروها
۳. توانایی برنامه‌ریزی فیوز بیت‌های هر یک از میکروها
٤. توانایی برنامه‌ریزی فیوز بیت‌ها برای تعیین کلاک سیستم دلخواه

۲-۱ خصوصیات AT90S1200

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 89 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 12MIPS در فرکانس 12MHZ

- **حافظه ، برنامه و دادهٔ غیر فرّار**
  - 1K بایت حافظه FLASH قابل برنامه‌ریزی داخلی
    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - 64 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

- **خصوصیات جانبی**
  - یک تایمر – کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا
  - یک مقایسه‌کننده آنالوگ داخلی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN – SYSTEM PROGRAMMING)

- **خصوصیات ویژه میکروکنترلر**
  - تغذیه کم در مُدهای IDLE و POWERDOWN
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - قابل انتخاب بودن اسیلاتور RC داخلی برای کاهش قسمتهای خارجی
  - عملکرد کاملاً ثابت
  - توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

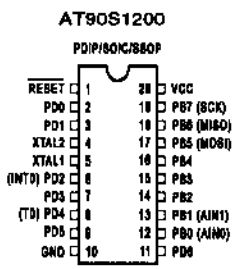
- **توان مصرفی در 4MHZ ، 3V ، 25^0C**
  - حالت فعال 2.0mA (ACTIVE MODE)
  - درحالت بی‌کاری 0.4mA (IDLE MODE)
  - در حالت 1μA > : POWER – DOWN

- **ولتاژهای عملیاتی (کاری)**
  - 2.7V تا 6V برای (AT90S1200-4)
  - 4V تا 6V برای (AT90S1200-12)

- **فرکانسهای کاری**
  - 0MHZ تا 4MHZ برای (AT90S1200-4)
  - 0MHZ تا 12MHZ برای (AT90S1200-12)

- **خطوط I/O و انواع بسته‌بندی**
  - 15 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 20 پایه (PIN) در انواع PDIP ، SOIC و SSOP

- **ترکیب بسته‌بندی**



فیوز بیت‌های AT90S1200

این میکرو دارای 2 فیوز بیت (SPIEN , RCEN) است. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

نکته فیوز بیت‌های AT90S1200 در زمان برنامه‌ریزی به صورت سریال قابل دسترس نمی‌باشند.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

RCEN : با برنامه‌ریزی کردن این بیت اسیلاتور RC داخلی 1MHZ فعال می‌شود و دیگر نیازی به کریستال خارجی نیست. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

۲-۲ خصوصیات AT90S2313

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 118 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 10MIPS در فرکانس 10MHZ

- **حافظه ، برنامه و دادهٔ غیرفرار**
  - 2K بایت حافظه FLASH قابل برنامه‌ریزی داخلی

پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن(WRITE / ERASE)

— 128 بایت حافظه SRAM

— 128 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)

— قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

• خصوصیات جانبی

— یک تایمر / کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا

— یک تایمر / کانتر (TIMER / COUNTER) 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE ، CAPTURE و PWM ، 8 ، 9 یا 10 بیتی

— یک مقایسه‌کننده آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN – SYSTEM PROGRAMMING)

— UART دوطرفه (FULL DUPLEX)

• خصوصیات ویژه میکروکنترلر

— تغذیه کم در مُدهای IDLE وPOWERDOWN

— منابع وقفه (INTERRUPT) داخلی و خارجی

— عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• توان مصرفی در 4MHZ ، 3V ، 25^0C

— حالت فعال 2.8mA (ACTIVE)

— درحالت بی‌کاری 0.8mA (IDLE)

— در حالت POWER – DOWN : < 1μA

• ولتاژهای عملیاتی (کاری)

— 2.7V تا 6V برای (AT90S2313-4)

— 4V تا 6V برای (AT90S2313-10)

• فرکانسهای کاری

— 0MHZ تا 4MHZ برای (AT90S2313-4)

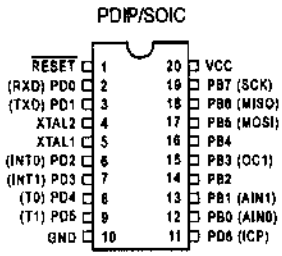
— 0MHZ تا 12MHZ برای (AT90S2313-10)

• خطوط I/O و انواع بسته‌بندی

— 15 خط ورودی / خروجی (I / O) قابل برنامه‌ریزی

— 20 پایه (PIN) درانواع PDIP ، SOIC

• **ترکیب پایه‌ها**



شکل ۲-۱ ترکیب بسته بندی

فیوز بیت‌های AT90S2313

این میکرو دارای 2 فیوز بیت (FSTRT , SPIEN) است. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

نکته فیوز بیت‌های AT90S2313 در زمان برنامه‌ریزی به صورت سریال قابل دسترس نمی‌باشند.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

FSTRT : با برنامه‌ریزی کردن این بیت کوتاه‌ترین زمان شروع (START UP) از ری‌ست و مُدهای SLEEP در نظر گرفته می‌شود. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده و طولانی‌ترین زمان در نظر گرفته شده است. جدول ۲-۱ مشخص کننده این زمان است.

جدول ۲-۱ RESET CHARACTERISTICS برای AT90S2313

VCC	PARAMETER	Min	Typ	Max	Units
VCC = 5.0V	RESET DELAY TIME-OUT PERIOD AT90S2313 FSTRT PROGRAMMED	0.25	0.28	0.31	ms
	RESET DELAY TIME-OUT PERIOD AT90S2313 FSTRT UNPROGRAMMED	11.0	16.0	21.0	ms

۲-۳ خصوصیات AT90S2323/LS2323/S2343/LS2343

• **از معماری AVR RISC استفاده می‌کند.**

— کارایی بالا و توان مصرفی کم

— دارای 118 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.

— 32*8 رجیستر کاربردی

— سرعت 10MIPS در فرکانس 10MHZ

• **حافظه ، برنامه و دادهٔ غیر فرّار**

— 2K بایت حافظه FLASH قابل برنامه‌ریزی داخلی

پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن

— 128 بایت حافظه SRAM

— 128 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن

— قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

• **خصوصیات دیگر**

— یک تایمر – کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN – SYSTEM PROGRAMMING)

• **خصوصیات ویژه میکروکنترلر**

— تغذیه کم در مُدهای IDLE و POWERDOWN

— منابع وقفه (INTERRUPT) داخلی و خارجی

— مدار POWER – ON RESET CIRCUIT

— قابل انتخاب بودن اسیلاتور RC داخلی برای کاهش قسمتهای خارجی

— عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• **توان مصرفی در 25^0C ، 3V ، 4MHZ**

— حالت فعال 2.4mA (ACTIVE)

— درحالت بی‌کاری 0.5mA (IDLE)

— در حالت POWER – DOWN : < 1µA

• **ولتاژهای عملیاتی (کاری)**

— 4V تا 6V برای (AT90S2323/AT90LS2343)

— 2.7V تا 6V برای (AT90LS2323/AT90LS2343)

• **فرکانسهای کاری**

— 0MHZ تا 10MHZ (AT90S2323/AT90S2343-10)

— 0MHZ تا 4MHZ (AT90LS2323/AT90LS2343-4)

— 0MHZ تا 1MHZ (AT90LS2343-1)

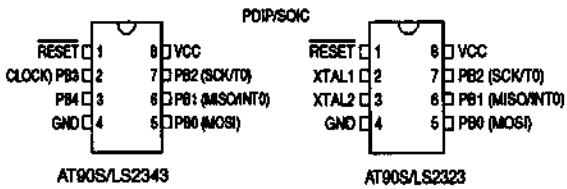
• **خطوط I/O و انواع بسته‌بندی**

— 3 خط ورودی / خروجی (I / O) قابل برنامه‌ریزی برای (AT90S/LS2323)

— 5 خط ورودی / خروجی (I / O) قابل برنامه‌ریزی برای (AT90S/LS2343)

— 8 پایه (PIN) در انواع PDIP و SOIC

- ترکیب پایه‌ها



شکل ۲-۲ ترکیب بسته بندی

فیوز بیت‌های AT90S/LS2323

این میکرو دارای 2 فیوز بیت (SPIEN , FSTRT) است. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

FSTRT : با برنامه‌ریزی کردن این بیت کوتاه‌ترین زمان شروع (START UP) از ری‌ست در نظر گرفته می‌شود. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده و طولانی‌ترین زمان در نظر گرفته شده است. جدول ۲-۲ مشخص کننده این زمان است.

جدول ۲-۲ Reset Characteristics برای AT90S/LS2323

VCC	PARAMETER	Min	Typ	Max	Units
VCC = 5.0V	RESET DELAY TIME-OUT PERIOD AT90S/LS2323 FSTRT PROGRAMMED	1.0	1.1	1.2	ms
	RESET DELAY TIME-OUT PERIOD AT90S/LS2323 FSTRT UNPROGRAMMED	11.0	16.0	21.0	ms
VCC = 3.0V	RESET DELAY TIME-OUT PERIOD AT90S/LS2323 FSTRT PROGRAMMED	2.0	2.2	2.4	ms
	RESET DELAY TIME-OUT PERIOD AT90S/LS2323 FSTRT UNPROGRAMMED	22.0	32.0	42.0	ms

فیوز بیت‌های AT90S/LS2343

این میکرو دارای 2 فیوز بیت (SPIEN,RCEN) است. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

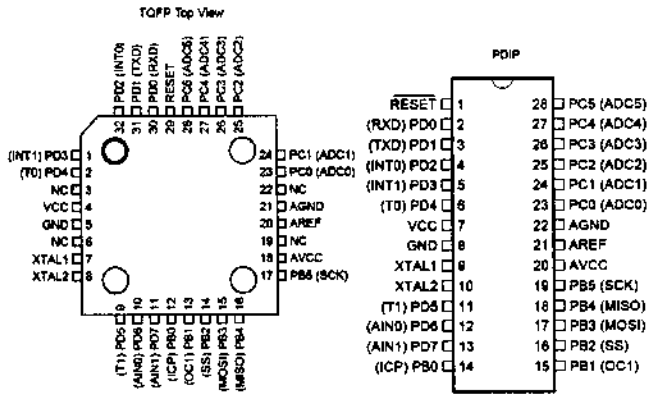
SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

RCEN : با برنامه‌ریزی کردن این بیت اسیلاتور RC داخلی 1MHZ فعال می‌شود و دیگر نیازی به کریستال خارجی نیست. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

۲-۴ خصوصیات AT90S2333/LS2333/S4433/LS4433

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 118 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعت 8MIPS در فرکانس 8MHZ
- **حافظه ، برنامه و دادهٔ غیر فرّار**
  - 2K/4K بایت حافظه FLASH قابل برنامه‌ریزی داخلی به ترتیب برای 2333/4433
    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 128 بایت حافظه SRAM
  - 128 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - قفل برنامهٔ SH /FL و حفاظت دادهٔ EEPROM
- **خصوصیات دیگر**
  - یک تایمر/ کانتر (TIMER/COUNTER) 8 بیتی با PRESCALER مجزا
  - یک تایمر/ کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE ، CAPTURE و PWM ، 8 ، 9 یا 10 بیتی
  - یک مقایسه‌کننده آنالوگ داخلی
  - WATCHDOG قابل برنامه‌ریزی با یک اسیلاتور داخلی
  - UART دوطرفه (FULL DUPLEX)
  - 6 کانال مبدل آنالوگ به دیجیتال 10 بیتی
  - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN – SYSTEM PROGRAMMING)
- **خصوصیات ویژه میکروکنترلر**
  - دارای مدار BROWN – OUT RESET
  - مدار POWER – ON RESET CIRCUIT
  - تغذیه کم درمُدهای IDLE وPOWERDOWN
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - عملکرد کاملاً ثابت
  - توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

- **توان مصرفی در 25^0C ، 3V ، 4MHZ**
  - حالت فعال 3.4mA (ACTIVE)
  - درحالت بی‌کاری 1.4mA (IDLE)
  - در حالت POWER – DOWN : < 1µA
- **ولتاژهای عملیاتی (کاری)**
  - 2.7V تا 6V برای (AT90LS2333/AT90LS4433)
  - 4V تا 6V برای (AT90S2333/AT90S4433)
- **فرکانسهای کاری**
  - 0MHZ تا 4MHZ برای (AT90LS2333/AT90LS4433)
  - 0MHZ تا 8MHZ (AT90S2333/AT90S4433)
- **خطوط I/O و انواع بسته‌بندی**
  - 20 خط ورودی / خروجی(I / O) قابل برنامه‌ریزی
  - 28 پایه (PIN) نوع PDIP و 32 پایه نوع TQFP
- **ترکیب پایه‌ها**



شکل ۲-۳ ترکیب بسته بندی

فیوز بیت‌های AT90S2333/4433

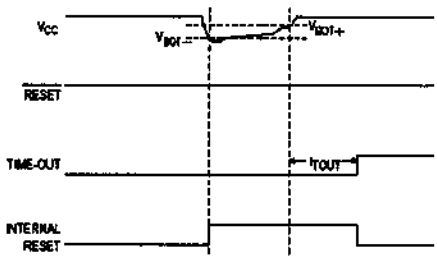
این میکروها دارای 6 فیوز بیت (SPIEN , BODLEVEL , BODEN , CKSEL2..0) است. فیوز بیت‌ها با پاک کردن(ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود. این بیت در زمان برنامه‌ریزی سریال قابل دسترس نمی‌باشد.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند ولی زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و طبق شکل۲-۴ سیستم را ری‌ست می‌کند. لازم به تذکر است که این بیت به همراه بیت‌های CKSEL2..0 زمان شروع (START UP) میکرو را نیز تعیین می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN – OUT این بیت بایستی برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.(جدول ۲-۳)

CKSEL2..0 : با برنامه‌ریزی کردن این بیت‌ها کلاک سیستم را می‌توان با توجه به جدول ۲-۴ در مُدهای مختلف تغییر داد. این بیت‌ها به صورت پیش‌فرض 010 هستند.



شکل ۲-۴ زمان بندی ری‌ست BROWN-OUT

جدول ۲-۳ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

جدول ۲-۴ انتخاب زمان RESET DELAY

CKSEL [2:0]	START-UP TIME, AT VCC = 2.7V	START-UP TIME, AT VCC = 5.0V	RECOMMENDED USAGE
000	16 ms + 6 CK	4 ms + 6 CK	EXTERNAL CLOCK, SLOWLY RISING POWER
001	6 CK	6 CK	EXTERNAL CLOCK,BOD ENABLED
010	256 ms + 16K CK	64 ms + 16K CK	CRYSTAL OSCILLATOR
011	16 ms + 16K CK	4 ms + 16K CK	CRYSTAL OSCILLATOR, FAST RISING POWER
100	16K CK	16K CK	CRYSTAL OSCILLATOR, BOD ENABLED
101	256 ms + 1K CK	64 ms + 1K CK	CERAMIC RESONATOR
110	16 ms + 1K CK	4 ms + 1K CK	CERAMIC RESONATOR, FAST RISING POWER
111	1K CK	1K CK	CERAMIC RESONATOR, BOD ENABLED

۲-۵ خصوصیات AT90S8515

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 118 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی

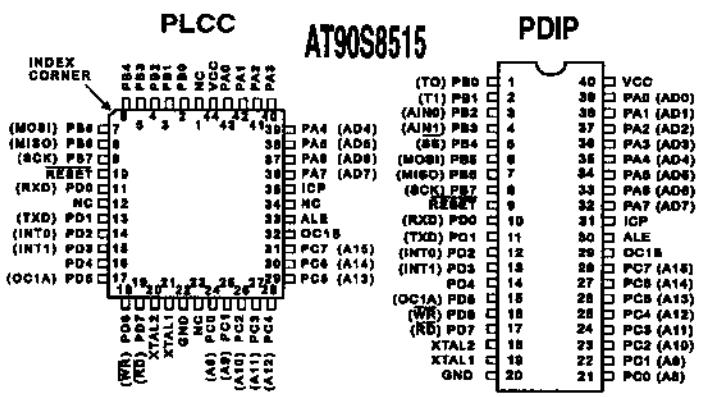
- **حافظه ، برنامه و دادهٔ غیر فرّار**
  - 8K بایت حافظه FLASH قابل برنامه‌ریزی داخلی
    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 512 بایت حافظه SRAM
  - 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

- **خصوصیات جانبی**
  - یک تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا
  - یک تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE ، CAPTURE و دو خروجی PWM ، 8 ، 9 یا 10 بیتی
  - یک مقایسه‌کننده آنالوگ داخلی
  - UART سریال قابل برنامه‌ریزی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING)
  - قابلیت ارتباط سریال SPI به صورت MASTER / SLAVE

- **خصوصیات ویژه میکروکنترلر**
  - تغذیه کم در مُدهای IDLE و POWERDOWN
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - عملکرد کاملاً ثابت
  - توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

- **توان مصرفی در 4MHZ ، 3V ، 25^0C**
  - حالت فعال 3.0mA (ACTIVE MODE)
  - درحالت بی‌کاری 1mA (IDLE MODE)
  - در حالت POWER - DOWN : < 1μA

- **ولتاژهای عملیاتی (کاری)**
  - 2.7V تا 6V برای (AT90S8515-4)
  - 4V تا 6V برای (AT90S8515-8)
- **فرکانسهای کاری**
  - 0MHZ تا 4MHZ برای (AT90S8515-4)
  - 0MHZ تا 8MHZ برای (AT90S8515-8)
- **خطوط I/O و انواع بسته‌بندی**
  - 32 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 40 پایه (PIN) نوع PDIP، 44 پایه نوع PLCC و TQFP
- **ترکیب پایه‌ها**



شکل ۲-۵ ترکیب بسته بندی

فیوز بیت‌های AT90S8515

این میکرو دارای 2 فیوز بیت (SPIEN , FSTRT) است. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

جدول ۲-۵ انتخاب زمان RESET DELAY

VCC	PARAMETER	Min	Typ	Max	Units
VCC = 5.0V	RESET DELAY TIME-OUT PERIOD AT90S/LS8515 FSTRT PROGRAMMED	0.25	0.28	0.31	ms
	RESET DELAY TIME-OUT PERIOD AT90S/LS8515 FSTRT UNPROGRAMMED	11.0	16.0	21.0	ms

نکته فیوز بیت‌های AT90S8515 در زمان برنامه‌ریزی به صورت سریال قابل دسترس نمی‌باشند.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

FSTRT : با برنامه‌ریزی کردن این بیت کوتاه‌ترین زمان شروع (START UP) از ری‌ست و مُدهای SLEEP در نظر گرفته می‌شود. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده و طولانی‌ترین زمان در نظر گرفته شده است. جدول ۲-۵ مشخص کننده این زمان است.

۲-۶ خصوصیات AT90S8535/ LS8535

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 118 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
- **حافظه ، برنامه و دادهٔ غیر فرّار**
  - 8K بایت حافظه FLASH قابل برنامه‌ریزی داخلی
    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 512 بایت حافظه SRAM
  - 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - دو تایمر/ کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE
  - یک تایمر/ کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE ، CAPTURE و دو خروجی PWM ، 8 ، 9 یا 10 بیتی
  - 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
  - یک مقایسه‌کننده آنالوگ داخلی
  - UART سریال قابل برنامه‌ریزی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN – SYSTEM PROGRAMMING)
  - قابلیت ارتباط سریال SPI به صورت MASTER / SLAVE
- **خصوصیات ویژه میکروکنترلر**
  - مدار POWER – ON RESET CIRCUIT

— دارای RTC (REAL-TIME CLOCK) بااسیلاتور مجزا
— تغذیه کم در مُدهای IDLE و POWERDOWN
— منابع وقفه (INTERRUPT) داخلی و خارجی
— دارای سه مُد SLEEP : IDLE ، POWER-DOWN و POWER-SAVE
— عملکرد کاملاً ثابت
— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

- **توان مصرفی در 4MHZ ، 3V ، 20^0C**

— حالت فعال 6.4mA (ACTIVE MODE)
— درحالت بی‌کاری 1.9mA (IDLE MODE)
— در حالت POWER – DOWN : < 1μA

- **ولتاژهای عملیاتی (کاری)**

— 2.7V تا 6V برای (AT90LS8535)
— 4V تا 6V برای (AT90S8535)

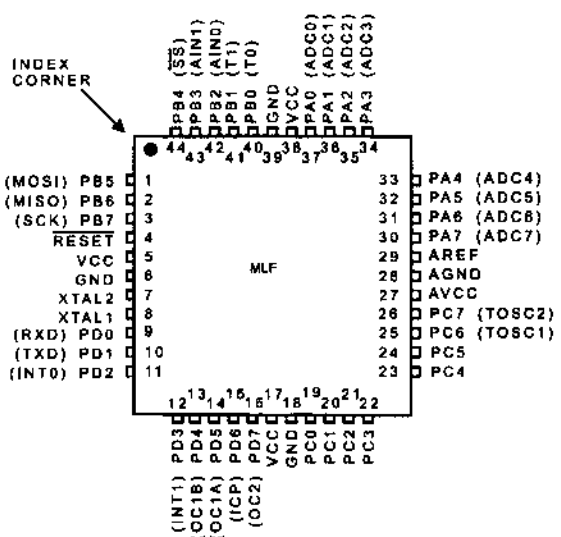
- **فرکانسهای کاری**

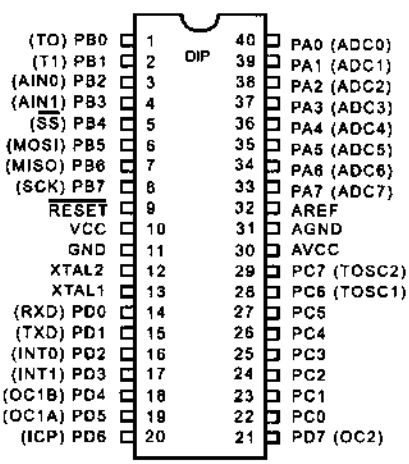
— 0MHZ تا 4MHZ برای (AT90LS8535)
— 0MHZ تا 8MHZ برای (AT90S8535)

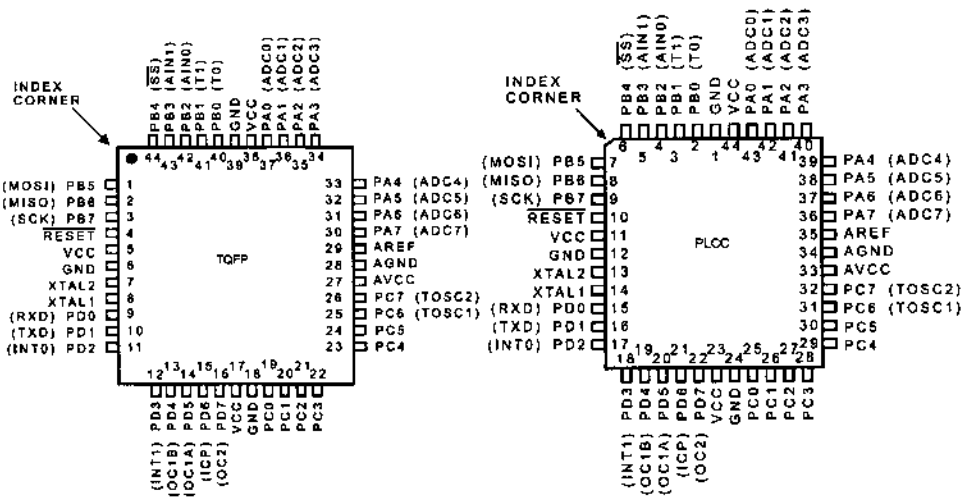
- **خطوط I/O و انواع بسته‌بندی**

— 32 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
— 40 پایه (PIN) نوع PDIP ،44 پایه نوع PLCC ،44 پایه نوع MLF و 44 پایه نوع TQFP

- **ترکیب پایه‌ها**







شکل ۲-۶ ترکیب بسته بندی

فیوز بیت‌های AT90S8535

این میکرو دارای 2 فیوز بیت (FSTRT , SPIEN) است. فیوز بیت‌ها با پاک کردن(ERASE) میکرو تاثیری نمی‌بینند. در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

FSTRT : با برنامه‌ریزی کردن این بیت کوتاه‌ترین زمان شروع (START UP) از ری‌ست و مُدهای SLEEP در نظر گرفته می‌شود. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده و طولانی‌ترین زمان در نظر گرفته شده است. جدول ۲-۶ مشخص کننده این زمان است.

جدول ۲-۶ انتخاب زمان Reset Characteristics

VCC	PARAMETER	Min	Typ	Max	Units
VCC = 5.0V	RESET DELAY TIME-OUT PERIOD AT90S/LS8535 FSTRT PROGRAMMED	1.0	1.1	1.2	ms
	RESET DELAY TIME-OUT PERIOD AT90S/LS8535 FSTRT UNPROGRAMMED	11.0	16.0	21.0	ms

۳

میکروکنترلرهای MEGAAVR

در این فصل به معرفی میکروکنترلرهای نوع MEGAAVR از سری میکروکنترلرهای AVR شرکت ATMEL می‌پردازیم. میکروهای MEGA نسبت به انواع (AT90S,TINY) دارای قابلیت بیشتری هستند. خصوصیات و قابلیت‌های هر یک از میکروهای نوع MEGAAVR تشریح و در ادامه فیوز بیت‌های هر یک به طور کامل بررسی شده است. فیوز بیت‌ها قسمتی از حافظه FLASH هستند که امکاناتی را در اختیار کاربر قرار می‌دهند. فیوز بیت‌ها با ERASE میکرو از بین نمی‌روند و می‌توانند توسط بیت‌های قفل مربوطه، قفل شوند. کلاک سیستم هر یک از میکروها در صورت نیاز به توضیح بیشتر بلافاصله بعد از فیوز بیت‌ها گفته شده است. دو بخش کلاک سیستم (۱) و (۲) به معرفی انواع کلاک سیستم میکروهای ارجاع شده به این دو بخش پرداخته است.

اهداف

۱. آشنایی کامل با انواع میکروهای MEGAAVR
۲. آشنایی کامل با فیوز بیت‌های هر یک از میکروها
۳. توانایی برنامه‌ریزی فیوز بیت‌های هر یک از میکروها
٤. توانایی برنامه‌ریزی فیوز بیت‌ها برای تعیین کلاک سیستم دلخواه

۳-۱ خصوصیات ATmega323 , ATmega323L

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 8MIPS در فرکانس 8MHZ
- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 32K بایت حافظه FLASH داخلی قابل برنامه‌ریزی
    پایداری حافظه FLASH : قابلیت 1,000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - 2K بایت حافظه داخلی SRAM
  - 1K بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **قابلیت ارتباط JTAG (IEEE Std.)**
  - برنامه‌ریزی برنامهٔ FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG
- **خصوصیات جانبی**
  - دو تایمر-کانتر (TIMER /COUNTER) 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE
  - یک تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE
  - 4 کانال PWM
  - 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
  - یک مقایسه‌کننده آنالوگ داخلی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING)
  - قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE
  - قابلیت ارتباط با پروتکل سریال دوسیمه (TWO - WIRE)
  - USART سریال قابل برنامه‌ریزی
- **خصوصیات ویژه میکروکنترلر**
  - مدار POWER - ON RESET CIRCUIT
  - BROWN-OUT DETECTION قابل برنامه‌ریزی
  - دارای 6 حالت SLEEP (POWER - DOWN ، POWER - SAVE ، IDLE ، STANDBY و EXTENDED STANDBY , ADC NOISE REDUCTION)

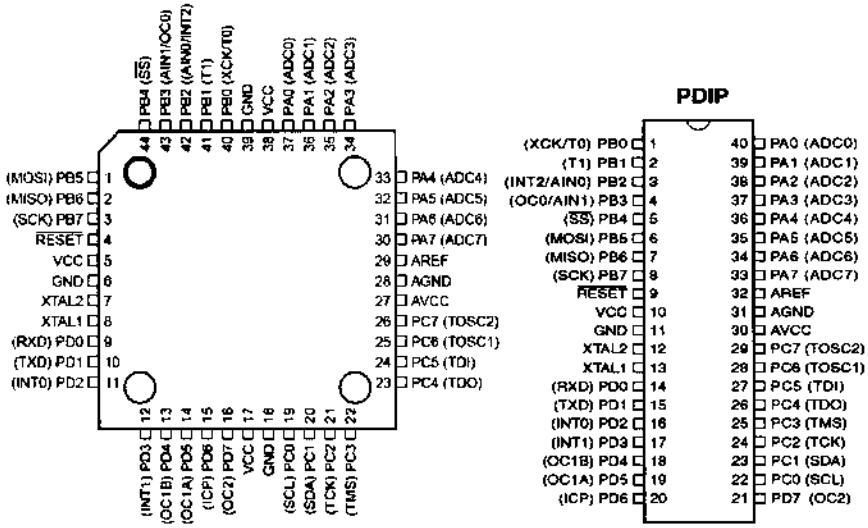
— منابع وقفه (INTERRUPT) داخلی و خارجی
— دارای اسیلاتور RC داخلی کالیبره شده
— عملکرد کاملاً ثابت
— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

- **ولتاژهای عملیاتی (کاری)**
  — 2.7V تا 5.5 برای(ATmega323L)
  — 4V تا 5.5V برای (ATmega323)

- **فرکانسهای کاری**
  — 0MHZ تا 4MHZ برای (ATmega323L)
  — 0MHZ تا 8MHZ برای (ATmega323)

- **خطوط I/O و انواع بسته‌بندی**
  — 32 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  — 40 پایه PDIP و 44 پایه TQFP

- **ترکیب پایه‌ها**



شکل ۳-۱ ترکیب بسته بندی

فیوز بیت‌های ATMEGA323

ATMEGA323 دارای دو بایت فیوز بیت طبق جدول‌های ۳-۱ و ۳-۲ می‌باشد .

جدول ۳-۱ بایت پر ارزش فیوز بیت میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD (ON CHIP DEBUG ENABLE)	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
-	4	-	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	1(UNPROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	1(UNPROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۲ بایت کم ارزش فیوز بیت میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
-	5	-	1(UNPROGRAMMED)
-	4	-	1(UNPROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

OCDEN : در صورتی که بیت‌های قفل برنامه‌ریزی نشده باشند برنامه‌ریزی این بیت به همراه بیت JTAGEN باعث می‌شود که سیستم ON CHIP DEBUG فعال شود. برنامه‌ریزی شدن این بیت به قسمتهایی از میکرو امکان می‌دهد که در مُدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می‌شود. این بیت به صورت پیش فرض برنامه‌ریزی نشده (1) است.

JTAGEN : بیتی برای فعال‌سازی برنامه‌ریزی میکرو از طریق استاندارد ارتباطی IEEE (JTAG) که در حالت پیش‌فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید . پایه‌های PC2..5 در این ارتباط استفاده می‌شود.

SPINE : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM پاک می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

جدول ۳-۳ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	256 words	4	$0000 - $3EFF	$3F00 - 3FFF	$3F00
1	0	512 words	8	$0000 - $3DFF	$3E00 - 3FFF	$3E00
0	1	1024 words	16	$0000 - $3BFF	$3C00 - 3FFF	$3C00
0	0	2048 words	32	$0000 - $37FF	$3800 - 3FFF	$3800

جدول۳ -۴ انتخاب آدرس بردار ری‌ست توسط فیوز بیت BOOTRST

BOOTRST	RESET ADDRESS
1(UNPROGRAMMED)	RESET VECTOR = APPLICATION RESET (ADDRESS $0000)
0(PROGRAMMED)	RESET VECTOR = BOOT LOADER RESET

BOOTSZ0,1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۳ برنامه‌ریزی می‌شوند و در صورت برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

BOOTRST : بیتی برای انتخاب بردار ری‌ست BOOT که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست $0000 است . در صورت برنامه‌ریزی این بیت آدرس بردار ری‌ست طبق جدول ۳-۴ به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند.

BODEN: برای فعال کردن عملکرد مدار BROWN - OUT ، این بیت بایستی برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است. (جدول ۳-۵)

جدول ۳-۵ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

جدول ۳-۶ انتخاب کلاک سیستم و زمان START-UP برای ATMEGA323

CKSEL3..0	Start-up Time, CC=2.7V,BODLEVEL UnProgrammed	Start-up Time, VCC = 4V, BODLEVEL programmed	Recommended Usage
0000	4.2 ms + 6 CK	5.8 ms + 6 CK	Ext. Clock, Fast Rising Power
0001	30 μs + 6 CK	10 μs + 6 CK	Ext. Clock, BOD Enabled
0010	67 ms + 6 CK	92 ms + 6 CK	Int. RC Oscillator, Slowly Rising Power(DEFAULT)
0011	4.2 ms + 6 CK	5.8 ms + 6 CK	Int. RC Oscillator, Fast Rising Power
0100	30 μs + 6 CK(4)	10 μs + 6 CK	Int. RC Oscillator, BOD Enabled
0101	67 ms + 6 CK	92 ms + 6 CK	Ext. RC Oscillator, Slowly Rising Power
0110	4.2 ms + 6 CK	5.8 ms + 6 CK	Ext. RC Oscillator, Fast Rising Power
0111	30 μs + 6 CK(4)	10 μs + 6 CK(5)	Ext. RC Oscillator, BOD Enabled
1000	67 ms + 32K CK	92 ms + 32K CK	Ext. Low-frequency Crystal
1001	67 ms + 1K CK	92 ms + 1K CK	Ext. Low-frequency Crystal
1010	67 ms + 16K CK	92 ms + 16K CK	Crystal Oscillator, Slowly Rising Power
1011	4.2 ms + 16K CK	5.8 ms + 16K CK	Crystal Oscillator, Fast Rising Power
1100	30 μs + 16K CK(4)	10 μs + 16K CK(5)	Crystal Oscillator, BOD Enabled
1101	67 ms 1K CK	92 ms + 1K CK	Ceramic Resonator/Ext.Clock, Slowly Rising Power
1110	4.2 ms + 1K CK	5.8 ms + 1K CK	Ceramic Resonator, Fast Rising Power
1111	30 μs + 1K CK(4)	10 μs + 1K CK(5)	Ceramic Resonator, BOD Enabled

CKSEL3 , CKSEL2 , CKSEL1 , CKSEL0 : عملکرد این بیت‌ها به همراه بیت BODLEVEL زمان شروع (START UP) و کلاک سیستم را با توجه به جدول ۳-۶ مشخص می‌کنند. کلاک پیش‌فرض میکرو (0010) INTERNAL RC OSCILLATOR @ 1MHZ با زمان شروع طولانی است.

۳-۲ خصوصیات ATmega32L , ATmega32

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 131 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 16MIPS در فرکانس 16MHZ

• **حافظه ، برنامه و دادهٔ غیرفرّار**

— 32K بایت حافظه FLASH داخلی قابل برنامه‌ریزی

پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن (WRITE / ERASE)

— 2K بایت حافظه داخلی SRAM

— 1024 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن

— قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

• **قابلیت ارتباط JTAG (IEEE Std.)**

— برنامه‌ریزی برنامهٔ FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG

• **خصوصیات جانبی**

— دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE.

— یک تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE ، CAPTURE

— 4 کانال PWM

— 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی

دارای دو کانال تفاضلی با کنترل گین 1x ، 10x و 200x

— یک مقایسه‌کننده آنالوگ داخلی

— دارای RTC (REAL -TIME CLOCK) بااسیلاتور مجزا

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING)

— قابلیت ارتباط سریال SPI به صورت MASTER یاSLAVE

— قابلیت ارتباط با پروتکل سریال دوسیمه (TWO - WIRE)

— USART سریال قابل برنامه‌ریزی

• **خصوصیات ویژه میکروکنترلر**

— POWER - ON RESET CIRCUIT.

— BROWN-OUT DETECTION قابل برنامه‌ریزی

— دارای 6 حالت SLEEP (POWER - DOWN ، POWER - SAVE ، IDLE ، STANDBY ، EXTENDED STANDBY و ADC NOISE REDUCTION)

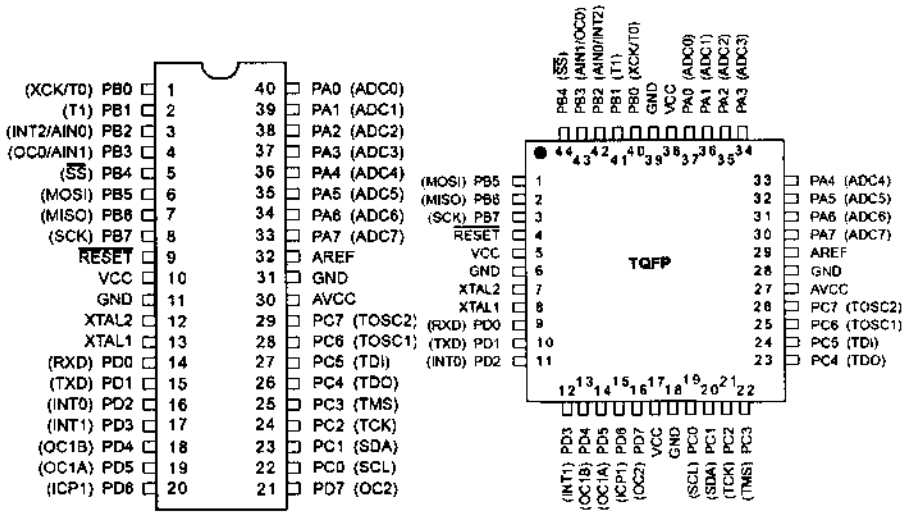
— منابع وقفه (INTERRUPT) داخلی و خارجی

— دارای اسیلاتور RC داخلی کالیبره شده

— عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

- **ولتاژهای عملیاتی (کاری)**
  - 2.7V تا 5.5 برای(ATmega32L)
  - 4.5V تا 5.5V برای (ATmega32)
- **فرکانسهای کاری**
  - 0MHZ تا 8MHZ برای (ATmega32L)
  - 0MHZ تا 16MHZ برای (ATmega32)
- **خطوط I/O و انواع بسته‌بندی**
  - 32 خط ورودی/خروجی (I/O) قابل برنامه‌ریزی
  - 40 پایه PDIP ، 44 پایه TQFP و 44 پایه MLF
- **ترکیب پایه‌ها**



شکل ۳-۲ ترکیب بسته بندی

فیوز بیت‌های ATMEGA32

ATMEGA32 دارای دو بایت فیوز بیت طبق جدول‌های ۳-۷ و۳-۸ می‌باشد.

OCDEN : در صورتی که بیت‌های قفل برنامه‌ریزی نشده باشند برنامه‌ریزی این بیت به همراه بیت JTAGEN باعث می‌شود که سیستم ON CHIP DEBUG فعال شود. برنامه‌ریزی شدن این بیت به قسمت‌هایی از میکرو امکان می‌دهد که در مُدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می‌شود. این بیت به صورت پیش فرض برنامه‌ریزی نشده (1) است.

JTAGEN : بیتی برای فعال‌سازی برنامه‌ریزی میکرو از طریق استاندارد ارتباطیIEEE (JTAG) که در حالت پیش‌فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید. پایه‌های PC2...5 در این ارتباط استفاده می‌شود.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

جدول ۳-۷ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD (ON CHIP DEBUG ENABLE)	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
CKOPT	4	OSCILLATOR OPTIONS	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۸ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

CKOPT : انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در قسمت کلاک سیستم (۱) در انتهای همین فصل آمده است.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM پاک می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

BOOTSZ0,1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۹ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از این آدرس حافظه BOOT آغاز خواهد شد.

BOOTRST : انتخاب بردار ری‌ست BOOT که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست 0000$ است و در صورت برنامه‌ریزی این بیت آدرس بردار ری‌ست طبق جدول ۳-۱۰ به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایهVCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و میکرو را ری‌ست می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN – OUT این بیت بایستی برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است. (جدول ۳-۱۱)

SUT0 ، SUT1 : برای انتخاب زمان START –UP بکار برده می‌شوند که عملکرد این دو بیت در بخش کلاک سیستم در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL0 ... CKSEL3 : عملکرد این بیت‌ها در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض کلاک برای این میکرو INTERNAL RC OSCILLATOR @ 1MHZ است.

جدول ۳-۹ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	256 words	4	$0000 - $3EFF	$3F00 - $3FFF	$3F00
1	0	512 words	8	$0000 - $3DFF	$3E00 - $3FFF	$3E00
0	1	1024 words	16	$0000 - $3BFF	$3C00 - $3FFF	$3C00
0	0	2048 words	32	$0000 - $37FF	$3800 - $3FFF	$3800

جدول ۳-۱۰ انتخاب آدرس بردار ری‌ست توسط فیوز بیت BOOTRST

BOOTRST	RESET ADDRESS
1(UNPROGRAMMED)	RESET VECTOR = APPLICATION RESET (ADDRESS $0000)
0(PROGRAMMED)	RESET VECTOR = BOOT LOADER RESET

جدول ۳-۱۱ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

۳-۳ خصوصیات Atmega128 , Atmega128L

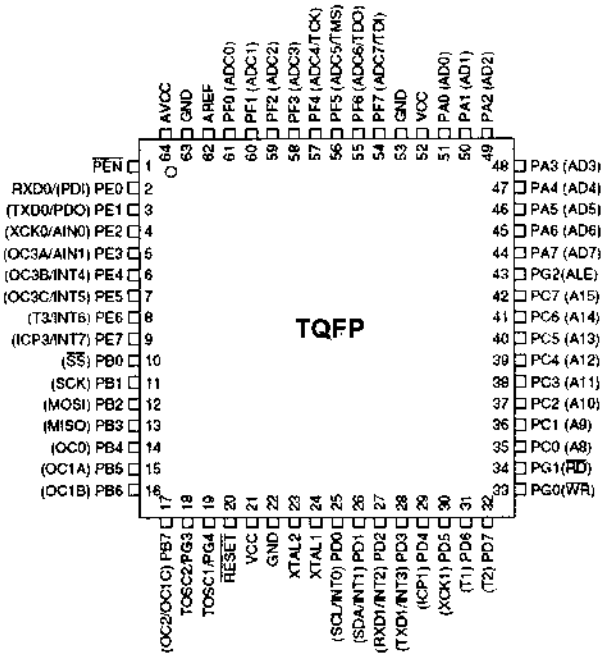
- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 133 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 16MIPS در فرکانس 16MHZ

- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 128K بایت حافظه FLASH داخلی قابل برنامه‌ریزی
    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 4K بایت حافظه داخلی SRAM
  - قابلیت آدرس‌دهی 64K بایت حافظه خارجی
  - 4K بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

- **قابلیت ارتباط JTAG (IEEE Std.)**
  - برنامه‌ریزی برنامهٔ FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG

- **خصوصیات جانبی**
  - دو تایمر – کانتر 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE
  - دو تایمر – کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE ، CAPTURE
  - 2 کانال PWM هشت بیتی
  - 6 کانال PWM با قابلیت وضوح 2 تا 16 بیتی
  - 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
    8 کانال مبدل SINGLE-ENDED
    7 کانال ADC تفاضلی
    دارای دو کانال با کنترل گین 1x ، 10x و200x
  - یک مقایسه‌کننده آنالوگ داخلی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN – SYSTEM PROGRAMMING)
  - قابلیت ارتباط SPI (SERIAL PERIPHERAL INTERFACE) به صورت SLAVE /MASTER
  - قابلیت ارتباط با پروتکل سریال دوسیمه (TWO – WIRE)
  - دو USART سریال قابل برنامه‌ریزی

- **خصوصیات ویژه میکروکنترلر**
  - POWER – ON RESET CIRCUIT.
  - BROWN-OUT DETECTION قابل برنامه‌ریزی
  - انتخاب نرم‌افزاری فرکانس کلاک سیستم
  - دارای 6 حالت SLEEP
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - دارای اسیلاتور RC داخلی کالیبره شده
- **ولتاژهای عملیاتی (کاری)**
  - 2.7V تا 5.5 برای(Atmega128L)
  - 4.5V تا 5.5V برای (Atmega128)
- **فرکانسهای کاری**
  - 0MHZ تا 8MHZ برای (Atmega128L)
  - 0MHZ تا 16MHZ برای (Atmega128)
- **خطوط I/O و انواع بسته‌بندی**
  - 53 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 64-lead TQFP و 64-pad MLF



شکل ۳-۳ ترکیب بسته بندی ATMEGA128

فیوز بیت‌های ATMEGA128

ATMEGA128 دارای سه بایت فیوز بیت طبق جدول‌های ۳-۱۲ و ۳-۱۳ و ۳-۱۴ می‌باشد .

جدول ۳-۱۲ بایت پیشرفته فیوز بیتهای میکرو

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1(UNPROGRAMMED)
-	6	-	1(UNPROGRAMMED)
-	5	-	1(UNPROGRAMMED)
-	4	-	1(UNPROGRAMMED)
-	3	-	1(UNPROGRAMMED)
-	2	-	1(UNPROGRAMMED)
M103C	1	MEGA103 COMPATIBILITY MODE	1(UNPROGRAMMED)
WDTON	0	WATCHDOG TIMER ALWAYS ON	1(UNPROGRAMMED)

جدول ۳-۱۳ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
CKOPT	4	OSCILLATOR OPTIONS	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS RESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۱۴ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

M103C : دو میکرو MEGA103 و MEGA128 بسیار با هم مطابقت دارند ولی برای رفع اندک اختلاف موجود، می‌توان با برنامه‌ریزی این بیت به طور 100% این دو میکرو را با هم مطابقت داد و در مدار به جای ATMEGA103 از ATMEGA128 استفاده نمود.

WDTON : در حالت پیش‌فرض WATCHDOG غیرفعال و کاربر بایستی نرم‌افزاری WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن و کاربر بایستی نرم افزاری WATCHDOG را ریست کند.

OCDEN : در صورتی که بیت‌های قفل برنامه‌ریزی نشده باشند برنامه‌ریزی این بیت به همراه بیت JTAGEN باعث می‌شود که سیستم ON CHIP DEBUG فعال شود. برنامه‌ریزی شدن این بیت به قسمتهایی از میکرو امکان می‌دهد که در مُدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می‌شود. این بیت به صورت پیش فرض برنامه‌ریزی نشده (1) است.

JTAGEN : بیتی برای فعال‌سازی برنامه‌ریزی میکرو از طریق استاندارد ارتباطی IEEE (JTAG) که در حالت پیش‌فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

CKOPT : انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در بخش کلاک سیستم (۱) آمده است.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM پاک می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

BOOTSZ1 , BOOTSZ0 : بیت‌هایی برای انتخاب مقدار حافظه BOOT که طبق جدول ۳-۱۵ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

BOOTRST : بیتی برای انتخاب بردار ریست BOOT که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ریست $0000 است و در صورت برنامه‌ریزی آدرس بردار ریست طبق جدول ۳-۱۶ به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

جدول ۳-۱۵ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	512 words	4	$0000 - $FDFF	$FE00 -$FFFF	$FE00
1	0	1024 words	8	$0000 - $FBFF	$FC00 - $FFFF	$FC00
0	1	2048 words	16	$0000 - $F7FF	$F800 - $FFFF	$F800
0	0	4096 words	32	$0000 - $EFFF	$F000 - $FFFF	$F000

جدول ۳-۱۶ انتخاب آدرس بردار ری‌ست توسط فیوز بیت BOOTRST

BOOTRST	RESET ADDRESS
1(UNPROGRAMMED)	RESET VECTOR = APPLICATION RESET (ADDRESS $0000)
0(PROGRAMMED)	RESET VECTOR = BOOT LOADER RESET

جدول ۳-۱۷ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN - OUT (طبق جدول ۳-۱۷) این بیت بایستی برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

SUT1، SUT0 : این دو بیت برای انتخاب زمان START -UP بکار برده می‌شوند. عملکرد آنها در بخش کلاک سیستم (۱) در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL3 ... CKSEL0 : عملکرد این بیت‌ها در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض INTERNAL RC OSCILLATOR @ 1MHZ است.

۳-۴ خصوصیات Atmega163L , Atmega163

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 8MIPS در فرکانس 8MHZ
- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 16K بایت حافظه FLASH داخلی قابل برنامه‌ریزی

    پایداری حافظه FLASH : قابلیت 1,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 1024 بایت حافظه داخلی SRAM

— 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن

— قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

• **خصوصیات جانبی**

— دو تایمر - کانتر 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE

— یک تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE

— 3 کانال PWM

— دارای RTC (REAL -TIME CLOCK) بااسیلاتور مجزا

— 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی

— یک مقایسه‌کننده آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING)

— قابلیت ارتباط SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE

— قابلیت ارتباط با پروتکل سریال دوسیمه (TWO - WIRE)

— UART سریال قابل برنامه‌ریزی

• **خصوصیات ویژه میکروکنترلر**

— مدار POWER - ON RESET داخلی

— مدار BROWN-OUT DETECTION قابل برنامه‌ریزی

— دارای 4 حالت SLEEP (POWER - DOWN ، POWER- SAVE ، ADC NOISE REDUCTION و IDLE)

— منابع وقفه (INTERRUPT) داخلی و خارجی

— دارای اسیلاتور RC داخلی کالیبره شده

— عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• **توان مصرفی در 4MHZ ، 3V ، 25^0C**

— حالت فعال 5.0mA (ACTIVE MODE)

— درحالت بی‌کاری 1.9mA (IDLE MODE)

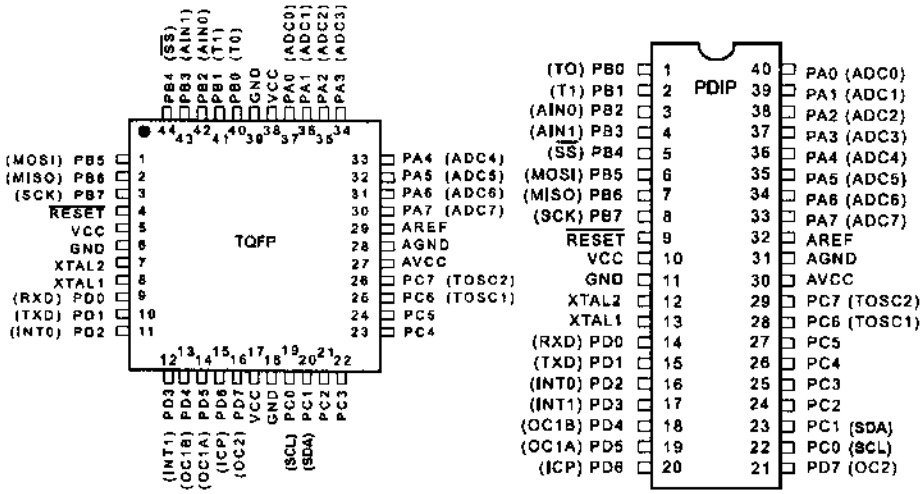
— در حالت POWER - DOWN : < 1µA

• **ولتاژهای عملیاتی (کاری)**

— 2.7V تا 5.5V برای (Atmega163L)

— 4.0V تا 5.5V برای (Atmega163)

- **فرکانسهای کاری**
  - 0MHZ تا 4MHZ برای (Atmega163L)
  - 0MHZ تا 8MHZ برای (Atmega163)
- **خطوط I/O و انواع بسته‌بندی**
  - 32 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 40 پای PDIP و 44 پایه TQFP
- **ترکیب پایه‌ها**



شکل ۳-۴ ترکیب بسته بندی

فیوز بیت‌های ATMEGA163

ATMEGA163 دارای دو بایت فیوز بیت طبق جدول‌های۳-۱۸ و ۳-۱۹ می‌باشد .

BOOTSZ0,1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۲۰ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

BOOTRST : بیتی برای انتخاب بردار ری‌ست BOOT که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست $0000 است و در صورت برنامه‌ریزی آدرس بردار ری‌ست طبق جدول ۳-۲۱ به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN - OUT ،این بیت بایستی برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

جدول ۳-۱۸ بایت پرارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1(UNPROGRAMMED)
-	6	-	1(UNPROGRAMMED)
-	5	-	1(UNPROGRAMMED)
-	4	-	1(UNPROGRAMMED)
-	3	-	1(UNPROGRAMMED)
BOOTSZ1	2	SELECT BOOT SIZE	1(UNPROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	1(UNPROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۱۹ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
-	4	-	1(UNPROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

جدول ۳-۲۰ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash ddresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	2	$0000 - $1F7F	$1F80 - $1FFF	$1F80
1	0	256 words	4	$0000 - $1EFF	$1F00 - $1FFF	$1F00
0	1	512 words	8	$0000 - $1DFF	$1E00 - $1FFF	$1E00
0	0	1024 words	16	$0000 - $1BFF	$1C00 - $1FFF	$1C00

جدول ۳-۲۱ انتخاب آدرس بردار ری‌ست توسط فیوز بیت BOOTRST

BOOTRST	RESET ADDRESS
1(UNPROGRAMMED)	RESET VECTOR = APPLICATION RESET (ADDRESS $0000)
0(PROGRAMMED)	RESET VECTOR = BOOT LOADER RESET

جدول ۳-۲۲ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

CKSEL3 ... CKSEL0 : عملکرد این بیت‌ها به همراه بیت BODLEVEL زمان شروع کلاک سیستم را با توجه به جدول ۳-۲۳ مشخص می‌کند. کلاک پیش‌فرض (0010) اسیلاتور RC داخلی INTERNAL RC OSCILLATOR @ 1MHZ با زمان شروع طولانی است.

جدول ۳-۲۳ انتخاب کلاک سیستم و زمان START-UP برای ATMEGA163

CKSEL 3..0	Start-up Time, VCC=2.7V, BODLEVEL Programmed	Start-up Time, VCC = 4V, BODLEVEL programmed	Recommended Usage
0000	4.2 ms + 6 CK	5.8 ms + 6 CK	Ext. Clock, Fast Rising Power
0001	30 μs + 6 CK	10 μs + 6 CK	Ext. Clock, BOD Enabled
0010	67 ms + 6 CK	92 ms + 6 CK	Int. RC Oscillator, Slowly Rising Power(DEFAULT)
0011	4.2 ms + 6 CK	5.8 ms + 6 CK	Int. RC Oscillator, Fast Rising Power
0100	30 μs + 6 CK	10 μs + 6 CK	Int. RC Oscillator, BOD Enabled
0101	67 ms + 6 CK	92 ms + 6 CK	Ext. RC Oscillator, Slowly Rising Power
0110	4.2 ms + 6 CK	5.8 ms + 6 CK	Ext. RC Oscillator, Fast Rising Power
0111	30 μs + 6 CK	10 μs + 6 CK	Ext. RC Oscillator, BOD Enabled
1000	67 ms + 32K CK	92 ms + 32K CK	Ext. Low-frequency Crystal
1001	67 ms + 1K CK	92 ms + 1K CK	Ext. Low-frequency Crystal
1010	67 ms + 16K CK	92 ms + 16K CK	Crystal Oscillator, Slowly Rising Power
1011	4.2 ms + 16K CK	5.8 ms + 16K CK	Crystal Oscillator, Fast Rising Power
1100	30 μs + 16K CK	10 μs + 16K CK	Crystal Oscillator, BOD Enabled
1101	67 ms 1K CK	92 ms + 1K CK	CeramicResonator/Ext.Clock,SlowlyRising Power
1110	4.2 ms + 1K CK	5.8 ms + 1K CK	Ceramic Resonator, Fast Rising Power
1111	30 μs + 1K CK	10 μs + 1K CK	Ceramic Resonator, BOD Enabled

۳-۵ خصوصیات Atmega8L , Atmega8

- **از معماری AVR RISC استفاده می‌کند.**

— کارایی بالا و توان مصرفی کم

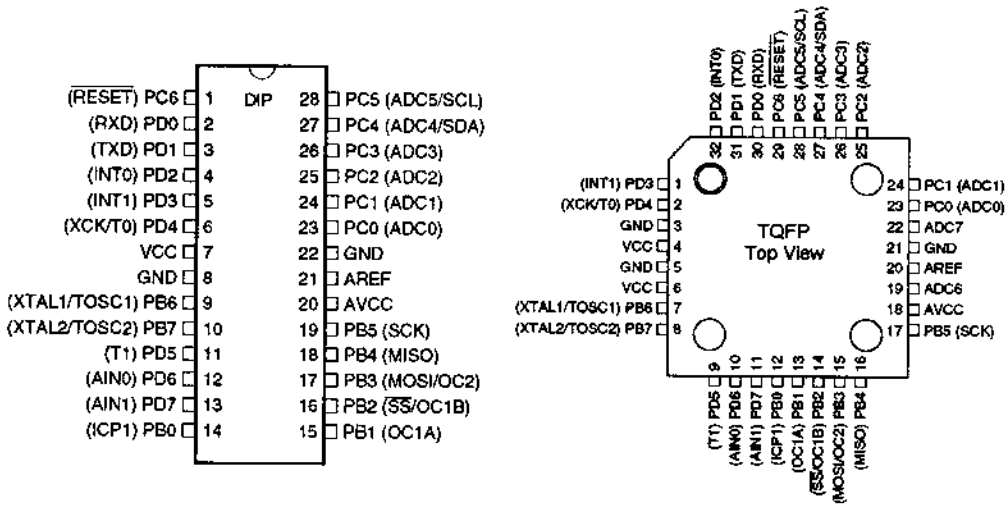
— دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.

— 32*8 رجیستر کاربردی

— سرعتی تا 16MIPS در فرکانس 16MHZ

- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 8K بایت حافظه FLASH داخلی قابل برنامه‌ریزی
    پایداری حافظه FLASH : قابلیت 10000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - 1024 بایت حافظه داخلی SRAM
  - 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - دو تایمر - کانتر 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE
  - یک تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE
  - 3 کانال PWM
  - 8 کانال مبدل آنالوگ به دیجیتال در بسته‌بندی های TQFP و MLF
    6 کانال با دقت 10 بیتی
    2 کانال با دقت 8 بیتی
  - 6 کانال مبدل آنالوگ به دیجیتال در بسته‌بندی PDIP
    4 کانال با دقت 10 بیتی
    2 کانال با دقت 8 بیتی
  - دارای RTC (REAL -TIME CLOCK) با اسیلاتور مجزا
  - یک مقایسه‌کننده آنالوگ داخلی
  - USART سریال قابل برنامه‌ریزی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING)
  - قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE
  - قابلیت ارتباط با پروتکل سریال دوسیمه (TWO - WIRE)
- **خصوصیات ویژه میکروکنترلر**
  - POWER - ON RESET CIRCUIT.
  - دارای 5 حالت SLEEP (POWER -DOWN ، POWER - SAVE، ADC NOISE REDUCTION
    ، IDLE و STANDBY)
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - دارای اسیلاتور RC داخلی کالیبره شده
  - عملکرد کاملاً ثابت
  - توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

- **توان مصرفی در 4MHZ ، 3V ، 25^0C**
  - حالت فعال 3.6 mA (ACTIVE MODE)
  - درحالت بی‌کاری 1.0mA (IDLE MODE)
  - در حالت POWER – DOWN : < 5µA
- **ولتاژهای عملیاتی (کاری)**
  - 2.7V تا 5.5 برای(Atmega8L)
  - 4.5V تا 5.5V برای (Atmega8)
- **فرکانسهای کاری**
  - 0MHZ تا 8MHZ برای (Atmega8L)
  - 0MHZ تا 16MHZ برای (Atmega8)
- **خطوط I/O و انواع بسته‌بندی**
  - 23 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 28 پایه PDIP و 32 پایه TQFP و MLF
- **ترکیب پایه‌ها**



شکل ۳–۵ ترکیب بسته بندی ATMEGA8

فیوز بیت‌های ATMEGA8

ATMEGA8 دارای دو بایت فیوز بیت است که در دو جدول ۳-۲۴ و۳–۲۵ نشان داده شده‌اند. منطق 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

جدول ۳-۲۴ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
RSTDISBL	7	SELECT IF PC6 IS I/O PIN OR RESET PIN	1(UNPROGRAMMED , PC6 IS RESET PIN)
WDTON	6	WDT ALWAYS ON	1(UNPROGRAMMED , WDT ENABLED BY WDTCR)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
CKOPT	4	OSCILLATOR OPTIONS	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول۳-۲۵ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

RSTDISBL : در حالت پیش‌فرض PC6 پایه ری‌ست است. با برنامه‌ریزی این بیت، پایه PC6 به عنوان پایه I/O استفاده می‌شود.

WDTON : در حالت پیش‌فرض WATCHDOG غیرفعال وکاربر بایستی نرم‌افزاری، WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن است.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود. این بیت در مُد برنامه‌ریزی سریال قابل دسترس نمی‌باشد.

CKOPT : بیت انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در بخش ۳-۱۸ در انتهای همین فصل آمده است.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM پاک می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو، محفوظ می‌ماند.

BOOTSZ0,1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۲۶ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

BOOTRST : بیتی برای انتخاب بردار ری‌ست BOOT که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست 0000$ است و در صورت برنامه‌ریزی آدرس بردار ری‌ست به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و میکرو را ری‌ست می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN –OUT ، این بیت بایستی برنامه‌ریزی شده باشد . (جدول ۳-۲۸)

جدول ۳-۲۶ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	4	0x000 - 0xF7F	0xF80 - 0xFFF	0xF80
1	0	256 words	8	0x000 - 0xEFF	0xF00 - 0xFFF	0xF00
0	1	512 words	16	0x000 - 0xDFF	0xE00 - 0xFFF	0xE00
0	0	1024 words	32	0x000 - 0xBFF	0xC00 - 0xFFF	0xC00

جدول ۳-۲۸ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

SUT1 ،SUT0 : برای انتخاب زمان START –UP بکار برده می‌شوند. عملکرد این دو بیت در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL3 .... CKSEL0 : عملکرد این بیت‌ها در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است.

۶-۳ خصوصیات ATMEGA8515L , ATMEGA8515

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - 16MIPS در فرکانس 16MHZ
- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 8K بایت حافظه FLASH قابل برنامه‌ریزی داخلی

    پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 512 بایت حافظه SRAM
  - قابلیت آدرس‌دهی 64K بایت حافظه خارجی
  - 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - یک تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و مُد مقایسه‌ای
  - یک تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE , CAPTURE
  - سه کانال PWM
  - USART سریال قابل برنامه‌ریزی
  - یک مقایسه‌کننده آنالوگ داخلی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING)
  - قابلیت ارتباط سریال SPI به صورت MASTER / SLAVE
- **خصوصیات ویژه میکروکنترلر**
  - POWER - ON RESET CIRCUIT و مدار BROWN - OUT قابل برنامه‌ریزی
  - دارای اسیلاتور RC داخلی کالیبره شده
  - تغذیه کم در مُدهای IDLE وPOWERDOWN
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - دارای 3 مُد SLEEP (POWER - DOWN ، STANDBY ، IDLE)
  - عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• **ولتاژهای عملیاتی (کاری)**

— 2.7V تا 5.5V برای (ATMEGA8515L)

— 4.5V تا 5.5V برای (ATMEGA8515)

• **فرکانسهای کاری**

— 0MHZ تا 8MHZ برای (ATMEGA8515L)

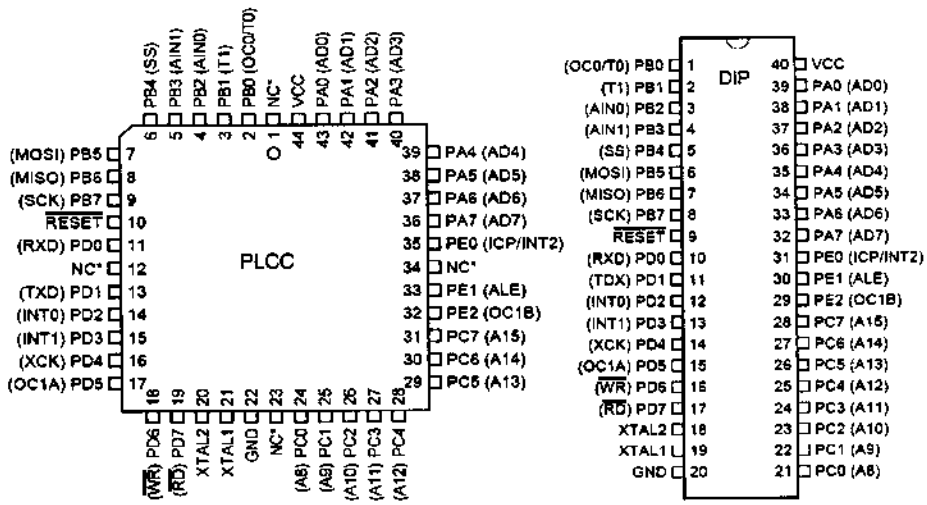
— 0MHZ تا 16MHZ برای (ATMEGA8515)

• **خطوط I/O و انواع بسته‌بندی**

— 35 خط ورودی/خروجی (I/O) قابل برنامه‌ریزی

— 40 پایه نوع PDIP ،44 پایه نوع PLCC ، TQFP و MLF

• **ترکیب پایه‌ها**



شکل ۳-۶ ترکیب بسته بندی

فیوز بیت‌های ATMEGA8515

ATMEGA8515 دارای دو بایت فیوز بیت طبق جدول‌های ۳-۲۹ و ۳-۳۰ است .

S8515C : ATMEGA8515 علاوه بر تمام امکانات AT90S8515 دارای امکانات دیگری نیز است. این دو میکرو بسیار با هم مطابقت دارند ولی برای رفع اندک اختلاف موجود، می‌توان با برنامه‌ریزی این فیوزبیت به طور 100% این دو میکرو را با هم مطابقت داد و در مدار به جای AT90S8515 از ATMEGA8515 استفاده نمود.

جدول۳-۲۹ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
S8515C	7	AT90S4414/8515 COMPATIBILITY MODE	1(UNPROGRAMMED)
WDTON	6	WDT TIMER ALWAYS ON	1(UNPROGRAMMED , WDT ENABLED BY WDTCR)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
CKOPT	4	OSCILLATOR OPTIONS	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۳۰ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

WDTON : در حالت پیش‌فرض WATCHDOG غیر فعال و کاربر بایستی نرم‌افزاری WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن است.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

CKOPT : بیتی جهت انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در بخش ۳-۱۸ در انتهای همین فصل آمده است.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM ری‌ست می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو ، محفوظ می‌ماند.

BOOTSZ0,1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۳۱ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST ، اجرای برنامه از آدرس حافظهBOOT آغاز خواهد شد.

جدول ۳-۳۱ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	4	0x000 - 0xF7F	0xF80 - 0xFFF	0xF80
1	0	256 words	8	0x000 - 0xEFF	0xF00 - 0xFFF	0xF00
0	1	512 words	16	0x000 - 0xDFF	0xE00 - 0xFFF	0xE00
0	0	1024 words	32	0x000 - 0xBFF	0xC00 - 0xFFF	0xC00

جدول ۳-۳۲ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

BOOTRST : بیتی برای انتخاب بردار ری‌ست BOOT است که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست $0000 است و در صورت برنامه‌ریزی آدرس بردار ری‌ست به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و میکرو را ری‌ست می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN – OUT ، این بیت بایستی برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.(جدول ۳-۳۲)

SUT0 ، SUT1 : برای انتخاب زمان START -UP بکار برده می‌شوند. عملکرد این دو بیت در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL0 ... CKSEL3 : عملکرد این بیت‌ها در بخش ۳-۱۸ در انتهای همین فصل به طور کامل توضیح داده شده است.

۳-۷ خصوصیات ATMEGA8535L , ATMEGA8535

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی

• **حافظه ، برنامه و دادهٔ غیرفرّار**

— 8K بایت حافظه FLASH قابل برنامه‌ریزی داخلی

پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن(WRITE / ERASE)

— 512 بایت حافظه SRAM

— 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن

— قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

• **خصوصیات جانبی**

— دو تایمر – کانتر 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE

— یک تایمر – کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای CAPTURE , COMPARE

— 4 کانال خروجی PWM

— 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی

8 کانال SINGLE-ENDED

دارای 7 کانال تفاضلی در بسته‌بندی TQFP

دارای دو کانال تفاضلی با کنترل گین 1x ، 10x و200x در بسته‌بندی TQFP

— یک مقایسه‌کننده آنالوگ داخلی

— USART سریال قابل برنامه‌ریزی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— قابلیت ارتباط با پروتکل سریال دوسیمه (TWO – WIRE)

— قابلیت ارتباط سریال SPI به صورت MASTER / SLAVE

• **خصوصیات ویژه میکروکنترلر**

— POWER – ON RESET CIRCUIT و مدار BROWN – OUT قابل برنامه‌ریزی

— دارای اسیلاتور RC داخلی کالیبره شده

— دارای RTC (REAL –TIME CLOCK) بااسیلاتور مجزا

— منابع وقفه (INTERRUPT) داخلی و خارجی

— دارای 6 حالت SLEEP (POWER – DOWN ، IDLE ، POWER – SAVE ، STANDBY ، EXTENDED STANDBY, و ADC NOISE REDUCTION)

— عملکرد کاملاً ثابت

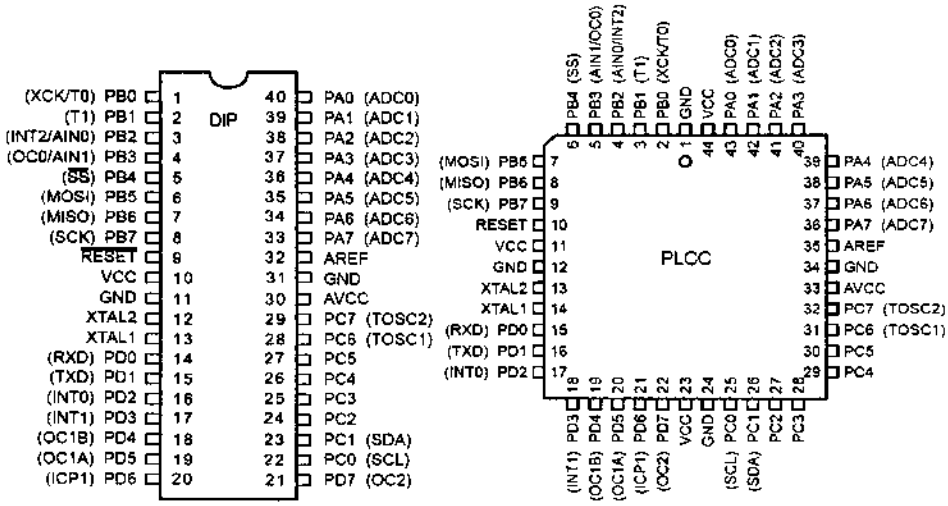
— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• **ولتاژهای عملیاتی (کاری)**

— 2.7V تا 5.5V برای (ATMEGA8535L)

— 4.5V تا 5.5V برای (ATMEGA8535)

- **فرکانسهای کاری**
  - — 0MHZ تا 8MHZ برای (ATMEGA8535L)
  - — 0MHZ تا 16MHZ برای (ATMEGA8535)
- **خطوط I/O و انواع بسته‌بندی**
  - — 32 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - — 40 پایه (PIN) نوع PDIP، 44 پایه نوع PLCC، 44 پایه نوع MLF و 44 پایه نوع TQFP
- **ترکیب پایه‌ها**



فیوز بیت‌های ATMEGA8535

این میکرو دارای دو بایت فیوز بیت طبق جدول‌های ۳-۳۳ و ۳-۳۴ می‌باشد.

جدول ۳-۳۳ بایت پرارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
S8535C	7	AT90S8535 COMPATIBILITY MODE	1(UNPROGRAMMED)
WDTON	6	WDT TIMER ALWAYS ON	1(UNPROGRAMMED , WDT ENABLED BY WDTCR)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
CKOPT	4	OSCILLATOR OPTIONS	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۳۴ بایت کم ارزش فیوزبیتهای میکرو

FUSE IOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

S8535C: ATMEGA8535 دارای تمام امکانات AT90S8535 است. به علاوه امکانات دیگری نیز به این میکرو اضافه شده است. این دو میکرو بسیار با هم مطابقت دارند ولی برای رفع اندک اختلاف موجود می‌توان با برنامه‌ریزی این بیت به طور 100% این دو میکرو را با هم مطابقت داد و در مدار به جای AT90S8535 از ATMEGA8535 استفاده نمود.

WDTON : در حالت پیش‌فرض WATCHDOG غیر فعال و کاربر بایستی نرم‌افزاری WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن است.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

CKOPT : بیت انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در قسمت منابع کلاک آمده است.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM ری‌ست می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو ، محفوظ می‌ماند.

BOOTSZ1 , BOOTSZ0 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳- ۳۵ برنامه‌ریزی می‌شوند.

جدول۳-۳۵ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	4	0x000 - 0xF7F	0xF80 -0xFFF	0xF80
1	0	256 words	8	0x000 - 0xEFF	0xF00 - 0xFFF	0xF00
0	1	512 words	16	0x000 - 0xDFF	0xE00 - 0xFFF	0xE00
0	0	1024 words	32	0x000 - 0xBFF	0xC00 - 0xFFF	0xC00

BOOTRST : بیتی برای انتخاب بردار ری‌ست BOOT است که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست 0000$ است و در صورت برنامه‌ریزی آدرس بردار ری‌ست به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و میکرو را ری‌ست می‌کند.

جدول ۳-۳۷ سطوح مختلف ولتاژ برای مدار BROWN-OUT

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

BODEN : برای فعال کردن عملکرد مدار BROWN-OUT ، این بیت بایستی برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.(جدول ۳-۳۷)

SUT0 ، SUT1 : برای انتخاب زمان START -UP بکار برده می‌شوند. عملکرد این دو بیت در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL3 .. 0 :عملکرد این بیت‌ها در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است.

۳-۸ خصوصیات Atmega161L , Atmega161

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 8MIPS در فرکانس 8MHZ
- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 16K بایت حافظه FLASH داخلی قابل برنامه‌ریزی

    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - 1024 بایت حافظه داخلی SRAM
  - قابلیت آدرس‌دهی 64K بایت حافظه خارجی
  - 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن

— قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

• **خصوصیات جانبی**

— دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و PWM

— یک تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE , CAPTURE و دو خروجی PWM 8 ، 9 یا 10 بیتی

— یک مقایسه‌کننده آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE

— دارای RTC (REAL –TIME CLOCK) بااسیلاتور مجزا

— دو UART سریال قابل برنامه‌ریزی

• **خصوصیات ویژه میکروکنترلر**

— POWER – ON RESET CIRCUIT.

— دارای 3 حالت SLEEP (POWER – DOWN ، POWER – SAVE ، IDLE)

— منابع وقفه (INTERRUPT) داخلی و خارجی

— عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• **توان مصرفی در 4MHZ ، 3V ، 25^0C**

— حالت فعال 3.0mA (ACTIVE MODE)

— درحالت بی‌کاری 1.2 mA (IDLE MODE)

— در حالت POWER – DOWN : < 1μA

• **ولتاژهای عملیاتی (کاری)**

— 2.7V تا 5.5 برای(Atmega161L)

— 4V تا 5.5V برای (Atmega161)

• **فرکانسهای کاری**

— 0MHZ تا 4MHZ برای (Atmega161L)

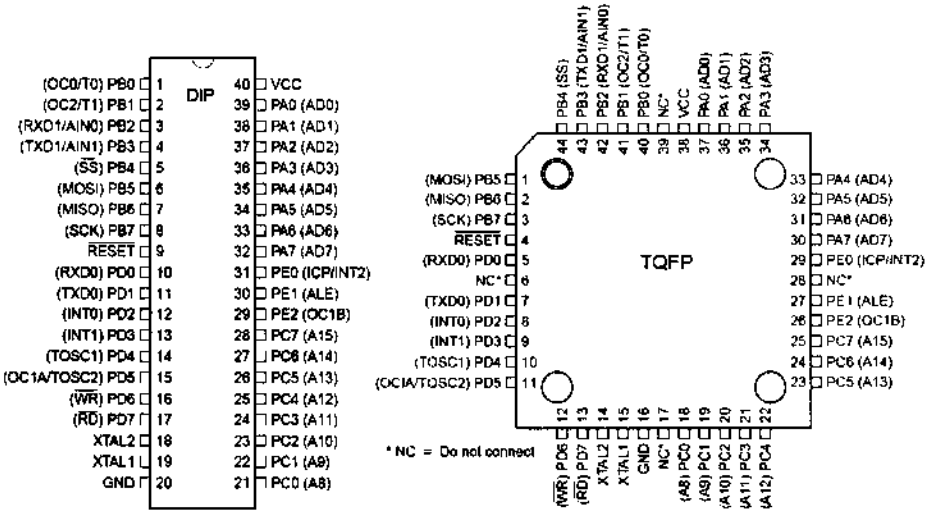
— 0MHZ تا 8MHZ برای (Atmega161)

• **خطوط I/O و انواع بسته‌بندی**

— 35 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی

— 40 پایه PDIP و 44 پایه TQFP

• ترکیب پایه‌ها



فیوز بیت‌های ATMEGA161

ATMEGA161 دارای 6 فیوز بیت است که در جدول ۳-۳۸ نشان داده شده‌اند. منطق 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت می‌باشد.

جدول ۳-۳۸ فیوزبیتهای میکرو ATMEGA161

FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BOOTRST	5	SELECT RESET VECTOR	1(UNPROGRAMMED)
SPIEN	4	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
SUT	3	SELECT START-UP TIME	1(UNPROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود. این بیت در مُد برنامه‌ریزی سریال قابل دسترس نمی‌باشد.

BOOTRST : بیتی برای انتخاب بردار ری‌ست BOOT که در حالت پیش‌فرض برنامه‌ریزی نشده است. زمانی که برنامه‌ریزی شود اجرای برنامه پس از ری‌ست از آدرس BOOT که $1E00 است آغاز می‌شود در غیر اینصورت یعنی زمانی که برنامه‌ریزی نشده باشد آدرس بردار ری‌ست $0000 و برنامه از ابتدای حافظه FLASH شروع به اجرا شدن می‌کند.

SUT : زمان START -UP توسط این بیت طبق جدول ۳-۳۹ تغییر داده می‌شود و در حالت پیش‌فرض برنامه‌ریزی نشده (1) است.

CKSEL0..2 : کلاک سیستم توسط این سه بیت طبق جدول ۳-۳۹ تعیین می‌شود که پیش‌فرض CKSEL0..2 = 010 هستند.

جدول ۳-۳۹ انتخاب کلاک سیستم و زمان START-UP برای میکرو

CKSEL2..0	START-UP TIME VCC=2.7V ,SUT UNPROGRAMMED	START-UP TIME VCC=4.0V SUT PROGRAMMED	RECOMMENDED USAGE
000	4.2ms+6 CK	5.8ms +6 CK	EXTERNAL CLOCK , FAST RISING POWER
001	6us + 6CK	10us + 6CK	EXTERNAL CLOCK
010	67ms + 16K CK	92ms + 16K CK	CRYSTAL OSCILLATOR , SLOWLY RISING POWER
011	4.2ms + 16K CK	5.8ms + 16K CK	CRYSTAL OSCILLATOR , FAST RISING POWER
100	30us + 16K CK	10us + 16K CK	CRYSTAL OSCILLATOR
101	67ms + 1K CK	92ms + 1K CK	CERAMICRESONATOR/EXTERNAL CLOCK , SLOWLY RISING POWER
110	4.2ms + 1K CK	5.8ms + 1K CK	CRYSTAL OSCILLATOR , FAST RISING POWER
111	30us + 1K CK	10us + 1K CK	CERAMIC RESONATOR

۳-۹ خصوصیات Atmega162V , Atmega162

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 131 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 16MIPS در فرکانس 16MHZ
- **حافظه ، برنامه و دادهٔ غیرفرار**
  - 16K بایت حافظه FLASH داخلی قابل برنامه‌ریزی
    پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - 1024 بایت حافظه داخلی SRAM
  - 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

• قابلیت ارتباط JTAG (IEEE Std.)

— برنامه‌ریزی برنامهٔ FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG

• خصوصیات جانبی

— دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و مُد COMPARE

— دو تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE ، CAPTURE

— 6 کانال PWM

— یک مقایسه‌کننده آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE

— دارای RTC (REAL -TIME CLOCK) بااسیلاتور مجزا

— دوUSART سریال قابل برنامه‌ریزی

• خصوصیات ویژه میکروکنترلر

— POWER - ON RESET CIRCUIT و مدار BROWN-OUT قابل برنامه‌ریزی

— دارای اسیلاتور RC داخلی کالیبره شده

— دارای 5 حالت SLEEP (POWER - DOWN ، POWER - SAVE ، IDLE ، STANDBY و EXTENDED STANDBY)

— منابع وقفه (INTERRUPT) داخلی و خارجی

— عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• ولتاژهای عملیاتی (کاری)

— 1.8V تا 5.5 برای(Atmega162V)

— 2.7V تا 5.5V برای (Atmega162)

• فرکانسهای کاری

— 0MHZ تا 8MHZ برای (Atmega162V)

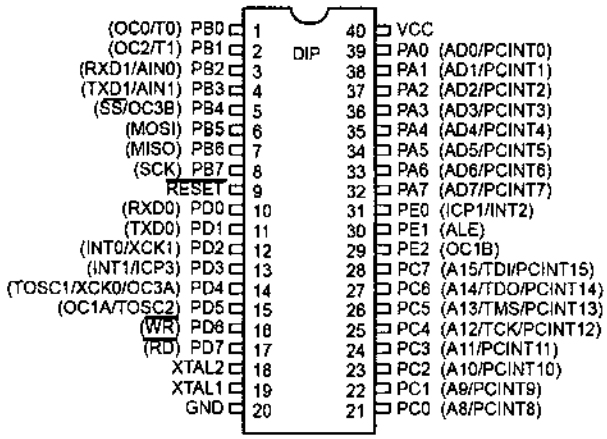
— 0MHZ تا 16MHZ برای (Atmega162)

• خطوط I/O و انواع بسته‌بندی

— 35 خط ورودی/خروجی (I/O) قابل برنامه‌ریزی

— 40 پایه PDIP ، 44 پایه TQFP و 44 پایه MLF

• ترکیب پایه‌ها



فیوز بیت‌های ATMEGA162

ATMEGA162 دارای سه بایت فیوز بیت طبق جدول‌های ۳-۴۰ و۳-۴۱ و۳-۴۲ می‌باشد .

جدول ۳-۴۰ بایت پیشرفته فیوزبیتهای میکرو

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1
-	6	-	1
-	5	-	1
M161C	4	ATMEGA161 COMPATIBILITY MODE	1(UNPROGRAMMED)
BODLEVEL2	3	BROWN-OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL1	2	BROWN-OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL0	1	BROWN-OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
-	0	-	1(UNPROGRAMMED)

جدول ۳-۴۱ بایت پرارزش فیوز بیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
WDTON	4	WATCH DOG TIMER ON	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول۳-۴۲ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
CKDIV8	7	DIVIDE CLOCK BY 8	0(PROGRAMMED)
CKOUT	6	CLOCK OUTPUT	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

جدول ۳-۴۳ انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

BODLEVEL FUSES 2 , 1 , 0	VBOT(VBOT- ,VBOT+)
111	BOD DISABLE
110	1.8v ± 25mv
101	2.7v ± 25mv
100	4.3v ± 25mv
011	2.3v ± 25mv
010	RESERVED
001	
000	

M161C : دو میکرو ATMEGA162 و ATMEGA161 بسیار با هم مطابقت دارند ولی برای رفع اندک اختلاف موجود، می‌توان با برنامه‌ریزی این بیت به طور 100% این دو میکرو را با هم مطابقت داد و در مدار به جای ATMEGA161 از ATMEGA162 استفاده نمود.

BODLEVEL0,1,2 : میکرو ATMEGA162 دارای مدار BROWN-OUT است که ولتاژ تریگ آن توسط فیوز بیت‌های BODLEVEL طبق جدول ۳-۴۳ قابل تنظیم می‌باشد.

OCDEN : زمانی که این بیت به همراه بیت JTAGEN برنامه‌ریزی شده باشد و بیت‌های قفل برنامه‌ریزی نشده باشند، ON CHIP DEBUG فعال است. برنامه‌ریزی شدن این بیت به قسمتهای از میکرو امکان می‌دهد که در مُدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می شود. این بیت به صورت پیش فرض برنامه‌ریزی نشده (1) است .

JTAGEN : بیتی برای فعال‌سازی برنامه‌ریزی میکرو از طریق استاندارد ارتباطی IEEE (JTAG) که در حالت پیش‌فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

WDTON : در حالت پیش‌فرض WATCHDOG غیر فعال و کاربر بایستی نرم‌افزاری آن را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن است.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM ری‌ست می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

BOOTSZ0,1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۴۴ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

جدول ۳-۴۴ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	2	0x0000 - 0x1F7F	0x1F80 - x1FFF	0x1F80
1	0	256 words	4	0x0000 - 0x1EFF	0x1F00 - x1FFF	0x1F00
0	1	512 words	8	0x0000 - 0x1DFF	0x1E00 - x1FFF	0x1E00
0	0	1024words	16	0x0000 - 0x1BFF	0x1C00 -x1FFF	0x1C00

BOOTRST : انتخاب بردار ری‌ست BOOT است که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست $0000 است و در صورت برنامه‌ریزی آدرس بردار ری‌ست به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

CKDIV8 : در حالت پیش‌فرض این فیوز برنامه‌ریزی شده است و کلاک سیستم بر 8 و زمانی که برنامه‌ریزی نشده باشد کلاک سیستم بر یک تقسیم می‌شود.

CKOUT : در صورت برنامه‌ریزی کردن این فیوز بیت کلاک سیستم در تمام مُدها ، در پایه خروجی PORTB.0 ایجاد می‌شود که برای راه‌اندازی مدارات خارجی مناسب می‌باشد.

SUT0 ، **SUT1** : بیت‌هایی برای انتخاب زمان START-UP هستند. عملکرد این دو بیت در بخش کلاک سیستم (2) در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL0 ... CKSEL3 : عملکرد این بیت‌ها در بخش ۳- ۱۹ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض INTERNAL RC OSCILLATOR @ 8MHZ است.

۳-۱۰ خصوصیات Atmega16 , Atmega16L

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 131 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی

— سرعتی تا 16 MIPS در فرکانس 16 MHZ

• **حافظه ، برنامه و دادهٔ غیرفرّار**

— 16K بایت حافظه FLASH داخلی قابل برنامه‌ریزی

پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن (WRITE / ERASE)

— 1024 بایت حافظه داخلی SRAM

— 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)

— قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM

• **قابلیت ارتباط JTAG (IEEE Std.)**

— برنامه‌ریزی برنامهٔ FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG

• **خصوصیات جانبی**

— دو تایمر – کانتر 8 بیتی با PRESCALER مجزا و مُد COMPARE

— یک تایمر – کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE

— 4 کانال PWM

— 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی

8 کانال SINGLE-ENDED

دارای 7 کانال تفاضلی در بسته‌بندی TQFP

دارای دو کانال تفاضلی با کنترل گین 1x ، 10x و200x

— یک مقایسه‌کننده آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— قابلیت ارتباط با پروتکل سریال دوسیمه (TWO – WIRE)

— قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE

— USART سریال قابل برنامه‌ریزی

• **خصوصیات ویژه میکروکنترلر**

— POWER – ON RESET CIRCUIT و BROWN-OUT قابل برنامه‌ریزی

— دارای اسیلاتور RC داخلی کالیبره شده

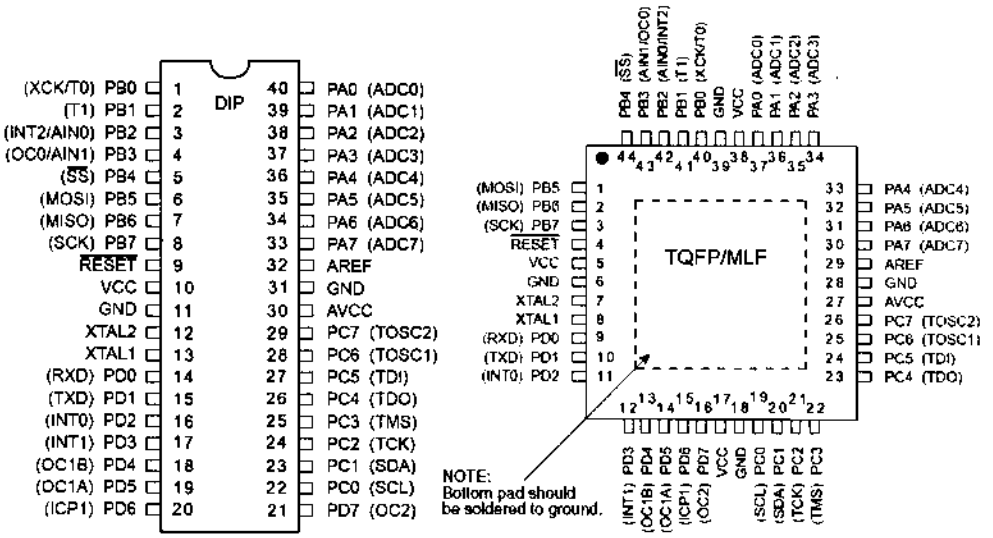
— دارای 6 حالت SLEEP (POWER – DOWN ، IDLE ، POWER – SAVE ، STANDBY ، EXTENDED STANDBY و ADC NOISE REDUCTION)

— منابع وقفه (INTERRUPT) داخلی و خارجی

— عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

- **توان مصرفی در 1MHZ ، 3V ، 25^0C برای ATMEGA16L**
  - حالت فعال 1.1mA (ACTIVE MODE)
  - درحالت بی‌کاری 0.35mA (IDLE MODE)
  - در حالت POWER – DOWN : < 1μA
- **ولتاژهای عملیاتی (کاری)**
  - 2.7V تا 5.5V برای (Atmega16L)
  - 4.5V تا 5.5V برای (Atmega16)
- **فرکانسهای کاری**
  - 0MHZ تا 8MHZ برای (Atmega16L)
  - 0MHZ تا 16MHZ برای (Atmega16)
- **خطوط I/O و انواع بسته‌بندی**
  - 32 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 40 پایه PDIP ، 44 پایه TQFP و 44 پایه MLF
- **ترکیب پایه‌ها**



فیوز بیت‌های ATMEGA16

ATMEGA16 دارای دو بایت فیوز بیت طبق جدول‌های ۳-۴۵ و ۳-۴۶ می‌باشد .

فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند ولی می‌توانند با برنامه‌ریزی بیت LB1 قفل شوند. منطق 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

جدول ۳-۴۵ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
CKOPT	4	OSCILLATOR OPTIONS	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۴۶ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

OCDEN : در صورتی که بیت‌های قفل برنامه‌ریزی نشده باشند برنامه‌ریزی این بیت به همراه بیت JTAGEN باعث می‌شود که سیستم ON CHIP DEBUG فعال شود. این بیت به صورت پیش فرض برنامه‌ریزی نشده (1) است.

JTAGEN : بیتی برای فعال‌سازی برنامه‌ریزی میکرو از طریق استاندارد ارتباطی IEEE (JTAG) که در حالت پیش‌فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

CKOPT : انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت به بیت‌های CKSEL بستگی دارد که در بخش کلاک سیستم (1) در انتهای همین فصل آمده است.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM پاک می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

BOOTSZ0..1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۴۷ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

جدول ۳-۴۷ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	2	0x0000 - 0x1F7F	0x1F80 -1FFF	0x1F80
1	0	256 words	4	0x0000 - 0x1EFF	0x1F00 -1FFF	0x1F00
0	1	512 words	8	0x0000 - x1DFF	0x1E00-x1FFF	0x1E00
0	0	1024 words	16	0x0000 - 0x1BFF	0x1C00-x1FFF	0x1C00

جدول ۳-۴۸ انتخاب آدرس بردار ری‌ست توسط فیوز بیت BOOTRST

BOOTRST	RESET ADDRESS
1(UNPROGRAMMED)	RESET VECTOR = APPLICATION RESET (ADDRESS $0000)
0(PROGRAMMED)	RESET VECTOR = BOOT LOADER RESET

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده (پیش‌فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و میکرو را ری‌ست می‌کند.

BODEN : برای فعال کردن عملکرد مدار BROWN – OUT این بیت بایستی برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.(جدول ۳-۴۹)

SUT0 ، SUT1 : عملکرد این دو بیت برای انتخاب زمان START–UP در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL0 ... CKSEL3 : عملکرد این بیت‌ها در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض INTERNAL RC OSCILLATOR @ 1MHZ است.

جدول ۳-۴۹ انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

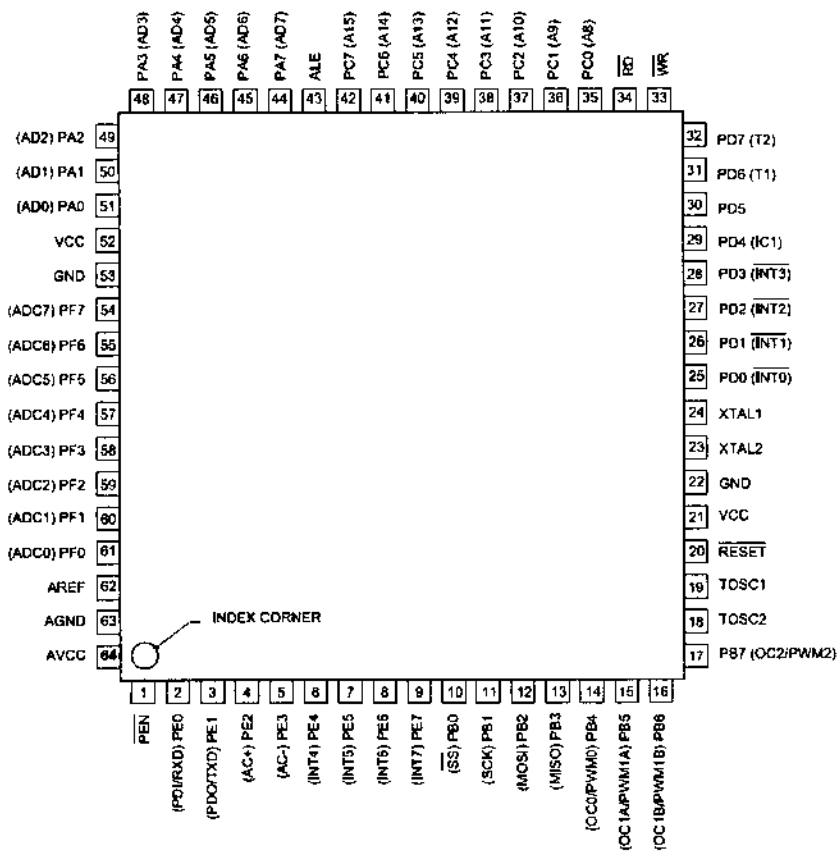
BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

۱۱-۳ خصوصیات Atmega103L , Atmega103

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 121 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 6 MIPS در فرکانس 6 MHZ
- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 128K بایت حافظه FLASH داخلی قابل برنامه‌ریزی
    پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 4K بایت حافظه داخلی SRAM
  - 4K بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و دارای PWM
  - یک تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE و و دو خروجی PWM 8 ، 9 یا 10 بیتی
  - 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
  - یک مقایسه‌کننده آنالوگ داخلی.
  - دارای RTC (REAL -TIME CLOCK) بااسیلاتور مجزا
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING)
  - قابلیت ارتباط سریال SPI به صورت MASTER یاSLAVE
  - UART سریال قابل برنامه‌ریزی
- **خصوصیات ویژه میکروکنترلر**
  - انتخاب نرم‌افزاری فرکانس کلاک
  - دارای 3 حالت SLEEP (POWER - DOWN ، POWER - SAVE ، IDLE)
  - منابع وقفه (INTERRUPT) داخلی و خارجی
  - دارای اسیلاتور RC داخلی کالیبره شده
  - عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

- **توان مصرفی در 4MHZ ، 3V ، 25^0C**
  - — حالت فعال 5.5mA (ACTIVE MODE)
  - — درحالت بی‌کاری 1.6mA (IDLE MODE)
  - — در حالت POWER – DOWN : < 1μA
- **ولتاژهای عملیاتی (کاری)**
  - — 2.7V تا 3.6V برای (Atmega103L)
  - — 4.0V تا 5.5V برای (Atmega103)
- **فرکانسهای کاری**
  - — 0MHZ تا 4MHZ برای (Atmega103L)
  - — 0MHZ تا 6MHZ برای (Atmega103)
- **خطوط I/O و انواع بسته‌بندی**
  - — 32 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - — 64 پایه TQFP



فیوز بیت‌های ATMEGA103

ATMEGA103 دارای 4 عدد فیوز بیت طبق جدول ۳-۵۰ است.

جدول ۳-۵۰ فیوزبیتهای میکرو ATMEGA103

FUSE BITS	BIT NO.	DESCRIPTION	DEFAULT VALUE
SPIEN	3	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
EESAVE	2	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
SUT1	1	START UP TIME SELECT	1(UNPROGRAMMED)
SUT0	0	START UP TIME SELECT	1(UNPROGRAMMED)

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM ری‌ست می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

SUT1 , SUT0 : این دو بیت طبق جدول ۳-۵۱ مشخص کننده زمان START-UP هستند.

جدول ۳-۵۱ انتخاب زمان RESET DELAY به ازاء VCC≈5V

PARAMETER	CONDITION SUT1,SUT0	MIN	TYP	MAX	UNITS
RESET DELAY TIME-OUT PERIOD	00		5		CPU CYCLES
	01	0.4	0.5	0.6	mS
	10	3.2	4.0	4.8	
	11	12.8	16.0	19.2	

۳-۱۲ خصوصیات Atmega169 , Atmega169L , Atmega169V

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 16 MIPS در فرکانس 16 MHZ
- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 16K بایت حافظه FLASH داخلی قابل برنامه‌ریزی

پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن(WRITE / ERASE)

— 1024 بایت حافظه داخلی SRAM

— 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)

— قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM.

• **قابلیت ارتباط JTAG (IEEE Std.)**

— برنامه‌ریزی برنامهٔ FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG

• **خصوصیات جانبی**

— دارای راه انداز 4x25 SEGMENT LCD

— دو تایمر - کانتر 8 بیتی با PRESCALER مجزا و مُد COMPARE

— یک تایمر - کانتر16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE

— 4 کانال PWM

— 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی

— یک مقایسه‌کننده آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE

— قابلیت ارتباط سریال USI (UNIVERSAL SERIAL INTERFACE)

— USART سریال قابل برنامه‌ریزی

— دارای RTC (REAL -TIME CLOCK) بااسیلاتور مجزا

— وقفه در اثرتغییر وضعیت پایه

• **خصوصیات ویژه میکروکنترلر**

— POWER - ON RESET CIRCUIT و BROWN-OUT قابل برنامه‌ریزی

— دارای اسیلاتور RC داخلی کالیبره شده

— دارای 5 حالتSLEEP (POWER - DOWN ، POWER - SAVE ، ADC NOISE REDUCTION ، STANDBY، IDLE)

— منابع وقفه (INTERRUPT) داخلی و خارجی

— عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• **توان مصرفی فوق العاده پایین**

— حالت فعال

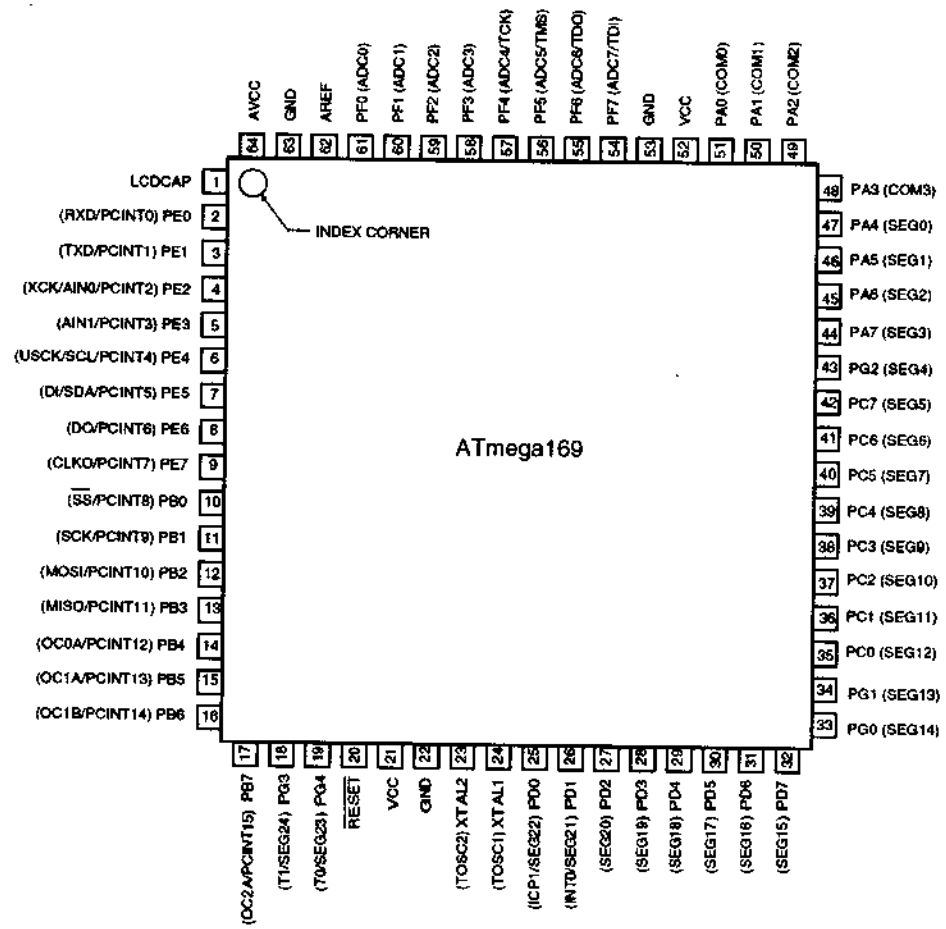
در 1MHZ و ولتاژ 1.8V : 400µA

در 32KHZ و ولتاژ 1.8V : 20µA (با اسیلاتور)

در 32KHZ و ولتاژ 1.8V: 40µA (با اسیلاتور و LCD)

— در حالت POWER – DOWN و ولتاژ 1.8V : 0.5µA

- **خطوط I/O و انواع بسته‌بندی**
  - 53 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
  - 64 پایه TQFP و MLF
- **ولتاژهای عملیاتی (کاری)**
  - 1.8 V تا 5.5 برای(Atmega169V)
  - 2.7V تا 5.5V برای (Atmega169L)
  - 4.5V تا 5.5V برای (Atmega169)
- **فرکانسهای کاری**
  - 0MHZ تا 1MHZ برای (Atmega169V)
  - 0MHZ تا 8MHZ برای (Atmega169L)
  - 0MHZ تا 16MHZ برای (Atmega169)
- **ترکیب پایه‌ها**



فیوز بیت‌های ATMEGA169

ATMEGA169 دارای سه بایت فیوز بیت طبق جدول‌های ۳-۵۲ و۳-۵۳ و۳-۵۴ می‌باشد .

جدول ۳-۵۲ بایت پیشرفته فیوزبیتهای میکرو

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	0	-	1(UNPROGRAMMED)
-	6	-	1(UNPROGRAMMED)
-	5	-	1(UNPROGRAMMED)
-	4	-	1(UNPROGRAMMED)
-	3	-	1(UNPROGRAMMED)
BODLEVEL2	2	BROWN-OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL1	1	BROWN-OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL0	0	BROWN-OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)

جدول ۳-۵۳ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
WDTON	4	WATCH DOG TIMER ON	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۵۴ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFA5ULT VALUE
CKDIV8	7	DIVIDE CLOCK BY 8	0(PROGRAMMED)
CKOUT	6	CLOCK OUTPUT	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

جدول۳-۵۵ انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

BODLEVEL FUSES 2 ,1, 0	VBOT(VBOT- ,VBOT+)
111	BOD DISABLE
110	1.8v ± 25mv
101	2.7v ± 25mv
100	4.3v ± 25mv
011	RESERVED
010	
001	
000	

BODLEVEL 2 , 1 , 0 : میکرو ATMEGA169 دارای مدار BROWN-OUT است که ولتاژ تریگ آن توسط فیوز بیت‌های BODLEVEL طبق جدول ۳-۵۵ قابل تنظیم می‌باشد.

OCDEN : در صورتی که بیت‌های قفل برنامه‌ریزی نشده باشند برنامه‌ریزی این بیت به همراه بیت JTAGEN باعث می‌شود که سیستم ON CHIP DEBUG فعال شود. برنامه‌ریزی شدن این بیت به قسمتهایی از میکرو امکان می‌دهد که در مُدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می شود. این بیت به صورت پیش فرض برنامه‌ریزی نشده (1) است.

JTAGEN : بیتی برای فعال‌سازی برنامه‌ریزی میکرو از طریق استاندارد ارتباطی IEEE (JTAG) که در حالت پیش‌فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

WDTON : در حالت پیش‌فرض WATCHDOG غیرفعال و کاربر بایستی نرم‌افزاری WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن است.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM ری‌ست می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

CKOPT : انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در بخش ۳-۱۹ آمده است.

BOOTSZ0..1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۵۶ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

جدول ۳-۵۶ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	2	0x0000 - 0x1F7F	0x1F80 - x1FFF	0x1F80
1	0	256 words	4	0x0000 - 0x1EFF	0x1F00 - x1FFF	0x1F00
0	1	512 words	8	0x0000 - 0x1DFF	0x1E00 - x1FFF	0x1E00
0	0	1024 words	16	0x0000 - 0x1BFF	0x1C00 -x1FFF	0x1C00

BOOTRST : بیتی جهت انتخاب بردار ری‌ست است که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست $0000 است و در صورت برنامه‌ریزی آدرس بردار ری‌ست به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

CKDIV8 : در حالتی که این فیوز بیت برنامه‌ریزی شده باشد کلاک سیستم بر 8 و زمانی که برنامه‌ریزی نشده باشد کلاک سیستم بر یک تقسیم می‌شود.

CKOUT : در صورت برنامه‌ریزی کردن این فیوز بیت کلاک سیستم در تمام مُدها ، در پایه خروجی PORTE.7 ایجاد می‌شود که برای راه‌اندازی مدارات خارجی مناسب می‌باشد.

SUT0,1 : عملکرد این دو بیت برای انتخاب زمان START-UP در بخش ۳-۱۹ توضیح داده شده است.

CKSEL0..3 : عملکرد این بیت‌ها در بخش ۳-۱۹ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض INTERNAL RC OSCILLATOR @ 8MHZ است.

۳-۱۳ خصوصیات Atmega64L , Atmega64

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - سرعتی تا 16 MIPS در فرکانس 16 MHZ
- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 64K بایت حافظه FLASH داخلی قابل برنامه‌ریزی

    پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - 4K بایت حافظه داخلی SRAM
  - قابلیت آدرس‌دهی 64K بایت حافظه خارجی
  - 2K بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - دو تایمر - کانتر 8 بیتی با PRESCALER مجزا و مُد COMPARE
  - دو تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE
  - 2 کانال PWM هشت بیتی
  - 6 کانال PWM از 1 تا 16 بیتی
  - 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی

    8 کانال SINGLE-ENDED

دارای 7 کانال تفاضلی

دارای دو کانال تفاضلی با کنترل گین 1x ، 10x و200x

— یک مقایسه‌کننده آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— قابلیت ارتباط با پروتکل سریال دوسیمه (TWO – WIRE)

— قابلیت ارتباط سریال SPI به صورتMASTER یا SLAVE

— دوUSART سریال قابل برنامه‌ریزی

— دارای RTC (REAL -TIME CLOCK) با اسیلاتور مجزا

• **خصوصیات ویژه میکروکنترلر**

— POWER – ON RESET CIRCUIT و BROWN-OUT قابل برنامه‌ریزی

— دارای اسیلاتور RC داخلی کالیبره شده

— دارای 6 حالت SLEEP (POWER – DOWN ، POWER – SAVE ، IDLE ، STANDBY ، EXTENDED STANDBY و NOISE ADC REDUCTION)

— منابع وقفه داخلی و خارجی

— انتخاب نرم‌افزاری فرکانس کلاک

— مطابقت با میکرو ATMEGA103 توسط انتخاب فیوز بیت

— غیرفعال کردن سراسری مقاومت‌های PULL-UP

— عملکرد کاملاً ثابت

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• **خطوط I/O و انواع بسته‌بندی**

— 53 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی

— 64 پایه در انواع TQFP و MLF (شبیه پایه های MEGA128)

• **ولتاژهای عملیاتی (کاری)**

— 2.7 V تا 5.5 برای(Atmega64L)

— 4.5V تا 5.5V برای (Atmega64)

• **فرکانسهای کاری**

— 0MHZ تا 8MHZ برای (Atmega64L)

— 0MHZ تا 16MHZ برای (Atmega64)

فیوز بیت‌های ATMEGA64

ATMEGA64 دارای سه بایت فیوز بیت طبق جدول‌های ۳-۵۷ ، ۳-۵۸ و۳-۵۹ می‌باشد .

جدول ۳-۵۷ بایت پیشرفته فیوزبیتهای میکرو

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1
-	6	-	1
-	5	-	1
-	4	-	1
-	3	-	1
-	2	-	1
M103C	1	MEGA103 COMPATIBILITY MODE	0(PROGRAMMED)
WDTON	0	WATCH DOG TIMER ON	1(UNPROGRAMMED)

جدول ۳-۵۸ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
CKOPT	4	OSCILLATOR OPTIONS	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۵۹ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

M103C : دو میکرو ATMEGA103 و ATMEGA64 بسیار با هم مطابقت دارند ولی برای رفع اندک اختلاف موجود ، می‌توان با برنامه‌ریزی این بیت به طور 100% این دو میکرو را با هم مطابقت داد و در مدار به جای ATMEGA103 از ATMEGA64 استفاده نمود.

WDTON : در حالت پیش‌فرض WATCHDOG غیر فعال وکاربر بایستی نرم‌افزاری WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن است.

OCDEN : در صورتی که بیت‌های قفل برنامه‌ریزی نشده باشند برنامه‌ریزی این بیت به همراه بیت JTAGEN باعث می‌شود که سیستم ON CHIP DEBUG فعال شود. برنامه‌ریزی شدن این بیت به قسمتهایی از میکرو امکان می‌دهد که در مُدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می شود. این بیت به صورت پیش فرض برنامه‌ریزی نشده (1) است.

JTAGEN : بیتی برای فعال‌سازی برنامه‌ریزی میکرو از طریق استاندارد ارتباطی IEEE (JTAG) که در حالت پیش‌فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید.

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

CKOPT : بیتی برای انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در بخش ۳-۱۸ در انتهای همین فصل آمده است.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM ری‌ست می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک‌شدن محفوظ می‌ماند.

BOOTSZ0,1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۶۰ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

جدول ۳-۶۰ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	512 words	2	0x0000 - 0x7DFF	0x7E00 –0x7FFF	0x7E00
1	0	1024 words	4	0x0000 - 0x7BFF	0x7C00 -0x7FFF	0x7C00
0	1	2048 words	8	0x0000 - 0x77FF	0x7800 - 0x7FFF	0x7800
0	0	4096 words	16	0x0000 - 0x6FFF	0x7000 - 0x7FFF	0x7000

BOOTRST : بیتی برای انتخاب بردار ری‌ست BOOT که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست $0000 است و در صورت برنامه‌ریزی آدرس بردار ری‌ست به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

BODLEVEL : زمانی که این بیت برنامه‌ریزی نشده باشد اگر ولتاژ پایه VCC از 2.7V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را ری‌ست می‌کند. زمانی که این بیت برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و میکرو را ری‌ست می‌کند.

BODEN : برای فعال کردن عملکرد BROWN - OUT این بیت بایستی برنامه‌ریزی شده باشد . این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

SUT1 ، SUT0 : انتخاب زمان START -UP که عملکرد این دو بیت در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL2 ، CKSEL1، CKSEL0 : عملکرد این بیت‌ها در بخش ۳-۱۸ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض INTERNAL RC OSCILLATOR @ 1MHZ است.

۳-۱۴ خصوصیات Atmega 48v/88v/168v , Atmega 48/88/168

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 131 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - عملکرد کاملاً ثابت
  - سرعتی تا 20 MIPS در فرکانس 20 MHZ
- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - 4/8/16K بایت حافظه FLASH داخلی قابل برنامه‌ریزی (Atmega48/88/168)

    پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن (WRITE / ERASE)
  - 4K بایت حافظه داخلی SRAM
  - قابلیت آدرس‌دهی 64K بایت حافظه خارجی
  - 256/512/512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

    پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - 512/1K/1K بایت حافظه SRAM داخلی
  - قفل برنامهٔ FLASH و حفاظت دادهٔ EEPROM
- **خصوصیات جانبی**
  - دو تایمر - کانتر 8 بیتی با PRESCALER مجزا و مُد COMPARE
  - دو تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE ، CAPTURE
  - 6 کانال PWM
  - 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی در دو نوع TQFP و MLF
  - 6 کانال مبدل آنالوگ به دیجیتال 10 بیتی در نوع PDIP

— یک مقایسه‌کننده آنالوگ داخلی
— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
— قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE
— USART سریال قابل برنامه‌ریزی
— دارای RTC (REAL –TIME COUNTER) با اسیلاتور مجزا

• **خصوصیات ویژه میکروکنترلر**

— POWER – ON RESET CIRCUIT و BROWN-OUT قابل برنامه‌ریزی
— دارای اسیلاتور کالیبره شده داخلی
— دارای 5 حالت (SLEEP ، POWER – DOWN ، IDLE ، POWER – SAVE ، STANDBY و NOISE ADC REDUCTION)
— منابع وقفه (INTERRUPT) داخلی و خارجی
— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• **توان مصرفی پایین**

— حالت فعال
در 1MHZ و ولتاژ 1.8V : 240UA
در 32KHZ و ولتاژ 1.8V : 15UA
— حالت POWER-DOWN
در ولتاژ 1.8V : 0.1UA

• **خطوط I/O و انواع بسته‌بندی**

— 23 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
— 28 پایه در نوع PDIP و32 پایه در انواع TQFP و MLF

• **ولتاژهای عملیاتی (کاری)**

— 1.8 V تا 5.5 برای(Atmega48v/88v/168v)
— 2.7V تا 5.5V برای (Atmega48/88/168)

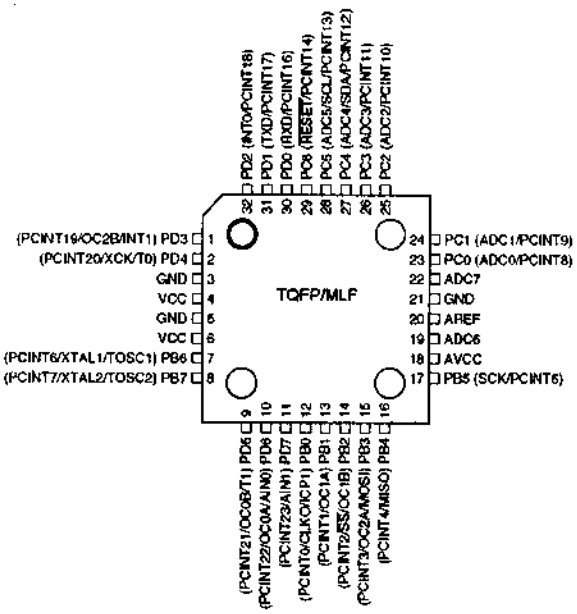
• **رنج حرارتی**

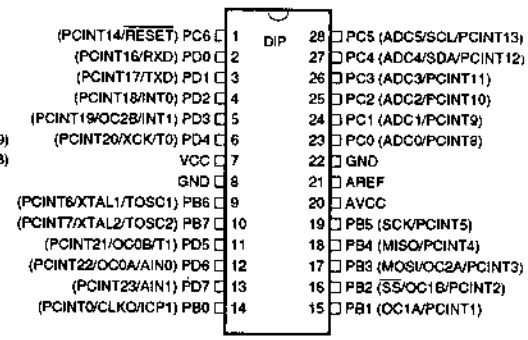
— -40^0C تا 85^0C

• **فرکانسهای کاری**

— ATmega48V/88V/168V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V
— 0MHZ تا 10MHZ برای (Atmega48/88/168)

• ترکیب پایه ها





فیوز بیت‌های ATMEGA 48/88/168

Atmega 48/88/168 دارای سه بایت فیوز بیت طبق جدول‌های ۳-۵۷، ۳-۵۸ و ۳-۵۹ می‌باشد .

جدول ۳-۵۷ بایت پیشرفته فیوزبیتهای میکرو mega48

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1
-	6	-	1
-	5	-	1
-	4	-	1
-	3	-	1
-	2	-	1
-	1	-	1
SELFPRGEN	0	SELF PROGRAMMING ENABLE	1(UNPROGRAMMED)

جدول ۳-۵۸ بایت پیشرفته فیوزبیتهای میکرو mega88/168

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1(UNPROGRAMMED)
-	6	-	1(UNPROGRAMMED)
-	5	-	1(UNPROGRAMMED)
-	4	-	1(UNPROGRAMMED)
-	3	-	1(UNPROGRAMMED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۵۹ بایت پر ارزش فیوز بیتهای میکروهای ATMEGA 48/88/168

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
DWEN	7	DEBUGWIRE ENABLE	1(UNPROGRAMMED)
SPIEN	6	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
EESAVE	5	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
WDTON	4	WATHDOG TIMER ALWAYS ON	1(UNPROGRAMMED)
BODLEVEL2	3	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL1	2	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL0	1	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
RSTDISBL	0	EXTERNAL RESET DESABLE	1(UNPROGRAMMED)

جدول ۳-۶۰ بایت کم ارزش فیوز بیتهای میکروهای ATMEGA 48/88/168

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
CKDIV8	7	DIVIDE CLICK BY 8	0(PROGRAMMED)
CKOUT	6	CLOCK OUTPUT	1(UNPROGRAMMED)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

SELFPRGEN : با برنامه‌ریزی این بیت می‌توان از دستور SPM در برنامه استفاده نمود.

BOOTSZ0..1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۶۱ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

BOOTRST : بیتی جهت انتخاب بردار ری‌ست است که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ری‌ست $0000 است و در صورت برنامه‌ریزی آدرس بردار ری‌ست به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

DWEN :در صورتی که این بیت برنامه‌ریزی نشده باشد، حافظه برنامه حتی اگر بیتهای قفل برنامه‌ریزی شده باشند می‌تواند توسط ارتباط DEBUGWIRE خوانده شود. بنابرین در صورتی که بخواهید برنامه به هیچ وجهی خوانده نشود این بیت را برنامه‌ریزی کنید.

جدول ۳-۶۱ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0.1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	2	0x0000 - 0x1F7F	0x1F80 - x1FFF	0x1F80
1	0	256 words	4	0x0000 - 0x1EFF	0x1F00 - x1FFF	0x1F00
0	1	512 words	8	0x0000 - 0x1DFF	0x1E00 - x1FFF	0x1E00
0	0	1024 words	16	0x0000 - 0x1BFF	0x1C00 -x1FFF	0x1C00

جدول ۳-۶۲ انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

BODLEVEL FUSES 2 , 1 , 0	VBOT(VBOT- ,VBOT+)
111	BOD DISABLE
110	1.8v ± 25mv
101	2.7v ± 25mv
100	4.3v ± 25mv
011	RESERVED
010	
001	
000	

SPIEN : در حالت پیش‌فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.

EESAVE : در حالت پیش‌فرض برنامه‌ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM ری‌ست می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می‌ماند.

WDTON : در حالت پیش‌فرض WATCHDOG غیرفعال و کاربر بایستی نرم‌افزاری WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن است.

BODLEVEL0,1,2 : این سه میکرو دارای مدار BROWN-OUT است که ولتاژ تریگ آن توسط فیوز بیت‌های BODLEVEL طبق جدول ۳-۶۲ قابل تنظیم می‌باشد.

RSTDISBL : در حالت پیش‌فرض PC6 پایه ری‌ست است. با برنامه‌ریزی این بیت، پایه PC6 به عنوان پایه I/O استفاده می‌شود.

CKDIV8 : در حالتی که این فیوز بیت برنامه‌ریزی شده باشد کلاک سیستم بر 8 و زمانی که برنامه‌ریزی نشده باشد کلاک سیستم بر یک تقسیم می‌شود.

CKOUT : در صورت برنامه‌ریزی کردن این فیوز بیت کلاک سیستم در تمام مُدها ، در پایه خروجی CLKO ایجاد می‌شود که برای راه‌اندازی مدارات خارجی مناسب می‌باشد.

SUT0,1 : عملکرد این دو بیت برای انتخاب زمان START-UP است.

CKSEL0..3 : عملکرد این بیت‌ها با توجه به جدول ۳-۶۳ است. مقدار پیش‌فرض INTERNAL RC OSC @ 8MHZ و با توجه به برنامه ریزی شدن بیت CKDIV8 ، کلاک سیستم برابر 1MHZ است.

جدول ۳-۶۳ مُد و بازه های فرکانس کاری

FREQUENCY MODE AND RANGE	CKSEL3...0
Low Power Crystal Oscillator(0.4-0.9MHZ)	1000-1010(C1,2= -)
Low Power Crystal Oscillator(0.9-3.0MHZ)	1011 - 1010(C1,2=12-22PF)
Low Power Crystal Oscillator(3.0-8.0MHZ)	1101 - 1100(C1,2=12-22PF)
Low Power Crystal Oscillator(8.0-16.0MHZ)	1111 - 1110(C1,2=12-22PF)
Full Swing Crystal Oscillator(0.4-20MHZ)	0111 - 0110(C1,2=12-22PF)
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128 kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010(8MHZ)
External Clock(0-100MHZ)	0000
Reserved	0001

۳-۱۵ خصوصیات Atmega 325/3250/645/6450

- **از معماری AVR RISC استفاده می کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - عملکرد کاملاً ثابت
  - سرعتی تا 16 MIPS در فرکانس 16 MHZ
- **حافظه ، برنامه و دادهٔ غیرفرّار**
  - حافظه FLASH داخلی قابل برنامه‌ریزی
    - 32k بایت (Atmega325/3250)
    - 64k بایت (Atmega645/6450)
    - پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - حافظه داخلی SRAM
    - 2k بایت (Atmega325/3250)
    - 4k بایت (Atmega645/6450)
  - حافظه EEPROM داخلی قابل برنامه‌ریزی
    - 1k بایت (Atmega325/3250)
    - 2k بایت (Atmega645/6450)
    - پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - قفل برنامه برای حفاظت نرم افزار

• **قابلیت ارتباط JTAG (IEEE Std.)**

— برنامه‌ریزی برنامهٔ FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG

• **خصوصیات جانبی**

— دو تایمر – کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و مُد COMPARE

— یک تایمر – کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE

— 4 کانال PWM

— 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی

— یک مقایسه‌کننده آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE

— قابلیت ارتباط سریال USI (UNIVERSAL SERIAL INTERFACE)

— USART سریال قابل برنامه‌ریزی

— دارای RTC (REAL –TIME COUNTER) بااسیلاتور مجزا

— وقفه در اثرتغییر وضعیت پایه

• **خصوصیات ویژه میکروکنترلر**

— POWER – ON RESET CIRCUIT و BROWN-OUT قابل برنامه‌ریزی

— دارای اسیلاتور داخلی کالیبره شده

— دارای 5 حالتSLEEP (POWER – DOWN ، POWER – SAVE ، ADC NOISE REDUCTION ، STANDBY، IDLE)

— منابع وقفه (INTERRUPT) داخلی و خارجی

• **توان مصرفی فوق العاده پایین**

— حالت فعال

در 1MHZ و ولتاژ 1.8V : 350µA

در 32KHZ و ولتاژ 1.8V : 20µA (با اسیلاتور)

— در حالت POWER – DOWN و ولتاژ 1.8V : 100nA

• **خطوط I/O و انواع بسته‌بندی**

— 53/68 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی

— 64 پایه TQFP و MLF و100 پایه TQFP

• **ولتاژها و فرکانسهای عملیاتی (کاری)**

— Atmega325v/3250v/645v/6450v: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V

— Atmega325/3250/645/6450: 0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V

فیوز بیتهای ATMEGA325/3250/645/6450

جدول ۳-۶۴ بایت پیشرفته فیوزبیتهای میکرو

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1(UNPROGRAMMED)
-	6	-	1(UNPROGRAMMED)
-	5	-	1(UNPROGRAMMED)
-	4	-	1(UNPROGRAMMED)
-	3	-	1(UNPROGRAMMED)
BODLEVEL 1	2	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL 0	1	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
RSTDISBL	0	EXTERNAL RESET DESABLE	1(UNPROGRAMMED)

جدول ۳ -۶۵ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD (ON CHIP DEBUG ENABLE)	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
WDTON	4	WATCHDOG TIMER ALWAYSE ON	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET ' ECTOR	1(UNPROGRAMMED)

جدول ۳-۶۶ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
CKDIV8	7	DIVIDE CLICK BY 8	0(PROGRAMMED)
CKOUT	6	CLOCK OUTPUT	1(UNPROGRAMMED)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

BODLEVEL0,1,2 : این سه میکرو دارای مدار BROWN-OUT است که ولتاژ تریگ آن توسط فیوز بیت‌های BODLEVEL طبق جدول ۳-۶۷ قابل تنظیم می‌باشد.

BOOTSZ0..1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۶۸ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

BOOTRST : بیتی جهت انتخاب بردار ری‌ست است که در حالت پیش‌فرض برنامه‌ریزی نشده است.

CKOUT : در صورت برنامه‌ریزی کردن این فیوز بیت کلاک سیستم در تمام مُدها ، در پایه خروجی CLKO ایجاد می‌شود که برای راه‌اندازی مدارات خارجی مناسب می‌باشد.

SUT0,1 : عملکرد این دو بیت برای انتخاب زمان START-UP است.

CKSEL0..3 : عملکرد این بیت‌ها با توجه به جدول ۳-۶۹ است. مقدار پیش‌فرض INTERNAL RC OSC @ 8MHZ و با توجه به برنامه ریزی شدن بیت CKDIV8 ، کلاک سیستم برابر 1MHZ است.

جدول ۳-۶۷ انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

BODLEVEL FUSES 2... 0	VBOT(VBOT- ,VBOT+)
11	BOD DISABLE
10	1.8v ± 25mv
01	2.7v ± 25mv
00	4.3v ± 25mv

جدول ۳-۶۸ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	256/512 words	4	0x00000x3EFF/ 0x0000 - 0x7DFF	0x3F00-0x3FFF/ 0x7E00 –0x7FFF	0x3F00/ 0x7E00
1	0	512/1024 words	8	0x00000x3DFF/ 0x0000 - 0x7BFF	0x3E00-0x3FFF/ 0x7C00 –0x7FFF	0x3E00/ 0x7C00
0	1	1024/2048 words	16	0x00000x3BFF/ 0x0000 - 0x77FF	0x3C00-0x3FFF/ 0x7800 –0x7FFF	0x3C00/ 0x7800
0	0	2048/4096 words	32	0x00000x37FF/ 0x0000 - 0x6FFF	0x3800-0x3FFF/ 0x7000 –0x7FFF	0x3800/ 0x7000

جدول ۳-۶۹ مُد و بازه های فرکانس کاری

FREQUENCY MODE AND RANGE	CKSEL3...0
Low Power Crystal Oscillator(0.4-0.9MHZ)	1000 (C1,2= -)
Low Power Crystal Oscillator(0.9-3.0MHZ)	1011 - 1010(C1,2=12-22PF)
Low Power Crystal Oscillator(3.0-8.0MHZ)	1101 - 1100(C1,2=12-22PF)
Low Power Crystal Oscillator(8.0-16.0MHZ)	1111 - 1110(C1,2=12-22PF)
Low Frequency Crystal Oscillator	0111 - 0110
Calibrated Internal RC Oscillator	0010(7.3-8.1MHZ)
External Clock(0-16MHZ)	0000
Reserved	0011, 0001, 0101, 0100

۳-۱۶ خصوصیات Atmega 329/3290/649/6490

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - عملکرد کاملاً ثابت
  - سرعتی تا 16 MIPS در فرکانس 16 MHZ
- **حافظه ، برنامه و دادهٔ غیرفرار**
  - حافظه FLASH داخلی قابل برنامه‌ریزی
    - 32k بایت (Atmega329/3290)
    - 64k بایت (Atmega649/6490)
    - پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - حافظه داخلی SRAM
    - 2k بایت (Atmega329/3290)
    - 4k بایت (Atmega649/6490)
  - حافظه EEPROM داخلی قابل برنامه‌ریزی
    - 1k بایت (Atmega329/3290)
    - 2k بایت (Atmega649/6490)
    - پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)
  - قفل برنامه برای حفاظت نرم افزار
- **قابلیت ارتباط JTAG (IEEE Std.)**
  - برنامه‌ریزی برنامهٔ FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG
- **خصوصیات جانبی**
  - دارای راه انداز 25× 4 SEGMENT LCD برای (Atmega329/649)
  - دارای راه انداز 40× 4 SEGMENT LCD برای (Atmega3290/6490)
  - دو تایمر - کانتر 8 بیتی با PRESCALER مجزا و مُد COMPARE
  - یک تایمر - کانتر 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE
  - 4 کانال PWM
  - 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
  - یک مقایسه‌کننده آنالوگ داخلی
  - WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی
  - قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE

— قابلیت ارتباط سریال USI (UNIVERSAL SERIAL INTERFACE)
— USART سریال قابل برنامه‌ریزی
— دارای RTC (REAL –TIME COUNTER) بااسیلاتور مجزا
— وقفه در اثرتغییر وضعیت پایه

• **خصوصیات ویژه میکروکنترلر**

— POWER – ON RESET CIRCUIT و BROWN-OUT قابل برنامه‌ریزی
— دارای اسیلاتور داخلی کالیبره شده
— دارای 5 حالتSLEEP (POWER – DOWN ، POWER – SAVE ، ADC NOISE REDUCTION ، IDLE ، STANDBY)
— منابع وقفه (INTERRUPT) داخلی و خارجی

• **توان مصرفی فوق العاده پایین**

— حالت فعال
در 1MHZ و ولتاژ 1.8V : 350µA
در 32KHZ و ولتاژ 1.8V : 20µA (با اسیلاتور)
در 32KHZ و ولتاژ 1.8V : 40µA (با اسیلاتورو LCD)
— در حالت POWER – DOWN و ولتاژ 1.8V : 100uA

• **خطوط I/O و انواع بسته‌بندی**

— 53/68 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی
— 64 پایه TQFP و MLF و100 پایه TQFP

• **ولتاژها و فرکانسهای عملیاتی (کاری)**

— Atmega329v/3290v/649v/6490v: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
— Atmega329/3290/649/6490: 0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V

فیوز بیت های Atmega 329/3290/649/6490

جدول ۳–۷۰ بایت پیشرفته فیوزبیتهای میکرو

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1(UNPROGRAMMED)
-	6	-	1(UNPROGRAMMED)
-	5	-	1(UNPROGRAMMED)
-	4	-	1(UNPROGRAMMED)
-	3	-	1(UNPROGRAMMED)
BODLEVEL1	2	BROWN OUT DETECTOR TRIGGER	1(UNPROGRAMMED)
BODLEVEL0	1	BROWN OUT DETECTOR TRIGGER	1(UNPROGRAMMED)
RSTDISBL	0	EXTERNAL RESET DESABLE	1(UNPROGRAMMED)

جدول۳ -۷۱ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
WDTON	4	WATCHDOG TIMER ALWAYSE ON	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۷۲ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
CKDIV8	7	DIVIDE CLICK BY 8	0(PROGRAMMED)
CKOUT	6	CLOCK OUTPUT	1(UNPROGRAMMED)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

BODLEVEL0,1,2 : این سه میکرو دارای مدار BROWN-OUT است که ولتاژ تریگ آن توسط فیوز بیت‌های BODLEVEL طبق جدول ۳-۷۳ قابل تنظیم می‌باشد.

BOOTSZ0..1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۷۴ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

BOOTRST : بیتی جهت انتخاب بردار ری‌ست است که در حالت پیش‌فرض برنامه‌ریزی نشده است.

CKOUT : در صورت برنامه‌ریزی کردن این فیوز بیت کلاک سیستم در تمام مُدها ، در پایه خروجی CLKO ایجاد می‌شود که برای راه‌اندازی مدارات خارجی مناسب می‌باشد.

SUT0,1 : عملکرد این دو بیت برای انتخاب زمان START-UP است.

CKSEL0..3 : عملکرد این بیت‌ها با توجه به جدول ۳-۷۵ است. مقدار پیش‌فرض INTERNAL RC OSC @ 8MHZ و با توجه به برنامه ریزی شدن بیت CKDIV8 ، کلاک سیستم برابر 1MHZ است.

جدول۳-۷۳ انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

BODLEVEL FUSES 2... 0	VBOT(VBOT- ,VBOT+)
11	BOD DISABLE
10	1.8v ± 25mv
01	2.7v ± 25mv
00	4.3v ± 25mv

جدول ۳-۷۴ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	256/512 words	4	0x00000x3EFF/ 0x0000 - 0x7DFF	0x3F00-0x3FFF/ 0x7E00 –0x7FFF	0x3F00/ 0x7E00
1	0	512/1024 words	8	0x00000x3DFF/ 0x0000 - 0x7BFF	0x3E00-0x3FFF/ 0x7C00 –0x7FFF	0x3E00/ 0x7C00
0	1	1024/2048 words	16	0x00000x3BFF/ 0x0000 - 0x77FF	0x3C00-0x3FFF/ 0x7800 –0x7FFF	0x3C00/ 0x7800
0	0	2048/4096 words	32	0x00000x37FF/ 0x0000 - 0x6FFF	0x3800-0x3FFF/ 0x7000 –0x7FFF	0x3800/ 0x7000

جدول ۳-۷۵ مُد و بازه های فرکانس کاری

FREQUENCY MODE AND RANGE	CKSEL3...0
Low Power Crystal Oscillator(0.4-0.9MHZ)	1000 (C1,2= -)
Low Power Crystal Oscillator(0.9-3.0MHZ)	1011 - 1010(C1,2=12-22PF)
Low Power Crystal Oscillator(3.0-8.0MHZ)	1101 - 1100(C1,2=12-22PF)
Low Power Crystal Oscillator(8.0-16.0MHZ)	1111 - 1110(C1,2=12-22PF)
Low Frequency Crystal Oscillator	0111 - 0110
Calibrated Internal RC Oscillator	0010(7.3-8.1MHZ)
External Clock(0-16MHZ)	0000
Reserved	0011, 0001, 0101, 0100

۳-۱۷ خصوصیات Atmega165v ,Atmega 165

- **از معماری AVR RISC استفاده می‌کند.**
  - کارایی بالا و توان مصرفی کم
  - دارای 130 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می‌شوند.
  - 32*8 رجیستر کاربردی
  - عملکرد کاملاً ثابت
  - سرعتی تا 16 MIPS در فرکانس 16 MHZ

• **حافظه ، برنامه و دادهٔ غیرفرّار**

— 16K بایت حافظه FLASH داخلی قابل برنامه‌ریزی

پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن(WRITE / ERASE)

— 1K بایت حافظه داخلی SRAM

— 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی

پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن(WRITE / ERASE)

— قفل برنامه برای حفاظت نرم افزار

• **قابلیت ارتباط JTAG (IEEE Std.)**

— برنامه‌ریزی برنامهٔ FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG

• **خصوصیات جانبی**

— دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و مُد COMPARE

— یک تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مُدهای COMPARE و CAPTURE

— 4 کانال PWM

— 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی

— یک مقایسه‌کننده آنالوگ داخلی

— WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی

— قابلیت ارتباط سریال SPI به صورت MASTER یا SLAVE

— قابلیت ارتباط سریال USI (UNIVERSAL SERIAL INTERFACE)

— USART سریال قابل برنامه‌ریزی

— دارای RTC (REAL -TIME COUNTER) بااسیلاتور مجزا

— وقفه در اثرتغییر وضعیت پایه

• **خصوصیات ویژه میکروکنترلر**

— POWER - ON RESET CIRCUIT و BROWN-OUT قابل برنامه‌ریزی

— دارای اسیلاتور داخلی کالیبره شده

— دارای 5 حالتSLEEP (POWER - DOWN ، POWER - SAVE ، ADC NOISE REDUCTION ، STANDBY، IDLE)

— منابع وقفه (INTERRUPT) داخلی و خارجی

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• **توان مصرفی فوق العاده پایین**

— حالت فعال

در 1MHZ و ولتاژ 1.8V : 350µA

در 32KHZ و ولتاژ 1.8V : 20µA (با اسیلاتور)

— در حالت POWER – DOWN و ولتاژ 1.8V : 0.1mA

• **خطوط I/O و انواع بسته‌بندی**

— 53 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی

— 64 پایه TQFP و MLF

• **ولتاژها و فرکانسهای عملیاتی (کاری)**

— Atmega165V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V

— Atmega165: 0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V

فیوز بیتهای ATMEGA165V , ATMEGA 165

جدول ۳-۷۶ بایت پیشرفته فیوزبیتهای میکرو

EXTENDED FUSE BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
-	7	-	1
-	6	-	1
-	5	-	1
-	4	-	1
BODLEVEL 2	3	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL 1	2	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODLEVEL 0	1	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
RESERVED	0	EXTERNAL RESET DESABLE	1(UNPROGRAMMED)

جدول۳ -۷۷ بایت پر ارزش فیوزبیتهای میکرو

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
WDTON	4	WATCHDOG TIMER ALWAYSE ON	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

جدول ۳-۷۸ بایت کم ارزش فیوزبیتهای میکرو

FUSE LOW BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
CKDIV8	7	DIVIDE CLICK BY 8	0(PROGRAMMED)
CKOUT	6	CLOCK OUTPUT	1(UNPROGRAMMED)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	1(UNPROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	0(PROGRAMMED)

BODLEVEL0,1,2 : این سه میکرو دارای مدار BROWN-OUT است که ولتاژ تریگ آن توسط فیوز بیت‌های BODLEVEL طبق جدول ۳-۷۹ قابل تنظیم می‌باشد.

BOOTSZ0..1 : برای انتخاب مقدارحافظه BOOT طبق جدول ۳-۸۰ برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

BOOTRST : بیتی جهت انتخاب بردار ری‌ست است که در حالت پیش‌فرض برنامه‌ریزی نشده است.

SUT0,1 : عملکرد این دو بیت برای انتخاب زمان START-UP است.

CKSEL0..3 : عملکرد این بیت‌ها با توجه به جدول ۳-۸۱ است. مقدار پیش‌فرض INTERNAL RC OSC @ 8MHZ و با توجه به برنامه ریزی شدن بیت CKDIV8 ، کلاک سیستم برابر 1MHZ است.

جدول۳-۷۹ انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

BODLEVEL FUSES 2... 0	VBOT(VBOT- ,VBOT+)
11	BOD DISABLE
10	1.8v ± 25mv
01	2.7v ± 25mv
00	4.3v ± 25mv

جدول ۳-۸۰ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	2	0x0000 - 0x1F7F	0x1F80 - x1FFF	0x1F80
1	0	256 words	4	0x0000 - 0x1EFF	0x1F00 - x1FFF	0x1F00
0	1	512 words	8	0x0000 - 0x1DFF	0x1E00 - x1FFF	0x1E00
0	0	1024 words	16	0x0000 - 0x1BFF	0x1C00 -x1FFF	0x1C00

جدول ۳-۸۱ مُد و بازه های فرکانس کاری

FREQUENCY MODE AND RANGE	CKSEL3...0
Low Power Crystal Oscillator(0.4-0.9MHZ)	1000 (C1,2= -)
Low Power Crystal Oscillator(0.9-3.0MHZ)	1011 - 1010(C1,2=12-22PF)
Low Power Crystal Oscillator(3.0-8.0MHZ)	1101 - 1100(C1,2=12-22PF)
Low Power Crystal Oscillator(8.0-16.0MHZ)	1111 - 1110(C1,2=12-22PF)
Low Frequency Crystal Oscillator	0111 - 0110
Calibrated Internal RC Oscillator	0010(7.3-8.1MHZ)
External Clock(0-16MHZ)	0000
Reserved	0011, 0001, 0101, 0100

۳-۱۸ کلاک سیستم (۱)

توزیع کلاک

کلاک سیستم میکرو طبق شکل۳-۷ توزیع شده است.

کلاک CLKCPU – CPU

این کلاک برای انجام عملیات AVR به طور مثال رجیسترها استفاده می‌شود. توقف و به مکث بردن این کلاک باعث می‌شود که عملیات و محاسبات AVR انجام نگیرد.

کلاک CLK I/O – I/O

این کلاک توسط بسیاری از ماژول‌های I/O به طـور مثـال تایمرهـا ، کانترهـا ، SPI وUSART استفاده می‌گردد.

کلاک CLKFLASH – FLASH

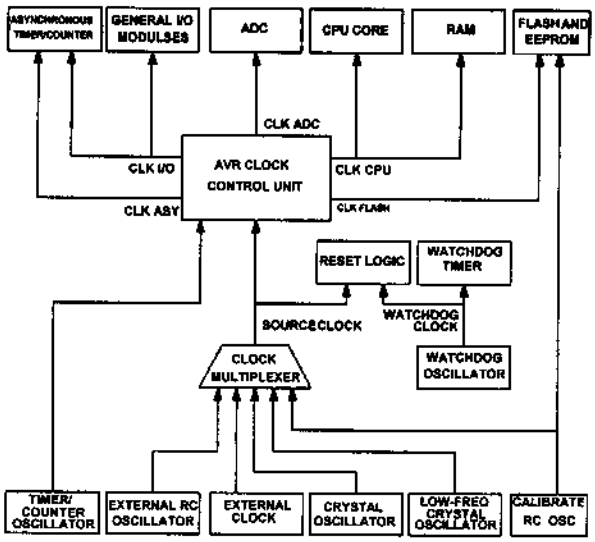
این کلاک عملیات ارتباطی با حافظه FLASH را کنترل می‌کند. کلاک FLASH معمولاً با کلاک CPU فعال می‌شود.

کلاک غیرهمزمان تایمر – CLK ASY

با این کلاک تایمر/کانتر به صورت غیرهمزمان توسط کریستال ساعت 32768 HZ کار می‌کند حتی اگر سیستم در حالت SLEEP باشد.

کلاک CLKADC – ADC

ADC از یک کلاک جداگانه حساس استفاده می‌کند که باعث می‌شود کلاک های CPU و I/O به حالت ایست (HALT) رفته تا نویز حاصل از مدار دیجیتال داخلی کاهش یافته و در نتیجه عملیات تبدیل با دقت بیشتری انجام یابد



شکل ۳-۷ نمودار توزیع کلاک سیستم میکرو

منابع کلاک (CLOCK SOURCE)

میکرو دارای انواع منابع کلاک اختیاری است که می‌توان انواع آن را به وسیله بیت‌های قابل برنامه‌ریزی FLASH (FLASH FUSE BITS) انتخاب کرد. کلاک انتخاب شده به عنوان ورودی کلاک AVR طبق جدول ۳-۸۲ در نظر گرفته شده وکلاک مناسب به هر قسمت سیستم داده می‌شود.

نکته

در تمام جداول فیوز بیت‌ها ، 0 به معنای بیت برنامه‌ریزی شده (PROGRAMMED) و 1 به معنای بیت برنامه‌ریزی نشده (UNPROGRAMMED) است.

جدول ۳-۸۲ انتخاب انواع کلاک سیستم میکرو

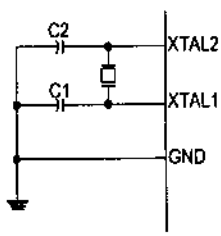
DEVICE CLOCKING OPTION	CKSEL3...0
EXTERNAL CRYSTAL/CERAMIC RESONATOR	1111 - 1010
EXTERNAL – LOW FREQUENCY CRYSTAL	1001
EXTERNAL RC OCSILLATOR	1000 - 0101
CALIBRATED INTERNAL RC SCILLATOR	0100 - 0001
EXTERNAL CLOCK	0000

هنگامی که CPU از مُد POWER-DOWN یا POWER-SAVE خارج می‌شود زمانی به نام زمان شروع (START-UP) برای رسیدن کریستال به شرایط پایدار ایجاد و سپس دستورات برنامه اجرا می‌شود و هنگامی که CPU از ری‌ست شروع به کار می‌کند، تاخیری اضافه (DELAY) برای رسیدن ولتاژ به سطح پایدار ایجاد شده و سپس اجرای برنامه آغاز می‌شود. برای ایجاد زمانبندی‌های مذکور از اسیلاتور WATCHDOG استفاده می‌شود.

اسیلاتور کریستالی(EXTERNAL CRYSTAL/CERAMIC RESONATOR)

در این حالت کریستال یا نوسانگر سرامیکی (CERAMIC RESONATOR) یا کریستال کوارتز (QUARTZ CRYSTAL) همانطور که در شکل ۳-۸ نشان داده شده است به دو پایه XTAL1 و XTAL2 وصل شود.

فیوز بیت CKOPT می‌تواند برای دو حالت مختلف استفاده شود. زمانی که محیط بسیار نویزی باشد ، این بیت برنامه‌ریزی می‌شود که رنج وسیعی از فرکانسها را شامل می‌شود. برنامه‌ریزی نکردن CKOPT باعث کاهش مصرف شده و بر خلاف قبل رنج محدودی از فرکانسها را شامل می‌شود. خازنهای C1 و C2 برای کریستال‌ها و نوسانگرها بایستی یک مقدار باشند و مقادیر آنها بستگی به کریستال ، نوسانگر و نویزهای الکترومغناطیسی محیط دارد. بعضی از خازنهای مورد استفاده برای کریستال‌های مختلف در جدول ۳-۸۳ آمده است. برای نوسانگرهای سرامیکی بایستی مقدار خازنهایی که توسط کارخانه پیشنهاد می‌گردد استفاده شود.



شکل ۳-۸ اتصال کریستال به میکرو در حالت اسیلاتور کریستالی

جدول ۳-۸۳ مُدهای عملیاتی اسیلاتور کریستالی

CKOPT	CKSEL3..1	FREQUENCY RANGE (MHZ)	RECOMMENDED RANGE FOR CAPACITORS C1 AND C2 FOR USE WITH CRYSTAL (PF)
1	101[1]	0.4 – 0.9	-
1	110	0.9 – 3.0	12 – 22
1	111	3.0 – 8.0	12 – 22
0	101 , 110 , 111	1.0≤	12 – 22

[1] – این انتخاب برای نوسانگر سرامیکی استفاده می‌شود و نباید آن را برای کریستال به کار برد..

در این حالت توسط فیوز بیت CKSEL0 و SUT1..0 زمان آغاز (START-UP) را می‌توان طبق جدول ۳-۸۴ انتخاب کرد.

اسیلاتور کریستالی فرکانس پایین(EXTERNAL – LOW FREQUENCY CRYSTAL)

برای استفاده از کریستال ساعت 32.768KHZ ، فیوز بیتهای CKSEL با 1001 برنامه‌ریزی می‌شوند و کریستال طبق شکل ۳-۸ به پایه‌های XTAL1 و XTAL2 متصل می‌شود. با برنامه‌ریزی کردن CKOPT می‌توان خازنهای داخلی را فعال نمود و در نتیجه خازنهای خارجی را برداشت. مقدار نامی خازنهای داخلی 36PF است.

هنگامی که این نوع کریستال انتخاب می‌شود ، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۳-۸۵ قابل انتخاب است.

جدول ۳-۸۴ انتخاب زمان START-UP برای کلاک اسیلاتور کریستالی

CKSEL0	SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
0	00	258 CK[1]	4.1ms	CERAMIC RESONATOR , FAST RISING POWER
0	01	258 CK[1]	65 ms	CERAMIC RESONATOR , SLOWLY RISING POWER
0	10	1K CK[2]	-	CERAMIC RESONATOR , BOD ENABLE
0	11	1K CK[2]	4.1 ms	CERAMIC RESONATOR , FAST RISING POWER
1	00	16K CK	65 ms	CERAMIC RESONATOR , SLOWLY RISING POWER
1	01	16K CK	-	CRYSTAL OSCILLATOR, BOD ENABLE
1	10	16K CK	4.1 ms	CRYSTAL OSCILLATOR , FAST RISING POWER
1	11	16K CK	65ms	CRYSTAL OSCILLATOR , SLOWLY RISING POWER

[1] - این گزینه‌ها زمانی که سیستم در فرکانسهای بالا کار نمی‌کند استفاده می‌گردد . انتخاب این گزینه‌ها برای کریستال‌ها مناسب نیست.

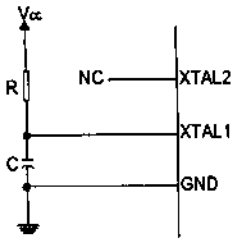
[2] - این گزینه‌ها برای نوسانگرهای سرامیکی استفاده می‌شود. همچنین می‌توانند برای کریستال‌ها زمانی که در فرکانسهای پایین کار می‌کنند استفاده شوند.

جدول ۳-۸۵ انتخاب زمان START-UP برای کلاک اسیلاتور کریستالی فرکانس پایین

SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	1K CK	4.1 ms	FAST RISING POWER OR BOD ENABLE
01	1K CK	65 ms	SLOWLY RISING POWER
10	32K CK	65 ms	STABLE FREQUENCY AT START -UP
11	RESERVED		

اسیلاتور RC خارجی (EXTERNAL RC OCSILLATOR)

اتصال RC به پایه‌های XTAL1 در شکل ۳-۹ آمده است. فرکانس تقریبی توسط معادله f =1 /(3RC) بدست می‌آید. مقدار خازن بایستی حداقل 22PF باشد. با برنامه‌ریزی کردن فیوز بیت CKOPT کاربر می‌تواند خازنهای داخلی36PF را بین XTAL1 وGND راه‌اندازی کند و در نتیجه دیگر نیازی به خازن خارجی نیست.



شکل ۳-۹ اتصال RC به میکرو در حالت اسیلاتور RC خارجی

اسیلاتور می‌تواند در 4 مُد فرکانسی کار کند که این فرکانس‌ها طبق فیوز بیت‌های CKSEL3..0 طبق جدول ۳-۸۶ قابل انتخاب است. هنگامی که فرکانس کاری انتخاب می‌شود، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۳-۸۷ قابل انتخاب است.

جدول ۳-۸۶ مُدهای عملیاتی اسیلاتور RC خارجی

CKSEL3..0	FREQUENCY RANGE (MHZ)
0101	≤ 0.9
0110	0.9 – 3.0
0111	3.0 – 8.0
1000	8.0 – 12.0

جدول۳-۸۷ انتخاب زمان START-UP برای کلاک اسیلاتور RC خارجی

SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	18 CK	-	BOD ENABLE
01	18 CK	4.1ms	FAST RISING POWER
10	18 CK	65 ms	SLOWLY RISING POWER
11	6 CK[1]	4.1ms	FAST RISING POWER OR BOD ENABLE

[1]- این گزینه زمانی که میکرو در فرکانسهای بالا کار می کند نباید انتخاب گردد.

اسیلاتور RC کالیبره شده داخلی (CALIBRATED INTERNAL RC SCILLATOR)

اسیلاتور RC کالیبره شده داخلی ، کلاک‌های نامی داخلی 1 ، 2 ، 4 و 8MHZ (جدول ۳-۸۸) را در ولتاژ 5V و 25^0C تولید می‌کند. این کلاک با برنامه‌ریزی کردن بیت‌های CKSEL می‌تواند به عنوان کلاک سیستم استفاده گردد که در این حالت نیازی به مدار خارجی نیست. زمانی که از این مُد استفاده می‌شود فیوز بیت CKOPT همیشه بایستی برنامه‌ریزی شده باشد.

جدول ۳-۸۸ مُدهای عملیاتی اسیلاتور RC کالیبره شده داخلی

CKSEL3..0	NOMINAL RANGE (MHZ)
0001[1]	1.0
0010	2.0
0011	4.0
0100	8.0

[1]- برای میکرو به صورت پیش‌فرض این گزینه انتخاب شده است.

هنگامی که فرکانس کاری انتخاب می‌شود ، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۳-۸۹ قابل انتخاب است.

جدول ۳-۸۹ انتخاب زمان START-UP برای کلاک اسیلاتور RC کالیبره شده داخلی

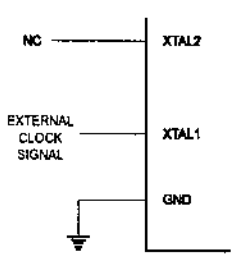
SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10[1]	6 CK	65 ms	SLOWLY RISING POWER
11	RESERVED		

[1]- برای میکرو به صورت پیش‌فرض این گزینه انتخاب شده است.

کلاک خارجی(EXTERNAL CLOCK)

برای راه اندازی میکرو توسط کلاک خارجی پایه XTAL1 طبق شکل ۳-۱۰ بایستی وصل شود. برای کار در این مُد بیت‌های CKSEL با 0000 برنامه‌ریزی می‌شوند. با برنامه‌ریزی کردن فیوز بیت CKOPT خازن داخلی 36PF بین پایه‌های XTAL1 و GND فعال می‌شود.

هنگامی که این نوع کلاک انتخاب می‌شود ، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۳-۹۰ قابل انتخاب است.



شکل ۳-۱۰ اتصال کلاک خارجی به پایه میکرو در حالت کلاک خارجی

جدول ۳-۹۰ انتخاب زمان START-UP برای کلاک خارجی

SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10	6 CK	65 ms	SLOWLY RISING POWER
11	RESERVED		

در این مُد باید از تغییرات ناگهانی فرکانس کلاک خارجی برای اطمینان از انجام پایدار و صحیح عملیات میکروکنترلر (MCU) جلوگیری کرد. تغییرات بیشتر از 2% در فرکانس کلاک خارجی ممکن است باعث رفتارهای غیر قابل انتظار میکرو شود. زمانی که قصد تغییر فرکانس کلاک را دارید بایستی میکرو در حالت RESET نگه داشته شود.

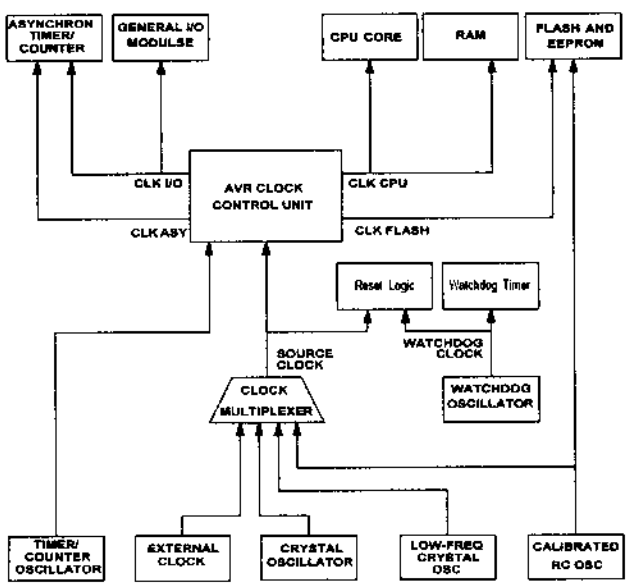
اسیلاتور تایمر/کانتر

برای میکروکنترلرهایی که دارای پایه TOSC1 و TOSC2 هستند، کریستال ساعت 32.768KHZ مستقیماً بین دو پایه قرار می‌گیرد و تایمر/کانتر 0 یا 2 به صورت آسنکرون از این دو پایه کلاک دریافت می‌کند.

۳-۱۹ کلاک سیستم (۲)

توزیع کلاک

کلاک سیستم میکرو طبق شکل ۳-۱۱ توزیع شده است.



شکل ۳-۱۱ نمودار توزیع کلاک سیستم میکرو

کلاک CLKCPU – CPU

این کلاک برای انجام عملیات AVR به طور مثال رجیسترها استفاده می‌شود. توقف و به وقفه بردن این کلاک باعث می‌شود که عملیات و محاسبات AVR انجام نگیرد.

کلاک CLK I/O – I/O

این کلاک توسط بسیاری از ماژول های I/O به طور مثال تایمرها ، کانترها ، SPI و USART استفاده می‌شود.

کلاک CLKFLASH – FLASH

این کلاک عملیات ارتباطی با حافظه FLASH را کنترل می‌کند. کلاک FLASH معمولاً با کلاک CPU فعال می‌شود.

کلاک غیرهمزمان تایمر – CLK ASY

با این کلاک تایمر/کانتر به صورت غیرهمزمان توسط کریستال ساعت 32768 HZ کار می‌کند حتی اگر سیستم در حالت SLEEP باشد.

منابع کلاک (CLOCK SOURCE)

میکرو دارای انواع منابع کلاک اختیاری است که می‌توان انواع آن را به وسیله بیت‌های قابل برنامه‌ریزی FLASH (FLASH FUSE BITS) انتخاب کرد. کلاک انتخاب شده به عنوان ورودی کلاک AVR طبق جدول ۳-۹۱ در نظر گرفته شده وکلاک مربوطه به هر قسمت سیستم داده می‌شود.

نکته

در تمام جداول فیوز بیت‌ها ، 0 به معنای بیت برنامه‌ریزی شده (PROGRAMMED) و 1 به معنای بیت برنامه‌ریزی نشده (UNPROGRAMMED) است.

جدول ۳-۹۱ انتخاب انواع کلاک سیستم میکرو

DEVICE CLOCKING OPTION	CKSEL3...0
EXTERNAL CRYSTAL/CERAMIC RESONATOR	1111 - 1000
EXTERNAL – LOW FREQUENCY CRYSTAL	0111 - 0100
CALIBRATED INTERNAL RC SCILLATOR	0010
EXTERNAL CLOCK	0000
RESERVED	0011, 0001

هنگامی که CPU از مُد POWER-DOWN یا POWER-SAVE خارج می‌شود زمانی به نام زمان شروع (START-UP) برای رسیدن کریستال به شرایط پایدار ایجاد و سپس دستورات برنامه اجرا می‌شود و هنگامی که CPU از ری‌ست شروع به کار می‌کند، تاخیری اضافه (DELAY) برای رسیدن

ولتاژ به سطح پایدار ایجاد شده و سپس اجرای برنامه آغاز می‌شود. برای ایجاد زمان‌بندی‌های مذکور از اسیلاتور WATCHDOG استفاده می‌شود.

اسیلاتور کریستالی(EXTERNAL CRYSTAL/CERAMIC RESONATOR)

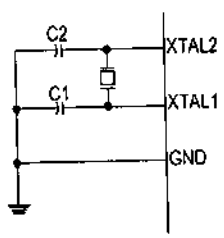
در این حالت کریستال یا نوسانگر سرامیکی (CERAMIC RESONATOR) یا کریستال کوارتز (QUARTZ CRYSTAL) همانطور که در شکل ۳-۱۲ نشان داده شده است به دو پایه XTAL1 و XTAL2 وصل می‌شود.

خازنهای C 1 وC2 برای کریستال‌ها و نوسانگرها بایستی یک مقدار باشند. مقادیر خازنها بستگی به کریستال ، نوسانگر و نویزهای الکترومغناطیسی محیط دارند. بعضی از خازنهای مورد استفاده برای کریستال‌های مختلف در جدول ۳-۹۲ آمده است. برای نوسانگرهای سرامیکی مقدار خازنهایی که توسط کارخانه پیشنهاد می‌شود بایستی استفاده گردد. توسط فیوز بیت‌های CKSEL0 و SUT1..0 زمان آغاز (START-UP) را می‌توان طبق جدول ۳-۹۳ انتخاب کرد.

اسیلاتور کریستالی فرکانس پایین(EXTERNAL – LOW FREQUENCY CRYSTAL)

برای استفاده از کریستال ساعت 32.768KHZ ، فیوز بیتهای CKSEL با 0100 ، 0101 ، 0110 یا 0111 برنامه‌ریزی می‌شوند و کریستال طبق حالت اسیلاتور کریستالی به پایه‌های XTAL1 و XTAL2 متصل می‌شود. با برنامه‌ریزی کردن CKSEL با مقادیر 0110 یا 0111 (جدول ۳-۹۴) می‌توان خازنهای داخلی را فعال نمود و در نتیجه خازنهای خارجی را برداشت. مقدار نامی خازنهای داخلی 10PF است.

هنگامی که این نوع کریستال انتخاب می‌شود ، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۳-۹۵ قابل انتخاب است.



شکل ۳-۱۲ اتصال کریستال به میکرو در حالت اسیلاتور کریستالی

جدول ۳-۹۲ مُدهای عملیاتی اسیلاتور کریستالی

CKSEL3..1	FREQUENCY RANGE (MHZ)	RECOMMENDED RANGE FOR CAPACITORS C1 AND C2 FOR USE WITH CRYSTAL (PF)
100[1]	0.4 – 0.9	-
101	0.9 – 3.0	12 – 22
110	3.0 – 8.0	12 – 22
111	8.0≤	12 – 22

[1] – این انتخاب برای نوسانگر سرامیکی استفاده می‌شود و نباید آن را برای کریستال به کار برد.

جدول ۳-۹۳ انتخاب زمان START-UP برای کلاک اسیلاتور کریستالی

CKSEL0	SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
0	00	258 CK[1]	4.1ms	CERAMIC RESONATOR , FAST RISING POWER
0	01	258 CK[1]	65 ms	CERAMIC RESONATOR , SLOWLY RISING POWER
0	10	1K CK[2]	-	CERAMIC RESONATOR , BOD ENABLE
0	11	1K CK[2]	4.1 ms	CERAMIC RESONATOR , FAST RISING POWER
1	00	16K CK	65 ms	CERAMIC RESONATOR , SLOWLY RISING POWER
1	01	16K CK	-	CRYSTAL OSCILLATOR, BOD ENABLE
1	10	16K CK	4.1 ms	CRYSTAL OSCILLATOR , FAST RISING POWER
1	11	16K CK	65ms	CRYSTAL OSCILLATOR , SLOWLY RISING POWER

[1]- این گزینه‌ها زمانی که سیستم در فرکانسهای بالا کار نمی‌کند استفاده می‌شود. انتخاب این گزینه‌ها برای کریستال‌ها مناسب نیست.

[2]- این گزینه‌ها برای نوسانگرهای سرامیکی استفاده می‌شود. همچنین می‌توانند برای کریستال‌ها زمانی که در فرکانسهای پایین کار می‌کنند استفاده شوند.

جدول ۳-۹۴ انتخاب زمان START-UP برای کلاک اسیلاتور کریستالی فرکانس پایین

CKSEL 3..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	INTERNAL CAPACITORS ENABLED	RECOMMENDED USAGE
0100	1K CK	NO	-
0101	32K CK	NO	STABLE FREQUENCY AT START UP
0110	1K CK	YES	-
0111	32K CK	YES	STABLE FREQUENCY AT START UP

جدول ۳-۹۵ انتخاب زمان RESET DELAY برای کلاک اسیلاتور کریستالی فرکانس پایین

SUT1..0	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	0ms	FAST RISING POWER OR BOD ENABLE
01	4.1 ms	FAST RISING POWER OR BOD ENABLE
10	65 ms	SLOWLY RISING POWER
11	RESERVED	

اسیلاتور RC کالیبره شده داخلی(CALIBRATED INTERNAL RC SCILLATOR)

اسیلاتور RC کالیبره شده داخلی، کلاک‌های نامی داخلی 1 ، 2 ، 4 و 8MHZ را در ولتاژ 5V و 25^0C تولید می‌کند. این کلاک با برنامه‌ریزی کردن بیت‌های CKSEL می‌تواند به عنوان کلاک سیستم استفاده شود که در این حالت نیازی به مدار خارجی نیست. پیش فرض میکرو اسیلاتور RC داخلی 8MHZ است. هنگامی که فرکانس کاری انتخاب می‌شود ، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۳-۹۶ قابل انتخاب است.

جدول ۳-۹۶ انتخاب زمان START-UP برای کلاک اسیلاتور RC کالیبره شده داخلی

SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10[1]	6 CK	65 ms	SLOWLY RISING POWER
11	RESERVED		

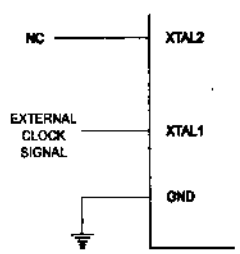
[1]- میکرو به صورت پیش فرض این گزینه انتخاب شده است.

کلاک خارجی(EXTERNAL CLOCK)

برای راه اندازی چیپ توسط کلاک خارجی پایه XTAL1 طبق شکل ۳-۱۳ بایستی وصل شود. برای کار در این مُد بیت‌های CKSEL با 0000 برنامه‌ریزی می‌شوند.

هنگامی که این نوع کلاک انتخاب می‌شود ، زمان شروع (START-UP) توسط فیوز بیت‌های SUT طبق جدول ۳-۹۷ قابل انتخاب است.

در این مُد باید از تغییرات ناگهانی فرکانس کلاک خارجی برای اطمینان از انجام پایدار و صحیح عملیات میکروکنترلر (MCU) جلوگیری کرد. تغییرات بیشتر از 2% در فرکانس کلاک خارجی ممکن است باعث رفتارهای غیر قابل انتظار میکرو شود. زمانی که قصد تغییر فرکانس کلاک را دارید بایستی میکرو در حالت RESET نگه داشته شود.



شکل ۳-۱۳ اتصال کلاک خارجی به پایه میکرو در حالت کلاک خارجی

جدول ۳-۹۷ انتخاب زمان START-UP برای کلاک خارجی

SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10	6 CK	65 ms	SLOWLY RISING POWER
11	RESERVED		

اسیلاتور تایمر/کانتر

برای میکرو کنترلرهایی که دارای پایه TOSC1 و TOSC2 هستند ، کریستال ساعت 32.768KHZ مستقیماً بین دو پایه قرار می گیرد. برای میکرو ATMEGA169 پایه‌های XTAL1 و TOSC1 ، XTAL2 و TOSC2 مشترک هستند و این بدان معناست که زمانی که سیستم با اسیلاتور RC داخلی کار می‌کند تایمر می‌تواند از این دو پایه کلاک دریافت کند.

۴

محیط برنامه‌نویسی BASCOMAVR

انواع متنوعی از کامپایلرهای AVR عرضه شده‌اند که در این میان کامپایلرهای BASCOM ، CODEVISION ، FASTAVR و AVR STUDIO از اهمیت و اعتبار بیشتری برخوردار هستند. در این فصل قصد داریم به معرفی یکی از قوی ترین آنها به نام BASCOM AVR ویرایش 1.11.7.4 بپردازیم. BASCOM تمام میکروهای AVR را حمایت کرده و از زبان BASIC برای برنامه‌نویسی AVR ها استفاده می‌نماید. در این فصل منوهای BASCOM به طور کامل تشریح شده‌اند. از قابلیت‌های بسیار ارزنده محیط BASCOM داشتن تحلیل‌گر یا به عبارتی SIMULATOR داخلی است که برای یادگیری برنامه‌نویسی AVR بسیار کارآمد است.

ورودی سیگنال آنالوگ ADC و مقایسه‌کننده آنالوگ ، ایجاد پالس بر روی پایه‌ای خاص، صفحه‌کلید 4×4 ، LCD ، ایجاد تمام وقفه‌ها به صورت اختیاری، نوشتن و خواندن حافظه های EEPROM و SRAM ، رویت تمام رجیسترها و متغییرهای محلی و سراسری برنامه ، اجرای برنامه به صورت خط به خط ، رویت صفر یا یک بودن تمام پایه توسط LED ، تغییر منطق پایه دلخواه و بسیاری از امکانات دیگر توسط محیط تحلیل‌گر (SIMULATOR) و از همه مهمتر برنامه نویسی ساده باعث شده است که این کامپایلر در کنار دیگر کامپایلرهای معروف مورد تایید و استفاده برنامه‌نویسان قرار گیرد.

در انتهای این فصل نیز قصد داریم به ساخت STK200/300 PROGRAMER به وسیله یک بافر و ساخت نوع دیگری که در عرض مدت کوتاهی قابل ساخت است بپردازیم.

اهداف

۱. یادگیری کامل منوهای محیط برنامه‌نویسی BASCOM AVR از قبیل محیط های SIMULATOR، PROGRAMER ، TERMINAL EMULATORE ، LCD DESIGNER و GRAPHIC CONVERTORE

۲. ساخت ISP-PROGRAMER میکروهای AVR

۴-۱ معرفی منوهای محیط BASCOM

پس از اجرای برنامه BASCOM پنجره محیط برنامه نویسی BASCOM ظاهر خواهد شد. اگر اولین راه‌اندازی شما باشد پنجره خالی خواهد بود در غیر این صورت آخرین فایلی که بسته شده، ظاهر می‌شود.

- **منوی FILE**

ایجاد فایل جدید (FILE NEW)

با انتخاب این گزینه یک پنجره جدید که شما قادر به نوشتن برنامه در آن هستید ایجاد می‌شود.

بازکردن فایل (FILE OPEN)

با انتخاب این گزینه شما قادر به فراخوانی فایلی که در حافظه موجود است می باشید.
BASCOM فایلها را به صورت استاندارد ASCII ذخیره می‌کند. بنابراین شما می‌توانید از ویرایشگری مانند NOTEPAD برای نوشتن برنامه استفاده کنید و سپس آن را به محیط انتقال دهید.

بستن فایل (FILE CLOSE)

این گزینه پنجره برنامه فعال را می‌بندد. اگر در فایل تغییری ایجاد کرده‌اید ابتدا باید قبل از بستن آن را ذخیره نمایید.

ذخیره فایل (FILE SAVE)

با این گزینه شما قادر به ذخیره فایل به صورت ASCII در حافظه کامپیوتر خواهید بود.

ذخیره کردن بعنوان (FILE SAVE AS)

با این گزینه قادر خواهید بود فایل موجود را با نام دیگر ذخیره کنید.

نمایش پرینت فایل (FILE PRINT PREVIEW)

این گزینه نشان می دهد که فایل متنی موجود برنامه در هنگام پرینت به چه صورت خواهد بود.

پرینت فایل (FILE PRINT)

با این گزینه شما می‌توانید فایل برنامه موجود را پرینت نمایید.

خروج از فایل (FILE EXIT)

با این گزینه شما قادر خواهید بود از محیط BASCOM خارج شوید ولی در صورتی که شما در برنامه‌تان تغییری داده‌اید و آن را ذخیره نکرده‌اید، پیش از خروج هشدار می دهد.

▪ منوی EDIT

EDIT UNDO
با این گزینه شما می‌توانید دستکاری اخیرتان در برنامه را از بین ببرید.

EDIT REDO
با این گزینه شما می‌توانید دستکاری اخیرتان را که از بین برده بودید دوباره برگردانید.

EDIT CUT
با این گزینه شما می‌توانید متن انتخاب شده را بریده و به محل جدیدی انتقال دهید.

EDIT COPY
با این گزینه شما می‌توانید متن انتخاب شده را کپی کرده و به محل جدیدی انتقال دهید.

EDIT PAST
با این گزینه شما می‌توانید متنی را که قبلا COPY یا CUT کرده بودید در محل مورد نظر بچسبانید.

EDIT FIND
با این گزینه شما می‌توانید متنی را در برنامه‌تان جستجو کنید.

EDIT FIND NEXT
با این گزینه شما می‌توانید متن مورد جستجو را دوباره جستجو نمایید.

EDIT REPLACE
با این گزینه شما می‌توانید متنی را جایگزین متن موجود در برنامه نمایید. در قسمت TEXT TO FIND متن مورد جستجو را که باید توسط متن دیگری جایگزین شود تایپ کنید و در قسمت REPLACE WITH متنی را که باید جایگزین شود وارد کنید.

EDIT GOTO
با این گزینه شما می‌توانید مستقیماً و به سرعت به خط دلخواهی بروید.

EDIT TOGGLE BOOKMAR
با این گزینه شما می‌توانید در جاهای خاصی از برنامه که مورد نظر شماست نشانه گذاری کنید و به آنها توسط دستور EDIT GOTO BOOKMARK دسترسی پیدا کنید.

EDIT GOTO BOOKMARK
با این گزینه شما می‌توانید به نشانه‌هایی که قبلا گذاشته‌اید پرش نمایید.

EDIT IDENT BLOCK
با این گزینه شما می‌توانید متن انتخاب شده را به اندازه یک TAB به سمت راست منتقل کنید.

EDIT UNIDENT BLOCK
با این گزینه شما می‌توانید متن انتخاب شده را به اندازه یک TAB به سمت چپ منتقل کنید.

▪ منوی PROGRAM

PROGRAM COMPILE
با این گزینه (یا کلید F7) شما قادر به ترجمه برنامه به زبان ماشین (COMPILE) خواهید بود.برنامه شما با انتخاب این گزینه پیش از COMPILE ذخیره خواهد شد و فایلهای زیر بسته به انتخاب شما در

قسمت OPTION COMPILER SETTING ایجاد خواهند شد.

XX. BIN	فایل باینری که می‌تواند در میکروکنترلر PROGRAM شود.
XX. DBG	فایل DEBUG که برای نرم افزار شبیه ساز BASCOM مورد نیاز است.
XX. OBJ	فایل OBJECT که برای نرم افزار AVR STUDIO مورد نیاز است.
XX. RPT	فایل گزارشی
XX. HEX	فایل هگزدسیمال اینتل که برای بعضی از انواع PROGRAMMER ها مورد نیاز است.
XX. ERR	فایل خطا که فقط در هنگام بروز خطا ایجاد می‌شود.
XX. EEP	داده‌هایی که باید در EEPROM برنامه ریزی شوند در این فایل نگهداری می گردند.

اگر خطایی در برنامه موجود باشد شما پیغام خطا را در یک کادر محاوره‌ای دریافت خواهید کرد و کامپایل متوقف می‌شود. با کلیک به روی هرکدام از آنها به خطی که خطا در آن رخ داده است پرش خواهید کرد و علامت Ⓔ در اول خط خطا قرار می‌گیرد.

PROGRAM SYNTAX CHECK

بوسیله این گزینه برنامه شما برای نداشتن خطای املایی چک می‌شود. اگر خطایی وجود داشته باشد، هیچ فایلی ایجاد نخواهد شد.

PROGRAM SHOW RESULT

از این گزینه برای دیدن نتیجه COMPILE می توان استفاده کرد.

PROGRAM SIMULATOR

با فشردن کلید F2 یا این گزینه از منو PROGRAM شبیه ساز داخلی فعال خواهد شد. شما در برنامه با نوشتن دستور $SIM قادر به شبیه‌سازی سریعتر برنامه می باشید. در صورت تمایل شما می‌توانید از شبیه‌سازهای دیگری مانند AVR STUDIO نیز استفاده کنید. برای شبیه‌سازی باید فایلهای DBGو OBJ ایجاد شده باشند. فایل OBJ در برنامه شبیه‌سازیAVR STUDIO و فایل DBG برای شبیه ساز داخلی مورد استفاده قرار می‌گیرد.

نکته فایل HEX ایجاد شده در صورت وجود دستور $SIM در متن برنامه معتبر نمی‌باشد.

SEND TO CHIP

توسط این گزینه یا کلید F4 پنجره محیط برنامه‌ریزی ظاهر خواهد شد. شما می‌توانید توسط این گزینه میکرو مورد نظر خود را PROGRAM نمایید.

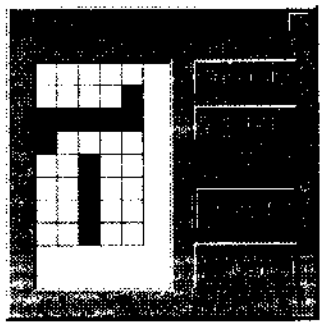
▪ منوی TOOLS

TERMINAL EMULATOR

توسط این گزینه یا کلیدهای CTRL+T با بالا آوردن TERMINAL EMULATOR می‌توانید از این محیط برای نمایش داده ارسالی و دریافتی در ارتباط سریال RS -232 بین میکرو و کامپیوتر استفاده نمایید.

LCD DESIGNER

توسط این گزینه می توانید کاراکترهای دلخواه خود را طراحی نماید و بر روی LCD نمایش دهید. با کلیک این دکمه پنجره۴-۱ ظاهر می شود.



پنجره ۴-۱

محیط LCD DESIGNER

ماتریس LCD برای هر حرف دارای 5 × 8 پیکسل می باشد که شما می‌توانید با کلیک چپ هرکدام از مربع‌ها را انتخاب و با کلیک دوباره آن را از حالت انتخاب خارج کنید. دکمه SET ALL همه نقاط را انتخاب وCLEAR ALL همه را از حالت انتخاب خارج می‌کند. پس از طراحی ، کلید OK را کلیک کنید. با این کار یک خط شامل تعدادی عدد مانند عبارت زیر به برنامه شما اضافه می‌شود.

DEFLCDCHAR ?, 14 , 21 , 21 , 27 , 27 , 21 , 21 , 14

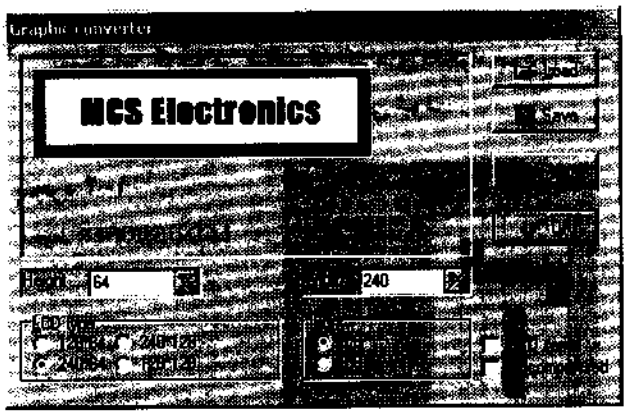
شما تنها می‌توانید هشت کاراکتر جدید را برای LCD تعریف کنید. پس به جای ? می‌توانید یک عدد بین 0 تا 7 جایگزین کنید. کاراکتر طراحی شده را می‌توان توسط دستور LCD CHR(?) بعد از دستور CLS بر روی LCD نمایش داد.

GRAPHIC CONVERTOR

با کلیک برروی این منو پنجره محیط GRAPHIC CONVERTOR (پنجره ۴-۲) برای تبدیل تصویر با پسوند BMP.* به تصویری با پسوند BGF.* که قابل نمایش بر روی GRAPHIC LCD است ظاهر می‌شود.

فایل دلخواه خود را با پسوندBMP .* توسط دکمه LOAD وارد کرده و سپس با دکمه SAVE آن را در کنار برنامه خود با پسوند BGF.* (BASCOM GRAPHIC FILE) ذخیره کنید. فایل تبدیل شده به صورت سیاه و سفید دوباره نمایش داده می‌شود و با کلیک بر روی دکمه OK می‌توان از محیط خارج شد. فایل ذخیره شده با فراخوانی در برنامه قابل نمایش بر روی LCD گرافیکی است. انتخاب نوع LCD توسط قسمت LCD TYPE انجام می‌گیرد. فونت نوشتاری نیز می‌تواند 8*6 یا 8*8 پیکسل باشد.

نکته رنگ‌هایی غیر از رنگ سفید در طی تبدیل به رنگ سیاه تبدیل خواهند شد.



پنجره ۴-۲

محیط GRAPHIC CONVERTOR

▪ منوی OPTIONS

OPTION COMPILER

با این منو شما می‌توانید گزینه‌های مختلف کامپایلر را طبق زیر اصلاح نمایید:

OPTION COMPILER CHIP

انتخاب میکرو برای برنامه‌ریزی توسط این گزینه انجام می‌شود. در صورتی که از دستور $REGFILE در برنامه استفاده کرده‌اید دیگر نیازی به انتخاب میکرو توسط این گزینه نیست.

OPTION COMPILER OUTPUT

این گزینه برای تعیین اینکه پس از کامپایل، کدام فایلها باید ایجاد شوند استفاده می شود. با انتخاب گزینه SIZE WARNING زمانی که حجم CODE از مقدار حافظه FLASH ROM تجاوز کرد کامپایلر تولید WARNING می‌کند. فایلهایی که محتوای آنها قابل مشاهده‌اند ، REPORT و ERROR می‌باشند.

OPTION COMPILER COMMUNICATION

نرخ انتقال (BAUD RATE) ارتباط سریال توسط این گزینه تعیین می‌شود که می‌توان یک نرخ جدید نیز تایپ کرد. گزینه FREQUENCY برای انتخاب فرکانس کریستال استفاده شده است که می‌تواند فرکانس اختیاری نیز باشد

OPTION COMPILER I2C,SPI,1WIRE

توسط این گزینه می‌توان پایه‌های مربوط به ارتباطات I2C، SPI و 1WIRE را برای میکرو تعیین کرد.

OPTION COMPILER LCD

گزینه OPTION COMPILER LCD نیز دارای قابلیت‌های زیر است :

در قسمت LCD TYPE نوع LCD را مشخص می‌کنیم. گزینه BUS MODE مشخص می‌کند LCD به صورت 8 بیتی یا 4 بیتی کار کند. توسط گزینه DATA MODE تعیین می‌کنیم LCD به صورت PIN کار کند یا BUS و گزینه LCD ADDRESS مشخص کننده آدرس LCD در مُد BUS است.

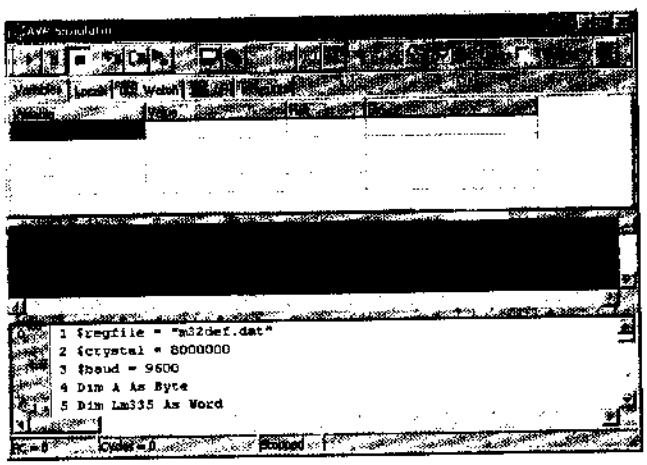
OPTIONS PROGRAMMER

در این منو شما می‌توانید PROGRAMMER موردنظر خود را انتخاب نمایید .در صورتی که بخواهید از پروگرامر انتهای فصل استفاده نمایید بایستی نوع STK200/300 را برای این منو انتخاب نمایید.

در صورت پیکره‌بندی هر یک از امکانات فوق در برنامه نرم افزاری نیازی به تنظیم کردن آنها در این منو نیست.

۴-۲ معرفی محیط شبیه‌سازی (SIMULATOR)

با اجرای محیط شبیه سازی پنجره ۴-۳ ظاهر خواهد شد. نوار ابزار شامل دکمه‌هایی است که با کلیک هرکدام عمل خاصی انجام می‌شود که قصد داریم در زیر به معرفی هر یک بپردازیم.



پنجره ۴-۳
محیط شبیه‌سازی

RUN : نام این کلید RUN می باشد. با کلیک کردن این دکمه شبیه‌سازی آغاز می‌شود.

PAUSE : این دکمه PAUSE می باشد، که باعث توقف موقت شبیه‌سازی می‌شود و با فشردن دکمه RUN شبیه‌سازی ادامه پیدا می‌کند.

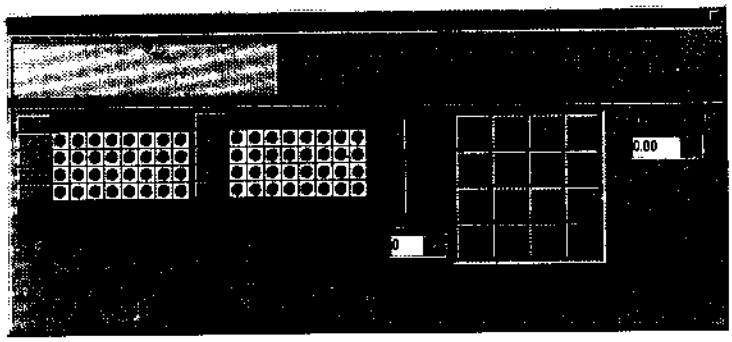
STOP : این دکمه باعث توقف کامل شبیه‌سازی برنامه جاری می‌شود.

STEP INTO CODE : این دکمه STEP INTO CODE نام دارد و به این معناست که شما می‌توانید برنامه را خط به خط اجرا کنید و می توان هنگام فراخوانی توابع به برنامه مربوطه رفته و مراحل اجرای آنها را بررسی کرد. این کار را با فشردن کلید F8 نیز می‌توانید انجام دهید. بعد از هر بار اجرای این دستور شبیه‌سازی به حالت PAUSE می رود.

STEP OVER : این دکمه شبیه دکمه قبلی است با این تفاوت که در هنگام فراخوانی توابع به داخل SUB ROUTINE نخواهید رفت. این کار را می‌توانید با فشردن کلید SHIFT F8 نیز انجام دهید.

RUN TO : دکمه RUN TO شبیه‌سازی را تا خط انتخاب شده انجام می دهد و سپس به حالت PAUSE می رود (خط جاری باید شامل کدهای قابل اجرا باشد).

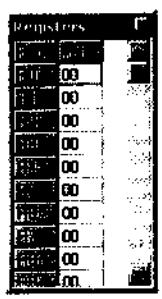
شبیه ساز سخت افزاری THE HARDWARE SIMULATOR
با کلیک روی این گزینه پنجره ۴-۴ نمایش داده می‌شود.



پنجره ۴-۴
محیط شبیه‌سازی
سخت‌افزاری

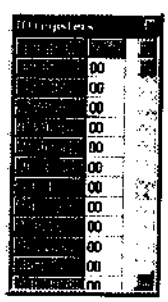
قسمت بالایی یک LCD مجازی می باشد که برای نشان دادن داده‌های ارسال شده به LCD استفاده می‌شود. نوار LED های قرمز رنگ پایین خروجی پورتها را نشان می دهد که روشن شدن هر یک به منزله یک شدن خروجی پایه پورت مربوطه است. با کلیک روی هر یک از LED های سبز رنگ که به عنوان ورودی هستند وضعیت معکوس می‌شود و روشن شدن LED به منزله یک کردن پایه پورت است. یک صفحه کلید نیز تعبیه شده است که با دستور ( )GETKBD در برنامه قابل خواندن می‌باشد. در ضمن مقدار آنالوگ نیز هم برای مقایسه‌کننده آنالوگ و هم برای کانال‌های مختلف ADC قابل اعمال است.

REGISTERES : این دکمه پنجره رجیسترها را با مقادیر فعلی نمایش می دهد. مقدارهای نشان داده شده در این پنجره هگزادسیمال می باشد که برای تغییر هر کدام از آنها روی خانه مربوطه کلیک کرده و مقدار جدید را وارد کنید.(پنجره۴-۵)



پنجره ۴-۵
نمایش رجیسترهای R0 - R31

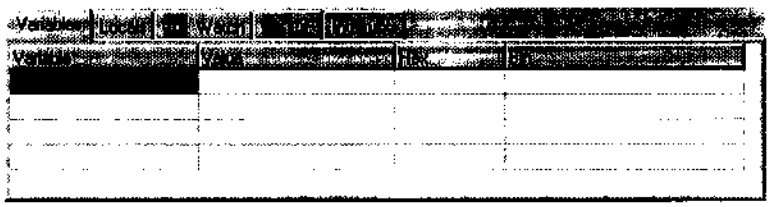
I/O REGISTERES : دکمه IO که برای نمایش رجیسترهای I/O استفاده می‌شود کـه ماننـد R قابـل مقداردهی است(پنجره۴-۶).



پنجره۴-۶
نمایش رجیسترهای I/O

گزینه VARIABLES

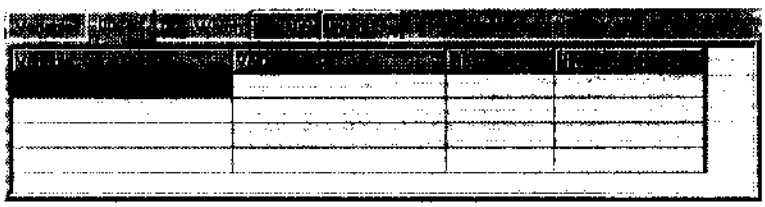
با انتخاب این منو پنجره ۴-۷ باز می شود سپس شما قادر به انتخاب متغیر با دوبار کلیک کردن در ستون VARIABLES می باشید. با زدن دکمه ENTER شما قادر به مشاهده مقدار جدید متغیر در هنگام اجرای برنامه هستید. همچنین می‌توانید مقدار هر متغیر را توسط VALUE تغییر دهید. برای تماشای مقدار یک متغیر آرایه‌ای شما می‌توانید نام متغیر همراه با اندیس آن را تایپ کنید و برای حذف هر سطر می‌توانید دکمه CTRL + DEL را بزنید.



پنجره ۴-۷
نمایش VARIABLES

گزینه LOCAL

پنجره LOCAL (پنجره ۴-۸) متغیرهای محلی موجود در SUB یا FUNCTION را نشان می دهد. شما نمی‌توانید متغیری را اضافه نمایید.



پنجره ۴-۸
نمایش متغیرهای محلی

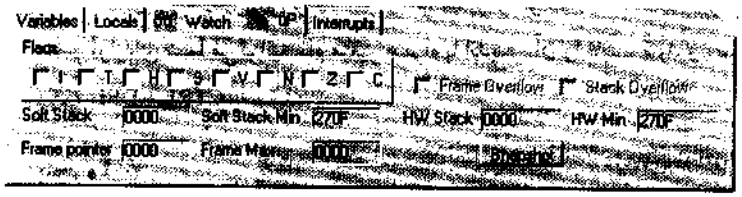
گزینه WATCH

پنجره WATCH (پنجره ۴-۹) می‌تواند برای وارد کردن وضعیتی که قرار است در خلال شبیه‌سازی ارزیابی شود مورد استفاده قرار گیرد و هنگامی که وضعیت مورد نظر صحیح شد شبیه‌سازی در حالت PAUSE قرار خواهد گرفت. حالت مورد نظر را در مکان متن تایپ نموده و دکمه ADD را کلیک کنید. هنگامی که دکمه MODIFY کلیک شود، وضعیت مورد نظر را مورد بازنگری قرار می‌دهد و می‌توان ارزش آن را تغییر داد. برای حذف هر وضعیت شما باید آن را انتخاب کرده و دکمه REMOVE را کلیک دهید.

پنجره ۴-۹
گزینه WATCH

گزینه UP

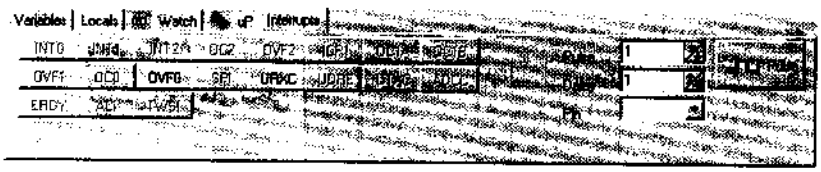
این گزینه (پنجره ۴-۱۰) وضعیت رجیستر وضعیت (STATUS REG) را نشان می‌دهد. FLAG ها (پرچمها) را می توان توسط کلیک بر روی CHECK BOX ها تغییر وضعیت داد.



پنجره ۴-۱۰
نمایش رجیستر وضعیت

گزینه INTERRUPTS

این گزینه (پنجره ۴-۱۱) منابع وقفه (INTERRUPT) را نشان می‌دهد. هنگامی که هیچ ISR (INTERRUPT SERVICE ROUTINE) برنامه نویسی نشده باشد، همه دکمه‌ها غیرفعال خواهند بود و اگر ISR نوشته شود، دکمه مربوط به آن فعال می‌شود و با کلیک بر روی هر کدام از این دکمه‌ها مثلا INT0 برنامه وقفه مربوطه اجرا می‌شود. در ضمن شما می‌توانید روی یک پایه خاص پالس نیز ایجاد نمایید.



پنجره ۴-۱۱
نمایش انواع وقفه‌ها

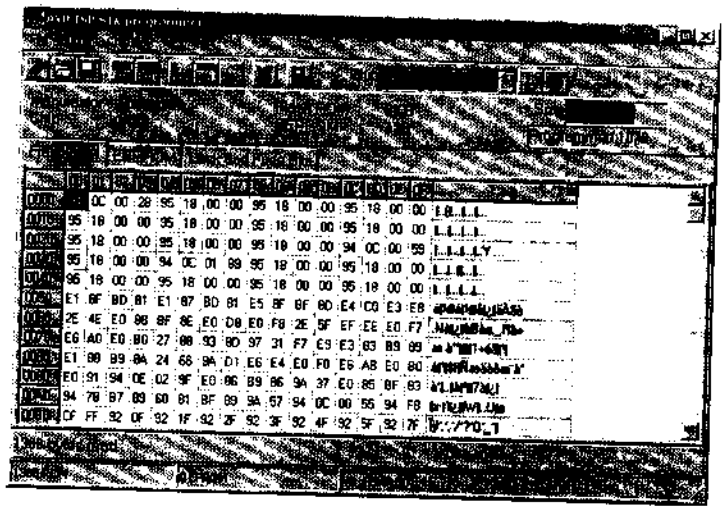
۴-۳ معرفی محیط برنامه‌ریزی STK200/300

پنجره ارسال برنامه به میکرو هنگامی که RUN PROGRAMMER انتخاب می‌شود ظاهر می‌شود. با کلیک بر روی این منو و یا با فشردن کلید F4 پنجره نشان داده شده در پنجره ۴-۱۲ نمایان خواهد شد. در این بخش قصد داریم به معرفی منوهای محیط برنامه‌ریزی بپردازیم. در صورتی که کامپایلر نتواند میکرو متصل به PROGRAMMER را شناسایی کند پنجره ۴-۱۳ نشان داده خواهد شد. زمانی که میکرو متصل شده به PROGRAMMER با میکرو که برای برنامه نوشته شده است مطابقت نداشته باشد پنجره ۴-۱۴ نمایش داده می شود.

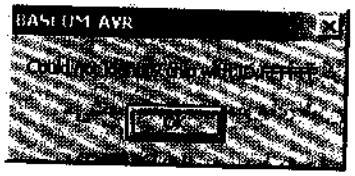
- **منوی FILE**

خروج از محیط برنامه‌ریزی

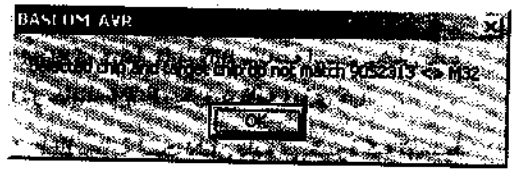
TEST : با این گزینه شما می‌توانید پایه‌های پورت LPT را یک کنید. این گزینه فقط برای زمانی که شما از SAMPLE ELECTRONIC PROGRAMEEER استفاده می‌کنید به کاربرده می‌شود.



پنجره ۴-۱۲
پنجره محیط برنامه‌ریزی



پنجره ۴-۱۳



پنجره ۴-۱۴

- منوی **BUFFER**

BUFFER CLEAR : این گزینه بافرFLASH ROM را پاک می کند.

LOAD FROM FILE : با این گزینه می‌توان بافر را با فایلی دلخواه پر کرد و آن را در حافظه میکرو برنامه‌ریزی کرد.

SAVE TO FILE : توسط این گزینه می توان بافر را در فایلی دلخواه ذخیره کرد. بافر می‌تواند محتوای حافظه یک میکرو باشد.

- منوی **CHIP**

CHIP IDENTIFY : با این گزینه می‌تو ن میکرو متصل شده به PROGRAMMER را شناسایی کرد.

WRITE BUFFER TO CHIP : توسط این گزینه می‌توان محتوای بافر را در حافظه ROM یا EEPROM میکرو برنامه‌ریزی کرد.

READ CHIPCODE INTO BUFFER : با این گزینه می‌توان حافظهFLASHROM میکرو را خواند و در بافر قرار داد.

BLANK CHECK : خالی بودن حافظه FLASH میکرو را مشخص می‌کند.

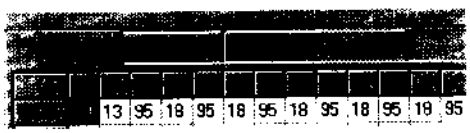
ERASE : این گزینه محتوای حافظه برنامه و داده EEPROM را پاک می‌کند.

VERIFY : این گزینه محتوای بافر و آنچه که در میکرو برنامه‌ریزی شده است را مقایسه می‌کند و در صورت تساوی پیغام VERIFY OK نمایش داده می‌شود.

AUTO PROGRAM : این گزینه حافظه میکرو را پاک کرده و برنامه موردنظر را در حافظه FLASH برنامه‌ریزی می‌کند و سپس عمل VERIFY را به صورت خودکار انجام می‌دهد.

RESET :این گزینه میکرو متصل به PROGRAMMER را ری‌ست می‌کند.

نوار زیر نیز به ترتیب بافر حافظه های FLASHROM ، EEPROM و بیت‌های LOCK AND FUSE را نشان می‌دهد.



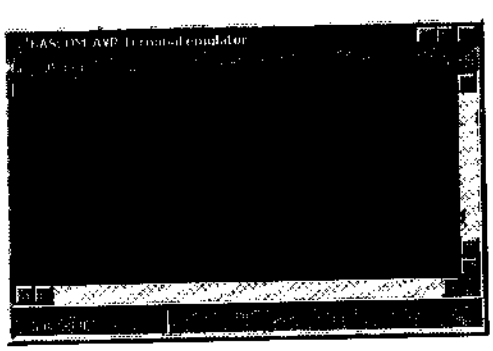
FLASHROM : با انتخاب این گزینه بافر مربوط به حافظه ROM که شامل فایل BIN برنامه است نشان داده می‌شود که توسط منوهای شرح داده شده می توان آن را در حافظه ROM میکرو برنامه ریزی کرد.

EEPROM : بافر مربوط به حافظه EEPROM که شامل فایل EEP برنامه است نشان داده می شود که توسط منوهای شرح داده شده می توان آن را در حافظه EEPROM میکرو برنامه ریزی کرد.

LOCK AND FUSE BITS : با این گزینه شما می‌توانید در صورت سالم بودن میکرو بیت‌های قفل و فیوز بیت‌ها را برنامه‌ریزی کنید. توسط دکمه فرمان WRITE LB می‌توان LOCK BITS را برنامه‌ریزی کرد. توسط کلیدهای WRITE FS ، WRITE FSH و WRITE FSE به ترتیب می‌توان LOW BYTE FUSE BITS ، HIGH BYTE FUSE BITS و EXTENDED BYTE FUSE BITS را برنامه‌ریزی کرد.

۴-۴ معرفی محیط TERMINAL EMULATOR

از این گزینه می‌توانید برای نمایش داده ارسالی و دریافتی در ارتباط سریال RS -232 بین میکرو و کامپیوتر استفاده کنید. (پنجره ۴-۱۵)



پنجره ۴-۱۵

محیط TERMINAL EMULATOR

اطلاعاتی که شما در این محیط تایپ می کنید به میکرو ارسال و اطلاعاتی که از پورت کامپیوتر دریافت می‌شود در این پنجره نمایش داده می‌شود. هنگامی که شما در برنامه از SERIAL IN و یا SERIAL OUT استفاده می کنید، می‌توانید پس از PROGRAM کردن میکرو و اتصال آن به پورت سریال PC ، داده‌های ارسالی توسط UART میکرو به بیرون را دریافت کرده و نمایش داد و از صحت وسقم آنها اطلاع یافت. همچنین اگر از دستوری مانند INKEY در برنامه استفاده کرده‌اید می‌توانید داده سریال خود را از طریق پنجره TERMINAL EMULATOR به میکرو بفرستید. توجه داشته باشید که شما باید از نرخ انتقال BAUD مشابه در میکرو و کامپیوتر استفاده نمایید. به طور مثال اگر از BAUD برابر با 9600 استفاده می کنید بایستی در گزینه COMMUNICATION SETTING نیز BAUD برابر با 9600 را انتخاب کنید. همچنین نرخ انتقال را در فایل REPORT نیز می‌توان مشاهده کرد.

- **منوهای محیط TERMINAL EMULATOR**

FILE UPLOAD : برنامه جاری در فرمت HEX را UPLOAD می‌کند.
FILE ESCAPE : صرفنظر کردن از UPLOAD کردن فایل
FILE EXIT : خروج از برنامه EMULATOR
TERMINAL CLEAR : محتوای پنجره ترمینال را پاک می‌کند.
TERMINAL OPEN LOG : فایل LOG را باز یا بسته می‌کند. هنگامی که فایل LOG وجود نداشته باشد از شما درخواست نامی برای باز کردن فایل گزارش می‌کند. تمام اطلاعاتی که در پنجره TERMINAL پرینت می‌شود داخل فایل LOG ثبت می‌شود.
SETTING : توسط این منو می‌توانید تنظیمات پورت COM و دیگر OPTION ها را انجام دهید.

۴-۵ ساخت STK200/300 PROGRAMMER

در این بخش قصد داریم به ساخت چند نوع STK200/300 PROGRAMMER توسط بافر 74HC244 بپردازیم. این نوع PROGRAMER از ارتباط SPI برای برنامه‌ریزی میکرو استفاده می‌کند در نتیجه میکروهایی که قابلیت ارتباط SPI را دارا هستند، می‌توان با آن برنامه‌ریزی کرد.

همانطور که در شکل۴-۱ نمایش داده شده است خروجی بافر به پایه‌های (POWER)VCC، RESET ، MISO ، MOSI ، SCK(CLOCK) و GND از میکرو اتصال می‌یابد.

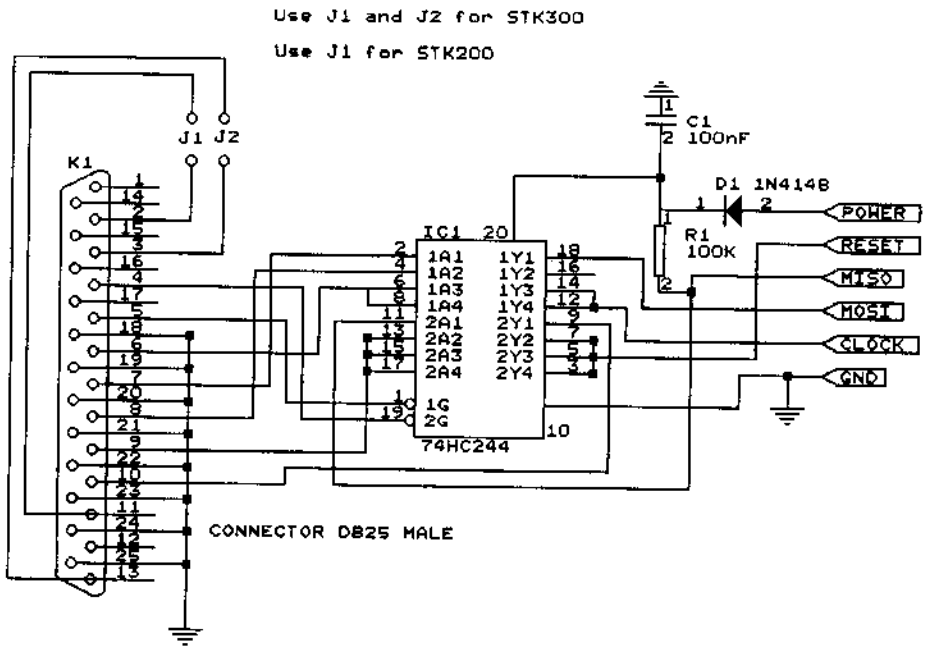
PROGRAMMER شکل ۴-۱ را می‌توان به صورت مدار شکل ۴-۲ بهینه و به صورت مدار شکل ۴-۳ساده کرد(این مدار به دلیل نداشتن امنیت برای پورت کامپیوتر توصیه نمی شود).

نکته

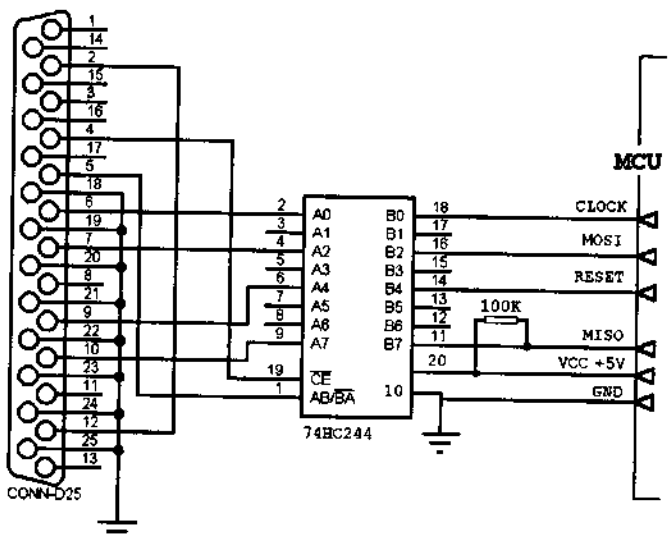
در میکروهای نوع ATMEGA64 و ATMEGA128 وATMEGA103 بایستی MISO از پروگرامر را به پایه TXD0 و MOSI از پروگرامر را به RXD0 از میکرو متصل کرد و بقیه پایه ها همانند بقیه میکروها به پایه های مربوطه اتصال می یابد.

نکته

در صورت استفاده از این نوع PROGRAMMER ها، بایستی در منوی OPTION و گزینه PROGRAMMER نوع STK200/300 PROGRAMMER را انتخاب نمایید.



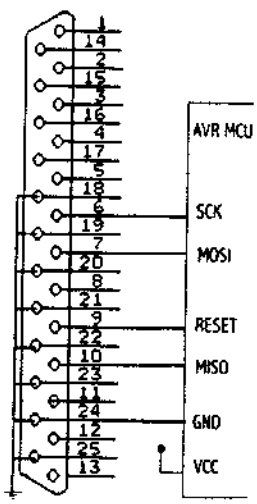
شکل ۴-۱ مدار STK200/300 PROGRAMER



شکل ۴-۲

مدار STK200/300 PROGRAMER

بهینه شده



شکل ۴-۳

مدار STK200/300 PROGRAMER

ساده شده

۵

دستورات و توابع محیط برنامه‌نویسی BASCOM

کامپایلر BASCOM دارای دستورات فراوانی است که در این فصل قصد داریم اکثر دستورات را معرفی و بررسی نماییم. از آنجا که در آموزش ارائه مثال باعث یادگیری و تسلط بیشتر می‌شود در این فصل سعی کرده‌ایم بعد از معرفی هر دستور با نوشتن برنامه‌ای، کاربرد و چگونگی کار با دستور را نشان دهیم. در بخش اول دستوراتی که برای نوشتن یک برنامه مورد نیاز است معرفی شده اند. در بخش دوم با دستوراتی که با اعداد و متغییرها سروکار دارند آشنا می‌شوید. در مورد توابع ریاضی و محاسباتی در بخش سوم توضیح داده شده است. بخش چهارم، معرفی دستورات و توابع تبدیل کدها و متغییرها به یکدیگر است. رجیستر و خانه های حافظه در بخش پنجم معرفی شده اند. بخش ششم شامل معرفی دستورالعملهای حلقه و پرش است. در بخش هفتم دستوراتی که برای ایجاد تأخیر در برنامه در یک برنامه مورد نیاز است معرفی شده اند بالاخره بخش هشتم، معرفی زیربرنامه و تابع است.

اهداف

۱. آشنایی کامل با تمام دستورات BASCOM

۲. تحلیل کردن برنامه‌های نوشته شده توسط SIMULATOR داخلی BASCOM

۵-۱ بدنه یک برنامه در محیط BASCOM

بدنه یک برنامه بیسیک در محیط BASCOM شامل تعیین میکرو مورد استفاده، کریستال ،پایان و گزینه‌های اختیاری دیگری است که در زیرمعرفی شده‌اند.

معرفی میکرو

$REGFILE = VAR

برای شروع یک برنامه در محیط BASCOM ابتدا بایستی میکرو موردنظر تعریف شود. VAR نام چیپ مورد استفاده است که می‌تواند یکی از موارد زیر باشد.

```
$regfile = " Attiny2313.dat"      'ATtiny2313 MCU
$regfile = " Attiny13def.dat"     'ATtiny13 MCU
$regfile = " At12def.dat"         'ATtiny12 MCU
$regfile = " At15def.dat"         'ATtiny15 MCU
$regfile = " At22def.dat"         'ATtiny22 MCU
$regfile = " At26def.dat"         'ATtiny26 MCU

$regfile = "1200def.dat"          'AT90S1200 MCU
$regfile = "2313def.dat"          'AT90S2313 MCU
$regfile = "2323def.dat"          'AT90S2323 MCU
$regfile = "2333def.dat"          'AT90S2333 MCU
$regfile = "2343def.dat"          'AT90S2343 MCU
$regfile = "4414def.dat"          'AT90S4414 MCU
$regfile = "4433def.dat"          'AT90S4433 MCU
$regfile = "4434def.dat"          'AT90S4434 MCU
$regfile = "8515def.dat"          'AT90S8535 MCU
$regfile = "8535def.dat"          'AT90S8535 MCU

$regfile = " M128103.dat"         'MEGA128 IN MEGA103 MODE MCU
$regfile = " M8535.dat"           'MEGA 8535 MCU
$regfile = " M8515.dat"           'MEGA8515 MCU
$regfile = " M603def.dat"         'MEGA603 MCU
$regfile = " M323def.dat"         'MEGA323 MCU
$regfile = " M169def.dat"         'MEGA169 MCU
$regfile = " M163def.dat"         'MEGA163 MCU
$regfile = " M162def.dat"         'MEGA162 MCU
$regfile = " M161def.dat"         'MEGA161 MCU
$regfile = " M128def.dat"         'MEGA 128 MCU
$regfile = " M103def.dat"         'MEGA103 MCU
$regfile = " M88def.dat"          'MEGA88 MCU
$regfile = " M64def.dat"          'MEGA64 MCU
$regfile = " M48def.dat"          'MEGA48 MCU
$regfile = " M32def.dat"          'MEGA 32 MCU
$regfile = " M16def.dat"          'MEGA16 MCU
$regfile = " M8def.dat"           'MEGA 8 MCU
```

کریستال

برای مشخص کردن فرکانس کریستال استفاده شده بر حسب هرتز از دستور زیر استفاده می‌نماییم.

$CRYSTAL =X

X فرکانس کریستال استفاده شده برحسب هرتز است.

نکته این دستور را حتی برای زمانی که با اسیلاتور داخلی میکرو کار می‌کنید بر بسید.

• مثال

```
$CRYSTAL = 14000000               '14MHZ external osc
$CRYSTAL = 8000000                '8MHZ external osc
$CRYSTAL = 1000000                '1MHZ internal osc
```

اسمبلی و بیسیک (اختیاری)

برای نوشتن برنامه اسمبلی در بین برنامه بیسیک از دستور زیر استفاده می‌نماییم.

```
$ASM
      ASSEMBLY PROGRAMME
$ENDASM
```

با دستور $ASM می‌توان در برنامه بیسیک شروع به نوشتن برنامهٔ مورد نظر اسمبلی کرده و پس از اتمام برنامه اسمبلی با دستور $ENDASM برنامهٔ اسمبلی را به پایان رساند و به نوشتن ادامهٔ برنامه بیسیک پرداخت.

• مثال

```
Dim C As Byte
Loadadr C , X              'load address of variable C into register X
 $Asm                      'start assembly program
   Ldi R24,1               'load register R24 with the constant 1
   St X,R24                'store 1 into var c
 $End Asm                  'end of assembly program
Print C                    'send c to serial port
End
```

یادداشت (اختیاری)

گاهی نیاز است یادداشتهایی برای اطلاعات بیشتر در برنامه اضافه کنید.

REM یا '

یادداشتها و نوشته‌های بعد از این دستور غیر فعال بوده و در برنامه برای یادداشت به کار می‌رود و کامپایل نخواهند شد وهمچنین به رنگ سبز در می‌آیند.

شما می‌توانید از دو علامت)' و (' برای شروع و اتمام متن یادداشتی استفاده نمایید.

همچنین شما از _ (UNDER LINE) برای ادامه دستورات و توابع در خط پایین‌تر می‌توانید استفاده کنید.

• مثال

```
Config Lcdpin = Pin , Db4 = Pinb.4 , Db5 = Pinb.5 , Db6 = Pinb.6  , _
Db7 = Pinb.7 , Rs = Pinb.2 , E = Pinb.3
```

• مثال

```
'(  start block comment                شروع متن یاداشتی
      This will not be compiled
')  end block comment                 پایان متن یاداشتی
```

• مثال

```
REM this sentence will not be compiled
Or you can write as below
'This sentence will not be compiled
```

آدرس شروع برنامه‌ریزی حافظه FLASH (اختیاری)

گاهی نیاز است که برنامه خود را از آدرسی دلخواه در حافظه FLASHROM قرار دهید.

$ROMSTART = ADDRESS

ADDRESS مکانی از حافظه است که برنامهٔ HEX از این آدرس در حافظه میکروکنترلر قرار می‌گیرد. در صورتی که از این دستور استفاده نشود کامپایلر به طور خودکار آدرس &H0000 را در نظر می‌گیرد.

• مثال

```
$ROMSTART = &H4000
```

تعیین کلاک (اختیاری)

با این دستور در بعضی از میکروهای سری MEGA AVR از جمله MEGA103 یا MEGA603 به صورت نرم‌افزاری می‌توان کلاک سیستم را تغییر داد. تقسیم کلاک به طور مثال برای کاهش مصرف تغذیه استفاده می‌شود.

CLOCKDIVISON = var

Var مقادیر معتبر بین اعداد 2 تا 128 می‌تواند باشد.

نکته

اگر شما از این دستور استفاده نمایید ، دستوراتی که مستقیماً با کلاک سیستم کار می‌کنند ممکن است درست کار نکنند . بطور مثال WAITms var دو برابر طول می‌کشد زمانی که از CLOCKDIVISON = 2 استفاده می‌نمایید.

• مثال

```
$baud = 2400
Clockdivision = 2
Print "Hello"
End
```

پایان برنامه

END

این دستور در انتهای برنامه قرار می‌گیرد و اجرای برنامه را متوقف می‌کند. با دستور END تمام وقفه‌ها غیر فعال شده و یک حلقه بی‌نهایت تولید و برنامه خاتمه می‌یابد.

• مثال

```
PRINT "Hello"              'print this
END                        'end program execution and disable all interrupts
```

۵-۲ اعداد و متغییرها و جداول LOOKUP

دیمانسیون متغییر

این دستور بُعد یک متغییر را نشان می‌دهد. با این دستور می‌توانید متغییرهایی که در برنامه به کار برده می‌شوند تعریف کنید.

DIM var AS [XRAM/SRAM/ERAM] data type [AT location] [OVERLAY]

VAR نام متغییری است که در برنامه به کار برده می‌شود. در صورت استفاده از حافظه جانبی آن را با XRAM مشخص کنید و SRAM را زمانی اختیار کنید که می‌خواهید متغییرها را در حافظه SRAM قرار دهید و ERAM متغییر مورد نظر را در EEPROM داخلی جای می‌دهد. data type نوع داده است که می‌تواند طبق جدول زیر BIT ، BYTE ، INTEGER ، LONG ، WORD ، STRING یا SINGLE باشد. در صورت استفاده از متغییر STRING ، بیشترین طول آن نیز باید نوشته شود . گزینه اختیاری OVERLAY متغییر تعریف شده را به صورت POINTER در نظر گرفته و فضایی را برای متغییر در نظر نمی‌گیرد.

DATA TYPE	STORES AS	VALUE RANGE
BIT	A BIT	0 OR 1
BYTE	UNSIGNED 8-BITS	0 TO 255
INTEGER	SIGNED 16-BITS	-32767 TO 32767
WORD	UNSIGNED 16-BIT	0 TO 65535
LONG	SIGNED 32-BITS	-2147483648 TO 214783647
SINGLE	SIGNED 32-BITS	1. 5x 10^– 45 to 3.4 x 10^38
STRING	0 – 254 BYTES	-

جدول انتخاب نوع داده

DIM S AS STRING*10 متغییر STRING به نهایت طول 10 کاراکتر

AT LOCATION به شما اجازه می‌دهد که متغییرتان را در آدرسی که می‌خواهید در حافظه ذخیره کنید. زمانی که محل آدرس‌دهی اشغال باشد ، اولین جای خالی در حافظه استفاده می‌شود.

نکته عدد HEX را با علامت &H و عدد BINARY را با علامت &B مشخص می‌کنیم.

• مثال

```
A=&H01DE          'HEX NUM
B=&B01011011      'BIN NUM
```

نکته می‌توان به جای استفاده از دستور DIM از دستور DEF××× استفاده نمایید.

```
Dim var As Bit          is equal to     DEFBIT var
Dim var as Byte         is equal to     DEFBYTE var
Dim var as Integer      is equal to     DEFINT var
Dim var As Word         is equal to     DEFWORD var
Dim var As Single       is equal to     DEFSNG var
Dim var As Long         is equal to     DEFLNG var
```

• مثال

```
Dim B1 As Bit                   'Bit Can Be 0 Or 1
Dim A As Byte                   'Byte Range From 0 -255
Dim C As Integer                'Integer Range From -32767 - + 32768
DEFLNG  L                       'Dim L As Long
Dim W As Word
Dim S As String * 11            'Length Can Be Up To 11 Characters
Dim K As Integer At 120          'You Can Specify The Address Of The Variabl
Dim Kk As Integer               'The Next Dimensioned Variable Will Be Placed After
'Variable S
B1 =  1                          'Assign Bits
Set  B1                         'is equal with B1 = 1
A = 12                          'Assign Bytes
A = A + 1
C = -12                         'Assign Integer
C = C + 100
Print C
W = 50000
Print W
L = 12345678                    'Assign Long
Print L
S = "Hello world"               'Assign Long
Print S
End
```

• مثال

```
Dim b1 as Byte at $60 OVERLAY
Dim b2 as Byte at $61 OVERLAY
```

دستور CONST

برای تعریف یک ثابت از این دستور استفاده می‌شود.

```
CONST SYMBOL = NUMCONST
CONST SYMBOL = STRINGCONST
CONST SYMBOL = EXPRESSION
```

SYMBOL نام ثابت و NUMCONST مقدار عددی انتساب یافته به SYMBOL ، STRINGCONST رشته انتساب یافته به SYMBOL و EXPRESSION می‌تواند عبارتی باشد که نتیجه آن به SYMBOL انتساب یابد.

• مثال

```
CONST S = "TEST"                'DECLARE A AS A CONSTANT
CONST A = 5                     'OR USE AN EXPRESSION TO ASSIGN A CONSTANT
CONST B1 = &B1001
CONST X = (B1 * 3 ) + 2
CONST SSINGLE = SIN(1)
DIM B AS BYTE
B= 6 * A                        'B=6*5
```

دستور ALIAS

از این دستور برای تغییر نام متغییر استفاده می‌شود.

• مثال

```
DIRECTION ALIAS PORTB.1
```

حال شما می‌توانید به جای PORTB.1 از متغییر DIRECTION استفاده نمایید.

```
SET DIRECTION                ' is equal with    set portb.1
```

دستور CHR

از این دستور برای تبدیل متغییر عددی یا یک ثابت به کاراکتر استفاده می‌شود. از این دستور می‌توانید زمانی که قصد دارید یک کاراکتر بر روی LCD نمایش دهید استفاده نمائید. در صورتی که از این به دستور به صورت PRINT CHR (VAR) استفاده نمائید ،کاراکتر اسکی VAR به پورت سریال ارسال می‌شود.

• مثال

```
Dim a As Byte        'dim variable
A=65                  'assign variable
Print a              'print value (65)
Print HEX(a)         'print hex value (41)
Print Chr(a)         'print ASCII character 65 (A)
End
```

دستور INSTR

این دستور محل و موقعیت یک زیر رشته را در رشته دیگر مشخص می‌کند.

```
Var = Instr( start , String , Substr )
Var = Instr( string , Substr )
```

VAR عددی است که مشخص کننده محل SUBSTR در رشته اصلی STRING می‌باشد و زمانی که زیر رشته مشخص شده در رشته اصلی نباشد صفر برگردانده می‌شود. START نیز عددی دلخواه است که مکان شروع جستجو زیر رشته در رشته اصلی را مشخص می‌کند. در صورتیکه START قید نشود تمام رشته از ابتدا جستجو می‌شود. رشته اصلی تنها باید از نوع رشته باشد ولی زیر رشته (SUBSTR) می‌تواند رشته و عدد ثابت هم باشد.

• مثال

```
Dim S As String * 15 , Z As String * 5
Dim Bp As Byte
S = "This is a test"
Z = "is"
Bp = Instr(s , Z) : Print Bp              'should print 3
Bp = Instr(4 , S , Z) : Print Bp          'should print 6
End
```

دستور INCR

```
INCR  var
```

این دستور یک واحد به متغییر عددی var می‌افزاید.

• مثال

```
Do                          'start loop
Incr A                      'increment a by 1  A=A+1
Print A                     'print a
Loop Until A > 10           'repeat until a is greater than 10
```

دستور DECR

DECR var

این دستور متغییر VAR را یک واحد کم می‌کند.

• مثال

```
Dim A As Byte
A = 5                                   'assign value to a
Decr A                                  'decr by one  A=A-1
Print A                                 'print it   A=4
End
```

دستور CHECKSUM

این دستور مجموع کد دسیمال اسکی رشته VAR را بر می‌گرداند که البته اگر مجموع کد اسکی رشته از عدد 255 بیشتر شود مقدار 256 از مجموع کم میشود.

• مثال

```
Dim S As String * 10          'Dim Variable
S="test"                    'Assign Variable
Print Checksum(s)           'Print Value(192)
S = "testnext"              'Assign Variable
Print Checksum(s)           'Print Value 127 (127=383-256)
End
```

دستور LOW

این دستور LSB (least significant byte) یک متغییر را برمی‌گرداند.

Var = LOW(s)

LSB متغییر S در Var قرار می‌گیرد.

• مثال

```
Dim I As Integer , Z As Byte
I = &H1001
Z = Low(i)                    ' is 1
End
```

دستور HIGH

این دستور پرارزش‌ترین بایت (MSB) یک متغییر را برمی‌گرداند.

var = HIGH(s)

مقدار MSB متغییر S در var جای می‌گیرد.

• مثال

```
Dim I As Integer , Z As Byte
I = &H1001
Z = High(i)           'Z Is 16  Z=&H10
I = &H1101
Z = High(i)           'Z Is 17 Z=&H11
I = 1012
Z = High(i)           ' I=&H3F4    , Z Is 3
End
```

دستور LCASE

این دستور تمام حروف رشته مورد نظر را تبدیل به حروف کوچک می‌کند.

```
Target = Lcase(source)
```

تمام حروف رشته source کوچک شده و در رشته Target جای داده می‌شود.

• **مثال**

```
Dim S As String * 12 , Z As String * 12
S = "Hello World"
Z = Lcase( s )              'Z=hello world
Print Z
End
```

دستور UCASE

این دستور تمام حروف رشته مورد نظر را تبدیل به حروف بزرگ می‌کند.

```
Target = Ucase(source)
```

تمام حروف رشته source بزرگ شده و در رشته Target جای داده می‌شود.

• **مثال**

```
Dim S As String * 12 , Z As String * 12
S = "Hello World"
Z = Ucase( s )              'Z=HELLO WORLD
Print Z
End
```

دستور RIGHT

با این دستور قسمتی از یک رشته را جدا می‌کنیم.

```
var = RIGHT( var1 , n )
```

از سمت راست رشته var1 ، تعداد n کاراکتر جدا شده و در رشته var قرار می‌گیرد.

• **مثال**

```
Dim S As String * 15 , Z As String * 15
S = "ABCDEFG"
Z = Right(s , 2)
Print Z                  'FG
End
```

دستور LEFT

این دستور کاراکترهای سمت چپ یک رشته را به تعداد تعیین شده جدا می‌کند.

```
var = Left( var1 , n )
```

از سمت چپ رشته var1 ، تعداد n کاراکتر جدا شده و در رشته var قرار می‌گیرد.

• **مثال**

```
Dim S As String * 15 , Z As String * 15
S = "abcdefg"
Z = Left(s , 5)
Print Z                          'abcde
Z = Left(s , 1)
Print Z                          'a
End
```

دستور LEN

این دستور طول یا به عبارتی تعداد کاراکترهای یک رشته را برمی‌گرداند.

var = LEN(string)

طول رشته STRING درمتغییر عددی VAR قرار می‌گیرد. رشته STRING نهایتاً 255 بایت می‌تواند طول داشته باشد. توجه داشته باشید که فضای خالی (SPACE BAR) خود یک کاراکتر به حساب می‌آید.

• مثال

```
Dim S As String * 12
Dim A As Byte
S = "test"
A = Len( s )
Print A                          'prints 4
Print Len(s)                    'prints 4
S = "test "
A = Len( s )
Print A                          'prints 5
End
```

دستور LTRIM

این دستور فضای خالی رشته را حذف می‌کند.

var = LTRIM(org)

فضای خالی رشته org برداشته می‌شود (حذف می‌شود) و رشته بدون فضای خالی در متغییر رشته‌ای var قرار می‌گیرد.

• مثال

```
Dim S As String * 10
S = "    AB "
Print Ltrim(s)        'AB
S = "    A B "
Print Ltrim(s)        'A B
End
```

دستورSWAP

SWAP var1 , var2

با اجرای این دستور محتوای متغییر var1 در متغییر var2 ومحتوای متغییر var2 در متغییر var1 قرار می‌گیرد.Var1 و var2 می‌توانند داده‌هایی از نوع BIT ، BYTE، INTEGER، WORD، LONG یا STRING باشند.

نکته دو متغییرVar1 و var2 بایستی از یک نوع داده باشند.

• مثال

```
Dim A As Integer , B1 As Integer
A = 1 : B1 = 2                   'assign two integers
Swap A , B1                      'swap them
Print A                          'prints 2
Print B1                           'prints 1
End
```

دستور MID

با این دستور می‌توان قسمتی از یک رشته را برداشت و یا قسمتی از یک رشته را با قسمتی از یک رشته دیگر عوض کرد.

۱-Var = Mid(var1 , St [, L])
۲-Mid(var , St [, L]) = Var1

۱- قسمتی از رشته var1 ، با شروع از کاراکتر St ام و طول L برداشته شده و در متغییر Var قرار می‌گیرد.
۲- رشته Var1 در رشته var با شروع از کاراکتر St ام و طول L قرار می‌گیرد.
در صورت قید نکردن گزینه اختیاری L ، بیشترین طول در نظر گرفته می‌شود.

• مثال

```
Dim S As Xram String * 15 , Z As Xram String * 15
S = "ABCDEFG"
Z = Mid(s , 2 , 3)
Print Z                              'BCD
Z = Mid(s , 2 , 1)
Print Z                              'B
Z = "12345"
Mid(s , 2 , 2) = Z
Print S                              'A12DEFG
End
```

دستور ROTATE

دستور زیر تمام بیتها را به چپ یا راست منتقل می‌کند ولی تمام بیتها محفوظ هستند و هیچ بیتی بیرون فرستاده نمی‌شود.

ROTATE var , LEFT/RIGHT [, shifts]

var می‌تواند داده‌ای از نوع BYTE ، INTEGER ، WORD یا LONG باشد. LEFT/RIGHT جهت چرخش بیتها و SHIFT که اختیاری می‌باشد تعداد چرخش بیتها را مشخص می‌کند. در صورت قید نکردن shifts به صورت پیش فرض مقدار یک در نظر گرفته می‌شود.

• مثال

```
Dim A As Byte
a = 128
Rotate A , Left , 2
Print A                    'A=2
```

دستور SPACE

برای ایجاد فضای خالی از این دستور استفاده می‌شود.

var = SPACE(x)

x تعداد فضای خالی است که به عنوان رشته در متغییر رشته‌ای var جای می‌گیرد.

• مثال

```
Dim S As String * 15 , Z As String * 15
```

```
S = Space(5)
Print  "{" ; S ; "}"              '{     } 5 space
Print  "{" ; Space(6) ; "}"       '{      } 6 space
```

تابع FORMAT

این دستور یک رشته عددی را شکل‌دهی (FORMAT) می‌کند.

target = Format (source , "mask")

source رشته‌ای است که شکل‌دهی شود و نتایج در target قرار می‌گیرد. mask نوع شکل‌دهی است که برای درک بیشتر به مثال زیر توجه کنید.

• مثال

```
Dim S As String * 10
Dim I As Integer
S= " 123 "
S=Format( s , "      ")      '5 space
Print S                      'S= "  123"   2 space first , then 123
S =  " 12345 "
S = Format(s, " + ")
Print S                      'S= +12345
S = "123"
S = Format(s , "00000")
Print S                      'S=00123
S = "12345"
S = Format(s , "000.000")
Print S                      'S=012.345
S = "12345"
S = Format(s , "000.00")
Print S                      'S=123.45
S = "12345"
S = Format(s , " +000.00")
Print S                      'S=+123.45
End
```

تابع FUSING

از این دستور برای روند کردن رشته‌های عددی استفاده می‌شود.

target = Fusing(source , "mask")

SOURCE رشته مورد نظر برای شکل‌دهی و MASK نوع شکل‌دهی است. نتایج این شکل‌دهی درTARGET قرار می‌گیرد. عمل MASK حتماً باید با علامت # شروع شود و حداقل باید یکی از علامتهای # و & را بعد از ممیز داشته باشد. علامتهای # و & با یکدیگر بعد از ممیز استفاده نمی‌شود. با استفاده ازعلامت # عدد روند می‌شود و در صورت استفاده ازعلامت & روندی صورت نمی‌گیرد.

• مثال

```
Dim S As Single , Z As String * 10  'Now Assign A Value To The Single
S = 123.45678
Print Fusing(s , "#.##")            'Prints 123.46
'Now Use Some Formatting With 2 Digits Behind The Decimal Point Without
'Rounding
Print Fusing(s , "#.&&" )           'Prints 123.45
'(The Mask Must Start With.#
It Must Have At Least One # Or & After The Point
You May Not Mix & And # After The Point ')
End
```

جدول LOOKUP

توسط این جدول می‌توان مقدار دلخواهی را از جدولی برگرداند.

var =LOOKUP(value , label)

lable برچسب جدول و value اندیس داده دلخواه در جدول است. داده برگشتی از جدول در متغییرvar قرار می‌گیرد. Value = 0 اولین داده در جدول را برمی‌گرداند. تعداد اندیس‌ها و مقدار داده برگشتی به ترتیب نهایتاً می‌توانند 255 و 65535 باشند.

نکته داده دو بایتی داخل جدول بایستی با علامت % پایان یابد.

• مثال

```
Dim B1 As Byte , I As Integer
B1 = Lookup(2 , Dta)
Print B1                        ' Prints 3 (zero based)
I = Lookup(0 , Dta2)
Print I                         ' print 1000
End

Dta:
Data 1 , 2 , 3 , 4 , 5

Dta2:
Data 1000 % , 2000%
```

جدول LOOKUPSTR

توسط این جدول می‌توان رشته دلخواهی را از جدولی برگرداند.

var =LOOKUPSTR(value , label)

lable برچسب جدول و value اندیس رشته دلخواه در جدول است. رشته برگشتی از جدول در متغییر رشته‌ای var قرار می‌گیرد. Value = 0 اولین رشته در جدول را برمی‌گرداند. تعداد اندیس‌ها نهایتاً می‌تواند 255 باشد.

• مثال

```
Dim S As String * 4 , Idx As Byte
Idx = 0 : S = Lookupstr(idx , Sdata)
Print S                 'will print ' This '
End

Sdata:
Data "This" , "is" , "a test"
```

۳-۵ توابع ریاضی و محاسباتی

عملگرهای ریاضی

از عملگرهای ریاضی زیر می‌توانید در محیط BASCOM استفاده نمایید و عملیات ریاضی خود را راحتر انجام دهید.

نماد		علامت
*	Asterisks (multiplication symbol)	علامت ضرب
+	Plus sign	علامت جمع
-	Minus sign	علامت تفریق
.	Period (decimal point)	علامت ممیز
/	Slash (division symbol)	علامت تقسیم
>	Less than	علامت کوچکتر از
=	Equal sign	علامت تساوی
<	Greater than	علامت بزرگتر از
^	Exponent	علامت بتوان
=>	Less than or equal to	علامت کوچکتر یا مساوی با
>=	Greater than or equal to	علامت بزرگتر یا مساوی با
<>	Inequality	علامت مخالف

جدول عملگرهای ریاضی محیط BASCOM

عملگرهای منطقی

عملگرهای منطقی در BASCOM به صورت زیر است .

نماد عملگرهای منطقی در BASCOM	معرفی
AND	Conjunction
OR	Disjunction
XOR	Exclusive or
NOT	Logical complement

جدول عملگرهای منطقی محیط BASCOM

• مثال

```
A = 63 And 19
PRINT A                 '19 PRINTS
A = 10 Or 9
PRINT A                 '11 PRITNS
```

تابع ABS

VAR = ABS(VAR2)

این دستور به معنای ریاضی | VAR2 | = VAR (قدر مطلق) است.

• مثال

```
Dim A As Integer , C As Integer
a = -1000
C = Abs(a)                          'c= | a |
Print C                             'c=1000
End
```

تابع EXP

Target = Exp(source)

Target برابر با e به توان source است. Target متغییری از نوع داده SINGLE است.

• مثال

```
Dim  X  As   Single
X = Exp(1.1 )
Print  X                    'Prints 3.004166124
X  =  1.1
X  =   Exp( x )
Print  X                    'prints 3.004164931
```

تابع LOG10

Target = Log10 (source)

لگاریتم پایه 10 متغییر یا ثابت source در متغییر Target قرار می‌گیرد. Target و source هر دو داده نوع SINGLE هستند.

• مثال

```
Dim S1 As Single , S2 As Single
S1  =  0.01
S2  =   Log10 ( s1 )
Print  S2
   For S1 = 1 To 100
         S2  =   Log10 ( s1 )
         Print  S1  ;   "   "   ;  S2
   Next
END
```

تابع LOG

این دستور لگاریتم طبیعی یک داده از نوع SINGLE را برمی‌گرداند.

Target = Log(source)

لگاریتم متغییر یا ثابت source از نوع داده SINGLE گرفته می‌شود و در متغییر Target از نوع داده SINGLE قرار می‌گیرد. این تابع برای اجرا شدن وقت زیادی می‌برد مخصوصاً زمانی که اعداد بزرگ استفاده می‌شود. همچنین با بزرگتر شدن اعداد دقت نیز پایین خواهد آمد.

• مثال

```
Dim X As Single
X = Log( 100 )
Print X                   'prints  4.605170
X = 100
X = Log( x )
Print X                   'Prints 4.605098
X = Log( 1.1 )
Print X                     'prints 0.095310147
X = 1.1
X = Log( x )
Print X                     'prints 0.095310147
'So a smaller number is calculated faster
```

تابع RND

این دستور یک عدد تصادفی برمی‌گرداند.

var = RND(limit)

عددی تصادفی بین 0 و limit بدست آمده ودر متغییرVAR قرار می‌گیرد. با هر بار استفاده از این دستور عدد مثبت تصادفی دیگری بدست خواهد آمد.

نکته limit باید یک عدد مثبت باشد.

• **مثال**

```
Dim I As Integer
Do
   I = Rnd(100)    'get random number
   Print I
   Wait 1
Loop
End
```

تابع SIN

var = SIN (source)

این دستور سینوس(SINE) ثابت یا متغییر source را در متغییر var از نوع داده SINGLE قرار می‌دهد. تمام دستورات مثلثاتی با رادیان کار می‌کنند و ورودی این دستور بایستی به رادیان باشد.

• **مثال**

```
Dim S2 As Single
Dim Vsin As Single
Const Pi = 3.14159265358979
S2 = Pi / 2
Vsin = Sin(s2 )                    ' VSIN= SIN(P/2)
Print "VSIN="; Vsin               'VSIN=0.99999332 WILL PRINT
End                                'end program
```

تابع COS

var = COS (source)

این دستور کسینوس(COSINE) ثابت یا متغییر source از نوع داده SINGLE را در متغییر var از نوع داده SINGLE قرار می‌دهد. تمام دستورات مثلثاتی با رادیان کار می‌کنند و ورودی این دستور بایستی به رادیان باشد.

• **مثال**

```
Dim S2 As Single
Dim Vcos As Single
Const Pi = 3.14159265358979
S2 = Pi / 2
Vcos = Cos(s2 )                    ' Vcos= COS(P/2)
Print "VCOS=" ; Vcos              'VCOS= -0.000006613 WILL PRINT
End                                'end program
```

تابع TAN

var = TAN (source)

این دستور تانژانت(TANGENT) ثابت یا متغییر source از نوع داده SINGLE را در متغییر var از نوع داده SINGLE قرار می‌دهد. تمام دستورات مثلثاتی با رادیان کار می‌کنند و ورودی این دستور بایستی به رادیان باشد.

• **مثال**

```
Dim S2 As Single
Dim Vtan As Single
Const Pi = 3.14159265358979
S2 = Pi * 2
Vtan = Tan (s2 )                   ' Vtan= TAN(P/2)
Print "VTAN=" ; Vtan              'VTAN= -0.000000357 WILL PRINT
End                                'end program
```

تابع SINH

var = SINH (source)

این دستور SINE HYPERBOLE یک ثابت یا متغیر از نوع داده SINGLE را به رادیان می‌دهد. سینوس هایپربولیک source در متغییر عددی var از نوع داده SINGLE قرار می‌گیرد. شما با دستورات RAD2DEG و DEG2RAD می‌توانید مقدار بدست آمده را به درجه تبدیل نمایید.

• مثال

```
Dim S1 As Single , S2 As Single
S1 = 0.512
S2 = Sinh(s1)
Print S1 ; "    "; "    " ; S2
END
```

تابع COSH

این دستور COSINE HYPERBOLE یک ثابت یا متغیر از نوع داده SINGLE را به رادیان می‌دهد.

var = COSH (source)

کسینوس هایپربولیک source در متغییر عددی var از نوع داده SINGLE قرار می‌گیرد. شما با دستورات RAD2DEG و DEG2RAD می‌توانید مقدار بدست آمده را به درجه تبدیل نمایید.

• مثال

```
Dim S1 As Single , S2 As Single
S1 = 0.512
S2 = Cosh(s1)
Print S1 ; "    "; "    " ; S2
END
```

تابع TANH

این دستور TANGENT HYPERBOLE یک ثابت یا متغیر از نوع داده SINGLE را به رادیان می‌دهد.

var = TANH (source)

تانژانت هایپربولیک source در متغییر عددی var از نوع داده SINGLE قرار می گیرد. شما توسط دستورات RAD2DEG و DEG2RAD می‌توانید مقدار بدست آمده را به درجه تبدیل نمایید.

• مثال

```
Dim S1 As Single , S4 As Single
S1 = 0.512
S4 = Tanh(s1)
Print S1 ; "    "; "    " ; S4
END
```

تابع ASIN

این دستور ARCSINE یک ثابت یا متغییر نوع SINGLE را به رادیان بر می‌گرداند.

var = ASIN (x)

متغییر ورودی X عددی از نوع SINGLE است و می‌تواند بین 1- تا 1+ باشد. نتیجه این محاسبه که بین Π/2 و Π/2- در متغییر VAR از نوع داده SINGLE قرار می‌گیرد. اگر عدد ورودی بیشتر از 1+ باشد ، Π/2 و اگر عدد ورودی کمتر از 1- باشد، Π/2 - برگردانده می‌شود. شما با دستورات RAD2DEG و DEG2RAD می‌توانید مقدار بدست آمده را به درجه تبدیل نمایید.

• مثال

```
Dim S As Single , X As Single
X = 0.5 : S = Asin(x)          '0.523595867 WILL PRINT
Print S
End                            'end program
```

تابع ACOS

این دستور ARCCOSINE یک ثابت یا متغییر نوع SINGLE را به رادیان بر می‌گرداند.

var = ACOS (x)

متغییر ورودی X عددی از نوع داده SINGLE است و می‌تواند بین 1- تا 1+ باشد. نتیجه این محاسبه که بین 0 و Π است در متغییر VAR از نوع داده SINGLE قرار می‌گیرد. اگر عدد ورودی بیشتر از 1+ باشد ، صفر واگر عدد ورودی کمتر از 1- باشد ، Π بر گردانده می‌شود. شما با دستورات RAD2DEG و DEG2RAD می‌توانید مقدار بدست آمده را به درجه تبدیل نمایید.

• مثال

```
Dim S As Single , X As Single , Y As Single
X = 0.5 : S = Acos(x)          '1.047200557 will print  (or  60 degree)
Print S
End
```

تابع ATN

این دستور ARCTANGENT یک ثابت یا متغییر از نوع داده SINGLE را به رادیان برمی‌گرداند.

var = ATN (x)

متغییر ورودی X می‌تواند عددی از نوع داده SINGLE باشد. نتیجه این محاسبه که بین Π/2 و Π/2- است در متغییر VAR از نوع داده SINGLE قرار می‌گیرد. شما با دستورات RAD2DEG و DEG2RAD می‌توانید مقدار بدست آمده را به درجه تبدیل نمایید.

• مثال

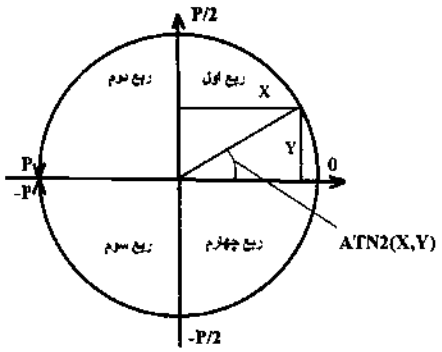
```
Dim S As Single
S = Atn(1) * 4
Print S                ' prints 3.141593
```

تابع ATN2

این دستور ARCTANGENT دو متغییر X , Y از نوع داده SINGLE را در هر چهار ربع دایره مختصات به رادیان بر می‌گرداند. دستور ATN خروجی بین Π/2 و Π/2- دارد در حالی که این دستور تمام مختصات دایره را از Π تا Π- پوشش می‌دهد.

var = ATN2 (x, y)

متغییرهای ورودی X و Y می‌توانند اعدادی از نوع داده SINGLE باشند که نمایانگر مختصات (X,Y) هستند. نتیجه این محاسبه در متغییر VAR از نوع داده SINGLE قرار می‌گیرد.



شکل خروجی دستور ATN2

QUADRANT	SING Y	SING X	ATN2
1	+	+	0 TO π/2
2	+	-	π/2 TO π
3	-	-	- π/2 TO - π
4	-	+	0 TO – π/2

جدول خروجی ATN2 به ازای علامات مختلف X و Y

شما با دستورات RAD2DEG و DEG2RAD می‌توانید مقدار بدست آمده را به درجه تبدیل نمایید.

• مثال

```
Dim  S  As   Single ,  x  As   Single,  y  As   Single
X = -0.5 : Y = -0.5
S = Atn2(x , Y)   'will print -2.356188056 radian equal to 225 OR -135 degree
Print S
End               'end program
```

تابع DEG2RAD

برای تبدیل درجه به رادیان از این دستور استفاده می‌شود.

VAR=DEG2RAD(SINGLE)

زاویه SINGLE به رادیان تبدیل می‌شود و در متغییر VAR از نوع داده SINGLE قرار می‌گیرد.

• مثال

```
Dim S1 As Single , S As Single
S1 = 180
S = Deg2rad(s1)
Print S                    '3.141592498 WILL PRINT
End                        'end program
```

تابع RAD2DEG

برای تبدیل رادیان به درجه از این دستور استفاده می‌شود.

VAR=RAD2DEG(SINGLE)

رادیان SINGLE به درجه تبدیل می‌شود و در متغیر VAR از نوع داده SINGLE قرار می‌گیرد.

• مثال

```
Dim S1 As Single , S As Single
S1 = 3.141592498
S = Rad2deg(s1)
Print S                                 '179.999984736 WILL PRINT
End                                     'end program
```

تابع ROUND

var = ROUND (x)

متغییر یا داده X از نوع SINGLE روند شده و در متغییر var از نوع داده SINGLE قرارمی‌گیرد.

```
Round(2.3) = 2 , Round(2.8) = 3
Round(-2.3) = -2 , Round(-2.8) = -3
```

• مثال

```
Dim S s Single,Z As Single
 For S = -10 To 10 Step 0.5
    Print S ; Spc(3) ; Round ( s )
 Next
End
```

۴-۵ تبدیل کدها و متغییرها به یکدیگر

دستور ASC

Var = ASC(string)

این دستور اولین کاراکتر یک متغییر از نوع داده STRING را به مقدار اسکی آن تبدیل می‌کند.

• مثال

```
Dim A As Byte , S As String * 10
S = "ABC"
A = Asc(s)
Print A          'will print 65
End
```

دستور HEX

این دستور یک داده از نوع BYTE , INTEGER , WORDیا LONG را بـه مقـدار هگزدسـیمال تبـدیل می‌کند.

var = Hex(x)

مقدار HEX متغییر یا ثابت X در متغییر var جای می‌گیرد.

• مثال

```
Dim A As Byte , S As String * 10
A = 123
S = Hex(a)
Print S                 '7B will print
Print Hex(a)            '7B will print too
End
```

دستور HEXVAL

دستور HEXVAL یک داده هگزدسیمال را به مقدار عددی (دسیمال) تبدیل می‌کند.

var = HEXVAL(x)

مقدار عددی داده هگزدسیمال x که می‌تواند BYTE , INTEGER , WORDیا LONG باشد در متغییر var جای می‌گیرد.

• مثال

```
Dim A As Integer , S As String * 15
S = "0A"
A = Hexval(s) : Print A    '10 will be print
```

دستور MAKEBCD

var1 = MAKEBCD(var2)

این دستور متغییر یا ثابت var2 را تبدیل به مقدار BCD اش می‌کند و در متغییر var1 جای می‌دهد.

• مثال

```
Dim A As Byte
A = 65
A = Makebcd( a )
Lcd A                '101 will show
End
```

دستور MAKEDEC

var1 = MAKEDEC(var2)

برای تبدیل یک داده BCD نوع BYTE ، WORD یا INTEGER به مقدار DECIMAL از این دستور استفاده می‌شود. مقدار دسیمال متغییر یا ثابت VAR2 در متغییر VAR1 قرار می‌گیرد.

• مثال

```
Dim A As Byte
A = 65
Lcd A
Lowerline
Lcd Bcd(a)
A = Makedec(a)          'A=101
Lcd "    " ; A
End
```

دستور MAKEINT

varn = MAKEINT(LSB , MSB)

این دستور دو بایت را به هم متصل می‌کند و یک داده نوع WORD یا INTEGER می‌سازد که LSB بایت کم‌ارزش و MSB بایت پرارزش متغییر دو بایتی varn را تشکیل می‌دهند.

varn = (256 * MSB) + LSB

• مثال

```
Dim A As Integer , I As Integer
a = 2
I = Makeint(a , 1)          'I = (1 * 256) + 2  = 258
End
```

دستور STR

با این دستور می‌توان یک متغیر عددی(X) را به رشته(var) تبدیل کرد.

```
var = Str( x )
```

• **مثال**

```
Dim A As Byte , S As String * 10
A = 123             'A is a num
S = Str(a)          'Now A is a string
Print S
End
```

دستور VAL

```
var = Val(s )
```

با این دستور می‌توان یک رشته (S) را به متغیر عددی (var) تبدیل کرد. این دستور دقیقاً عکس دستور STR عمل می‌کند.

• **مثال**

```
Dim A As Byte , S As String * 10
S = "123"                           'Now s is a string
A = Val(s)                          'convert string to num
Print A
A=A*2                               'Now you can use it as a num
Print A                             '246 Prints
End
```

دستور STRING

```
var =  STRING(m , n)
```

این دستور کد اسکی n را با تعداد تکرار m تبدیل به رشته کرده و در متغییر var قرار می‌دهد. در صورت قرار دادن m = 0 یک رشته به طول 255 کارکتر تولید می‌شود. همچنین قرار دادن n = 0 قابل قبول نیست.

• **مثال**

```
Dim S As String * 15
S = String(5 , 65)
Print S              'AAAAA
End
```

تابع BIN2GREY

```
var1 =  bin2grey ( var2 )
```

متغییر var2 که می‌تواند داده‌ای از نوع BYTE , INTEGER ,WORD یا LONG باشد به کُدگری تبدیل شده و در متغییر var1 قرار می‌گیرد.

• **مثال**

```
Dim B As Byte           'could be word ,integer or long too
For B = 0 To 15
   Print Bin2grey(B)  '0 1 3 2 6 7 5 4 12 13 15 14 10 11 9 8 prints
Next
END
```

تابع GRAY2BIN

var1 = grey2bin (var2)

کُدگری var2 ، به مقدار باینری تبدیل شده و در متغییر var1 که می‌تواند داده‌ای از نوع BYTE INTEGER , WORD یا LONG باشد قرار می‌گیرد.

• **مثال**

```
Dim B As Byte           'could be word,integer or long too
For B = 0 To 15
   Print grey2bin (B)  '0 1 3 2 7 6 4 5 15 14  12 13 8 9 11 10 prints
Next
END
```

۵-۵ رجیسترها و آدرس‌های حافظه

در این فصل قصد داریم به معرفی دستوراتی که با رجیسترهای میکرو وآدرس‌های حافظه کار می‌کند بپردازیم. تمام میکروهای AVR دارای 32 رجیستر 8 بیتی (R0-R31) همه‌منظوره در CPU خود هستند. رجیسترهای R31(MSB) با R30 (LSB) ، R29(MSB) با R28 (LSB) و R27(MSB) با R26 (LSB) تشکیل سه رجیستر 16 بیتی به ترتیب با نامهای Z ، Y و X را می‌دهند.

دستور SET

توسط این دستور می‌توان یک بیت را یک کرد.

```
set Bit/pin
set Var.x
```

Bit می‌تواند یک بیت و یا یک SFR مانند PORTB.1 باشد و Var متغییری از نوع داده BYTE ، WORD INTEGER یا LONG است. X برای BYTE می‌تواند 0 تا 7 ، 0 تا 15 برای WORD و برای LONG می‌تواند 0 تا 31 باشد.

• **مثال**

```
Dim B1 As Bit , B2 As Byte , C As Word , L As Long
set Portb.1           'set bit 1 of  port B
Set B1                'bit variable
Set B2.1              'set bit 1 of var b2
Set C.15              'set highest bit of Word
Set L.31              'set MS bit of LONG
```

دستور TOGGLE

این دستور مقدار منطقی یک پایه یا یک بیت را معکوس می‌کند.

```
TOGGLE  pin/Bit
```

pin می‌تواند یک SFR مانند PORTB.1 و یا یک بیت باشد.

• **مثال**

```
Dim Var As Byte
Config Pinb.0 = Output          'portB.0 is an output now
Toggle Portb.0                  'toggle state
Waitms 1000                     'wait for 1 sec
Toggle Portb.0                    'toggle state again
```

دستور RESET

توسط این دستور می‌توان یک بیت را صفر کرد.

```
Reset Bit/pin
Reset Var.x
```

Bit می‌تواند یک بیت و یا یک SFR مانند PORTB.1 باشد و Var متغییری از نوع داده BYTE، INTEGER، WORD یا LONG است و X برای BYTE می‌تواند 0 تا 7، 0 تا 15 برای WORD و برای LONG می‌تواند 0 تا 31 باشد.

• مثال

```
Dim B1 As Bit , B2 As Byte , I As Integer
Reset Portb.3                          'reset bit 3 of port B
Reset B1                               'bit variable
Reset B2.0                             'reset bit 0 of byte variable b2
Reset I.15                             'reset MS bit from I
```

دستور BITWAIT

```
BITWAIT X , SET/RESET
```

توسط این دستور اجرای برنامه تا زمانی که بیت X، SET (=1) یا RESET(=0) شود در خط جاری متوقف می‌ماند. در صورت TRUE شدن شرایط، اجرای برنامه از خط بعد ادامه می‌یابد. X می‌تواند یک بیت رجیستر داخلی مانند PORTB.Y باشد که Y می‌تواند بین اعداد 0 تا 7 تغییرکند.

• مثال

```
Dim a as bit
Bitwait A , Set                    'Wait Until Bit A Is Set
Bitwait Portb.7 , Reset            'Wait Until Bit 7 Of Port B Is 0
```

دستور CPEEK

```
var = CPEEK(address)
```

از این دستور برای برگرداندن بایتی که در آدرسی از حافظه کدی ذخیره شده است استفاده می‌کنیم. با این دستور می‌توانید به رجیسترهای داخلی نیز دسترسی پیدا کنید. لازم به تذکر است که با این دستور شما نمی‌توانید در حافظه داخلی بنویسید.

• مثال

```
Dim I As Integer , B1 As Byte
For I = 0 To 31                    'only 32 registers in AVR
    B1 = Peek(i)                   'get byte from internal memory (R0-R31)
    Print Hex(b1)
Next
```

دستور CPEEKH

با این دستور می‌توان بایت ذخیره شده در صفحه (PAGE) بالای حافظه کُدی (FLASH MEM) میکرو MEGA103 یا دیگر میکروها که دارای 128K حافظه است را خواند.

```
var = CPEEKH( address )
```

address آدرس حافظه و محتوای آدرس در متغییر یک بایت var قرار می‌گیرد. (0)Cpeekh محتوای

اولین بایت حافظه بالای 64K را بر می‌گرداند .

دستور LOADADR

LOADADR var , reg

با این دستور می‌توانید آدرس یک متغییر را در یک جفت رجیستر ذخیره کنید. VAR متغییری است که آدرس آن در متغییرهای دو بایتی X ، Y و Z ذخیره می‌شود و REG رجیسترهای X ، Y و Z هستند. این دستور جزء دستورات اسمبلی است و برای کمک به برنامه‌نویسان اضافه شده است.

• مثال

```
Dim S As String * 12
Dim A As Byte
$asm
       Loadadr S , X          'load address into R26 and R27
       ld _templ, X         'load value of location R26/R27 into R24 (_templ)
$end Asm
End
```

دستور OUT

توسط این دستور می‌توان یک بایت به یک پورت سخت‌افزاری یا آدرس حافظه داخلی/خارجی ارسال کرد.

OUT address , value

value به آدرس address که می‌تواند بین 0H تا FFFFH باشد ارسال می‌شود. دستور OUT می‌تواند در تمام مکانهای حافظه AVR بنویسد. توجه کنید که برای address یک WORD تعریف شود. در ضمن برای نوشتن در مکان حافظه خارجی (XRAM) باید در محیط BASCOM و در منوی (OPTION → COMPILER → CHIP)، گزینه (EXTERNAL ACCESS ENABLE) را فعال کنید.

• مثال

```
Dim A As Byte
Out &H8000 , 1          'send  1 to the databus(d0-d7) at hex address 8000
End
```

دستور INP

توسط این دستور می‌توان یک بایت از پورت سخت‌افزاری یا آدرس حافظه داخلی / خارجی خواند.

var = INP (address)

محتوای آدرس address که می‌تواند بین 0H تا FFFFH باشد خوانده شده و در متغییر var قرار می‌گیرد . دستور INP می‌تواند از تمام مکانهای حافظه AVR بخواند. درضمن برای خواندن از مکان حافظه خارجی (XRAM) باید در محیط BASCOM و در منوی (OPTION →COMPILER → CHIP) گزینه (EXTERNAL ACCESS ENABLE) را فعال کنید.

• مثال

```
Dim A As Byte
A =Inp(&H8000)
'read value that is placed on databus(d0-d7)at hex address 8000
Print  A : End
```

دستور PEEK

این دستور محتوای یک رجیستر را برمی‌گرداند.

```
var = PEEK( address )
```

address آدرس رجیسترهای R0 تا R31 است که بین 0 تا 31 می‌باشد. محتوای رجیستر در متغییر var جای می‌گیرد. دستور( )PEEK فقط می‌تواند محتوای رجیسترها را بخواند ولی ( )INP از تمام مکانهای حافظه توانایی خواندن را داراست.

- **مثال**

```
Dim A As Byte
A = Peek( 0 )                'return the first byte of the internal memory (r0)
End
```

دستور POKE

با این دستور می‌توانیم یک بایت داده را در یکی از رجیسترها بنویسیم.

```
POKE  address , value
```

مقدار متغییر یا ثابت یک بایتی value در آدرس address که بین 0 تا 31 برای رجیسترهای R0-R31 است نوشته می‌شود.

- **مثال**

```
Poke 1 , 5          'write 5 to R1
End
```

دستور VARPTR

این دستور آدرس یک متغییر را در مکان حافظه بر می‌گرداند.

```
var = VARPTR( var2 )
```

آدرس متغییر var2 در مکان حافظه بدست آمده و در متغییر var قرار می‌گیرد.

- **مثال**

```
Dim B As Xram Byte At &H300 , I As Integer , W As Word
W = Varptr(b)
Print Hex(w)          'Print &H0300
End
```

۶-۵ دستورالعملهای حلقه و پرش

دستور GOTO و JMP

```
GOTO  label
JMP   lable
```

با این دستورات می‌توان به برچسب label پرش کرد. برچسب label باید با علامت : (collon) پایان یابد و می‌تواند تا 32 کاراکتر طول داشته باشد. به خاطر داشته باشید زمانی که از دو label هم نام

استفاده شود کامپایلر به شما هشدار(Warning) می‌دهد. دستور RETURN بـرای برگـشت از برچـسب وجود ندارد.

• مثال

```
Start :                 'A Label Must End With A Colon
A = A + 1               'Increment A
If A < 10 Then          'Is It Less Than 10
Goto Start              'or    jmp start
End If                  'Close If
End
```

دستورالعمل DO – LOOP

فرم کلی دستور DO...LOOP به صورت زیر می‌باشد.

```
DO
   statements
LOOP [ UNTIL  expression ]
```

دستورالعمل Statements تا زمانی کـه expression دارای ارزش TRUE یـا غیـر صـفر اسـت تکـرار خواهد شد بنابراین این نوع حلقه، حداقل یکبار تکـرار مـی شـود. DO – LOOP بتنهـایی یـک حلقـه بینهایت است که با EXIT DO می‌توان از درون حلقه خارج شد و اجرای برنامه در خط بعـد از حلقـه ادامه یابد.

• مثال

```
Dim A As Byte
Do                              'Start The Loop
A = A + 1                       'Increment A
Print A                         'Print It
Loop Until A = 10               'Repeat Loop Until A = 10
Print A
```

دستورالعمل FOR – NEXT

فرم کلی دستور FOR...NEXT به صورت زیر می‌باشد.

```
FOR  var = start  TO  end  [ STEP  value ]
        statements
Next var
```

VAR بعنوان یک کانتر عمل می‌کند که START مقدار اولیه و END مقـدار پایـانی اسـت و هـر دو می‌توانند یک ثابت عددی یـا متغییـر عـددی باشـند. VALUE مقدارعـددی STEP (قـدمها) را نـشان می‌دهد که می‌تواند مثبت یا منفی باشد. در صـورت حـذف کـردن STEP VALUE کامپـایلر بـصورت پیش‌فرض مقدار یک را در نظر می‌گیرد.

نکته نوشتن نام متغییر var بعد از NEXT الزامی نیست .

• مثال

```
Dim A As Byte , B1 As Byte , C As Integer
For A = 1 To 10 Step 2
      Print "This is A " ; A
Next A
```

```
For C = 10  To  -5  Step  -1              'Use A Negative Stepsize
        Print "This is C " ; C
Next

For B1 =  1  To  10
        Print  "This is B1 " ; B1
Next            'Note That You Do Not Have To Specify The Parameter
End
```

دستورالعمل WHILE-WEND

دستورالعمل WHILE-WEND تشکیل یک حلقه تکرار می‌دهد که تکرار این حلقه زمانی ادامه می‌یابد که عبارت بکار برده شده نتیجه FALSE را حاصل کند و یا دارای مقدار صفرشود. دستورالعمل WHILE به صورت ورود به حلقه با شرط می‌باشد، یعنی قبل از ورود به حلقه ، شرط حلقه تست می‌شود و در صورت TRUE بودن کنترل اجرای برنامه به حلقه وارد می‌گردد. بنابراین حلقه WHILE ممکن است در حالت‌هایی اصلاً اجرا نشود حتی یعنی یک بار هم مراحل حلقه طی نشود.

```
While Condition
    statements
Wend
```

بخش statements تا وقتی که حاصل Condition صفر یا FALSE نشده است تکرار خواهد شد.

• مثال

```
Dim a as byte
A = 1
While A <= 10              'if a is smaller or equal to 10
   PRINT a                'print variable a
   INCR a
Wend
```

دستورالعمل IF

در کلیه حالتهای زیر عبارت statement می‌تواند یک دستورالعمل‌ساده یا چند دستورالعمل‌مرکب باشد .

حالت 0:

```
If  Expression Then statement
```

دستورالعمل statement زمانی اجرا می‌شود که عبارت expression دارای ارزش TRUE باشد.

حالت 1 :

```
If  Expression  Then
        Statement1
 Else
        Statement2
End If
```

در صورتی که عبارت expression دارای ارزش TRUE باشد دستورالعمل statement1 اجـرا خواهـد شد ، در غیر این صورت دستورالعمل statement2 اجرا می‌شود .

حالت 2 :

```
If Expression1 Then
    Statement1
Elseif  [Expression2 Then]
    Statement2
Else
    Statement3
End If
```

در صورتی که عبارت expression1 دارای ارزش TRUE باشد دستورالعمل statement1 اجرا خواهد شد. در صورتی که عبارت expression1 دارای ارزش FALSE ولی عبارت اختیاری expression2 دارای ارزش TRUE باشد، دستورالعمل statement2 اجرا خواهد شد و در غیر این‌صورت یعنی در حالتی که هر دو عبارت expression2 و expression1 دارای ارزش FALSE باشند دستورالعمل statement3 اجرا خواهد شد.

همچنین با دستور IF می‌توان یک یا صفر بودن یک بیت از یک متغییر را امتحان کرد.

IF bit = 1 THEN or IF bit = 0 THEN

یا

```
Dim Var As Byte , Idx As Byte
Idx = 1
If Var.idx = 1 Then                    'if bit 1 of var is 1 then
Set portb.0
Else
```

• مثال

```
Dim A As Integer
A = 10
If A = 10 Then                                  'Test Expression
      Print "This part is executed."             'This Will Be Printed
Else
Print "This will never  be executed."            'This Not
End If
If A = 10 Then Print "New in BASCOM"
If A = 10 Then Goto Label1 Else Print "A<>10"

Label1:
IF A.15 = 1 THEN                                'test for bit
      Print "BIT 15 IS SET"
End If
' the following example shows the 1 line use of IF THEN [ELSE]
If A.15 = 0 Then Print "BIT 15 is cleared" Else Print "BIT 15 is set"
```

دستورالعمل CASE

کنترل اجرای دستورات یک برنامه دارای ترتیب بالا به پایین است ولی در صورت نیاز می توان توسط دستورالعمل‌های انشعاب یا پرش جهت کنترل اجرای دستورات یک برنامه را تغییر داد. یکی از این دستورات SELECT – CASE است که می توان یکی از چندین دستور را با توجه به مقدار ورودی اجرا کرد.

```
Select Case Var
    CASE test1 : statement1
  [ CASE test2 : statement2 ]
    CASE ELSE : statement3
End Select
```

اگرمتغییرVar با مقدار test1 برابر باشد statement1 اجرا می‌شود و سپس اجرا برنامه بعد از End Select ادامه می‌یابد در غیر اینصورت اگرمتغییرVar با مقدار test1 برابر نباشد ولی با مقدار test2 برابر باشد statement2 اجرا می‌شود و سپس اجرا برنامه بعد از End Select ادامه می‌یابد و نهایتاً

اگرمتغییرVar با هیچکدام از مقادیر test1 وtest2 برابر نباشد، statement3 اجـرا می‌شود و سپس اجـرای برنامه بعد از End select ادامه می‌یابد.

شما می‌توانید به صورت‌های زیر نیز متغییر را امتحان کنید :

CASE IS > 2 : اگر متغییر مورد نظر بزرگتر از دو باشد.

و یا می‌توان محدوده‌ای را برای امتحان کردن در نظر گرفت :

CASE 2 TO 5 : اگر متغییر مورد نظر بین 2 تا 5 باشد.

• مثال

```
Dim X As Byte
Do
  Input "X ? " , X
  Select Case  X                        'test X
     'if  1<X>3 then "1 , 2 or 3 will be ok" will print
     Case 1 To 3   :Print "1 , 2 or 3 will be ok"
     Case 4 :Print    "4"               'if X=4 then  "4" will print
     Case Is >10   :Print    ">10" 'if X>10 then  ">10" will print
     Case Else :Print "no"          'else  "no" will print
   End Select

Loop
End
```

دستور EXIT

با این دستور می‌توانید فقط از یک ساختار یا حلقه خارج شوید و ادامـه برنامـه را بعـد از سـاختار یـا حلقه ادامه دهید.

```
EXIT A FOR..NEXT, DO..LOOP, WHILE..WEND, SUB..END SUB or FUNCTION..END FUNCTION.
EXIT  FOR
EXIT  DO
EXIT  WHILE
EXIT  SUB
EXIT FUNCTION
```

• مثال

```
Do                                      'begin a DO..LOOP
        A = A +  1                      'incr a
        If A = 100 Then                 'test for a = 100
              Exit Do                   'exit the DO..LOOP
        End  If                         'end the IF..THEN
Loop                                    'end the DO
End
```

دستورالعمل ON VALUE

با این دستور با توجه به مقدار متغییر می‌توان به توابع یا برچسب‌های مختلفی پرش کرد.

```
ON  var  [ GOTO ] [ GOSUB ]  label1  [ , label2  ]
```

var متغییر مورد نظر برای امتحان شدن که می تواند یک SFR مانند PORTD باشد و LABEL1 ، LABEL2 ،.. برچسب‌هایی هستند که با توجه به مقدار var به آنها پرش می‌شود. توجه داشته باشید زمانی که var = 0 باشد به اولین برچسب پرش می‌شود.

نکته زمانی که GOSUB استفاده می‌نمایید باید با دستور RETURN از برنامه برگردید.

• **مثال**

```
Dim X As Byte
X = 1
On X Gosub Lbl2 , Lbl3                  'jump to sub lbl3
X = 0
On X Goto Lbl1 , Lbl4                   'jump to label lbl1
Lbl4:
Print X
Lbl1:
Incr X
Print X                                 'PRINTS 1
End

Lbl2:
Print " lbl3 "
Return

Lbl3:
Decr X
Print X                                 'PRINTS 0
Return
```

۵-۷ ایجاد تأخیر در برنامه

دستور DELAY

این دستور برای مدت کوتاهی به مقدار 1000 میکروثانیه در اجرای برنامه تأخیر ایجاد می‌کند.

• **مثال**

```
DELAY               'wait for hardware to be ready
```

دستور WAITuS

برای ایجاد تأخیر در برنامه از این دستور استفاده می‌شود.

WAITus microseconds

اجرای برنامه به مدت microseconds میکروثانیه متوقف می‌شود ، پس از سپری شدن زمان مشخص شده اجرای برنامه از خط بعد ادامه می‌یابد. microsecond می‌تواند اعداد بین (255 – 1) باشد.

نکته دستورات تأخیری زمان دقیق را به شما نمی‌دهد. برای بدست آوردن زمان دقیق از تایمرها استفاده نمایید.

• **مثال**

```
Waitus 10                       'wait for 10 uS
Print "bascom"
End
```

دستور WAITms

WAITms milliseconds

اجرای برنامه به مدت milliseconds میلی ثانیه متوقف می‌شود ، پس از سپری شدن زمان مشخص شده اجرای برنامه از خط بعد ادامه می‌یابد. milliseconds می‌تواند اعداد بین (65535 - 1) باشد.

• مثال

```
Waitms 10                    'wait for 10 mS
Print "bascom"
End
```

دستور WAIT

WAIT seconds

اجرای برنامه به مدت seconds ثانیه متوقف می‌شود ، پس از سپری شدن زمان مشخص شده اجرای برنامه از خط بعد ادامه می‌یابد.

• مثال

```
Wait 3                       'wait for three seconds
Print "bascom"
```

۸-۵ زیربرنامه و تابع

زیربرنامه‌ها یا توابع ابتدا بایستی معرفی شوند سپس توسط دستور CALL فراخوانی شوند و در انتها بعد از دستور END ،برنامه مربوط به هر کدام نوشته شوند.

معرفی تابع (DECLARE FUNCTION)

از این دستور برای معرفی تابع در ابتدای برنامه استفاده می‌شود. زمانی که بخواهیم تابعی را فراخوانی کنیم بایستی تابع معرفی شده باشد. در صورت استفاده از تابع می‌بایستی یک داده برگردانده شود.

DECLARE FUNCTION TEST[([BYREF/BYVAL] var as type1)] As type2

TEST نام تابع مورد نظر است. انتقال داده بصورت BYVAL باعث می‌شود که یک کپی از متغییر به تابع فرستاده شود و در محتوای آن هیچ تغییری ایجاد نشود ولی در حالت BYREF آدرس متغییر ارسال و تغییرات در آن تاثیر می‌دارد و داده برگشتی در صورت انجام عملیات برروی آن با مقدار اولیه خود برابر نخواهد بود. در صورت عدم استفاده از گزینه [BYREF/BYVAL] بصورت پیش‌فرض، داده بصورت BYREF فرستاده می‌شود. type1 نوع داده ارسال شده و type2 نوع داده برگشتی است که هر دو می‌توانند داده نوع BYTE ، INTEGER ، WORD ، LONG یا STRING باشند.

• مثال

در مثال زیر I بصورت BYVAL فرستاده شده است بنابراین یک کپی از مقدار I به زیر تابع فرستاده می‌شود و هیچ تغییری در محتوای آن ایجاد نمی‌شود. S بصورت BYREF فرستاده می‌شود و تغییر در آن صورت می‌گیرد. فراخوانی تابع MYFUNCTION با K و Z از نوع داده INTEGER و STRING است و مقدار برگشتی از نوع INTEGER است که در متغییر T قرار می‌گیرد. شما می‌توانید در محدوده تابع یک متغییر محلی معرفی نمائید.

```
'A user function must be declare before it can be used.
'A function must return a type
Declare Function Myfunction(byval I As Integer , S As String) As Integer
'The byval paramter will pass the parameter by value so the original value
'will not be changed by the function

Dim K As Integer
Dim Z As String * 10
Dim T As Integer
'assign the values
K = 5
Z = "123"

T = Myfunction(k , Z)
Print T
End


Function Myfunction(byval I As Integer , S As String) As Integer
  'you can use local variables in subs and functions
  Local P As Integer

  P = I

  'because I is passed by value, altering will not change the original
  'variable named k
  I = 10

  P = Val(s) + I

  'finally assign result
  'Note that the same data type must be used !
  'So when declared as an Integer function, the result can only be
  'assigned with an Integer in this case.
  Myfunction = P
End Function
```

معرفی زیربرنامه (DECLARE SUB)

از این دستور برای معرفی زیربرنامه استفاده می‌کنیم. زیربرنامه‌ای که قصد فراخوانی آن را داریم بایستی در ابتدای برنامه و یا حداقل قبل از فراخوانی آن معرفی شده باشد.

DECLARE SUB TEST[([BYREF/BYVAL] var as type)]

زیربرنامه بر خلاف تابع ، مقداری برنمی‌گرداند. در زمان ارسال داده به صورت BYREF (پیش‌فرض کامپایلر) آدرس داده به زیربرنامه فرستاده شده و در محتوای آن تغییر ایجاد می‌شود ولی در حالت BYVAL یک کپی از داده فرستاده می‌شود و به هیچ وجه در محتوای آن، تغییری ایجاد نمی‌شود. TEST نام زیربرنامه و VAR نام متغییر ارسالی به زیربرنامه و TYPE نوع آن است که می‌تواند

داده نوع BYTE ، INTEGER ، WORD یا STRING باشد.

برای نوشتن زیربرنامه ابتدا نام آن را توسط دستور زیر تعریف کرده و سپس شروع به نوشتن زیربرنامه می‌کنیم.

SUB Name[(var1)]

NAME نام زیربرنامه که باید توسط دستور Declare معرفی شده باشد و با دستور End Sub پایان یابد.

• مثال

```
Dim a As Byte, b1 As Byte, c As Byte
Declare Sub Test (a As Byte)
a = 1 : b1 = 2: c = 3                  'assign values in  a line by sign :
Print a ; b1 ; c                       '123 will print
Call Test (b1)
Print a ; b1 ; c                       '223 will print
End                                    'end program

Sub Test(a As Byte)
          Print a ; b1 ; c             '123 will print
End Sub
```

• مثال

```
Dim A As Byte
Declare Sub Test
A = 1
Print A          ' 1
Call Test
Print A          ' 2
End

Sub Test
      A = A + 1
End Sub
```

• مثال

```
Dim A As Byte , B1 As Byte , C As Byte
Declare Sub Test(a As Byte)
A = 1 : B1 = 2 : C = 3
Print A ; B1 ; C              '123 prints
Call Test(b1)                 ' = Test(b1)
Print A ; B1 ; C              '124  prints
End
Sub Test(a As Byte)
Print A ; B1 ; C              '223 prints
C= C + 1
End Sub
```

فراخوانی (CALL)

توسط این دستور زیربرنامه یا تابعی را فراخوانی(CALL) می‌کنیم.

CALL TEST(VAR1 , VAR2 , ...)

VAR1 و VAR2 متغییرهایی که به زیربرنامه انتقال می‌یابند ، هستند. می‌توان زیربرنامه را بصورت زیر نیز فراخوانی کرد.

TEST VAR1 , VAR2 , ...

لازم به تذکر است که نام زیربرنامه قبل از فراخوانی آن ، باید توسط دستور DECLARE فراخوانی شود. اگر بخواهیم عدد ثابت را به زیربرنامه انتقال دهیم بایستی حتماً با آرگومان BYVAL آن را انتقال دهیم.

• مثال

```
Dim A As Byte , B As Byte      'dimension variables A and B
Declare Sub Test(b1 As Byte, BYVAL b2 As Byte) 'declare the SUB Test program
a=65                          'assign a value to variable A
Call test(a,5)        'call test with parameter A and constant
Test a,5               'alternative call
Print A              'now print the new value (A is changed A=10)
End
'end program

Sub Test(b1 As Byte , Byval B2 As Byte)
'use the same variable names as b1 and b2
Lcd B1                                      'put it on the LCD
Lowerline
LCD BCD(b2)
B1 = 10
'reassign the variable
B2 = 15
'reassign the variable
End Sub
```

• مثال

```
Dim A As Byte
Declare Sub Test
A = 1
Print A          ' 1
Test             ' =  CALL Test
Print A          ' 2
End

Sub Test
     A = A + 1
End Sub
```

بکارگیری متغییر محلی یا LOCAL

از این دستور برای تعریف متغییر محلی در زیر تابع یا زیربرنامه استفاده می‌نماییم.

LOCAL var As Type

VAR نام متغییر و TYPE نوع داده است که می‌توانند BYTE , INTEGER , WORD , STRING یا SINGLE یا LONG باشد. نوع داده‌های ERAM ، SRAM ، XRAM و آرایه‌ها نمی‌توانند محلی تعریف شوند. یک متغییر محلی یک متغییر موقت است که فقط در هنگام فراخوانی زیربرنامه مربوطه برای آن فضا در نظر گرفته می‌شود و با برگشت از زیربرنامه عمر متغییر (LIFE TIME) به اتمام می‌رسد.

نکته

متغییرهای بیتی نمی‌توانند بصورت محلی تعریف شوند.

• مثال

```
'First the SUB programs must be declared
'Try a SUB without parameters
Declare Sub Test2
Do
Call Test2                'call sub
'Test2                   'or You can use without CALL
Loop
End                      'end program

Sub Test2                'sub without parameters
       Local A As Byte
       Incr A            'A=A+1
       Lcd A
End Sub
```

پرش به زیربرنامه توسط دستور GOSUB

توسط این دستور می توان به زیربرنامه پرش کرد و اجرای برنامه را از آدرس برچسب ادامه داد.

GOSUB label

LABEL نام برچسب زیربرنامه است که به آن پرش می‌شود. توسط دستور RETURN می‌توان اززیربرنامه برگشت کرد و اجرای برنامه بعد از دستور GO SUB ادامه یابد.

• مثال

```
Dim X as byte
Gosub Routine                   'jump to Routine
Print "Hello"                   'After come back from Routin print "Hello"
End                             'Terminate Program

Routine:                        'This Is A Subroutine
X = X + 2                       'Perform Some Math
Print X                         ' Print Result  2
Return                          'Return
```

۶

پیکره‌بندی و کار با امکانات AVR در BASCOM

امکانات تمام میکروهای AVR قبل از استفاده در محیط Bascom بایستی ابتدا پیکره‌بندی (CONFIG) شوند. در این فصل به پیکره‌بندی یا همان راه‌اندازی امکانات AVR پرداخته‌ایم. در ابتدا هر بخش رجیسترهای وسیله شرح داده شده‌اند وسپس نحوه پیکره‌بندی و دستورات مربوط به وسیله کاملاً تشریح شده و از آنجا که ارائه مثال در یادگیری بسیار مفید است در انتها نیز مثالهایی برای درک بیشتر با کاربرد و نحوه کار با وسیله آمده است. در این بخش سعی شده دستورات مربوط به امکانات AVR در کنار پیکره‌بندی قرار گیرد تا کاربر بتواند با مطالعه بخش مربوط به وسیله دلخواه بتواند به راحتی وسیله را راه‌اندازی و با آن کار کند.

اهداف

۱. آشنایی با تمام امکانات و رجیسترهای مربوطه میکروهای AVR
۲. یادگیری کامل تمام پیکره‌بندی‌های امکانات AVR و دستورات مربوطه
۳. راه‌اندازی وسیله دلخواه و تحلیل نرم‌افزاری توسط تحلیل‌گر داخلی BASCOM
٤. راه‌اندازی وسیله دلخواه و تحلیل نرم‌افزاری توسط تحلیل‌گر PROTEUS

۶-۱ پیکره‌بندی پورت‌ها

برای تعیین جهت پایه پورت‌ها از این پیکره‌بندی استفاده می‌نماییم. جهت یک پایه می‌تواند ورودی یا خروجی باشد.

```
Config Portx = State
Config Pinx.y = State
```

Y و X بسته به میکرو می‌توانند به ترتیب پایه‌های 0 تا 7 پورت‌های A,B,C,D,E,F,G برای میکروهای مختلف باشند. State نیز می‌تواند یکی از گزینه‌های زیر باشد :

INPUT یا 0 : در این حالت رجیستر جهت داده پایه یا پورت انتخاب شده صفر (0) می‌شود و پایه یا پورت به عنوان ورودی استفاده می‌شود.

OUTPUT یا 1 : در این حالت رجیستر جهت داده پایه یا پورت انتخاب شده یک (1) می‌شود و پایه یا پورت به عنوان خروجی استفاده می‌شود.

زمانی که بخواهید از پورتی بخوانید بایستی از رجیستر PIN پورت مربوطه استفاده کنید و در هنگام نوشتن در پورت بایستی در رجیستر PORT بنویسید.

نکته

در صورتی که بخواهید یک پایه پورت را PULL-UP و به عنوان ورودی استفاده کنید بایستی DDRX.Y=0 و PORTX.Y=1 قرار داده شود که X پورت مربوطه(A,B,...) و y پایه دلخواه(0,1,...7) از پورت است.برای PULL-UP کردن تمام پورت نیز بایستی DDRX=0 و PORTX=255 قرار گیرد.

• مثال

```
DIM A AS BYTE
DDRC.0=0 : PORTC.0=1                    'SET PINC.0 INPUT AND PULL UP RESISTOR ON

IF PINC.0=0 THEN INCR A                 'IF KEY PRESSED THEN INCR A
'NORMALY PINC.0=1 BECAUSE OF PULL-UP RESISTOR (NO EXTERNAL RESISTOR NEEDED)

DDRD=0:PORTD=255                        'SET ALL PINS OF PORTD INPUT AND ACTIVE
'PULL-UP RESISTOR(NO EXTERNAL RESISTOR NEEDED)

IF PIND.0=0 THEN DECR A
```

• مثال

```
Dim A As Byte , Count As Byte
'configure PORT D for input mode
Config Portd = Input
A = Pind                                'Read Data On Portd
A = A And Portd                         'A=A & PORTD
Print A                                   'print it
Bitwait Pind. 7 , Reset                   'wait until bit is low
'We will use port B for output
Config Portb = Output
'assign value
Portb = 10                                'set port B to 10
Portb = Portb And 2                       'set bit 0 of port B to 1
Set Portb.0
Incr Portb
Count = 0
Do
   Incr Count
```

```
   Portb = 1
   For A = 1 To 8
      Rotate Portb , Left              'rotate bits left
      Wait 1
   Next
Loop Until Count = 10
End
```

بررسی پورت‌های میکرو ATMEGA32

در این بخش قصد داریم برای آشنایی بیشتر با عملکرد پورت‌ها و رجیسترهای مربوط به طور نمونه به بررسی پورت‌های میکرو ATMEGA32 بپردازیم.

پورت A

پورت A یک I/O دو طرفه 8 بیتی است. سه آدرس از مکان حافظه I/O اختصاص به PORTA دارد. یک آدرس برای رجیستر داده PORTA ، دومی رجیستر جهت داده DDRA و سومی پایه ورودی پورت A , PINA است. آدرس پایه‌های ورودی پورت A فقط قابل خواندن است درصورتیکه رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند. تمام پایه‌های پورت دارای مقاومت Pull-Up مجزا هستند. بافر خروجی پورت A جریانی تا 20mA را Sink می‌کند و در نتیجه می‌تواند LED را مستقیماً راه‌اندازی کند. هنگامی که که پایه‌هایPA0-PA7 با مقاومت‌های Pull-Down خارجی،به عنوان خروجی استفاده می‌شوند، آنها SOURCE جریان می‌شوند، زمانی که مقاومت‌های Pull-Up داخلی فعال باشند.

رجیسترهای پورت A

رجیسترهای پورت A عبارتند از:

رجیستر داده پورت A– PORTA [PORT A DATA REGISTER]

Bit	7	6	5	4	3	2	1	0
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

رجیستر جهت داده پورت A– DDRA [PORT A DATA DIRECTION REGISTER]

Bit	7	6	5	4	3	2	1	0
	DDA7	DDA6	DDR5	DDA4	DDA3	DDA2	DDA1	DDA0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

بایت آدرس پایه‌های ورودی پورت PINA–A [PORT A INPUT PINS ADDRESS]

Bit	7	6	5	4	3	2	1	0
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

استفاده از پورت A بعنوان یک I/O عمومی دیجیتال

جدول ۶-۱ تاثیر تغییرات DDAn بر روی پایه‌های PORTA

DDAn	PORTAn	I/O	Pull-up	Comment
0	0	Input	No	Tri –State
0	1	Input	Yes	PAn Will source current if ext.pulled low
1	0	Output	No	Push-pull Zero output
1	1	Output	No	Push-pull One output

تمام 8 پایه موجود زمانیکه بعنوان پایه‌های I/O دیجیتال استفاده می‌شوند دارای عملکرد مساوی هستند. PAn ، پایه I/O عمومی : بیت DDAn در رجیستر DDRA مشخص کننده جهت پایه است.با توجه به جدول ۶-۱ اگر DDAn یک باشد ، PAn بعنوان یک پایه خروجی مورد استفاده قرار می‌گیرد و اگر DDAn صفر باشد، PAn بعنوان یک پایه ورودی در نظر گرفته می‌شود. اگرPort An یک باشد هنگامیکه پایه بعنوان ورودی تعریف شود ، مقاومت PULL-UP فعال می‌شود برای غیرفعال کردن مقاومت باید Port An صفر شود یا اینکه پایه بعنوان خروجی تعریف شود. پایه‌های پورت زمانیکه ری‌ست اتفاق می‌افتد به حالت Tri –state می‌روند.

دیگر کاربردهای پورت A

پورت A به عنوان ADC هم استفاده می‌شود. این نکته بسیار مهم است که اگر تعدادی از پایه‌های پورت A خروجی تعریف شوند، در زمان نمونه برداری از سیگنال آنالوگ توسط ADC ، سوئیچ نشوند. این کار ممکن است عملیات تبدیل ADC را نامعتبر کند.

پورت B

پورت B یک I/O دو طرفه 8 بیتی است. سه آدرس از مکان حافظه I/O اختصاص به PORTB دارد. یک آدرس برای رجیستر داده PORTB ،دومی رجیستر جهت داده DDRB و سومی پایه ورودی پورت B , PINB است. آدرس پایه‌های ورودی پورت B فقط قابل خواندن است درصورتیکه رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند. پایه‌های پورت دارای مقاومت PULL-UP مجزا هستند. بافر خروجی پورت B جریانی تا 20mA را Sink کند و در نتیجه می‌تواند LED را مستقیماً راه‌اندازی کند. هنگامی که که PB0-PB7 با مقاومت‌های Pull-Down ،به عنوان خروجی استفاده می‌شوند، آنها SOURCE جریان می‌شوند زمانی که مقاومت‌های Pull-Up داخلی فعال باشند.

رجیسترهای پورت B

رجیسترهای پورت B عبارتند از:

رجیستر داده پورت B- PORTB [PORT B DATA REGISTER]

Bit	7	6	5	4	3	2	1	0
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

رجیستر جهت داده پورت DDRB -B [PORT B DATA DIRECTION REGISTER]

Bit	7	6	5	4	3	2	1	0
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

بایت آدرس پایه‌های ورودی پورت PINB-B [PORT B INPUT PINS ADDRESS]

Bit	7	6	5	4	3	2	1	0
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

استفاده از پورت B بعنوان یک I/O عمومی دیجیتال

تمام 8 پایه موجود زمانی که بعنوان پایه‌های I/O دیجیتال استفاده می‌شوند دارای عملکرد مساوی هستند. PBn ، پایه I/O عمومی : بیت DDBn در رجیستر DDRB مشخص کننده جهت پایه است. اگر DDBn یک باشد ، PBn به عنوان یک پایه خروجی مورد استفاده قرار می‌گیرد و اگر DDBn صفر باشد ، PBn بعنوان یک پایه ورودی در نظر گرفته می‌شود(جدول ۶-۲). اگر PortBn یک باشد هنگامیکه پایه بعنوان ورودی تعریف شود ، مقاومت Pull- Up فعال می‌شود برای خاموش کردن مقاومت Pull-Up باید Port Bn صفر شود یا اینکه پایه به عنوان خروجی تعریف شود. پایه‌های پورت زمانی که ری‌ست اتفاق می‌افتد به حالت Tri –state می‌روند.

جدول ۶-۲ تاثیر تغییرات DDBn بر روی پایه‌های PORTB

DDBn	PORTBn	I/O	Pull-up	Comment
0	0	Input	No	Tri-State
0	1	Input	Yes	PBn Will source current if ext.pulled low
1	0	Output	No	Push-pull Zero output
1	1	Output	No	Push-pull One output

دیگر کاربردهای پورت B

- **PORTB.7–SCK**

SCK :کلاک خروجی Master و کلاک ورودی Slave برای ارتباط SPI است. زمانی که SPI بعنوان Slave شکل‌دهی می‌شود این پایه با توجه به تنظیم DDB7 ورودی ودر حالت Master خروجی تعریف می‌شود.

- **PORTB.6– MISO**

MISO : ورودی داده Master و خروجی داده Slave که برای ارتباط SPI استفاده می‌شود. زمانی که SPI بعنوان Master شکل‌دهی می‌شود این پایه با توجه به تنظیمات DDB6 ورودی و در حالت Slave بعنوان خروجی استفاده می‌شود.

- **PORTB.5– MOSI**

MOSI : ورودی داده Slave و خروجی داده Master که برای ارتباط SPI استفاده می‌شود. زمانی که

جدول ۶-۳ دیگر کاربردهای پورت B

Port B	Alternate Functions
PB0	T0 (Timer/Counter0 External Counter Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB2	AIN0 (Analog Comparator Positive Input)
PB3	AIN1 (Analog Comparator Negative Input)
PB4	SS (SPI Slave Select Input)
PB5	MOSI (SPI Bus MasterOutput/Slave Input)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB7	SCK (SPI Bus Serial Clock)

SPI بعنوان Master شکل‌دهی می‌شود این پایه با توجه به تنظیمات DDB5 خروجی و در حالت Slave بعنوان ورودی استفاده می‌شود.

- **PORTB.4 – SS**

SS : زمانی که SPI بعنوان Slave شکل‌دهی شود PB.4 با توجه به DDB4 ورودی تعریف می‌شود و در Slave با Low شدن این پایه SPI فعال می‌شود. این پایه در Master می‌تواند خروجی یا ورودی تعریف شود.

- **PORTB.3 – OC0 , AIN1**

AIN1 : ورودی منفی مقایسه‌کننده آنالوگ است.

OC0 : دیگر کاربرد این پایه بعنوان خروجی مُد مقایسه‌ای Timer/Counter0 است. پایه PB3 با یک کردن DDB7 می‌تواند برای خروجی مُد مقایسه‌ای Timer/Counter0 شکل‌دهی شود.

- **PORTB.2 – INT2 , AIN0**

AIN0 : ورودی مثبت مقایسه‌کننده آنالوگ است.

INT2 : دیگرکاربرد این پایه بعنوان منبع وقفه خارجی دو است. پایه PB2 می‌تواند بعنوان منبع وقفه خارجی برای میکرو استفاده شود.

- **PORTB.1 – T1**

T1 : ورودی کلاک برای Timer/Counter1 است.

- **PORTB.0 – XCK , T0**

T0 : ورودی کلاک برای Timer/Counter0 است.

XCK : این پایه نیز می‌تواند بعنوان کلاک خارجی USART استفاده شود. این پایه فقط زمانی که USART در مُد آسنکرون کار می‌کند فعال می‌شود.

پورت C

پورت C یک I/O دو طرفه 8 بیتی است. سه آدرس از مکان حافظه I/O اختصاص به PORTC دارد. یک

آدرس برای رجیستر داده PORTC ،دومی رجیستر جهت داده DDRC و سومی پایه ورودی پورت C, PINC است. آدرس پایه‌های ورودی پورت C فقط قابل خواندن است درصورتیکه رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند. تمام پایه‌های پورت دارای مقاومت (Pull-Up) مجزا هستند. بافر خروجی پورت C می‌تواند تا 20mA را Sink کند و در نتیجه LED را مستقیماً راه‌اندازی می‌کند. هنگامی که که PC0-PC7 با مقاومت‌های Pull-Down ،خروجی استفاده می‌شوند، آنها SOURCE جریان می‌شوند زمانی که مقاومت‌های Pull-Up داخلی فعال باشند.

رجیسترهای پورت C

رجیسترهای پورت C عبارتند از:

رجیستر داده پورت C- PORTC [PORT C DATA REGISTER]

Bit	7	6	5	4	3	2	1	0
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

رجیستر جهت داده پورت C- DDRC [PORT C DATA DIRECTION REGISTER]

Bit	7	6	5	4	3	2	1	0
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

بایت آدرس پایه‌های ورودی پورت C- PINC [PORT C INPUT PINS ADDRESS]

Bit	7	6	5	4	3	2	1	0
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

استفاده از پورت C بعنوان یک I/O عمومی دیجیتال

تمام 8 پایه موجود زمانی که به عنوان پایه‌های I/O دیجیتال استفاده می‌شوند دارای عملکرد مساوی هستند. PCn ، پایه I/O عمومی : بیت DDCn در رجیستر DDRC مشخص کننده جهت پایه است. طبق جدول ۶-۴ اگر DDCn یک باشد ، PCn بعنوان یک پایه خروجی مورد استفاده قرار می‌گیرد و اگر DDCn صفر باشد ، PCn بعنوان یک پایه ورودی در نظر گرفته می‌شود. اگر PORTCn یک باشد هنگامی که پایه به عنوان ورودی تعریف شود ، مقاومت Pull- Up فعال می‌شود برای خاموش کردن مقاومت Pull-Up باید PORTCn صفر شود یا اینکه پایه به عنوان خروجی تعریف شود. پایه‌های پورت در زمانی که ری‌ست اتفاق می‌افتد به حالت Tri - State می‌روند.

جدول ۶-۴ تاثیر تغییرات DDCn بر روی پایه‌های PORTC

DDCn	PORTCn	I/O	Pull-up	Comment
0	0	Input	No	Tri-State
0	1	Input	Yes	PCn Will source current if ext.pulled low
1	0	Output	No	Push-pull Zero output
1	1	Output	No	Push-pull One output

دیگر کاربردهای پورت C

جدول ۶-۵ دیگر کاربردهای پورت C

Port Pin	Alternate Functions
PC7	TOSC2 (Timer Oscillator Pin 2)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC5	TDI (JTAG Test Data In)
PC4	TDO (JTAG Test Data Out)
PC3	TMS (JTAG Test Mode Select)
PC2	TCK (JTAG Test Clock)
PC1	SDA (Tow-Wire Serial Bus Data Input/Output Line)
PC0	SCL (Tow-Wire Serial Bus Clock Line)

- **PORTC. 7 –TOSC2**

TOSC2 : زمانی که تایمر/کانتر2 در مُد آسنکرون کار می‌کند به این پایه و پایه TOSC1 کریستال ساعت متصل می‌شود. در این حالت دیگر نمی‌توان این پایه را به عنوان I/O استفاده نمود.

- **PORTC. 6 –TOSC1**

TOSC1 : زمانی که تایمر /کانتر 2 در مُد آسنکرون کار می‌کند به این پایه و پایه TOSC2 کریستال متصل می‌شود. در این حالت دیگر نمی‌توان این پایه را به عنوان I/O استفاده نمود.

- **PORTC. 5 – TDI**

TDI : در زمان ارتباط JTAG به عنوان ورودی داده سریال عمل می‌کند و دیگر نمی‌توان از این پایه بعنوان I /O استفاده نمود.

- **PORTC.4 – TDO**

TDO : در زمان ارتباط JTAG ، به عنوان خروجی داده سریال عمل می‌کند و دیگر نمی‌توان از این پایه به عنوان I /O استفاده نمود.

- **PORTC.3 – TMS**

TMS : در زمان ارتباط JTAG استفاده می‌شود و دیگر نمی‌توان از این پایه به عنوان I/O استفاده نمود.

- **PORTC.2 – TCK**

TCK : در زمان ارتباط JTAG استفاده می‌شود و دیگر نمی‌توان از این پایه به عنوان I/O استفاده نمود.

- **PORTC.1 – SDA**

SDA : در زمان ارتباط WIRE – 2 به عنوان خط داده استفاده می‌شود.

- **PORTC.0 – SCL**

SCL : در زمان ارتباط WIRE – 2 به عنوان خط کلاک استفاده می‌شود.

پورت D

پورت D یک I/O دو طرفه 8 بیتی است. سه آدرس از مکان حافظه I/O اختصاص به PORTD دارد. یک آدرس برای رجیستر داده PORTD ،دومی رجیستر جهت داده DDRD و سومی پایه ورودی پورت D ، PIND است. آدرس پایه‌های ورودی پورت D فقط قابل خواندن است درصورتی که رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند. تمام پایه‌های پورت دارای مقاومت Pull-Up مجزا هستند. بافر خروجی پورت D می‌تواند تا 20mA را Sink کند و در نتیجه LED را مستقیماً راه‌اندازی می‌کند. هنگامی که که PD0-PD7 با مقاومت‌های Pull-Down ،به عنوان خروجی استفاده می‌شوند، آنها SOURCE جریان می‌شوند زمانی که مقاومت‌های Pull-Up داخلی فعال باشند.

رجیسترهای پورت D

رجیسترهای پورت D عبارتند از:

رجیستر داده پورت D- [PORT D DATA REGISTER] PORTD

Bit	7	6	5	4	3	2	1	0
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

رجیستر جهت داده پورت D- [PORT D DATA DIRECTION REGISTER] DDRD

Bit	7	6	5	4	3	2	1	0
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

بایت آدرس پایه‌های ورودی پورت D- [PORT D INPUT PINS ADDRESS] PIND

Bit	7	6	5	4	3	2	1	0
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

استفاده از پورت D بعنوان یک I/O عمومی دیجیتال

تمام 8 پایه موجود زمانی که به عنوان پایه‌های I/O دیجیتال استفاده می‌شوند دارای عملکرد مساوی هستند. PDn ، پایه I/O عمومی : بیت DDDn در رجیستر DDRD مشخص کننده جهت پایه است. اگر DDDn یک باشد ، PDn بعنوان یک پایه خروجی مورد استفاده قرار می‌گیرد و اگر DDDn صفر باشد ، PDn بعنوان یک پایه ورودی در نظر گرفته می‌شود(جدول ۶-۶). اگر PortDn یک باشد هنگامیکه پایه بعنوان ورودی تعریف شود ، مقاومت Pull- Up فعال می‌شود برای خاموش کردن مقاومت Pull-Up باید Port Dn صفر شود یا اینکه پایه بعنوان خروجی تعریف شود. پایه‌های پورت در زمانیکه ری‌ست اتفاق می‌افتد به حالت Tri - State می‌روند.

دیگر کاربردهای پورت D

جدول ۶-۶ دیگر کاربردهای پورت D

PORT PIN	ALTERNATE FUNCTION
PD7	OC2 (T/C2 OUTPUT COMPARE MATCH OUTPUT)
PD6	ICP (T/C1 INPUT CAPTURE PIN)
PD5	OC1A (T/C 1 OUTPUT COMPAREA MATCH OUTPUT
PD4	OC1B (T/C 1 OUTPUT COMPAREB MATCH OUTPUT
PD3	INT1 (EXTERNAL INTERRUPT 1 INPUT)
PD2	INT0 (EXTERNAL INTERRUPT 0 INPUT)
PD1	TXD (UART OUTPUT LINE)
PD0	RXD (UART INPUT LINE)

- **PORTD.7–OC2**

OC2 : خروجی مُد مقایسه‌ای تایمر/کانتر2 . PD7 با یک شدن DDD7 می‌تواند بعنوان پایه خروجی مُد مقایسه‌ای Timer/Counter2 شکل‌دهی شود. این پایه همچنین برای خروجی PWM تایمر استفاده می‌شود.

- **PORTD.6 –ICP**

ICP : PD6 می‌تواند بعنوان پایه ورودی CAPTURE تایمر/کانتر1 عمل کند.

- **PORTD . 5 – OC1A**

OC1A : خروجی مُد مقایسه‌ای Timer/Counter1 . پایه PD5 با یک شدن DDD5 می‌تواند برای خروجی مُد مقایسه‌ای Timer/Counter1 شکل‌دهی شود. این پایه همچنین برای خروجی PWM تایمر1 استفاده می‌شود.

- **PORTD.4 – OC1B**

OC1B : خروجی مُد مقایسه‌ای Timer/Counter1

پایه PD4 با یک شدن DDD4 می‌تواند برای خروجی مُد مقایسه‌ای Timer/Counter1 شکل‌دهی شود. این پایه همچنین برای خروجی PWM تایمر استفاده می‌شود.

- **PORTD. 3– INT1**

INT1 : منبع وقفه خارجی یک

پایه PD3 می‌تواند بعنوان منبع وقفه خارجی برای میکرو استفاده شود.

- **PORTD. 2– INT0**

INT0 : منبع وقفه خارجی صفر

پایه PD2 می‌تواند بعنوان منبع وقفه خارجی برای میکرو استفاده شود.

- **PORTD.1– TXD**

TXD : ارسال داده (پایه خروجی داده برای USART)

زمانیکه ارسال USART فعال می‌شود پایه با توجه به DDD1 بعنوان خروجی شکل‌دهی می‌شود.

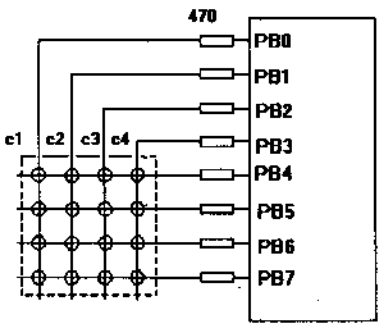
- **PORTD.0– RXD**

RXD : دریافت داده (پایه ورودی داده برای USART)

زمانی که دریافت USART فعال می‌شود پایه با توجه به DDD0 بعنوان ورودی شکل‌دهی می‌شود.

۶-۲ پیکره‌بندی صفحه‌کلید 4×4

اسکن صفحه‌کلید در BASCOM کار ساده است و تنها کافی است صفحه‌کلید خود را طبق شکل ۶-۱ به یکی از پورت‌های میکرو وصل نمایید و توسط دستور CONFIG KBD آن را پیکره‌بندی کنید.



شکل ۶-۱ اتصال صفحه کلید ۴×۴ به پورت میکرو

صفحه کلید توسط دستور زیر پیکره‌بندی می‌شود.

CONFIG KBD = PORTx , DEBOUNCE = value [, DELAY = value]

PORTx مشخص کننده پورتی است که صفحه کلید به آن متصل می‌شود. DEBOUNCE به صورت پیش فرض 20 است و می‌تواند ماکسیمم مقدار 255 را داشته باشد. DELAY نیز پارامتری اختیاری است و مشخص کننده تأخیری بر حسب میلی‌ثانیه است که در زمان خواندن کلید توسط دستور GETKBD() ایجاد می‌شود. این گزینه برای کاهش نویزهای محیط ایجاد شده است و مقدار DELAY=100 به طور مثال این مشکل را می‌تواند برطرف کند.

پیکره‌بندی فوق را می‌توان به صورت ساده شده زیر نیز نوشت ولی شما بایستی برای ایجاد تأخیر در زمان اسکن صفحه کلید از دستورات تأخیر استفاده نمایید.

CONFIG KBD = PORTx

زمانی که نیاز باشد صفحه کلیدی را با 6 سطر بخوانید می‌توانید از دستور زیر به جای دستور فوق استفاده نمایید.

CONFIG KBD = PORTX , DEBOUNCE = VALUE , ROWS=6, ROW5=PINX.Y, ROW6=PINA.B

در این حالت سطر پنجم به پایه PINX.Y و سطر ششم به پایه PINA.B اتصال می‌یابد.

• مثال

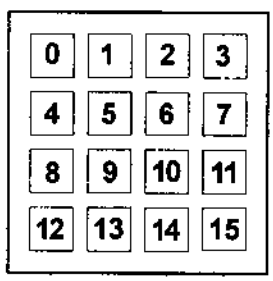
```
CONFIG KBD = PORTC , DEBOUNCE = 50 , rows=6, row5=pinD.6, row6=pind.7
```

با توجه به این پیکره‌بندی سطر پنجم و ششم به ترتیب به پایه‌های PIND.6 و PIND.7 متصل می‌شوند.

دستور GETKBD()

SOURCE=GETKBD()

توسط این دستور میکرو صفحه کلید را خوانده و عدد متناظر با کلید فشرده شده را در متغییر SOURCE قرار می‌دهد. این دستور زمانی که کلیدی فشرده نشده باشد عدد 16 بر می‌گرداند. شما از جدول LOOKUP() می‌توانید برای تبدیل عدد بدست آمده از صفحه کلید به مقدار دلخواه خود استفاده نمایید. در صورت اتصال صفحه کلید طبق شکل ۶-۱ عدد متناظر با کلید فشرده شده توسط دستور GETKBD() به صورت شکل ۶-۲ دریافت خواهد شد.



شکل ۶-۲ عدد متناظر با کلید فشرده شده توسط دستور GETKBD()

• مثال

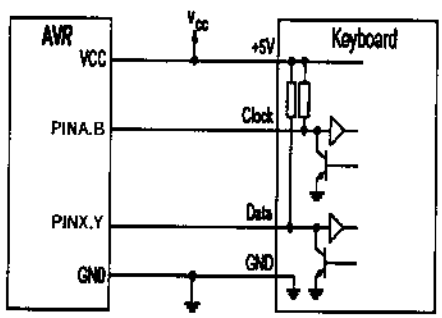
```
$regfile = "m32def.dat
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Pinb.4 , Db5 = Pinb.5 , Db6 = Pinb.6 , Db7_
= Pinb.7 , Rs = Pinb.2 ,E = Pinb.3
Config Lcd = 16 * 2
Config Kbd = Portc, Debounce = 50, Delay = 100
Dim A As Byte
Main:
   A = Getkbd()
   ' Wait 100ms By Compiler Because Of Delay=100
   If A > 15 Then Goto Main
   Home
   Lcd A
JMP Main
End
```

۶-۳ پیکره‌بندی صفحه‌کلید کامپیوتر

در زمان اتصال صفحه کلید کامپیوتر به پایه‌های میکرو از پیکره‌بندی زیر استفاده می‌نمائیم که PINA.B به پایه کلاک و PINX.Y به پایه DATA از صفحه کلید وصل شده و از جدول KEYDATA برای تبدیل کلید فشرده شده به کد اسکی استفاده می‌شود زیرا کدهای گرفته شده از صفحه کلید، اسکی نیستند.(کدهای گرفته شده نرم افزار Bascom در پیوست کتاب آمده است).

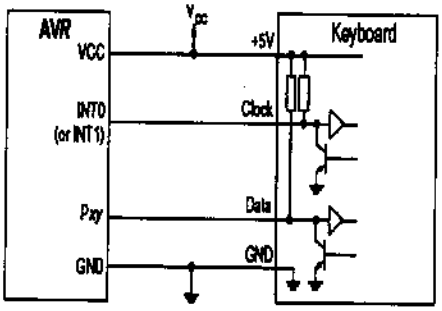
CONFIG KEYBOARD = PINA.B , DATA = PINX.y , KEYDATA = table

صفحه‌کلید کامپیوتر فقط به وسیله دو پایه DATA و CLOCK به میکرو وصل می‌شود. در صورت اتصال صفحه‌کلید طبق پیکره‌بندی بالا به دو پایه I/O صفحه‌کلید بایستی مطابق شکل ۶-۳ اتصال یابد. می‌توان صفحه‌کلید را طبق شکل ۶-۴ بصورت وقفه‌ای اتصال داد و پایه کلاک را به یکی از پایه‌های وقفه خارجی وصل کرد و زمانی که کلیدی فشرده شود وقفه خارجی فعال شده و نرم افزار در زیربرنامه وقفه ، DATA را خوانده و مقدار را می‌گیرد.



شکل ۶-۳

اتصال صفحه‌کلید به دو پایه I/O از میکرو

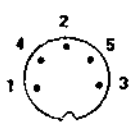


شکل ۶-۴

اتصال صفحه‌کلید بصورت وقفه‌ای

6-PIN MINI DIN PS2 STYLE AT COMPUTER

DIN41524 FEMALE AT COMPUTER ,5-PIN DIN



ترکیب پایه صفحه‌کلید در پشت کامپیوتر

جدول ۶-۷ ترکیب پایه‌های کانکتور صفحه‌کلید کامپیوتر

AT COMPUTER		
SIGNALS	DIN41524 , FEMALE AT COMPUTER , 5-PIN DIN 180⁰	6-PIN MINI DIN PS2 STYLE FEMALE AT COMPUTER
CLOCK	1	5
DATA	2	1
NC	3	2 , 6
GND	4	3
+5V	5	4
SHIELD	SHELL	SHELL

دستور GETATKBD

دستور زیرکلید فشرده شده از صفحه کلید کامپیوتر را می‌خواند .

VAR = GETATKBD( )

عدد گرفته شده از صفحه‌کلید در متغییر VAR قرار می‌گیرد. اگر کلید فشرده نشده باشد عدد صفر برگردانده می‌شود. VAR باید یک بایت یا یک متغییر رشته‌ای باشد. برای اتصال صفحه‌کلید به دو پایه ورودی و یک جدول تبدیل ، برای تبدیل کلید فشرده شده به مقدار اسکی نیاز است.شما می‌توانید اعداد صفر را به اعداد غیر تکراری جدول تغییر دهید وکلیدهای بیشتری را فعال کنید. لازم به یادآوری است که صفحه‌کلید قبل از استفاده از این دستور باید پیکره‌بندی شود. این دستور وقفه‌ای نیست و تا زمانی که کلید فشرده شود منتظر می‌ماند.

• مثال

```
'The Purpose Of This Sample Was How To Use A Pc At Keyboard
'Connect Pc At Keyboard Clock To Pind.2 On The MEGA32
'Connect Pc At Keyboard Data To Pind.4 On The MEGA32
'The Getatkbd() Function Does Not Use An Interrupt
'But It Waits Until A Key Was Pressed
'Configure The Pins To Use For The Clock And Data
'Can Be Any Pin That Can be An Input
'Keydata Is The Label Of The Key Translation Table
Config Keyboard = Pind.2 , Data = Pind.4 , Keydata = Keydata
'Dim Used Variable
Dim B As Byte

Do
        'The Following Code Show How To Use The Getatkbd() Function
        B = Getatkbd()   'Get A Byte And Store It Into Byte Variable
  'When No Real Key Is Pressed The Result Is 0 So Test If The Result Was > 0
        If B > 0 Then
        Print B ; Chr(b)
        End If
Loop

End

'This Is The Key Translation Table Calls when a key pressed
Keydata :
'Normal Keys Lower Case

Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , &H5E , 0
Data 0 , 0 , 0 , 0 , 0 , 113 , 49 , 0 , 0 , 0 , 122 , 115 , 97 , 119 , 50 , 0
Data 0 , 99 , 120 , 100 , 101 , 52 , 51 , 0 , 0 , 32 , 118 , 102 , 116 , 114 , 53 , 0
Data 0 , 110 , 98 , 104 , 103 , 121 , 54 , 7 , 8 , 44 , 109 , 106 , 117 , 55 , 56 , 0
Data 0 , 44 , 107 , 105 , 111 , 48 , 57 , 0 , 0 , 46 , 45 , 108 , 48 , 112 , 43 , 0
Data 0 , 0 , 0 , 0 , 0 , 92 , 0 , 0 , 0 , 0 , 13 , 0 , 0 , 92 , 0 , 0
Data 0 , 60 , 0 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0

'shifted keys UPPER case

Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
Data 0 , 0 , 0 , 0 , 0 , 81 , 33 , 0 , 0 , 0 , 90 , 83 , 65 , 87 , 34 , 0
Data 0 , 67 , 88 , 68 , 69 , 0 , 35 , 0 , 0 , 32 , 86 , 70 , 84 , 82 , 37 , 0
Data 0 , 78 , 66 , 72 , 71 , 89 , 38 , 0 , 0 , 76 , 77 , 74 , 85 , 47 , 40 , 0
Data 0 , 59 , 75 , 73 , 79 , 61 , 41 , 0 , 0 , 58 , 95 , 76 , 48 , 80 , 63 , 0
Data 0 , 0 , 0 , 0 , 0 , 96 , 0 , 0 , 0 , 0 , 13 , 94 , 0 , 42 , 0 , 0
Data 0 , 62 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0 , 0
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0
```

همانطور که گفته شده دستور GETATKBD وقفه‌ای نیست ولی گاهی برنامه بایستی طوری نوشته شود که در خط این دستور متوقف نشود. با یک کردن بیتی با نام ERR که از متغییرهای BASCOM است، می‌توان از این دستور به خط بعد پرش کرد . بنابراین به مثال زیر توجه کنید.

• مثال

```
$regfile = "M64DEF.DAT"
$crystal = 8000000
$BAUD=9600
Enable Interrupts
Enable Timer0
Stop Timer0
Timer0 = 0
On Timer0 Timer0_int
Config Pinb.6 = Input
Config Pinb.7 = Input
Config Keyboard = Pinb.6 , Data = Pinb.7 , Keydata = Keydata
Declare Sub Keybord
Dim Key As Byte

Do
Call Keybord
Waitms 500
Loop

End

Timer0_int:
   Set Err
   Timer0 = 0
Return

Sub Keybord
   Start Timer0
   Timer0 = 150
   Key = Getatkbd()
   Stop Timer0
      If Key > 0 Then
      IF A KEY PRESSED
      Print Key
      End If
End Sub

Keydata:
'normal keys lower case
Data 1 ,2 ,3 ,4 ,5 , 6 , 31 , 32 , 9 , 10 , 11 , 12 , 14 , 15 , &H5E , 16
Data 17 ,18 ,19 ,20 ,21 ,113 ,49 ,22 ,23 ,24 ,122 ,115 ,97 ,119 , 50 , 25
Data 26 ,99,120 ,100 ,101 ,52 ,51 ,27 ,28 ,32 ,118 ,102 ,116 ,114 ,53 ,29
Data 30,110 ,98 ,104 ,103 ,121 ,54 ,7 ,8 ,44 ,109 ,106 ,117 ,55 , 56 , 33
Data 34 ,44 ,107 ,105 ,111 ,48 ,57 ,35 ,36 ,46 ,45 ,108 ,48 ,112 ,43 , 37
Data 38 ,39 ,130 ,131 ,132 ,133,134 ,135,136,137,13 ,138,139,92,140 , 141
Data 0,60,143,144,145,146,8,147,148,49 , 149 , 52 , 55 , 150 , 151 , 152
Data 48,44,50,53,54,56,153 , 0 , 155 , 43 , 51 , 45 , 42 , 57 , 156 , 157

'shifted keys UPPER case
Data 158,159,160,161,0,163,164,165,166,167,168,169, 170 , 171 , 172 , 173
Data 174,175,176,177,178,81,33,179,180,181 , 90 , 83 , 65 , 87 , 34 , 182
Data 183,67,88,68,69,184,35,185 , 186 , 32 , 86 , 70 , 84 , 82 , 37 , 187
Data 188,78,66,72,71,89,38,189 , 190 , 76 , 77 , 74 , 85 , 47 , 40 , 191
Data 192,59,75,73,79,61,41, 193 , 194 , 58 , 95 , 76 , 48 , 80 , 63 , 195
Data 196,197,198,199,200,96,201,202,203,204,13, 94 , 205 , 42 , 206 , 207
Data 0,62,209,210,211,8,212,213,49,214 , 52 , 55 , 215 , 216 , 217 , 218
Data 48,44,50,53,54,56,219,220,221 , 43 , 51 , 45 , 42 , 57 , 222 , 223
```

۶-۴ پیکره‌بندی LCD

اتصال پایه‌های LCD به میکرو

پایه‌های LCD برای اتصال به پایه‌های میکرو بصورت زیر پیکره‌بندی می‌شوند.

CONFIG LCDPIN = PIN , DB4 = PN , DB5 = PN , DB6 = PN , DB7 = PN , E = PN , RS =PN

PN : پایه‌ای دلخواه از میکرو که پایه LCD به آن اتصال می‌یابد به طور مثال PORTB.7

• مثال

```
CONFIG  LCDPIN= PIN , DB4 = PORTB.4 , DB5 = PORTB.5 , DB6 = PORTB.6 , DB7 =_
PORTB.7 , E = PORTB.3 , RS = PORTB.2
```

دقت کنید که پیکره‌بندی پایه‌های LCD باید در یک و یا ادامه آن با علامت _ (UNDER LINE) در خط بعد نوشته شود. پایه‌های LCD (PIN) با توجه به پیکره‌بندی بالا باید به صورت جدول ۶-۸ متصل شود.

جدول ۶-۸ ترکیب پایه‌های LCD

LCD DISPLAY	LCD PIN	PORT
DB7	14	PORTB.7
DB6	13	PORTB.6
DB5	12	PORTB.5
DB4	11	PORTB.4
E(ENABLE)	6	PORTB.3
R/W	5	GROUND
RS	4	PORTB.2
VO(CONTRAST)	3	GROUND OR POT(5K)
VDD	2	VCC
VSS	1	GROUND

تعیین نوع LCD

CONFIG LCD = LCDtype

LCDTYPE می‌تواند انواع زیر باشد :

4*40 = دارای 40 ستون و 4 سطر

16*1a = دارای 16 ستون و 1 سطر است. این نوع LCD ، نوع ویژه‌ای است که بصورت LCD 8*2 استفاده می‌شود که دارای خط دومی در ستون نهم یا آدرس &H8 است.

16*2 = دارای 16 ستون و 2 سطر است که بصورت پیش‌فرض قرار می‌گیرد. اگر از این نوع LCD استفاده شود تعیین نوع LCD اختیاری است .

و نیز می‌تواند از انواع 16*4 – 20*2 – 20*4 – 16*1 باشد.

• مثال

```
CONFIG LCD = 40 * 4
LCD "Hello"                    'display on LCD
FOURTHLINE                     'select line 4
LCD "4"                        'display 4
END
```

پیکره‌بندی باس LCD

CONFIG LCDBUS = constant

دستور فوق را در صورتی که از انتقال داده به LCD بصورت 4 بیتی(پیش‌فرض) یا 8 بیتی استفاده کنیم به کار می‌بریم که CONSTANT می‌تواند عدد 4 برای انتقال اطلاعات بصورت 4 بیتی و عدد 8 برای انتقال اطلاعات بصورت 8 بیتی باشد. زمانی که از انتقال داده 4 بیتی استفاده می‌نمایید نیازی به نوشتن این پیکره‌بندی نیست.

• مثال

```
Config LCDBUS = 4                              '4 bit mode
LCD "hello"
```

دستورات و توابع مربوط به LCD

دستور LCD

این دستور یک یا چند عبارت ثابت یا متغییر را بر روی LCD نمایش می‌دهد.

```
LCD x
LCD "constant"
```

x متغییر و constant ثابتی است که نمایش داده می‌شود.

برای نمایش چند عبارت پشت سر هم بین آنها علامت ; (semicollon) را قرار می‌دهیم.

```
LCD a ; b1 ; "constant"
```

• مثال

```
Dim A As Byte
Config Lcd = 16 * 4                'configure lcd screen
Config Lcdpin = Pin , Db4 = Portb.1 , Db5 = Portb.2 , Db6 = Portb.3 , Db7=_
Portb.4 , E = Portb.5 , Rs =  Portb.6
Cls                                'clear the LCD display
Lcd "Hello world."                 'display this at the top line
Wait 1
Lowerline                          'select the lower line
Wait 1
Lcd "Shift this."                  'display this at the lower line
Wait 1
For A = 1 To 10
Shiftlcd Right                     'shift the text to the right
Wait 1                             'wait 1s
Next
For A = 1 To 10
Shiftlcd Left                      'shift the text to the left
Wait 1                             'wait 1s
Next
Locate 2 , 1                       'set cursor position
Lcd "*"                            'display this
Wait 1                             'wait 1s
Shiftcursor Right                  'shift the cursor
Lcd "@"                           'display this
Wait 1                             'wait 1s
Home Upper                         'select line 1 and return home
Lcd "Replaced."                   'replace the text
Wait 1                             'wait a moment
Cursor Off Noblink                 'hide cursor
Wait 1                             'wait 1s
```

```
Cursor On Blink                    'show cursor
Wait 1                             'wait 1s
Display Off                        'turn display off
Wait 1                             'wait 1s
Display On                         'turn display on
Thirdline
Lcd "Line 3"
Fourthline
Lcd "Line 4"
Home Third                         'goto home on line three
Home Fourth
Home F                             'first letter also works
Locate 4 , 1 : Lcd "Line 4"
End
```

دستور CLS

این دستور مخفف CLEAR SCREEN است که باعث پاک شدن تمام صفحه نمایش LCD می‌شود.

• مثال

```
Cls                          'Clear LCD display
LCD "Hello"                  'show Hello
End
```

دستور DISPLAY

DISPLAY ON / OFF

توسط این دستور می‌توانید صفحه نمایش را روشن (ON) یا خاموش (OFF) کنید.

• مثال

```
Dim a as byte
a = 255
LCD a
Wait 1
DISPLAY OFF
Wait 1
DISPLAY ON
End
```

دستور CURSOR

توسط این دستور می‌توان مکان‌نمای LCD را تنظیم کرد.

CURSOR ON / OFF BLINK / NOBLINK

شما می‌توانید روشن (ON) یا خاموش (OFF) و چشمک‌زدن (BLINK) یا چشمک‌نزدن (NOBLINK) مکان‌نما را تنظیم کنید. در حالت پیش‌فرض مکان‌نما در حالت روشن و چشمک نزدن است.

• مثال

```
Dim a As Byte
a = 255
LCD a
CURSOR OFF              'hide cursor
Wait 1                  'wait 1 second
CURSOR ON BLINK         'blink cursor
End
```

دستور HOME

این دستور مکان‌نما را به ترتیب در اولین ستون سطراول ، سطر دوم ، سطر سوم یا سطرچهارم قرار می‌دهد.

HOME UPPER / LOWER /THIRD / FOURTH

دستورات فوق را به صورت ساده شده زیر نیز می توان نوشت :

HOME U / L / T / F

اگر دستور HOME به تنهایی نوشته شود مکان‌نما در سطر و ستون اول قرار می‌گیرد.

• مثال

```
home lower              'LOCATE 2,1
Lcd "Hello"
Home U                  'LOCATE 1,1
Lcd "Upper"
```

دستور LOCATE

این دستور مکان‌نما را به مکان دلخواه در صفحه LCD می‌برد.

LOCATE X , Y

X ثابت یا متغیری از (1-4) مشخص کننده سطر و y ثابت یا متغیری از (1-64) که مشخص کننده ستون LCD است.

• مثال

```
Lcd  " Hello "
Locate 1 , 10
Lcd  " * "
End
```

دستور SHIFT CURSOR

این دستور مکان‌نمای LCD را یک واحد به چپ یا راست انتقال می‌دهد.

SHIFTCURSOR LEFT / RIGHT

• مثال

```
Lcd "Hello"
Shiftcursor Left
End
```

دستورSHIFTLCD

SHIFTLCD LEFT / RIGHT

این دستور کل صفحه نمایش LCD را یک واحد به چپ یا راست انتقال می‌دهد.

• مثال

```
Lcd "BASCOM"
Shiftlcd Left
Wait 1
Shiftlcd Right
End
```

دستور LOWERLINE

LOWERLINE

این دستور مکان‌نما را به خط پایین‌تر می‌برد.

- **مثال**

```
Lcd "Test"
Lowerline
Lcd "Hello"
End
```

- **دستور UPPERLINE**

UPPERLINE

این دستور مکان‌نما را به خط بالاتر می‌برد.

- **مثال**

```
Dim A As Byte
a = 255
Lcd A
Lowerline
Lcd A
Upperline
End
```

دستور THIRDLINE

THIRDLINE

این دستور مکان‌نما را به خط سوم می‌برد.

- **مثال**

```
Dim A As Byte
A = 255
Lcd A
Thirdline
Lcd A
End
```

دستور FOURTH LINE

در صورت استفاده از LCD چهار سطر این دستورمکان نما را به اول خط چهارم می‌برد. این دستور فقط برای LCD های چهار خط معتبر است.

- **مثال**

```
Dim A As Byte
A = 255
Lcd A
Fourthline
Lcd A
Upperline
End
```

تابع DEFLCDCHAR

با این دستور می‌توانید حرف یا علامتی که خودتان در منوی TOOLS و قسمت LCD DESIGNER محیط BASCOM طراحی نموده‌اید برروی صفحه LCD نمایش دهید. بعد از طراحی حرف یا علامت

دلخواه در LCD DESIGNER و کلیک کردن برروی دکمه OK خط زیر در محیط برنامه نویسی ظاهر خواهد شد.

DEFLCDCHAR ? ,r1,r2,r3,r4,r5,r6,r7,r8

r1 تا r8 با توجه به طراحی، توسط نرم افزار نوشته می‌شوند و شما می‌توانید به جای ؟ عددی بین 0 تا 7 قرار دهید. بدین صورت شما می‌توانید تا 8 کاراکتر را طراحی کنید و برروی LCD نمایش دهید. نمایش کاراکتر طراحی شده توسط دستور LCD CHR(?) بعد از دستور CLS انجام می‌گیرد.

• مثال

```
Deflcdchar 0 , 32 , 32 , 17 , 17 , 31 , 32 , 4 , 32
Cls
LCD Chr ( 0 )                'show the character "ب"
End
```

۶-۵ پیکره‌بندی تایمر/کانترها

AVR ها (بجز MEGA256, MEGA128,MEGA64 و MEGA162 که 4 تایمر دارند) نهایتاً دارای 3 تایمر/کانتر هستند. به علت وجود این سه تایمر/کانتر در میکرو نمونه AT90S8535 در این بخش قصد داریم به معرفی تمام تایمرها و رجیسترهای مربوطه و سپس پیکره‌بندی آنها در محیط BASCOM بپردازیم. در صورت وجود هر یک از تایمرها در میکرویی که شما با آن کار می‌کنید، می‌توانید به راحتی آن را در محیط BASCOM پیکره‌بندی و با آن کار کنید.

تایمر/کانتر صفر

معرفی تایمر/ کانتر صفر و رجیسترهای مربوطه

تایمر/ کانتر هشت بیتی صفر می‌تواند کلاک خود را از سیستم ، تقسیمی از کلاک سیستم و یا از پایه خروجی تامین کند. تایمر/ کانتر صفر توسط رجیستر کنترلی TCCR0 می‌تواند متوقف شود. وقفه‌های تایمر/کانتر توسط رجیستر TIMSK (TIMER/COUNTER1 INTERRUPT MASK ENABLE) می‌توانند فعال/ غیرفعال شوند. پرچم سرریزی (OVER FLOW) در رجیستر TIFR موجود می‌باشد.

زمانی که تایمر/ کانتر از پایه خروجی کلاک دریافت می‌کند، سیگنال خروجی با فرکانس اسیلاتور CPU سنکرون (SYNCHRONIZE) می‌شود. بنابراین برای اطمینان از نمونه‌برداری مناسب، بایستی زمان بین دو کلاک خروجی حداقل برابر یک دوره تناوب کلاک CPU داخلی باشد. کلاک خروجی در لبه بالارونده کلاک داخلی CPU نمونه‌برداری می‌شود.

رجیستر کنترلی تایمر /کانتر صفر - TCCR0 [TIMER/COUNTER0 CONTROL REGISTER]

Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	CS02	CS01	CS00
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

بیت‌های 7..3 : بیت‌های رزرو شده

بیت‌های 2 , 1 , 0 - CS02 , CS01 , CS00 : انتخاب کلاک تایمر/کانتر صفر

این بیت‌ها طبق جدول ۶-۹ مشخص کننده PRESCALE برای TIMRE/COUNTER0 یا به عبارتی کلاک تایمر/ کانتر صفر هستند.

جدول ۶-۹ مشخص کننده PRESCALE برای TIMRE/COUNTER0

CS02	CS01	CS00	DESCRIPTION
0	0	0	STOP , TIMER/COUNTER0 IS STOP
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	EXTERNAL PIN T0 ,FALLING ADGE
1	1	1	EXTERNAL PIN T0 ,RISING ADGE

رجیستر تایمر/ کانتر0 : TCNT0 [TIMER/COUNTER0]

Bit	7	6	5	4	3	2	1	0
	MSB							LSB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

این رجیستر 8 بیتی محتوای تایمر/کانتر را در خود جای می‌دهد. تایمر/کانتر به عنوان UP- COUNTER با قابلیت خواندن / نوشتن استفاده می‌شود.

پیکره‌بندی تایمر/کانتر صفر در محیط BASCOM

پیکره‌بندی به صورت تایمر

```
CONFIG  TIMER0 = TIMER , PRESCALE = 1|8|64|256|1024
```

در این حالت تایمر/کانتر در مُد تایمر با فرکانسهای سیستم تقسیم بر 1 ، 8 ، 64 ، 256 و 1024 کار می‌کند. با دستور START TIMER تایمر شروع به شمردن کرده و با دستور STOP TIMER تایمر را متوقف می‌کنیم. زمانی که تایمر روشن می‌شود تایمر با آخرین مقدار قرار گرفته شده در TCNT0 یا TIMER0 شروع به شمارش می‌کند. با دستور TIMER0 = INITIAL VALUE می‌توان مقدار اولیه‌ای را در تایمر صفر قرار داد و نیز محتوای تایمر/کانتر صفر را می‌توان با دستور VAR = TIMER0 خواند که VAR متغییری از نوع BYTE است.

تایمر پس از شمردن تا مقدار $FF پرچم سرریزی خود را با نام OVF0 یک می‌کند. در صورتی که وقفه سرریزی با دستور ENABLE OVF0 و وقفه سراسری با ENABLE INTERRUPTS فعال شده باشند می‌توان در زمان سرریزی تایمر با دستور ON OVF0 LABLE یا ON TIMER0 LABLE به LABLE پرش کرد و ISR سرریزی را اجرا کرد.

نکته برگشت از وقفه سرریزی با دستور RETURN انجام می‌گیرد.

پیکره‌بندی به صورت کانتر

CONFIG TIMER0 = COUNTER , EDGE =RISING/FALLING

در این پیکره‌بندی تایمر/کانتر صفر بصورت کانتر استفاده شده و می‌توان شمارش آن را با لبه بالارونده یا پایین‌رونده فعال کرد. با انتخاب EDGE = RISING ، اعمال هر لبه بالارونده به پایه T0 باعث می‌شود که محتوای رجیستر TCNT0 یا متغییر COUNTER0 یک واحد افزایش یابد و همچنین با انتخاب EDGE = FALLING ، اعمال هر لبه پایین‌رونده به پایه T0 باعث می‌شود که محتوای رجیستر TCNT0 یا COUNTER0 یک واحد افزایش یابد.

کانتر پس از شمردن تا مقدار $FF و به تعداد $FF+1 پالس، پرچم سرریزی خود را با نام OVF0 یک می‌کند. در صورتی که وقفه سرریزی با دستور ENABLE OVF0 و وقفه سراسری با ENABLE INTERRUPTS فعال شده باشند می‌توان در زمان سرریزی کانتر با دستور ON OVF0 LABLE یا ON COUNTER0 LABLE به LABLE پرش کرد و ISR سرریزی را اجرا کرد. محتوای تایمر/ کانتر صفر را می‌توان با دستور VAR = COUNTER0 خواند که VAR متغییری از نوع BYTE است.

• مثال کانتر

در این مثال تایمر/کانتر صفر بصورت کانتر، شمارنده در لبه پایین‌رونده پیکره‌بندی شده است. در صورتی که یک لبه پایین‌رونده به پایه T_0 اعمال شود ، کانتر یک واحد افزایش می‌یابد.

```
Config Timer0 = Counter , Edge = falling
```

رجیستر Tcnt0 تعداد پالس شمارش شده را نشان می‌دهد که در ابتدا صفر کرده‌ایم.

```
Tcnt0 = 0
Do
    Print Tcnt0             'OR         Print Counter0
Loop Until Tcnt0 >= 10      'When 10 Pulses are Counter The Loop Is Exited
End
```

• مثال تایمر

دراین مثال تایمر/کانتر صفر ، بعنوان تایمر استفاده شده است. کلاک تایمر صفر را می‌توان از کلاک خود سیستم و یا تقسیمی از آن تأمین کرد. این تقسیمات 8 ، 16 ، 256 ، 1024 می‌باشند. به طور مثال با کریستال 8MHZ و PRESCALER = 1024 ، تایمر با فرکانس 8MHZ / 1024 = 7.8125KHZ کار خواهد کرد.

```
$crystal = 8000000
Enable Interrupts
Config Timer0 = Timer , Prescale = 1      'timer0 osc=8MHZ
Stop Timer0
Start Timer0
Do
   Print Tcnt0                            'evry 125ns tcnt0 increses one
Loop
End
```

• مثال تایمر با وقفه

مثال زیر کار با وقفه سرریزی (overflow) تایمر/کانتر صفررا نشان می‌دهد. زمانی که تایمر سرریز شود وقفه سرریزی رُخ می‌دهد و زیربرنامه وقفه اجرا می‌شود.

```
Config Timer0 = Timer , Prescale = 1024
Enable Interrupts          'Global interrupt should Enable ,when using any
'interrupt
Enable Timer0              'enable timer0 interrupt
On Ovf0 Tim0_isr            ' Or On Timer0 Tim0_isr
Do
'your program goes here
Loop
End

'the following code is executed when the timer overflows
Tim0_isr:
Print  " in interrupt routin "
Return
```

تایمر/کانتر یک

معرفی تایمر/ کانتر یک و رجیسترهای مربوطه

تایمر/ کانتر 16 بیتی یک می‌تواند کلاک خود را از سیستم ، تقسیمی از کلاک سیستم و یا از پایه خروجی T1 تامین کند. تایمر/کانتر1 توسط رجیسترکنترلی TCCR1A و TCCR1B می‌تواند متوقف شود. وقفه‌های تایمر/کانترتوسط رجیستر TIMSK (TIMER/COUNTER1 INTERRUPT MASK ENABLE) می‌توانند فعال/غیرفعال شوند.

زمانی که تایمر/ کانتر از پایه خروجی کلاک دریافت می کند ، سیگنال خروجی با فرکانس اسیلاتور CPU سنکرون (SYNCHRONIZE) می‌شود. بنابراین برای اطمینان از نمونه‌برداری مناسب ، بایستی زمان بین دو کلاک خروجی حداقل برابر یک دوره تناوب کلاک CPU داخلی باشد. کلاک خروجی در لبه بالارونده کلاک داخلی CPU نمونه‌برداری می‌شود.

تایمر/ کانتر یک دارای دو خروجی مقایسه‌ای (OUTPUT COMPARE) است که دو رجیستر OCR1A و OCR1B مقدار مقایسه را در خود جای می‌دهند و با محتوای تایمر/ کانتر مقایسه می‌شوند. در زمان تساوی محتوای رجیستر مقایسه ومحتوای تایمر/ کانتر وضعیت پایه‌های خروجی مُد مقایسه‌ای OC1A و OC1B (OUTPUT COMPARE PINS) می‌توانند تغییر یابند.

تایمر/ کانتر همچنین می‌تواند بعنوان PWM (PULSE WIDTH MODULATOR) 8 , 9 یا 10 بیتی استفاده شود. در این مُد پایه‌های OC1A و OC1B بعنوان خروجی PWM به کار برده می‌شوند.

تایمر/کانتر در مُد CAPTURE نیز می‌تواند کار کند. با تحریک پایه ICP (INPUT CAPTURE PIN) می‌توان محتوای تایمر/ کانتر را در رجیستر ورودی CAPTURE (ICR1) قرار داد.

خروجی مقایسه کننده آنالوگ (ANALOG COMPARATOR) نیز می‌تواند به عنوان تریگر ورودی CAPTURE قرار گیرد.

رجیستر کنترلی تایمر/ کانتر A -TCCR1A [TIMER/COUNTER 1 CONTROL REGISTER A]

Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

بیت‌های 6,7 – COMPARE OUTPUT MODE 1 A : COM1A0 , COM1A1

این دو بیت عملکرد پایه خروجی مُد مقایسه‌ای A را در زمان تساوی محتوای رجیستر مقایسه‌ای و محتوای تایمر/ کانتر را طبق جدول ۶-۱۰ نشان می‌دهد. پایه OC1A (OUTPUT COMPAREA PIN1) خروجی مُد مقایسه A است که باید بعنوان خروجی تعریف شود.

بیت‌های 4,5 – COMPARE OUTPUT MODE 1 B : COM1B0 , COM1B1

این دو بیت عملکرد پایه خروجی مُد مقایسه‌ای B را در زمان تساوی محتوای رجیستر مقایسه‌ای و محتوای تایمر/ کانتر را طبق جدول ۶-۱۰ نشان می‌دهد. پایه OC1B (OUTPUT COMPAREB PIN1) خروجی مُد مقایسه‌ای B است که باید بعنوان خروجی تعریف شود.

جدول ۶-۱۰ انتخاب عملکرد پایه‌های خروجی مُد مقایسه‌ای1 (X= A OR B)

COM1X1	COM1X0	DESCRIPTION
0	0	TIMER/COUNTER DISCONNECTED FROM OUTPUT PIN OC1X
0	1	TOGGLE THE OC1X OUTPUT LINE
1	0	CLEAR THE OCX1 OUTPUT LINE(TO ZERO)
1	1	SET THE OC1X OUTPUT LINE(TO ONE)

نکته دو بیت COM1X1, COM1X0 در حالت PWM دارای عملکرد متفاوتی هستند.

بیت‌های 2,3 – بیت‌های رزرو شده

بیت‌های 0,1 – PULSE WIDTH MODULATOR SELECT BITS : PWM11 , PWM10

این دو بیت ، تایمر/ کانتر را بعنوان PWM با توجه به جدول۶-۱۱ به کار می‌برند :

جدول ۶-۱۱ انتخاب مُدهای PWM

PWM11	PWM10	DESCRIPTION
0	0	PWM OPERATION OF TIMER/COUNTER1 IS DISABLE
0	1	TIMER/COUNTER1 IS AN 8-BIT PWM
1	0	TIMER/COUNTER1 IS AN 9-BIT PWM
1	1	TIMER/COUNTER1 IS AN 10-BIT PWM

رجیستر کنترلی تایمر/ کانتر B -TCCR1B [TIMER/COUNTER1 CONTROL REGISTER B]

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

بیت 7 - ICNC1 : INPUT CAPTURE 1 NOISE CANCELER (4CKS)

زمانی که این بیت یک است عملکرد کاهش نویز تریگر ورودی CAPTURE فعال است و زمانی که فعال شود ورودی وارد شده به پایه ICP1 فیلتر می‌شود و خروجی ICP زمانی که چهار نمونه یکسان را در ورودی دریافت کند تغییر می‌یابد. بنابراین سیگنال ورودی CAPTURE باید برای چهار کلاک سیکل سیستم موجود باشد.

بیت 6 - ICES1 : انتخاب لبه ورودی CAPTURE1 INPUT CAPTURE1 EDGE SELECT-

زمانی که بیت ICES1 صفر است ، محتوای تایمر/ کانتر در لبه پایین‌رونده سیگنال تحریک شده به پایه ورودی CAPTURE (ICP) در رجیستر ورودی CAPTURE (ICR1) قرار می‌گیرد. زمانی که بیت ICES1 یک است ، محتوای تایمر/ کانتر در لبه بالارونده سیگنال تحریک شده به پایه ورودی CAPTURE (ICP) در رجیستر ورودی CAPTURE (ICR1) جای داده می‌شود.

بیت‌های 5 , 4 - بیت‌های رزرو شده

بیت 3 - CTC1 : صفر شدن محتوای تایمر/ کانتر در زمان تطابق مقایسه‌ای

زمانی که بیت CTC1 یک باشد ، تایمر/ کانتر در اولین کلاک سیکل پس از تطابق مقایسه با عدد $0000 ری‌ست می‌شود ولی زمانی که این بیت صفر است در زمان تطابق مقایسه تایمر/ کانتر به شمردن ادامه می‌دهد. منظور از تطابق مقایسه زمانی است که محتوای رجیستر مقایسه با محتوای تایمر/کانتر یکسان می‌شود.

• مثال

زمانی که برای تایمر/ کانتر PRESCALE = 1 در نظر گرفته شده باشد و مقدار رجیستر مقایسه A برابر C باشد ، اگر بیت CTC1 یک باشد تایمر به صورت زیر خواهد شمرد:

.....|C-5|C-4|C-3|C-2|C-1|C|0|1|2|.....

زمانی که برای تایمر/ کانتر PRESCALE = 8 در نظر گرفته شده باشد و مقدار رجیستر مقایسه A برابر C باشد ، اگر بیت CTC1 یک باشد تایمر به صورت زیر خواهد شمرد:

.....|C-2،C-2،C-2،C-2،C-2،C-2،C-2،C-2|C-1،C-1،C-1،C-1،C-1،
....|2،2،2،2،2،2،2،2|1،1،1،1،1،1،1،1|2،2،2،2،2،2،2،2|0،0،0،0،0،0،0،0|1،1،1،1،1،1،1،1|2،2،2،2،2،2،2،2|
C-1،C-1،C-1|C|0،0،0،0،0،0،0،0|1،1،1،1،1،1،1،1|2،2،2،2،2،2،2،2|....

نکته این بیت در مُد PWM تأثیری ندارد.

بیت‌های2 , 1 , 0 - CS12 , CS11 , CS10 : انتخاب کلاک TIMER/COUNTER 1

این بیتها طبق جدول ۶- ۱۲ PRESCALE تایمر/ کانتر و یا به عبارتی فرکانس کاری تایمر/ کانتر را با توجه به فرکانس اسیلاتور مشخص می‌سازند.

زمانی که تایمر/کانتر از پایه خروجی کلاک دریافت می‌کند، بایستی تنظیمات مربوطه با توجه به جدول ۶-۱۲ صورت گیرد.

جدول ۶-۱۲ انتخاب کلاک تایمر/کانتر ۱

CS12	CS11	CS10	DESCRIPTION
0	0	0	STOP , TIMER/COUNTER1 IS STOP
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	EXTERNAL PIN T1 ,FALLING ADGE
1	1	1	EXTERNAL PIN T1 ,RISING ADGE

رجیستر تایمر/ کانتر1 [TIMER/COUNTER1] -TCNT1H AND TCNT1L

Bit	15	14	13	12	11	10	9	8
	MSB							
								LSB
Bit	7	6	5	4	3	2	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Initial Value	0	0	0	0	0	0	0	0

رجیستر16 بیتی TCNT1 محتوای تایمر/ کانتر را در خود جای می‌دهد. تایمر/ کانتر بعنوان یک شمارنده UP-COUNTER و UP/ DOWN COUNTER در حالت PWM با قابلیت خواندن/ نوشتن به کار برده می‌شود.

رجیستر خروجی مقایسه‌ای A تایمر/ کانتر 1 – OCR1AH , OCR1AL

Bit	15	14	13	12	11	10	9	8
	MSB							
								LSB
Bit	7	6	5	4	3	2	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Initial Value	0	0	0	0	0	0	0	0

رجیستر خروجی مقایسه‌ای B تایمر/ کانتر 1 – OCR1BH , OCR1BL

Bit	15	14	13	12	11	10	9	8
	MSB							
								LSB
Bit	7	6	5	4	3	2	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Initial Value	0	0	0	0	0	0	0	0

رجیسترهای 16 بیتی خروجی مقایسه‌ای تایمر/ کانتر یک، خواندنی/ نوشتنی هستند.

محتوای رجیستر خروجی مقایسه‌ای پیوسته با TCNT1 مقایسه می‌شود. وضعیتی که برای پایه‌های خروجی مقایسه‌ای در زمان تطابق مقایسه اتفاق می‌افتد، در رجیسترهای کنترلی و وضعیت تایمر/ کانتر قابل تنظیم است.

زمانی که CPU بخواهد در رجیسترهای OCR1A یا OCR1B بنویسد از رجیستر موقتی TEMP

استفاده می‌کند. هنگامی که CPU بایت بالا (OCR1AH یا OCR1BH) را بنویسد ابتدا این بایت در رجیستر TEMP قرار گرفته و سپس هنگامی که CPU بایت پایین (OCR1AL یا OCR1BL) را بنویسد، رجیستر TEMP نیز در بایت بالا (OCR1AH یا OCR1BH) جای می‌گیرد.

در زمان تطابق مقایسه (COMPARE MATCH) یعنی زمانی که محتوای رجیستر مقایسه با محتوای تایمر/ کانتر برابر شود ، پرچم وقفه مقایسه (COMPARE INTERRUPT FALG) یک می‌شود.

رجیستر ورودی ICR1H AND ICR1L – CAPTURE

Bit	15	14	13	12	11	10	9	8
	MSB							
								LSB
Bit	7	6	5	4	3	2	1	0
Read/Write	R	R	R	R	R	R	R	R
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0
Initial Value	0	0	0	0	0	0	0	0

اگر لبه بالارونده یا پایین‌رونده سیگنال (طبق تنظیمات لبه ورودی CAPTURE (ICES1)) در پایه ورودی CAPTURE (ICP) دریافت شود ، محتوای تایمر/کانتر (TCNT1) در رجیستر ورودی CAPTURE (ICR1) قرار می‌گیرد و در همان لحظه پرچم وقفه ورودیCAPTURE (ICP1) یک می‌شود.

زمانی که CPU بایت پایین (ICR1L) رجیستر ICR1 را می‌خواند، داده موجود به CPU ارسال می‌شود و بایت بالا (ICR1H) در رجیستر موقتی TEMP قرار می‌گیرد. هنگامیکه CPU بایت بالا (ICR1H) را بخواند، محتوای رجیستر TEMP به CPU فرستاده می‌شود. بنابراین برای دسترسی به 16 بیت ICR1 ، ابتدا بایستی بایت پایین (ICR1L) خوانده شود. از رجیستر TEMP در زمان دسترسی به TCNT1 , OCR1A , OCR1B نیز استفاده شد.

تایمر/کانتر یک در حالت PWM

در مدولاسیون عرض پالس (PULSE WIDTH MODULATOR) دامنه پالسها ثابت و عرض آنها متغییر است بدین صورت که باریکترین پالس نشان دهنده منفی‌ترین مقدار و عریض‌ترین پالس نشان دهنده مثبت‌ترین مقدار است.

زمانی که تایمر/ کانتر یک در حالت PWM استفاده می‌شود ، رجیستر مقایسه A (OCR1A) ورجیستر مقایسه‌ای B (OCR1B) در حالت های 8، 9 یا 10 بیتی برای تولید پالس PWM در پایه‌های OC1A و OC1B استفاده می‌شوند.

تایمر/ کانتر یک در مُد PWM به صورت UP/DOWN COUNTER کار می‌کند. تایمر/ کانتر یک در زمانUP-COUNTER از $0000 تا TOP و در زمان DOWN-COUNTER از TOP تا $0000 می‌شمارد. زمانی که محتوای کانتر با محتوای OCR1A یا OCR1B برابر شد، پایه‌های OCR1A/OCR1B طبق تنظیمات بیتهای COM1A1/COM1A0 یا COM1B1/COM1B0 در رجیستر کنترلی تایمر/ کانتر یک (TCCR1A) ، یک (5.0V) یا صفر (0.0V) می‌شوند. فرکانس پالس PWM نیز با توجه به جدول ۶-۱۳ بدست می‌آید که FTCK1 در این جدول به معنای فرکانس کاری تایمر/کانتر یک است.

با تغییر بیت‌های COM1X1 و COM1X0 می‌توان مُدهای مختلف PWM را طبق جدول ۶-۱۴ انتخاب کرد.

جدول ۶-۱۳ فرکانسهای مختلف پالس PWM

PWM RESOLUTION	TIMER TOP VALUE	FREQUENCY
8-BIT	$00FF(255)	FTCK1/510
9-BIT	$01FF(511)	FTCK1/1022
10-BIT	$03FF(1023)	FTCK1/2046

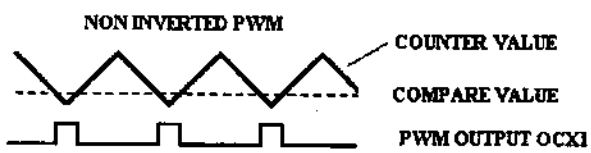
جدول ۶-۱۴ انتخاب مُدهای مختلف پالس PWM

COM1X1	COM1X0	EFFECT ON OCX1
0	0	NOT CONNECTED
0	1	NOT CONNECTED
1	0	CLEAR ON COMPARE MATCH, UP-COUNTING. SET ON COMPARE MATCH ,DOWN COUNTING (NON-INVERTED PWM)
1	1	CLEAR ON COMPARE MATCH, DOWN-COUNTING. SET ON COMPARE MATCH ,UP COUNTING (INVERTED PWM)

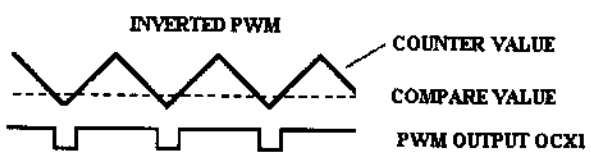
برای درک بیشتر تفاوت INVERTED PWM و NON-INVERTED PWM به جدول ۶-۱۵ و شکلهای ۶-۵ و ۶-۶ توجه کنید.

جدول ۶-۱۵ خروجی پالس PWM به ازاء مقادیر مختلف OCR1X

COM1X1	COM1X0	OCR1X	OUTPUT OC1X
1	0	$0000	L
1	0	TOP	H
1	1	$0000	H
1	1	TOP	L



شکل ۶-۵ خروجی پالس NON INVERTED PWM



شکل ۶-۶ خروجی پالس INVERTED PWM

پیکره‌بندی تایمر/کانتر یک در محیط BASCOM

پیکره‌بندی تایمر/کانتر یک در حالت تایمر

Config Timer1 = Timer , PRESCALE = 1|8|64|256|1024

تایمر UP – COUNTER یک در مُد TIMER به کار برده شده و می‌تواند فرکانس کلاک خود را از فرکانس اسیلاتور بخش بر 1 ، 8 ، 64 ، 256 ، 1024 تامین کند. تایمر پس از شمردن تا مقدار $FFFF پرچم سرریزی خود را با نام OVF1 یک می‌کند. در صورتی که وقفه سرریزی با دستور ENABLE OVF1 و وقفه سراسری با ENABLE INTERRUPTS فعال شده باشند در زمان سرریزی تایمر می‌توان با دستور ON OVF1 LABLE یا ON TIMER1 LABLE به LABLE پرش کرد و ISR سرریزی را اجرا کرد. با دستور VAR = TIMER1 می‌توان محتوای تایمر/ کانتر1 را خواند که VAR متغییری از نوع WORD است . با دستور TIMER1= INITIAL VALUE می‌توان مقدار اولیه‌ای را در تایمر یک قرار داد. در این حالت تایمر از مقدار داده شده شروع به شمردن خواهد کرد.

نکته

- برگشت از برنامه وقفه سرریزی با دستور RETURN انجام می‌گیرد.
- دستور ENABLE TIMER1 تمام وقفه‌های تایمر یک را فعال می‌کند.
- تمام دستورات CONFIG بایستی حتماً در یک خط نوشته شود و یا ادامه آن با علامت _ (UNDER LINE) در خط بعد نوشته شود.

• مثال تایمر

در مثال زیر تایمر/ کانتر در مُد TIMER استفاده شده و میکرو MEGA8535 با کلاک اسیلاتورRC داخلی 1 MHZ کار می‌کند. در این حالت دیگر نیازی به قرار دادن کریستال خارجی در پایه‌های 12 و 13 نیست. فرکانس کلاک تایمر 1MHZ است بنابراین پس از 65536 میکرو ثانیه TIMRE سرریز می‌شود سپس روتین وقفه OVF1 به نام Ovf1routin اجرا می‌شود. با دستور START TIMER1 تایمر یک را به کار انداخته و با دستور STOP TIMER1 آن را متوقف می‌کنیم و از روتین سرریزی با دستور RETURN برگشت می‌کنیم. با دستور TIMER1= INITIAL VALUE می‌توان مقدار اولیه‌ای را در تایمر یک قرار داد.

```
$regfile = "M8535.DAT"
$crystal=1000000
'INTERNAL 1MHZ RC OSC IS DEFAULT
Config Timer1 = Timer , Prescale = 1
Enable Interrupts
Enable Timer1
Enable Ovf1
On Ovf1 Ovf1routin
Start Timer1
Do
Print Timer1
Loop
**** T/C1 OVER FLOW INTERRUPT SERVICE ROUTIN****
Ovf1routin:
Print "OVERFLOW OCCURES"
Return
```

• **مثال**

کریستال داخلی میکرو مثال قبل را به 8 MHZ تغییر داده و Prescale = 8 قرار می‌دهیم. فرکانس کلاک تایمر تغییری نمی‌کند و همان 1MHZ است. مقدار اولیه 64536 در تایمر قرار گرفته بنابراین تایمر پس از 1000 میکروثانیه سرریز می‌شود. در روتین سرریزی دوباره مقدار اولیه را در تایمر قرار داده تا تایمر مدام در هر 1000 میکروثانیه سرریز شود.

```
$regfile = "M8535.DAT"
'WE CHANGE  INTERNAL RC OSC TO 8MHZ AND WORK WITH IT
$Baud=9600
Config Timer1 = Timer , Prescale = 8
Dim A as word
Enable Interrupts
Enable Timer1
Enable Ovf1
On Ovf1 Ovf1routin
Timer1 = 64536
Start Timer1
Do
A = Timer1
Print  A          'or      Print Timer1
Loop

Ovf1routin:
Stop Timer1
Print "OVERFLOW OCCURES"
Timer1 = 64536
Start Timer1
Return
```

پیکره‌بندی تایمر/کانتر یک در حالت کانتر

Config Timer1 = Counter , Eedge = Rising | Falling , Prescale = 1|8|64|256|1024

یا می‌توان چنین نوشت :

Config Timer1 = Counter , Eedge = Rising | Falling

تایمر/ کانتر یک می‌تواند در مُد کانتر نیز کار کند. در این حالت کانتر از پایه ورودی T1 کلاک خود را دریافت می‌کند که می‌تواند نسبت به لبه بالارونده (RISING) یا پایین‌رونده (FALLING) حساس باشد. محتوای کانتر را می‌توان با دستور VAR = COUNTER1 خواند و با دستور COUNTER1 = VAR در محتوای کانتر نوشت. در هر دو حالت VAR متغییر از نوع داده WORD است.

کانتر بعد از شمردن تعداد1 + $FFFF پالس، سرریز می‌شود و سپس پرچم سرریزی خود را با نام OVF1 یک می‌کند. در صورتی که وقفه سرریزی با دستور ENABLE OVF1 و وقفه سراسری با ENABLE INTERRUPTS فعال شده باشند می‌توان در زمان سرریزی کانتر با دستورON OVF1 LABLE یا ON COUNTER1 LABLE به LABLE پرش کرد و ISR سرریزی را اجرا کرد .

• **مثال**

پس از شمرده شدن 10 پالس توسط کانتر در لبه بالارونده حلقه پایان می‌یابد.

```
Config Timer1 = Counter , Edge = Rising
```

```
Counter1 = 0
Do
    Print Counter1
Loop Until Counter1 >= 10     'When 10 Pulses are Count The Loop Is Exited
End
```

پیکره‌بندی تایمر/کانتر یک در مُد مقایسه‌ای (COMPARE)

کانتر یک و مُد مقایسه‌ای

```
Config Timer1 = Counter , Edge = Rising | Falling , Compare A = Clear | Set | Toggle | Disconnect ,_
Compare B = Clear | Set | Toggle | Disconnect , Prescale = 1|8|64|256|1024 , Clear Timer = 1|0
```

طبق پیکره‌بندی بالا تایمر/کانتر یک در حالت کانتر استفاده شده و کلاک خود را از پایه خروجی T1 با لبه بالارونده (RISING) یا پایین‌رونده (FALLING) دریافت می‌کند.

تایمر/ کانتر یک دارای دو رجیستر مقایسه‌ای دو بایتی A و B است که مدام با محتوای تایمر/کانتر مقایسه می‌شوند و زمانی که با هم مساوی شدند (تطابق مقایسه(COMPARE MATCH)) وضعیت پایه‌های خروجی OC1A یا OC1B بنا به تعریف می‌توانند تغییر کنند. محتوای رجیستر مقایسه‌ای A یا B را می‌توان با دستور COMPARE1A | B = VAR تغییر داد که در آن VAR یک عدد ثابت یا یک متغییر نوع BYTE ، WORD یا INTEGER با مقادیر مثبت است. از این رجیسترها می‌توان با دستور VAR = COMPARE1A | B خواند که VAR متغییر نوع WORD است.

Compare A = Clear| Set | Toggle | Disconnect : در زمان تطابق مقایسه (COMPARE MATCH) پایه خروجی OC1A می‌تواند یک (SET) ، صفر (CLEAR) ، معکوس (TOGGLE) و یا ارتباط پایه با کانتر قطع (DISCONNECT) شود.

Compare B = Clear | Set | Toggle | Disconnect : در زمان تطابق مقایسه پایه خروجی OC1B می‌تواند یک (SET) ، صفر (CLEAR) ، معکوس (TOGGLE) ویا ارتباط پایه با کانتر قطع (DISCONNECT) شود.

Clear Timer = 1|0 : با انتخاب گزینه 1 ، محتوای تایمر/کانتریک در زمان تطابق مقایسه‌ای ری‌ست و یا به عبارتی $0000 خواهد شد.

تایمر یک و مُد مقایسه‌ای

```
Config Timer1 = Timer , Compare A = Clear | Set | Toggle | Disconnect , Compare B = Clear | Set |
Toggle | Disconnect , Prescale = 1|8|64|256|1024
```

طبق این پیکره‌بندی تایمر/ کانتر 1 در حالت تایمر استفاده می‌شود و می‌تواند فرکانس کلاک خود را از فرکانس اسیلاتور بخش بر 1 ، 8 ، 64 ، 256 ، 1024 تامین کند. تایمر با فرکانس تعیین شده همانطور که در قسمت پیکره‌بندی TIMER / COUNTER 1 در حالت TIMER گفته شد کار می‌کند. محتوای رجیستر مقایسه‌ای A و B با محتوای تایمر/ کانتر یک مقایسه می‌شوند و زمانی که با هم مساوی شدند (تطابق مقایسه) وضعیت پایه‌های خروجی OC1A یا OC1B بنا به تعریف می‌توانند تغییر کنند. محتوای رجیستر مقایسه‌ای A یا B را می‌توان با دستور COMPARE1A | B = VAR تغییر داد که در آن

VAR می‌تواند یک عدد ثابت یا یک متغییر نوع BYTE ، WORD یا INTEGER با مقادیر مثبت باشد. از این رجیستر می‌توان با دستور VAR = COMPARE1A | B خواند که VAR متغییر نوع WORD است.

Compare A = Clear | Set | Toggle | Disconnect : در زمان تطابق مقایسه پایه خروجی OC1A می‌تواند یک (SET) ، صفر (CLEAR) ، معکوس (TOGGLE) و یا ارتباط پایه با تایمر قطع (DISCONNECT) شود.

Compare B = Clear | Set | Toggle | Disconnect : در زمان تطابق مقایسه پایه خروجی OC1B می‌تواند یک (SET) ، صفر (CLEAR) ، معکوس (TOGGLE) و یا ارتباط پایه با تایمر قطع (DISCONNECT) شود.

Clear Timer = 1|0 : با انتخاب گزینه 1 ، محتوای تایمر/ کانتر یک در زمان تطابق مقایسه‌ای ری‌ست و یا به عبارتی $0000 خواهد شد.

طرز کار با وقفه تطابق مقایسه (COMPARE MATCH)

پرچم وقفه‌های تطابق مقایسه برای هر یک از رجیسترهای A و B متفاوت است. پرچم وقفه تطابق مقایسه رجیستر A ، OC1A و پرچم وقفه تطابق مقایسه رجیستر B ، OC1B نام دارد. برای پرش به روتین وقفه تطابق مقایسه‌ای A | B از دستور ON OC1A|B LABLE استفاده می‌کنیم. زمانی که محتوای رجیسترهای مقایسه‌ای A یا B با محتوای تایمر یا کانتر برابر شود، زیر برنامه وقفه LABLE اجرا خواهد شد.

نکته

برای اجرا شدن وقفه تطابق مقایسه A | B بایستی وقفه‌های تطابق هر یک با دستور ENABLE OC1A و ENABLE OC1B به همراه وقفه سراسری با دستور ENABLE INTERRUPTS فعال شده باشند.

• مثال

```
$regfile = "8535DEF.DAT
$crystal = 8000000
Config Timer1 = Counter , Edge = Falling , Compare A = Set , Compare B=_
Toggle , Prescale = 1
Enable Interrupts
Enable Oc1a
On Oc1a Comparematch
Compare1a = 100
Do
' YOU CAN WRITE YOUR PROGRAM HERE
Loop
End                                                    'end program

Comparematch :
Print "COMPARE MATCH OCCURES"
Return
```

پیکره‌بندی تایمر/کانتر یک در مُد CAPTURE

تایمر/ کانتر یک در مُد CAPTURE نیز می‌تواند کار کند. در این مُد پایه ICP به عنوان ورودی در نظر گرفته می‌شود و زمانی که سیگنالی به این پایه در لبه بالارونده یا پایین‌رونده اعمال شود محتوای

رجیستر تایمر/ کانتر یک در رجیستر دو بایتی CAPTURE جای می‌گیرد و پرچم وقفه CAPTURE یک می‌شود و در صورت فعال بودن وقفه مربوطه، زیربرنامه وقفه اجرا می‌شود.

کانتر یک و مُد CAPTURE

```
Config Timer1 = Counter , Edge = Falling | Rising , Capture Edge = Falling | Rising , Noise Cancel_
= 1 | 0 , Prescale = 1|8|64|256|1024
```

در دستور فوق تایمر/ کانتر یک در حالت COUNTER حساس به لبه بالارونده یا پایین‌رونده در نظر گرفته می‌شود. لبه CAPTURE نیز می‌تواند حساس به لبه بالارونده یا پایین‌رونده قرار گیرد به طور مثال زمانی که از لبه بالارونده (Capture Edge = Rising) استفاده می‌کنید اعمال یک لبه بالارونده به پایه ICP باعث می‌شود که محتوای تایمر/ کانتر یک در همان لحظه در رجیستر CAPTURE قرار گیرد. در صورت استفاده از Noise Cancel می‌توانید آن را 1 قرار دهید.

تایمر یک و مُد CAPTURE

```
Config Timer1 = Timer , Prescale = 1|8|64|256|1024 , Capture Edge = Falling | Rising ,_
Noise Cancel = 1| 0
```

در دستور فوق تایمر/ کانتر یک در حالت TIMER در نظر گرفته می‌شود. لبه CAPTURE نیز می‌تواند حساس به لبه بالارونده یا پایین‌رونده قرار گیرد به طور مثال زمانی که از لبه پایین رونده (Capture Edge = Falling) استفاده می‌کنید اعمال یک لبه پایین‌رونده به پایه ICP باعث می‌شود که محتوای تایمر/ کانتر1 در همان لحظه در رجیستر CAPTURE قرار گیرد. محتوای رجیستر CAPTURE را با می‌توان دستور VAR = CAPTURE خواند و با دستور CAPTURE = VAR می‌توان در این رجیستر نوشت که VAR ثابت یا متغییری دو بایتی است.

طرز کار با وقفه CAPTURE

در صورت اعمال پالس مطلوب به پایه ICP پرچم وقفه CAPTURE یک شده و محتوای تایمر/کانتر در رجیستر CAPTURE قرار می‌گیرد. با دستور ENABLE ICP1 به همراه دستور ENABLE INTERRUPTS می‌توان وقفه CAPTURE را فعال کرد و با دستور ON ICP1 LABLE در زمان رُخ داد CAPTURE به زیربرنامه وقفه LABLE پرش و ISR مربوطه را اجرا کرد.

• مثال

```
$regfile = "8515DEF.DAT
Config Timer1= Timer ,Edge =Falling , Capture Edge= Falling, Prescale=_ 1024
Enable Interrupts
Enable Timer1
Enable Icp1
On Icp1 Captureevent
Start Timer1
Do
' Your programs gose here
Loop
End                                    'end program

Captureevent:
Print Timer1
Return
```

پیکره‌بندی تایمر/کانتر یک در مُد مدولاسیون عرض پالس (PWM)

تایمر/ کانتر یک دارای دو خروجی PWM ، 8 ، 9 و 10 بیتی نیز می‌باشد. در این حالت پایه‌های OC1A و OC1B به عنوان خروجی PWM عمل می‌نمایند.

Config Timer1 = Pwm , Pwm = 8 | 9 | 10, Compare A Pwm = Clear Up| Clear Down | Disconnect , Compare B Pwm = Clear Up| Clear Down | Disconnect , Prescale = 1|8|64|256|1024

PWM می‌تواند 8 ، 9 و 10 بیتی باشد که در مُد 8 ، 9 و 10 بیتی مقدار بالای تایمر به ترتیب $FF ، $1FF و $3FF است.

Clear Up : در صورت استفاده از این گزینه PWM به صورت INVERTED در پایه خروجی OC1A یا OC1B ظاهر می‌شود.

Clear Down : در صورت استفاده از این گزینه PWM به صورت NON- INVERTED در پایه خروجی OC1A یا OC1B ظاهر می‌شود.

Disconnect : در صورت استفاده از این گزینه PWM در زمان تطابق مقایسه از پایه خروجی OC1A یا OC1B قطع می‌شود.

Prescale : برای تولید PWM با فرکانسهای مختلف از این گزینه استفاده می‌شود.

برای تولید PWM می‌توانید در رجیستر PWM که همان رجیسترهای مقایسه‌ای A و B هستند با دستورات PWM1A = VAR ، COMPARE1A = VAR و یا همچنین با دستورات PWM1B = VAR ، COMPARE1B = VAR بنویسید که VAR می‌تواند ثابت یا متغییری 1 یا 2 بایتی باشد.

فرکانس PWM با توجه به معادله‌های زیر بدست می‌آید که Fosc فرکانس کلاک سیستم است :

PWM ، 8 بیتی : PWM FREQUNCY = Fosc / (510 * Prescale)

PWM ، 9 بیتی : PWM FREQUNCY = Fosc / (1022 * Prescale)

PWM ، 10 بیتی : PWM FREQUNCY = Fosc / (2046 * Prescale)

نکته پایه مربوط به پالس PWM را خروجی تعریف نمایید.

• مثال

فرکانس پالس PWM تولید شده برای خروجی A و B در مثال زیر 8000000/8*2046= 488.7585HZ است.

```
$regfile = "M8535.DAT
$CRYSTAL = 8000000
Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Up , Compare B Pwm_
=Clear Down , Prescale = 8
Config Pind.4= Output :Config Pind.5=Output
Do
Pwm1a   =  100
Pwm1b = 200
' Compare1a    =  100
' Compare1b    =  200
Loop : End
```

تایمر/کانتر دو

معرفی تایمر/کانتر دو و رجیسترها

تایمر/ کانتر هشت بیتی دو قابلیت انتخاب کلاک از کلاک سیستم ، تقسیمی از کلاک سیستم یا از پایه‌های خروجی به صورت آسنکرون را داراست. تایمر/ کانتر دو با توجه به تنظیمات رجیستر کنترلی (T/C2 CONTROL REGISTER) می‌تواند متوقف شود. پرچم‌های سرریزی (OVER FLOW) و مُد مقایسه‌ای (COMPARE MODE) در رجیستر TIFR موجود می‌باشند. فعال/غیرفعال کردن وقفه‌های تایمر/ کانتر دو در رجیستر TIMSK قابل تنظیم می‌باشند . از تایمر/ کانتر دو بیشتر برای سرعت‌های پایین و ایجاد زمانهای دقیق با دقت و وضوح بالا استفاده می‌شود.

تایمر/کانتر نیز دارای یک خروجی مقایسه‌ای (OUTPUT COMPARE) که از رجیستر مقایسه‌ای خروجی OCR2 (OUTPUT COMPARE REGISTER) برای مقایسه با محتوای تایمر/ کانتر دو استفاده می‌کند. خروجی مُد مقایسه‌ای تایمر/ کانتر دو پایه OC2 است که در زمان تطابق مقایسه وضعیت پایه OC2 می‌تواند تغییر یابد. ضمناً تایمر/ کانتر در زمان تطابق مقایسه می‌تواند به شمردن خود ادامه دهد و یا با عدد $00 ری‌ست شود.

نکته در حالت تطابق مقایسه ، نوشتن بر روی پایه OC2 نمی‌تواند وضعیت این پایه را تغییر دهد.

تایمر/ کانتر2 همچنین به عنوان PWM (PULSE WIDTH MODULATOR) 8 بیتی استفاده می‌شود.

رجیستر کنترلی تایمر/ کانتر2 – TCCR2 [TIMER/COUNTER2 CONTROL REGISTER]

Bit	7	6	5	4	3	2	1	0
	-	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

بیت 7 - بیت رزرو شده

بیت 6 -PWM2 : فعال کننده PULSE WIDTH MODULATOR ENABLE - PWM

این بیت زمانی که یک است ، مُد PWM برای TIMER/ COUNTER2 فعال است.

بیت 5 , 4 - COM21 , COM 20 : خروجی مُد مقایسه‌ای- COMPARE OUTPUT MODE

این دو بیت مشخص کننده وضعیت پایه OC2 (OUTPUT COMPARE) در زمان تطابق مقایسه هستند. (جدول ۶-۱۶) .

جدول ۶-۱۶ انتخاب عملکرد پایه‌های خروجی مُد مقایسه‌ای 2

COM21	COM20	DESCRIPTION
0	0	TIMER/COUNTER DISCONNECTED FROM OUTPUT PIN OC2
0	1	TOGGLE THE OC2 OUTPUT LINE
1	0	CLEAR THE OC2 OUTPUT LINE(TO ZERO)
1	1	SET THE OC2 OUTPUT LINE(TO ONE)

بیت 3 – CTC2 : صفر شدن محتوای تایمر/ کانتر در زمان تطابق مقایسه

در زمان تطابق مقایسه اگر این بیت یک باشد محتوای تایمر/ کانتر با عدد $00 ری‌ست می‌شود در غیر اینصورت یعنی زمانی که CTC2 صفر است تایمر/ کانتر به شمارش خود ادامه می‌دهد.

• مثال

زمانی که برای تایمر/ کانتر PRESCALE = 1 در نظر گرفته شده باشد و مقدار رجیستر مقایسه برابر C باشد ، اگر بیت CTC2 یک باشد تایمر به صورت زیر خواهد شمرد :

.....|C-5|C-4|C-3|C-2|C-1|C|0|1|2|.....

زمانی که برای تایمر/ کانتر PRESCALE = 8 در نظر گرفته شده باشد و مقدار رجیستر مقایسه برابر C باشد ، اگر بیت CTC2 یک باشد تایمر به صورت زیر خواهد شمرد :

....|C-2 ،C-2 ،C-2 ،C-2 ،C-2 ،C-2 ،C-2 ،C-2|C-1 ،C-1،C-1 ،C-1 ،C-1 ،C-1

.....|2 ،2 ،2 ،2 ،2 ،2 ،2 ،2|1 ،1 ،1 ،1 ،1 ،1 ،1 ،1|0 ،0 ،0 ،0 ،0 ،0 ،0 ،0|C-1 ،C-1|C|0 ،

نکته بیت CTC2 در مُد PWM تأثیری ندارد.

بیت‌های CS22 , CS21 , CS20 : 2 , 1 , 0 : انتخاب کلاک CLOCK SELECT

این بیت‌ها طبق جدول ۶-۱۷ مشخص کننده PRESCALE برای TIMRE/COUNTER2 یا به عبارتی کلاک تایمر/ کانتر2 هستند.

جدول ۶-۱۷ انتخاب کلاک تایمر/کانتر 2

CS22	CS21	CS20	DESCRIPTION
0	0	0	STOP , TIMER/COUNTER2 IS STOPED
0	0	1	PCK2
0	1	0	PCK2/8
0	1	1	PCK2/32
1	0	0	PCK2/64
1	0	1	PCK2/128
1	1	0	PCK2/256
1	1	1	PCK2/1024

رجیستر تایمر/ کانتر دو [TIMER/COUNTER2]-TCNT2

Bit	7	6	5	4	3	2	1	0
	MSB							LSB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

این رجیستر 8 بیتی محتوای تایمر/کانتر را در خود جای می‌دهد. تایمر/کانتر به عنوان UP- COUNTER یا UP - DOWN COUNTER در حالت PWM با قابلیت خواندن/ نوشتن استفاده می‌شود.

رجیستر خروجی مقایسه‌ای تایمر /کانتر2 - OCR2 [TIMER/COUNTER2 OUPUT COMPARE]

Bit	7	6	5	4	3	2	1	0
	MSB							LSB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

این رجیستر 8 بیتی مقدار مقایسه را در خود جای می دهد. محتوای این رجیستر مدام با محتوای تایمر/کانتر (TCNT2) مقایسه می‌شود و تغییراتی که در زمان تطابق مقایسه (زمانی که محتوای OCR2 با TCNT2 یکی شود) بر روی پایه OC2 می‌دهد در رجیستر TCCR2 مشخص شده است.

نکته

نوشتن یک مقدار یکسان در TCNT2 و OCR2 باعث می‌شود که هیچ تطابق مقایسه‌ای روی ندهد. پرچم وقفه مقایسه (COMPARE INTERRUPT FLAG) در اولین کلاک CPU بعد از تطابق مقایسه‌ای یک می‌شود.

تایمر/ کانتر دو در حالت PWM

در مدولاسیون عرض پالس (PULSE WIDTH MODULATOR) دامنه پالسها ثابت و عرض آنها متغیر است بدین صورت که باریکترین پالس نشان دهنده منفی‌ترین مقدار و عریض‌ترین پالس نشان دهنده مثبت‌ترین مقدار است.

زمانی که تایمر/کانتر دو در حالت PWM استفاده می‌شود ، رجیستر مقایسه‌ای OCR2 در حالت‌های 8 بیتی برای تولید PWM در پایه OC2 استفاده می‌شود.

تایمر/کانتر دو در مُد PWM به صورت UP/DOWN COUNTER کار می‌کند. تایمر/کانتر دو در زمان UP-COUNTER از $00 تا $FF و در زمان DOWN-COUNTER از $FF تا $00 می‌شمارد. زمانی که محتوای کانتر با محتوای OCR2 برابر شد، پایه OC2 طبق تنظیمات بیتهای COM21/COM20 در رجیستر کنترلی تایمر/کانتر دو (TCCR2)،یک (5.0V) یا صفر (0.0V) می‌شوند. فرکانس پالس PWM نیز با توجه به جدول ۶-۱۸ بدست می‌آید. FTCK2 فرکانس کلاک تایمر/کانتر دو است.

جدول ۶-۱۸ فرکانسهای مختلف پالس PWM

PWM RESOLUTION	TIMER TOP VALUE	FREQUENCY
8-BIT	$FF(255)	FTCK2/510

می‌توان با تغییر بیت‌های COM1X1 و COM1X0 مُدهای مختلف PWM را طبق جدول ۶-۱۹ انتخاب کرد.

جدول ۶-۱۹ انتخاب مُدهای مختلف پالس PWM

COM21	COM20	EFFECT ON OC2
0	0	NOT CONNECTED
0	1	NOT CONNECTED
1	0	CLEAR ON COMPARE MATCH, UP-COUNTING. SET ON COMPARE MATCH ,DOWN COUNTING (NON-INVERTED PWM)
1	1	CLEAR ON COMPARE MATCH, DOWN-COUNTING. SET ON COMPARE MATCH ,UP COUNTING (INVERTED PWM)

برای درک بیشتر تفاوت INVERTED PWM و NON-INVERTED PWM به جدول ۶-۲۰ و شکل‌های ۶-۵ و ۶-۶ در بخش تایمر/کانتر یک در حالت PWM توجه کنید.

جدول ۶-۲۰ خروجی پالس PWM به ازاء مقادیر مختلف OCR2

COM21	COM20	OCR2	OUTPUT OC2
1	0	$00	L
1	0	$FF	H
1	1	$00	H
1	1	$FF	L

پیکره‌بندی تایمر/کانتر دو در محیط BASCOM

تایمر/ کانتر8 بیتی دو بسته به نوع میکرو می‌تواند در مُدهای TIMER ، COUNTER ، COMPARE و PWM کار کند. در بعضی از میکروها از جمله MEGA103 ، تایمر/ کانتر دو به صورت کانتر نیز می‌تواند کار کند ولی در بعضی دیگر از جمله MEGA32 تایمر/کانتر دو نمی‌تواند در مُد کانتر عمل نماید.

پیکره‌بندی تایمر/کانتر دو

پیکره‌بندی تایمر/کانتر دو در مُد تایمر

CONFIG TIMER2 = TIMER, ASYNC = ON | OFF, PRESCALE = 1 | 8 | 32 | 64 | 128 | 256 | 1024

تایمر UP - COUNTER دو در مُد TIMER به کار برده شده و می‌تواند فرکانس کلاک خود را از فرکانس سیستم بخش بر 1 , 8 , 32 , 64 , 128 , 256 , 1024 تامین کند. تایمر پس از شمردن تا مقدار $FF پرچم سرریزی خود را با نام OVF2 یک می‌کند. با دستور ON OVF2 LABLE در زمان سرریزی تایمر می‌توان به LABLE پرش کرد اگر وقفه سرریزی تایمر با دستور ENABLE OVF2 و وقفه سراسری با دستور ENABLE INTERRUPTS فعال شده باشند. محتوای تایمر/ کانتر دو با دستور VAR =TIMER2 خوانده می‌شود که VAR متغییری از نوع BYTE است. ASYNC زمانی ON انتخاب می‌شود که تایمر به صورت آسنکرون از پایه‌های TOSC2و TOSC1 با کریستال 32768HZ کلاک دریافت می‌کند. در این حالت با PRESCALE=128 دقیقاً تایمر بعد از یک ثانیه OVERFLOW می‌دهد(1s = 32768 / 128*256) که می‌تواند برای طراحی یک ساعت استفاده شود. در غیر اینصورت OFF انتخاب می‌شود.

نکته

برگشت از برنامه وقفه سرریزی با دستور RETURN انجام می‌گیرد.
تمام دستورات CONFIG بایستی حتماً در یک خط نوشته شود و یا ادامه آن با علامت _ (UNDER LINE) در خط بعد نوشته شود.

• مثال تایمر

در مثال زیر تایمر/ کانتر در مُد TIMER قرار می‌گیرد و با فرکانس 1MHZ کریستال RC داخلی میکرو MEGA8 کار می‌کند. در این حالت دیگر نیازی به قرار دادن کریستال خارجی در پایه‌های XTAL1 و XTAL2 نیست. فرکانس کلاک تایمر 1 MHZ است بنابراین تایمر پس از 256 میکرو ثانیه سرریز می‌شود سپس روتین وقفه OVF2 به نام Ovf1routin اجرا می‌شود. با دستورSTART TIMER2 تایمر را به کار انداخته و با دستور STOP TIMER2 آن را متوقف می‌کنیم. از روتین سرریزی با دستور RETURN برگشت می‌کنیم. با دستور TIMER2= INITIAL VALUE می‌توان مقدار اولیه‌ای را در تایمر قرار داد.

```
$regfile = "M8DEF.DAT"
'INTERNAL 1MHZ RC OSC IS DEFAULT AND IF WE WORK WITH IT
$crystal = 1000000
Config Timer2 = Timer , Prescale = 1
Enable Interrupts
Enable Timer2
Enable Ovf2
On Ovf2 Ovf1routin
Start Timer2
Do
Print Timer2
Loop

Ovf1routin:
Print "OVERFLOW OCCURES"
Return
```

• مثال

کریستال داخلی میکرو ATMEGA8 را به 8 MHZ تغییر و Prescale = 8 را قرار می‌دهیم. فرکانس کلاک تایمر 1 MHZ است. مقدار اولیه 6 در تایمر قرار گرفته بنابراین تایمر پس از 250 میکروثانیه سرریز می‌شود. در روتین سرریزی دوبار مقدار اولیه را در تایمر قرار داده تا تایمر مدام هر 250 میکروثانیه سرریز شود.

```
$regfile = "M8def.DAT"     'MEGA8  MCU
'WE CHANGE  INTERNAL RC OSC TO 8MHZ
$crystal = 8000000
Config Timer2 = Timer , Prescale = 8
Dim A As Word
Enable Interrupts
Enable Timer2
Enable Ovf2
On Ovf2 Ovf1routin
Timer2 = 6
Start Timer2
Do
A = Timer2
Print A        ' or print Timer2
Loop

Ovf1routin:
Stop Timer2
```

```
Print "OVERFLOW OCCURES"
Timer2 = 6
Start Timer2
Return
```

پیکره‌بندی تایمر/کانتر دو در مُد کانتر

CONFIG TIMER2 = COUNTER , EDGE = FALLING | RISING , PRESCALE = 1| 8| 64| 256| 1024

یا می‌توان چنین نوشت :

CONFIG TIMER2 = COUNTER , EDGE = FALLING | RISING

تایمر/ کانتر2 در این حالت در مُد کانتر کار می‌کند. در این حالت کانتر از پایه ورودی T2 کلاک می‌خورد. که می‌تواند نسبت به لبه بالارونده (RISING) یا پایین‌رونده (FALLING) حساس باشد. محتوای کانتر با دستور VAR = COUNTER2 خوانده می‌شود و با دستور COUNTER2 = VAR می‌توان در محتوای کانتر نوشت. در هر دو حالت VAR متغییر WORD است. کانتر بعد از شمردن تعداد $FF+1 پالس سرریز می‌شود.

- **مثال**

```
$regfile = "M103DEF.DAT"
Config Timer2 = Counter , Edge = Rising
Counter2 = 0
Do
    Print Counter2
Loop Until Counter2 >= 10    'When 10 Pulses are Counter The Loop Is Exited
End
```

پیکره‌بندی تایمر/کانتر دو در مُد مقایسه‌ای (COMPARE)

تایمر دو و مُد مقایسه‌ای

CONFIG TIMER2 = TIMER , COMPARE = CLEAR | SET | TOGGLE | DISCONNECT_
, PRESCALE = 1 | 8 | 32 | 64 | 128 | 256 | 1024 , CLEAR TIMER = 1|0

توسط پیکره‌بندی فوق تایمر/ کانتر دو در حالت تایمر استفاده شده و می‌تواند فرکانس کلاک خود را از فرکانس سیستم بخش بر 1 , 8 , 32 , 64 , 128 , 256 , 1024 تامین کند. تایمر با فرکانس تعیین شده همانطور که در بخش پیکره‌بندی TIMER / COUNTER 2 در حالت TIMER گفته شد کار می‌کند. محتوای رجیستر مقایسه‌ای با محتوای تایمر/ کانتر دو مقایسه می‌شوند و زمانی که با هم مساوی شدند (تطابق مقایسه) وضعیت پایه خروجی OC2 بنا به تعریف می‌تواند تغییر کند. محتوای رجیستر مقایسه‌ای با دستورOCR2 = VAR قابل تغییر است که در آن VAR می‌تواند یک عدد ثابت یا یک متغییر نوع BYTE ، WORD یا INTEGER با مقادیر مثبت باشد. از این رجیستر می‌توان با دستور VAR = COMPARE خواند که VAR متغییر نوع WORD است.

Compare = Clear | Set | Toggle | Disconnect : در زمان تطابق مقایسه پایه خروجی OC2 می‌تواند یک(SET) ، صفر(CLEAR) ، معکوس(TOGGLE) و یا ارتباط پایه با تایمر قطع(DISCONNEC) شود.

Clear Timer = 1|0 : با انتخاب گزینه 1 ، محتوای تایمر/کانتر دو در زمان تطابق مقایسه‌ای ری‌ست و یا به عبارتی $00 خواهد شد.

کانتر دو و مُد مقایسه‌ای

```
CONFIG  TIMER2 = COUNTER , EDGE = FALLING |RISING, PRESCALE = 1 | 8 | 64 | 256 | _
1024, COMPARE  = CLEAR | SET | TOGGLE | DISCONNECT
```

با این دستور تایمر/ کانتر دو در حالت کانتر استفاده شده و کلاک خود را از پایه خروجی T2 با لبه بالارونده (RISING) یا پایین‌رونده (FALLING) دریافت می‌کند.

تایمر/ کانتر دو دارای رجیسترمقایسه‌ای دو است که مدام با محتوای تایمر/ کانتر مقایسه می‌شوند و زمانی که با هم مساوی شدند (تطابق مقایسه(COMPARE MATCH)) وضعیت پایه خروجی OC2 بنا به تعریف تغییر می‌کند. محتوای رجیستر مقایسه‌ای 2 را می‌توان با دستور OCR2=VAR تغییر داد که در آن VAR یک عدد ثابت یا یک متغییر نوع BYTE ، WORD یا INTEGER با مقادیر مثبت می‌باشد. از این رجیستر می‌توان با دستور VAR = OCR2 خواند که VAR متغییر نوع WORD است.

Compare = Clear | Set | Toggle | Disconnect : در زمان تطابق مقایسه پایه خروجی OC2 می‌تواند یک(SET) ، صفر(CLEAR) ، معکوس(TOGGLE) و یا ارتباط پایه با تایمر قطع (DISCONNECT) شود.

Clear Timer = 1|0 : با انتخاب گزینه1 ، محتوای تایمر/ کانتر2 در زمان تطابق مقایسه‌ای ری‌ست یا به عبارتی $00 خواهد شد.

طرز کار با وقفه تطابق مقایسه (COMPARE MATCH)

پرچم وقفه تطابق مقایسه OC2 نام دارد. برای پرش به روتین وقفه تطابق مقایسه‌ای از دستور ON OC2 LABLE استفاده می‌کنیم. زمانی که محتوای رجیستر مقایسه‌ای با محتوای تایمر برابر شود ، زیر برنامه LABLE اجرا خواهد شد.

نکته

برای اجرا شدن وقفه تطابق مقایسه بایستی وقفه تطابق با دستور ENABLE OC2 و وقفه سراسری با دستور ENABLE INTERRUPTS فعال شده باشند.

• مثال

```
$regfile = "M8DEF.DAT"                  'MEGA8 MCU
$crystal = 8000000
Config Timer2 = Timer , Compare = Set , Prescale = 8
Enable Interrupts
Enable Timer2
Enable Oc2
On Oc2 Comparematch
Start Timer2
OCR2 = 100
Do
Print Timer2
' YOU CAN WRITE YOUR PROGRAM HERE
Loop
End                 'end program

Comparematch:
Print "COMPARE MATCH OCCURES"
Return
```

پیکره‌بندی تایمر/کانتر دو در مُد مدولاسیون عرض پالس (PWM)

تایمر/ کانتر دو دارای خروجی PWM ، 8 بیتی نیز می‌باشد. در این حالت پایه OC2 به عنوان خروجی پالس PWM عمل می‌نماید.

```
CONFIG TIMER2 = PWM , PRESCALE = 1 | 8 | 32 | 64 | 128 | 256 | 1024 , PWM = ON | OFF_
, COMPARE PWM = CLEAR UP| CLEAR DOWN | DISCONNECT
```

Prescale : تعیین فرکانس کلاک تایمر/کانتر که برای تولید PWM با فرکانسهای مختلف از این گزینه استفاده می‌شود.

Pwm = on | off : برای استفاده تایمر در مُد PWM گزینه ON را استفاده می‌نماییم.

Clear Up : در صورت استفاده از این گزینه PWM به صورت INVERTED در پایه خروجی OC2 ایجاد می‌شود.

Clear Down : در صورت استفاده از این گزینه PWM به صورت NON - INVERTED در پایه خروجی OC2 ایجاد می‌شود.

Disconnect : در صورت استفاده از این گزینه PWM در زمان تطابق مقایسه از پایه خروجی OC2 قطع می‌شود.

برای تولید PWM می‌توانید در رجیستر PWM که همان رجیسترهای مقایسه‌ای است با دستور OCR2 = VAR بنویسید که VAR ثابت یا متغییری 1 بایتی است.

فرکانس PWM با توجه به معادله زیر بدست می‌آید که Fosc فرکانس کلاک سیستم است :

PWM ، 8 بیتی : PWM FREQUNCY = Fosc / (510 * Prescale)

تایمر/کانتر سه

پیکره‌بندی تایمر/کانتر سه در محیط BASCOM

تایمر/کانتر سه در بعضی از میکروهای AVR موجود می باشد بنابراین به علت یکسان بودن رجیسترهای این تایمر با دیگر تایمرها مستقیم به پیکره‌بندی آن در محیط BASCOM می پردازیم.

پیکره‌بندی تایمر/کانتر سه در حالت تایمر

```
Config Timer3 = Timer , PRESCALE = 1|8|64|256|1024
```

تایمر UP – COUNTER سه در مُد TIMER به کار برده شده و می‌تواند فرکانس کلاک خود را از فرکانس اسیلاتور بخش بر 1 ، 8 ، 64 ، 256 ، 1024 تامین کند. تایمر پس از شمردن تا مقدار $FFFF پرچم سرریزی خود را با نام OVF3 یک می‌کند. در صورتی که وقفه سرریزی با دستور ENABLE OVF3 و وقفه سراسری با ENABLE INTERRUPTS فعال شده باشند در زمان سرریزی تایمر می‌توان با دستور ON OVF3 LABLE یا ON TIMER3 LABLE به LABLE پرش کرد و ISR سرریزی را اجرا کرد. با دستور VAR = TIMER3 می‌توان محتوای تایمر/ کانتر1 خواند که VAR متغییری از نوع WORD است . با دستور TIMER3= INITIAL VALUE می‌توان مقدار اولیه‌ای را در تایمر یک قرار داد. در این حالت تایمر از مقدار داده شده شروع به شمردن خواهد کرد.

• **مثال**

```
$regfile = "M64DEF.DAT"
$CRYSTAL=8000000
'8MHZ INTERNAL RC OSC IS DEFAULT AND IF WE WORK WITH IT
Config Timer3 = Timer , Prescale = 1
Enable Interrupts
Enable Timer3
Enable Ovf3
On Ovf3 Ovf1routin
Start Timer3
Do
Print Timer3
Loop

**** T/C3 OVER FLOW INTERRUPT SERVICE ROUTIN****

Ovf1routin:
Print "OVERFLOW OCCURES"
Return
```

پیکره‌بندی تایمر/کانتر سه در حالت کانتر

Config Timer3 = Counter , Eedge =Rising |Falling ,Prescale = 1|8|64|256|1024

یا می‌توان چنین نوشت :

Config Timer3 = Counter , Eedge = Rising | Falling

تایمر/ کانتر 3 می‌تواند در مُد کانتر نیز کار کند. در این حالت کانتر از پایه ورودی T3 کلاک خود را دریافت می‌کند که می‌تواند نسبت به لبه بالارونده (RISING) یا پایین‌رونده (FALLING) حساس باشد. محتوای کانتر را می‌توان با دستور VAR = COUNTER3 خواند و با دستور COUNTER3 = VAR در محتوای کانتر نوشت. در هر دو حالت VAR متغییر از نوع داده WORD است.

کانتر بعد از شمردن تعداد1 + $FFFF پالس، سرریز می‌شود و سپس پرچم سرریزی خود را با نام OVF3 یک می‌کند. در صورتی که وقفه سرریزی با دستور ENABLE OVF3 و وقفه سراسری با ENABLE INTERRUPTS فعال شده باشند می‌توان در زمان سرریزی کانتر با دستورON OVF3 LABLE یا ON COUNTER3 LABLE به LABLE پرش کرد و ISR سرریزی را اجرا کرد .

• **مثال**

پس از شمرده شدن 10 پالس توسط کانتر در لبه بالارونده حلقه پایان می‌یابد.

```
Config Timer3 = Counter , Edge = Rising
Counter3 = 0
Do
    Print Counter3
Loop Until Counter3 >= 10     'When 10 Pulses are Count The Loop Is Exited
End
```

پیکره‌بندی تایمر/کانتر سه در مُد مقایسه‌ای (COMPARE)

کانتر سه و مُد مقایسه‌ای

Config Timer3 = Counter , Edge = Rising | Falling , Compare A = Clear | Set | Toggle | Disconnect ,_
Compare B = Clear | Set | Toggle | Disconnect , Prescale = 1|8|64|256|1024 , Clear Timer = 1|0

طبق پیکره‌بندی بالا تایمر/کانتر سه در حالت کانتر استفاده شده و کلاک خود را از پایه خروجی T3 با لبه بالارونده (RISING) یا پایین‌رونده (FALLING) دریافت می‌کند. تایمر/ کانتر سه دارای سه

رجیستر مقایسه‌ای دو بایتی A ، B و C ست که مدام با محتوای تایمر/کانتر مقایسه می‌شوند و زمانی که با هم مساوی شدند (تطابق مقایسه(COMPARE MATCH)) وضعیت پایه‌های خروجی OC3A یا OC3B بنا به تعریف می‌توانند تغییر کنند. محتوای رجیستر مقایسه‌ای A یا B را می‌توان با دستور COMPARE3A | B = VAR (یا OC3A | B = VAR)تغییر داد که در آن VAR یک عدد ثابت یا یک متغییر نوع BYTE ، WORD یا INTEGER با مقادیر مثبت است. از این رجیسترها می‌توان با دستور VAR = COMPARE3A | B (یا VAR= OC3A | B) خواند که VAR متغییر نوع WORD است.

Compare A = Clear| Set | Toggle | Disconnect : در زمان تطابق مقایسه (COMPARE MATCH) پایه خروجی OC3A می‌تواند یک (SET) ، صفر (CLEAR) ، معکوس (TOGGLE) و یا ارتباط پایه با کانتر قطع (DISCONNECT) شود.

Compare B = Clear | Set | Toggle | Disconnect : در زمان تطابق مقایسه پایه خروجی OC3B می‌تواند یک (SET) ، صفر (CLEAR) ، معکوس (TOGGLE) ویا ارتباط پایه با کانتر قطع (DISCONNECT) شود.

Clear Timer = 1|0 : با انتخاب گزینه 1 ، محتوای تایمر/کانتر سه در زمان تطابق مقایسه‌ای ری‌ست و یا به عبارتی $0000 خواهد شد.

تایمر سه و مُد مقایسه‌ای

```
Config Timer3 = Timer , Compare A = Clear | Set | Toggle | Disconnect ,_
Compare B = Clear | Set | Toggle | Disconnect , Prescale = 1|8|64|256|1024
```

طبق این پیکره‌بندی تایمر/ کانتر در حالت تایمر استفاده می‌شود و می‌تواند فرکانس کلاک خود را از فرکانس اسیلاتور بخش بر 1 ، 8 ، 64 ، 256 ، 1024 تامین کند. تایمر با فرکانس تعیین شده همانطور که در قسمت پیکره‌بندی تایمر/ کانتر سه در حالت تایمر گفته شد کار می‌کند. محتوای رجیستر مقایسه‌ای A و B با محتوای تایمر/ کانتر سه مقایسه می‌شوند و زمانی که با هم مساوی شدند (تطابق مقایسه) وضعیت پایه‌های خروجی OC3A یا OC3B بنا به تعریف می‌توانند تغییر کنند. محتوای رجیستر مقایسه‌ای A یا B را می‌توان با دستور COMPARE3A | B = VAR (یا OC3A | B =VAR)تغییر داد که در آن VAR می‌تواند یک عدد ثابت یا یک متغییر نوع BYTE ، WORD یا INTEGER با مقادیر مثبت باشد. از این رجیستر می‌توان با دستورB | VAR = COMPARE3A (یا VAR=OC3A | B) خواند که VAR متغییر نوع WORD است.

Compare A = Clear| Set| Toggle| Disconnect : در زمان تطابق مقایسه پایه خروجی OC3A می‌تواند یک(SET) ، صفر(CLEAR) ، معکوس(TOGGLE) و یا ارتباط پایه با تایمر قطع (DISCONNECT) شود.

Compare B = Clear| Set| Toggle| Disconnect : در زمان تطابق مقایسه پایه خروجی OC3B می‌تواند یک(SET) ، صفر(CLEAR) ، معکوس(TOGGLE) و یا ارتباط پایه با تایمر قطع (DISCONNECT) شود.

Clear Timer = 1|0 : با انتخاب گزینه 1 ، محتوای تایمر/ کانتر سه در زمان تطابق مقایسه‌ای ری‌ست و یا به عبارتی $0000 خواهد شد.

طرز کار با وقفه تطابق مقایسه (COMPARE MATCH)

پرچم وقفه‌های تطابق مقایسه برای هر یک از رجیسترهای A و B متفاوت است. پرچم وقفه تطابق مقایسه رجیستر A ، OC3A و پرچم وقفه تطابق مقایسه رجیستر B ، OC3B نام دارد.

برای پرش به روتین وقفه تطابق مقایسه‌ای A | B از دستور ON OC3A|B LABLE استفاده می‌کنیم. زمانی که محتوای رجیسترهای مقایسه‌ای A یا B با محتوای تایمر یا کانتر سه برابر شود، زیر برنامه وقفه LABLE اجرا خواهد شد. برای اجرا شدن وقفه تطابق مقایسه A | B بایستی وقفه‌های تطابق هر یک با دستور ENABLE OC3A و ENABLE OC3B به همراه وقفه سراسری با دستور ENABLE INTERRUPTS فعال شده باشند.

• مثال

```
$regfile = "M128DEF.DAT"
$crystal = 8000000
Config Timer3 = Counter , Edge = Falling , Compare A = Set , Compare B=_
Toggle , Prescale = 1
Enable Interrupts
Enable Oc3a
On Oc3a Comparematch
Compare3a = 100
Do
' YOU CAN WRITE YOUR PROGRAM HERE
Loop
End                                                    'end program

Comparematch:
Print "COMPARE MATCH OCCURES"
Return
```

پیکره‌بندی تایمر/کانتر سه در مُد CAPTURE

تایمر/ کانتر سه در مُد CAPTURE نیز می‌تواند کار کند. در این مُد پایه IC3 به عنوان ورودی در نظر گرفته می‌شود و زمانی که سیگنالی به این پایه در لبه بالارونده یا پایین‌رونده اعمال شود محتوای رجیستر تایمر/ کانتر سه در رجیستر دو بایتی CAPTURE3 جای می‌گیرد و پرچم وقفه CAPTURE یک می‌شود و در صورت فعال بودن وقفه مربوطه، زیربرنامه وقفه اجرا می‌شود.

کانتر سه و مُد CAPTURE

```
Config Timer3=Counter ,Edge =Falling |Rising,Capture Edge =Falling| Rising , _
Noise Cancel = 1 | 0 Prescale = 1|8|64|256|1024
```

در دستور فوق تایمر/ کانتر سه در حالت COUNTER حساس به لبه بالارونده یا پایین‌رونده در نظر گرفته می‌شود. لبه CAPTURE نیز می‌تواند حساس به لبه بالارونده یا پایین‌رونده قرار گیرد به طور مثال زمانی که از لبه بالارونده (Capture Edge = Rising) استفاده می‌کنید اعمال یک لبه بالارونده به پایه IC3 باعث می‌شود که محتوای تایمر/ کانتر سه در همان لحظه در رجیستر CAPTURE 3 قرار گیرد. در صورت استفاده از Noise Cancel می‌توانید آن را 1 قرار دهید.

تایمر سه و مُد CAPTURE

```
Config Timer3 = Timer , Prescale = 1|8|64|256|1024 , Capture Edge = Falling | Rising , _
Noise Cancel = 1| 0
```

در دستور فوق تایمر/ کانتر سه در حالت TIMER در نظر گرفته می‌شود. لبه CAPTURE نیز می‌تواند حساس به لبه بالارونده یا پایین‌رونده قرار گیرد به طور مثال زمانی که از لبه بالارونده (Capture Edge = Falling) استفاده می‌کنید اعمال یک لبه پایین‌رونده به پایه IC3 باعث می‌شود که محتوای تایمر/ کانتر در همان لحظه در رجیستر CAPTURE قرار گیرد. محتوای رجیستر CAPTURE را با می‌توان دستور VAR = CAPTURE خواند و با دستور CAPTURE 3 = VAR می‌توان در این رجیستر نوشت که VAR ثابت یا متغییری دو بایتی است.

طرز کار با وقفه CAPTURE

در صورت اعمال پالس مطلوب به پایه ICP پرچم وقفه CAPTURE یک شده و محتوای تایمر/کانتر در رجیستر CAPTURE قرار می‌گیرد. با دستور ENABLE ICP3 به همراه دستور ENABLE INTERRUPTS می‌توان وقفه CAPTURE را فعال کرد و با دستور ON ICP3 LABLE در زمان رُخ داد CAPTURE به زیربرنامه وقفه LABLE پرش و ISR مربوطه را اجرا کرد.

• مثال

```
$regfile = "M64DEF.DAT"
Config Timer3= Timer ,Edge =Falling , Capture Edge= Falling, Prescale=_ 1024
Enable Interrupts
Enable Timer3
Enable Icp3
On Icp3 Captureevent
Start Timer3
Do
' Your programs gose here
Loop
End                                    'end program

Captureevent:
Print CAPTURE3
Return
```

پیکره‌بندی تایمر/کانتر سه در مُد مدولاسیون عرض پالس (PWM)

تایمر/ کانتر سه دارای دو خروجی PWM ، 8 ، 9 و 10 بیتی نیز می‌باشد. در این حالت پایه‌های OC3A و OC3B به عنوان خروجی PWM عمل می‌نمایند.

```
Config Timer3 = Pwm , Pwm = 8 | 9 | 10, Compare A Pwm = Clear Up| Clear Down | Disconnect ,
Compare B Pwm = Clear Up| Clear Down | Disconnect , Prescale = 1|8|64|256|1024
```

PWM می‌تواند 8 ، 9 و 10 بیتی باشد که در مُد 8 ، 9 و 10 بیتی مقدار بالای تایمر به ترتیب $FF ، $1FF و $3FF است.

Clear Up : در صورت استفاده از این گزینه PWM به صورت INVERTED در پایه خروجی OC3A یا OC3B ظاهر می‌شود.

Clear Down : در صورت استفاده از این گزینه PWM به صورت NON- INVERTED در پایه خروجی OC3A یا OC3B ظاهر می‌شود.

Disconnect : در صورت استفاده از این گزینه PWM در زمان تطابق مقایسه از پایه خروجی OC3A یا OC3B قطع می‌شود.

Prescale : برای تولید PWM با فرکانسهای مختلف از این گزینه استفاده می‌شود.

برای تولید PWM می‌توانید در رجیستر PWM که همان رجیسترهای مقایسه‌ای A و B هستند با دستورات COMPARE3A = VAR ، COMPARE3B = VAR بنویسید که VAR می‌تواند ثابت یا متغییری 1 یا 2 بایتی باشد.

فرکانس PWM با توجه به معادله‌های زیر بدست می‌آید که Fosc فرکانس کلاک سیستم است :

PWM ، 8 بیتی : PWM FREQUNCY = Fosc / (510 * Prescale)

PWM ، 9 بیتی : PWM FREQUNCY = Fosc / (1022 * Prescale)

PWM ، 10 بیتی : PWM FREQUNCY = Fosc / (2046 * Prescale)

• مثال

فرکانس پالس PWM تولید شده برای خروجی A وB در مثال زیر 488.7585HZ =8000000/8*2046 است.

```
$regfile = "M128DEF.DAT"
$CRYSTAL = 8000000
Config Timer3 = Pwm , Pwm = 10 , Compare A Pwm = Clear Up ,Compare B Pwm
=Clear_ Down , Prescale = 8
CONFIG PORTE = OUTPUT
Do
Compare3a = 100
Compare3b = 200
Loop
End
```

۶-۶ ارتباط با پورت سریال

در BASCOM ارتباط سریال می‌تواند در دو حالت UART سخت‌افزاری و UART نرم‌افزاری پیکره‌بندی شود. در حالت سخت‌افزاری پایه‌های RXD(0-1) و TXD(0-1) برای UART0 و UART1 به عنوان پایه ورودی و خروجی داده سریال مورد استفاده می‌شود. در حالت UART نرم‌افزاری پایه RXD و TXD به صورت مجازی توسط کاربر به هر پایه دلخواه انتساب می‌یابد. در این حالت پایه‌های TXD و RXD میکرو می‌توانند به عنوان I/O عادی و یا بعنوان پایه TXD و RXD سریال در کنار پایه‌های مجازی با پورت سریال ارتباط برقرار کنند. به همین منظور پیکره‌بندی UART یا USART در مُد UART را در دو بخش سخت‌افزاری و نرم‌افزاری توضیح داده می‌شود.

UART سخت‌افزاری

در صورت استفاده از این حالت از دستورات زیر می‌توانید برای ارتباط استفاده نمایید.

فعال کردن UART های میکرو

برای فعال کردن UART0 نیازی به دستور خاصی نمی باشد و فقط کافی است که BAUD درست را تعریف و از دستورات موجود برای ارسال داده به پورت استفاده نمایید.

برای فعال کردن UART1 بایستی دستور زیر برای فعال سازی آن استفاده شود:

```
OPEN "com1:" for binary as #1
```

سپس می توان از دستورات موجود برای این UART استفاده کرد.

تعیین میزان باود

```
$BAUD = X       'UART0 BAUD RATE SELECTION
$BAUD1 = X      'UART1 BAUD RATE SELECTION
```

X مقدار باودی است که در ارتباط سریال استفاده می‌شود. این مقدار با فرکانس کریستال رابطهٔ مستقیم دارد و خطای حاصل از آن در ارتباط سریال با مقدار بیش از 1% قابل قبول نمی‌باشد. به طور مثال در صورت استفاده از کریستال 11.059MHZ خطا در باودهای 4800 به بالا صفر می‌باشد بنابراین در ارتباط سریال سعی شود باود انتخاب شده مضربی از کریستال باشد (11.059/9600=1152).

- **مثال**

```
$baud = 9600                    'UART0
$Crystal = 8000000               '8 MHz crystal
Print  "Hello"                   'send  "hello" to serial Port
End
```

تغییر میزان باود در برنامه

```
BAUD = VAR2
```

این دستور برای تغییر میزان باود UART0 در برنامه اضافه می‌شود. مقدار اولیه با دستور $BAUD = X مقداردهی شده است.

- **مثال**

```
$Baud = 2400
$crystal = 14000000              '14 MHz crystal
Print "Hello"                    'send to serial port 2400 bps
Baud = 9600                      'changing  baud rate
Print "Hello"                    'send to serial port 9600 bps
End
```

ارسال داده سریال در حالت UART سخت‌افزاری

پیکره‌بندی SERIALOUT

زمانی که بخواهید از محیط TERMINAL EMULATOR برای نمایش داده گرفته شده از پورت سریال استفاده نمایید، می‌توانید از این پیکره‌بندی استفاده کنید. توسط این دستور می‌توان برای داده‌های ارسالی به پورت سریال کامپیوتر بافری توسط UART سخت‌افزاری در نظر گرفت. حافظه بافر از حافظه SRAM تامین می شود.

```
CONFIG SERIALOUT = BUFFERED , SIZE = size
```

Size مشخص کننده تعداد بایت بافر است.

• مثال

```
$baud = 9600
$crystal = 4000000
'setup to use a serial output buffer
'and reserve 20 bytes for the buffer
Config Serialout = Buffered , Size = 20
'enable the interrupts
Enable Interrupts
Print "Hello world"
Do
   Wait 1
   Print "test"
Loop
End
```

دستور PRINT

این دستور داده را به پورت سریال ارسال می‌کند.

PRINT var ; " constant "

VAR مقدار عددی و constant رشته اختیاری است که به پورت سریال RS – 232 فرستاده می‌شود. شما با استفاده از علامت ; می‌توانید در یک خط مقدارهای مختلفی را بفرستید. پایه‌های سریال RS – 232 میکرو می‌توانند به پورت سریال کامپیوتر وصل شوند، در این صورت شما می‌توانید از TERMINAL EMULATOR نرم‌افزار BASCOM به عنوان خروجی داده سریال استفاده نمایید. دستور PRINT به تنهایی باعث ایجاد یک خط خالی در TERMINAL EMULATOR می شود.

• مثال

```
Dim A As Byte , B1 As Byte , C As Integer
A = 1
Print "print variable a " ; A
Print                          'empty line(new line)
Print "Text to print."         'constant to print
B1 = 10
Print Hex(b1)                  'print in hexa notation
C = &HA000                     'assign value to c
Print Hex(c)                   'print in hex notation
Print C                        'print in decimal notation
C = -32000
Print C
Print Hex(c)
Rem Note That Integers Range From -32767 To 32768
End
```

دستور PRINTBIN

Printbin Var [; Varn]

با این دستور محتوای متغییرVAR و متغییراختیاری VARN به باینری تبدیل می‌شوند و به پورت سریال ارسال می‌شوند.همچنین چندین متغییر می‌تواند فرستاده شود که با علامت ; از یکدیگر جدا می‌شوند.

• مثال

```
Dim A(10) As Byte , C As Byte
For C = 1 To 10
A(c) = A                       'fill array
Next
Printbin A(1)                  'print content
```

دریافت داده سریال در حالت UART سخت‌افزاری(بدون وقفه)

پیکره‌بندی SERIALIN

زمانی که بخواهید از محیط TERMINAL EMULATOR برای ارسال داده به میکرو استفاده نمائید، این دستور می‌تواند به کار برده شود. توسط این دستور می‌توان برای داده‌های دریافتی پورت سریال میکرو، بافری توسط UART سخت‌افزاری در نظر گرفت. حافظه بافر از حافظه SRAM تامین میشود.

CONFIG SERIALIN = BUFFERED , SIZE = size

Size مشخص کننده تعداد بایت بافر است.

- **مثال**

```
$crystal = 4000000
$baud    =  9600
Config Serialin = Buffered , Size = 20
'dim a variable
Enable Interrupts
Dim  Name  As   String   *  10
Print "Start"
  Do
  'get a char from the UART
  Name  =   Inkey ()
  If Name > 0 Then                    'was there a char
  Print  Name                         'print it
  Waitms 100
  End If
  Loop
End
```

دستور WAITKEY

Var = Waitkey( )

این دستور تا زمانی که در بافر سریال (UART) کاراکتری دریافت شود منتظر می‌ماند، پس از دریافت کاراکتر آن را در متغییر var می‌ریزد و برنامه ادامه می‌یابد.

- **مثال**

```
Dim A As Byte
A = Waitkey()          'wait for character
Print A
End
```

دستور INKEY

این دستور مقداراسکی اولین کاراکتر دریافت شده از پورت سریال (RS-232) را برمی‌گرداند.

Var = Inkey()

VAR می‌تواند BYTE، INTEGER، WORD و یا LONG باشد.

نکته در صورت خالی بودن بافر دستور INKEY عدد صفر را بر می‌گرداند.

- **مثال**

```
Do                  'Start Loop
A = Inkey()         'Look For Character
```

```
If A > 0 Then          'Is Variable > 0Print
Print A
End If
Loop                   'Loop Forever
                       'The Example Above Is For The Hardware Uart
```

دستور INPUT

زمانی که از محیط TERMINAL EMULATOR استفاده می‌کنید این دستور به کار برده می‌شود. توسط این دستور می‌توانید زمانی که در TERMINAL EMULATOR محیط BASCOM هستید از صفحه کلید کامپیوتر به عنوان ورودی استفاده کنید بدین صورت که در زمان وارد کردن داده از طریق صفحه کلید در محیط TERMINAL ، داده از طریق پورت سریال به میکرو ارسال می‌شود.توسط دستور INPUT شما می‌توانید هر رشته‌ای را وارد کنید.

INPUT [" prompt "], var [, varn] [NOECHO]

داده گرفته شده از صفحه کلید در متغییر var و متغییر اختیاری varn قرار می‌گیرد. به صورت پیش‌فرض داده در محیط TERMINAL نمایش داده می‌شود ولی در صورت استفاده از دستور NOECHO داده گرفته از صفحه کلید در محیط TERMINAL نمایش داده نخواهد شد.

• مثال

```
$baud = 9600
$crystal = 8000000
Dim C As Integer , D As Byte
Dim S As String * 15

Input "Enter integer " , C
Print C

Input "More variables " , C , D
Print C ; " " ; D

Input "Enter your name " , S
Print "Hello " ; S
Input S Noecho                 'without echo
Print S
End
```

دستور INPUTBIN

این دستور داده باینری را از پورت سریال (RS – 232) می‌خواند.

INPUTBIN var1 [,var2]

زمانی که از محیط TERMINAL EMULATOR استفاده می‌کنید این دستور به کار برده می‌شود. توسط این دستور می‌توانید زمانی که در TERMINAL EMULATOR محیط BASCOM هستید از صفحه کلید کامپیوتر به عنوان ورودی استفاده کنید بدین صورت که در زمان وارد کردن داده از طریق صفحه کلید در محیط TERMINAL داده از طریق پورت سریال به میکرو ارسال می‌شود. داده گرفته شده از صفحه کلید در متغییر var1 ومتغییراختیاری var2 جای می‌گیرد که اگر به عنوان BYTE تعریف شده باشد یک بایت، به ازاء INTEGER دو بایت و اگر یک آرایه تعریف شود به تعداد آرایه‌ها کاراکتر دریافت می‌کند. این دستور برای گرفتن تعداد بایتها در همان خط می‌ایستد.

• مثال

```
Dim A As Byte , C As Integer
Inputbin A , C          'Wait For 3 Characters
End
```

دستور INPUTHEX

این دستور اجازه می‌دهد که در هنگام اجرای برنامه از صفحه کلید ورودی دریافت کنیم.

INPUTHEX [" prompt "] , var [, varn] [NOECHO]

زمانی که از محیط TERMINAL EMULATOR استفاده می‌کنید این دستور به کار برده می‌شود. توسط این دستور می‌توانید زمانی که در TERMINAL EMULATOR محیط BASCOM هستید از صفحه کلید به عنوان ورودی استفاده کنید بدین صورت که در زمان وارد کردن داده در TERMINAL، این دستور داده وارد شده را که می‌تواند بین 0-9 و A-F باشد در متغییر var و متغییر اختیاری varn قرار می‌دهد و داده از طریق پورت سریال به میکرو ارسال می‌شود. PROMPT یک رشته ثابت دلخواه و اختیاری که می‌خواهید قبل از کاراکتر وارد شده در محیط TERMINAL نوشته شود. NOECHO نیز باعث می‌شود که مقدار وارد شده از صفحه کلید در محیط TERMINAL نمایش داده نشود. تعداد بایت ورودی بستگی به تعریف متغییرهای var و varn دارد که به صورت زیر تعیین می‌شوند :

- اگر var یا varn بایت تعریف شده باشند ورودی هر یک نهایتاً 2 کاراکتر می‌تواند طول داشته باشد.
- اگر var یا varn ، INTEGER یا WORD تعریف شده باشند، ورودی هر یک نهایتاً 4 کاراکتر می‌تواند طول داشته باشد.
- اگر var یا varn ، LONG تعریف شده باشند، ورودی هر یک نهایتاً 8 کاراکتر می‌تواند طول داشته باشد.

• مثال

```
$baud = 9600
$crystal = 8000000
Dim C As Integer , D As Byte
Dim S As String * 15
Inputhex "Enter hex number (4 bytes) " , C
```

رشته Enter hex number (4 bytes) در Terminal Emulator نوشته می‌شود و دستور Inputhex داده را از صفحه کلید گرفته و آن را در C می‌ریزد و از طریق پورت سریال به میکرو ارسال می‌کند .

```
Print C
Inputhex "Enter hex byte (2 bytes) " , D
Print D
Inputhex "More variables " , C , D
Print C ; " " ; D
```

دقت کنید که در هیچ یک از دستورات بالا عبارت NOECHO نوشته نشده بود و زمانی که شما در محیط TERMINAL EMULATOR هستید با زدن دکمه‌ای از KEYBOARD کاراکتر شما نمایش داده می‌شود ولی زمانی که از دستور NOECHO استفاده می‌نمایید کاراکتری در محیط TERMINAL EMULATOR نوشته نمی‌شود ولی کاراکتر شما در بافر جای خواهد گرفت.

```
Inputhex S Noecho               'without echo
Print S
End
```

دریافت داده سریال در حالت UART سخت‌افزاری(با وقفه)

زمانی که از دستور WAITKEY() استفاده می کنید PC COUNTER در خط این دستور منتظر دریافت داده سریال است که این خود در بعضی از برنامه ها مشکل بزرگی است.دستور INKEY نیز در پاره ای از اوقات کار خود را به خوبی انجام نمی دهد لذا در استفاده از وقفه پورت سریال اهمیت زیادی می یابد.

دستور ENABLE URXC و ENABLE URXC1

جهت فعال کردن وقفه دریافت UART0 از دستور ENABLE URXC و وقفه دریافت UART1 از دستور ENABLE URXC1 استفاده می شود.

تعریف برچسب اینتراپت

برای پرش به زیربرنامه وقفه دریافت از پورت سریال از دستورات زیر استفاده می کنیم :

ON URXC LABEL0 پرش به برچسب LABEL0 در زمان وقفه دریافت پورت سریال UART0

ON URXC1 LABEL1 پرش به برچسب LABEL1 در زمان وقفه دریافت پورت سریال UART1

خواندن محتوای بافر UART

توسط دو رجیستر UDR برای UART0 و رجیستر UDR1 برای UART1 می توان محتوای بافر را در یک متغییر بایتی قرار داد.

- **مثال**

```
Dim a as byte
Dim b as byte
A=udr
B=udr1
```

برنامه کامل دریافت اینتراپتی

حال می خواهیم برنامه کامل از دریافت وقفه ای پورت سریال برای UART0,1 بنویسیم.

- **مثال**

```
$regfile="m64def.dat"
$crystal=8000000
$baud=9600
enable interrupts
enable urxc
on urxc serial_data_recive_by_uart0
dim a as byte
do
'your program gose here
print a         'prints whats recives from Uart0
wait 1
loop
end'end program

serial_data_recive_by_uart0:
a=udr
return
```

• مثال

```
$regfile="m64def.dat"
$crystal=8000000
$baud1=9600
enable interrupts
OPEN "com1:"  for binary as #1
enable urxc1
on urxc1 serial_data_recive_by_uart1
dim a as byte
do
'your program goes here
print #1,a        'prints whats recives from Uart1
wait 1
loop
end'end program

serial_data_recive_by_uart1:
a=udr1
return
```

UART نرم‌افزاری

برای ایجاد پایه‌های UART نرم‌افزاری باید طبق دستور زیر آن را باز کنیم.

```
Open " device " For Mode As # Channel
Close # Channel
```

DEVICE : برای به کار بردن UART نرم‌افزاری، برای کامپایلر باید مشخص باشد که شما کدام پایه را با چه سرعتی و برای چه جهتی (ورودی یا خروجی) برای ارتباط سریال استفاده می‌کنید.
طرز پیکره‌بندی UART نرم‌افزاری بصورت زیر است :

```
Open "COMpin:speed,data,parity,stopbits[,INVERTED] " For Mode As # Channel
```

pin نام پایه پورت است و stopbits می‌تواند یک یا دو بیت ، انتقال داده (data) می‌تواند هفت یا هشت بیت داده باشد و بیت پریتی (parity) می‌تواند N برای NONE ، O برای ODD (فرد) و E برای EVEN (زوج) استفاده شود. پارامتر اختیاری INVERTED منطق 232 – RS را معکوس می‌کند.
MODE : می‌تواند گزینه INPUT برای بکارگیری pin بعنوان پایه RXD مجازی و یا گزینهOUTPUT برای بکارگیری pin بعنوان پایه TXD مجازی باشد.
CHANNEL : عدد مشخص کننده کانال باز شده است که باید حتماً عددی مثبت باشد. بنابراین در دستور COMB.0 : 9600,8,N,2 از پایه صفر پورت B با نرخ انتقال9600 و دو بیت STOP BIT و هشت بیت داده برای تبادل سریال توسط UART نرم‌افزاری استفاده می‌شود. دستورات PRINT ، INPUT ، INPUTHEX ، INKEY ، WAITKEY کانالهای فوق را نیز پوشش می‌دهند. هر کدام از کانالهای باز شده بعد از استفاده با دستور CLOSE #CHANNLE بسته می‌شوند.

• مثال

```
Open "COMD.1:9600,8,N,1,INVERTED" For Output As #1
Close #1
```

پایه PORTD.1 به عنوان خروجی داده سریال کانال 1 برای ارتباط سریال با نرخ انتقال 9600 (baud rate) ، یک بیت STOP، هشت بیت داده و منطق معکوس RS-232 استفاده میشود. سپس توسط دستور close #1 بسته می‌شود.

• مثال

```
Dim B As Byte

'open channel for output and use inverted logic

Open "comd.1:9600,8,n,1,inverted" For Output As #1
Print #1 , B
Print #1 , "serial output "
Close #1

'Now open a pin for input and use inverted logic

Open "comd.2:9600,8,n,1,inverted" For Input As #2
Input #2 , B          'noeho
Close #2

'use normal hardware UART for printing
Print B
End
```

تعین میزان باود

$BAUD #CHANNEL , X

X مقدار باودی است که در ارتباط سریال استفاده می‌شود. این مقدار با فرکانس کریستال رابطهٔ مستقیم دارد و خطای حاصل از آن در ارتباط سریال با مقدار بیش از 1% قابل قبول نمی‌باشد. به طور مثال در صورت استفاده از کریستال 11.059MHZ خطا در باودهای 4800 به بالا صفر می‌باشد بنابراین در ارتباط سریال سعی شود باود انتخاب شده مضربی از کریستال باشد. CHANNEL کانال باز شده توسط UART نرم‌افزاری می‌باشد.

• مثال

```
$baud #1, 9600
$CRYSTAL = 8000000              '8 MHz crystal
Print  #1, "Hello"              'send  "hello" to serial Port
End
```

تغییر میزان باود در برنامه

BAUD #CHANNEL , VAR2

از این دستور برای تغییر میزان باود در برنامه اضافه می‌شود. مقدار اولیه باود با دستور BAUD #CHANNEL,X قرار داده شده است. VAR2 میزان باود برای جایگزینی با مقدار اولیه است.

• مثال

```
$BAUD #1, 2400
$crystal = 8000000              '8 MHz crystal
Print #1,"Hello"                 'send to serial port 2400 bps
Baud #1, 9600                    'changing  baud rate
Print #1,"Hello"                 'send to serial port 9600 bps
End
End
```

ارسال داده در حالت UART نرم‌افزاری

دستور PRINT

این دستور از طریق UART نرم‌افزاری، داده به پورت سریال 232 – RS ارسال می‌کند.

PRINT #CHANNEL ,var

که VAR متغیر، مقدار عددی یا رشته اختیاری است که به پورت سریال 232 – RS ارسال می‌شود. پایه‌های تعیین شده برای UART نرم‌افزاری می‌تواند به پورت سریال کامپیوتر وصل شود، در این صورت شما می‌توانید از TERMINAL EMULATOR نرم‌افزار BASCOM به عنوان خروجی استفاده نمایید.

نکته

استفاده از PRINT var در کنار دستور PRINT #CHANNEL ,var به معنای ارسال داده از پورت سریال اصلی میکرو(پایه TXD) است.

• مثال

```
Dim A As Byte ,B As Byte
Open "comd.1:9600,8,n,1,inverted" For Output As #1
Open "comd.2:9600,8,n,1,inverted" For Input As #2
A = 1
Print #1 , A
Print #1,"Text to print."            'constant to print
Do
B = Inkey(# 2 )
'when the value > 0 we got something
If  B  >  0  Then
Print   # 1  ,   Chr ( b )          'print the character
End   If
Loop   Until  B  =  27
Close   # 2
Close   # 1
'use normal hardware UART for printing
Print B
End
```

دستور PRINTBIN

Printbin #channel ,Var [; Varn]

channel شماره کانال باز شده توسط UART نرم‌افزاری است. محتوای متغیرVAR ومتغییر اختیاری VARN به باینری تبدیل و به پورت سریال نرم‌افزاری فرستاده می‌شوند. همچنین چندین متغییر می‌تواند فرستاده شود که با علامت ; از یکدیگر جدا می‌شوند.

• مثال

```
Dim A(10) As Byte , C As Byte
Open "comd.1:9600,8,n,1,inverted" For Output As #1
For C = 1 To 10
A(c) = A                    'fill array
Next
Printbin #1 ,A(1)           'print content of A(1) to serial port(PORTD.1)
```

دریافت داده در حالت UART نرم‌افزاری

دستور WAITKEY

Var = Waitkey(#Channel)

این دستور تا زمانی که در بافر UART نرم‌افزاری (SOFT UART) با شماره کانال Channel ، کاراکتری دریافت شود منتظر می‌ماند و پس از دریافت کاراکتر آن را در متغییر var قرار می‌دهد و اجرای برنامه از خط بعد ادامه پیدا می‌کند.

• مثال

```
Dim A As Byte
Open "comd.2:9600,8,n,1,inverted" For Input As #2
A = Waitkey(#2)                    'wait for character
Print A
End
```

دستور INKEY

این دستور مقداراسکی اولین کاراکتر دریافت شده از پورت سریال نرم‌افزاری با شماره کانال Channel، را در متغییر VAR قرار می‌دهد و اجرای برنامه از خط بعد ادامه پیدا می‌کند.

Var = Inkey(#Channel)

VAR می‌تواند BYTE ، INTEGER ، WORD و یا LONG باشد.

نکته در صورت خالی بودن بافر دستور Inkey(#Channel) عدد صفر را بر می‌گرداند.

• مثال

```
Open "comd.2:9600,8,n,1,inverted" For Input As #2
Do                          'Start Loop
A = Inkey(#2)               'Look For Character
If A > 0 Then               'Is Variable > 0
Print A                     'Send A to hardware Uart(TXD PIN)
End If
Loop                        'Loop Forever
```

دستور INPUT

INPUT #Channel , var

این دستور همانند دستور INPUT در UART سخت‌افزاری عمل می‌نماید که #Channel شماره کانال UART نرم‌افزاری است. با این دستور می‌توان هر نوع داده‌ای را وارد کرد.

این دستور به صورت پیش فرض NOECHO است و در صورت وارد کردن داده توسط صفحه کلید کامپیوتر در محیط TERMINAL EMULATOR نمایش داده نمی‌شود.

• مثال

```
$regfile = "M32DEF.dat"
$crystal = 8000000
Open "COMB.0:9600,8,N,1" For Input As #1
Open "COMB.1:9600,8,N,1" For Output As #2
```

```
Dim S As String * 10
Print #2 , "INPUT "
Input #1 , S
Print #2,S
End
```

دستور INPUTBIN(#CHANNEL)

این دستور مقدار باینری را از پورت سریال (RS – 232) می‌خواند.

INPUTBIN #channel , var1 [,var2]

زمانی که از محیط TERMINAL EMULATOR استفاده می‌کنید این دستور به کار برده می‌شود. توسط این دستور می‌توانید زمانی که در TERMINAL EMULATOR محیط BASCOM هستید از صفحه کلید به عنوان ورودی استفاده کنید بدین صورت که در زمان وارد کردن داده در TERMINAL این دستور داده وارد شده را به باینری تبدیل کرده و از طریق پورت سریال به UART نرم‌افزاری با شماره کانال Channle میکرو ارسال می‌کند. مقدار باینری خوانده شده از پورت سریال نرم‌افزاری در متغییر var1 ومتغییر اختیاری var2 جای می‌گیرد که اگر به عنوان BYTE تعریف شده باشند یک بایت ، به ازاء INTEGER دو بایت و اگر یک آرایه تعریف شوند به تعداد آرایه‌ها کاراکتر دریافت می‌کنند. این دستور برای گرفتن تعداد بایتها در همان خط می‌ایستد.

- **مثال**

```
Dim A As Byte , C As Integer
Open "COMB.0:9600,8,N,1" For Input As #1
Inputbin #1,A , C           'Wait For 3 Characters
End
```

دستور INPUTHEX(#CHANNEL)

این دستور اجازه می‌دهد که در هنگام اجرای برنامه از صفحه کلید ورودی دریافت کنیم.

INPUTHEX #Channel , var [, varn]

زمانی که از محیط TERMINAL EMULATOR استفاده می‌کنید این دستور به کار برده می‌شود. توسط این دستور می‌توانید زمانی که در TERMINAL EMULATOR محیط BASCOM هستید از صفحه کلید به عنوان ورودی استفاده کنید بدین صورت که در زمان وارد کردن داده در TERMINAL این دستور داده وارد شده را که می‌تواند بین 0-9 و A-F باشد در متغییر var و متغییر اختیاری varn قرار می‌دهد و داده از طریق پورت سریال به UART نرم‌افزاری با شماره کانال Channel میکرو ارسال می‌کند.

این دستور NOECHO است و در صورت وارد کردن داده توسط صفحه کلید کامپیوتر داده ورودی در محیط TERMINAL EMULATOR نمایش داده نمی‌شود. تعداد بایت ورودی بستگی به تعریف متغییرهای var و varn دارد که به صورت زیر تعیین می‌شوند :

- اگر var یا varn بایت تعریف شده باشند ورودی هر یک نهایتاً 2 کاراکتر می‌تواند طول داشته باشد.
- اگر var یا varn ، INTEGER یا WORD تعریف شده باشند، ورودی هر یک نهایتاً 4 کاراکتر می‌تواند طول داشته باشد.
- اگر var یا varn ، LONG تعریف شده باشند، ورودی هر یک نهایتاً 8 کاراکتر می‌تواند طول داشته باشد.

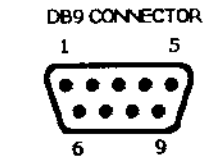
• مثال

```
$baud = 9600
$crystal = 8000000
Open "COMB.0:9600,8,N,1" For Input As #1
Dim C As Integer
Inputhex #1 , C
Print C
End
```

اتصال AVR به RS232

در این قسمت قصد داریم اتصال AVR ها را به کانکتور RS-232 نشان دهیم. همانطور که میدانید استاندارد RS-232 با TTL سازگار نیست بنابراین برای تبدیل سطح ولتاژ RS-232 به سطح ولتاژ TTL یا بالعکس از تراشه MAX232 یا MAX233 استفاده می‌شود.

میکروهای AVR دارای USART یا UART ، دو پایه دارند که برای ارسال و دریافت داده سریال به کار می‌روند. این دو پایه TXD و RXD نامیده می‌شوند. پایه TXD برای ارسال و پایه RXD برای دریافت داده سریال استفاده می‌شود.



1 DATA CARRIER DETECT
2 RECEIVE DATA (RXD)
3 TRANSMIT DATA (TXD)
4 DATA TERMINAL READY(DTR)
5 GND
6 DATA SET READY
7 REQUEST TO SEND
8 CLEAR TO SEND
9 RING INDICATOR

ترکیب پایه های کانکتور DB9

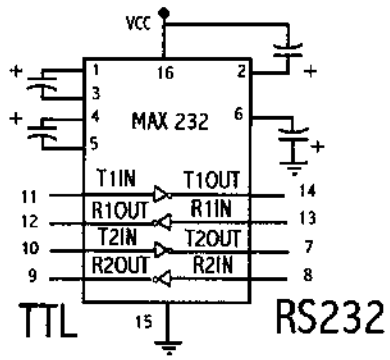
تراشه MAX232

چون منطق RS-232 با میکروهای AVR سازگار نیست از تراشه MAX232 برای تبدیل سطوح ولتاژ استفاده می شود. یکی از مزایای این تراشه این است که از منبع تغذیه 5 ولت استفاده می‌کند.

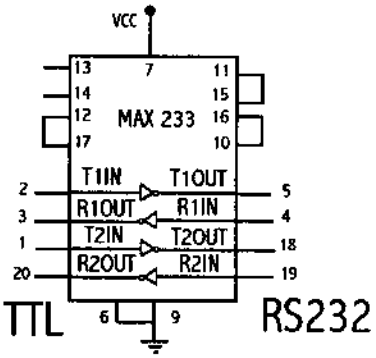
MAX232 دارای دو مجموعه راه‌انداز برای ارسال و دریافت داده است. MAX232 به چهار خازن 1 تا 22µF نیاز دارد که بیشترین نوع مصرفی خازن22µF است.

تراشه MAX233

تراشه MAX233 هم مانند MAX232 عمل می‌نماید با این تفاوت که دیگر نیازی به خازن‌های خارجی نیست.



شکل ۶- ۹ پایه های تراشه MAX232



شکل ۶-۱۰ پایه های تراشه MAX233

۶-۷ مبدل آنالوگ به دیجیتال(ANALOG TO DIGITAL CONVERTOR)

متداول‌ترین انواع ADC ها به قرار زیر است :

۱. مبدل ADC نوع شمارشی (COUNTING ANALOG TO DIGITAL CONVERTOR)
۲. مبدل ADC نوع تقریبهای متوالی (SUCCESSIVE – APPROXIMATION CONVERTOR)
۳. مبدل ADC با مقایسه موازی (PARALLEL – COMPARATOR ADC)
٤. مبدل ADC دو شیبه (DUAL – SLOP OR RATIOMETRIC ADC)

مبدل نوع SUCCESSIVE – APPROXIMATION

مبدل آنالوگ به دیجیتال داخلی میکروهای AVR که ADC دارند از این نوع است به همین دلیل قصد داریم در مورد این نوع ADC مختصری توضیح دهیم.

بجای شمارنده در این طرح از یک میکروکنترلر یا میکروپروسسور استفاده می‌شود.

با برنامه‌ای MSB یک شده و در یک DAC به آنالوگ تبدیل شده، خروجی DAC در مقایسه‌گر با سیگنال آنالوگ مقایسه می‌شود و اگر خروجی DAC بزرگتر باشد، MSB صفر شده و MSB بعدی 1 می‌شود و مقایسه می‌شود و اگر کوچکتر باشد MSB ، 1 باقی مانده وMSB بعدی 1 می‌شود و این عمل به همین ترتیب ادامه پیدا می‌کند تا سیگنال آنالوگ خروجی DAC با سیگنال آنالوگ حاضر در پایه ADC برابر شود.

مبدل آنالوگ به دیجیتال داخلی میکرو

خصوصیات مبدل آنالوگ به دیجیتال داخلی AVR به شرح زیر است:

- وضوح 10 بیتی
- صحت مطلق ±2 LSB
- زمان تبدیل (CONVERSION TIME) 65 – 260 µs
- وضوح 15 KSPS در بالاترین حد

- کانالهای مولتی پلکس شده
- مُدهای تبدیل FREE و SINGLE
- ولتاژ ورودی از 0 V تا VCC
- پرچم وقفه پایان تبدیل ADC
- حذف کننده نویز (NOISE CACELER)

ADC بسته به میکرو به چند کانال آنالوگ مالتی پلکس شده که به هر یک از پایه‌های پورت اجازه می‌دهد که به عنوان یک ورودی مبدل آنالوگ به دیجیتال عمل نماید. مبدل داخلی میکرو دارای وضوح 10 بیتی است و برای تبدیل با این وضوح ، نیاز به فرکانس کلاکی بین 50KHZ تا 200KHZ دارد و این کلاک را از تقسیم فرکانس کریستال تامین می‌کند. در صورت که نیاز به وضوح بالا(کمتر از 10 بیت) نیست می‌توان کلاکی بالاتر از 200KHZ به آن اعمال کرد.ADC دارای یک SAMPLE AND HOLD است که باعث می‌شود ولتاژ ورودی ADC در زمان تبدیل در سطح ثابت نگه داشته شود تا عملیات تبدیل با دقت بیشتری انجام شود.

ADC دارای دو منبع ولتاژ آنالوگ مجزا است. AVCC و AGND که AGND بایستی به زمین یا ولتاژ زمین آنالوگ متصل شود و AVCC نباید بیشتر ا ز 0.3V ± نسبت به VCC اختلاف داشته باشد.

ولتاژ مرجع (VOLTAGE REFERENCE) خارجی در صورت وجود باید به پایه AREF وصل شود که این ولتاژ بایستی بین ولتاژ موجود بر روی پایه های AVCC – AGND باشد. در غیراینصورت به VCC وصل می‌شود. ADC مقدار آنالوگ ورودی را با تقریب متوالی به مقدار دیجیتال 10 بیتی تبدیل می‌کند. کمترین مقدار، نشان دهنده مقدار آنالوگ موجود در پایه AGND و بالاترین مقدار، نشان دهنده ولتاژ پایه AREF منهای یک LSB است.

به طور مثال اگر پایه به ولتاژ AREF = 3.5V و AGND=0V وصل شده باشد ، مقدار دیجیتال شده 1023 نشان دهنده ولتاژ 3.5V و مقدار 0 نشان دهنده ولتاژ 0.0V بر روی پایه مبدل ADC انتخاب شده است که برای محاسبه مقدار دیجیتال ولتاژ موجود بر روی پایه ADC به فرمول زیر توجه کنید :

ADC(Digital value of Analog On ADC Pin)=(1024*Vin)/Vref

در فرمول فوق Vin ولتاژ موجود بر روی پایه ADC و Vref ولتاژ پایه Vref است.

ADC دارای دو مُد تبدیل SINGLE و FREE است. مُد SINGLE بایستی توسط کاربر پیکره‌بندی و کانال دلخواه برای نمونه‌برداری انتخاب شود. در مُد FREE ، ADC با یک ثابت نمونه‌برداری، رجیستر داده ADC را UPDATE می‌کند.

تکنیک‌های کاهش نویز ADC

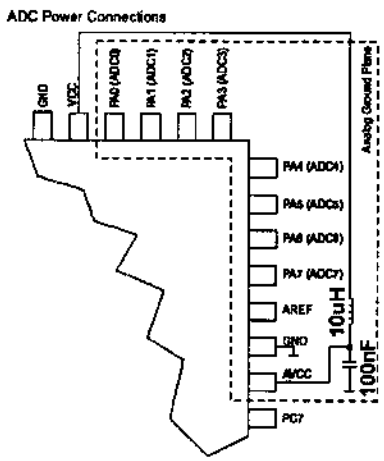
مدارات دیجیتال در داخل و خارج میکرو ایجاد نویز کرده و ممکن است در صحت اندازه‌گیری ADC تاثیر بگذارند. اگر صحت و دقت تبدیل قابل قبول نیست، با استفاده از تکنیک‌های زیر می‌توان نویز را کاهش داد.

۱. بخش آنالوگ چیپ و تمام قسمت‌های آنالوگ باید دارای زمین جداگانه باشند. این زمین‌ها با زمین دیجیتال به وسیله یک مسیر به هم متصل می‌شوند.

۲. مسیرهای آنالوگ را تا می‌توانید کوتاه کنید و از تماس نداشتن مسیرهای آنالوگ و زمین‌های آنالوگ اطمینان پیدا کنید و آنها را از مسیرهای دیجیتال با فرکانس بالا دور کنید.

۳. پایه AVCC میکرو را به VCC با فیلتر پایین گذر نشان داده شده در شکل ۶-۱۱ وصل نمایید.

٤. از مُدهای SLEEP و حالت ADC NOISE CANCELER برای کاهش نویز القاء شده توسط CPU استفاده کنید.



شکل ۶-۱۱ طرز اتصال پایه AVCC به پایه VCC با فیلتر پایین گذر برای کاهش نویز

پیکره‌بندی ADC در محیط BASCOM

مبدل آنالوگ به دیجیتال باید برای میکروهایی که ADC دارند ،توسط این دستور پیکره‌بندی شود تا بتوان از آن استفاده نمود. مبدل داخلی میکرو دارای وضوح 10 بیتی است و برای تبدیل با این وضوح ، نیاز به کلاکی بین 50KHZ تا 200KHZ دارند واین کلاک را از تقسیم کریستال تامین می‌کند.

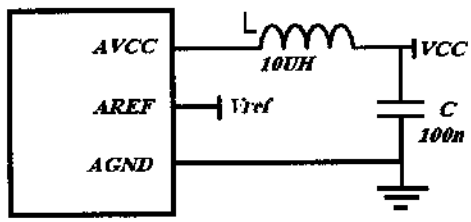
CONFIG ADC = SINGLE| FREE , PRESCALER = AUTO, REFERENCE = OPTIONAL

CONFIG ADC = SINGLE | FREE : برای تبدیل سیگنال آنالوگ خود به دیجیتال می‌توانید از دو مُد FREE و SINGLE استفاده نمایید. زمانی که مُد SINGLE را انتخاب می‌کنید ، باید از دستور GETADC( ) استفاده کنید.

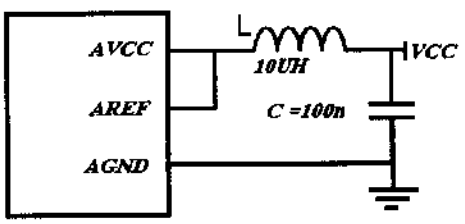
PRESCALER : این گزینه کلاک ADC را مشخص می‌کند. با قرار دادن PRESCALER = AUTO کامپایلر با توجه به فرکانس اسیلاتور ، بهترین کلاک را برای ADC تعیین می‌کند. دیگر مقادیر معتبر 2 ، 4 ، 8 ، 16 ، 32 ، 64 یا 128 می‌باشند.

REFERENCE = OPTIONAL : گزینه‌ای اختیاری برای ولتاژ مرجع (VOLTAGE REFERENCE) در بعضی از میکروها از جمله MEGA8 است. OPTIONAL می‌تواند گزینه های زیر باشد :

OFF : برای خاموش کردن ولتاژ مرجع داخلی(INTERNAL REFERENCE TURNED OFF) و استفاده از ولتاژ موجود بر روی پایه AREF به عنوان ولتاژ مرجع انتخاب می‌شود. در این حالت اتصال پایه های AVCC و AREF به صورت شکل ۶-۱۲ است.



شکل ۶-۱۲



شکل ۶-۱۳

AVCC : زمانی که ولتاژ پایه AVCC به عنوان ولتاژ مرجع در نظر گرفته می‌شود. در این حالت اتصال پایه های AVCC و AREF به صورت شکل ۶-۱۳ است.

INTERNAL : زمانی که ولتاژ مرجع داخلی 2.56V با خازن خارجی بر روی پایه AREF استفاده شود. در این حالت اتصال پایه های AVCC و AREF به صورت شکل ۶-۱۳ است.انتخاب این گزینه برای میکروهایی که ولتاژ مرجع داخلی ندارند، هیچ تاثیری ندارد.

دستور GETADC

با این دستور سیگنال آنالوگ وارده شده به کانالهای (7 - 0) به مقدار دیجیتال تبدیل می‌شود و در متغیر var از نوع داده WORD قرار می‌گیرد.

```
var = GETADC( channel )
```

var نتیجه تبدیل و channel کانال مبدل آنالوگ به دیجیتال داخلی انتخاب شده است که می‌تواند بین 0 تا 7 باشد. این دستور فقط برای چیپهای AVR که دارای مبدل آنالوگ به دیجیتال هستند به طور نمونه ATMEGA32 استفاده می‌شود. از پورتهای آنالوگ به دیجیتال می‌توانید به عنوان پورتهای ورودی و خروجی استفاده کنید ولی هنگامی که پورت به عنوان ADC پیکره‌بندی می‌شود دیگر نمی‌توانید از آن به عنوان I/O استفاده کنید.

دستورات START و STOP

توسط دستورADC START ، ADC شروع به نمونه‌برداری از سیگنال آنالوگ کرده و توسط STOP ADC تغذیه را از ADC قطع می‌کنیم و به عملیات تبدیل پایان می‌دهیم. این دستورات برای شروع و توقف ADC بایستی نوشته شود.

- **مثال**

```
$Regfile="M32def.dat"
Config Adc = Single , Prescaler = Auto 'Now Give Power To The Chip
```

```
Stop Adc
Start Adc              'With Stop Adc , You Can Remove The Power From The Chip
Dim W As Word , Channel As Byte
Channel = 0
Do
W = Getadc( channel )       'Now Read A / D Value From Channel 0 to 7
Print "Channel " ; Channel ; " value " ; W
Incr Channel
If Channel > 7 Then Channel = 0
Loop
END
```

کار با وقفه ADC

زمانی که کار نمونه‌برداری ADC از سیگنال آنالوگ به پایان رسید ، ADC پرچم اتمام تبدیل خود به نام ADC را یک می‌کند. با فعال کردن وقفه سراسری با دستور ENABLE INTERRUPTS و فعال کردن وقفه ADC با دستور ENABLE ADC می‌توان به زیربرنامه وقفه ADC توسط ON ADC LABLE پرش کرد که LABLE نام دلخواه زیربرنامه وقفه ADC یا ISR مربوطه است. می‌توان برای کاهش نویز سیستم میکرو را در زمان نمونه‌برداری در مُد IDLE یا ADC NOISE REDUCTION قرار داد و سپس میکرو خودکار با بالا رفتن پرچم وقفه اتمام تبدیل ADC از این مُد بیدار شده و مقدار دیجیتال شده را در متغیر نوع WORD قرار می‌دهد.

• مثال

```
$regfile = "4433def.dat"
$crystal = 4000000
$baud = 9600
'configure single mode and auto prescaler setting
'The AUTO feature, will select the highest clockrate possible
'the ADC needs a clock from 50-200 KHz
Config Adc = Single , Prescaler = Auto
On Adc Adc_isr
Enable Adc
Enable Interrupts
Dim W As Word , Channel As Byte
'now read A/D value from channel 0
Channel = 0
'Now give power to the ADC
Start Adc
Do
      W=Getadc(channel)
      'idle will put the micro into sleep
      'an interrupt will wake the micro
      Idle
Loop
End

Adc_isr:
   Print "Channel " ; Channel ; " value " ; W
   Waitms 500
Return
```

در صورتی که بخواهید سیگنال آنالوگی به طور مثال دما را اندازه بگیرید، بایستی از سیگنال خود چندین نمونه گرفته و از آنها میانگین گیری کنید. با این کار هم خطا و پایداری مقدار دیجیتال خود را بهبود می بخشید. مثلا در هر یک ثانیه ۱۰ نمونه با فاصله ۱۰۰ میلی ثانیه گرفته و مقدارها را با هم جمع کرده و بر ۱۰ تقسیم نمایید. برای درک بیشتر به فصل پروژه های عملی رجوع کنید.

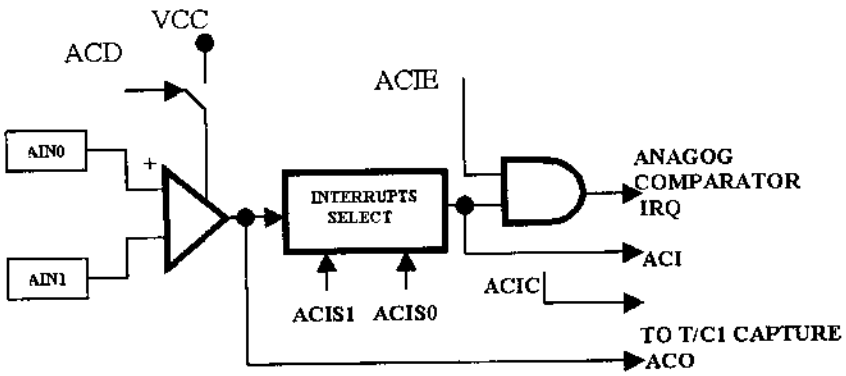
۸-۶ مقایسه‌کننده آنالوگ

مقایسه‌کننده آنالوگ و رجیسترها

مقایسه‌کننده آنالوگ (ANALOG COMPARATOR) مقادیر آنالوگ در دو ورودی پایه مثبت (AIN0) و پایه منفی (AIN1) را با هم مقایسه می‌کند. زمانی که ولتاژ موجود در ورودی مثبت (AIN0) بالاتر از ولتاژ موجود در ورودی منفی (AIN1) باشد ، خروجی مقایسه‌کننده (ACO) یک می‌شود.

مقایسه‌کننده همچنین دارای یک پرچم وقفه مجزا است. کاربر می‌تواند نحوه تریگر شدن وقفه خروجی مقایسه‌کننده را در لبه بالارونده، پایین‌رونده یا TOGGLE انتخاب کند.

خروجی مقایسه‌کننده می‌تواند به عنوان تریگر ورودی CAPTURE تایمر/کانتر یک نیز استفاده شود.



بلوک دیاگرام مقایسه‌کننده آنالوگ

پیکره‌بندی مقایسه‌کننده آنالوگ در BASCOM

مقایسه‌کننده آنالوگ باید توسط دستور زیر پیکره‌بندی شود تا بتوان از آن استفاده نمود. مقایسه‌کننده آنالوگ ، ولتاژ موجود در دو پایه مثبت (AIN0) و پایه منفی (AIN1) را با هم مقایسه می‌کند.

CONFIG ACI = ON|OFF, COMPARE = ON|OFF, TRIGGER =TOGGLE|RISING|FALLING

CONFIG ACI = ON|OFF : باید در زمان استفاده از مقایسه‌کننده ON باشد.

COMPARE = ON|OFF :در صورت انتخابON، خروجی مقایسه‌کننده آنالوگ(ACO) مستقیماً به ورودی CAPTURE تایمر/کانتر یک وصل است و در صورت انتخاب OFF این اتصال برقرار نیست.

TRIGGER =TOGGLE|RISING|FALLING : این گزینه نحوه روی دادن وقفه مقایسه‌کننده را نشان می دهد.

در صورت انتخاب FALLING ، یک لبه پایین‌رونده در خروجی مقایسه‌کننده باعث یک شدن پرچم وقفه مقایسه‌کننده و اجرا شدن برنامه وقفه خواهد شد.

در صورت انتخاب RISING ، یک لبه بالارونده در خروجی مقایسه‌کننده باعث یک شدن پرچم وقفه مقایسه‌کننده و اجرا شدن برنامه وقفه خواهد شد.

در صورت انتخاب TOGGLE ، یک (HIGH) به صفر (LOW) یا صفر به یک شدن خروجی مقایسه‌کننده باعث یک شدن پرچم وقفه مقایسه‌کننده و اجرا شدن برنامه وقفه خواهد شد.

کار با وقفه مقایسه‌کننده آنالوگ

پرچم وقفه مقایسه‌کننده ACI نام دارد. این وقفه با دستور ENABLE ACI و دستور فعال کننده وقفه سراسری ENABLE INTERRUPTS فعال می‌شود. توسط دستور ON ACI LABLE می‌توان در زمان رویداد وقفه مقایسه‌کننده به LABLE پرش و توسط دستور RETURN از ISR مربوطه برگشت کرد.

• مثال

زمانی که خروجی مقایسه‌کننده معکوس شود ، وقفه مقایسه‌کننده روی داده و زیر برنامه Analoginit اجرا می‌شود. در این مثال خروجی مقایسه‌کننده به capture تایمر/کانتر یک متصل نیست.

```
$regfile = "m32def.dat"
$crystal = 8000000
Config Aci = On , Compare = off , Trigger = Toggle
$baud = 9600
Enable Interrupts
Enable Aci
On Aci Analoginit
Do
Loop
End       'end program

Analoginit:
Print "on aci"
Return
```

۹-۶ پیکره‌بندی GRAPHICAL LCD DISPLAY

پیکره‌بندی T6963 GRAPHICAL LCD DISPLAY

برای راه‌اندازی LCD گرافیکی از پیکره‌بندی زیر استفاده می‌نماییم. پیکره‌بندی LCD بر اساس چیپ T6963C که در اکثر LCD های گرافیکی استفاده می‌شود، طراحی شده است.

```
Config GRAPHLCD = type , DATAPORT = port , CONTROLPORT= port , CE = pin , CD = pin
, WR = pin , RD=pin, RESET= pin , FS=pin , MODE = mode
```

• مثال

```
Config Graphlcd =240*128,Dataport= Porta ,Controlport = Portc,Ce =2,Cd =_
3, Wr =0 ,Rd = 1 , Reset =4 , Fs = 5 ,Mode = 8
```

TYPE : که می‌تواند انواع 240*128 , 240*64 , 128*128 و 64*128 باشد. برای LCD های نوع SED به طور مثال از 64SED*128 استفاده نمایید.

DATAPORT : مشخص کننده پورتی است که به عنوان ورودی داده LCD استفاده می‌شود. به طور مثال DATAPORT= PORTA که در این صورت پایه‌های D0-D7 از LCD به ترتیب به پایه‌های PORTA.0 – PORTA.7 متصل می‌شود.

طبق مثال صفحه قبل طرز اتصال پایه‌های LCD به میکرو در جدول ۶-۲۱ آمده است.

جدول ۶-۲۱ پایه های GRAPHICAL LCD

PIN.NUM	LCD PIN	CONNECTED TO
1	GND	GND
2	GND	GND
3	+5V	+5V
4	-9V	-9V BY POT
5	WR	PORTC.0
6	RD	PORTC.1
7	C/E	PORTC.2
8	C/D	PORTC.3
9	NC	NC
10	RESET	PORTC.4
11 - 18	D0 – D7	PORTA0-PORTA.7
19	FS	PORTC.5
20	NC	NC

CONTROLPORT : مشخص کننده پورتی است که از پایه‌های آن برای کنترل LCD استفاده می‌شود به طور مثال PORTC.

CE (CHIP ENABLE) : شماره پایه‌ای است که برای فعال کردن چیپ موجود در LCD استفاده می‌شود به طور مثال اگر CONTROLPORT= PORTC باشد ، CE= 0 به معنای اتصال PORTC.0 به پایه CE از LCD می‌باشد.

CD (CODE/ DATA) : شماره پایه‌ای است که برای کنترل کردن پایه CD موجود در LCD استفاده می‌شود به طور مثال اگر CONTROLPORT= PORTC باشد ، CD = 1 به معنای اتصال PORTC.1 به پایه CD از LCD می‌باشد.

WR (WRITE) : شماره پایه‌ای است که برای کنترل کردن پایه WR موجود در LCD استفاده می‌شود به طور مثال اگر CONTROLPORT= PORTC باشد، WR= 2 به معنای اتصال PORTC.2 به پایه WR از LCD می‌باشد.

RD (READ) : شماره پایه‌ای است که برای کنترل کردن پایه RD موجود در LCD استفاده می‌شود به طور مثال اگر CONTROLPORT= PORTC باشد ، RD= 3 به معنای اتصال PORTC.3 به پایه RD از LCD می‌باشد.

FS (FONT SELECT) : شماره پایه‌ای است که برای کنترل کردن پایه FS موجود در LCD استفاده می‌شود به طور مثال اگر CONTROLPORT= PORTC باشد ، FS= 4 به معنای اتصال PORTC.4 به پایه FS از LCD می‌باشد.

RESET : شماره پایه‌ای است که برای کنترل کردن پایه RESET موجود در LCD استفاده می‌شود به طور مثال اگر CONTROLPORT= PORTC باشد ، RESET= 5 به معنای اتصال PORTC.5 به پایه RESET از LCD می‌باشد.

MODE : مشخص کننده تعداد ستون متنی LCD است که می‌توانند 8 یا 6 باشد. زمانی که از عدد 6

استفاده می نمایید نهایتاً X-PIXEL / 6 ستون متنی خواهید داشت و زمانی که از عدد 8 استفاده نمایید نهایتاً X–PIXEL / 8 ستون متنی خواهید داشت. این گزینه همچنین مشخص کننده نوع فونت است. به طور مثال اگر تعداد 24= (X- PIXEL) باشد با MODE = 8 ، تعداد ستونها برابر 3 و در صورت استفاده از MODE = 6 ، تعداد ستونها برابر 4 است.

دستورات کار با LCD

دستور CLS

این دستور تمام صفحه نمایش LCD چه قسمت متنی و چه گرافیکی را پاک می‌کند.

دستور CLS GRAPH

این دستور فقط قسمت گرافیکی را پاک می‌کند .

دستور CLS TEXT

این دستور فقط قسمت متنی را پاک می‌کند.

دستور LCD

این دستور برای نوشتن متن بر روی LCD استفاده می‌شود. این دستور همانند دستور LCD ، برای LCD های ماتریسی عادی عمل می‌کنند.

- **مثال**

```
Locate 1 , 1
Lcd "MCS Electronics"
'And some othe text on line 2
Locate 2 , 1 : Lcd "T6963c support"
Locate 3 , 1 : Lcd "12345678901234567890123456789012345678901234567890"
Locate 16 , 1 : Lcd "write this to the lower line"
```

دستور PSET X , Y , COLOR

این دستور یک PIXEL را در مختصات (X , Y) به ازای COLOR = 0 خاموش و به ازای COLOR = 1 روشن می‌کند. X از 239 – 0 و Y از 127 - 0 می‌تواند تغییر کند.

- **مثال**

```
FOR X = 0 TO 140
    PSET X , 20 , 1      ' SET THE PIXEL
NEXT

FOR X = 0 TO 140
    PSET X , 127 , 1     ' SET THE PIXEL
NEXT
```

دستور LOCATE ROW,COLUMN

این دستور مکان‌نما را در مکان سطر (ROW) و ستون (COLUMN) مشخص شده قرار می‌دهد. ROW می‌تواند از 1 تا 16 تغییر کند. تغییرات COLUMN بستگی به انتخاب MODE دارد که می‌تواند از 1 تا 40 تغییر کند.

دستور CURSOR ON/OFF BLINK/NOBLINK

این دستور برای قسمت‌های متنی استفاده می‌شود. مکان‌نما می‌تواند در حالتهای ON یا OFF و چشمک زدن(BLINK) یا چشمک نزدن (NOBLINK) باشد.

دستور Line(X0 , Y0) –(X1 , Y1) , COLOR

با این دستور از PIXEL اول با مختصات (X0, Y0) به PIXEL دوم با مختصات (X1 , Y1) خطی با رنگ COLOR کشیده می‌شود. 0 = COLOR خط را پاک کرده و به ازای 255 = COLOR خطی با رنگ سیاه رسم خواهد شد.

- **مثال**

```
Line(0 , 0 ) -(239 , 127 ) , 255          ' diagonal line
Line(0 , 127 ) -(239 , 0 ) , 255          ' diagonal line
Line(0 , 0 ) -(240 , 0 ) , 255            ' horizontal upper line
Line(0 , 127 ) -(239 , 127 ) , 255        ' horizontal lower line
Line(0 , 0 ) -(0 , 127 ) , 255            ' vertical left line
Line(239 , 0 ) -(239 , 127 ) , 255          ' vertical right line
```

دستور CIRCLE (X0,Y0) , RADIUS, COLOR

این دستور دایره‌ای به مختصات مرکزیت (X0 , Y0) و شعاع RADIUS و رنگ COLOR رسم خواهد کرد. 0 = COLOR دایره را پاک کرده وبه ازای 255 = COLOR دایره با رنگ سیاه رسم خواهد شد.

- **مثال**

```
For X = 1 To 10
   Circle( 20 , 20 ) , 20 , 255           ' show circle
   Wait  1
   Circle( 20 , 20 ) , 20 , 0             'remove circle
   Wait  1
Next
```

دستور SHOWPIC X, Y , LABLE

برای نمایش عکسی که در منوی TOOLS و قسمت GRAPHIC CONVERTER ذخیره کرده‌اید استفاده می‌شود. X مکان قرارگیری افقی و Y مکان قرار گیری عمودی عکس را نشان می‌دهد. LABLE نام برچسبی است که اطلاعات عکس مورد نظر در آن قرار دارد. زمانی که در پنجره GRAPHIC CONVERTER عکس خود را LOAD کرده اید، دقت کنید که نوع LCD و فونت خود را انتخاب نمایید. گزینه SED نبایستی انتخاب شود چون LCD شما از نوع SED نیست.

برچسب $BGF "FILE.BGF"

اشاره به فایل BGF و یا همان عکس مورد نظر که با فرمت BGF و با نام دلخواه FILE در کنار برنامه اصلی ذخیره شده است، دارد.

- **مثال**

در مثال زیر عکس مورد نظر در محیط GRAPHIC CONVERTER ، LOAD شده و با نام MCS در کنار برنامه زیر ذخیره شده است سپس آن را توسط دستور SHOWPIC بر روی LCD نمایش می‌دهیم .

```
Showpic  0  ,  0  ,  Plaatje
End                                    'end program
This label holds the image data
Plaatje:
'$BGF will put the bitmap into the program at this location
$bgf   "mcs.bgf"
```

پیکره‌بندی SED(KS108) GRAPHICAL LCD DISPLAY

نوع دیگری از LCD های گرافیکی به نام SED موجود می باشند که نسب به نـوع قبلـی ارزانتـر و کـم سرعت تر هستند.این نوع LCD ها دارای دو چیپ هستند و LCD به دو قسمت تقسیم شده است.

به طور مثال در مدل LCD، 128*64SED به دو قسمت 64*64 تقـسیم شـده و بنـابراین LCD دارای دو CHIP SELECT(CS) است. جهت کار بـا ایـن نـوع LCD بایـستی از کتابخانـه GLCDKS108.LBX در فولدر LIB در مسیر نصب BASCOM استفاده نمود. در صورتی که پسوند آن LIB باشد، بایـستی آن را به LBX تغییر دهید. همچنین برنامه FONT8*8.FONT در فولدر SAMPLE در مسیر نـصب BASCOM را به برنامه توسط دستور $INCLUDE اضافه کنید زیرا فونت 8*8 در LCD استفاده می شـود. بنـابراین زمانی که از این نوع LCD استفاده می شود دو فایل زیر در کنار برنامه بایستی قرار گیـرد و در ابتـدای برنامه نوشته شود :

```
$LIB "GLCDKS108.LBX"
$INCLUDE "FONT8*8.FONT"
```

برای راه‌اندازی LCD گرافیکی از پیکره‌بندی زیر استفاده می‌نماییم.

```
Config GRAPHLCD = type, DATAPORT = port , CONTROLPORT= port , CE = pin ,CE2=pin,
CD = pin , RD=pin, RESET= pin ,ENABLE=pin
```

• مثال

```
Config GRAPHLCD = type, DATAPORT = porta , CONTROLPORT= portc , CE = 0 ,CE2= 1,
CD = 2 , RD= 3, RESET = 4 , ENABLE = 5
```

TYPE : که می‌تواند انواع 128*64SEDو 120*64SED باشد.

DATAPORT : مشخص کننده پورتی است که به عنوان ورودی داده LCD استفاده می‌شـود. بـه طـور مثال DATAPORT= PORTA که در این صـورت پایـه‌هـای D0–D7 از LCD بـه ترتیـب بـه پایـه‌هـای PORTA.0 – PORTA.7 متصل می‌شود.

CONTROLPORT : مشخص کننده پورتی است که از پایه‌های آن برای کنترل LCD استفاده می‌شـود. به طور مثال PORTC.

CE (**CHIP ENABLE**) : شماره پایه‌ای است که برای فعال کردن چیپ اول موجود در LCD استفاده می‌شود. به طور مثال اگر CONTROLPORT= PORTC باشد ، CE= 0 به معنای اتصال PORTC.0 به پایه CE از LCD می‌باشد.

CE2 (**CHIP ENABLE 2**) : شماره پایه‌ای است که برای فعال کردن چیپ دوم موجود در LCD استفاده می‌شود. به طور مثال اگر CONTROLPORT= PORTC باشد ، CE2= 1 به معنای اتصال PORTC.1 به پایه CE2 از LCD می‌باشد.

CD (**CODE/ DATA**) : شماره پایه‌ای است که برای کنترل کردن پایه CD موجود در LCD استفاده می‌شود. به طور مثال CD = 2 به معنای اتصال PORTC.2 به پایه CD از LCD می‌باشد.

RD (READ) : شماره پایه‌ای است که برای کنترل کردن پایه RD موجود در LCD استفاده می‌شود به طور مثال اگر CONTROLPORT= PORTC باشد ، RD= 3 به معنای اتصال PORTC.3 به پایه RD از LCD می‌باشد.

RESET : شماره پایه‌ای است که برای کنترل کردن پایه RESET موجود در LCD استفاده می‌شود به طور مثال اگر CONTROLPORT= PORTC باشد ، RESET= 4 به معنای اتصال PORTC.4 به پایه RESET از LCD می‌باشد.

ENABLE : شماره پایه‌ای است که برای کنترل کردن پایه ENABLE موجود در LCD استفاده می‌شود. به طور مثال اگر CONTROLPORT= PORTC باشد ، ENABL= 5 به معنای اتصال PORTC.5 به پایه ENABL از LCD می‌باشد.

طبق مثال صفحه قبل طرز اتصال پایه‌های LCD به میکرو در جدول ۶-۲۲ آمده است.

جدول ۶-۲۲ اتصال SED LCD به پایه‌های میکرو

PIN.NUM	LCD PIN	CONNECTED TO
1	CE	PORTC.0
2	CE2	PORTC.1
3	VSS	GND
4	VDD	+5V
5	VO	LCD DRIVER SUPPLY VOLTAGE
6	D/I	PORTC.2
7	R/W	PORTC.3
8	EN	PORTC.5
9-16	D0 – D7	PORTA0-PORTA.7
17	RESET	PORTC.4
18	VEE	VEE IS CONNECTED BY THE SAME VOLTAGE

دستورات کار با SED LCD

دستور CLS

این دستور تمام صفحه نمایش LCD چه قسمت متنی و چه گرافیکی را پاک می‌کند.

دستور CLS GRAPH

این دستور فقط قسمت گرافیکی را پاک می‌کند .

دستور CLS TEXT

این دستور فقط قسمت متنی را پاک می‌کند.

دستور LCDAT

این دستور برای نوشتن متن بر روی SED LCD استفاده می‌شود.

• مثال

```
lcdat 2 , 1 , "T6963c support"
lcdat 3 , 1 , "1234567890"
lcdat 16 , 1 , "lower line"
```

دستور PSET X , Y , COLOR

این دستور یک PIXEL را در مختصات (X , Y) به ازای COLOR = 0 خاموش و به ازای COLOR = 1 روشن می‌کند. X از 239 – 0 و Y از 127 - 0 می‌تواند تغییر کند.

دستور LOCATE ROW,COLUMN

این دستور مکان‌نما را در مکان سطر (ROW) و ستون (COLUMN) مشخص شده قرار می‌دهد. ROW می‌تواند از 1 تا 16 تغییر کند. تغییرات COLUMN بستگی به انتخاب MODE دارد که می‌تواند از 1 تا 40 تغییر کند.

دستور CURSOR ON/OFF BLINK/NOBLINK

برای قسمت‌هایی متنی استفاده می‌شود. مکان‌نما می‌تواند در حالتهای ON یا OFF و چشمک زدن (BLINK) یا چشمک نزدن (NOBLINK) باشد.

دستور Line(X0 , Y0) –(X1 , Y1) , COLOR

با این دستور از PIXEL اول با مختصات (X0, Y0) به PIXEL دوم با مختصات (X1 , Y1) خطی با رنگ COLOR کشیده می‌شود. COLOR = 0 خط را پاک کرده و به ازای COLOR = 255 خطی با رنگ سیاه رسم خواهد شد.

دستور CIRCLE (X0,Y0) , RADIUS, COLOR

این دستور دایره‌ای به مختصات مرکزیت (X0 , Y0) و شعاع RADIUS و رنگ COLOR رسم خواهد کرد. COLOR = 0 دایره را پاک کرده وبه ازای COLOR = 255 دایره با رنگ سیاه رسم خواهد شد.

دستور SHOWPIC X, Y , LABLE

برای نمایش عکسی که در منوی TOOLS و قسمت GRAPHIC CONVERTER ذخیره کرده‌اید استفاده می‌شود. X مکان قرارگیری افقی و Y مکان قرار گیری عمودی عکس را نشان می‌دهد. LABLE نام برچسبی است که اطلاعات عکس مورد نظر در آن قرار دارد. زمانی که در پنجره GRAPHIC CONVERTER عکس خود را LOAD کرده اید، دقت کنید که نوع LCD و فونت خود را 8*8 انتخاب نمایید. گزینه SED بایستی انتخاب شود چون LCD شما از نوع SED است.

برچسب $BGF "FILE.BGF"

اشاره به فایل BGF و یا همان عکس مورد نظر که با فرمت BGF و با نام دلخواه FILE در کنار برنامه اصلی ذخیره شده است، دارد.

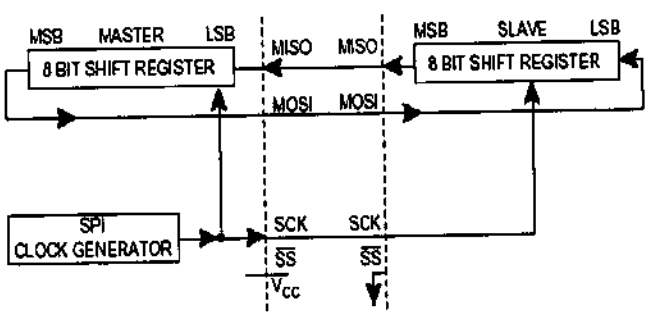
۶-۱۰ ارتباط سریال SPI

ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) یک پروتکل ارتباطی سریال سنکرون با سرعت بالا است که می‌تواند برای ارتباط میکروهای AVR با یکدیگر و یا با وسیله‌های دیگر که قابلیت ارتباط با این نوع پروتکل را دارا هستند به کار برده شود. رجیسترهای مربوط به این ارتباط در AVR ها یکسان است. به همین دلیل در این بخش به معرفی رجیسترهای میکرو نمونه ATMEGA32 پرداخته‌ایم.

خصوصیات

- FULL – DUPLEX ، ارسال داده همزمان (SYNCHRONOUS) سه سیمه (3 – WIRE)
- ارتباط به صورت‌های SLAVE / MASTER
- ارسال ابتدای MSB یا LSB
- بیت‌های قابل برنامه‌ریزی برای تنظیم سرعت
- پرچم وقفه اتمام ارسال
- بیدار شدن از حالت بیکاری (IDLE)

پایه SCK خروجی کلاک برای MASTER و ورودی کلاک برای SLAVE است. با نوشتن رجیستر داده SPI (SPI DATA REGISTER) در MASTER، CPU شروع به تولید کلاک SPI کرده و داده از پایه MOSI (MASTER OUT SLAVE IN) خارج شده و به پایه MOSI در SLAVE وارد می‌شود. بعد از انتقال کامل داده توسط MASTER ، کلاک SPI قطع و پرچم وقفه پایان ارسال داده (SPIF) یک می‌شود و برنامه وقفه اجرا می‌شود. دو شیفت رجیستر (SHIFT REGISTER) 8 بیتی در MASTER و SLAVE را می‌توان به عنوان یک شیفت رجیستر چرخشی 16 بیتی در نظر گرفت. این موضوع در شکل ۶-۱۴ دیده می‌شود. زمانی که داده‌ای از MASTER به SLAVE ارسال می‌شود، می‌توان در همان حال، در جهت مخالف، داده‌ای از SLAVE به MASTER انتقال یابد. بدین صورت که در طول هشت کلاک SPI ، داده های MASTER و SLAVE با هم عوض شود.



شکل ۶-۱۴ طرز اتصال SLAVE/MASTER در ارتباط SPI

طرز اتصال MASTER/ SLAVE

زمانیکه SPI فعال شده باشد جهت پایه‌های $\overline{SS}$ ، SCK ، MISO ، MOSI با توجه به جدول ۶-۲۲ تعیین می‌شوند. SPI در حالت ارسال تک بافره و در حالت دریافت دو بافره می‌باشد. در حالت ارسال زمانی که داده قبلی کاملاً ارسال نشده باشد نمی‌توان در رجیستر داده SPI (SPDR) یا همان شیفت رجیستر نوشت. زمانی که دریافت کامل شد، داده بلافاصله در بافر قرار می‌گیرد. در حالت دریافت، داده پیشین قبل از اتمام دریافت بایت جدید بایستی خوانده شود در غیر اینصورت بایت جدید بر روی بایت قبلی نوشته می‌شود.

جدول ۶-۲۲ نحوه اتصال پایه‌های ارتباط SPI (SPI PIN OVERRIDES)

PIN	DIRECTION , MASTER SPI	DIRECTION , SLAVE SPI
MOSI	USER DEFINED	INPUT
MISO	INPUT	USER DEFINED
SCK	USER DEFINED	INPUT
SS	USER DEFINED	INPUT

طرز کار پایه $\overline{SS}$ در مُد MASTER

زمانیکه SPI به عنوان MASTER استفاده می‌شود (بیت MSTR در SPCR یک است) کاربر می‌تواند جهت پایه $\overline{SS}$ را تعیین کند. اگر $\overline{SS}$ خروجی تعریف شده باشد از آن به عنوان پایه خروجی عادی استفاده می‌شود بدین صورت که هیچ تأثیری در ارتباط SPI ندارد و می‌تواند به عنوان انتخاب SLAVE و یا به عنوان I/O استفاده شود.

اگر پایه $\overline{SS}$ ورودی تعریف شود بایستی حتماً بالا باشد تا عملیات MASTER با اطمینان انجام شود. اگر این پایه زمانی که به عنوان ورودی تعریف شده است بوسیله مدار جانبی دیگر پایین (صفر) شود در حالتی که SPI در مُد MASTER پیکره‌بندی شده است سیستم آن را به عنوان انتخاب MASTER دیگری برای خود تلقی کرده و وقفه می‌دهد و بحالت SLAVE رفته و شروع به ارسال داده برای MASTER دیگر می‌کند. بنابراین زمانی که انتقال SPI بصورت MASTER انجام می‌گیرد و احتمال پایین شدن پایه $\overline{SS}$ وجود داشته باشد وضعیت بیت MSTR باید همیشه قبل از ارسال بیت جدید چک شود که آیا هنوز یک است. هنگامی که بیت MSTR به علت انتخاب MASTER دیگر صفر شده باشد کاربر باید آن را یک کند تا مُد MASTER را دوباره فعال کند.

طرز کار پایه $\overline{SS}$ در مُد SLAVE

زمانی که SPI در حالت SLAVE پیکره‌بندی می‌شود پایه $\overline{SS}$ همیشه ورودی است. هنگامی که پایه $\overline{SS}$ پایین می‌شود SPI فعال شده و پایه MISO درصورت تعریف کاربر، به عنوان خروجی در نظر گرفته می‌شود. پایه‌های دیگر همه ورودی هستند.

زمانی که پایه SS بالا می‌شود تمام پایه‌ها ورودی هستند و SPI بیکار است. بدین معنی که هیچگونه داده‌ای را دریافت و یا ارسال نمی‌کند. اگر پایه SS در هنگام انتقال داده بالا رود SPI ارسال و دریافت را بلافاصله قطع کرده و اطلاعات چه دریافت و چه ارسال شده باشند از دست می‌روند .

ارتباط SPI و رجیسترهای مربوطه

رجیستر کنترلی SPI – SPCR [SPI CONTROL REGISTER]

Bit	7	6	5	4	3	2	1	0
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

بیت 7– SPIE : فعال کننده وقفه SPI

این بیت و بیت وقفه سراسری (I) اگر یک باشند وقفه SPI فعال می‌شود.

بیت 6 – SPE : فعال‌کننده SPI

زمانی که این بیت یک باشد SPI فعال می‌شود. این بیت برای انجام هرگونه عملیات SPI باید یک باشد.

بیت 5–DORD : DATA ORDER

اگر این بیت یک باشد LSB داده ابتدا فرستاده می‌شود.

اگر این بیت صفر باشد MSB داده ابتدا فرستاده می‌شود.

بیت 4 – MSTR : انتخاب MASTER / SLAVE

یک بودن این بیت مشخص کننده SPI در حالت MASTER و صفر بودن آن مشخص کننده SPI در حالت SLAVE است.

بیت3–CPOL : CLOCK POLARITY

در حالت بیکاری SPI، اگر این بیت یک باشد پایه SCK بالا خواهد بود درغیر اینصورت پایه SCK پایین خواهد بود.

بیت2–CPHA : CLOCK PHASE

مُدهای اطلاعاتی (DATA MODE)

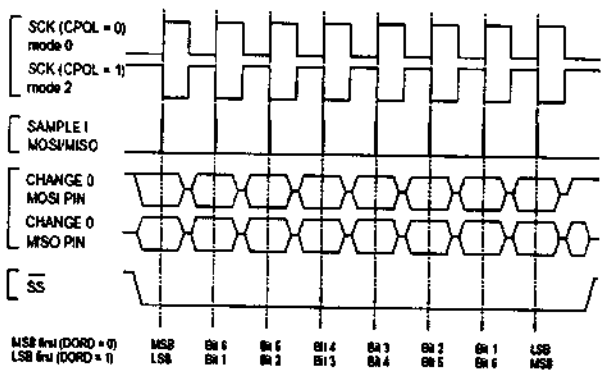
چهار نوع ترکیب PHASE و POLARITY با توجه به داده سریال طبق جدول ۶-۲۴ موجود می‌باشد که توسط بیتهای کنترلی CPHA وCPOL مشخص می‌شوند. مُدهای انتقال داده SPI در شکلها و جدول‌های زیر مشخص شده‌اند. CPOL (CLOCK POLARITY) نقشی در نحوه ارسال داده ندارد و فقط وضعیت پایه SCK را در حالت بیکاری SPI مشخص می‌کند.

نکته SLAVE و MASTER باید در یک مُد پیکره‌بندی شوند.

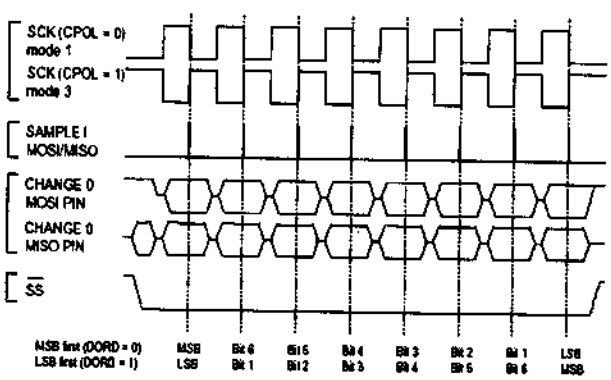
جدول ۶-۲۴ انتخاب مُدهای ارتباطی اطلاعاتی SPI

SPI MODE	CPOL	CPHA	SHIFT SCK EDGE	CAPTURE SCK EDGE
0	0	0	FALLING	RISING
1	0	1	RISING	FALLING
2	1	0	RISING	FALLING
3	1	1	FALLING	RISING

دو نمودارهای زمانی زیر ارتباط MASTER/SLAVE را در مُدهای اطلاعاتی مختلف نشان می‌دهد.



نمودارهای زمانی مُدهای0 و 2 SPI TRANSFER MODE SELECTION WITH CPHA =0



نمودارهای زمانی مُدهای1 و 3 SPI TRANSFER MODE SELECTION WITH CPHA =1

بیت **SPR1 – 0 , 1** و **SPR0** : SPI CLOCK SELECT

این دو بیت فرکانس کلاک SPI (پایه SCK) را برای MASTER تعیین می‌کنند.

SPR0 و SPR1 هیچ تأثیری بر روی SLAVE نخواهند داشت. ارتباط بین SCK و فرکانس کلاک اسیلاتور fosc در جدول ۶-۲۵ آمده است.

جدول ۶-۲۵ انتخاب فرکانس کلاک SPI با توجه به فرکانس OSC

SPI2X	SPR1	SPR0	SCK FREQUENCY
0	0	0	fosc/4
0	0	1	fosc/16
0	1	0	fosc/64
0	1	1	fosc/128
1	0	0	fosc/2
1	0	1	fosc/8
1	1	0	fosc/32
1	1	1	fosc/64

رجیستر وضعیت SPI – SPSR [SPI STATUS REGISTER]

Bit	7	6	5	4	3	2	1	0
	SPIF	WCOL	-	-	-	-	-	SPI2X
Read/Write	R/W	R/W	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

بیت7– SPIF : پرچم وقفه

زمانی که ارسال داده تکمیل شد بیت SPIF یک می‌شود در صورتی که بیت SPIE در رجیستر SPCR و وقفه سراسری (I) فعال شده باشند.

بیت6–WCOL : WRITE COLLISION FLAG

اگر در زمان انتقال داده در رجیستر SPDR نوشته شود بیت WCOL یک می‌شود. در زمان یک بودن بیت WCOL، اولین خواندن از رجیستر وضعیت باعث صفرشدن بیت WCOL و همچنین بیت SPIE می‌شود و سپس دسترسی به رجیستر داده (SPDR) انجام می‌گیرد.

بیت 5...1 : بیت‌های رزرو شده

بیت0 –SPI2X : DOUBLE SPI SPEED BIT

زمانی که منطق یک در این بیت نوشته می‌شود، سرعت کلاک (فرکانس کلاک SPI) در مُد MASTER دو برابر می‌شود. این بدین معنی است که کلاک SPI می‌تواند تا نصف کلاک سیستم افزایش یابد. زمانی که میکرو در مُد SLAVE قرار دارد همچنان بیشترین فرکانس کلاک SPI برابر FOSC/4 است. این بیت در بعضی از میکروهای AVR موجود است.

رجیستر داده SPI – SPDR [SPI DATA REGISTER]

Bit	7	6	5	4	3	2	1	0
	MSB	-	-	-	-	-	-	LSB
Read/Write	R/W	R/W	R	R	R	R	R	R
Initial Value	X	X	X	X	X	X	X	X

X= تعریف نشده

رجیستر داده SPI یک رجیستر خواندنی/نوشتنی است که برای انتقال و یا ارسال داده SPI استفاده

می‌شود. نوشتن بر روی این رجیستر ، داده را به باس SPI ارسال می‌کند و خواندن از این رجیستر داده موجود در بافر دریافتی ، شیفت رجیستر خوانده می‌شود.

پیکره‌بندی SPI در محیط BASCOM

در BASCOM پایه‌های SPI (SERIAL PERIPHERAL INTERFACE) به دو صورت نرم‌افزاری و سخت‌افزاری پیکره‌بندی می‌شوند. زمانی که آن را بصورت سخت‌افزاری پیکره‌بندی می‌کنید پایه‌های پیش فرض ، یعنی پایه‌های (SS (SLAVE SELECT ، (MISO (MASTER IN SLAVE OUT ، (MOSI (MASTER OUT SLAVE IN ، (CLK (CLOCK به کار می‌روند و نمی‌توان آنها را تغییر داد. با استفاده از پیکره‌بندی نرم‌افزاری می‌توان هر کدام از پایه‌های میکرو را به جای پایه‌های فوق استفاده نمود. به همین منظور پیکره‌بندی SPI در دو مُد فوق مجزا توضیح داده شده است.

پیکره‌بندی سخت‌افزاری SPI

Syntax for hardware SPI :

CONFIG SPI = HARD, INTERRUPT= ON|OFF , DATA ORDER = LSB|MSB , MASTER = YES|NO , POLARITY = HIGH|LOW , PHASE = 0|1, CLOCKRATE = 4|16|64|128 , NOSS =0|1

INTERRUPT= ON|OFF : در صورت استفاده از وقفه در ارتباط SPI از گزینه ON استفاده می‌شود.

DATAORDER = LSB/MSB :در صورت انتخاب LSB ، ابتدا LSB وسپس MSB داده ارسال خواهد شد و در صورت انتخاب MSB ابتدا MSB و سپس LSB داده ارسال خواهد شد.

MASTER = YES/NO : اگر میکرویی که در حال برنامه‌نویسی برای آن هستیم MASTER باشد گزینه YES و اگر SLAVE باشد گزینه NO را برمی‌گزینیم.

PHASE = 0/1 : انتخاب صفر توصیه می‌شود.

POLARITY = HIGH/LOW : اگر بخواهیم زمانی که SPI در حالت بیکاری (IDLE) است پایه کلاک بالا باشد ، گزینه HIGH انتخاب می‌شود. انتخاب LOW باعث پائین قرارگرفتن پایه کلاک می‌شود.

CLOCK RATE : مشخص کننده فرکانس کلاک SPI که می‌تواند 1/128 ، 1/64 ، 1/16 ، 1/4 فرکانس سیستم باشد.

NOSS=0|1 : زمانی که در حالت MASTER نمی‌خواهید سیگنال SS ایجاد شود، 1 را انتخاب کنید. در این حالت کاربر بایستی نرم‌افزاری پایه SLAVE موردنظر را LOW کند.

پیکره‌بندی سخت‌افزاری را می‌توان به صورت دستور ساده زیر نیز نوشت:

CONFIG SPI = HARD

که در این حالت بصورت پیش فرض اول MSB فرستاده می‌شود و POLARITY= HIGH، MASTER = YES، PHASE = 0، CLOCKRATE = 4 در نظر گرفته می‌شوند.

پیکره‌بندی نرم‌افزاری SPI

در صورت انتخاب این نوع پیکره‌بندی می‌توان برای هر یک از خطوط ارتباط SPI پایه‌ای به دلخواه انتخاب کرد.

Syntax for software SPI :

CONFIG SPI = SOFT, DIN = PIN, DOUT = PIN , SS = PIN|NONE , CLOCK = PIN

DIN : پایه (MASTER IN SLAVE OUT) MISO است که از پایه دلخواه PIN استفاده می‌شود.

DOUT : پایه (MASTER OUT SLAVE IN) MOSI است که از پایه دلخواه PIN استفاده می‌شود.

پایه‌های $\overline{SS}$ و CLOCK هم با پایه‌های دلخواه پیکره‌بندی می‌شوند. زمانی که نمی‌خواهید سیگنال $\overline{SS}$ داشته باشید از گزینه $\overline{SS}$=NONE استفاده کنید در این حالت کاربر باید توسط نرم‌افزاری SLAVE موردنظر را با صفر کردن پایه $\overline{SS}$ انتخاب و ارتباط SPI را برقرار کند.

• مثال

```
Config Spi = Soft , Din = Pinb.0 , Dout = Portb.1 , Ss = Portb.2 , Clock =
Portb.3
Dim Var As Byte
Spiinit                    'Initialing  SPI state and pins.
Spiout Var , 1             'send 1 byte  to SPI line or bus
```

دستورات مربوط به ارتباط SPI

دستور SPIINIT

توسط این دستور پایه‌های به کار برده شده در ارتباط SPI ، INITIAL می‌شوند. این دستور بعد از پیکره‌بندی SPI بایستی برای قرارگیری پایه‌های استفاده شده در جهت مناسب نوشته شود.

دستور SPIIN

SPIIN var, bytes

توسط این دستور به تعداد bytes از باس SPI ، بایت دریافت می‌شود و در متغییر var قرار می‌گیرد.

• مثال

```
Dim A As Byte
Config Spi = Soft , Din = Pinb.0 , Dout = Portb.1 , Ss = Portb.2 ,Clock_
=Portb.3
Spiinit
Spiin A , 1                'read 1 bytes
End
```

دستور SPIOUT

SPIOUT var , bytes

با این دستور به تعداد bytes ، داده var به باس SPI ارسال خواهد شد.

• مثال

```
Config Spi = Soft , Din = Pind.5 , Dout = Portd.7 , Ss = Portd.2 , Clock=_
Portd.3
Spiinit
Dim A(10) As Byte , X As Byte
```

```
Spiout A(1) , 5                         'send 5 bytes
Spiout X , 1                            'send 1 byte
End
```

دستور SPIMOVE

var = SPIMOVE(byte)

متغییر یا ثابت byte به باس SPI ارسال شده وهمزمان داده دریافت شده از باس SPI در متغییر var جای می‌گیرد.

• مثال

```
Config Spi = Soft , Din = Pinb.0 , Dout = Portb.1 , Ss = Portb.2 , Clock =
Portb.3
Spiinit
Dim A(10) As Byte , X As Byte
Spiout A(1) , 5                                 'send 5 bytes
Spiout X , 1                                    'send 1 byte
A(1) = Spimove(5)       'move 5 to SPI and store received byte in a(1)
End
```

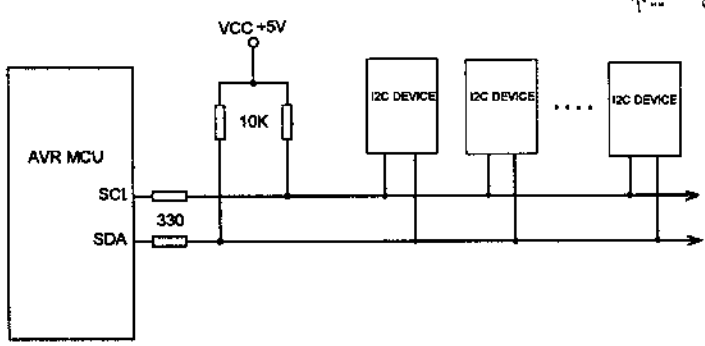
۶-۱۱ پیکره‌بندی ارتباط سریال I2C (2-WIRE)

ارتباط سریال I2C یک پروتکل ارتباطی 2-WIRE است که توسط PHILIPS طراحی شده است. البته برای برقراری ارتباط نیاز به VCC و زمین است پس در واقع 2-WIRE ارتباطی 4 سیمه است که بالاترین فرکانس کلاک آن می‌تواند 400KHZ باشد. شکل ۶-۱۵ طرز اتصال وسایل I2C در یک باس را نشان می‌دهد.

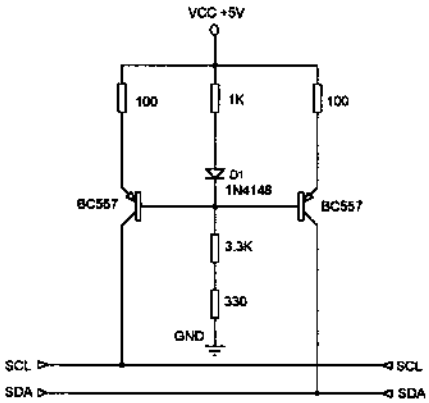
در ساده‌ترین حالت خطوط I2C با مقاومت‌های 10K ، PULL-UP می‌شوند ولی برای برطرف کردن مشکل فاصله I2C مدار زیر پیشنهاد شده است که می‌تواند ارتباط I2C در فرکانس 400KHZ بدون هیچ مشکلی تا 80cm طول داشته باشد. مدار شکل ۶-۱۶ در انتهای خطوط I2C قرار می‌گیرد.

تعیین کلاک I2C

کلاک ارتباط I2C بستگی به کلاک سیستم دارد. از این پیکره‌بندی برای تعیین فرکانس کاری ارتباط I2C استفاده می نماییم.



شکل ۶-۱۵ اتصال وسایل I2C در یک باس



شکل ۶-۱۶ مدار ارتباطی وسایل I2C برای فواصل دور

CONFIG I2CDELAY= X

X مقدار عددی بین 1 تا 255 برای تعیین فرکانس کلاک I2C است که عدد بالاتر، فرکانس کلاک پایین‌تری ایجاد می‌کند. X به صورت پیش‌فرض 5 است و کلاکی برابر 200KHZ را در ارتباط I2C ایجاد می‌کند و یا به طور مثال X= 10 کلاک 100KHZ را تولید می‌کند.

• مثال

```
CONFIG I2CDELAY = 10        '100KHZ CLOCK
```

تعیین پایه SDA

CONFIG SDA = pin

برای ارتباط میکرو با وسایل جانبی دیگر از طریق پروتکل I2C (2-wire) ، با این دستور پایه SDA (DATA) پیکره‌بندی می‌شود.

• مثال

```
CONFIG SDA = PORTC.7          ' PORTC.7 is the SDA line
```

تعیین پایه SCL

CONFIG SCL = pin

برای ارتباط میکرو با وسایل جانبی دیگر از طریق پروتکل I 2C (2-wire) ، با این دستور پایه SCL (CLOCK) پیکره بندی می‌شود.

• مثال

```
CONFIG SCL = PORTC.6          ' PORTC.6 is the SCL line
```

دستور I2C RECEIVE

این دستور از خط یا باس سریال I2C (2 -WIRE) داده دریافت می‌کند .

I2creceive Slave , Var
I2creceive Slave , Var , B2w , B2r

SLAVE : یک متغییر نوع BYTE ، INTEGER ، WORD یا عدد ثابت که حاوی آدرس Slave است .

VAR : یک متغییر نوع BYTE ، INTEGER یا WORD که داده دریافت شده از خط یا باس I2C را در خود جای می‌دهد .

B2W : مشخص کننده تعداد بایت دلخواه برای ارسال به خط یا باس سریال I2C است.

B2R : مشخص کننده تعداد بایت دلخواه برای دریافت از خط یا باس سریال I2C است .

• مثال

```
X = 0 Reset Variable
Slave = &H40                    'Slave Address
I2creceive Slave , X            'Get The Value
Print X                         'Print The Received Byte
```

• مثال

```
Dim Buf(10) As Byte
Buf(1) = 1 . Buf(2) = 2
I2creceive Slave , Buf(1) , 2 , 1       'Send Two Bytes And Receive One Byte
Print Buf(1)                            'Print The Received Byte
```

دستور I2C SEND

این دستور داده را به خط یا باس I2C (WIRF -2) می‌فرستد .

```
I2csend Slave , Var
I2csend Slave , Var , Bytes
```

SLAVE : عدد ثابت یا متغییر نوع BYTE ، INTEGER ، WORD ، که حاوی آدرس Slave است.

VAR : داده‌ای که قصد ارسال به خط یا باس I2C را داریم که می‌تواند یک BYTE ، INTEGER ،WORD یا عدد ثابت باشد.

BYTES : مشخص کننده تعداد بایت دلخواه برای ارسال به خط یا باس سریال I2C است.

• مثال

```
I2csend &H40 , 255
```

دستورات I2CSTART , I2CSTOP , I2CRBYTE , I2CWBYTE

```
I2cstart
I2cstop
I2crbyte Var , Ack / Nack
I2cwbyte Val
```

I2CSTART : که باعث ایجاد شروع (START CONDITION) در پروتکل ارتباطی I2C می‌شود .

I2CSTOP : که باعث ایجاد پایان (STOP CONDITION) در پروتکل ارتباطی I2C می‌شود.

I2CRBYTE : یک بایت از خط یا باس I2C می‌گیرد. VAR متغیری است که داده را از خط یا باس I2C می‌گیرد و ACK را برای دریافت بیش از یک بایت و NACK را زمانی که آخرین بایت را می‌خوانیم ، ایجاد می‌کنیم.

I2CWBYTE : یک بایت به خط یا باس I2C می‌فرستد. VAL متغیر یا ثابتی است که به خط یا باس I2C ارسال می‌شود.

• مثال

```
DIM A AS BYTE
I2cstart                'Generate Start
```

```
I2cwbyte 3              'Send A Value
I2crbyte A , Nack       'Receive Value Into A. Nack Means Last Byte To Receive
Print A
I2cstop                 'Generate Stop
```

۱۲-۶ پیکره‌بندی WATCHDOG

تایمر WATCHDOG از اسیلاتور جداگانه داخلی کلاک دریافت می‌کند. با کنترل کلاک WATCHDOG و منبع تغذیه میکرو بعد از سپری شدن زمان مشخص شده سیستم ری‌ست خواهد شد. 8 زمان مختلف موجود می‌باشد که به وسیله آنها می‌توان زمان ری‌ست را مشخص کرد. بعد از سپری شدن زمان ، میکرو ری‌ست شده و برنامه از بردار ری‌ست اجرا می‌شود.

CONFIG WATCHDOG = TIME

TIME : میکرو پس از سپری شدن TIME ری‌ست شده و اجرای برنامه را از بردار ری‌ست آغاز می‌کند. مقادیر معتبر برای TIME می‌توانند 2048 , 1024 , 512 , 256, 128 , 64 , 32 , 16 میلی ثانیه باشند.

CONFIG WATCHDOG = 2048

این دستور میکرو را پس از گذشت 2048 میلی ثانیه بعد از دستور START WATCHDOG ری‌ست می‌کند.

نکته

با دستور CONFIG WATCHDOG ، WATCHDOG شروع به کار نمی کنند بلکه فقط پیکره‌بندی می‌شود. WATCHDOG با دستور START WATCHDOG شروع به کار کرده و با دستور STOP WATCHDOG متوقف و تایمر آن توسط دستور RESET WATCHDOG ری‌ست می‌شود.

• **مثال**

```
$BAUD=1200
Config Watchdog = 2048              'reset after 2048 mSec
Start Watchdog                      'start the watchdog timer
Dim I As Word
For I = 1 To 1000
Print I                             'print value
'RESET WATCHDOG
```

در صورت استفاده نکردن از RESET WATCHDOG در داخل حلقه ، حلقه FOR کامل اجرا نمی‌شود زیرا تا زمانی که I به مقدار 1000 برسد ، 2048 میلی ثانیه گذشته و میکرو ری‌ست می‌شود. در صورت استفاده کردن از RESET WATCHDOG در داخل حلقه ، حلقه FOR کامل اجرا می‌شود زیرا متناوباً دستور RESET WATCHDOG اجرا شده و تایمر WATCHDOG مدام با عدد $00 ری‌ست می‌شود.

```
Next
End
```

۱۳-۶ وقفه‌ها

در این بخش قصد داریم با انواع وقفه‌های میکروهای AVR در محیط BASCOM آشنا شده و فعال‌سازی و پرش به زیربرنامه وقفه یا همان ISR را فرا بگیریم.

دستورات ENABLE , DISABLE

```
DISABLE interrupt
ENABLE interrupt
```

DISABLE: این دستور برای غیرفعال کردن وقفه استفاده می‌شود.

ENABLE : این دستور برای فعال کردن وقفه استفاده می‌شود.

نکته

برای فعال کردن هر یک از وقفه‌ها علاوه براینکه وقفه مربوطه بایستی توسط دستورات فوق فعال شده باشد، وقفه سراسری نیزباید توسط دستور ENABLE INTERRUPTS فعال شده باشد. این دستور اجازه استفاده از همه وقفه‌ها را می‌دهد.

دستور ON INTERRUPT

زمانی که وقفه اتفاق بیافتد سیستم اجرای برنامه را متوقف کرده و به وقفه پاسخ می‌دهد و به برچسبی که برای آن وقفه تعریف شده پرش می‌کند و بعد از برگشت اجرای برنامه ادامه پیدا می‌کند.

```
ON  interrupt label  [ NOSAVE ]
```

در دستور فوق lable نام برچسبی است که به هنگام وقوع وقفه interrupt برنامه به آن پرش می‌کند و بقیه وقفه‌ها را غیرفعال می‌کند تا زمانی که از برنامه وقفه خارج شود. به کار بردن گزینه اختیاری NO SAVE باعث می‌شود که هیچ کدام از رجیسترها (رجیستر وضعیت ، رجیسترهای R0 تا R11 و R16 تا R31) ذخیره نشوند و درون برنامه وقفه تغییر یابند ولی در صورت استفاده نکردن از این گزینه تمام رجیسترهای استفاده شده برای کل برنامه تغییر نمی‌کند.

لازم به یادآوری است که برای برگشت از وقفه نیاز به RETURN داریم. اگر از چندین RETURN استفاده کنیم، اولین RETURN که داخل شرط یا حلقه نباشد به عنوان RETI (یعنی برگشت از وقفه استفاده می‌شود) و بقیه به عنوان RETURN استفاده می‌شود.

نکته

امکان اینکه شما برای وقفه‌ها اولویت تعیین کنید نیست، و هر وقفه‌ای که در آدرس پایین‌تر حافظه نوشته شده باشد دارای اولویت بالاتر است.

پیکره‌بندی وقفه‌های خارجی(EXTERNAL INTERRUPT)

برای تعیین نحوه تریگ شدن وقفه خارجی از پیکره‌بندی زیر استفاده می‌کنیم.

```
CONFIG INTX = STATE
```

بسته به نوع میکرو X می‌تواند از 0 برای ورودی وقفه خارجی صفر تا 7 برای ورودی وقفه

خارجی هفتم تغییر کند. STATE نیز بسته به نوع میکرو می‌تواند یکی از گزینه‌های زیر باشد :

LOW LEVEL : در این حالت اعمال یک سطح پایین یا صفر به پایه INTX باعث رُخ دادن وقفه برای ورودی وقفه خارجی X می‌شود.

FALLING : در این حالت اعمال یک لبه‌پایین رونده به پایه INTX باعث روی دادن وقفه خارجی می‌شود.

RISING : در این حالت اعمال یک لبه بالارونده به پایه INTX باعث روی دادن وقفه خارجی می‌شود.

جدول۶ -۲۶ انواع وقفه میکروهای AVR و نام آنها در محیط BASCOM

INTERRUPT NAME	EXTERNAL INTERRUPTS	
INT0	External Interrupt 0	وقفه خارجی صفر
INT1	External Interrupt 1	وقفه خارجی یک
INT2 –INT7	External Interrupt for some chips	وقفه‌های خارجی 2 تا 7 برای بعضی از میکروها
	TIMER/COUNTER0 INTERRUPTS	
TIMER0,COUNTER0	Enable TIMER/COUNTER0 interrupts	فعال / غیرفعال کننده تمام وقفه‌های موجود برای T/C0
OVF0	TIMER0 overflow interrupt	وقفه سریزی برای TIMER / COUNTER0
	TIMER/COUNTER1 INTERRUPTS	
TIMER1,COUNTER1	Enable TIMER/COUNTER1 interrupts	فعال / غیرفعال کننده تمام وقفه‌های موجود برای T/C1
OVF1	TIMER1 overflow interrupt	وقفه سریزی TIMER / COUNTER1
CAPTURE1 , ICP1	INPUT CAPTURE TIMER1 interrupt	وقفه ورودی CAPTURE برای T/C1
COMPARE1A,OC1A	OUTPUT COMPARE A interrupt	وقفه تطابق مقایسه مُد A برای T/C1
COMPARE1B,OC1B	OUTPUT interrupt COMPARE B	وقفه تطابق مقایسه مُد B برای T/C1
	TIMER/COUNTER2 INTERRUPTS	
TIMER2,COUNTER2	Enable interrupts TIMER/COUNTER2	فعال/ غیرفعال کننده تمام وقفه‌های موجود برای T/C2
OC2	TIMER2 OUTPUT COMPARE	وقفه تطابق مقایسه‌ای
OVF2	TIMER2 overflow interrupt	وقفه سریزی برای TIMER / COUNTER2
	UART INTERRUPTS	
URXC	Serial RX complete interrupt	وقفه دریافت کامل داده توسط UART
UDRE	Serial data register empty interrupt	وقفه خالی بودن رجیستر داده UART
UTXC	Serial TX complete interrupt	وقفه ارسال کامل داده توسط UART
SERIAL	Disables URXC, UDRE and UTXC	فعال / غیرفعال کننده تمام وقفه‌های UART
	OTHERS	
ACI	Analog comparator interrupt	وقفه مقایسه کننده آنالوگ
SPI	SPI interrupt	وقفه SPI
TWSI	TWI interrupt	وقفه ارتباط سریال TWO-WIRE
ERDY	EEPROM ready interrupt	وقفه آمادگی EEPROM
ADC	A/D converter interrupt	وقفه اتمام تبدیل توسط ADC

• مثال

```
Config INT0 = falling
Enable Interrupts
Enable Int0                    'enable the interrupt
On Int0 Label2 Nosave          'jump to label2 on INT0
Do                             'endless loop
Loop
End

Label2 :
Dim A As Byte
If A > 1 Then
Return                         'generates a RET because it is inside a condition
End If
Return                         'generates a  RETI because it is the first
Return                         'generates a RET because it is the second RETURN
```

۱۴-۶ حافظه EEPROM داخلی میکروهای AVR

بعضی از میکروهای AVR دارای EEPROM داخلی هستند. برای دسترسی به این حافظه می‌توان از دستورات زیر برای خواندن و نوشتن در EEPROM استفاده نمود.

دستور WRITEEEPROM

WRITEEEPROM VAR , ADDRESS

محتوای متغییر VAR در آدرس ADDRESS حافظه EEPROM داخلی نوشته می‌شود. بعد از دستور WRITEEEPROM با توجه به VCC بایستی 4ms - 2.5 تأخیر ایجاد کنید تا عملیات نوشتن تکمیل شود. آدرس می‌تواند یک عدد ثابت یا یک متغییر بسته به حافظه از نوع داده WORD یا BYTE باشد.

- شما همچنین می‌توانید متغییر برای EEPROM تعریف کنید.

Dim var As Eram var type

که var type می‌تواند داده‌های نوع byte , word , integer,long ,single و string باشد.

- شما همچنین می‌توانید متغییرهای آرایه‌ای برای EEPROM استفاده کنید.

Dim ar(10) as Eram Byte

در این حالت 10 بایت اول از حافظه EEPROM برای متغییر آرایه‌ای ()ar در نظر گرفته می‌شود.

- همچنین شما می‌توانید داده خود را در آدرسی دلخواه در EEPROM قرار دهید.

Dim Eb As Eram Byte At 13

محتوای متغییر Eb در آدرس 13 از حافظه EEPROM قرار می‌گیرد.

نکته در صورت مشخص نکردن آدرس حافظه، داده‌ها به ترتیب نوشتن در برنامه، از آدرس0 شروع به جای گرفتن در حافظه می‌کنند.

• مثال

این برنامه محتوای آدرس‌های 0 تا 9 حافظه EEPROM را با عدد 2 و محتوای آدرس‌های 10 و 11 را به ترتیب با عدد 1B و 30 پر می‌کند. (HEX(12315)= 301B)

نکته LSB داده در حافظه پایین‌تر EEPROM قرار می‌گیرد.

```
$regfile = "8535DEF.DAT"
Dim Ar(10) As Eram Byte
Dim B As Byte
Dim C As Eram Word
For B = 0 To 10
Ar(b) = 2
Waitms 4
Next
C = 12315
End
```

• مثال

```
Dim B As Byte, A As Byte
B=12
Writeeeprom B , 1            'store at second position
Waitms 5                    'after 5ms writing eeprom will be completed
```

دستور READEEPROM

READEEPROM VAR , ADDRESS

توسط این دستور محتوای EEPROM از آدرس دلخواه ADDRESS خوانده می‌شود و در متغییر VAR از نوع داده BYTE ذخیره می‌شود. آدرس می‌تواند یک عدد ثابت یا یک متغییر بسته به حافظه از نوع داده WORD یا BYTE باشد.

• مثال

```
Dim B As Byte
Writeeeprom B , 0              'store at first position
Waitms 10
Readeeprom B, 0               'read byte back
```

۱۵-۶ مُدهای SLEEP

معرفی انواع مُدهای SLEEP

مُدهای SLEEP برای متوقف کردن (SHUT DOWN) قسمت‌ها و امکانات استفاده نشده میکرو و همچنین کاهش و صرفه‌جویی در توان مصرفی به کاربرده می‌شوند. AVRها مُدهای مختلفی از SLEEP را برای استفاده کاربر مهیا ساخته‌اند. بیشترین تعداد مُدهای SLEEP ، 6 حالت می‌باشد. بعلت وجود تمام این حالات در میکرو نمونه ATMEGA32 در این بخش قصد داریم به بررسی انواع مُدها بپردازیم و با کاربرد آنها بیشتر آشنا شویم.

مُد IDLE (بیکاری)

در مُد IDLE کلاک CPU متوقف شده ولی میکرو به SPI ، USART ، ANALOG COMPARATOR ، ADC ، TIMER/COUNTERS ، WATCHDOG ، ارتباط سریال TWO-WIRE و وقفه‌های سیستم اجازه می‌دهد که کار کنند (اگر فعال شده باشند). به طور مثال اگر تریگر به پایه اینتراپت فعال شده خارجی اعمال شود و یا وقفه سرریزی یکی از تایمرها روی دهد ، میکرو (MCU) از حالت بیکاری خارج می‌شود و ISR متعلق به وقفه را اجرا می‌کند.

مُد ADC NOISE REDUCTION

در مُد IDLE کلاک CPU متوقف شده ولی میکرو به ADC، TIMER/COUNTER2، WATCHDOG ،

ارتباط سریال TWO-WIRE و وقفه‌های سیستم اجازه می‌دهد که کار کنند (اگر فعال شده باشند). به طور مثال اگر تریگر به پایه اینتراپت فعال شده خارجی اعمال شود و یا وقفه سرریزی تایمر دو روی دهد ، میکرو (MCU) از حالت بیکاری خارج می‌شود و ISR متعلق به وقفه را اجرا می‌کند. این مُد اصولاً کلاک‌های FLASH ، I/O و CPU را به حالت مکث (HALT) می‌برد ولی به کلاک‌های دیگر اجازه می‌دهد که کار کنند.

این مُد بیشتر برای کاهش نویز سیستم در زمان نمونه برداری ADC طراحی شده است که باعث تبدیل با وضوح بیشتر ADC می‌شود. زمانی که ADC فعال شده باشد ،وارد شدن به این مُد باعث شروع نمونه برداری ADC از سیگنال آنالوگ می شود. علاوه بر وقفه اتمام تبدیل ADC ، فقط ری‌ست خارجی ، ری‌ست WATCHDOG ، ری‌ست BROWN – OUT ، وقفه دریافت آدرس صحیح ارتباط سریال TWO-WIRE (TWI) ، وقفه تایمر دو ، وقفه آمادگی EEPROM ، وقفه خارجی حساس به سطحINT0 ، INT1 و وقفه خارجی INT2 می‌توانند میکرو را از مُد ADC NOISE REDUCTION بیدار کنند.

مُد POWER-DOWN

در این مُد ، اسیلاتور خارجی متوقف می‌شود ولی وقفه‌های خارجی، WATCHDOG و دریافت آدرس صحیح ارتباط سریال TWO-WIRE (TWI)، به عملیاتشان (اگر فعال شده باشد) ادامه می‌دهند. فقط ری‌ست خارجی ، ری‌ست WATCHDOG ، ری‌ست BROWN – OUT ، وقفه دریافت آدرس صحیح ارتباط سریال TWO-WIRE (TWI) ، وقفه خارجی حساس به سطح INT0 وINT1 و وقفه خارجی INT2 می‌توانند میکرو را از مُد POWER– DOWN بیدار کنند. تأخیری که طول می‌کشد میکرو از مُد POWER-DOWN بیدار شود توسط فیوز بیت CKSEL قابل تنظیم است.

مُد POWER- SAVE

این مُد با مد Power Down یکسان است فقط با این تفاوت که در این مُد تایمر/کانتر دو می‌تواند با یک شدن بیت AS2 در رجیستر ASSR بصورت غیرهمزمان (ASYNCHRON) کار کند. زمانی که از تایمر دو در مُد غیرهمزمان استفاده نمی‌شود، مُد POWER-DOWN به مُد POWER-SAVE ترجیح داده می‌شود.

مُد STANDBY

این مُد تنها در حالتی که میکرو با کریستال یا نوسانگر خارجی کار می‌کند معتبر و در دسترس است. مُد STANDBY با مُد POWER-DOWN یکسان است با این تفاوت که در این مُد کریستال خارجی قطع نمی‌شود. میکرو پس از گذشت 6 کلاک سیکل از این مُد بیدار می‌شود.

مُد EXTENDED –STANDBY

این مُد نیز تنها در حالتی که میکرو با کریستال یا نوسانگر خارجی کار می‌کند معتبر و در دسترس است. مُد STANDBY با مُد POWER-SAVE یکسان است با این تفاوت که در این مُد کریستال

خارجی قطع نمی‌شود. میکرو پس از گذشت 6 کلاک سیکل از این مُد بیدار می‌شود.

دستورات اجرای مُدهای SLEEP در BASCOM

دستور IDLE

IDLE

توسط این دستور میکرو وارد مُد IDLE می‌شود.

• مثال

میکرو در ابتدا با ارسال کلمه start به پورت سریال به حالت IDLE می‌رود، پس از سپری شدن 16.777216 sec و سرریزی تایمر یک ، از این مُد بیدار شده و ISR سرریزی تایمر با نام Timer1_isr را اجرا می‌کند. بعد از برگشت از ISR میکرو مجدد به حالت IDLE می‌رود. این روند تکرار می‌شود.

```
$regfile = "8515def.dat
$baud = 9600
$crystal = 4000000
Print "start"
Config Timer1 = Timer , Prescale = 1024
'at 4 MHz it gives an overflow at  (1024*65536)/4000000= 16 777216 sec
Enable Timer1
Enable Interrupts
On Timer1 Timer1_isr
Do
   Idle
Loop
End

Timer1_isr:
    Print "in isr"
Return
```

دستور POWERDOWN

POWERDOWN

توسط این دستور میکرو وارد مُد POWERDOWN می‌شود.

• مثال

میکرو در ابتدا با ارسال کلمات start و power down به پورت سریال به حالت power-down می‌رود. در صورت اعمال یک پالس بالا رونده به پایه INT0 میکرو از این مُد بیدار شده و ISR وقفه خارجی صفر را اجرا می‌کند، بعد از برگشت از ISR کلمه return from power down به پورت سریال ارسال و مجدد میکرو به حالت power-down می‌رود. این روند تکرار می‌شود.

```
$regfile = "8515def.dat
$baud = 9600
$crystal = 4000000
Print "start"
Config Int0 = Rising
Enable Int0
Enable Interrupts
On Int0 Int0_isr
Print "power down"
Do
Powerdown
  Print "return from power down"
```

```
Loop
End

Int0_isr:
  Print "in isr"
Return

End
```

دستور POWERSAVE

POWERSAVE

توسط این دستور میکرو وارد مُد POWERSAVE می‌شود. این مُد بیشتر زمانی که تایمر دو در مُد آسنکرون کار می‌کند استفاده می‌شود. در این مُد تایمر دو با کلاک ایجاد شده توسط کریستال ساعت در دو پایه TOSC1 و TOSC2 کار می‌کند.

۶-۱۶ کار با حافظه BOOT (BOOT LOADER FLASH SECTION)

حافظه FLASH میکروکنترلرهای AVR که دارای BOOTLOADER هستند از دو بخش اصلی با نامهای APPLICATION و BOOTLOADER تشکیل شده است. برنامه کاربردی در قسمت حافظه APPLICATION ذخیره می شود. اندازه حافظه BOOTLOADER در این میکروها با دو فیوز بیت BOOTSZ1,2 قابل انتخاب است.

نکته	در قسمت حافظه BOOT فقط قادر به نوشتن برنامه به زبان ASSEMBLY هستید.

وارد شدن به برنامه BOOTLOADER

از فیوز بیت BOOTRST می توان برای انتخاب آدرس بردار ری ست استفاده نمود. در صورتی که این بیت برنامه ریزی نشده (1) باشد آدرس بردار ری ست $0000 است. در صورتی که برنامه ریزی شود بردار ری ست به آدرسی که فیوز بیت های BOOTSZ تعیین کرده اند تغییر می یابد.(جدول ۶-۲۷)

جدول ۶-۲۷ انتخاب آدرس بردار ری‌ست توسط فیوز بیت BOOTRST

BOOTRST	RESET ADDRESS
1(UNPROGRAMMED)	RESET VECTOR = APPLICATION RESET (ADDRESS $0000)
0(PROGRAMMED)	RESET VECTOR = BOOT LOADER RESET

نوع دیگر وارد شدن به حافظه BOOT می تواند توسط دستورات CALL یا JMP در قسمت حافظه APPLICATION انجام گیرد.

کار با BOOTLOADER در محیط BASCOM

در محیط BASCOM می توان با دستور JMP $BOOTADDRESS در هر جای برنامه APPLICATION به برنامه BOOTLOADER پرش کرد. در صورتی که فیوز بیت BOOTRST برنامه ریزی شود برنامه

BOOTLOADER پس از ری ست اجرا می شود. لازم به ذکر است که توسط دستور JMP $0000 در داخل برنامه BOOTLOADER می توان به آدرس $0000 حافظه FLASH پرش کرد.

• مثال

```
$regfile = "M8DEF.DAT"
$crystal = 8000000
$baud = 9600
Dim A As Byte
Do
For A = 0 To 10
Print A
Next
JMP $F80
Loop
End                    'end program
```

```
$boot = $f80
Main:
SBIC PINB,1
RCALL Main
JMP $0000
```

۱. در مثال بالا در صورتی که فیوز بیتهای BOOTRST=0 و BOOTSZ0,1=11 برنامه ریزی شده باشند، اجرای برنامه پس از ری ست به آدرس $F80 رفته و پایه PINB.1 امتحان می شود و در صورتی که پایه بالا باشد دوباره حلقه MAIN اجرا می شود در غیر اینصورت یعنی زمانی که پایه زمین باشد اجرای برنامه از آدرس ابتدای حافظه FLASH ($0000) دوباره آغاز خواهد شد. در برنامه APPLICATION نیز دوباره پس از ارسال چند بایت به پورت سریال اجرای برنامه دوباره از برنامه BOOTLOADER ($F80) ادامه پیدا می کند.

۲. در مثال بالا در صورتی که فیوز بیتهای BOOTRST=1 و BOOTSZ0,1=11 برنامه ریزی شده باشند، اجرای برنامه پس از ری ست به آدرس $0000 رفته، در برنامه APPLICATION پس از ارسال چند بایت به پورت سریال اجرای برنامه به برنامه BOOTLOADER ($F80) پرش می کند و پایه PINB.1 امتحان می شود و در صورتی که پایه بالا باشد دوباره حلقه MAIN اجرا می شود در غیر اینصورت یعنی زمانی که پایه زمین باشد اجرای برنامه از آدرس ابتدای حافظه FLASH ($0000) دوباره آغاز خواهد شد.

آدرس شروع حافظهBOOT طبق جدول ۶-۲۸ و با توجه به نحوه برنامه ریزی فیوزهای BOOTSZ1,2 در نظر گرفته می شود.

جدول ۶-۲۸ انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1 برای ATMEGA8

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	4	0x000 - 0xF7F	0xF80 - 0xFFF	0xF80
1	0	256 words	8	0x000 - 0xEFF	0xF00 - 0xFFF	0xF00
0	1	512 words	16	0x000 - 0xDFF	0xE00 - 0xFFF	0xE00
0	0	1024 words	32	0x000 - 0xBFF	0xC00 - 0xFFF	0xC00

۱۷-۶ دستورات و پیکره‌بندی‌های جانبی

دستور DEBOUNCE

توسط این دستور می‌توان کلیدی را به پایه‌ای از میکرو متصل کرد.

DEBOUNCE Px.y , state , label [, SUB]

PX.Y پایه‌ای مانند PINB.1 است که کلید به آن متصل است. State می‌تواند 0 برای پرش زمانی که پایه یک است و 1 برای پرش زمانی که پایه صفر باشد. Lable برچسبی است که در زمان روی دادن حالت تعیین شده توسط state به آن پرش می‌شود. گزینه اختیاریSUB را برای زمانی که می‌خواهید به جای پرش به برچسب lable به یک زیربرنامه با نام برچسب lable پرش کنید استفاده می‌شود.

دستور بدین صورت کار می‌کند که ابتدا پایه تعریف شده خوانده می‌شود و اگر شرط STATE را داشته باشد ، به مدت 25ms (پیش‌فرض) تأخیر ایجاد می‌شود و دوباره پایه خوانده می‌شود و اگر هنوز وضعیت پایه تغییر نکرده باشد آن را به منزله فشرده شدن کلید در نظر می‌گیرد و به lable پرش می‌کند. زمانی که دستور DEBOUNCE بخواهد دوباره اجرا شود بایستی وضعیت کلید به حالت قبل تغییر کرده باشد.

نکته دستور DEBOUNCE منتظر می‌ماند که حالت state اتفاق بیفتد.

این پیکره‌بندی تأخیری که در هنگام استفاده از دستور DEBOUNCE ایجاد می‌شود را مشخص می‌کند.

CONFIG DEBOUNCE = VAR

VAR مدت زمانی تأخیر بر حسب میلی‌ثانیه است. زمانی که DEBOUNCE پیکره‌بندی نمی‌شود مقدار ثابت 25ms به صورت پیش‌فرض در نظر گرفته می‌شود.

• مثال

```
Config Debounce = 30          'when the config statement is not used a
'default of 25mS will be used
Do
  Debounce Pind.0 , 0 , Pr , Sub
                      ^-------label to branch to
                   ^--------------Branch when P1.0 goes low(0)
              ^-----------------Examine PD.0
'When Pind.0 goes low jump to subroutine Pr
'Pind.0 must go high again before it jumps again
'to the label Pr when Pind.0 is low
Loop
End

Pr:
  Print "PIND.0 was/is low"
Return
```

دستور PULSEOUT

توسط این دستور می‌توان یک پالس با توجه به کلاک سیستم بر روی پایه‌ای دلخواه ایجاد کرد که

مدت زمان این پالس برای فرکانس 4MHZ برابر 1us است.

PULSEOUT PORT , PIN , PERIOD

PORT نام پورت دلخواه و PIN متغییر یا ثابتی که نشان دهنده ، شماره یکی از پایه‌های پورت است به طور مثال PORT=PORTA و PIN=0 به معنای PORTA.0 است. PERIOD متغییر یا ثابتی که مشخص کننده تعداد واحد زمانی است که پایه در منطق یک یا صفر می‌ماند. واحد زمانی PERIOD هنگامی که از کریستال 4MHZ استفاده می‌شود بر حسب us است. در صورت استفاده از کریستال با فرکانس‌های دیگر این مقدار تغییر می‌کند. برای ایجاد پالس‌های دقیق از تایمرها استفاده نمایید زیرا این دستور دقت زیادی ندارد.

این دستور با معکوس کردن وضعیت پایه کار می‌کند در نتیجه وضعیت اولیه در حالت پالس خروجی نقش دارد.

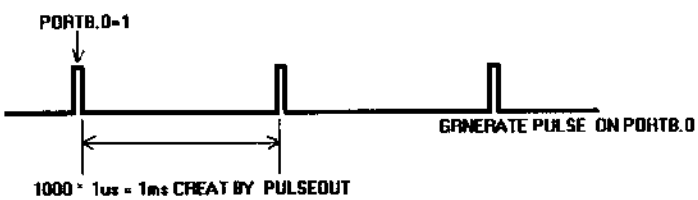
نکته

پایه استفاده شده بایستی خروجی تعریف شده باشد.

• مثال

در صورتی که دستورPulseout در حلقه قرار نگیرد فقط یک پالس ایجاد می‌شود. شکل ۶-۱۷ پالس خروجی پایه portb.0 را نشان میدهد.

```
$crystal = 4000000
Config Portb = Output                    'PORTB all output pins
Portb = 1                               'PORTB.0=1
Do
   Pulseout Portb , 0 , 1000             'generate a pulse
Loop                                     'loop for ever
End
```



شکل ۶-۱۷ پالس ایجاد شده توسط دستور PULSEOUT

دستور PULSEIN

توسط این دستور می‌توان مدت زمان بین تغییر وضعیت پایه دلخواه را از منطق 1 به 0 و یا بلعکس آشکار کرد.

PULSEIN var , PINX , PIN , STATE

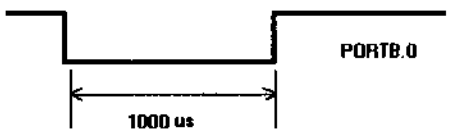
Var متغییری از نوع داده WORD است که مدت زمان مذکور را در خود جای می‌دهد. PINX و PIN نیز مشخص کننده پایه موردنظر برای امتحان کردن هستند. به طور مثال PINX=PIND و PIN=1 به

معنای امتحان شدن PIND.1 است. STATE می‌تواند 0 یا 1 باشد. 0 به معنای تغییر وضعیت پایه از سطح منطقی 0 به 1 است و 1 به معنای تغییر وضعیت پایه از سطح منطقی 1 به 0 است.

این دستور از هیچ یک از تایمرها استفاده نمی‌کند ولی یک کانتر 16 بیتی به کار گرفته می‌شود و هر 10us یک واحد افزایش می‌یابد که این مقدار بستگی به کریستال دارد و در نتیجه بیشترین مدت زمان بین تغییر وضعیت پایه می‌تواند 65535us=65.535ms باشد. در صورتی که در عرض 65.535 میلی‌ثانیه وضعیت پایه تغییر نکند اجرای برنامه بعد از دستور Pulsein ادامه پیدا می‌کند و متغییر خطا با نام ERR یک می‌شود. شما می‌توانید با تست کردن این متغییر از ایجاد خطا در اندازه‌گیری زمان استفاده کنید.

• مثال

```
Dim W As Long
Baud = 9600
Config PortB = Input
Pulsein W , PinB , 0 , 0                    'detect time from 0 to 1
Print W                                   ' 100 WILL PRINT   1000us = 100*10us
Loop
End
```



شکل ۶-۱۸ پالس وارد شده به پایه PORTB.0

دستور SOUND

توسط این دستور می‌توان پالس‌هایی را به پایه دلخواه ارسال کرد.

SOUND pin, duration, pulses

PIN نام یکی از I/O ها مانند PORTA.0 می‌تواند باشد. Duration ثابت یا متغییری است که مشخص کننده تعداد پالس ارسالی است و ثابت یا متغییر pulses نمایانگر مدت زمان بالا و پایین بودن پایه موردنظر است. از این دستور بیشتر برای ایجاد صدا ، زمانی که بلندگو یا BUZZER به یکی از پایه ها متصل است استفاده می‌شود.

این دستور برای ایجاد فرکانس‌های دقیق توصیه نمی‌شود. برای ایجاد فرکانس‌های دقیق از تایمرها استفاده نمایید.

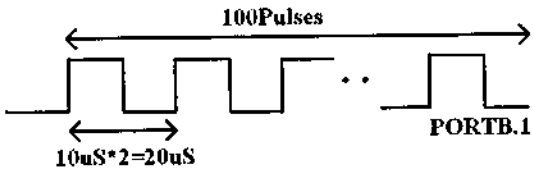
نکته

duration و pulses بایستی بین اعداد 1 تا 65535 باشند.

• مثال

```
$CRYSTAL = 8000000
CONFIG PORTB=OUTPUT
```

```
SOUND PORTB.1 , 100, 10          'LIKE   BEEP SOUND
End
```



شکل ۶-۱۹ پالس ایجاد شده توسط دستور SOUND

۷

حافظه‌های EEPROM سریال 2-WIRE

۷-۱ معرفی انواع حافظه‌های سری..AT24

زمانی که نیازی به ارسال داده با سرعت‌های بالا نباشد از حافظه‌های سریال که حجم کمی نسبت به حافظه‌های موازی اشغال می‌کنند، استفاده می‌شود . تعداد پایه‌های کم حافظه سریال نسبت به حافظه موازی باعث می‌شود که پایه‌های بیشتری از میکرو آزاد بماند .این خصوصیت زمانی ارزش پیدا می‌کند که شما با میکروهایی که پایه کمی دارند کار می‌کنید.

شرکت ATMEL رنج وسیعی از حافظه‌های EEPROM سریال باقابلیت ارتباط بصورت (2- WIRE) I2C را از سری AT24 به بازار عرضه کرده است. انواع آنها در جدول ۷-۱ آمده است.

جدول ۷-۱ انواع حافظه‌های سریال 2- WIRE

DEVICE	SIZE (BITS)	PAGE SIZE (BYTE)	MAX PER BUS	ADDRESSES USED
AT24C01	1K	4	1	NONE
AT24C21	1K	8	1	NONE
AT24C01A	1K	8	8	A0, A1, A2
AT24C02	2K	8	8	A0, A1, A2
AT24C04	4K	16	4	A1,A2
AT24C08	8K	16	2	A2
AT24C16	16K	16	1	NONE

AT24C164	16K	16	8	A0, A1, A2
AT24C32	32K	32	8	A0, A1, A2
AT24C64	64K	32	8	A0, A1, A2
AT24C128	128K	64	4	A0, A1
AT24C256	256K	64	4	A0, A1
AT24C512	512K	128	4	A0, A1
AT24C1024	1M	256	2	A1

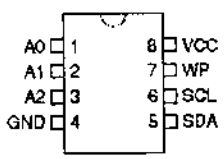
(ادامه جدول صفحه قبل)

معرفی نوع های AT2402A/04A/08A

در این بخش قصد داریم به طور نمونه انواع حافظه‌های AT24C02A/04A/08A را انتخاب کرده و در مورد ترکیب پایه‌ها (شکل ۷-۱) ، مشخصات و چگونگی ارتباط با این نوع از حافظه‌ها آشنا شویم . در صورت یادگیری این بخش شما می‌توانید به راحتی با انواع دیگر کار و ارتباط برقرار کنید.

خصوصیات

- مصرف کم و ولتاژهای کاری استاندارد
  (VCC = 4.5V to 5.5V) 5.0
  (VCC = 2.7V to 5.5V) 2.7
  (VCC = 2.5V to 5.5V) 2.5
  (VCC = 1.8V to 5.5V) 1.8
- سازماندهی حافظه داخلی به طور مثال برای AT24C04 بصورت 8bit *512
- قابلیت ارتباط بصورت 2- WIRE
- دارای اشمیت تریگر و ورودی فیلتر شده برای کاهش نویز
- قابلیت ارتباط دو طرفه
- ارتباط 400KHz در ولتاژ 5 ولت ، 100KHz در ولتاژهای 1.7 – 2.5 – 2.7
- دارای پایه حفاظت از نوشتن (WRITE PROTECT) WP برای استفاده نرم‌افزاری یا سخت‌افزاری
- مُد نوشتاری 8 بایتی
- زمان ماکسیمم 10ms برای نوشتن
- نگهداری داده تا 100 سال
- قابلیت نوشتن تا 1000,000 بار
- نوع بسته‌بندی
- پایه انواع PDIP ، TSSOP و JEDEC SOIC
- ترکیب بسته بندی



شکل ۷-۱ ترکیب پایه های حافظه‌های AT24C02A/04A/08A در انواع PDIP ، TSSOP و JEDEC SOIC

جدول ۷-۲ عملکرد پایه های حافظه سریال

Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect

AT24C02A/04A/08A دارای مقدار 8192/4096/2048 بیت حافظه به صورت 1024 /512/256 کلمه 8 بیتی هستند. این وسایل برای کار در محیط‌های تجاری و صنعتی و در مکانهایی که استفاده از ولتاژ و توانهای کم احتیاج باشد بهینه‌سازی شده‌اند. محدوده‌های کاری حافظه‌ها، در جدول ۷-۳ آمده است.

جدول ۷-۳ محدوده‌های کاری حافظه‌ها

Operating temperature	-55°C to +125°C
Storage temperature	65°C to +150°C
Voltage on any pin With respect to ground	-1.0v to +7.0v
Maximum oprating voltage	6.25v
Dc output current	5.0mA

۷-۲ معرفی پایه‌ها

پایه کلاک سریال (SCL)

از لبه مثبت SCL برای ورودی داده به هر یک از EEPROM ها و از لبه منفی برای خروج داده از EEPROMها استفاده می شود.

پایه داده سریال (SDA)

SDA یک پایه دو طرفه برای ارتباط داده‌ها است.

پایه‌های A0-A1-A2 آدرس سخت‌افزاری EEPROM

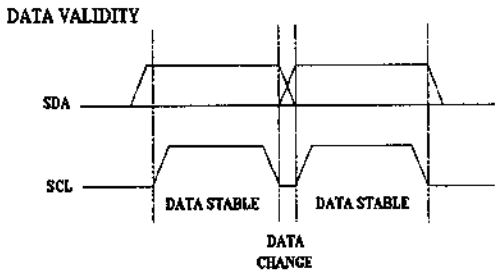
پایه‌های ورودی A0-A1-A2 مشخص کنندهٔ آدرس EEPROM است. در صورتی که به صورت سخت‌افزاری بخواهیم وسیله را آدرس‌دهی کنیم بسته به نوع حافظه تا هشت EEPROM می‌توانند در یک BUS قرار گیرند.

پایه محافظت از نوشتن (WP)

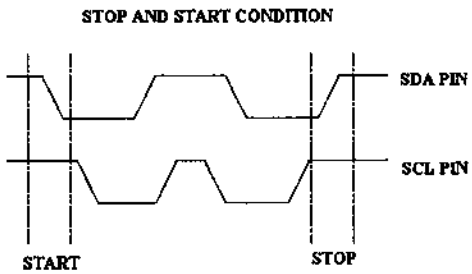
این پایهٔ ورودی زمانی که زمین و یا بدون اتصال باشد نوشتن به صورت عادی انجام می‌گیرد . اگر این پایه یک شود از نوشتن در EEPROM جلوگیری می شود.

۳-۷ طرز کار حافظه

انتقال کلاک و داده : پایهٔ SDA به طور معمول توسط وسیله خارجی بالا نگه داشته می شود. داده بر روی SDA زمانی که SCL پایین است تغییر می‌یابد (با توجه به شکل ۷-۲). تغییر SDA در زمان بالا بودن SCL به منزلهٔ ایجاد یک START CONDITION یا STOP CONDITION است.



شکل ۷-۲ انتقال داده



شکل ۷-۳ دیاگرام زمانی ایجاد شرایط START و STOP

ایجاد START CONDITION : یک به صفر (HIGH TO LOW) کردن SDA در زمان بالا بودن SCL به منزله ایجاد یکSTART CONDITION است(با توجه به شکل۷-۳).

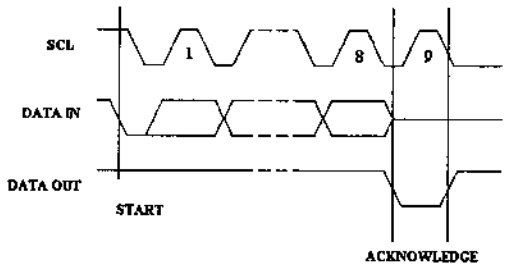
ایجاد STOP CONDITION : صفر به یک (LOW TO HIGH) کردن SDA در زمان بالا بودن SCL به منزلهٔ ایجاد یک STOP CONDITION است(با توجه به شکل۷-۳).

شناسایی (ACKNOWLAGE): تمام آدرس‌ها و داده‌ها در قالب WORD به صورت هشت بیتی از EEPROM به خارج یا از خارج به EEPROM انتقال می‌یابد. EEPROM با فرستادن صفر در کلاک نهم مشخص می کند که هر یک از کلمات (WORD) را طبق دیاگرام زمانی شکل ۷-۴ دریافت کرده است.

RESET شدن حافظه: برای ری‌ست شدن حافظه مراحل زیر بایستی طی شود:

۱. بالا بودن کلاک برای 9 کلاک سیکل
۲. بالا بودن SDA در هر یک از سیکل‌ها هنگامی که SCL یک است.

۳. ایجاد START CONDITON در زمان بالا بودن SDA



شکل ۷-۴ دیاگرام زمانی ارسال ACK توسط EEPROM

۷-۴ آدرس‌دهی سخت‌افزاری حافظه

منظور از آدرس‌دهی سخت‌افزاری، آدرسی که توسط پایه‌های A0 ، A1 و A2 برای انتخاب چیپ دلخواه در باس ایجاد می‌شود، است. این نوع از حافظه‌ها نیاز به 8 بیت آدرس سخت‌افزاری دارند.

2402	1	0	1	0	A_2	A_1	A_0	R/W
	MSB							LSB
2404	1	0	1	0	A_2	A_1	P0	R/W
2408	1	0	1	0	A_2	P1	P0	R/W

شکل ۷-۵ بایت آدرس‌دهی سخت‌افزاری برای 2402/04/08

پایه‌های A0/A1/A2 برای آدرس‌دهی نوع AT2402A استفاده می‌شوند پس بنابراین نهایتاً 8 وسیله می‌توانند در یک باس قرار گیرند. بیت اول نیز برای خواندن و نوشتن مورد استفاده می‌شود (شکل ۷-۵). زمانی که بخواهید در حافظه بنویسید بیت اول را صفر و زمانی که بخواهید از حافظه بخوانید بیت اول را یک می‌کنید.

• مثال

آدرس سخت‌افزاری چیپ AT2402A جهت خواندن از این وسیله زمانی که پایه‌های A0 ، A1 و A2 همگی به زمین وصل شده‌اند برابر 161 = &B10100001 است .

آدرس سخت‌افزاری چیپ AT2402A جهت نوشتن در این وسیله زمانی که پایه‌های A0 ، A1 و A2 همگی به یک وصل شده‌اند برابر 174 = &B10101110 است .

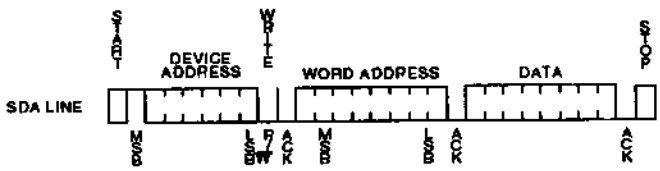
بیت‌های A1/A2 برای آدرس‌دهی نوع AT2404A استفاده می‌شوند. بیت اول نیز برای خواندن و نوشتن مورد استفاده قرار می‌گیرد . زمانی که بخواهید در حافظه بنویسید بیت اول را صفر و زمانی که بخواهید از حافظه بخوانید بیت اول را یک کنید . بیت P0 مشخص کننده آدرس صفحه (PAGE) حافظه

است .در این نوع حافظه پایه‌های A1/A2 استفاده میشود و پایه A0 آزاد می‌باشد. پس بنابراین نهایتاً 4 وسیله می‌توانند در یک باس مشترک استفاده شوند.

بیت A2 برای آدرس‌دهی نوع AT2408A استفاده می‌شود . بیت اول نیز برای خواندن و نوشتن استفاده می‌شود . زمانی که بخواهید در حافظه بنویسید بیت اول را صفر و زمانی که بخواهید از حافظه بخوانید بیت اول را یک کنید . بیت P1/P0 مشخص کننده آدرس صفحه (PAGE) حافظه است. در این نوع حافظه پایه‌های A2 استفاده می‌شود. پایه A0/A1 آزاد می‌باشد پس بنابراین نهایتاً 2 وسیله می‌توانند در یک باس مشترک استفاده شوند.

۵-۷ انواع عملیات نوشتن حافظه

۱. **نوشتن بایتی(BYTE WRITE) :** آدرس‌دهی خانه‌های حافظه AT2402/04/08 به ترتیب توسط 8/9/10 بیت انجام می‌گیرد. عملیات نوشتن بایتی مطابق دیاگرام زمانی شکل۷-۶ انجام می‌شود.



شکل ۷-۶ دیاگرام زمانی نوشتن بایتی در EEPROM سریال

START : ایجاد START CONDITION که می‌توان با دستور I2CSTART آن را در محیط BASCOM ایجاد کرد.

DEVICE ADDRESS :منظور همان آدرس‌دهی سخت‌افزاری است که توسط بایت آدرس‌دهی سخت‌افزاری بدست آمده است و با دستور I2CWBYTE ADDRESS می‌توان آن را به خط یا باس SDA ارسال کرد .توجه کنید که بیت R/W = 0 است.

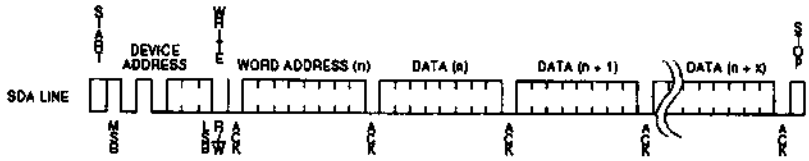
ACK : در تمام موارد مورد نیاز این سیگنال توسط کامپایلر به صورت خودکار ایجاد شده و نیازی به نوشتن برنامه اضافه برای آن نیست.

WORD ADDRESS : آدرس خانه دلخواه حافظه که توسط دستور I2CWBYTE address به خط یا باس SDA ارسال می‌شود.

DATA : داده دلخواه که توسط دستور I2CWBYTE data به خط یا باس SDA ارسال می‌شود تا در خانه حافظه، حفظ شود.

STOP : ایجاد STOP CONDITION که توسط دستور I2CSTOP در محیط BASCOM به خط یا باس SDA ارسال می‌شود.

۲. **نوشتن صفحه‌ای (PAGE WRITE) :** این مُد مانند نوشتن بایتی است با این تفاوت که بعد از نوشتن اولین بایت STOP CONDITION ایجاد نمی‌شود .تعداد بایت نوشتن به صورت صفحه‌ای برای AT2402، 8 بایت و برای AT2404/08 ، 16 بایت است . به طور مثال برای AT2404 بعد از نوشتن اولین بایت در حافظه با ارسال سیگنال ACK می‌توان تا 8 بایت را به همین منوال نوشت . برای درک بیشتربه دیاگرام زمانی شکل ۷-۷ توجه نمایید.

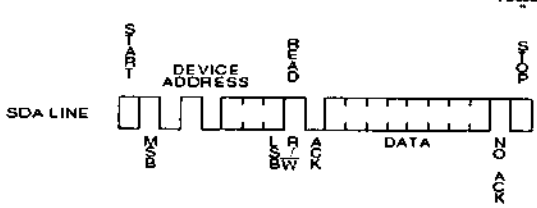


شکل ۷-۷ دیاگرام زمانی نوشتن صفحه ای For AT2404 ، X=7 AND For AT2404/08، X=15

۷-۶ انواع عملیات خواندن حافظه

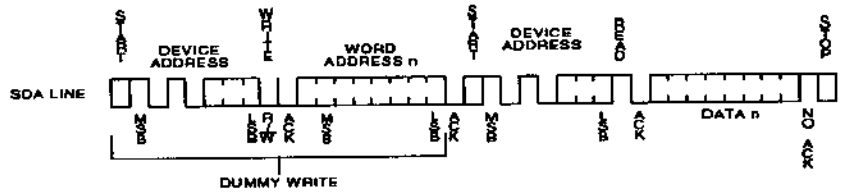
خواندن از حافظه شبیه نوشتن بر روی آن است با این تفاوت که در تنظیم بایت آدرس‌دهی سخت‌افزاری بیت W/R به یک تغییر می‌کند . سه نوع مختلف نوشتن موجود می‌باشد که در زیر آمده است.

۱. **خواندن آدرس اخیر(CURRENT ADDRESS READ) :** یک آدرس بالاتر از آدرس آخرین خانه حافظه چه از آن خوانده شده یا در آن نوشته شده باشد در کانتر داخلی حافظه EEPROM موجود می‌باشد . این آدرس تا زمانی معتبر است که VCC حافظه برقرار باشد . آدرس خانه حافظه در زمان نوشتن در آخرین خانه حافظه صفحه جاری به اولین خانه صفحه جاری بر می‌گردد و در زمان خواندن از آخرین خانه حافظه صفحه آخر به اولین خانه صفحه اول حافظه بر می‌گردد . زمانی که آدرس سخت‌افزاری EEPROM با بیت R/W=1 به چیپ ارسال می‌شود، داده خانه حافظه اخیر توسط EEPROM به بیرون ارسال می‌شود. برای درک بیشتر به دیاگرام زمانی شکل ۷-۸ توجه کنید.



شکل ۷-۸ دیاگرام زمانی خواندن آدرس اخیرحافظه

۲. **خواندن از آدرس دلخواه(RANDOM READ) :** توسط این نوع خواندن می‌توان داده را از آدرس دلخواه خواند . برای درک طرز عملکرد به شکل ۷-۹ و توضیحات توجه نمایید.



شکل ۷-۹ دیاگرام زمانی خواندن از آدرس دلخواه حافظه

START : ایجاد START CONDITION که می‌توان با دستور I2CSTART آن را ایجاد کرد.

DEVICE ADDRESS : منظور همان آدرس‌دهی سخت‌افزاری است که توسط بایت آدرس‌دهی سخت‌افزاری برای نوشتن بدست آمده است که با دستور I2CWBYTE ADDRESS می‌توان آن را به خط یا باس SDA ارسال کرد .توجه کنید که بیت R/W = 0 است.

ACK : در تمام موارد مورد نیاز این سیگنال توسط کامپایلر به صورت خودکار ایجاد شده و نیازی به نوشتن برنامه اضافه برای آن نیست.

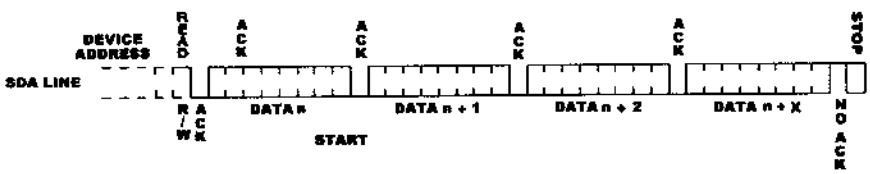
WORD ADDRESS : آدرس خانه دلخواه حافظه که توسط دستور I2CWBYTE ADDRESS به خط یا باس SDA ارسال می‌شود.

DEVICE ADDRESS :منظور همان آدرس‌دهی سخت‌افزاری است که توسط بایت آدرس‌دهی سخت‌افزاری برای خواندن بدست آمده است و با دستور I2CWBYTE ADDRESS می‌توان آن را به خط یا باس SDA ارسال کرد .توجه کنید که بیت R/W = 1 است.

DATA : داده دلخواه از خانه حافظه WORD ADDRESS توسط دستور I2CRBYTE data,nack/ack خوانده شده و در متغییر بایت DATA قرار می‌گیرد.

STOP : ایجاد STOP CONDITION که توسط دستور I2CSTOP به خط یا باس SDA ارسال می‌شود.

۳. خواندن دنباله‌ای (SEQUENTIAL READ) : خواندن دنباله‌ای می‌تواند هم در زمان خواندن آدرس اخیر و هم آدرس دلخواه استفاده شود . زمانی که میکرو داده را دریافت کرد سیگنال ACK را برای EEPROM می‌فرستد و حافظه تا زمانی که سیگنال ACK را دریافت می‌کند، آدرس خانه حافظه را افزایش می‌دهد و داده متناظر با آدرس را به صورت دنباله‌ای ارسال می‌کند . زمانی که آدرس به آخرین خانه حافظه رسید، آدرس به اولین خانه حافظه بر می‌گردد و خواندن دنباله‌ای ادامه پیدا می‌کند. خواندن دنباله‌ای با ایجاد شدن سیگنال NO ACK یا به عبارتی منطق 1 پایان می‌یابد ولی بایستی STOP CONDITION بعد از آن تولید شود .دیاگرام زمانی آن در شکل ۷-۱۰ آمده است.



شکل ۷-۱۰ دیاگرام زمانی خواندن دنباله‌ای از حافظه

۸

پروژه‌های عملی

پروژهای عملی در این فصل برای تسلط و یادگیری بیشتر امکانات میکروکنترلرهای AVR ارائه شده است. برد با نام STARTER KIT (برد آموزشی) به کتاب اضافه شده است. این برد دارای مدارات پروگرامر، LCD ،7SEG سه تایی، کی بورد 4*4 ، RS232 ، سنسور دمای LM335 ، رله تک کنتاکت ، کی بورد کامپیوتر ،خروجی آنالوگ توسط پالس PWM ، کلید و منبع تغذیه 5 و 12 ولت است. میکرو برد کتاب نیز می تواند از انواع ATMEGA32/8535/16/323/163/AT90S8535 باشد. مدارات موجود در این فصل به دو قسمت دسته بندی شده اند.قسمت اول (از بخش ۸-۱ تا ۸-۱۱) مداراتی است که بر روی برد کتاب بسته و تست شده اند و قسمت دوم (از بخش ۸-۱۱ به بعد) مداراهایی که بیرون از برد کتاب انجام شده اند.

اهداف

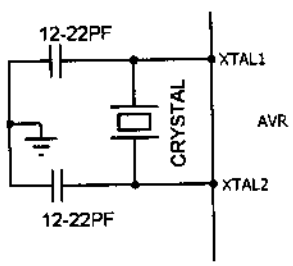
۱. تحلیل کامل برنامه‌های نوشته شده
۲. یادگیری کامل برنامه‌نویسی و کار با تمام امکانات AVR ها
۳. تحلیل کردن برنامه‌های نوشته شده توسط SIMULATOR داخلی BASCOM

۸-۱ مدار و PCB برد کتاب

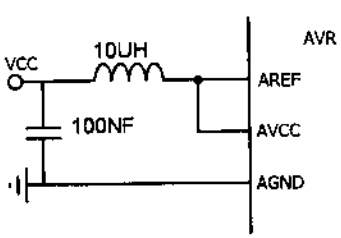
شماتیک و PCB یک رو این برد توسط نرم افزار PROTEL 99SE کشیده شده است و فایل آن در CD موجود است.در کلیه مدارهای این فصل به علت فضای محدود کریستال خارجی ،مدار ری ست و فیلتر قسمت آنالوگ کشیده نشده است.به ترتیب این مدارات طبق شکل های ۸-۱ ،۸-۲ و ۸-۳ است.

شماتیک و PCB برد کتاب در شکل های ۸-۴ ، ۸-۵ آمده است. بر روی برد PCB دو خط به صورت بریده بریده دیده می شود که بایستی توسط سیم این دو نقطه به هم وصل شود(JUMPER) . همچنین برای اینکه ممکن است بخواهید از پایه های میکرو برای مقاصد دیگر در بیرون برد استفاده نمایید در کنار هر پین PAD ای اضافه شده است.همچنین BOM(BILL OF MATERIAL) که شامل نام قطعه، کمیت (در صورت وجود) ،FOOTPRINT و تعداد مورد نیاز قطعه است توسط نرم افزار PROTEL تهیه شده (جدول ۸-۱) که بتوانید قطعات را به راحتی خریداری و مونتاژ نمایید.

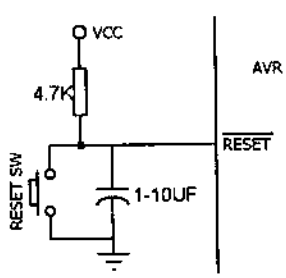
بر روی برد پروگرامر نیز تعبیه شده است. طرز اتصال کانکتور مخابراتی ۷ تایی روی برد به کانکتور نری LPT در شکل ۸-۶ نشان داده شده است. سپس می توانید کانکتور نری LPT را به پورت کامپیوتر خود وصل نمایید.



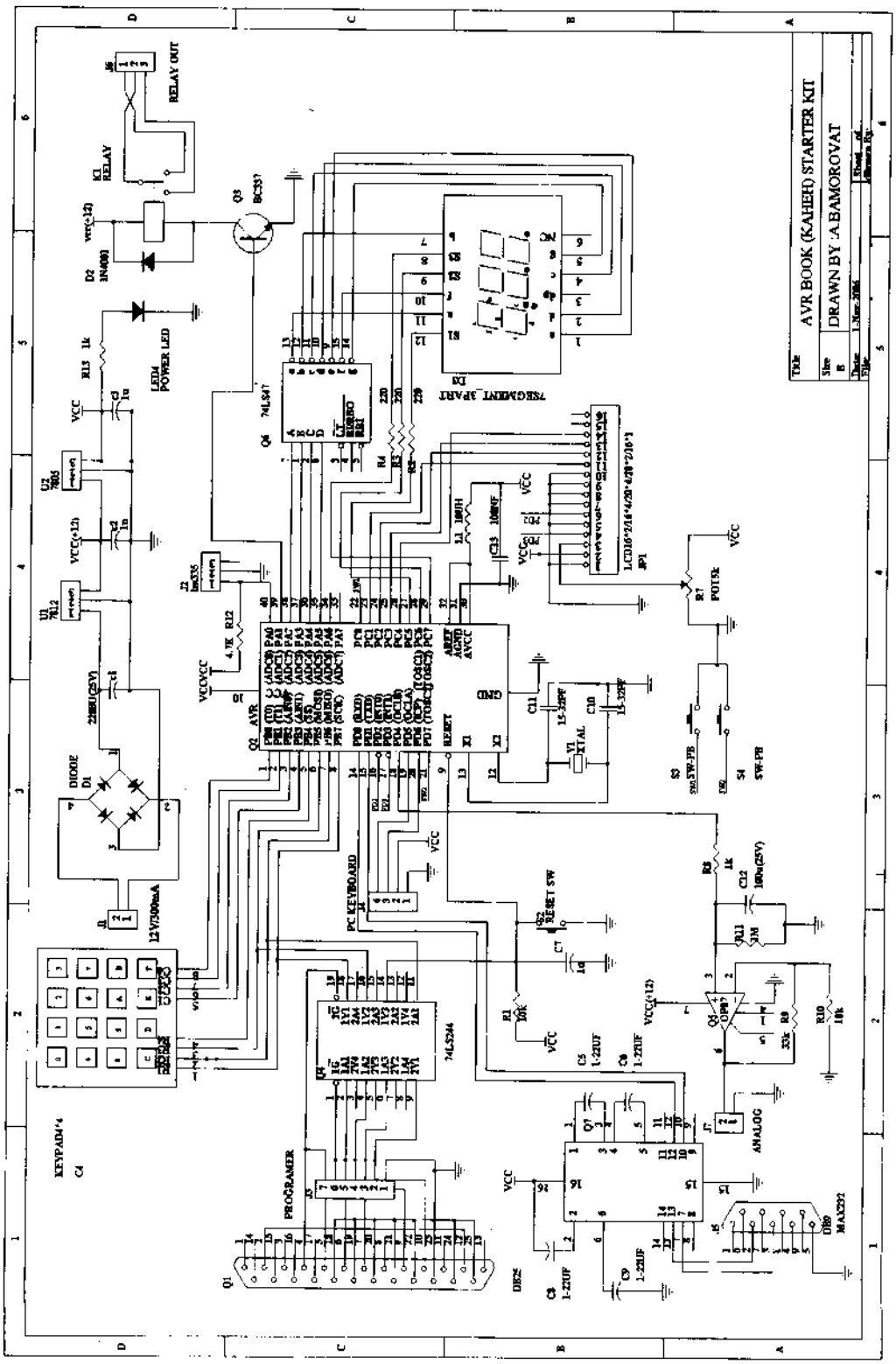
شکل ۸-۱ مدار کریستال خارجی AVR



شکل ۸-۳ مدار فیلتر آنالوگ AVR



شکل ۸-۲ مدار ری ست AVR

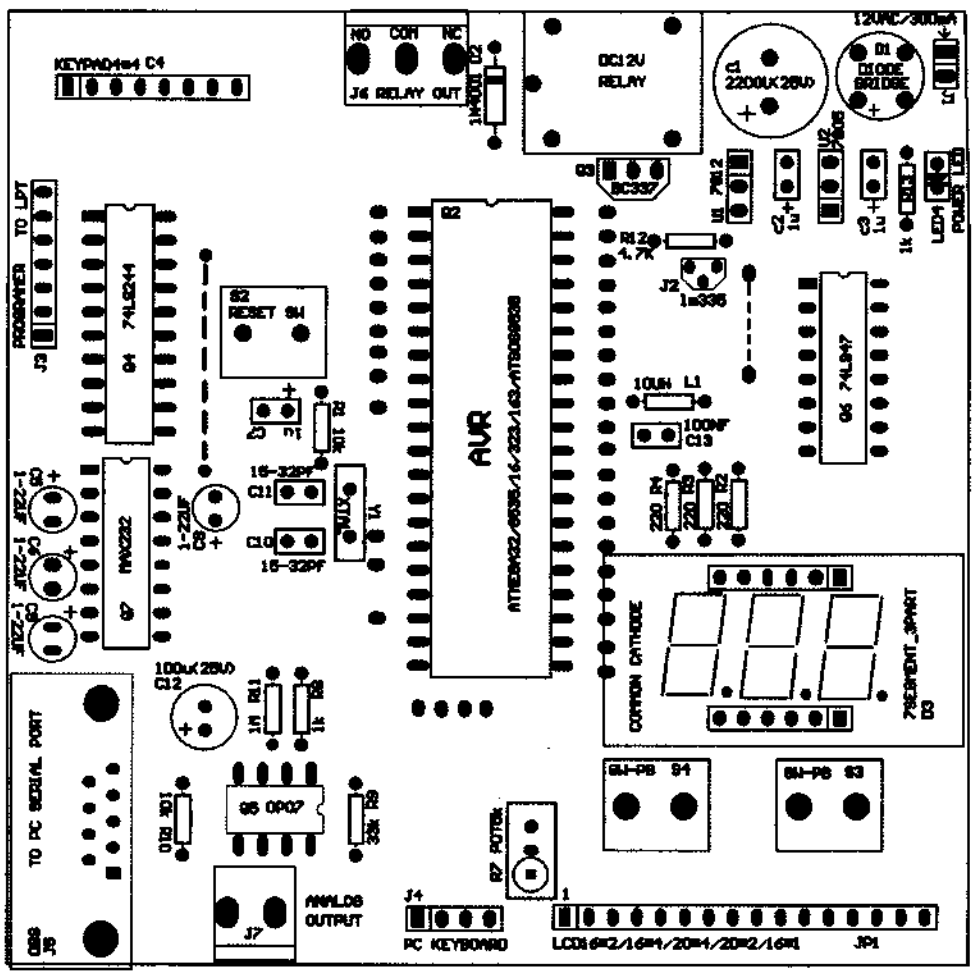

AVR BOOK (KAHEH) STARTER KIT
DRAWN BY :ABAMOROVAT
RELAY OUT
RELAY
POWER LED
DIODE
12V/300mA
KEYPAD4*4
PROGRAMER
PC KEYBOARD
RESET SW
ANALOG
MAX232
74LS244
74LS47
BC337
POT5K
XTAL
SW-PB
LCD16*2/4*40*2/4*20*2
VCC
GND

جدول ۸-۱ لیست قطعات برد کتاب

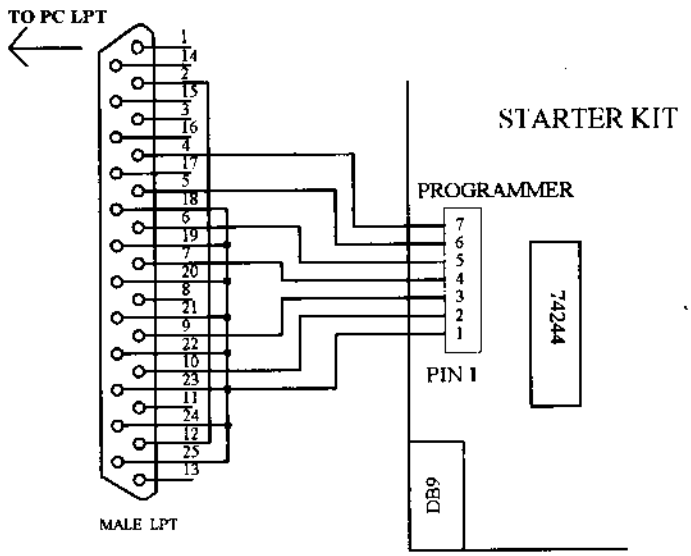
Comment	Pattern	Quantity	Components
1-22UF(16v)	RAD0.1	4	C5, C6, C8, C9
100NF	RAD0.1	1	C13
100u(25V)	RAD0.1	1	C12
10UH	AXIAL0.3	1	L1
10k	AXIAL0.3	2	R1, R10
15-32PF	RAD0.1	2	C10, C11
1M	AXIAL0.3	1	R11
1N4001	AXIAL0.4	1	D2
1k	AXIAL0.3	2	R8, R13
1u(tantalum)	RAD0.1	2	C2, C3
1u(tantalum)	RAD0.1	1	C7
220	AXIAL0.3	3	R2, R3, R4
2200U(25V)	C2200	1	C1
33k	AXIAL0.3	1	R9
4.7K	AXIAL0.3	1	R12
74LS244	DIP20	1	Q4
74LS47	DIP16	1	Q6
7805	SIP3	1	U2
7812	SIP3	1	U1
7SEGMENT_3PART	7SEG	1	D3
BC337	NPN	1	Q3
KEYPAD4*4	SIP8	1	C4
LCD16*2/16*4/20*4/20*2/16*1	SIP16	1	JP1
MAX232	DIP16	1	Q7
OP07	DIP-8	1	Q5
POT5k	POT	1	R7
POWER LED	RAD0.1	1	LED4
RELAY	RELAYAF	1	K1
RELAY OUT	POWER3	1	J6
RESET SW	PB	1	S2
SW-PB	PB	2	S3, S4
XTAL 8MHZ	XTAL-1	1	Y1
lm335	TO-92A	1	J2

ANALOG(کانکتور قدرت آبی رنگ دوتایی)	POWER2	1	J7
AVR(یکی از میکروهای گفته شده)	DIP40	1	Q2
DB9(کانکتور پورت سریال رو بردی)	DB-9RA/F	1	J5
DIODE(پل دیود گرد 1 Amper)	DBC	1	D1
PC KEYBOARD(کی بورد کامپیوتروکانکتور مخابراتی ۴ تایی)	SIP4	1	J4
15VAC/300mA(ترانس دو سر وکانکتور مخابراتی دوتایی)	SIP2	1	J1
PROGRAMER (کانکتور نری وکانکتور مخابراتی ۷ تایی)	SIP-7	1	J3

ادامه جدول صفحه قبل



شکل ۸-۵ شکل PCB مدار کتاب



شکل ۸-۶ اتصال کانکتور LPT نری به کانکتور مخابراتی پروگرامر برد کتاب

۸-۲ اتصال کلید به میکرو توسط دستور DEBOUNCE

این برنامه برای آشنایی بیشتر با دستور DEBOUNCE نوشته شده است. در این برنامه دو کلید یکی به پایه PORTC.0 و دیگری به PORTD.7 طبق شکل ۸-۷ در برد STARTER KIT کتاب متصل شده است. مقاومت PULL-UP پایه های متصل شده به کلیدها فعال شده اند بنابراین در زمان فشردن کلید پایه مربوطه از حالت منطقی یک به صفر تغییر وضعیت میدهد. مدت زمان تاخیر DEBOUNCE در حالت پیش فرض 25ms است که توسط دستور CONFIG DEBOUNCE قابل تغییر است.

در این برنامه از میکرو ATMEGA32 و فرکانس اسیلاتور داخلی 8MHZ استفاده شده است. LCD نوع 2 * 16 است و پایه متصل شده به کلیدها ورودی تعریف شده‌اند. دو زیر تابع INCREMENT و DECREMENT توسط دستور DECLARE تعریف شده‌اند. اجرای برنامه داخل حلقه DO – LOOP انجام شده و ابتدا پایه PINC.0 امتحان می‌شود اگر پایه صفر باشد برای مدت زمان 30ms تاخیر ایجاد می‌شود ، پس از سپری شدن زمان اگر هنوز پایه صفر باشد آن را به منزله فشرده شدن کلید در نظر می‌گیرد و زیر تابع DECREMENT اجرا می‌شود در غیر اینصورت اگر پایه در زمان امتحان شدن یک باشد تاخیری ایجاد نشده و خط بعد اجرا می‌شود . پایه PIND.7 هم به همین منوال امتحان می‌شود.

در زیر برنامه DECREMENT عدد A یک واحد کم می‌شود و در زیر برنامه INCRMENT متغییر یک واحد افزایش می‌یابد و این مقدار مدام در حافظه EEPROM نوشته میشود. در زمان اجرا شدن اولین بار برنامه از ی ست نیز آخرین مقدار A از حافظه خوانده میشود.

- برنامه

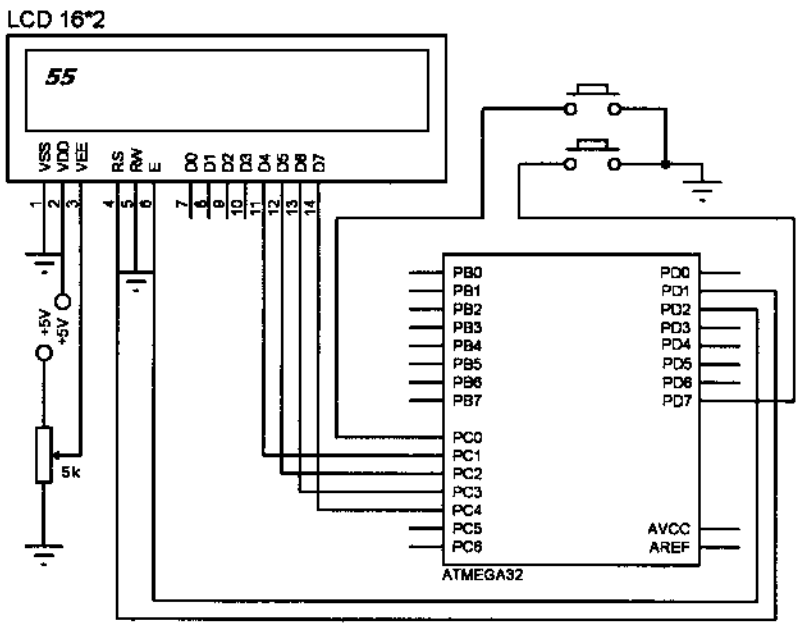
```
$regfile = "M32DEF.DAT"    'mega32 used
$crystal = 8000000         'internal or external crystal
Config Lcdpin = Pin , Db4 = Portc.1 , Db5 = Portc.2 , Db6 = Portc.3 ,Db7 = _
Portc.4 , E = Portd.2 , Rs = Portd.3  'lcd connection
Config Lcd = 16 * 2
Config Debounce = 30       'config debounce delay
Dim A As Word               'dimension a
Dim Lsb_a As Byte           'dimension lsb_A(use to keep lsb of a)
Dim Msb_a As Byte           'dimension lsb_A(use to keep msb of a)
Ddrc.0 = 0 : Portc.0 = 1    'pinc.0 pull up resistor active
Ddrd.7 = 0 : Portd.7 = 1    'pinc.0 pull up resistor active
Declare Sub Increment       'called when increment button pressed
Declare Sub Decrement       'called when decrement button pressed
Readeeprom Lsb_a , 10       'read lsb of a from add 10 of eeprom
Readeeprom Msb_a , 11       'read msb of a from add 11 of eeprom
A = Makeint(lsb_a , Msb_a)  'attching lsb and msb due to make A
Cls                         'lcd cls
Lcd A                       'show A

Do
   Debounce Pind.7 , 0 , Increment , Sub  'Read Push Button repeatedly
   Debounce Pinc.0 , 0 , Decrement , Sub  'Acts When Become Low
Loop
End   'end program


Sub Increment
  Cls
  Incr A
  Lcd A ; "  "
  Home
  Lsb_a = Low(a)           'load lsb of A to variable lsb_a
  Writeeeprom Lsb_a , 10   'write lsb of a to add 10 of eeprom
  Msb_a = High(a)          'load msb of A to variable msb_a
  Writeeeprom Msb_a , 11   'write lsb of a to add 11 of eeprom
  Waitms 4
End Sub


Sub Decrement
  Cls
  Decr A
  Lcd A ; "  "
  Home
  Lsb_a = Low(a)
  Writeeeprom Lsb_a , 10
  Msb_a = High(a)
  Writeeeprom Msb_a , 11
  Waitms 4
End Sub
```

در صورتی که بخواهید از دستورهای شرطی برای خواندن کلیدها استفاده نمایید از برنامه زیر استفاده نمایید.

- برنامه

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Portc.1 , Db5 = Portc.2 , Db6 = Portc.3 , Db7= _
```

```
Portc.4 , E = Portd.2 , Rs = Portd.3
Config Lcd = 16 * 2
Dim A As Byte
Config Pinc.0 = Input
Config Pind.7 = Input
Ddrc.0 = 0 : Portc.0 = 1
Ddrd.7 = 0 : Portd.7 = 1
Cls
Do
 If Pind.7 = 0 Then Incr A
 If Pinc.0 = 0 Then Decr A

 Lcd A ; "  "
 Home
 Waitms 250
Loop

End
```

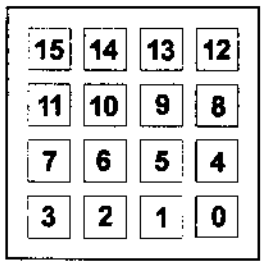


شکل ۸-۷ مدار بسته شده برای برنامه اتصال کلید به میکرو در برد STARTER KIT

۸-۳ اسکن صفحه‌کلید ۴×۴

برنامه زیر برای میکرو ATMEGA32 نوشته شده است. شما با تغییر REGFILE به میکرو دلخواه ، میتوانید این برنامه را برای دیگر میکروها استفاده نمایید. در برنامه زیر از LCD نوع 16*2 استفاده شده است. صفحه کلید خوانده می‌شود و یا به عبارتی GETKBD() عددی کوچکتر یا مساوی 15 را برگرداند، اگر کلید F1 باشد به معنای BACKSPACE ، کلید F2 به معنای SPACE، F3 به معنای ابتدای LCD ، F4 به معنای انتهای خط اول ، ENTER به معنای ابتدای خط دوم ،SET به معنای CLS است.

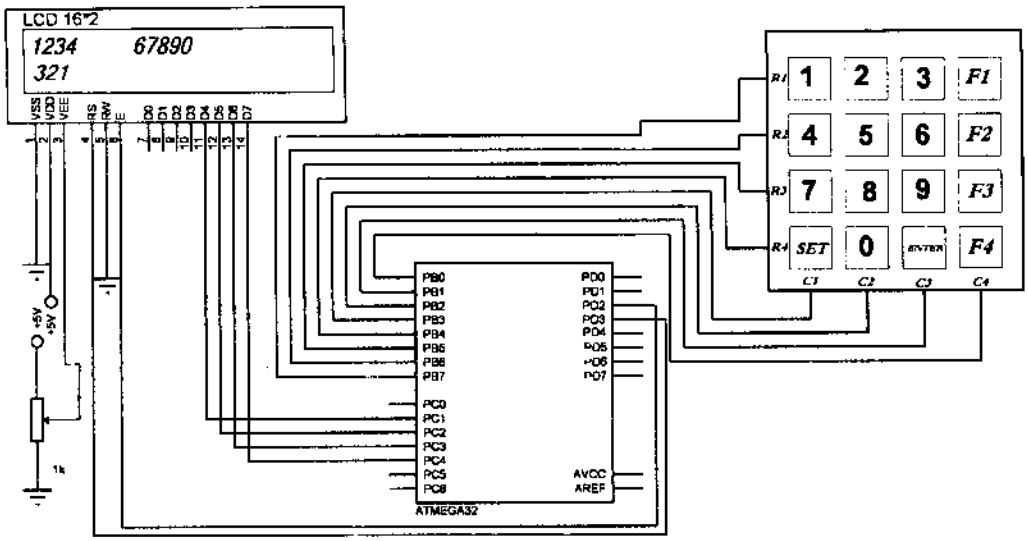
در برد STARTER KIT اگر سطر و ستون کی بورد طبق شکل ۸-۸ اتصال یابد اعداد گرفته شده توسط دستور GETKBD به صورت زیر است .



اعداد گرفته شده توسط دستور GETKBD در برد STARTER KIT

- **برنامه**

```
$regfile = "m32def.dat"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Portc.1 , Db5 = Portc.2 , Db6 = Portc.3,Db7= _
Portc.4 , E = Portd.2 , Rs = Portd.3
Config Lcd = 16 * 2
Dim A As Byte
Config Kbd = Portb , Debounce = 50 , Delay = 255
Cls
Home
 Do
 A = Getkbd()
 A = Lookup(a , Read_key)
 If A < 16 Then
  Select Case A
  Case 0 To 9:                    'number0...9
   Lcd A
  Case 10:                        'F1(backspace)
   Shiftcursor Left
   Lcd " "
   Shiftcursor Left
  Case 11:                        'F2(space)
   Shiftcursor Right
  Case 12:                       'F3
   Home
  Case 13:                       'F4
   Locate 1 , 16
  Case 14:                       'ENTER(NEXT LINE)
   Home L
  Case 15:
   Cls                           'set(clearscreen)
 End Select
End If
Loop
End                             'end program

Read_key:
Data 13 , 14 , 0 , 15 , 12 , 9 , 8 , 7 , 11 , 6 , 5 , 4 , 10 , 3 , 2 , 1
```



شکل ۸-۸ مدار اسکن صفحه کلید ۴×۴ طبق برد STARTER KIT

۸-۴ اسکن صفحه‌کلید کامپیوتر

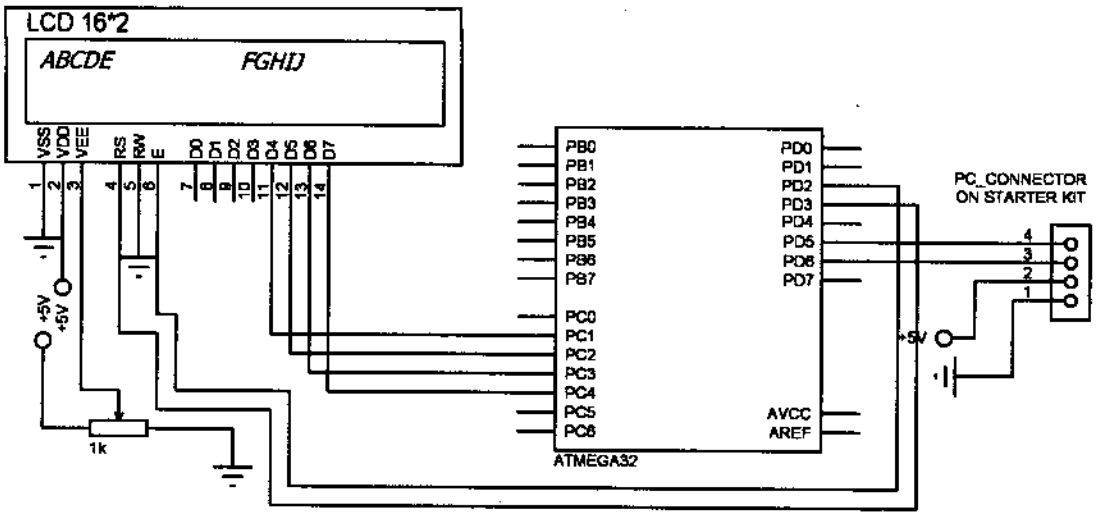
صفحه‌کلید توسط دستور CONFIG KEYBOARD پیکره‌بندی شده است. پایه CLOCK صفحه‌کلید به پایه PIND.5 و پایه DATA صفحه‌کلید به پایه PIND.6 متصل شده‌اند(شکل ۸-۹). دستور GETATKBD() مقدار اسکی کلیدها را بر نمی‌گرداند پس بنابراین برای تبدیل آنها به مقدار اسکی نیاز به یک جدول مترجم است. این جدول KEYDATA است که در زمان دریافت یک مقدار از صفحه‌کلید ، مقدار اسکی معادل آن از جدول برگردانده می‌شود . کلید فشرده شده از صفحه‌کلید توسط دستور GETATKBD( ) خوانده شده و در متغییر A قرار می‌گیرد.

در این برنامه با فشردن کلیدهای ↓ ، → و← به ترتیب مکان‌نمای LCD به خط پایین ، ستون راست و ستون چپ حرکت خواهد کرد. با فشردن کلید DELET (44) تمام صفحه نمایش پاک شده و مکان‌نما به سطر و ستون اول پرش می‌کند. با فشردن کلید HOME (56) مکان‌نما تنها به HOME (ستون و سطر اول) می‌رود. با فشردن کلید BACKSPACE (8) حرف ستون قبل پاک شده ومکان‌نما در آن مکان می‌ماند. با فشردن کلید ENTER (13) مکان‌نما به خط پایین پرش می‌کند و در نهایت با فشرده شدن دکمه TAB (9) مکان‌نما 4 ستون به راست جهش می‌کند.

- **برنامه**

```
$regfile = "M32def.dat"
$crystal = 8000000
```

```
Config Lcdpin = Pin , Db4 = Portc.1 , Db5 = Portc.2 , Db6 = Portc.3,Db7= _
Portc.4 , E = Portd.2 , Rs = Portd.3
Config Lcd = 16 * 2
Config Keyboard = Pind.5 , Data = Pind.6 , Keydata = Keydata
Dim A As Byte
Waitms 500
Cls
Main:
A = Getatkbd()
Select Case A:
      Case 50:
            Lowerline
      Case 56:
            Home
      Case 44:
            Cls
            Home
      Case 52:
            Shiftcursor Left
      Case 54:
            Shiftcursor Right
      Case 13:
          Lowerline
       Case 9:
            Shiftcursor Right , 4
      Case 8:
            Shiftcursor Left
           Lcd " ";
               Shiftcursor Left
        Case Else Lcd String(1 , A)
End Select
jmp main
End
'end program

Keydata:
'normal keys lower case
Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 27 , 0 , 0 , 0 , 0 , 9 , 9 , &H5E , 0
Data 0 , 0 , 0 , 0 , 0 , 113 , 49 , 0 , 0 , 0 , 122 , 115 , 97 , 119 , 50 , 0
Data 0 , 99 , 120 , 100 , 101 , 52 , 51 , 0 , 0 , 32 , 118 , 102 , 116 , 114 , 53 , 0
Data 0 , 110 , 98 , 104 , 103 , 121 , 54 , 7 , 8 , 44 , 109 , 106 , 117 , 55 , 56 , 0
Data 0 , 44 , 107 , 105 , 111 , 48 , 57 , 0 , 0 , 46 , 45 , 108 , 48 , 112 , 43 , 0
Data 0 , 0 , 0 , 0 , 0 , 92 , 0 , 0 , 0 , 0 , 13 , 0 , 0 , 92 , 0 , 0
Data 0 , 60 , 0 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0

'shifted keys UPPER case
Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
Data 0 , 0 , 0 , 0 , 0 , 81 , 33 , 0 , 0 , 0 , 90 , 83 , 65 , 87 , 34 , 0
Data 0 , 67 , 88 , 68 , 69 , 0 , 35 , 0 , 0 , 32 , 86 , 70 , 84 , 82 , 37 , 0
Data 0 , 78 , 66 , 72 , 71 , 89 , 38 , 0 , 0 , 76 , 77 , 74 , 85 , 47 , 40 , 0
Data 0 , 59 , 75 , 73 , 79 , 61 , 41 , 0 , 0 , 58 , 95 , 76 , 48 , 80 , 63 , 0
Data 0 , 0 , 0 , 0 , 0 , 96 , 0 , 0 , 0 , 0 , 13 , 94 , 0 , 42 , 0 , 0
Data 0 , 62 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0 , 0
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0
```



شکل ۸-۹ مدار اسکن صفحه کلید کامپیوتر

۸-۵ برنامه ساعت

در این برنامه از میکرو نوع ATMEGA32 و کریستال 8MHZ استفاده شده است. پایه‌های PINC.0 و PIND.7 که کلیدهای تنظیم ساعت و دقیقه به آنها متصل شده است به عنوان ورودی پیکره‌بندی شده‌اند و مقاومت های PULL-UP آنها فعال شده است. زمانی که هر یک از کلیدها فشرده شود ، زیرتابع مربوطه اجرا می‌شود. در این برنامه برای ایجاد زمان پایه 1s از دستور WAITms استفاده شده است. شما می‌توانید این زمان را با تایمرها بسازید. این برنامه توسط SIMULATOR داخلی BASCOM به طور کامل قابل تحلیل است.شکل مداری این پروژه همانند شکل ۸-۷ است.

- **برنامه**

```
$regfile = "M32def.dat"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Portc.1 , Db5 = Portc.2 , Db6 = Portc.3,Db7= _
Portc.4 , E = Portd.2 , Rs = Portd.3
Ddrc.0 = 0 : Portc.0 = 1
Ddrd.7 = 0 : Portd.7 = 1
Config Lcd = 16 * 2
Declare Sub Incr_h
Declare Sub Incr_m
Dim S As Byte , M As Byte , H As Byte
Dim A As Bit
Main:
S = 0 : M = 0 : H = 1
Cls : Home : Lcd "time:"
Do
A = Pinc.0
        If A = 0 Then Call Incr_h
        If Pind.7 = 0 Then Call Incr_m
        Locate 2 , 1
```

```
        Lcd "" ; H ; ":" ; M ; ":" ; S
        Waitms 995
        Incr S
      If S > 59 Then
            S = 0
            Incr M
            Shiftcursor Left , 2
            Lcd " "
            If M > 59 Then
               Incr H
               M = 0
            If H > 12 Then
             jmp main
            End If
            End If
        End If
Loop
End

Incr_m:         ' OR Sub Incr_m
        Incr M
        If M > 59 Then
                Cls : Home : Lcd "time:"
                M = 0
        End If
Return          'OR End Sub Incr_m

Incr_h:         ' OR Sub Incr_h
        Incr H
        If H > 12 Then
                H = 1 : Cls : Home : Lcd "time:"
        End If
Return          'OR End Sub Incr_h
```

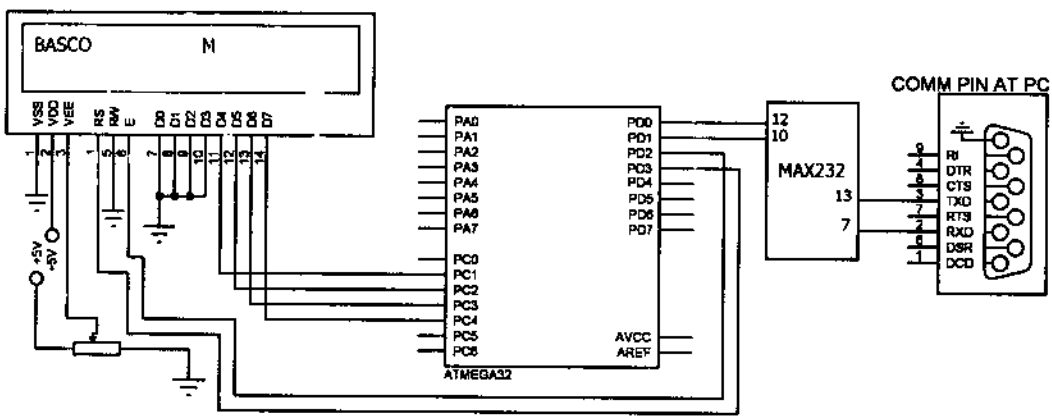
۸-۶ تابلو روان توسط LCD

این برنامه متنی را از محیط TERMINAL EMULATOR دریافت کرده و آن را حروف به حروف از سمت راست به چپ LCD انتقال می‌دهد. در ابتدای برنامه در محیط TERMINAL جمله INPUT TEXT THEN PRESS ENTER : نوشته می‌شود. با نوشتن متن مورد نظر که نهایتاً می‌تواند 10 کاراکتر طول داشته باشد و فشار دادن کلید ENTER برنامه اجرا می شود. این برنامه به طور کامل توسط SIMULATOR داخلی BASCOM قابل تحلیل است. توسط $SIM تحلیل سریعتر انجام می‌گیرد. با اتصال برد STARTER KIT به پورت سریال کامپیوتر می توان این برنامه را توسط TERMINAL EMULATOR محیط BASCOM ، UPLOAD کرده و اجرا کنید.

- **برنامه**

```
$regfile = "M32def.dat"
$crystal = 8000000
$baud = 9600
'$sim
Config Lcdpin = Pin , Db4 = Portc.1 , Db5 = Portc.2 , Db6 = Portc.3,Db7= _
Portc.4 , E = Portd.2 , Rs = Portd.3
Config Lcd = 16 * 2
Cls
Dim A As String * 10
```

```
Dim Pice As String * 1
Dim L As Byte
Dim W As Byte
Dim X As Byte
Dim Y As Byte
Dim I As Byte
Do
Cls
Home
Input "INPUT TEXT THEN PRESS ENTER:" , A
L = Len(A)
For W = 1 To L
   Pice = Mid(A , W , 1)
   Y = 16 - W
      For I = 1 To Y
         X = 16 - I
         Locate 1 , X
         Lcd Pice
         Incr X
         Locate 1 , X
         Lcd " "
         Waitms 100
   Next
Next
Wait 10
Loop
End                'end program
```



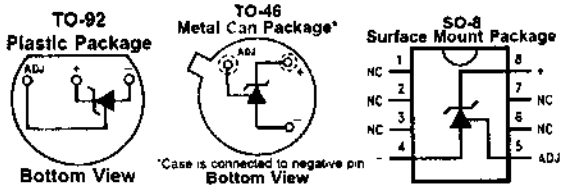
شکل ۸-۱۰ مدار بسته شده برای برنامه تابلو روان توسط LCD

۸-۷ نمایش دما بر روی LCD توسط سنسور دمای LM335

در این برنامه از سنسور دمای LM335 و اتصال آن به طور مستقیم به یکی از کانالهای ADC استفاده شده است. سری LM135 ، سنسورهای دقیق حرارت مدار مجتمع هستند که ولتاژ خروجی آنها به طور خطی متناسب با دمای کلوین است و در مدلهای LM135، LM235 و LM335 و بسته بندهای مختلف موجود است . جدول ۸-۲ زیر راهنمای انتخاب برای سنسورهای دمای سری LM135 می‌باشد. انواع بسته بندی سنسورهای سری LM135 در شکل ۸-۱۱ آمده است.

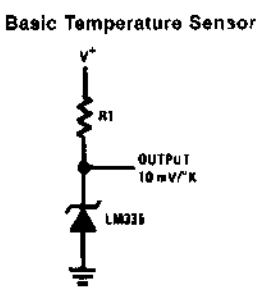
جدول ۸-۲ راهنمای انتخاب برای سنسورهای دمای سری LM135

DEVICE	TEMP RANGE	ACCURECY	OUTPUT SCALE	PACKAGE
LM335/335A	-40°C to +100°C	1. 5°C	+10 mV/°K	TO-92/SO-8
LM235/235A	-40°C to +125°C	1. 0°C	+10 mV/°K	TO-46
LM135/135A	-55°C to +150°C	1. 0°C	+10 mV/°K	TO-46

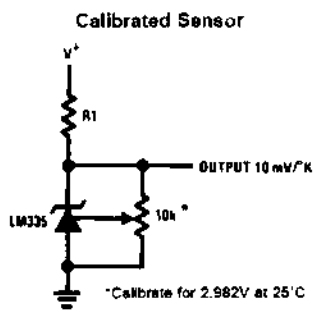


شکل ۸-۱۱ انواع بسته بندی سنسورهای سری LM135

نحوه بایاس کردن سنسور میتواند به صورت شکل ۸-۱۲ در ساده ترین شکل و یا به صورت شکل ۸-۱۳ با قابلیت کالیبره شدن باشد. +V می تواند از ولتاژ 5 تا 40 ولت تغییر کند.R1 نیز بایستی برای جریانهای 400uA تا 5mA محاسبه شود.



شکل ۸-۱۲



شکل ۸-۱۳

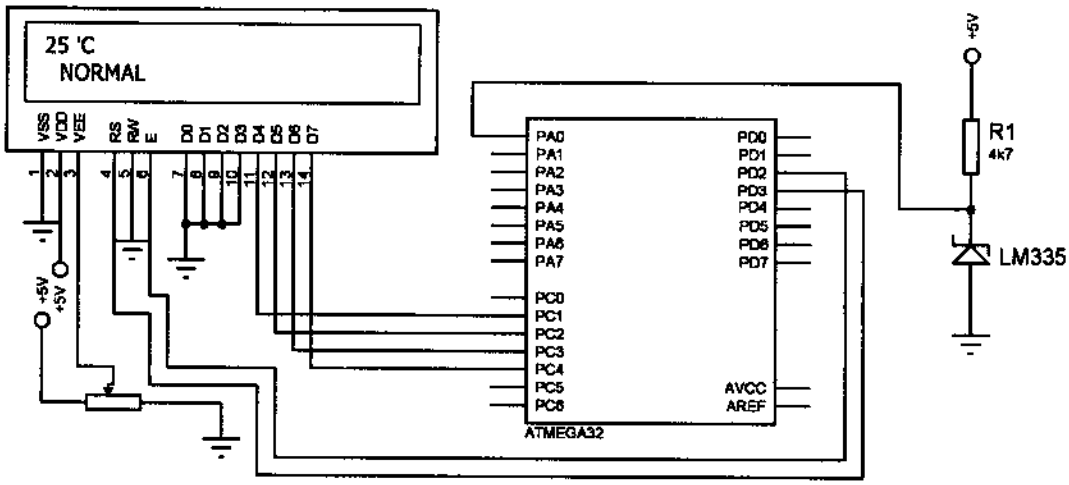
به ازاء ولتاژ ورودی 2.73v توسط LM335 به کانال ADC داخلی میکرو، عدد 558 در متغییر مربوطه قرار می‌گیرد . عدد 558(2.73v) نمایانگر 0 °C و 763(3.73v) نمایانگر 100 °C است. بنابراین با کم کردن مقدار افست و تقسیم مقدار دیجیتال به 2 می‌توان به سادگی دما را اندازه گرفت. شما از مدارات ZERO-SPAN می‌توانید برای تبدیل رنج به دامنه های بیشتراستفاده نمایید. در برنامه زیر از میکرو ATMEGA32 و LCD نوع 2 * 16 استفاده شده است (شکل ۸-۱۴). برای تنظیم CONTRAST LCD از پتانسیومتر مربوطه استفاده کنید.ADC در مُد SINGLE استفاده شده و کلاک آن (AUTO) با توجه به اسیلاتور توسط کامپایلر انتخاب می‌شود. به وسیله ADC داخلی ، سیگنال آنالوگ به دیجیتال تبدیل

شده و توسط دستور GETADC() گرفته شده و در متغیر مربوط نوشته می‌شود. در این برنامه از مقدار آنالوگ دما در عرض حدود 1 ثانیه 15 نمونه در فواصل 65 میلی ثانیه برداشته شده است و در نهایت از کل نمونه های برداشتی، میانگین گیری شده است.این مقدار بر 2 تقسیم شده و دمای بدست آمده بر روی LCD با علامت 'C که توسط Deflcdchar تعریف شده است نمایش داده می‌شود .مقدار دیجیتال با عدد 10 (10'C) مقایسه شده و در صورت کوچکتر بودن در ستون هشتم LCD عبارت Low نمایش داده خواهد شد ودر غیر اینصورت اگر دما بین 11 تا 34 درجه باشد عبارت Normal و در صورت بزرگتر بودن از 35 درجه سانتیگراد عبارت High نمایش داده خواهد شد. برنامه زیر توسط SIMULATOR داخلی BASCOM به طور کامل قابل تحلیل است.

- **برنامه**

```
$regfile = "m32def.dat"
'We use internal osc 4000000 for atmega32
$crystal = 4000000
Config Lcdpin = Pin , Db4 = Portc.1 , Db5 = Portc.2 , Db6 =Portc.3 ,Db7 = _
Portc.4 , E = Portd.2 , Rs = Portd.3
Config Lcd = 16 * 2
Config Adc = Single , Prescaler = Auto
Start Adc
Config Timer0 = Timer , Prescale = 1024
Enable Interrupts
Enable Timer0
On Timer0 Temp_sample
Start Timer0
Dim Sample As Word
Dim Temp As Word
Dim Temp_final As Integer
Dim Sum As Integer
Deflcdchar 0 , 24 , 24 , 32 , 32 , 32 , 32 , 32 , 32    '' sign -> (')
Cursor Off
Cls
Do

Loop
end                         'end program

Temp_sample:               'every 65 ms
Incr Sample                'incr sampling counter
Temp = Getadc(0)           'read adc(0) every 65ms
Sum = Sum + Temp           'sum the sample data
If Sample = 15 Then        '20 samples after about 1s
   Sample = 0              'reset sample counter
   Stop Timer0             'stop timer to calculation
   Sum = Sum / 15          'mean calculation
   Sum = Sum - 558         'decrementing offset
   Temp_final=Sum/2 'every 10mv is almost 2 step of adc (step=5/1023=4.88mv)
   Select Case Temp_final
      Case Is =< 10 :      'compare temperature
         Home L
         Lcd " Low    "
      Case 11 To 34 :      'compare temperature
         Home L
         Lcd " Normal  "
      Case Is >= 35 :      'compare temperature
         Home L
         Lcd " High    "
   End Select
```

```
    Home
    Lcd Temp_final ; Chr(0) ; "C     "   'show temperature
    Start Timer0
    Sum = 0
End If
Return
```



شکل۸-۱۴ مدار بسته شده برای برنامه نمایش دما بر روی LCD

۸-۸ خروجی آنالوگ توسط پالس PWM و VISUAL BASIC

در این پروژه قصد داریم از طریق برنامه ویژوال بیسیک مقدار آنالوگ را به پورت سریال ارسال کرده و میکرو متصل شده به پورت سریال عدد را دریافت کند و پالس PWM متناسب با ولتاژ آنالوگ را در پایه خروجی OCR1B ایجاد کند. به همین منظور بایستی دو برنامه یکی برای ویژوال و دیگری برای میکرو نوشته شود.

برنامه VB

محیط گرافیکی برنامه به صورت پنجره ۸-۱ است. در این پنجره می توان مقدار آنالوگ خود را که عددی بین 2500-9000mv را وارد کرد و توسط دکمه SEND_ANALOG فریمی را با فرمت زیر به میکرو ارسال کرد.

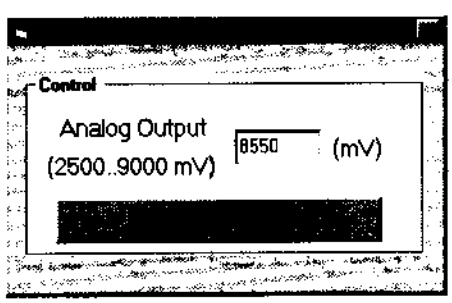
FIRST BYTE(&HAA)	LSB OF ANLOGE VALUE	MSB OF ANALOG VALUE	CHECK SUM OF 3 PREVIOUS BYTE

بایت اول (&HAA) برای شناسایی ابتدای فریم ارسال می شود. میکرو با دریافت این بایت سه بایت دیگر را پی در پی دریافت می کند. مقدار آنالوگ به ترتیب در دو بایت کم ارزش و پر ارزش ارسال می شود. در انتهای فریم نیز سه بایت ابتدایی با هم جمع شده و بایت کم ارزش آن به عنوان CHECK SUM(CS) ارسال می شود. این بایت جهت یافتن خطای ارسال دیتا به فریم ارسال می شود و

در صورتی که در انتهای خط مقدار آن با جمع سه بایت قبلی درست نباشد یکی از بایتهای دوم،سوم و یا خود CS در طول خط دچار خطا شده اند. برای ارتباط با پورت سریال در VB از MSCOMM استفاده شده است.

- **برنامه**

```
Dim F_BYTE As Variant
Dim lsb_Ao As Integer
Dim msb_Ao  As Integer
Dim cs As Integer
Dim tmp As Variant
--------------------------------
Private Sub Form_Load()
   MSComm1.Settings = "9600,N,8,1"
   MSComm1.RThreshold = 1
   MSComm1.SThreshold = 1
   MSComm1.InputLen = 1
   MSComm1.PortOpen = True
End Sub
--------------------------------
Private Sub SEND_COMMAND_Click()              'WHEN SEND_ANALOG CLICK
   tmp = Ao_change                            'READ THE VALUE OF ANALOG_OUTPUT
   F_BYTE = &HAA                              'FIRST BYTE OF FRAME
   cs = F_BYTE
   MSComm1.Output = Chr$(F_BYTE)              'SEND FIRST BYTE TO SERIAL PORT
   lsb_Ao = CInt(tmp Mod 256)                 'CALCULATE LSB OF ANALOG_OUTPUT
   MSComm1.Output = Chr$(lsb_Ao)              'SEND LSB OF ANALOG_OUTPUT
   msb_Ao = tmp / 256 - 0.5                   'CALCULATE MSB OF ANALOG_OUTPUT
   MSComm1.Output = Chr$(msb_Ao)              'SEND MSB OF ANALOG_OUTPUT
   cs = &HAA + lsb_Ao + msb_Ao5               'CALCULATE THE CS
   cs = cs And 255                            'CALCULATE THE LSB FO CS
   MSComm1.Output = Chr$(cs)                  'SEND CS OF SENDED BYTES
End Sub
```

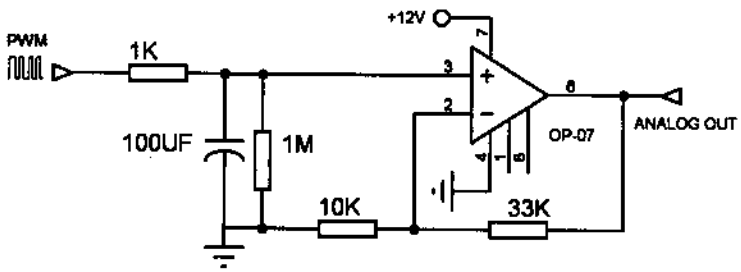


پنجره ۸-۱ نمای گرافیکی برنامه ساخت پالس PWM توسط VB

برنامه BASCOM

در این برنامه میکرو عدد ارسالی از برنامه VB به پورت سریال را دریافت و آن را تبدیل به پالس PWM می‌کند. فرکانس پالس PWM برابر با 391 HZ = (2046*10)/8000000 است. با تغییر PRESCALE و OSC شما به راحتی می‌توانید فرکانس‌های مختلف برای PWM ده بیتی ایجاد کنید. خروجی پالس

PWM پایه OCR1B است. پالس ایجاد شده به مدار شکل ۸-۱۵ وارد می شود. شکل کلی مدار را می توانید در شکل ۸-۱۹ که از کشیدن قسمت تبدیل پالس PWM به DC صرفنظر شده است، ملاحظه کنید. خروجی مقدار آنالوگ در برد STARTER-KIT از 2.5V تا 9.0V است. در برد STARTER-KIT به علت نداشتن ولتاژ منفی برای OP-AMP آفستی به اندازه 2.1V وجود دارد. اگر بخواهید ولتاژهای پایین تر را نیز داشته باشید بایستی از OP-AMP های SINGLE SUPPLY با ولتاژ OFFSET پایین استفاده نمایید.در برد کتاب از آپ امپ ، OP07 استفاده شده است. با مقدار PWM1B=120 ولتاژ 2.5V و با PWM1B=430 ولتاژ 9.0V بدست می آید. بنابراین با ضرب کردن مقدار دریافتی از نرم افزار VB در عدد 0.0477 مقدار خروجی پالس PWM به دست می آید.

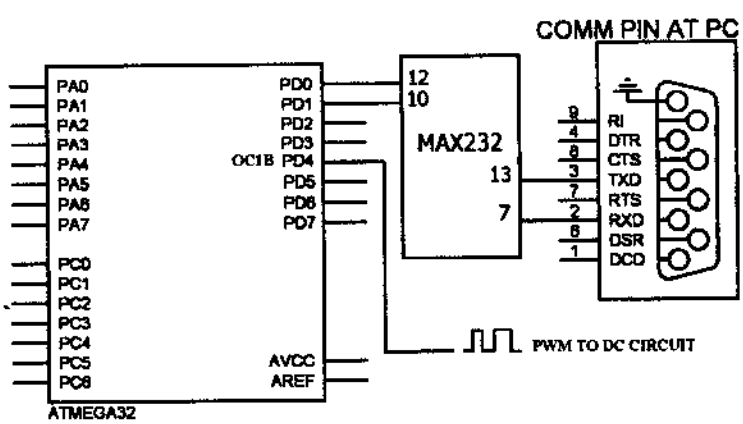


شکل ۸-۱۵ مدار تبدیل پالس PWM به ولتاژ آنالوگ

- **برنامه**

```
$regfile = "m32def.dat"
$crystal = 8000000
Config Timer1 = Pwm ,Prescale = 8 ,Pwm = 10 ,Compare A Pwm = Disconnect , _
Compare B Pwm = Clear Down
Config Pind.4 = Output
Config Lcdpin = Pin ,Db4 = Portc.1 ,Db5 = Portc.2 ,Db6 = Portc.3 ,Db7 = _
Portc.4 , E = Portd.2 , Rs = Portd.3
Config Lcd = 16 * 2
$baud = 9600
Enable Urxc
Enable Interrupts
On Urxc Ao_value_from_serial_port
Dim Serial_data(3) As Byte
Dim I As Integer
Dim Ao As Word
Dim Ao_mv As Word
Dim Msb_ao As Byte
Dim Lsb_ao As Byte
Dim Flag As Bit
Dim Cs As Byte
Dim Count As Byte
Dim Ao_output As Single
Dim Uart As Byte
Cls
Home
Lcd "Ao=" ; "2000" ; "mv " ; "PWM=" ; Pwm1b ; " "
Locate 2 , 1
```

```
Lcd " Analog Out Proj "
Do

Loop
End     'end program

Ao_value_from_serial_port:  'uart data reciving interrupt

Uart = Udr                  'reading uart data register (udr)
If Flag = 0 Then
   If Uart = &HAA Then      'finding the first byte of frame
      Cs = &HAA
      Flag = 1
      Count = 1
   End If
Else                        'goes here when detecing first byte
   Serial_data(count) = Uart
   If Count = 3 Then        'counting 3 byte after detecting first byte
      If Cs = Serial_data(3) Then 'compare calculated cs with reciving cs
         Lsb_ao = Serial_data(1)  'the cs is good   byte 2 if lsb
         Msb_ao = Serial_data(2)  'the cs is good   byte 3 if msb
         Ao = Makeint(lsb_ao , Msb_ao)  'making word
         Ao_mv = Ao
         Ao_output = Ao * 0.0477        'pwm value calculation
         Ao = Int(ao_output)
         Pwm1b = Ao                     'send pwm
         Home
         Lcd "Ao=" ; Ao_mv ; "mv " ; "PWM=" ; Pwm1b ; " "    'show pwm & Ao
         Locate 2 , 1
         Lcd " Analog Out Proj "
      End If
   Count = 0            'reset serial data counter
   Flag = 0
   End If
Incr Count
Cs = Cs + Uart          'cs(check sum) calculation
End If

Return
```



شکل ۸-۱۶ خروجی آنالوگ توسط پالس PWM

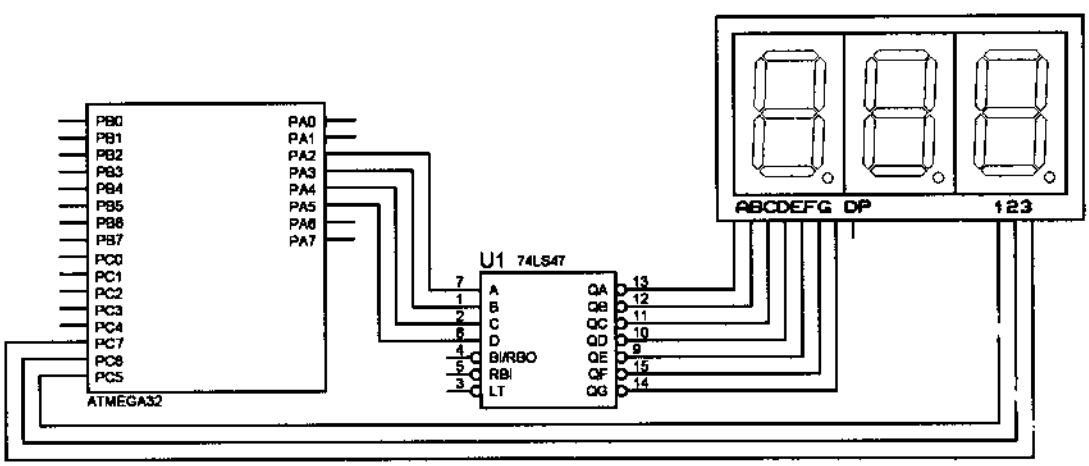
۹-۸ کار با 7Segment سه تایی

در این برنامه قصد داریم از 7seg سه تایی تعبیه شده بر روی برد کتاب استفاده نماییم. 7seg استفاده شده کاتد مشترک و برای راه اندازی آن از یک7447 استفاده شده است. هر یک از 7seg جداگانه و به ترتیب با فرکانسی که توسط تایمر صفر (150HZ) درست شده است،انتخاب میشوند و مقدار مربوط به آن که ممکن است یکان، دهگان یا صدگان بر روی پایه های متصل به 7447 قرار داده می شود. به طور مثال در اینتراپت اول 7seg اول، در اینتراپت دوم 7seg دوم و در اینتراپت سوم 7seg سوم انتخاب و عدد مربوط به هر یک نمایش داده می شود. این روند به همین ترتیب ادامه پیدا می کند.
در این برنامه متغییری با نام A هر یک ثانیه افزایش پیدا می کند و سپس بر روی 7SEG نمایش داده می شود.

- **برنامه**

```
$regfile = "m32def.dat"
$crystal = 8000000
Config Timer0 = Timer , Prescale = 1024
Enable Interrupts
Enable Timer0
On Timer0 7seg
Start Timer0
Timer0 = 204                            '150hz timer freq
Config Porta = Output
Config Portc = Output
Dim A As Byte
Dim Sel_7seg As Byte
Dim Sadgan As Byte
Dim Dahgan As Byte
Dim Yekan As Byte
Dim Flag As Byte
Dim Tmp As Byte

Do

Incr A
Sadgan = A / 100                        'calculating sadgan
Tmp = A Mod 100
Dahgan = Tmp / 10                       'calculating dadgan
Yekan = Tmp Mod 10                      'calculating yekan
Wait 1
Loop

End                                     'end program


7seg:
```

```
Timer0 = 204
Incr Sel_7seg
Select Case Sel_7seg                'every 150HZ ONE OF 7SEG SELECTED
 Case 1:
     If Sadgan <> 0 Then
     Portc.5 = 1 : Portc.6 = 0 : Portc.7 = 0    'select sadgan 7seg
     Flag = Sadgan                  'load value of sadgan to flag reg
     End If
 Case 2:
 If Dahgan <> 0 Or Sadgan > 0 Then
     Portc.5 = 0 : Portc.6 = 1 : Portc.7 = 0    'select dahgan 7seg
     Flag = Dahgan                  'load value of sadgan to flag reg
     End If
 Case 3:
     Portc.5 = 0 : Portc.6 = 0 : Portc.7 = 1    'select yekan 7seg
     Flag = Yekan                   'load value of sadgan to flag reg
     Sel_7seg = 0
End Select
Porta.2 = Flag.0 : Porta.3 = Flag.1 : Porta.4 = Flag.2 : Porta.5 = Flag.3
'send the value of the flag to 7447

Return
```



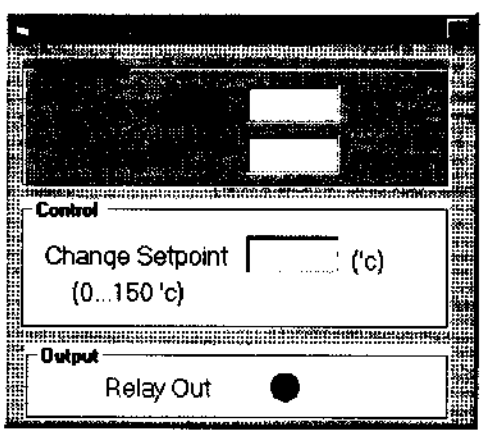
شکل مدار 7SEG

۸-۱۰ مانیتورینگ دما توسط نرم افزار VISUAL BASIC 6.0

در این برنامه ولتاژ آنالوگ پایه خروجی سنسور LM335 متناظر با دمای محیط توسط مبدل آنالوگ به دیجیتال داخلی میکرو به عدد دیجیتال تبدیل و به پورت سریال فرستاده و سپس توسط برنامه VISUAL BASIC نمایش داده می‌شود. نمای گرافیکی نمایش دما در محیطVISUAL BASIC در پنجره ۸-۲ نشان داده شده است.در قسمت MONITORING دو پارامتر دما و SETPOINT که از میکرو گرفته شده اند فقط نمایش داده می شوند.در قسمت CONTROL می توان SETPOINT جدیدی را وارد نمود و با زدن دکمه ENTER مقدار جدید را به میکرو ارسال کرد. زمانی که میکرو مقدار جدید را دریافت کرد رله(خروجی) را با توجه به این مقدار کنترل می کند و مقدار جدید را دوباره به کامپیوتر برای نمایش ارسال می کند. در صورتی که رله وصل باشد LED قسمت OUTPUT قرمز، و در حالت قطع به رنگ سبز در می آید.

برنامه VB

سیگنال آنالوگ توسط ADC میکرو دیجیتال شده و با باود 9600 توسط UART به پورت سریال کامپیوتر فرستاده می‌شود و در این سمت توسط برنامه VB با همان باود داده گرفته ، و دمای بدست آمده در TEMP_MON نمایش داده می‌شود.



پنجره ۸-۲ نمای گرافیکی نمایش دما در محیط VISUAL BASIC

- **برنامه**

```
Dim buf As Variant
Dim data(5) As Variant
Dim data_dorost(3)  As Variant
Dim counter As Variant
Dim cs As Variant
Dim sending As Boolean
Private Sub Form_Load()
MSComm1.Settings = "9600,N,8,1"
MSComm1.RThreshold = 1
MSComm1.SThreshold = 1
```

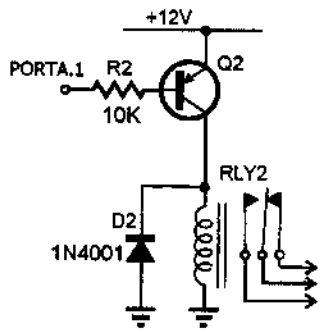
```
MSComm1.InputLen = 1
MSComm1.PortOpen = True
End Sub
-------------------------------------------------------
Private Sub MSComm1_OnComm()                    'PORT RECIVE INTERRUPT
Select Case MSComm1.CommEvent
   Case comEvReceive                          'RECIVING A BYTE FROM SERIAL PORT
       buf = MSComm1.Input                    'READING THE BYTE
       buf = Asc(buf)
       If buf = &HEB And counter = 0 Then 'IF THIS BYTE IS THE FIRST BYTE
          counter = 1
          data(counter) = buf
          cs = buf
          Exit Sub
       End If

       If counter <> 0 Then                   'IF THIS BYTE IS NOT THE FIRST BYTE
          counter = counter + 1
          data(counter) = buf
            If counter = 5 Then              'WHEN READING 5 BYTE
            cs = cs And 255
               If data(5) = cs Then          'IF CS IS GOOD
               temp_mon.Text = data(2) 'teparature
               setpoint_mon.Text = data(3) 'set point
               If data(4) = 0 Then relay_out.BackColor = RGB(0, 255, 0)_
               Else relay_out.BackColor = RGB(255, 0, 0) 'relay out
               End If
             counter = 0

             If sending = True Then                  'SEND SETPOINT NEW VALUE
                sending = False
                F_BYTE = &HAA                        'START BYTE
                cs = F_BYTE
                MSComm1.Output = Chr$(F_BYTE)        'SEND START BYTE
                MSComm1.Output = Chr$(setpoint_change.Text) 'SEND SETPOINT
                cs = F_BYTE + setpoint_change.Text'CS CALCULATING
                cs = cs And 255                      'LSB OF CS
                MSComm1.Output = Chr$(cs)            'SEND CHECK SUM
             End If


          End If
          cs = cs + buf
   End If
End Select

End Sub
-------------------------------------------------------
Private Sub setpoint_change_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then sending = True          'IF ENTER PRESS SET SENDING FLAG
End Sub
```

برنامه BASCOM

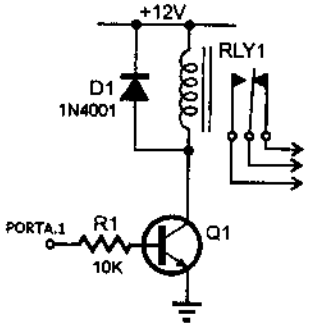
در این برنامه از میکرو ATMEGA32 با کریستال داخلی 8MHZ استفاده شده است. از سنسور LM335 در 1 ثانیه تعداد 30 نمونه با فواصل زمانی 32ms گرفته شده است. میانگین نمونه های دما به همراه SETPOINT و وضعیت رله در قالب یک فریم به شکل زیر به PC ارسال می شود. از طرف دیگر SETPOINT ارسالی از کامپیوتر توسط پورت سریال و به صورت اینتراپتی دریافت میشود.

START BYTE (&HEB)	TEMP	SET POINT	RELAY	CHECK SUM

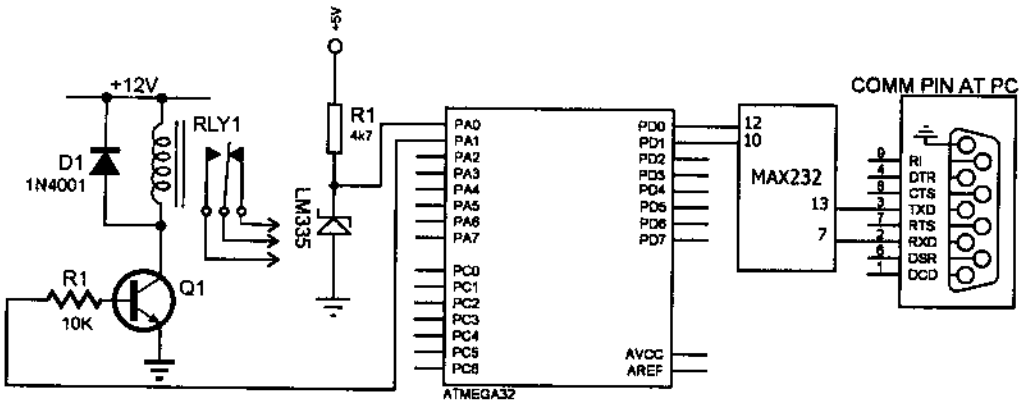
خروجی کنترلی این برنامه رله است. زمانی که دما از مقدار SETPOINT تعیین شده بالاتر رود رله خاموش و در غیر اینصورت رله روشن می شود. رله را می توان به صورت دو شکل ۸-۱۷ و ۸-۱۸ راه اندازی کرد.



شکل ۸-۱۷ درایو رله توسط ترازیستور PNP



شکل ۸-۱۸ درایو رله توسط ترازیستور NPN



شکل ۸-۱۹ شکل مدار بسته شده برای برنامه مانیتورینگ دما توسط VB

- **برنامه**

```
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 9600
Dim A As Byte
Dim Lm335 As Word
Dim Tmp As Word
Dim Temp_average As Word
Dim Final_temp As Byte
Dim Setpoint As Byte
Dim Relay_out As Byte
Dim Cs_temp As Word
Dim Count As Byte
Dim Serial_data(2) As Byte
Dim Uart As Byte
Dim Flag As Bit
```

```
Dim Cs As Byte
Config Pina.1 = Output                    'relay pin
Relay Alias Porta.1
Readeeprom Setpoint , 100                 'read saved setpoint from eeprom
Config Adc = Single , Prescaler = Auto ', Reference = Internal
Start Adc
Config Timer0 = Timer , Prescale = 1024
Enable Interrupts
Enable Timer0
On Timer0 32ms        'initial timer0 for 32ms overflow
Enable Urxc           'enable uart data revive interrupt
On Urxc Setpoint_from_pc 'jump to this label when reciving from serial port
Do

Loop
End          'end program

32ms:
Incr Count                      'counter for sampling tempreture
Lm335 = Getadc(0)               'reading temp
Cs_temp = Lm335 + Cs_temp       'adding samples
If Count = 30 Then              'when 1s expire and collect 30 samples
Stop Timer0
Temp_average = Cs_temp / 30 'mean of 30 samples
Temp_average = Temp_average - 558  'decrement offset
Final_temp = Temp_average / 2       'calculate total vaule
A = &HEB : A = String(1 , A) : Udr = A : Waitms 5  'send first byte of frame
A = Final_temp : A = String(1 , A) : Udr = A : Waitms 5  'send temprature
A = Setpoint : A = String(1 , A) : Udr = A : Waitms 5  'send setpoint value
A = Relay_out : A = String(1 , A) : Udr = A : Waitms 5   'send satuse of
relay
Cs_temp = &HEB + Final_temp
Cs_temp = Cs_temp + Setpoint
Cs_temp = Cs_temp + Relay_out                    'calculate check sum
A = Cs_temp : A = String(1 , A) : Udr = A : Waitms 5
'send check sum of previous 4 byte
Count = 0 : Cs_temp = 0

If Final_temp => Setpoint Then     'compare setpoint with process value(temp)
   Relay_out = 0                   'relay off
Elseif Final_temp < Setpoint Then 'compare setpoint with process value(temp)
   Relay_out = 1                   'relay on
End If
Relay = Relay_out.0                'acting the relay pin

Start Timer0
End If
Return

Setpoint_from_pc:
Uart = Udr                         'reading uart data register (udr)
If Flag = 0 Then
   If Uart = &HAA Then             'finding the first byte of frame
      Cs = &HAA
      Flag = 1
      Count = 1
   End If
Else
   Serial_data(count) = Uart
      If Count = 2 Then           'counting 2 byte after detecting first byte
      If Cs = Serial_data(2) Then  'compare calculated cs with reciving cs
         Setpoint = Serial_data(1) 'the cs is good   byte 2 if setpoint
         Writeeeprom Setpoint , 100
      End If
```

```
   Count = 0                          'reset serial data counter
   Flag = 0
   End If
Incr Count
Cs = Cs + Uart                        'cs(check sum) calculation
End If
Return
```

۸-۱۱ فرکانس‌متر دیجیتال

در این برنامه به طرح یک فرکانس متر دیجیتال می‌پردازیم. این برنامه بیشتر برای یادگیری با طرز کار تایمرها ، کانترها و نحوه کار با وقفه‌های موجود نوشته شده است. در این برنامه توسط تایمر 0 زمانی در حدود 1s درست شده و در طی این زمان تایمر 1 به صورت کانتر کار کرده و تعداد پالسهای TTL وارد شده به پایه T1 (PORTB.1) را می‌شمارد. پس از سپری شدن 1s ، تعداد پالسهای وارد شده به پایه T0 مشخص کننده فرکانس پالس ورودی است. در این برنامه بالاترین فرکانس اندازه‌گیری شونده 8MHZ است.

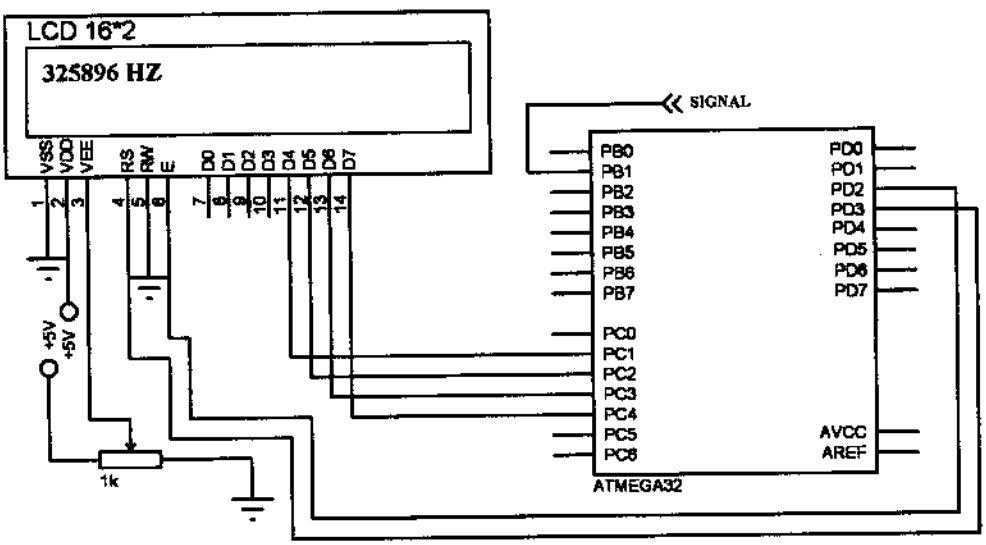
تولید 1s توسط T/C0 : $256*1024*30/8000000 \approx 0.983040s$

تایمر یک به صورت کانتر عمل می‌کند و در صورت سریز شدن یعنی زمانی که تعداد 65536 پالس را شمرد ، به زیربرنامه وقفه خود رفته و محتوای کانتر را با عدد صفر ری‌ست می‌کند و دوباره شروع به شمردن می‌کند. پس از سپری زمان 1s تعداد پالسهای شمرده شده در زیربرنامه وقفه تایمر صفر مشخص شده و در LCD نمایش داده می‌شود. ورودی کانتر ۱ نیز PULL-UP شده است تا فرکانس متر کار خود را درست انجام دهد.

- **برنامه**

```
$regfile = "M32DEF.dat"
'internal crystal 8000000
Config Lcdpin = Pin , Db4 = Portc.1 , Db5 = Portc.2 , Db6 =Portc.3 ,Db7 = _
Portc.4 , E = Portd.2 , Rs = Portd.3
Config Lcd = 16 * 2
Config Timer1 = Counter , Edge = Rising
DDRB.1=0:PORTB.1=1              'PULL UP RESISTOR ACTIVATED
Config Timer0 = Timer , Prescale = 1024
Enable Counter1
Enable Interrupts
Enable Timer0
Enable Timer1
On Ovf1 Pulsecount
On Ovf0 Ovf0occures
Dim A As Long , I As Long , B As Byte
B = 0
Cls
Start Timer0
Do

Loop
End             'end program

Ovf0occures:
```

```
Incr I
If I > 30 Then
Stop Timer0
Cls : Home
A = B * 65536
A = A + Counter1
Lcd "FREQ IS :" ; A ; "HZ
B = 0
I = 0 : Counter1 = 0
Start Timer0
End If
Return

Pulsecount:
Incr B
Counter1 = 0
Return
```

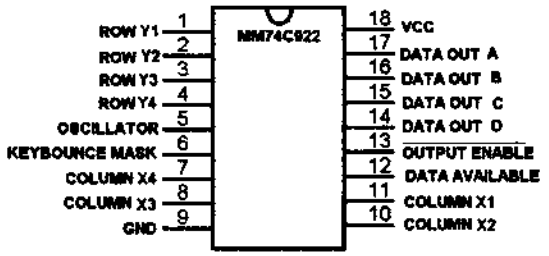


شکل ۸-۲۰ شکل مدار بسته شده برای برنامه فرکانس‌متر دیجیتال

۸-۱۲ اسکن صفحه‌کلید 4×4 توسط انکدر MM74C922

برای کاهش پایه‌های مصرفی میکرو در زمان اسکن صفحه کلید می‌توان از وسایل جانبی مانند انکدرها استفاده نمود. در این برنامه انکدر 74C922 (شکل ۸-۲۱) را به کار برده‌ایم. 74C922 انکدر 16 کلیدی و 74C923 انکدر 20 کلیدی است. برای اسکن کی بورد 4x4 از MM74C922 استفاده می‌کنیم.

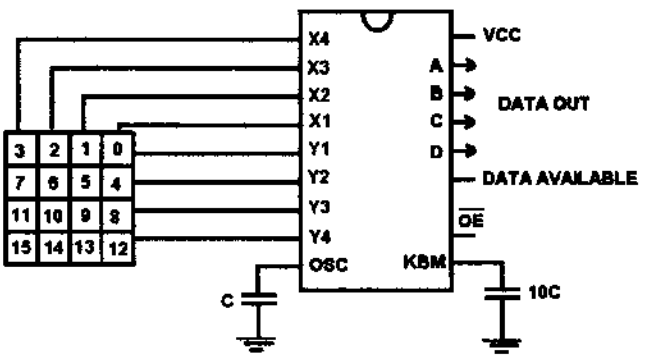
زمانی که کلیدی فشرده می‌شود پایه DATA AVAILABLE انکدر یک پالس مربعی ایجاد می‌کند. این پایه به پایه INT0 از میکرو متصل می‌شود زمانی که کلیدی فشرده شود میکرو وارد زیربرنامه وقفه شده و عدد متناظر را از روی پورت می‌خواند. پایه OE که فعال پایین است، نیز زمین می‌شود.طرز اتصال انکدر به کی بورد در شکل ۸-۲۲ آمده است. مدار کلی این پروژه در شکل ۸-۲۴ آمده است.



شکل ۸-۲۱ ترکیب پایه‌های انکدر MM74C922

نکته

مقدار خازنCKBM = 10COSC است.



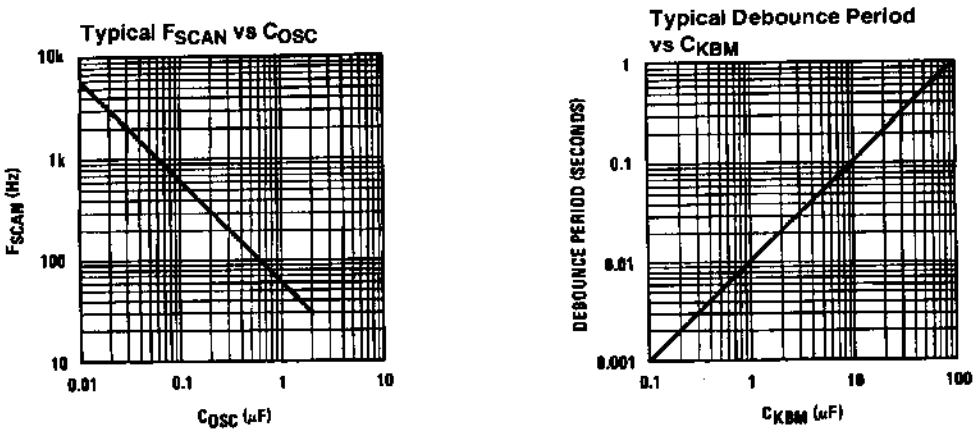
شکل۸-۲۲ اتصال صفحه کلید به انکدر MM74C922

برای بدست آوردن خازن متصل بین زمین و پایه‌های OSCILLATOR ، KEYBOUNCE MASK و یا به عبارتی برای تعیین زمان DEBOUNCD و فرکانس اسکن صفحه کلید از دو نمودار شکل ۸-۲۳ استفاده می‌شود. در این پروژه COSC=1uf و CKBM=10uf است.

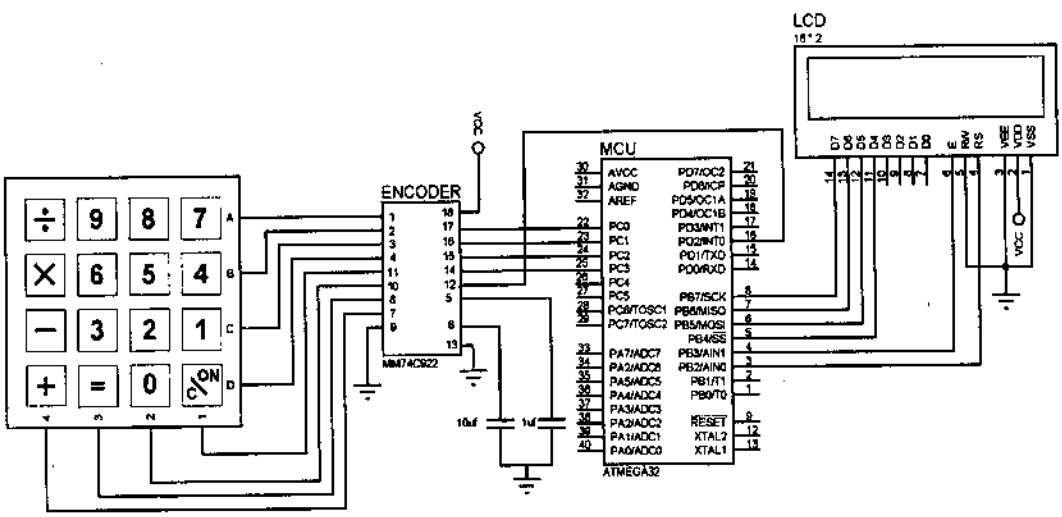
- **برنامه**

```
$regfile = "M16def.dat"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Pinb.4 , Db5 = Pinb.5 , Db6 = Pinb.6 , Db7 =_
Pinb.7 , Rs =Pinb.2 , E = Pinb.3
Config Lcd = 16 * 2
Config Portc = Input
Config Portb = Output
Dim Asd As Byte
Dim A As Byte
Wait 1
Cls
Enable Interrupts
Enable Int0
Config Int0 = Rising
On Int0 Int11
Do
```

```
Loop
End                                    'end program

Int11:
A = Pinc
A = A And &B00001111
Home
Lcd A
Return
```



شکل ۸-۲۳ نمودارهای تعیین فرکانس اسکن و زمان DEBOUNCE برای اسکن صفحه کلید



شکل ۸-۲۴ مدار بسته شده برای پروژه اسکن صفحه کلید ۴×۴ توسط انکدر MM74C922

۸-۱۳ موتور پله‌ای STEPER MOTOR

موتور پله‌ای وسیله پرمصرفی است که پالس‌های الکتریکی را به حرکت مکانیکی تبدیل می‌کند. در کاربردهای همچون راه‌اندازهای دیسک سخت ، چاپگرهای مغناطیسی و رباتیک ، از موتور پله‌ای برای کنترل موقعیت استفاده شده است. هر موتور پله‌ای دارای یک هسته متحرک مغناطیسی دایمی است که روتور یا شفت هم خوانده می‌شود و بوسیله یک بخش ثابت به نام استاتور احاطه شده است. معمول‌ترین موتورهای پله‌ای ، چهار سیم استاتور دارند که با سر وسط جفت شده اند. این نوع موتورها معمولاً به موتور پله‌ای چهار فاز معروفند . موتور پله‌ای که در این برنامه استفاده شده است دارای 5 سیم است : 4 سیم برای چهار سیم پیچ استاتور و سر دیگر که ولتاژ تغذیه متصل می‌گردد. با اعمال رشته‌هایی از تغذیه یا پالس به هر سیم پیچ استاتور ، روتور خواهد چرخید. رشته‌های مرسوم متعددی موجودند که هر یک دقت متفاوتی را دارا هستند. جدول ۸-۳ یک رشته چهار پله نرمال را نشان می‌دهد. باید توجه داشت که اگر چه می‌توان با هر یک از رشته‌های موجود در جدول فوق آغاز کرد ولی به محض شروع باید ترکیب رعایت شود.

جدول۸-۳ جدول رشته چهار پله

در جهت ساعت	# پله	سیم پیچ A	سیم پیچ B	سیم پیچ C	سیم پیچ D	خلاف جهت ساعت
	1	1	0	0	0	
	2	0	1	0	0	
	3	0	0	1	0	
	4	0	0	0	1	

زاویه پله

زاویه پله مشخصه موتور را نشان می‌دهد که به ازاء هر پله موتور چند درجه چرخش دارد. جدول ۸ -۴ بعضی از زوایا را برای انواع موتورها نشان می‌دهد.

جدول ۸-۴ زاویه پله برای انواع موتورها

پله در دور	زاویه پله
500	0.72
200	1.8
180	2.0
144	2.5
72	5.0
48	7.5
24	15

پله در ثانیه و دور در دقیقه (RPM)

رابطه بین RPM یـا دور در دقیقـه ، پلـه در دور(SPT) و پلـه در ثانیـه (SPS) از رابطـه زیـر بدسـت می‌آید :

SPS = RPM * SPT / 60

یک چرخش کامل

موتور پله‌ای مورد استفاده در این پروژه ، دارای زاویه پله 1.8 درجه می‌باشد و برای یک دور چـرخش نیازمند 200 پالس می‌باشد (200= 1.8 / 360). با شمارش پالسها ، میکروکنترلر AVR می‌توانـد موقعیـت روتور را تنظیم کرده و آن را کنترل کند.

راه‌انداز نیم‌پله

تحریک نیم‌پله موجب می‌شود که دقت موتور دو برابر شود. در این حالت با توجه به موتور استفاده شده نیاز به 400 پالس می‌باشد. جدول ۸-۴ منطق مورد نیاز برای برنامه را نشان می‌دهد. وقتی به انتهای جدول رسیدید ، این روند را تکرار کرده و دوباره از ابتدای جدول شروع کنید.

جدول ۸-۴ رشته نیم پله

در جهت ساعت	#پله	سیم پیچ A	سیم پیچ B	سیم پیچ C	سیم پیچ D	خلاف جهت ساعت
	1	1	0	0	0	
	2	1	1	0	0	
	3	0	1	0	0	
	4	0	1	1	0	
	5	0	0	1	0	
	6	0	0	1	1	
	7	0	0	0	1	
	8	1	0	0	1	

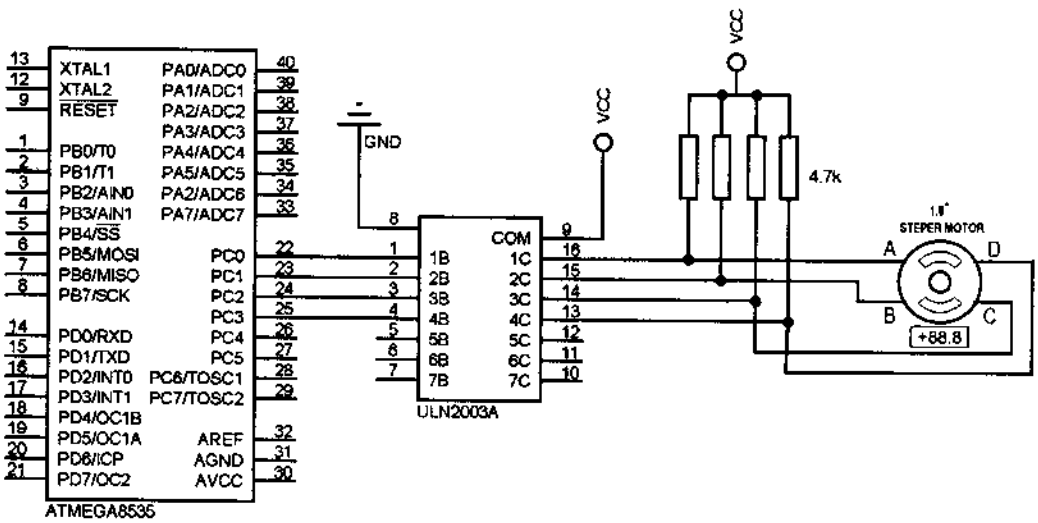
برنامه موتور پله‌ای

در این پروژه برای راه‌اندازی موتور پله‌ای از راه‌انداز ULN2003 استفاده شده ولی می توان از راه اندازهای پیشرفته و با قابلیت بالاتر همچون ترکیب L297 و L298 استفاده کرد. درایورULN2003 دارای 7 پایه راه‌انداز کلکتور باز است و بیشترین جریان خروجی آن برای هر پایه 500mA می‌باشد. جدول ۸-۵ چهار نوع از خانواده ULN را نشان می‌دهد.

جدول ۸–۵ چهار نوع از خانواده ULN

ULN2001A	GENERAL PURPOSE , DTL , TTL, PMOS, CMOS
ULN2002A	14 – 25V PMOS
ULN2003A	5V TTL , CMOS
ULN2004A	6- 15V ,CMOS, PMOS

این برنامه یک موتور پله‌ای با دقت 1.8 درجه را مدام یک بار پله کامل و بار دیگر به صورت نیم‌پله به اندازه 360 درجه در جهت ساعت می‌چرخاند. شماتیک مدار بسته شده در شکل ۸–۲۵ آمده است.



شکل ۸–۲۵ مدار بسته شده برای برنامه موتور پله ای

- **برنامه**

```
$regfile = "m32def.dat"              ' we use the ATMEGA32
$crystal=8000000
config PORTC = output
Dim A As Byte , b as Byte , e as Byte , f as Byte
Do
for a = 1 to 50
      e = 128
   for b = 1 to 4
      rotate e , left
      portc = e
      waitms 20
   next b
next a
wait 1
for a = 1 to 50
   f = 129
   e =128
      for b = 1 to 4
```

```
      rotate e , left
      portc = e
      waitms 20
      rotate f , left
      if f = 24 then f = 9
      portc = f
      waitms 20
    next b
next a
Loop
End                'end program
```

۱۴-۸ ارتباط سریال دو میکرو از طریق باس SPI

در این پروژه دو میکرو با استفاده از پروتکل ارتباطی سریال SPI با یکدیگر ارتباط برقرار می‌کنند. در این ارتباط یکی از میکروها MASTER و دیگری SLAVE پیکره‌بندی می‌شود. صفحه کلیدی به MASTER متصل است ، با فشرده شدن کلید عدد متناسب با کلید فشرده شده از طریق خط SPI به SLAVE انتقال و توسط SLAVE بر روی LCD نمایش داده می‌شود.

ارتباط سریال دو میکرو از طریق باس SPI بدون استفاده از وقفه

برنامه MASTER

MASTER میکرو نوع ATMEGA32 با کریستال 8MHZ است. ارتباط SPI به صورت سخت‌افزاری و با پایه‌های پیش فرض که در پورت B قرار دارد انجام می‌گیرد. وقفه غیرفعال است و LSB داده ابتدا فرستاده می‌شود ، در حالت بیکاری پایه کلاک بالا نگه داشته می‌شود. فرکانس کلاک SPI برابر با (128 / فرکانس اسیلاتور) می‌باشد. توسط دستور SPIINIT ارتباط شکل‌دهی می‌شود. عدد متناسب با کلید توسط دستور ()GETKBD گرفته شده و با دستور SPIOUT به باس SPI ارسال می‌شود.

- برنامه

```
$regfile = "m32def.dat"
$crystal = 8000000
Config Spi = Hard , Interrupt = Off , Data Order = Lsb , Master = Yes ,_
Polarity = High , Phase = 1 , Clockrate = 128
Config Kbd = Portc
Dim B As Byte
Spiinit
Do
    B = Getkbd()
    If B < 16 Then
      Spiout B , 1
    End If
Loop
End
```

برنامه SLAVE

SLAVE میکرو نوع ATMEGA8535 با کریستال داخلی8MHZ است. ارتباط SPI به صورت سخت افزاری و با پایه‌های پیش فرض که در پورت B قرار دارد انجام می گیرد. وقفه غیرفعال است و LSB داده ابتدا گرفته می‌شود ، در حالت بیکاری پایه کلاک بالا نگه داشته می‌شود. فرکانس انتقال داده برابر با (128 / فرکانس اسیلاتور) می‌باشد. توسط دستور SPIINIT ارتباط initial می‌شود. عدد متناسب با کلید که توسط MASTER ارسال شده است با دستور SPIIN توسط SLAVE از باس SPI دریافت شده و برروی LCD نمایش داده می‌شود.

- **برنامه**

```
$regfile = "M8535.dat"
' internal crystal = 8000000
Config Lcdpin = Pin , Db4 = Pinc.4 , Db5 = Pinc.5 , Db6 = Pinc.6 , Db7 =_
Pinc.7 , E = Pinc.3 , Rs = Pinc.2
Config Spi = Hard , Interrupt = Off , Data Order = Lsb , Master = No_
,Polarity = High , Phase = 0 , Clockrate = 128
Dim A As Byte
Spiinit
Do
Spiin A , 1
Home
Lcd A
Loop
End
```

ارتباط سریال دو میکرو از طریق SPI با استفاده از وقفه

در این ارتباط یکی از میکروها MASTER و دیگری SLAVE پیکره‌بندی می‌شود. صفحه کلیدی به MASTER متصل است ، با فشرده شدن کلید عدد متناسب با کلید فشرده شده از طریق خط SPI به SLAVE انتقال داده و توسط SLAVE بر روی LCD نمایش داده می‌شود. در این ارتباط از وقفه SPI استفاده شده است. زمانی که انتقال داده SPI کامل شود ، وقفه SPI روی می‌دهد. در این برنامه از این وقفه استفاده شده است ، اجرای برنامه روال عادی خود را دارد و زمانی که MASTER داده‌ای به SLAVE ارسال کند، SLAVE برنامه را موقتاً متوقف نموده و به وقفه پاسخ می‌دهد و داده را از باسSPI می‌گیرد و سپس اجرای ادامه برنامه انجام می‌گیرد.

برنامه MASTER

MASTER میکرو نوع ATMEGA32 با کریستال 8MHZ است. ارتباط SPI به صورت سخت‌افزاری و با پایه‌های پیش‌فرض که در پورت B قرار دارد انجام می‌گیرد. وقفه MASTER غیر فعال است و LSB داده ابتدا فرستاده می‌شود ، در حالت بیکاری پایه کلاک بالا نگه داشته می‌شود. فرکانس انتقال داده برابر با (128 / فرکانس اسیلاتور) می‌باشد. توسط دستور SPIINIT ارتباط شکل‌دهی می‌شود. عدد متناسب با کلید توسط دستور GETKBD() گرفته شده و با دستور SPIOUT به باس SPI ارسال می‌شود.

- **برنامه**

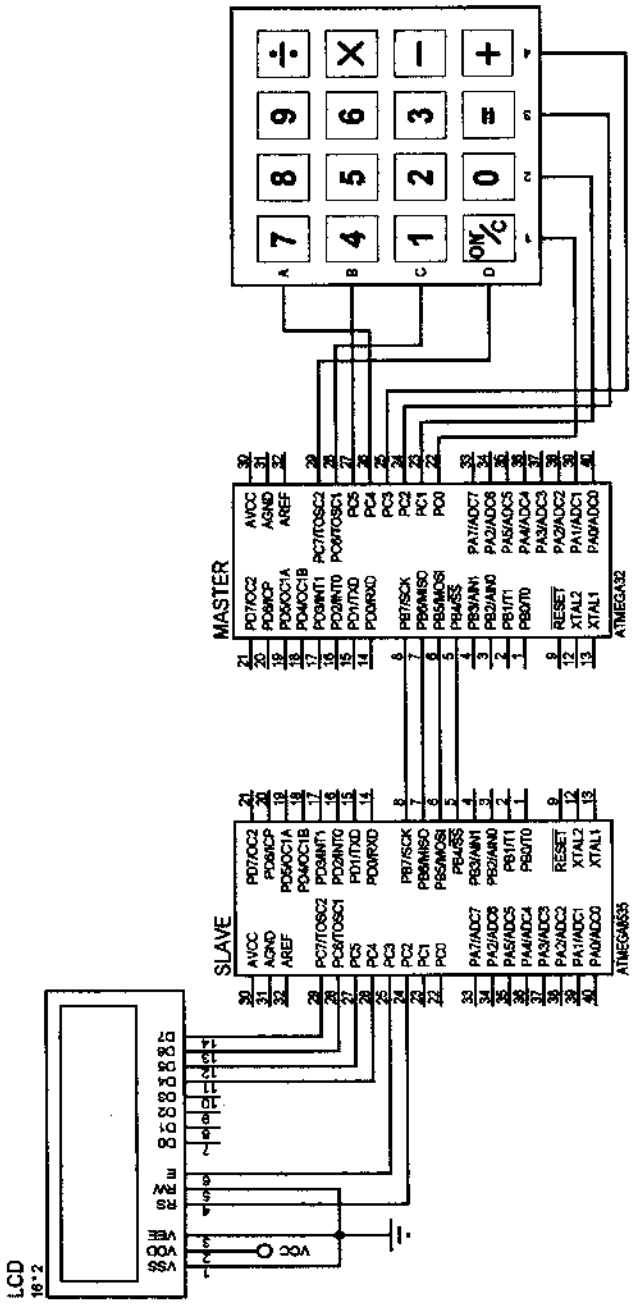
```
$regfile = "m32def.dat"
$crystal = 8000000
Config Spi = Hard , Interrupt = On , Data Order = Lsb , Master = Yes ,_
Polarity = High , Phase = 1 , Clockrate = 128
Config Kbd = Portc
Dim B As Byte
Spiinit
Do
    B = Getkbd()
    If B < 16 Then
      Spiout B , 1
    End If
Loop
End
```

برنامه SLAVE

MASTER میکرو نوع ATMEGA8535 با کریستال داخلی8MHZ است. ارتباط SPI به صورت سخت افزاری و با پایه‌های پیش‌فرض که در پورت B قرار دارد انجام می‌گیرد. وقفه فعال است و LSB داده ابتدا گرفته می‌شود ، در حالت بیکاری پایه کلاک بالا نگه داشته می‌شود. فرکانس انتقال داده برابر با (128 / فرکانس اسیلاتور) می باشد. توسط دستور SPIINIT ارتباط پیکره‌بندی می‌شود. نمایش ON PROGRAM بر روی LCD نمایانگر اجرای عادی برنامه است و زمانی که عدد متناسب با کلید توسط MASTER ارسال شود باعث ایجاد وقفه SPI در SLAVE می‌شود و در نتیجه زیربرنامه وقفه به نام MASTERSELECT اجرا شده ومحتوای رجیستر داده SPI با نام SPDR بر روی LCD نمایش داده می‌شود.

- **برنامه**

```
$regfile = "m8535.dat"
'internal crystal = 8000000
Config Lcdpin = Pin , Db4 = Pinc.4 , Db5 = Pinc.5 , Db6 = Pinc.6 , Db7 =_
Pinc.7 , E = Pinc.3 , Rs = Pinc.2
$baud = 9600
Config Spi = Hard , Interrupt = On , Data Order = Lsb , Master = No ,_
Polarity = High , Phase = 0 , Clockrate = 128
Enable Interrupts
Enable Spi
On Spi Masterselect
Dim B As Byte
Spiinit
Do
Home
Lcd "on program"
Loop
End

Masterselect:
Disable Interrupts
Cls
Home
Lcd Spdr
Wait 1
Enable Interrupts
Return
```



شکل ۸-۲۶ مدار بسته شده برای دو برنامه ارتباط SPI با وقفه و بدون وقفه

۸-۱۵ ارتباط با حافظه EEPROM سریال AT24C256

حافظ EEPROM سریال AT24C256 دارای 256KBIT حافظه است که قابلیت ارتباط با پروتکل سریال 2-WIRE (I2C) را داراست. آدرس سخت‌افزاری آن طبق بایت زیر بدست می‌آید.

1	0	1	0	0	A1	A0	R/W
MSB							LSB

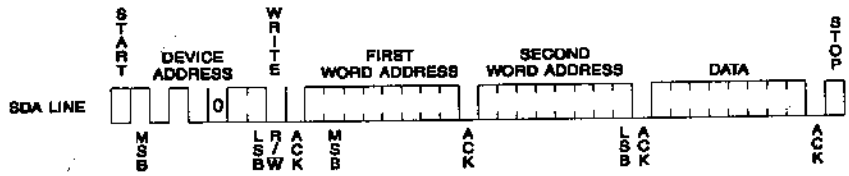
(بایت آدرس‌دهی سخت‌افزاری)

در صورتی که بخواهیم در حافظه بنویسیم بیت R/W را صفر و در صورتی که قصد خواندن داشته باشیم آن را یک قرار می‌دهیم.پس طبق بایت فوق، آدرس سخت‌افزاری نوشتن حافظه‌ای که پایه‌های A0 و A1 آن زمین شده است برابر 160 = &B10100000 است و آدرس سخت‌افزاری خواندن حافظه‌ای که پایه‌های A0 و A1 آن زمین شده است برابر 161 = &B10100001 است. ارسال آدرس حافظه با دو بیت انجام می‌گیرد، ابتدا بیت بالای آدرس اول و سپس بیت پایین آدرس و سپس بیت داده ارسال می‌گردد.

برنامه نوشتن در حافظه

برای نوشتن حافظه به صورت بایتی بایستی مراحل زیر بر روی پایه SDA طی شود: (طبق شکل۸-۲۷)

۱. ایجاد STARTCONDITION توسط دستور I2CSTART
۲. ارسال بایت آدرس‌دهی سخت‌افزاری برای نوشتن (160) توسط دستور I2CWBYTE که توسط بایت آدرس‌دهی سخت‌افزاری بدست آمده است.
۳. ایجاد ACK که کامپایلر خودکار آن را ایجاد کرده و دیگر نیازی نیست برنامه‌ای برای آن نوشته شود.
٤. ارسال ابتدای بایت بالای آدرس حافظه دلخواه توسط دستور I2CWBYTE . این بایت می‌تواند توسط دستور HIGH از یک متغییر دو بایتی بدست آید.
٥. ایجاد ACK که کامپایلر خودکار آن را ایجاد کرده و دیگر نیازی نیست برنامه‌ای برای آن نوشته شود.
٦. ارسال بایت پایین آدرس حافظه دلخواه توسط دستور I2CWBYTE . این بایت می‌تواند توسط دستور LOW از یک متغییر دو بایتی بدست آید.
۷. ایجاد ACK که کامپایلر خودکار آن را ایجاد کرده و دیگر نیازی نیست برنامه‌ای برای آن نوشته شود.
۸. ارسال داده به چیپ آدرس‌دهی (سخت‌افزاری) شده.
۹. ایجاد STOP CONDITION توسط دستور I2CSTOP
۱۰. ایجاد تاخیر توسط دستور WAITms به مقدار 5ms برای تکمیل نوشتن



شکل ۸-۲۷ مراحل نوشتن در حافظه سریال

برنامه زیر در تمام آدرس‌های حافظه EEPROM عدد 7 را ذخیره می‌کند. با دستورات HIGH و LOW بیت بالا و پایین متغییر address از نوع داده WORD جدا شده‌اند.

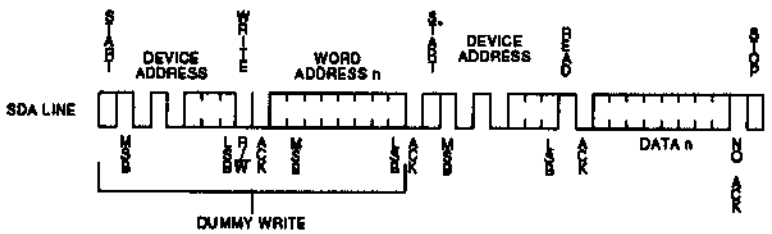
- **برنامه**

```
$regfile = "M32def.dat"
$crystal = 8000000
$baud = 9600
Config Serialout = Buffered , Size = 10
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7_
= Portc.7 , E = Portc.3 , Rs = Portc.2
Config Sda = Portc.0
Config Scl = Portc.1
Config I2cdelay = 1
Config Kbd = Porta
Declare Sub Writeeepromserial
Const Eewrite = 160
Const Eeread = 161
Dim A As Byte
dim b as Byte
Dim Address As word
Lcd "Start Writing..."
address = &h0000

while address < &h7FFF
a = high(address)
b = low(address)
Call Writeeepromserial
Incr Address
wend

End                                    'end program

Sub Writeeepromserial
   I2cstart
   I2cwbyte Eewrite
   i2cwbyte a
   I2cwbyte b
   i2cwbyte 7
   I2cstop
   Waitms 10
End Sub Writeeepromserial
```

برنامه خواندن از حافظه

برای خواندن از حافظه به صورت بایتی بایستی مراحل زیر بر روی پایه SDA طی شود (طبق شکل ۸-۲۸) :

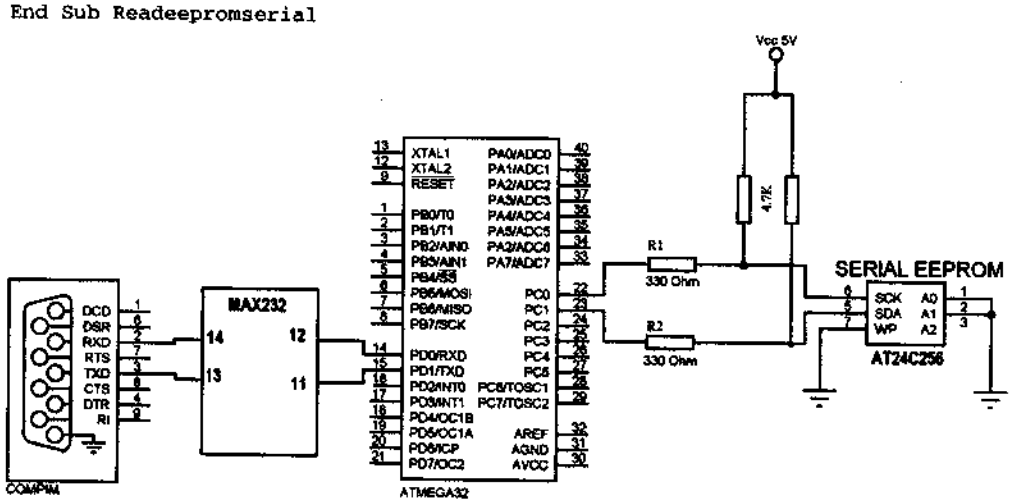
۱. ایجاد STARTCONDITION توسط دستور I2CSTART
۲. ارسال بایت آدرس‌دهی سخت‌افزاری برای نوشتن (160) توسط دستور I2CWBYTE که توسط بایت آدرس‌دهی سخت‌افزاری بدست آمده است.
۳. ایجاد ACK که کامپایلر خودکار آن را ایجاد کرده و دیگر نیازی نیست برنامه‌ای برای آن نوشته شود.
٤. ارسال ابتدای بایت بالای آدرس حافظه دلخواه توسط دستور I2CWBYTE . این بایت می‌تواند توسط دستور HIGH از یک متغییر دو بایتی بدست آید.
۵. ایجاد ACK که کامپایلر خودکار آن را ایجاد کرده و دیگر نیازی نیست برنامه‌ای برای آن نوشته شود.
٦. ارسال بایت پایین آدرس حافظه دلخواه توسط دستور I2CWBYTE . این بایت می‌تواند توسط دستور LOW از یک متغییر دو بایتی بدست آید.
۷. ایجاد ACK که کامپایلر خودکار آن را ایجاد کرده و دیگر نیازی نیست برنامه‌ای برای آن نوشته شود.
۸. ایجاد STARTCONDITION مجدد توسط دستور I2CSTART
۹. ارسال بایت آدرس‌دهی سخت‌افزاری برای خواندن (161) توسط دستور I2CWBYTE که توسط بایت آدرس‌دهی سخت‌افزاری بدست آمده است.
۱۰. دریافت داده از چیپ آدرس‌دهی(سخت‌افزاری) شده برای خواندن توسط دستور I2CRBYTE data , NACK .
۱۱. ایجاد STOP CONDITION توسط دستور I2CSTOP



شکل ۸-۲۸ مراحل خواندن از حافظه سریال

برنامه زیر از تمام آدرس EEPROM می‌خواند. در این برنامه شما می‌توانید از محیط TERMINAL EMULATOR محیط BASCOM و یا از HYPER TERMINAL محیط ویندوز برای نمایش داده خوانده شده از حافظه استفاده نمایید. با دستورات HIGH و LOW بایت بالا و پایین متغییر address از نوع داده WORD جدا می‌شوند.

```
$regfile = "M32def.dat"
$crystal = 8000000
$baud = 9600
Config Serialout = Buffered , Size = 255
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 =_
Portc.7 , E = Portc.3 , Rs =  Portc.2
Config Sda = Portc.0
Config Scl = Portc.1
Config I2cdelay = 1
Config Kbd = Porta
Declare Sub Readeepromserial
Const Eewrite = 160
Const Eeread = 161
Dim A As Byte
dim b as Byte
Dim Address As word
Lcd "Start Reading..."
address = &h0000
while address < &h7FFF
      a = high(Address)
      print "a:" ; hex(a)
      b = low(address)
      print "b:" ; hex(B)
      Call Readeepromserial
      print address
wend
End                                    'end program

Sub Readeepromserial
      I2cstart
      I2cwbyte Eewrite
      I2cwbyte a
      i2cwbyte b
      I2cstart
      I2cwbyte Eeread
      I2crbyte A , Nack
      I2cstop
      print a
   Incr Address
End Sub Readeepromserial
```



شکل ۸-۲۹ مدار بسته شده برای برنامه ارتباط با حافظه سریال

۱۶-۸ مقایسه‌کننده آنالوگ و CAPTURE تایمر یک

همانطور که می‌دانید خروجی مقایسه‌کننده مستقیماً به تریگر CAPTURE تایمر یک متصل است و خروجی مقایسه‌کننده می‌تواند بعنوان تریگر CAPTURE استفاده شود. در مثال زیر CAPTURE در لبه پایین رونده فعال شده است بدین معنی که اگر خروجی مقایسه‌کننده لبه پایین رونده داشته باشد ، وقفه CAPTURE اجرا می‌شود در صورتی که وقفه مقایسه‌کننده آنالوگ در حالت TOGGLE فعال می‌شود. در این برنامه SERIALOUT پیکره‌بندی شده است و شما می توانید از محیط TERMINAL EMULATOR برای اجرای برنامه استفاده نمایید.

زمانی که خروجی مقایسه‌کننده آنالوگ معکوس شود زیربرنامه Analogcominit اجرا می‌شود ولی زمانی که خروجی مقایسه‌کننده آنالوگ یک خروجی با لبه پایین رونده داشته باشد ، CAPTURE تایمر یک تریگ شده و زیربرنامه Capturetimer1 اجرا می‌شود.

- **برنامه**

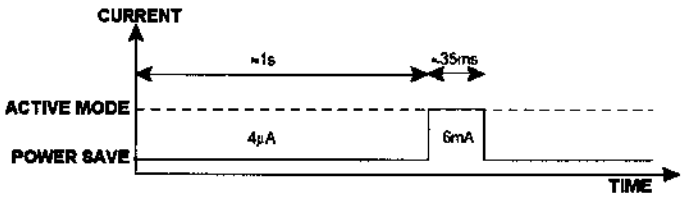
```
$regfile = "m32def.dat"
$crystal = 8000000
Config Aci = On , Compare = On , Trigger = Toggle
Config Timer1 = Timer , Prescale = 1024 , Capture Edge = Falling
Config Serialout = Buffered , Size = 10
$baud = 9600
Enable Interrupts
Enable Timer1
Enable Icp1
Enable Aci
On Icp1 Capturetimer1
On Aci Analogcominit
Start Timer1
Do
Loop
End                         'end program

Capturetimer1:
Print "timer1" ; Timer1
Return

Analogcominit:
Print "on aci"
Return
```

۱۷-۸ RTC (REAL TIME CLOCK)

تایمر/کانتر دو یا صفر در تعدادی از میکروهای AVR می‌تواند به صورت آسنکرون کار کند و بعنوان ساعت وقت واقعی یا RTC زمان و تقویم را حتی در حالت POWER-SAVE تنظیم کند. در این حالت کریستالی جدا از کریستال میکرو به مقدار 32.768KHZ در پایه‌های TOSC1 و TOSC2 برای تامین کلاک تایمر قرار می‌گیرد. در صورتی که PRESCALE=128 برای تایمر در نظر گرفته شود تایمر پس از 1s سریز می‌شود و برنامه RTC در زیربرنامه وقفه زمان را به روز می‌کند. برای کاهش مصرف ، میکرو در حالت POWER - SAVE قرار گرفته و پس از یک شدن پرچم سریزی تایمر از مُد POWER SAVE

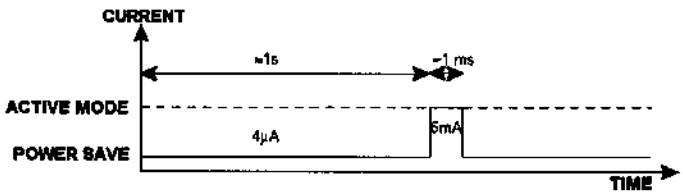
به حالت ACTIVE رفته و ISR مربوطه را انجام داده و سپس به حالت POWER – SAVE برمی گردد.

تمام جریان مصرفی در یک ثانیه برای کریستال بین پایه‌های XTAL 1,2 با زمان STARTUP=35ms برابر است با :



=(1sec * 4uA) + (35ms * 6mA) = 4uAs + 210uAs = 214uAs

تمام جریان مصرفی در یک ثانیه برای نوسانگر سرامیکی (CERAMIC RESONATOR) بین پایه‌های XTAL 1,2 با زمان STARTUP=0.5ms برابر است با :



= (1 sec * 4 uA) + (1 ms * 6 mA) = 4 uAs + 6 uAs = 10 uAs

جدول ۸–۶ مقدار جریان مصرفی در مُدهای مختلف برای میکروکنترلرهای AVR

MODE	TYPICAL	MAX
Active 4 MHz, 3.3 VCC	4 mA	6.0 mA
Idle 4 MHz, 3.3 VCC	1.8 mA	2.0 mA
Power Down 4 MHz, 3.3 VCC	<1.0 uA	2.0 uA
Power Save 4 MHz, 3.3 VCC	<4.0 uA	6.0 uA

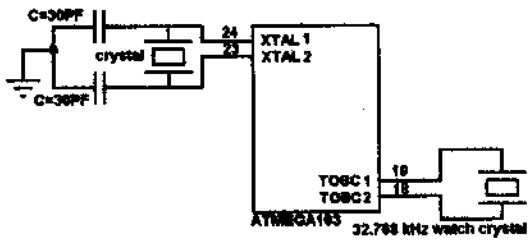
زمانهای مختلفی که می‌توان با کریستال 32768HZ توسط تایمر صفر یا دو ساخت در جدول ۸–۷ آمده است.

جدول۸–۷ انتخاب کلاک تایمر/کانتر 0 یا 2 به ازاء CK2,0=32,768 HZ در مُد آسنکرون

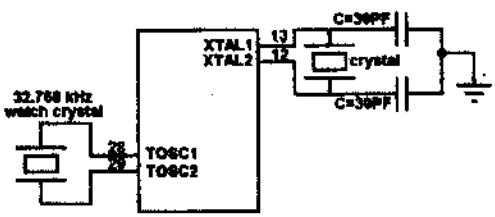
[Note: CK2,0 = 32,768 Hz]

DESCRIPTION	OVERFLOW PERIOD
STOP , TIMER/COUNTER IS STOPED	-
CK2,0	1/64 S
CK2,0 /8	1/8 S
CK2,0 /32	1/4 S
CK2,0 /64	1/2 S
CK2,0 /128	1 S
CK2,0 /256	2 S
CK2,0 /1024	8 S

طرز اتصال کریستال ساعت به پایه‌های TOSC در شکل‌های ۸-۳۰ و ۸-۳۱ آمده است.



شکل ۸-۳۰ اتصال کریستال ساعت به پایه های TOSC1,2 برای ATMEGA103



شکل ۸-۳۱ اتصال کریستال ساعت به پایه های TOSC1,2 در مُد آسنکرون تایمر/کانتر دو

در برنامه زیر از CONFIG CLOCK استفاده شده است. این پیکره‌بندی توسط BASCOM برای ساده‌تر کردن کار با تایمر در مُد آسنکرون طراحی شده است و برای تایمر دو در AT90S8535 و تایمر صفر در ATMEGA103 قابل اجرا است. این دستور خودکار، تایمر را در مُد آسنکرون پیکره‌بندی کرده و زمان 1s را تولید می‌کند.

Config Clock = Soft [, Gosub = Sectic]

بعد از سرریزی تایمر (1s) زیربرنامه اختیاری Sectic اجرا می‌شود. نوشتن و استفاده کردن از این برچسب اختیاری است. زمانی که از این پیکره‌بندی استفاده می‌نمایید متغییرهای TIME$ (hh:mm:ss) و DATE$ (mm:dd:yy) به ترتیب ساعت و تقویم به روز شده را در خود جای می‌دهند. به طور مثال دستور LCD TIME$ ساعت به روز شده را بر روی LCD نمایش خواهد داد. همچنین می‌توان مقدار اولیه‌ای را به متغییرهای DATE$ و TIME$ نسبت داد.

- **برنامه**

```
$regfile = "8535def.dat"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Pinb.4 , Db5 = Pinb.5 , Db6 = Pinb.6 , Db7 =_
Pinb.7 , Rs = Pinb.2 , E = Pinb.3
Config Clock = Soft , Gosub = Sectic
Enable Interrupts
Date$ = "09/09/82"          'initial date
Time$ = "11:38:56"          'initial time
Do
'you can use powersave here
```

```
Loop
End
'$sim                          'end program
Sectic:                        'when 1sec expire
Home
Lcd Time$
Locate 2 , 1                   'or   home L
Lcd Date$
Return
```

برنامه فوق را می‌توان به صورت زیر نیز نوشت :

- **برنامه**

```
$regfile = "8535def.dat"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Pinb.4 , Db5 = Pinb.5 , Db6 = Pinb.6 , Db7 =_
Pinb.7 , Rs = Pinb.2 , E = Pinb.3
Config Clock = Soft
Enable Interrupts
Date$ = "09/09/82"             'initial date
Time$ = "11:38:56"             'initial time
Do
Home
Lcd Time$
Locate 2 , 1                   'or   home L
Lcd Date$
Loop
End
'$sim
```

برنامه‌های فوق را می‌توان با تحلیل‌گر داخلی BASCOM بررسی و تحلیل کرد.

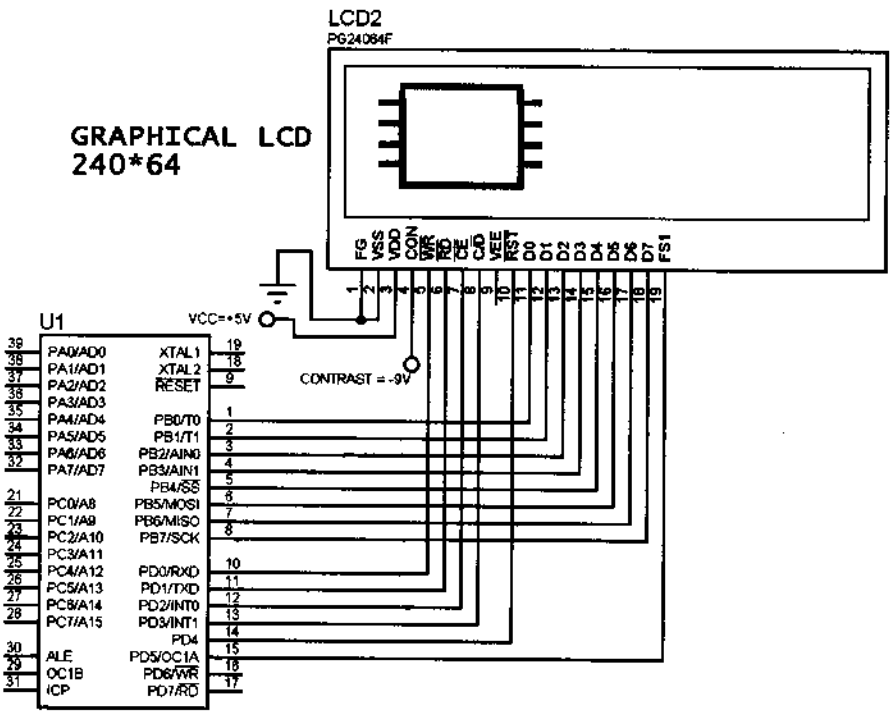
۱۸-۸ LCD گرافیکی با چیپ T6963

در محیط BASCOM ، LCD گرافیکی را پیکره‌بندی می‌کنیم. پیکره‌بندی، دستورات و طرز اتصال پایه‌های LCD به میکرو در فصل ششم آمده است. در این برنامه عکسی با پسوند BMP.* در محیط PAINT ویندوز کشیده شده است و سپس با باز کردن محیط GRAPHIC CONVERTOR ،توسط کلید LOAD عکس کشیده شده را به این محیط وارد می‌کنیم. سپس آن را با فرمت و نام BASCOM.BGF در کنار برنامه اصلی توسط دکمه SAVE ذخیره می‌کنیم. توسط دستور SHOWPIC با تعیین محل نمایش (0,0) ، عکس از برچسب GLCD فراخوانی شده و بر روی LCD نمایش داده می‌شود. پس از گذشت 1s ، با دستور CLS GRAPH عکس کشیده شده پاک می‌شود. با بردن مکان‌نما به موقعیت (1,1) متن MCS ELECTRONIC توسط دستور LCD نمایش داده می‌شود و بعد از 1s متن نوشته شده را با دستور CLS TEXT پاک می‌شود. توسط دستور CIRCLE دایره‌ای با قطر 10 پیکسل به رنگ مشکی در موقعیت (60,30) می‌شود. در این برنامه از LCD با اندازه 240*64 پیکسل استفاده شده است. شما می‌توانید از نوع‌های دیگر نیز استفاده نمایید.شماتیک مدار این پروژه در شکل ۸-۳۲ آمده است.

- **برنامه**

```
$regfile = "8515DEF.DAT"
$crystal = 8000000
```

```
Config Graphlcd = 240 * 64 , Dataport = Portb , Controlport = Portd ,_
Ce = 2 , Cd = 3 , Wr = 0 , Rd = 1 , Reset = 4 , Fs = 5 , Mode = 8
Cursor Off
Showpic 0 , 0 , Glcd
Wait 1
Cls Graph
Locate 1 , 1
Lcd "MCS ELECTRONIC"
Wait 1
Cls Text
Circle(60 , 30) , 10 , 255
End                              'end program
Glcd:
$bgf    "BASCOM.BGF"
```



شکل ۸-۳۲ مدار بسته شده برای برنامه LCD گرافیکی

۸-۱۹ کار با محیط TERMINAL EMULATOR

کار با محیط TERMINAL EMULATOR توسط UART سخت‌افزاری

محیط TERMINAL در BASCOM محیط مناسبی برای نمایش داده ارسالی و دریافتی از پورت سریال است. زمانی که شما از دستور PRINT در برنامه خود استفاده می‌کنید، از این محیط می‌توانید برای نمایش داده ارسالی به کامپیوتر استفاده نمایید. این محیط تا حدی شبیه HYPER TERMINAL در

ویندوز است. توسط دستورات INPUT می‌توان داده تایپ شده در محیط TERMINAL را به میکرو ارسال کرد. برنامه زیر نحوه کار با محیط TERMINAL را نشان می‌دهد. زمانی که برنامه زیر را در میکرو برنامه‌ریزی کردید، پس از ارتباط سریال سخت‌افزاری بین میکرو و PC محیط TERMINAL را با CTRL+T بالا بیاورید و برنامه را از منوی FILE ، UPLOAD کنید. بعد از ری‌ست میکرو، رشته INPUT PASSWORD در محیط ظاهر می‌گردد. شما می‌توانید رشته‌ای را تا 10 کاراکتر تایپ کنید. رشته تایپ شده در متغییر S قرار می‌گیرد. در صورتی که رشته وارد شده BASCOM باشد ، جمله VALID PASSWORD در محیط نوشته می‌شود و برنامه نام شما را درخواست می‌کند در غیر این صورت یعنی زمانی که PASSWORD وارد شده غیرمعتبر باشد برنامه با جملات END PROGRAM و INVALID PASSWORD پایان می‌یابد.

- **برنامه**

```
$regfile = "m32DEF.dat"
$crystal = 8000000              '11059200
$baud = 9600
Dim S As String * 10

   Do
      Input "INPUT PASSWORD  " , S
      If S <> "BASCOM" Then
         Print "INVALIDE PASSWORD"
         Exit Do
      End If

      Print "VALID PASSWORD"
      Input "NAME?" , S
      Print "HELLO " ; S
      Exit Do
   Loop

Print "PROGRAM END"
End                         'end program
```

پیوست

الف

خطاهای AVR

خطاهای میکروکنترلرهای AVR

AVR® ATMEL®

MCUS ERRORS

خطاهای میکروهای AVR در ویرایش‌های مختلف

AT90S1200/A

Errata
Rev. F
- **Reset During EEPROM Write**
- **Serial Programming at Voltages below 2.9 Volts**

2. Reset During EEPROM Write
If reset is activated during EEPROM write the result is not what should be expected. The EEPROM write cycle completes as normal, but the address registers are reset to 0. The result is that both the address written and address 0 in theEEPROM can be corrupted.
Problem Fix/Workaround
Avoid using address 0 for storage, unless you can guarantee that you will not get a reset during EEPROM write.
1. Serial Programming at Voltages below 2.9 Volts
At voltages below 2.9 Volts, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 2.9 Volts or higher during in-system programming.

AT90S2313

Rev. B
Errata
- **Lock Bits at High VCC**
- **Reset During EEPROM Write**
- **Serial Programming at Voltages Below 2.9 Volts**
- **UART Loses Synchronisation if RDX Line is Low when UART Receive Disabled**

4. Lock Bits at High VCC
On some devices, the lock bits will not erase at high VCC. In this situation, it will not be possible to reprogram the devices when the lock bits are set.
Problem Fix/Workaround
Lower VCC below 4.0V before you perform a chip-erase. Then the device will unlock, and it will be possible to reprogram the device at any VCC.
3. Reset During EEPROM Write
If reset is activated during EEPROM write the result is not what should be expected. The EEPROM write cycle completes as normal, but the address registers are reset to 0. The result is that both the address written and address 0 in the EEPROM can be corrupted.
Problem Fix/Workaround
Avoid using address 0 for storage, unless you can guarantee that you will not get a reset during EEPROM write.
2. Serial Programming at Voltages Below 2.9 Volts
At voltages below 2.9 Volts, serial programming might fail.
Problem Fix/Workaround
Keep VCC above 2.9 Volts during in-system programming.
1. UART Loses Synchronization if RXD Line is Low when UART Receive Disabled.
The UART will detect a UART start-bit and start reception even if the UART is not enabled. If this occurs, the first byte after re-enabling the UART will be corrupted.
Problem Fix/Workaround
Make sure that the RX line is high at startup and when the UART is disabled. An external RS232 level converter keeps the line high during start-up.

AT90S/LS2323

Rev. F Errata
- **Clearing Lockbits at High VCC or Temperature**
- **Wrong Latching of FSTRT Fuse**
- **Wrong Clearing of XTRF in MCUSR**
- **Reset during EEPROM Write**
- **Verifying EEPROM in System**
- **Serial Programming at Voltages below 3.0 Volts**

6. Clearing Lockbits at High VCC or Temperature
If the temperature is too high, and/or the programming voltage is too high, theclearing of lockbits might fail.
Problem Fix/Workaround
Keep VCC below 5.0 volts at room temperature when performing a chip erase.

5. Wrong Latching of FSTRT fuse
If VCC goes below GND and then up to the operating voltage, the FSTRT fuse can be read as unprogrammed even if it is programmed. The result of this is that the device uses the long start-up period instead of the short.
Problem Fix/Workaround
Avoid that VCC goes below GND. If the device has been started with the FSTRT fuse read wrong, it can be restarted in the correct mode again by taking VCC up to the operating voltage, then below 0.5V and then up again.
4. Wrong Clearing of XTRF in MCUSR
The XTRF flag in MCUSR will be cleared when clearing the PORF flag. The flag does not get cleared by writing a "0" to it.
Problem Fix/Workaround
Finish the test of both flags before clearing any of them. Clear both flags simultaneously by writing "0" to both PORF and XTRF in MCUCR.
3. Reset during EEPROM Write
If reset is activated during EEPROM write the result is not what should be expected. The EEPROM write cycle completes as normal, but the address registers are reset to "0". The result is that both the address written and address 0 in the EEPROM can be corrupted.
Problem Fix/Workaround
Avoid using address 0 for storage unless you can guarantee that you will not get a reset during EEPROM write.
2. Verifying EEPROM in System
EEPROM verify in In-System Programming mode cannot operate with maximum clock frequency. This is independent of the SPI clock frequency.
Problem Fix/Workaround
- Reduce the clock speed, or avoid using the EEPROM verify feature.
1. Serial Programming at Voltage below 3.0 Volts
At voltages below 3.0 volts, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 3.0 volts or higher during in-system programming.

AT90S/LS2343

Rev. F Errata

- **Clearing Lockbits at High VCC or temperature**
- **Wrong Clearing of XTRF in MCUSR**
- **Reset During EEPROM Write**
- **Verifying EEPROM in System**
- **Serial Programming at Voltages below 3.0 Volts**
- **High ICC in Power Down with External Clock Running**
- **Wrong Latching of RCEN Fuse**

7. Clearing Lockbits at High VCC or Temperature
If the temperature is too high, and/or the programming voltage is too high,clearing of lockbits might fail.
Problem Fix/Workaround
Keep VCC below 5.0 volts at room temperature when performing a chip erase.
6. Wrong Clearing of XTRF in MCUSR
The XTRF flag in MCUSR will be cleared when clearing the PORF flag. The does not get cleared by writing a "0" to it.
Problem Fix/Workaround
Finish the test of both flags before clearing any of them. Clear both flags simultaneously by writing 0 to both PORF and XTRF in MCUCR.
5. Reset During EEPROM Write
If reset is activated during EEPROM write the result is not what should expected. The EEPROM write cycle completes as normal, but the address registers are reset to 0. The result is that both the address written and address 0 in EEPROM can be corrupted.
Problem Fix/Workaround
Avoid using address 0 for storage, unless you can guarantee that you will not a reset during EEPROM write.
4. Verifying EEPROM in System
EEPROM verify in In-System Programming mode cannot operate with maximum clock frequency. This is independent of the SPI clock frequency.
Problem Fix/Workaround
Reduce the clock speed, or avoid using the EEPROM verify feature.
3. Serial Programming at Voltages below 3.0 Volts
At voltages below 3.0 volts, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 3.0 volts or higher during in-system programming.
2. High ICC in Power Down with External Clock Running
When the external clock is running while the device is in power down, the powerconsumption will be higher that specified.
Problem Fix/Workaround
Stop the external clock while the device is in power down.

AT90S/LS4433

Rev. C
Errata

- **Bandgap Reference Stabilizing Time**
- **Brown-out Detection Level**
- **Serial Programming at Voltages below 2.9V**
- **UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled**

4. Bandgap Reference Stabilizing Time
The time for the internal voltage reference for the Analog Comparator to stabilize is longer than specified. The stabilizing period starts after the bandgap reference has been selected, and can go on for as much as 10 seconds.
Problem Fix/Workaround
The Band-gap reference will be stable immediately if the internal Brown-out Detector is enabled.
3. Brown-out Detection Level
The Brown-out Detection level can increase when there is heavy I/O-activity on the device. The increase can be significant when some of the I/O pins are driving heavy loads.
Problem Fix/Workaround
Select a VCC well above the Brown-out Detection level.
Avoid loading I/O ports with high capacitive or resistive loads.
2. Serial Programming at Voltages below 2.9V
At voltages below 2.9V, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 2.9V or higher during In-System Programming.
1. UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled
The UART will detect a UART start bit and start reception even if the UART is not enabled. If this occurs, the first byte after re-enabling the UART will be corrupted.
Problem Fix/Workaround
Make sure that the RX line is high at start-up and when the UART is disabled. An external RS232-level converter keeps the line high during start-up.

AT90S/LS4434

Rev. D
Errata

- **Incorrect Channels Change in Free Running Mode**
- **32 kHz Oscillator may Fail at Higher Voltages**
- **Incorrect Start-up Time**
- **Lock Bits at High VCC and Temperature**
- **Error in Half Carry Flag**
- **Error in Writing Reset Status Bits**
- **Wake-up from Sleep Executes Instructions before the Interrupt is Serviced**
- **The SPI can Send Wrong Byte**
- **Output Compare Output Value Corrupted by Writing to Port**
- **Serial Programming at Voltages below 3.0V**
- **Wake-up from Power Save without Global Interrupt Enabled**

12. Incorrect Channel Changes in Free Running Mode
If the ADC operates in Free Running Mode and channels are changed by writing to ADMUX shortly after the ADC Interrupt Flag (ADIF in ADCSR) is set, the new setting in ADMUX may affect the ongoing conversion.
Problem Fix/Workaround
Use Single Conversion Mode when scanning channels, or avoid changing ADMUX until at least 0.5 ADC clock cycles after ADIF goes high.
11. 32 kHz Oscillator may Fail at Higher Voltages
When using an external 32 kHz crystal as asynchronous clock source for Timer2, the timer may count incorrectly at voltages above 4.0V.
Problem Fix/Workaround
Keep the supply voltage below 4.0V when clocking Timer2 from an external crystal.
10. Incorrect Start-up Time
When the FSTRT fuse is programmed, the start-up time from reset may still be 16 ms instead of 1 ms.
Problem Fix/Workaround
Leave the FSTRT fuse unprogrammed and design the system to allow a start-up time of 16 ms.
9. Lock Bits at High VCC and Temperature
On some devices, the lock bits will not erase at high VCC and temperature. In this situation, it will not be possible to reprogram the devices when the lock bits are set.
Problem Fix/Workaround
Lower VCC below 4.0V at room temperature before you perform a chip-erase. Then the device will unlock, and it will be possible to reprogram the device at any VCC.

8. Error in Half Carry Flag
The half carry flag is undefined after executing the commands "ror", "asr" and "lsr".
Problem Fix/Workaround
Do not use the half carry flag value after executing the above instructions.
7. Error in Writing Reset Status Bits
The EXTRF flag in MCUSR will be cleared when clearing the PORF-flag. The flag does not get cleared by writing a "0" to it.
Problem Fix/Workaround
Finish the test of both flags before clearing any of them. Clear both flags simultaneously by writing "0" to both PORF and EXTRF in MCUCR.
6. Wake-up from Sleep Executes Instructions before the Interrupt is Serviced
When waking up from power save, some instructions are executed before the interrupt is called. If the device is woken up by an external interrupt, 2 instruction cycles are executed. If it is woken up by the asynchronous timer, 3 instructions are executed before the interrupt.
Problem Fix/Workaround
Make sure that the first two or three instructions following sleep are not dependent on the executed interrupt.
5. The SPI can Send Wrong Byte If the SPI is in Master mode, it will restart the old transfer if new data is written on the same clock edge as the previous transfer is finished.
Problem Fix/Workaround
When writing to the SPI, first wait until it is ready, then write the byte to transmit.
4. Output Compare Output Value Corrupted by Writing to Port
When writing to the PORTD I/O location, the OC1A and OC1B output compare values will assume the values written to bits 5 and 4, respectively. This means that even when writing to another bit in the same port register (such as a read-modify-write to another pin in the same port), the output compare values will be affected. Effectively, if the output compare function is used, the other pins in the same port cannot be changed, unless the intention is to write the output compare values simultaneously.
Problem Fix/Workaround
Avoid updating the other port bits when using the output compare function.
3. Serial Programming at Voltages below 3.0V
At voltages below 3.0V, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 3.0V or higher dur ing In-System Programming.
2. Wake-up from Power Save without Global Interrupt Enabled
When an asynchronous timer interrupt is used to wake up the part from power save, the part will wake up even if global interrupts are disabled.
Problem Fix/Workaround
No workaround necessary.
1. UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled
The UART will detect a UART start bit and start reception even if the UART is not enabled. If this occurs, the first byte after re-enabling the UART will be corrupted.
Problem Fix/Workaround
Make sure that the RX line is high at start-up and when the UART is disabled. An external RS232-level converter keeps the line high during start-up.

AT90S/LS4434

Rev. F Errata

- **Incorrect Channel Change in Free Running Mode**
- **32 kHz Oscillator may Fail at Higher Voltages**
- **Error in Half Carry Flag**
- **Error in Writing Reset Status Bits**
- **Wake-up from Sleep Executes Instructions before the Interrupt is Serviced**
- **The SPI can Send Wrong Byte**
- **Serial Programming at Voltages below 3.0 V**
- **Wake-up from Power Save without Global Interrupt Enabled**
- **UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled**

9. Incorrect Channel Change in Free Running Mode
If the ADC operates in Free Running Mode and channels are changed by writing to ADMUX shortly after the ADC Interrupt Flag (ADIF in ADCSR) is set, the new setting in ADMUX may affect the ongoing conversion.
Problem Fix/Workaround
Use Single Conversion Mode when scanning channels, or avoid changing ADMUX until at least 0.5 ADC clock cycles after ADIF goes high.
8. 32 kHz Oscillator may Fail at Higher Voltages
When using an external 32 kHz crystal as asynchronous clock source for Timer2, the timer may count incorrectly at voltages above 4.0V.
Problem Fix/Workaround
Keep the supply voltage below 4.0V when clocking Timer2 from an external crystal.
7. Error in Half Carry Flag
The half carry flag is undefined after executing the commands "ror", "asr" and "lsr".

Problem Fix/Workaround
Do not use the half carry flag value after executing the above instructions.
6. Error in Writing Reset Status Bits
The EXTRF flag in MCUSR will be cleared when clearing the PORF-flag. The flag does not get cleared by writing a "0" to it.
Problem Fix/Workaround
Finish the test of both flags before clearing any of them. Clear both flags simultaneously by writing "0" to both PORF and EXTRF in MCUCR.
5. Wake-up from Sleep Executes Instructions before the Interrupt is Serviced
When waking up from power save, some instructions are executed before the interrupt is called. If the device is woken up by an external interrupt, 2 instruction cycles are executed. If it is woken up by the asynchronous timer, 3 instructions are executed before the interrupt.
Problem Fix/Workaround
Make sure that the first two or three instructions following sleep are not dependent on the executed interrupt.
4. The SPI can Send Wrong Byte
If the SPI is in Master mode, it will restart the old transfer if new data is written on the same clock edge as the previous transfer is finished.
Problem Fix/Workaround
When writing to the SPI, first wait until it is ready, then write the byte to transmit.
3. Serial Programming at Voltages below 3.0V
At voltages below 3.0V, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 3.0V or higher during In-System Programming.
2. Wake-up from Power Save without Global Interrupt Enabled
When an asynchronous timer interrupt is used to wake up the part from power save, the part will wake up evenif global interrupts are disabled.
Problem Fix/Workaround
No workaround necessary.
1. UART Loses Synchronization if RXD Line is Low
when UART Receive is Disabled The UART will detect a UART start bit and start reception
even if the UART is not enabled. If this occurs, the first byte after re-enabling the UART will be corrupted.
Problem Fix/Workaround
Make sure that the RX line is high at start-up and when the UART is disabled. An external RS232-level converter keeps the line high during start-up.

AT90S/LS8535

Rev. D
Errata (All Date Codes)
- **Incorrect Channels Change in Free Running Mode**
- **32 kHz Oscillator may Fail at Higher Voltages**
- **Incorrect Start-up Time**
- **Lock Bits at High VCC and Temperature**
- **Error in Half Carry Flag**
- **Error in Writing Reset Status Bits**
- **Wake-up from Sleep Executes Instructions before the Interrupt is Serviced**
- **The SPI can Send Wrong Byte**
- **Output Compare Output Value Corrupted by Writing to Port**
- **Serial Programming at Voltages below 3.0V**
- **Wake-up from Power Save without Global Interrupt Enabled**
- **UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled**

Errata (Date Codes before 9836)
- **High Current Consumption at High VCC**
- **Leakage Current on Tri-stated I/O Pins**
- **32 kHz Oscillator**

These errata apply for all rev D devices.

15. Incorrect Channel Changes in Free Running Mode
If the ADC operates in Free Running Mode and channels are changed by writing to ADMUX shortly after the ADC Interrupt Flag (ADIF in ADCSR) is set, the new setting in ADMUX may affect the ongoing conversion.
Problem Fix/Workaround
Use Single Conversion Mode when scanning channels, or avoid changing ADMUX until at least 0.5 ADC clock cycles after ADIF goes high.
14. 32 kHz Oscillator may Fail at Higher Voltages
When using an external 32 kHz crystal as asynchronous clock source for Timer2, the timer may count incorrectly at voltages above 4.0V.
Problem Fix/Workaround
Keep the supply voltage below 4.0V when clocking Timer2 from an external crystal.

13. Incorrect Start-up Time
When the FSTRT fuse is programmed, the start-up time from reset may still be 16 ms instead of 1 ms.
Problem Fix/Workaround
Leave the FSTRT fuse unprogrammed and design the system to allow a start-up time of 16 ms.
12. Lock Bits at High VCC and Temperature
On some devices, the lock bits will not erase at high VCC and temperature. In this situation, it will not be possible to reprogram the devices when the lock bits are set.
Problem Fix/Workaround
Lower VCC below 4.0V at room temperature before you perform a chip-erase. Then the device will unlock, and it will be possible to reprogram the device at anyVCC.
11. Error in Half Carry Flag
The half carry flag is undefined after executing the commands "ror", "asr" and "lsr".
Problem Fix/Workaround
Do not use the half carry flag value after executing the above instructions.
10. Error in Writing Reset Status Bits
The EXTRF flag in MCUSR will be cleared when clearing the PORF-flag. The flag does not get cleared by writing a "0" to it.
Problem Fix/Workaround
Finish the test of both flags before clearing any of them. Clear both flags simultaneously by writing "0" to both PORF and EXTRF in MCUCR.
9. Wake-up from Sleep Executes Instructions before the Interrupt is Serviced
When waking up from power save, some instructions are executed before the interrupt is called. If the device is woken up by an external interrupt, 2 instruction cycles are executed. If it is woken up by the asynchronous timer, 3 instructions are executed before the interrupt.
Problem Fix/Workaround
Make sure that the first two or three instructions following sleep are not dependent on the executed interrupt.
8. The SPI can Send Wrong Byte
If the SPI is in Master mode, it will restart the old transfer if new data is written on the same clock edge as the previous transfer is finished.
Problem Fix/Workaround
When writing to the SPI, first wait until it is ready, then write the byte to transmit.
7. Output Compare Output Value Corrupted by Writing to Port
When writing to the PORTD I/O location, the OC1A and OC1B output compare values will assume the values written to bits 5 and 4, respectively. This means that even when writing to another bit in the same port register (such as a read-modify-write to another pin in the same port), the output compare values will be affected. Effectively, if the output compare function is used, the other pins in the same port cannot be changed, unless the intention is to write the output compare values simultaneously.
Problem Fix/Workaround
Avoid updating the other port bits when using the output compare function.
6. Serial Programming at Voltages below 3.0V
At voltages below 3.0V, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 3.0V or higher dur ing In-System Programming.
5. Wake-up from Power Save without Global Interrupt Enabled
When an asynchronous timer interrupt is used to wake up the part from power save, the part will wake up even if global interrupts are disabled.
Problem Fix/Workaround
No workaround necessary.
4. UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled
The UART will detect a UART start bit and start reception even if the UART is not enabled. If this occurs, the first byte after re-enabling the UART will be corrupted.
Problem Fix/Workaround
Make sure that the RX line is high at start-up and when the UART is disabled. An external RS232-level converter keeps the line high during start-up. In addition to the above, these errata apply for devices with date code marking before 9836.
3. High Current Consumption at High VCC
Some of the early samples have higher current consumption than specified. The current consumption in power-down/power save mode is 100 to 500 µA at 6V, rather than the specified 50 µA. The current consumption increases exponentially with supply voltage, and is strongly varying from sample to sample.
Problem Fix/Workaround
Use devices with date codes later than 9836.
2. Leakage Current on Tri-stated I/O Pins
On some of the early samples tri-stated I/O pins may source up to 20 µA and sink up to 6 µA at 6V supply voltage. This means that input pins will effectively have an input impedance of down to 300K. This may cause an unfortunate input offset voltage, particularly noticeable for the ADC and analog comparator pins.
Problem Fix/Workaround
Drivers for the analog and digital input signals to the MCU must be designed to drive a load of 300Kper pin. Or use devices with date codes later than 9836.

AT90S/LS8535

Rev. E
Errata

- **Incorrect Channel Change in Free Running Mode**
- **32 kHz Oscillator may Fail at Higher Voltages**
- **Error in Half Carry Flag**
- **Error in Writing Reset Status Bits**
- **Wake-up from Sleep Executes Instructions before the Interrupt is Serviced**
- **The SPI can Send Wrong Byte**
- **Serial Programming at Voltages below 3.0V**
- **Wake-up from Power Save without Global Interrupt Enabled**
- **UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled**

9. Incorrect Channel Change in Free Running Mode
If the ADC operates in Free Running Mode and channels are changed by writing to ADMUX shortly after the ADC Interrupt Flag (ADIF in ADCSR) is set, the new setting in ADMUX may affect the ongoing conversion.
Problem Fix/Workaround
Use Single Conversion Mode when scanning channels, or avoid changing ADMUX until at least 0.5 ADC clock cycles after ADIF goes high.
8. 32 kHz Oscillator may Fail at Higher Voltages
When using an external 32 kHz crystal as asynchronous clock source for Timer2, the timer may count incorrectly at voltages above 4.0V.
Problem Fix/Workaround
Keep the supply voltage below 4.0V when clocking Timer2 from an external crystal.
7. Error in Half Carry Flag
The half carry flag is undefined after executing the commands "ror", "asr" and "lsr".
Problem Fix/Workaround
Do not use the half carry flag value after executing the above instructions.
6. Error in Writing Reset Status Bits
The EXTRF flag in MCUSR will be cleared when clearing the PORF-flag. The flag does not get cleared by writing a "0" to it.
Problem Fix/Workaround
Finish the test of both flags before clearing any of them. Clear both flags simultaneously by writing "0" to both PORF and EXTRF in MCUCR.
5. Wake-up from Sleep Executes Instructions before the Interrupt is Serviced
When waking up from power save, some instructions are executed before theinterrupt is called. If the device is woken up by an external interrupt, 2 instructioncycles are executed. If it is woken up by the asynchronous timer, 3 instructionsare executed before the interrupt.
Problem Fix/Workaround
Make sure that the first two or three instructions following sleep are not dependenton the executed interrupt.
4. The SPI can Send Wrong Byte
If the SPI is in Master mode, it will restart the old transfer if new data is written on the same clock edge as the previous transfer is finished.
Problem Fix/Workaround
When writing to the SPI, first wait until it is ready, then write the byte to transmit.
3. Serial Programming at Voltages below 3.0V
At voltages below 3.0V, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 3.0V or higher dur ing In-System Programming.
2. Wake-up from Power Save without Global Interrupt Enabled
When an asynchronous timer interrupt is used to wake up the part from power save, the part will wake up even if global interrupts are disabled.
Problem Fix/Workaround
No workaround necessary.
1. UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled
The UART will detect a UART start bit and start reception even if the UART is not enabled. If this occurs, the first byte after re-enabling the UART will be corrupted.
Problem Fix/Workaround
Make sure that the RX line is high at start-up and when the UART is disabled. An external RS232-level converter keeps the line high during start-up.

AT90S4414

Rev. A/B
Errata

- **The SPI can Send Wrong Byte**
- **Reset During EEPROM Write**
- **SPI Interrupt Flag can be Undefined After Reset**
- **Verifying EEPROM in System**

• Serial Programming at Voltages below 3.0 Volts
• Skip Instruction with Interrupts

6. The SPI can Send Wrong Byte
If the SPI is in master mode, it will restart the old transfer if new data is written on the same clock edge as the previous transfer is finished.
Problem Fix/Workaround
When writing to the SPI, first wait until it is ready, then write the byte to transmit.
5. Reset During EEPROM Write
If reset is activated during EEPROM write the result is not what should be expected. The EEPROM write cycle completes as normal, but the address registers are reset to 0. The result is that both the address written and address 0 in the EEPROM can be corrupted.
Problem Fix/Workaround
Avoid using address 0 for storage, unless you can guarantee that you will not get a reset during EEPROM write.
4. SPI Interrupt Flag can be Undefined After Reset
In certain cases when there are transitions on the SCK pin during reset, or the SCK pin is left unconnected, the start-up value of the SPI interrupt flag is be unknown. If the flag is not reset before enabling the SPI interrupt, a pending SPI interrupt may be executed.
Problem Fix/Workaround
Clear the SPI interrupt flag before enabling the interrupt.
3. Verifying EEPROM in System
EEPROM verify in In-System Programming mode cannot operate with maximum clock frequency. This is independent of the SPI clock frequency.
Problem Fix/Workaround
Reduce the clock speed, or avoid using the EEPROM verify feature.
2. Serial Programming av Voltaages below 3.0 Volts
At voltages below 3.0 Volts, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 3.0 Volts or higher during In-System Programming
1. Skip Instruction with Interrupts
A skip instruction (SBRS, SBRC, SBIS, SBIC, CPSE) that skips a two-word instruction needs three clock cycles. If an interrupt occurs during the first or second clock cycle of this skip-instruction, the return address will not be stored correctly on the stack. In this situation, the address of the second word in the twoword instruction is stored. This means that on return from interrupt, the second word of the two-word command will be decoded and executed as an instruction. The AT90S4414 has two two-word instructions: LDS and STS.
Note: This can only occur if all of the following conditions are true:
- A skip instruction is followed by a two-word instruction.
- The skip instruction is actually skipping the two-word instruction.
- Interrupts are enabled, and at least one interrupt source can generate an interrupt.
- An interrupt arrives in the first or second cycle of the skip instruction.
Note 2: This will only cause problems if the address of the following LDS or STS command points to an address beyond 400 Hex.
Problem Fix/Workaround
For C-programs, use the IAR compiler version 1.40b or later. The compiler will never generate the sequence. For assembly program, avoid skipping a two word instruction if interrupts are enabled.

AT90S8515

Rev. B
Errata
• Lock Bits at High VCC
• The SPI can Send Wrong Byte
• Reset during EEPROM Write
• SPI Interrupt Flag can be Undefined after Reset
• Serial Programming at Voltages below 3.0 Volts
• Skip Instruction with Interrupts

6. Lock Bits at High VCC
On some devices, the lock bits will not erase at high VCC and temperature. In this situation, it will not be possible to reprogram the devices when the lock bits are set. This errata strongly depends on temperature and should in most cases not cause any problems at room temperature.
Problem Fix/Workaround
Lower VCC below 4.0V before you perform a chip erase. Then the device will unlock at any temperature and it will be possible to reprogram the device at any VCC.
5. The SPI can Send Wrong Byte
If the SPI is in master mode, it will restart the old transfer if new data is written on the same clock edge as the previous transfer is finished.
Problem Fix/Workaround
When writing to the SPI, first wait until it is ready; then write the byte to transmit.

4. Reset during EEPROM Write
If reset is activated during EEPROM write, the result is not what should be expected. The EEPROM write cycle completes as normal, but the address registers are reset to 0. The result is that both the address written and address 0 in the EEPROM can be corrupted.
Problem Fix/Workaround
Avoid using address 0 for storage, unless you can guarantee that you will not get a reset during EEPROM write.
3. SPI Interrupt Flag can be Undefined after Reset
In certain cases when there are transitions on the SCK pin during reset, or the SCK pin is left unconnected, the start-up value of the SPI interrupt flag is unknown. If the flag is not reset before enabling the SPI interrupt, a pending SPI interrupt may be executed.
Problem Fix/Workaround
Clear the SPI interrupt flag before enabling the interrupt.
2. Serial Programming at Voltages below 3.0 Volts
At voltages below 3.0 volts, serial programming might fail.
Problem Fix/Workaround
Keep VCC at 3.0 volts or higher during in-system programming.
1. Skip Instruction with Interrupts
A skip instruction (SBRS, SBRC, SBIS, SBIC, CPSE) that skips a 2-word instruction needs three clock cycles. If an interrupt occurs during the first or second clock cycle of this skip instruction, the return address will not be stored correctly on the stack. In this situation, the address of the second word in the 2-word instruction is stored. This means that on return from interrupt, the second word of the 2-word command will be decoded and executed as an instruction. The AT90S4414 has two 2-word instructions: LDS and STS. Note 1: This can only occur if all of the following conditions are true:
- A skip instruction is followed by a 2-word instruction.
- The skip instruction is actually skipping the 2-word instruction.
- Interrupts are enabled, and at least one interrupt source can generate an interrupt.
- An interrupt arrives in the first or second cycle of the skip instruction.
Note 2: This will cause problems only if the address of the following LDS or STS command points to an address beyond 400 Hex.
Problem Fix/Workaround
For C-programs, use the IAR compiler version 1.40b or later. The compiler will never generate the sequence. For assembly program, avoid skipping a 2-word instruction if interrupts are enabled.

ATtiny11

Rev. B Errata
- **High-power Consumption in Power-down Mode**
- **Limited Voltage Range/Frequency Operation Range**

2. High-power Consumption in Power-down Mode
The power consumption in Power-down mode might be up to 500 μA at 4.0V operating voltage.
Problem Fix/Workaround
Use ATtiny11 revision C.
1. Limited Voltage Range/Frequency Range
The voltage range is limited to 2.7V to 4.0V. The maximum operating frequency is limited to 2 MHz in the entire range.
Problem Fix/Workaround
Use ATtiny 11 revision C.

ATtiny28

Rev. D
Errata
- **No Known Errors**

Errata
ATtiny2313 Rev B

- **Wrong values read after Erase Only operation**
- **Parallel Programming does not work**
- **Watchdog Timer Interrupt disabled**
- **EEPROM can not be written below 1.9 volts**

1. Wrong values read after Erase Only operation
At supply voltages below 2.7 V, an EEPROM location that is erased by the Erase Only operation may read as programmed (0x00).

Problem Fix/Workaround
If it is necessary to read an EEPROM location after Erase Only, use an Atomic Write operation with 0xFF as data in order to erase a location. In any case, the Write Only operation can be used as intended. Thus no special considerations are needed as long as the erased location is not read before it is programmed.

2. Parallel Programming does not work
Parallel Programming is not functioning correctly. Because of this, reprogramming of the device is impossible if one of the following modes are selected:
– In-System Programming disabled (SPIEN unprogrammed)
– Reset Disabled (RSTDISBL programmed)

Problem Fix/Workaround
Serial Programming is still working correctly. By avoiding the two modes above, the device can be reprogrammed serially.

3. Watchdog Timer Interrupt disabled
If the watchdog timer interrupt flag is not cleared before a new timeout occurs, the watchdog will be disabled, and the interrupt flag will automatically be cleared. This is only applicable in interrupt only mode. If the Watchdog is configured to reset the device in the watchdog time-out following an interrupt, the device works correctly.

Problem fix / Workaround
Make sure there is enough time to always service the first timeout event before a new watchdog timeout occurs. This is done by selecting a long enough time-out period.

4. EEPROM can not be written below 1.9 volts
Writing the EEPROM at VCC below 1.9 volts might fail.

Problem fix / Workaround
Do not write the EEPROM when VCC is below 1.9 volts.

Errata
ATtiny13 Rev. D

• EEPROM can not be written below 1.9 Volt

1. EEPROM can not be written below 1.9 Volt
Writing the EEPROM at VCC below 1.9 volts might fail.

Problem Fix/Workaround
Do not write the EEPROM when VCC is below 1.9 volts.

ATtiny13 Rev. B

• Wrong values read after Erase Only operation
• High Voltage Serial Programming Flash, EEPROM, Fuse and Lock Bits may fail
• Device may lock for further programming
• debugWIRE communication not blocked by lock-bits
• Watchdog Timer Interrupt disabled
• EEPROM can not be written below 1.9 Volt

1. Wrong values read after Erase Only operation
At supply voltages below 2.7 V, an EEPROM location that is erased by the Erase Only operation may read as programmed (0x00).

Problem Fix/Workaround
If it is necessary to read an EEPROM location after Erase Only, use an Atomic Write operation with 0xFF as data in order to erase a location. In any case, the Write Only operation can be used as intended. Thus no special considerations are needed as long as the erased location is not read before it is programmed.

2. High Voltage Serial Programming Flash, EEPROM, Fuse and Lock Bits may fail
Writing to any of these locations and bits may in some occasions fail.

Problem Fix/Workaround
After a writing has been initiated, always observe the RDY/BSY signal. If the writing should fail, rewrite until the RDY/BSY verifies a correct writing. This will be fixed in revision D.

3. Device may lock for further programming
Special combinations of fuse bits will lock the device for further programming effectively turning it into an OTP device. The following combinations of settings/fuse bits will cause this effect:
– 128 kHz internal oscillator (CKSEL[1..0] = 11), shortest start-up time (SUT[1..0] = 00), Debugwire enabled (DWEN = 0) or Reset disabled RSTDISBL = 0.

– 9.6 MHz internal oscillator (CKSEL[1..0] = 10), shortest start-up time (SUT[1..0] = 00), Debugwire enabled (DWEN = 0) or Reset disabled RSTDISBL = 0.
– 4.8 MHz internal oscillator (CKSEL[1..0] = 01), shortest start-up time (SUT[1..0] = 00), Debugwire enabled (DWEN = 0) or Reset disabled RSTDISBL = 0.

Problem fix/ Workaround
Avoid the above fuse combinations. Selecting longer start-up time will eliminate the problem.

4. debugWIRE communication not blocked by lock-bits
When debugWIRE on-chip debug is enabled (DWEN = 0), the contents of program memory and EEPROM data memory can be read even if the lock-bits are set to block further reading of the device.

Problem fix/ Workaround
Do not ship products with on-chip debug of the tiny13 enabled.

5. Watchdog Timer Interrupt disabled
If the watchdog timer interrupt flag is not cleared before a new timeout occurs, the watchdog will be disabled, and the interrupt flag will automatically be cleared. This is only applicable in interrupt only mode. If the Watchdog is configured to reset the device in the watchdog time-out following an interrupt, the device works correctly.

Problem fix / Workaround
Make sure there is enough time to always service the first timeout event before a new watchdog timeout occurs. This is done by selecting a long enough time-out period.

6. EEPROM can not be written below 1.9 Volt
Writing the EEPROM at VCC below 1.9 volts might fail.

Problem Fix/Workaround
Do not write the EEPROM when VCC is below 1.9 volts.

ATmega103(L)

Rev. L
Errata

- **Wake-up from Power Save Executes Instructions before Interrupt**
- **SPI can Send Wrong Byte**
- **Wrong Clearing of EXTRF in MCUSR**
- **Reset during EEPROM Write**
- **SPI Interrupt Flag can be Undefined after Reset**
- **Skip Instruction with Interrupts**
- **Signature Bytes**
- **Read Back Value during EEPROM Polling**
- **MISO Active during In-System Programming**
- **The ADC has no Free-running Mode**
- **UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled**

11. Wake-up from Power Save Executes Instructions before Interrupt
When waking up from power save, some instructions are executed before the interrupt is called. If the device is woken up by an external interrupt, 2 instruction cycles are executed. If it is woken up by the asynchronous timer, 3 instructions are executed before the interrupt.

Problem Fix/Workaround
Make sure that the first two or three instructions following sleep are not dependent on the executed interrupt.

10. The SPI can Send Wrong Byte
If the SPI is in Master mode, it will restart the old transfer if new data is written on the same clock edge as the previous transfer is finished.

Problem Fix/Workaround
When writing to the SPI, first wait until it is ready, then write the byte to transmit.

9. Wrong Clearing of EXTRF in MCUSR
The EXTRF flag in MCUSR will be cleared when clearing the PORF-flag. The flag does not get cleared by writing a "0" to it.

Problem Fix/Workaround
Finish the test of both flags before clearing any of them. Clear both flags simultaneously by writing "0" to both PORF and EXTRF in MCUCR.

8. Reset during EEPROM Write
If reset is activated during EEPROM write, the result is not what should be expected. The EEPROM write cycle completes as normal, but the address registers are reset to 0. The result is that both the address written and address 0 in the EEPROM can be corrupted.

Problem Fix/Workaround
Avoid using address 0 for storage, unless you can guarantee that you will not get a reset during EEPROM write.

7. SPI Interrupt Flag can be Undefined after Reset

In certain cases when there are transitions on the SCK pin during reset, or the SCK pin is left unconnected, the start-up value of the SPI interrupt flag is unknown. If the flag is not reset before enabling the SPI interrupt, a pending SPI interrupt may be executed.
Problem Fix/Workaround
Clear the SPI interrupt flag before enabling the interrupt.
6. Skip Instruction with Interrupts
A skip instruction (SBRS, SBRC, SBIS, SBIC, CPSE) that skips a two-word instruction needs three clock cycles. If an interrupt occurs during the first or second clock cycle of this skip instruction, the return address will not be stored correctly on the stack. In this situation, the address of the second word in the two-word instruction is stored. This means that on return from interrupt, the second word of the two-word command will be decoded and executed as an instruction. The ATmega103 has four two-word instructions: LDS, STS, JMP and CALL.
Notes: 1. This can only occur if all of the following conditions are true:
- A skip instruction is followed by a two-word instruction.
- The skip instruction is actually skipping the twoword instruction.
- Interrupts are enabled, and at least one interrupt source can generate an interrupt.
- An interrupt arrives in the first or second cycle of the skip instruction.
2. This will only cause problems if the address of the following LDS or STS command points to an address beyond 400 Hex.
Problem Fix/Workaround
For C-programs, use the IAR compiler version 1.40b or later. The compiler will never generate the sequence.
For assembly program, avoid skipping a two-word instruction if interrupts are enabled.
5. Signature Bytes
The signature bytes of the first few lots of the ATmega103 have been shipped with wrong signature bytes. Also in the datasheet, the wrong signature bytes have been given. The correct signature bytes are: $1E $97 $01.
Problem Fix/Workaround
Programmers must allow both $1E $97 $01 and $1E $01 $01 as valid signature bytes.
4. Read Back Value during EEPROM Polling
When a new EEPROM byte is being programmed into the EEPROM with In-System Programming, reading the address location being programmed will give the value P1 (see table 1) until the Auto-erase is finished. Then the value P2 will follow until programming is finished. At the time the device is ready for a new EEPROM byte, the programmed value will read correctly.
Note: This is only a problem for In-System Programmers.
Reading and writing the EEPROM during normal operation is not affected by this.
Problem Fix/Workaround
Programmers must allow both $80 and $7F as read back values if data polling is used for the EEPROM.
Polling will not work for either of the values P1 and P2, so when programming these values, the user will have to wait the prescribed time tWD_EEPROM before programming the next byte.
3. MISO Active during In-System Programming
During In-System Programming, the MISO line (pin 13) of the ATmega103 is active, although the UART pins are used for programming. If pin 13 is used as an input in the application, a collision may occur on this line.
Problem Fix/Workaround
- If the MISO pin is used as an input, make sure that there is a current-limiting resistor in series with the line.
- If the pin is used as an output, make sure that whatever is connected to the line can accept that the pin is toggling during programming.
2. The ADC has no Free-running Mode
Early versions of the ATmega603/103 datasheet described an ADC Free-running Mode. This mode is not available in this device, and bit number 5 in the ADCSR register must always be written as "0".
Problem Fix/Workaround
Use Single-conversion Mode and always use the latest revision of the datasheet.
1. UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled
The UART will detect a UART start bit and start reception even if the UART is not enabled. If this occurs, the first byte after re-enabling the UART will be corrupted.
Problem Fix/Workaround
Make sure that the RX line is high at start-up and when the UART is disabled. An external RS232-level converter keeps the line high during start-up.

ATmega8
ATmega8L

Erratas The revision letter in this section refers to the revision of the ATmega8 device.
ATmega8 Rev. D, E, and F
• CKOPT Does not Enable Internal Capacitors on XTALn/TOSCn Pins when 32 KHz Oscillator is Used to Clock the Asynchronous Timer/Counter2
1. CKOPT Does not Enable Internal Capacitors on XTALn/TOSCn Pins when
32 KHz Oscillator is Used to Clock the Asynchronous Timer/Counter2
When the internal RC Oscillator is used as the main clock source, it is possible to run the Timer/Counter2 asynchronously by connecting a 32 KHz Oscillator between

XTAL1/TOSC1 and XTAL2/TOSC2. But when the internal RC Oscillator is selected as the main clock source, the CKOPT Fuse does not control the internal capacitors on XTAL1/TOSC1 and XTAL2/TOSC2. As long as there are no capacitors connected to XTAL1/TOSC1 and XTAL2/TOSC2, safe operation of the Oscillator is not guaranteed.
Problem fix/Workaround
Use external capacitors in the range of 20 - 36 pF on XTAL1/TOSC1 and XTAL2/TOSC2. This will be fixed in ATmega8 Rev. G where the CKOPT Fuse will control internal capacitors also when internal RC Oscillator is selected as main clock source. For ATmega8 Rev. G, CKOPT = 0 (programmed) will enable the internal capacitors on XTAL1 and XTAL2. Customers who want compatibility between Rev. G and older revisions, must ensure that CKOPT is unprogrammed (CKOPT = 1).

ATmega8515(L)

Rev. B There are no errata for this revision of ATmega8515.

ATmega161

Rev. E
Errata
- **PWM not Phase Correct**
- **Increased Interrupt Latency**
- **Interrupt Return Fails when Stack Pointer Addresses the External Memory**
- **Writing UBBRH Affects both UART0 and UART1**
- **Store Program Memory Instruction May Fail**

5. PWM not Phase Correct
In phase correct PWM mode, a change from OCRx = TOP to anything less than TOP does not change the OCx output. This gives a phase error in the following period.
Problem Fix/Workaround
Make sure this issue is not harmful to the application.
4. Increased Interrupt Latency
In this device, some instructions are not interruptable, and will cause the interrupt latency to increase. The only practical problem concerns a loop followed by a twoword instruction while waiting for an interrupt. The loop may consist of a branch instruction or an absolute or relative jump back to itself like this:
loop: rjmp loop
<Two-word instruction>
In this case, a dead-lock situation arises.
Problem Fix/Workaround
In assembly, insert a nop instruction immediately after a loop to itself. The problem will normally be detected during development. In C, the only construct that will give this problem is an empty "for" loop: "for(;;)". Use "while(1)" or "do{} while (1)" to avoid the problem.
3. Interrupt Return Fails when Stack Pointer Addresses the External Memory
When Stack Pointer addresses external memory (SPH:SPL > $45F), returning from interrupt will fail. The program counter will be updated with a wrong value and thus the program flow will be corrupted.
Problem Fix/Workaround
Address the stack pointer to internal SRAM or disable interrupts while Stack Pointer addresses external memory.
2. Writing UBBRH Affects Both UART0 and UART1
Writing UBRRHI updates baud rate generator for both UART0 and UART1. The baud rate generator's counter is updated each time either UBRR or UBRRHI are written. Since the UBRRHI regiSter is shared by UART0 and UART1, changing thebaud rate for one UART will affect the operation of the other UART.
Problem Fix/Workaround
Do not update UBRRHI for one UART when transmitting/receiving data on the other.
1. Store Program Memory Instruction May Fail
At certain frequencies and voltages, the store program memory (SPM) instruction may fail.
Problem Fix/Workaround
Avoid using the SPM instruction.

ATmega163(L)

Erratas
Rev. F
- **Increased Interrupt Latency**
- **Interrupts Abort TWI Power-down**
- **TWI Master Does not Accept Spikes on Bus Lines**
- **TWCR Write Operations Ignored**

- **PWM not Phase Correct**
- **TWI is Speed Limited in Slave Mode**

6. Increased Interrupt Latency
In this device, some instructions are not interruptable, and will cause the interrupt latency to increase. The only practical problem concerns a loop followed by a twoword instruction while waiting for an interrupt. The loop may consist of a branch instruction or an absolute or relative jump back to itself like this:
loop: rjmp loop
<Two-word instruction>
In this case, a dead-lock situation arises.
Problem Fix/Workaround
In assembly, insert a nop instruction immediately after a loop to itself. The problem will normally be detected during development. In C, the only construct that will give this problem is an empty "for" loop; "for(;;)". Use "while(1)" or "do{ } while (1)" to avoid the problem.
5. Interrupts Abort TWI Power-down
TWI Power-down operation may be aborted by other interrupts. If an interrupt (e.g., INT0) occurs during TWI Power-down address watch and wakes the CPU up, the TWI aborts operation and returns to its idle state.
Problem Fix/Workaround
Ensure that the TWI Address Match is the only enabled interrupt when entering Power-down.
4. TWI Master Does not Accept Spikes on Bus Lines
When the part operates as Master, and the bus is idle (SDA = 1; SCL = 1), generating a short spike on SDA (SDA = 0 for a short interval), no interrupt is generated, and the status code is still $F8 (idle). But when the software initiates a new start condition and clears TWINT, nothing happens on SDA or SCL, and TWINT is never set again.
Problem Fix/Workaround
Either of the following:
1. Ensure that no spikes occur on SDA or SCL lines.
2. Receiving a valid START condition followed by a STOP condition provokes a bus error reported as a TWI interrupt with status code $00.
3. In a Single Master systems, the user should write the TWSTO bit immediately before writing the TWSTA bit.
3. TWCR Write Operation Ignored
Repeated write to TWCR must be delayed. If a write operation to TWCR is immediately followed by another write operation to TWCR, the first write operation may be ignored.
Problem Fix/Workaround
Ensure at least one instruction (e.g., nop) is executed between two writes to TWCR.
2. PWM not Phase Correct
In Phase-correct PWM mode, a change from OCRx = TOP to anything less than TOP does not change the OCx output. This gives a phase error in the following period.
Problem Fix/Workaround
Make sure this issue is not harmful to the application.
1. TWI is Speed Limited in Slave Mode
When the two-wire Serial Interface operates in Slave mode, frames may be undetected if the CPU frequency is less than 64 times the bus frequency.
Problem Fix/Workaround
Ensure that the CPU frequency is at least 64 times the TWI bus frequency.

ATmega16

ATmega16(L) Rev. G. There are no errata for this revision of ATmega16.
ATmega16(L) Rev. H. There are no errata for this revision of ATmega16.

ATmega169 Rev B

Erratas
- **Internal Oscillator Runs at 4 MHz**
- **LCD Contrast Voltage is not Correct**
- **External Oscillator is Non-functional**
- **USART**
- **ADC Measures with Lower Accuracy than Specified**
- **Serial Downloading**

6. Internal Oscillator Runs at 4 MHz
The Internal Oscillator runs at 4 MHz instead of the specified 8 MHz. Therefore, all Flash/EEPROM programming times are twice as long as specified. This includes Chip Erase, Byte programming, Page programming, Fuse programming, Lock bit programming, EEPROM write from the CPU, and Flash Self-Programming. For this reason, rev-B samples are shipped with the CKDIV8 Fuse unprogrammed.
Problem Fix/Workaround
If 8 MHz operation is required, apply an external clock (this will be fixed in rev. C).

5. LCD Contrast Voltage is not Correct
The LCD contrast voltage between 1.8V and 3.1V is incorrect. When the VCC is between 1.8V and 3.1V, the LCD contrast voltage drops approx. 0.5V. The current consumption in this interval is higher than expected.
Problem Fix/Workaround
Contrast will be wrong, but display will still be readable, can be partly compensated for using the contrast control register (this will be fixed in rev. C).
4. External Oscillator is Non-functional
The external oscillator does not run with the setup described in the data sheet.
Problem Fix/Workaround
Use other clock source (this will be fixed in rev. C).
Alternative Problem Fix/Workaround
Adding a pull-down on XTAL1 will start the Oscillator.
3. USART
Writing TXEN to zero during transmission causes the transmission to suddenly stop. The data sheet description tells that the transmission should complete before stopping the USART when TXEN is written to zero.
Problem Fix/Workaround
Ensure that the transmission is complete before writing TXEN to zero (this will be fixed in rev. C).
2. ADC Measures with Lower Accuracy than Specified
The ADC does not work as intended. There is a positive offset in the result.
Problem Fix/Workaround
This will be fixed in rev. C.
1. Serial downloading
When entering Serial Programming mode the second byte will not echo back as described in the Serial Programming algorithm.
Problem Fix/Workaround
Check if the third byte echoes back to ensure that the device is in Programming mode (this will be fixed in rev. C).

ATmega169 Rev C

- **High Current Consumption In Power Down when JTAGEN is Programmed**
- **LCD Contrast Control**
- **Some Data Combinations Can Result in Dim Segments on the LCD**
- **LCD Current Consumption**

4. High Current Consumption In Power Down when JTAGEN is Programmed
The input buffer on TDO (PF6) is always enabled and the pull-up is always disabled when JTAG is programmed. This can leave the output floating.
Problem Fix/Workaround
Add external pull-up to PF6. Unprogram the JTAGEN Fuse before shipping out the end product.
3. LCD Contrast Control
The contrast control is not working properly when using synchronous clock (chip clock) to obtain an LCD clock, and the chip clock is 125 kHz or faster.
Problem Fix/Workaround
Use a low chip clock frequency (32 kHz) or apply an external voltage to the LCDCAP pin.
2. Some Data Combinations Can Result in Dim Segments on the LCD
All segments connected to a common plane might be dimmed (lower contrast) when a certain combination of data is displayed.
Problem Fix/Workaround
Default waveform: If there are any unused segment pins, loading one of these with a 1 nF capacitor and always write '0' to this segment eliminates the problem.
Low power waveform: Add a 1 nF capacitor to each common pin.
1. LCD Current Consumption
In an interval where VCC is within the range VLCD -0.2V to VLCD + 0.4V, the LCD current consumption is up to three times higher than expected. This will only be an issue in Power-save mode with the LCD running as the LCD current is negligible compared to the overall power consumption in all other modes of operation.
Problem Fix/Workaround
No known workaround.

ATmega323

Rev. B
Errata
- **Interrupts Abort TWI Power-down**
- **TWI Master Does not Accept Spikes on Bus Lines**
- **TWCR Write Operations Ignored when Immediately Repeated**
- **PWM not Phase Correct**

- **TWI is Speed Limited in Slave Mode**
- **Problems with UBRR Settings**
- **Missing OverRun Flag and Fake Frame Error in USART**

7. Interrupts Abort TWI Power-down
TWI Power-down operation may wake up by other interrupts. If an interrupt (e.g., INT0) occurs during TWI Power-down address watch and wakes up the CPU, the TWI aborts operation and returns to its idle state. If the interrupt occurs in the middle of a Power-down Address Match (i.e., during reading of a slave address), the received address will be lost and the Slave will not return an ACN.
Problem Fix/Workaround
Ensure that the TWI Address Match is the only enabled interrupt when entering Power-down. The Master can handle this by resending the request if NACH is received.
6. TWI Master Does not Accept Spikes on Bus Lines
When the part operates as Master, and the bus is idle (SDA = 1; SCL = 1), generating a short spike on SDA (SDA = 0 for a short interval), no interrupt is generated, and the status code is still $F8 (idle). But when the software initiates a new start condition and clears TWINT, nothing happens on SDA or SCL, and TWINT is never set again.
Problem Fix/Workaround
Either of the following:
1. Ensure no spikes occur on SDA or SCL lines.
2. Generate a valid START condition followed by a STOP condition on the bus. This provokes a bus error reported as a TWI interrupt with status code $00.
3. In a Single-master system, the user should write the TWSTO bit immediately before writing the TWSTA bit.
5. TWCR Write Operation Ignored when Immediately Repeated
Repeated write to TWCR must be delayed. If a write operation to TWCR is immediately followed by another write operation to TWCR, the first write operation may be ignored.
Problem Fix/Workaround
Ensure at least one instruction (e.g., NOP) is executed between two writes to TWCR.
4. PWM not Phase Correct
In phase-correct PWM mode, a change from OCRx = TOP to anything less than TOP does not change the OCx output. This gives a phase error in the following period.
Problem Fix/Workaround
Make sure this issue is not harmful to the application.
3. TWI is Speed Limited in Slave Mode
When the Two-wire Serial Interface operates in Slave mode, frames may be undetected if the CPU frequency is less than 64 times the bus frequency.
Problem Fix/Workaround
Ensure that the CPU frequency is at least 64 times the TWI bus frequency.
2. Problems with UBRR Settings
The baud rate corresponding to the previous UBRR setting is used for the first transmitted/ received bit when either UBRRH or UBRRL is written. This will disturb communication if the UBRR is changed from a very high to a very low baud rate setting, as the internal baud rate counter will have to count down to zero before using the new setting. In addition, writing to UBRRL incorrectly clears the UBRRH setting.
Problem Fix/Workaround
UBRRH must be written after UBRRL because setting UBRRL clears UBRRH. By doing an additional dummy write to UBRRH, the baud rate is set correctly. The following is an example on how to set UBRR. UBRRH is updated first for upward compability with corrected devices.

```
ldi r17, HIGH(baud)
ldi r16, LOW(baud)
out UBRRH, r17 ; Added for upward compability
out UBRRL, r16 ; Set new UBRRL, UBRRH incorrectly cleared
out UBRRH, r17 ; Set new UBRRH
out UBRRH, r17 ; Loads the baud rate counter with new (correct) value
```

1. Missing OverRun Flag and Fake Frame Error in USART
When the USART has received three characters without any of them been read, the USART FIFO is full. If the USART detects the start bit of a fourth character, the Data OverRun (DOR) Flag will be set for the third character. However, if a read from the USART Data Register is performed just after the start bit of the fourth byte is received, a Frame Error is generated for character three. If the USART Data Register is read between the reception of the first data bit and the end of the fourth character, the Data OverRun Flag of character three will be lost.
Problem Fix/Workaround
The user should design the application to never completely fill the USART FIFO. If this is not possible, the user must use a high-level protocol to be able able to detect if any characters were lost and request a retransmission if this happens.

ATmega32 Rev. A

There are no errata for this revision of ATmega32.

ATmega165

There are no errata for this revision of ATmega165.

ATmega128 Rev. F

There are no errata for this revision of ATmega128.

ATmega128 Rev. G

There are no errata for this revision of ATmega128.

ATmega325

There are no errata for this revision of ATmega325.

ATmega3250

There are no errata for this revision of ATmega3250.

ATmega645

There are no errata for this revision of ATmega645.

ATmega6450

There are no errata for this revision of ATmega6450.

ATmega329 REV.A

• LCD contrast voltage too high

1. LCD contrast voltage too high

When the LCD is active and using low power waveform, the LCD contrast voltage can be too high. This occurs when VCC is higher than VLCD, and when using low LCD drivetime.

Problem Fix/Workaround

There are several possible workarounds:

- Use normal waveform instead of low power waveform
- Use drivetime of 375 µs or longer

ATmega3290 rev. A

• LCD contrast voltage too high

1. LCD contrast voltage too high

When the LCD is active and using low power waveform, the LCD contrast voltage can be too high. This occurs when VCC is higher than VLCD, and when using low LCD drivetime.

Problem Fix/Workaround

There are several possible workarounds:

- Use normal waveform instead of low power waveform
- Use drivetime of 375 µs or longer

ATmega649 rev. A

There is no errata for this revision of ATmega649.

ATmega6490 rev. A

There is no errata for this revision of ATmega6490.

Errata ATmega48

The revision letter in this section refers to the revision of the ATmega48 device.

Rev A • Wrong values read after Erase Only operation

• Watchdog Timer Interrupt disabled

• Start-up time with Crystal Oscillator is higher than expected

• High Power Consumption in Power-down with External Clock

• Asynchronous Oscillator does not stop in Power-down

1. Wrong values read after Erase Only operation

At supply voltages below 2.7 V, an EEPROM location that is erased by the Erase Only operation may read as programmed (0x00).

Problem Fix/Workaround

If it is necessary to read an EEPROM location after Erase Only, use an Atomic Write operation with 0xFF as data in order to erase a location. In any case, the Write Only operation can be used as intended. Thus no special considerations are needed as long as the erased location is not read before it is programmed.

2. Watchdog Timer Interrupt disabled

If the watchdog timer interrupt flag is not cleared before a new timeout occurs, the watchdog will be disabled, and the interrupt flag will automatically be cleared. This is only applicable in interrupt only mode. If the Watchdog is configured to reset the device in the watchdog time-out following an interrupt, the device works correctly.

Problem fix / Workaround

Make sure there is enough time to always service the first timeout event before a new watchdog timeout occurs. This is done by selecting a long enough time-out period.

3. Start-up time with Crystal Oscillator is higher than expected

The clock counting part of the start-up time is about 2 times higher than expected for all start-up periods when running on an external Crystal. This applies only when waking up by reset. Wake-up from power down is not affected. For most settings, the clock counting parts is a small fraction of the overall start-up time, and thus, the problem can be ignored. The exception is when using a very low frequency crystal like for instance a 32 kHz clock crystal.

Problem fix / Workaround

No known workaround.

4. High Power Consumption in Power-down with External Clock

The power consumption in power down with an active external clock is about 10 times higher than when using internal RC or external oscillators.

Problem fix / Workaround

Stop the external clock when the device is in power down.

5. Asynchronous Oscillator does not stop in Power-down

The Asynchronous oscillator does not stop when entering power down mode. This leads to higher power consumption than expected.

Problem fix / Workaround

Manually disable the asynchronous timer before entering power down.

Instruction Set Nomenclature:

Status Register (SREG)

SREG:	Status register
C:	Carry flag in status register
Z:	Zero flag in status register
N:	Negative flag in status register
V:	Two's complement overflow indicator
S:	N ⊕V, For signed tests
H:	Half Carry flag in the status register
T:	Transfer bit used by BLD and BST instructions
I:	Global interrupt enable/disable flag

Registers and Operands

Rd:	Destination (and source) register in the register file
Rr:	Source register in the register file
R:	Result after instruction is executed
K:	Constant data
k:	Constant address
b:	Bit in the register file or I/O register (3 bit)
s:	Bit in the status register (3 bit)
X,Y,Z:	Indirect address register

(X=R27:R26, Y=R29:R28 and Z=R31:R30)

A:	I/O location address
q:	Displacement for direct addressing (6 bit)

I/O Registers

RAMPX, RAMPY, RAMPZ
Registers concatenated with the X, Y and Z registers enabling indirect addressing of the whole data space on MCUs with more than 64K bytes data space, and constant data fetch on MCUs with more than 64K bytes program space.

RAMPD
Register concatenated with the Z register enabling direct addressing of the whole data space on MCUs with more than 64K bytes data space.

EIND
Register concatenated with the instruction word enabling indirect jump and call to the whole program space on MCUs with
more than 64K bytes program space.

Stack
STACK: Stack for return address and pushed registers
SP: Stack Pointer to STACK

Flags
↔ : Flag affected by instruction
0 : Flag cleared by instruction
1 : Flag set by instruction
- : Flag not affected by instruction

Conditional Branch Summary

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	Z← (N ⊕V) = 0	BRLT[1]	Rd ← Rr	Z+(N ⊕ V) = 1	BRGE*	Signed
Rd ≥ Rr	(N ⊕V) = 0	BRGE	Rd < Rr	(N ⊕ V) = 1	BRLT	Signed
Rd = Rr	Z = 1	BREQ	Rd ← Rr	Z = 0	BRNE	Signed
Rd ≤ Rr	Z+(N ⊕V) = 1	BRGE[1]	Rd > Rr	Z•(N ⊕ V) = 0	BRLT*	Signed
Rd < Rr	(N⊕ V) = 1	BRLT	Rd ← Rr	(N ⊕ V) = 0	BRGE	Signed
Rd > Rr	C + Z = 0	BRLO[1]	Rd ← Rr	C + Z = 1	BRSH*	Unsigned
Rd ≥ Rr	C = 0	BRSH/BRCC	Rd < Rr	C = 1	BRLO/BRCS	Unsigned
Rd = Rr	Z = 1	BREQ	Rd ← Rr	Z = 0	BRNE	Unsigned
Rd ≤ Rr	C + Z = 1	BRSH[1]	Rd > Rr	C + Z = 0	BRLO*	Unsigned
Rd < Rr	C = 1	BRLO/BRCS	Rd ← Rr	C = 0	BRSH/BRCC	Unsigned
Carry	C = 1	BRCS	No carry	C = 0	BRCC	Simple
Negative	N = 1	BRMI	Positive	N = 0	BRPL	Simple
Overflow	V = 1	BRVS	No overflow	V = 0	BRVC	Simple
Zero	Z = 1	BREQ	Not zero	Z = 0	BRNE	Simple

Note: 1. Interchange Rd and Rr in the operation before the test. i.e. CP Rd,Rr ← CP Rr,Rd

Complete Instruction Set Summary

Notes: 1. Not all instructions are available in all devices. Refer to the device specific instruction summary.
2. Cycle times for data memory accesses assume internal memory accesses, and are not valid for accesses via the external RAM interface. For LD, ST, LDS, STS, PUSH, POP, add one cycle plus one cycle for each wait state. For CALL, ICALL, EICALL, RCALL, RET, RETI in devices with 16 bit PC, add three cycles plus two cycles for each wait state. For CALL,ICALL, EICALL, RCALL, RET, RETI in devices with 22 bit PC, add five cycles plus three cycles for each wait state.

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
Arithmetic and Logic Instructions					
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1
ADIW	Rd, K	Add Immediate to Word	Rd+1:Rd ← Rd+1:Rd + K	Z,C,N,V,S	2

SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1
SBIW	Rd, K	Subtract Immediate from Word	Rd+1:Rd ← Rd+1:Rd - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕Rr	Z,N,V,S	1
COM	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1
CLR	Rd	Clear Register	Rd ← Rd ⊕Rd	Z,N,V,S	1
SER	Rd	Set Register	Rd ← $FF	None	1
MUL	Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr (UU)	Z,C	2
MULS	Rd,Rr	Multiply Signed	R1:R0 ← Rd x Rr (SS)	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr (SU)	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr)<<1 (UU)	Z,C	2
FMULS	Rd,Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr)<<1 (SS)	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr)<<1 (SU)	Z,C	2

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
Branch Instructions					
RJMP	k	Relative Jump	PC← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC(15:0) ← Z	None	2
EIJMP		Extended Indirect Jump to(Z)	PC(15:0) ← Z	None	2
JMP	k	Jump	PC ← k	None	3
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	3 / 4
ICALL		Indirect Call to (Z)	PC(15:0) ← Z	None	3 / 4
EICALL		Extended Indirect Call to (Z)	PC(15:0) ← Z	None	4
CALL	k	Call Subroutine	PC ← k	None	4 / 5
RET		Subroutine Return	PC ← STACK	None	4 / 5
RETI		Interrupt Return	PC ← STACK	I	4 / 5
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V ,S,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V ,S,H	1
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V ,S,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	A, b	Skip if Bit in I/O Register Cleared	If (I/O(A,b)=0) PC← PC + 2 or 3	None	1 / 2 / 3
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ←PC+k +1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2

BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕V= 0) then PC← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than, Signed	if (N ⊕V= 1) then PC← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is	if (V = 0) then PC ← PC + k + 1	None	1 / 2
		Cleared			
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2

Data Transfer Instructions

MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LDS	Rd, k	Load Direct from data space	Rd ← (k)	None	2
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2
LD	Rd, -X	Load Indirect and Pre- Decrement	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, -Y	Load Indirect and Pre- Decrement	Y ← Y - 1, Rd← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre- Decrement	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
STS	k, Rr	Store Direct to data space	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post- Increment	(X) ← Rr, X ← X + 1	None	2
ST	-X, Rr	Store Indirect and Pre- Decrement	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post- Increment	(Y) ← Rr, Y ← Y + 1	None	2
ST	-Y, Rr	Store Indirect and Pre- Decrement	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post- Increment	(Z) ← Rr, Z ← Z + 1	None	2

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
ST	-Z, Rr	Store Indirect and Pre- Decrement	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post- Increment	Rd ← (Z), Z ←Z + 1	None	3
ELPM		Extended Load Program Memory	R0 ← (RAMPZ:Z)	None	3
ELPM	Rd, Z	Extended Load Program Memory	Rd ← (RAMPZ:Z)	None	3
ELPM	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd ← (RAMPZ:Z), Z ←Z + 1	None	3
SPM		Store Program Memory	(Z) ←R1:R0	None	-

ESPM		Extended Store Program Memory	(RAMPZ:Z) ←R1:R0	None	-
IN	Rd, A	In From I/O Location	Rd ←I/O(A)	None	1
OUT	A, Rr	Out To I/O Location	I/O(A) ←Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ←Rr	None	2
POP	Rd	Pop Register from Stack	Rd ←STACK	None	2
Bit and Bit-test Instructions					
LSL	Rd	Logical Shift Left	Rd(n+1)←Rd(n),Rd(0) ←0,C←Rd(7)	Z,C,N,V,H	1
LSR	Rd	Logical Shift Right	Rd(n←Rd(n+1),Rd(7←0,C←Rd(0)	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0←C,Rd(n+1) ←Rd(n),C←Rd(7)	Z,C,N,V,H	1
ROR	Rd	Rotate Right Through Carry	Rd(7) ←C,Rd(n) ←Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0) ← Rd(7..4)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
SBI	A, b	Set Bit in I/O Register	I/O(A, b) ← 1	None	2
CBI	A, b	Clear Bit in I/O Register	I/O(A, b) ← 0	None	2
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ←1	C	1
CLC		Clear Carry	C ←0	C	1
SEN		Set Negative Flag	N ←1	N	1
CLN		Clear Negative Flag	N ←0	N	1
SEZ		Set Zero Flag	Z ←1	Z	1
CLZ		Clear Zero Flag	Z ←0	Z	1
SEI		Global Interrupt Enable	I ←1	I	1
CLI		Global Interrupt Disable	I ←0	I	1
SES		Set Signed Test Flag	S ←1	S	1
CLS		Clear Signed Test Flag	S ←0	S	1
SEV		Set Two's Complement Overflow	V ←1	V	1
CLV		Clear Two's Complement Overflow	V ←0	V	1
SET		Set T in SREG	T ←1	T	1
CLT		Clear T in SREG	T ←0	T	1
SEH		Set Half Carry Flag in SREG	H ←1	H	1
CLH		Clear Half Carry Flag in SREG	H ←0	H	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1

BASCOM Editor Keys

Key	Action
LEFT ARROW	One character to the left
RIGHT ARROW	One character to the right
UP ARROW	One line up
DOWN ARROW	One line down
HOME	To the beginning of a line
END	To the end of a line
PAGE UP	Up one window
PAGE DOWN	Down one window
CTRL+LEFT	One word to the left
CTRL+RIGHT	One word to the right
CTRL+HOME	To the start of the text
CTRL+END	To the end of the text
CTRL+ Y	Delete current line
INS	Toggles insert/overstrike mode
F1	Help (context sensitive)
F3	Find next text

F4	Send to chip (run flash programmer)
F5	Run
F7	Compile File
F8	Step
F9	Set breakpoint
F10	Run to
CTRL+F7	Syntax Check
CTRL+F	Find text
CTRL+G	Go to line
CTRL+K+x	Toggle bookmark. X can be 1-8
CTRL+L	LCD Designer
CTRL+M	File Simulation
CTRL+N	New File
CTRL+O	Load File
CTRL+P	Print File
CTRL+Q+x	Go to Bookmark. X can be 1-8
CTRL+R	Replace text
CTRL+S	Save File
CTRL+T	Terminal emulator
CTRL+P	Compiler Options
CTRL+W	Show result of compilation
CTRL+X	Cut selected text to clipboard
CTRL+Z	Undo last modification
SHIFT+CTRL+Z	Redo last undo
CTRL+INS	Copy selected text to clipboard
SHIFT+INS	Copy text from clipboard to editor
CTRL+SHIFT+J	Indent Block
CTRL+SHIFT+U	Unindent Block
Select text	Hold the SHIFT key down and use the cursor keys to select text. or keep the left mouse key pressed and tag the cursor over the text to select.

RESERVED WORDS

The following table shows the reserved BASCOM statements or characters.

FOR	CPEEKH()	ACK	^
FOURTH	CRYSTAL	ABS()	!
FOURTHLINE	CURSOR	ALIAS	:
FUNCTION		AND	$BAUD
	DATA	AS	$CRYSTAL
GATE	DATE$	ASC()	$DATA
GETAD()	DEBOUNCE	AT	
GETRC5()	DECR		$DEFAULT
GOSUB	DECLARE	BAUD	
GOTO	DEFBIT	BCD()	$END
	DEFBYTE	BIT	$EEPROM
HEXVAL()	DEFLNG	BITWAIT	$EXTERNAL
HIGH()	DEFWORD	BLINK	$INCLUDE
HOME	DEGSNG	BOOLEAN	$LCD
	DEFLCDCHAR	BYTE	$LCDRS
I2CRECEIVE	DEFINT	BYVAL	$LCDPUTCTRL
I2CSEND	DEFWORD		$LCDPUTDATA
I2CSTART	DELAY	CALL	$LIB
I2CSTOP	DIM	CAPTURE1	$REGFILE
I2CRBYTE	DISABLE	CASE	$SERIALINPUT
I2CWBYTE	DISPLAY	CHR()	$SERIALINPUT2LC
IDLE	DO	CLS	D
	DOWNTO	CLOSE	
If			
INCR	ELSE	Compare1a	$SERIALOUTPUT
INKEY	ELSEIF	COMPARE1B	$WAITSTATE
INP()	ENABLE		$XRAMSIZE
INPUT	END	Config	$XRAMSTART
INPUTBIN	ERAM	CONST	
INPUTHEX	ERASE	COUNTER	
INT0	ERR	COUNTER0	1WRESET
INT1	EXIT	COUNTER1	1WREAD
INTEGER	EXTERNAL	COUNTER2	1WWRITE
INTERNAL		CPEEK()	

TIME$
THIRD
THIRDLINE
TIMER0
TIMER1
TIMER2
TO
TRIM()

UCASE()
UNTIL
UPPER
UPPERLINE

VAL()
VARPTR()

WAIT
WAITKEY()
WAITMS
WAITUS
WATCHDOG
WRITEEEPROM
WEND
WHILE
WORD

XOR

XRAM

RETURN
RIGHT
RIGHT()
ROTATE
RTRIM()

SELECT
SERIAL
SET
SHIFT
SHIFTLCD
SHIFTCURSOR
SHIFTIN
SHIFTOUT
SOUND
SPACE()
SPIINIT
SPIIN
SPIMOVE
SPIOUT
START
STEP
STR()
STRING()
STOP
STOP TIMER
SUB
SWAP

THEN

NOSAVE
NOT

OFF
ON
OR
OUT
OUTPUT

PEEK()
POKE
PORTA
PORTB
PORTC
PORTD
PORTE
PORTF
POWERDOWN
PRINT
PRINTBIN

Pulseout
PWM1A
PWM1B

READ
READEEPROM
REM
RESET
RESTORE

INSTR
IS

LCASE()
LCD
LEFT
LEFT()
LEN()
LOAD
LOCAL
LOCATE
LONG
LOOKUP()
LOOKUPSTR()
LOOP
LTRIM()
LOW()
LOWER
LOWERLINE

MAKEBCD()
MAKEDEC()
MAKEINT()
MID()
MOD
MODE

NACK
NEXT
NOBLINK

Error Codes In bascom AVR

The following table lists errors that can occur.

Error	Description
24	Out of EPROM space
25	Variable already dimensioned
26	AS expected
27	parameter expected
28	IF THEN expected
29	SELECT CASE expected
30	BIT's are GLOBAL and can not be erased
31	Invalid data type
32	Variable not dimensioned
33	GLOBAL variable can not be ERASED
34	Invalid number of parameters
35	3 parameters expected
36	THEN expected
37	Invalid comparison operator
38	Operation not possible on BITS
39	FOR expected
40	Variable can not be used with RESET
41	Variable can not be used with SET
42	Numeric parameter expected
43	File not found
44	2 variables expected
45	DO expected
46	Assignment error
47	UNTIL expected

Error	Description
1	Unknown statement
2	Unknown structure EXIT statement
3	WHILE expected
4	No more space for IRAM BIT
5	No more space for BIT
6	. expected in filename
7	IF THEN expected
8	BASIC source file not found
9	Maximum 128 aliases allowed
10	Unknown LCD type
11	INPUT, OUTPUT, 0 or 1 expected
12	Unknown CONFIG parameter
13	CONST already specified
14	Only IRAM bytes supported
15	Wrong data type
16	Unknown Definition
17	9 parameters expected
18	BIT only allowed with IRAM or SRAM
19	STRING length expected (DIM S AS STRING * 12 ,for example)
20	Unknown DATA TYPE
21	Out of IRAM space
22	Out of SRAM space
23	Out of XRAM space

Error	Description
202	.EQU not found, probably using functions that are not supported by the selected chip
203	Error in LD or LDD statement
204	Error in ST or STD statement
205	} expected
206	Library file not found
207	Library file already registered
210	Bit definition not found
211	External routine not found
212	LOW LEVEL, RISING or FALLING expected
213	String expected for assignment
214	Size of XRAM string 0
215	Unknown ASM mnemonic
216	CONST not defined
217	No arrays allowed with BIT/BOOLEAN data type
218	Register must be in range from R16-R31
219	INT0-INT3 are always low level triggered in the MEGA
220	Forward jump out of range
221	Backward jump out of range
222	Illegal character
223	* expected
224	Index out of range
225	() may not be used with constants
226	Numeric of string constant expected
227	SRAM start greater than SRAM end
228	DATA line must be placed after the END statement
229	End Sub or End Function expected
230	You can not write to a PIN register
231	TO expected
232	Not supported for the selected micro
233	READ only works for normal DATA lines, not for EPROM data
234	') block comment expected first
235	'(block comment expected first
236	Value does not fit into byte
238	Variable is not dimensioned as an array
239	Invalid code sequence because of AVR hardware bug
240	END FUNCTION expected
241	END SUB expected
242	Source variable does not match the target variable
243	Bit index out of range for supplied data type
244	Do not use the Y pointer
245	No arrays supported with IRAM variable
246	No more room for .DEF definitions
247	. expected
248	BYVAL should be used in declaration
249	ISR already defined
250	GOSUB expected
251	Label must be named SECTIC
252	Integer or Word expected
253	ERAM variable can not be used
254	Variable expected

Error	Description
50	Value doesn't fit into INTEGER
51	Value doesn't fit into WORD
52	Value doesn't fit into LONG
60	Duplicate label
61	Label not found
62	SUB or FUNCTION expected first
63	Integer or Long expected for ABS()
64	,expected
65	device was not OPEN
66	device already OPENED
68	channel expected
70	BAUD rate not possible
71	Different parameter type passed then declared
72	Getclass error. This is an internal error.
73	Printing this FUNCTION not yet supported
74	3 parameters expected
80	Code does not fit into target chip
81	Use HEX(var) instead of PRINTHEX
82	Use HEX(var) instead of LCDHEX
85	Unknown interrupt source
86	Invalid parameter for TIMER configuration
87	ALIAS already used
88	0 or 1 expected
89	Out of range : must be 1-4
90	Address out of bounds
91	INPUT, OUTPUT, BINARY, or RANDOM expected
92	LEFT or RIGHT expected
93	Variable not dimensioned
94	Too many bits specified
95	Falling or rising expected for edge
96	Prescale value must be 1,8,64,256 or 1024
97	SUB or FUNCTION must be DECLARED first
98	SET or RESET expected
99	TYPE expected
100	No array support for IRAM variables
101	Can't find HW-register
102	Error in internal routine
103	= expected
104	LoadReg error
105	StoreBit error
106	Unknown register
107	LoadnumValue error
108	Unknown directive in device file
109	= expected in include file for .EQU
110	Include file not found
111	SUB or FUNCTION not DECLARED
112	SUB/FUNCTION name expected
113	SUB/FUNCTION already DECLARED
114	LOCAL only allowed in SUB or FUNCTION
115	#channel expected
116	Invalid register file
117	Unknown interrupt
200	.DEF not found
201	Low Pointer register expected

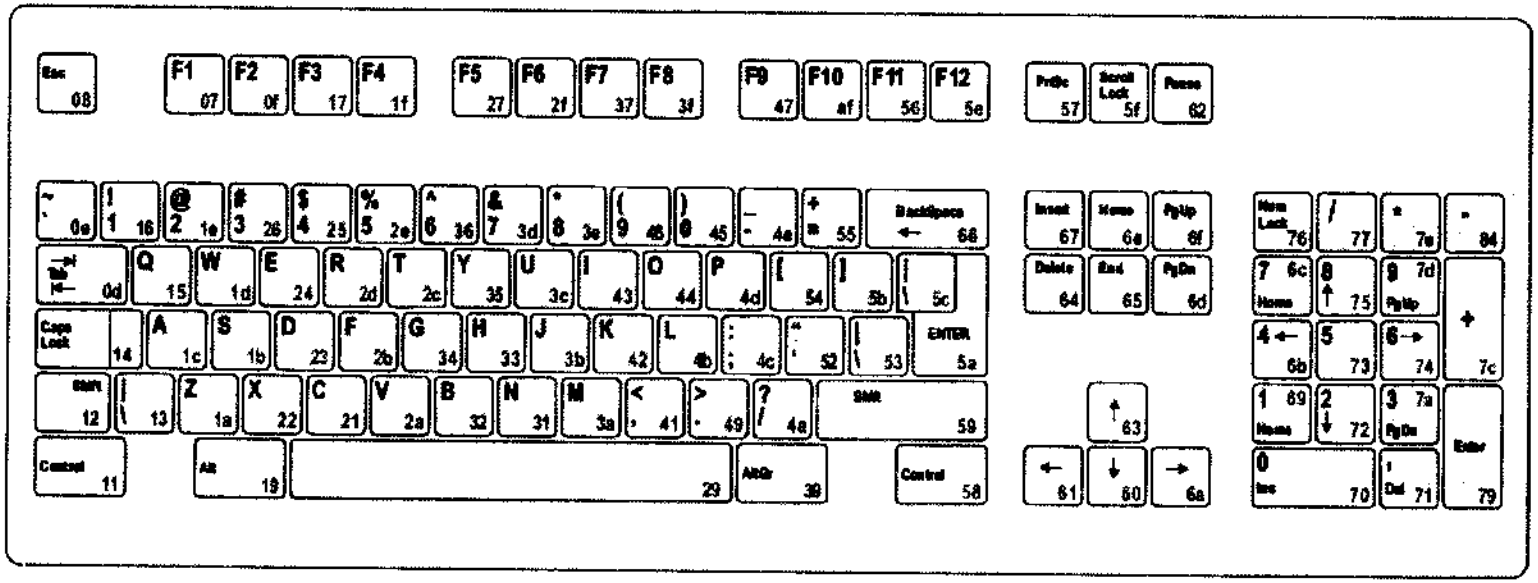
261 Array expected
262 ON or OFF expected
263 Array index out of range
264 Use ECHO OFF and ECHO ON instead
265 offset expected in LDD or STD like Z+1

325 Pre-scale value must be 1,8,32,128,256 or 1024
326 #ENDIF expected
327 Maximum size is 255
328 Not valid for SW UART
999 DEMO/BETA only supports 2048 bytes of code

255 Z or Z+ expected
256 Single expected
257 " " expected
258 SRAM string expected
259 - not allowed for a byte
260 Value larger than string length
266 TIMER0, TIMER1 or TIMER2 expected
267 Numeric constant expected
268 Param must be in range from 0-3
269 .END SELECT expected
270 Address already occupied
322 Data type not supported with statement
323 Label too long
324 Chip not supported by I2C slave library

Standard AT PC keyboard layout

Scan code SET 3 - hex numbers in red



پیوست

ب

برد مدار چاپی آموزشی

توضیحاتی درباره برد کتاب

برد آموزشی کتاب می‌تواند برای تمام کاربران میکروکنترلرهای AVR با نرم‌افزارهای مختلفی همچون bascom، codevision ، fastavr و .. استفاده شود. برد آموزشی مونتاژ نشده است و بایستی قطعات مورد نیاز را تهیه و بر روی آن لحیم کنید. با این کار مقدار زیادی در هزینه‌های خود نسبت به خرید بردهای آماده صرفه‌جویی کرده‌اید.

امکانات برد آموزشی

برد یک رو با چاپ سبز و مارکاژ با طراحی مناسب برای هدفهای آموزشی در ابعاد ۱۰×۱۰ سانتیمتر به همراه کتاب (البته اختیاری) ارائه شده است. میکرو AVR استفاده شده در برد می‌تواند یکی از انواع ATMEGA32/8535/16/323/163/AT90S8535 در بسته‌بندی ۴۰ پایه DIP باشد. پروگرامر استفاده شده از نوع موازی است و باقر آن روی برد تعبیه شده است. مدارهای برد شامل پروگرامر، LCD (در انواع مختلف متنی) ،7SEG سه تایی، کی بورد 4*4 ،پورت RS232 ، سنسور دمای LM335 ، رله تک کنتاکت، کی بورد کامپیوتر، خروجی آنالوگ توسط پالس PWM، کلید، منبع تغذیه 5 و 12 ولت و مدارهای مورد نیاز برای راه‌اندازی میکرو هستند.

در بخش پروژه‌های عملی (فصل هشتم) ۱۱ برنامه برای راه‌اندازی امکانات گفته شده نوشته و آزمایش شده‌اند ولی می‌توان پروژه‌های گوناگون دیگری نیز برای آن تعریف و اجرا کرد.

قیمت قطعات

در جدول زیر قیمت هر یک از قطعات موردنیاز برای کتاب را که در آذر ۱۳۸۵ جمع‌آوری شده اند، می‌بینید. بدیهی است که در اثر گذشت زمان قیمتها تغییر خواهند کرد. (تذکر این نکته ضروری است که خرید برد و تهیه قطعات آن از بازار ۵۰٪ صرفه جویی را نسبت به بردهای آماده در برخواهد داشت)

جدول ب-۱ لیست قطعات و قیمت تقریبی آنها در آذرماه ۸۵

Comment	Price(Rial) Per one	Quantity	Components
1-22UF(16v)	250	4	C5, C6, C8, C9
100NF	100	1	C13
100u(25V)	500	1	C12
10UH	500	1	L1
10k	35	2	R1, R10
15-32PF	250	2	C10, C11
1M	35	1	R11
1N4001	250	1	D2
1k	35	2	R8, R13
1u(tantalum)	500	2	C2, C3
1u(tantalum)	500	1	C7
220	35	3	R2, R3, R4
2200U(25V)	1200	1	C1
33k	35	1	R9
4.7K	35	1	R12
74LS244	1200	1	Q4
74LS47	3000	1	Q6
7805	1200	1	U2
7812	1200	1	U1
7SEGMENT_3PART(COMMON ANODE)	4000	1	D3
BC337	250	1	Q3
KEYPAD4*4	16000	1	C4
LCD16*2/16*4/20*4/20*2/16*1	35000(2*16)	1	JP1
MAX232	3500	1	Q7
OP07	1500	1	Q5
POT5k	1000	1	R7
POWER LED	100	1	LED4

www.edugroup.ir