

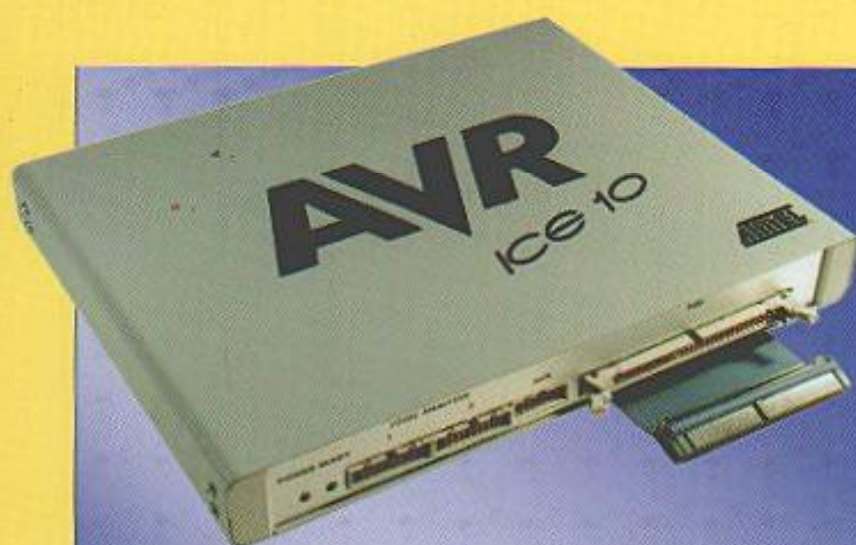


میکروکنترلرهای

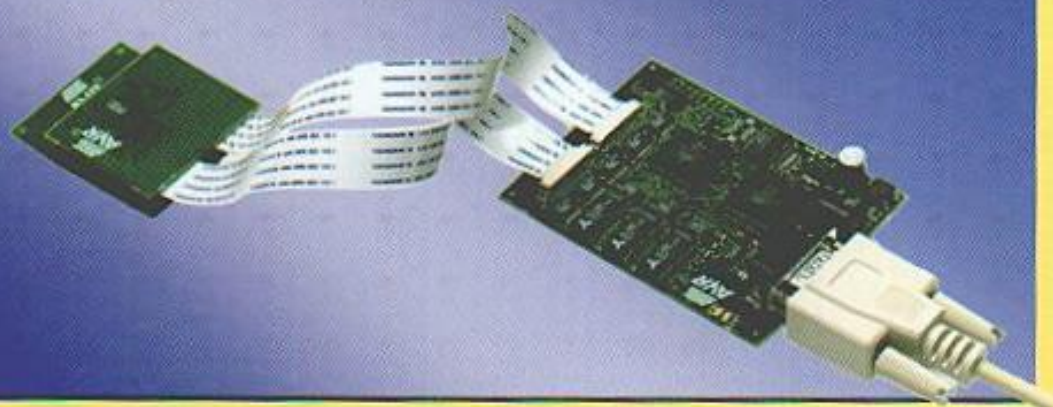
AVR

چاپ سوم

مهندس علی کاهه



سخت افزار
نرم افزار
برنامه ریزی
ارتباط دهی





همراه با

میکروکنترلرهای

AVR

علی کاهه



موسسه علمی فرهنگی

کاهه ، علی

میکروکنترلرهای AVR [ای.وی.آر] / تألیف علی کاهه.

تهران نص ۱۳۸۴

۳۲۰ ص. : مصور، جدول ، نمودار.

ISBN:964-410-012-3

با CD: ۳۲۰۰۰ ریال

فهرست نویسی بر اساس اطلاعات فیبا.

۱. کنترل کننده های برنامه پذیر. ۲. سیستم های کنترل رقمی. ۳. کامپیوترهای ریز. الف. عنوان

۶۲۹ / ۸۹۵

TJ۱۲۲۳/م۹م۲۵

۱۶۳۶-۸۳م

کتابخانه ملی ایران



موسسه علمی فرهنگی

میکروکنترلرهای AVR

مؤلف : مهندس علی کاهه

چاپ اول : بهار ۸۳ / چاپ دوم : پاییز ۸۳

چاپ سوم : بهار ۸۴

چاپ و صحافی : سازمان چاپ و انتشارات وزارت فرهنگ و ارشاد اسلامی

طراحی و آماده سازی: موسسه علمی فرهنگی نص

قیمت با CD: ۳۲۰۰ تومان

تهران میدان انقلاب خ اردیبهشت بن بست مبین شماره شماره ۲۲۷

فکس: ۶۴۱۲۳۸۵ تلفن ۴-۶۹۵۳۸۸۳ / ص. پ. ۱۳۱۴۵/۸۶۳

ISBN:964-410-012-3

شابک: ۹۶۴-۴۱۰-۰۱۲-۳

پیشگفتار

تقدیم به پدر و مادرم

مطالعه و کار با یک میکروکنترلر غالباً برای دانشجویان لازم و ضروری است و چه بهتر که این یادگیری به‌روز باشد و زمانی را که دانشجو برای مطالعه صرف می‌کند، برای میکروکنترلی جدید باشد. 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 دانشجویان قرار گرفته است. در راستای آموزش و یادگیری، کتب و نشریات آموزشی نقش موثری را ایفا می‌کنند. به همین دلیل به نشر کتابی که در دست شماست پرداخته‌ام.

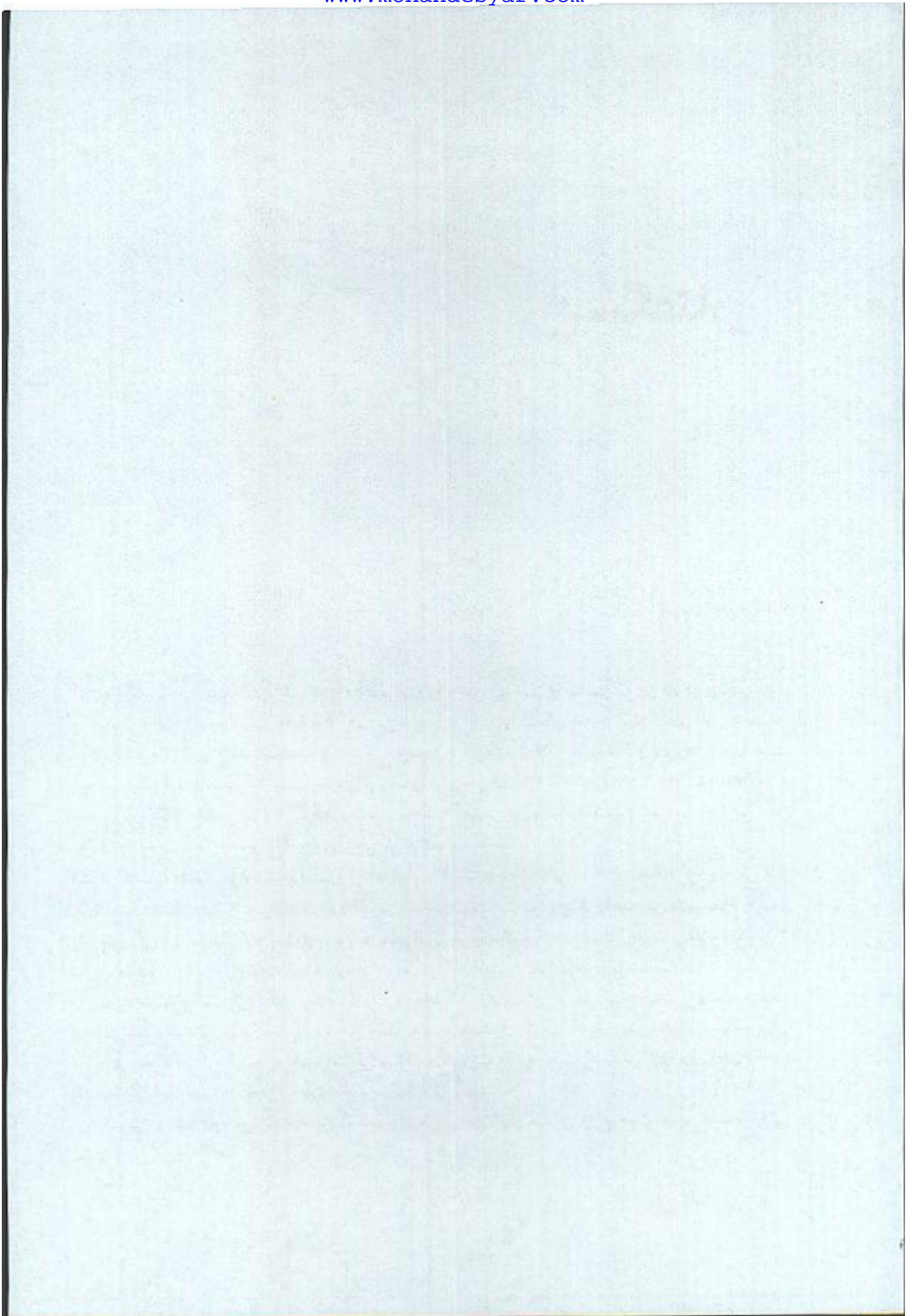
اکنون کتاب به چاپ دوم رسیده است و بر آن شدم تا نکات و یا مشکلاتی را که در هنگام کار با میکروهای AVR ممکن است برای دانشجویان به وجود آید در ادامه پیوست تحت عنوان **چند راهنمایی مهم** مطرح کرده و راه حلی برای آنها پیشنهاد کنم. امیدوارم، نوانسته باشم قدمی هر چند کوچک در جهت برانگیختن علاقه دانشجویان عزیز بردارم و از دانش‌پژوهان و اساتید محترم می‌خواهم مرا از انتقادات و نظرات خود بهره‌مند سازند.

در ادامه لازم است از اساتید محترم گروه برق دانشگاه آزاد اسلامی واحد کرج مهندس بهنام، مهندس فرجادی‌نسب، مهندس حاجی‌حسینی، مهندس حسین زاده و همکاران گرامی آقایان مهدی قلخانی، مهندس امیر محمدی، محمد عبدلی، فرشید پیراهن‌سیاه، علی سرلک و مدیریت محترم سایت کامپیوتر گروه برق آقای برازنده که در تالیف این کتاب به نحوی مرا یاری دادند سپاسگزاری کنم.

همچنین بر خود لازم می‌دانم از مدیریت محترم انتشارات نص آقای مهندس زارع جهت چاپ کتاب قدردانی

نمایم.

علی کاهه - پاییز ۸۳



مقدمه

مقدمه

کتابی که در حال خواندن آن هستید در مورد انواع میکروکنترلرهای 8 بیتی AVR است که سعی شده است 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 بتوانند از آن استفاده نمایند.

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 MEGA AVR 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 MEGA AVR 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 مختلف یک نوع PROGRAMMER 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 AVRها 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 حافظه‌های سریال 2-WIRE ارائه شده است.

در نهایت مدار و برنامه چند پروژه عملی نیز در فصل هشتم آمده است. خطاهای میکروکنترلرهای AVR در ویرایشهای مختلف به همراه کدهای 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 REDUCED 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 غیرفرار شرکت ATMEL برای برنامه‌ریزی AVRها 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.

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

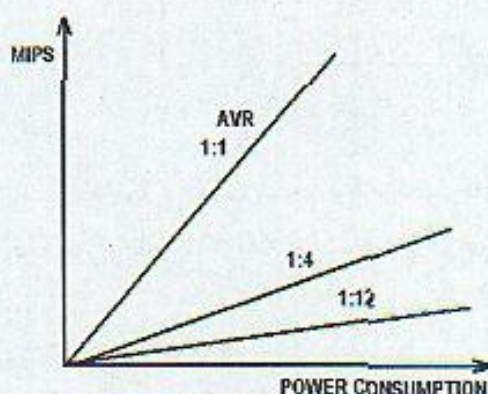
عملیات تک سیکل

با انجام تک سیکل دستورات، کلاک اسلاتور با کلاک داخلی سیستم یکی می‌شود. هیچ تقسیم کننده‌ای در داخل AVR قرار ندارد که ایجاد اختلاف فاز کلاک کند. اکثر میکروها کلاک اسلاتور به سیستم را با نسبت 1:4 یا 1:12 تقسیم می‌کنند که خود باعث کاهش سرعت

۷ مقدمه

می‌شود. بنابراین AVR ها 4 تا 12 بار سریعتر و مصرف آنها نیز 4 - 12 بار نسبت به میکروکنترلرهای مصرفی کنونی کمتر است زیرا در تکنولوژی CMOS استفاده شده در میکروهای AVR، مصرف توان سطح منطقی متناسب با فرکانس 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 افزایش
در MIPS/POWER Consumption
AVR با دیگر میکروکنترلرها

طراحی برای زبان‌های BASIC و C

زبانهای BASIC و C بیشترین استفاده را در دنیای امروز بعنوان زبانهای HLL دارند. تا امروزه 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 متغیر محلی به جای متغیر سراسری در داخل زیربرنامه تعریف کرد، در این صورت فقط در زمان اجرای زیربرنامه مکانی از حافظه RAM برای متغیر اشغال می‌شود در صورتی که اگر متغیری به عنوان سراسری تعریف گردد در تمام وقت مکانی از حافظه FLASH ROM را اشغال کرده است.

برای دسترسی سریعتر به متغیرهای محلی و کاهش کد، نیاز به افزایش رجیسترهای همه منظوره است. AVR ها دارای 32 رجیستر هستند که مستقیماً به LOGIC ALU (ARITHMETIC 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 صفحه بعد نشان می‌دهد که چگونه تعداد مناسب رجیسترهای همه منظوره (در AVR ها) می‌توانند با معماری CISC با یک ACCUMULATOR مقایسه گردند. برای این منظور می‌خواهیم از معادله صفحه بعد A را بدست بیاوریم. می‌بینیم که با کدهای AVR این محاسبه در عرض 4 کلاک سیکل و با کدهای CISC در عرض 48-96 کلاک سیکل انجام می‌گیرد.

Function: $A = ((A \text{ .and. } 84h) + (B \text{ .eor. } C)) \text{ .or. } 80h$

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

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

ارتباط با مؤلف

شما می‌توانید از طریق آدرس زیر و یا انشادات نص با مؤلف ارتباط برقرار کنید:

E-MAIL : alikaheh@yahoo.com

فهرست مطالب

فصل ۱: میکروکنترلرهای TINYAVR

۱۶	۱-۱ خصوصیات ATTINY12, ATTINY11, ATTINY10
۱۷	فیوز بیت‌های ATTINY12 و ATTINY11
۱۹	منابع کلاک ATTINY11/12
۲۱	۲-۱ خصوصیات ATTINY15L
۲۲	فیوز بیت‌های ATTINY15L
۲۳	۳-۱ خصوصیات ATTINY26L, ATTINY26
۲۵	فیوز بیت‌های ATTINY26
۲۷	کلاک سیستم ATTINY26
۳۳	۴-۱ خصوصیات ATTINY28L, ATTINY28V
۳۵	فیوز بیت‌های ATTINY28
۳۵	منابع کلاک ATTINY28

فصل ۲: میکروکنترلرهای AVR

۳۸	۱-۲ خصوصیات AT90S1200
۳۹	فیوز بیت‌های AT90S1200
۳۹	۲-۲ خصوصیات AT90S2313
۴۱	فیوز بیت‌های AT90S2313
۴۱	۲-۲ خصوصیات AT90S2323/LS2323/S2343/LS2343
۴۳	فیوز بیت‌های AT90S/LS2323
۴۳	فیوز بیت‌های AT90S/LS2343
۴۴	۳-۲ خصوصیات AT90S2333/LS2333/S4433/LS4433
۴۵	فیوز بیت‌های AT90S2333/4433
۴۷	۴-۲ خصوصیات AT90S8515
۴۸	فیوز بیت‌های AT90S8515
۴۹	۵-۲ خصوصیات AT90S8535/LS8535
۵۱	فیوز بیت‌های AT90S8535

فصل ۳: میکروکنترلرهای MEGA AVR

۵۴	۱-۳ خصوصیات ATMEGA323, ATMEGA323L
۵۵	فیوز بیت‌های ATMEGA323
۵۸	۲-۳ خصوصیات ATMEGA32L, ATMEGA32
۶۰	فیوز بیت‌های ATMEGA32
۶۳	۳-۳ خصوصیات ATMEGA128, ATMEGA128L
۶۴	فیوز بیت‌های ATMEGA128

۶۷ ۴-۳ خصوصیات ATMEGA163L , ATMEGA163
۶۹ فیوز بیت های ATMEGA163
۷۱ ۵-۳ خصوصیات ATMEGA8L , ATMEGA8
۷۳ فیوز بیت های ATMEGA8
۷۵ ۶-۳ خصوصیات ATMEGA8515L , ATMEGA8515
۷۷ فیوز بیت های ATMEGA8515
۷۹ ۷-۳ خصوصیات ATMEGA8535L , ATMEGA8535
۸۱ فیوز بیت های ATMEGA8535
۸۳ ۸-۳ خصوصیات ATMEGA161L , ATMEGA161
۸۵ فیوز بیت های ATMEGA161
۸۶ ۹-۳ خصوصیات ATMEGA162V , ATMEGA162
۸۸ فیوز بیت های ATMEGA162
۹۰ ۱۰-۳ خصوصیات ATMEGA16 , ATMEGA161
۹۲ فیوز بیت های ATMEGA16
۹۴ ۱۱-۳ خصوصیات ATMEGA103L , ATMEGA103
۹۶ فیوز بیت های ATMEGA103
۹۶ ۱۲-۳ خصوصیات ATMEGA169 , ATMEGA169L , ATMEGA169V
۹۸ فیوز بیت های ATMEGA169
۱۰۰ ۱۳-۳ خصوصیات ATMEGA64L , ATMEGA64
۱۰۲ فیوز بیت های ATMEGA64
۱۰۴ ۱۴-۳ کلاک سینم (۱)
۱۱۰ ۱۵-۳ کلاک سینم (۲)

فصل ۴ : محیط برنامه نویسی BASCOM AVR

۱۱۸ ۱-۴ معرفی متوهای محیط BASCOM
۱۲۳ ۲-۴ معرفی محیط شبیه سازی (SIMULATOR)
۱۲۶ ۳-۴ معرفی محیط برنامه ریزی
۱۲۸ ۴-۴ معرفی محیط TERMINAL EMULATORE
۱۲۹ ۵-۴ ساخت ISP PROGRAMMER

فصل ۵ : دستورات و توابع محیط برنامه نویسی BASCOM

۱۳۲ ۱-۵ بدنه یک برنامه در محیط BASCOM
۱۳۲ معرفی میکرو، کریستال، اسمبلی و بیسیک
۱۳۳ یادداشت و آدرس شروع برنامه ریزی حافظه FLASH
۱۳۴ تعیین کلاک و پایان برنامه
۱۳۴ ۲-۵ اعداد و متغیرها و جداول LOOKUP
۱۳۴ دیمانیون متغیر
۱۳۶ دستورات CHR , CONST و ALIAS
۱۳۷ دستورات INSTR , INCR , DECR و CHECKSUM

فهرست مطالب ۱۱

۱۳۸	دستورات RIGHT, LCASE, UCASE, HIGHT, LOW
۱۳۹	دستورات LTRIM, LEFT, LEN
۱۴۰	دستورات ROTATE و SWAP, MID
۱۴۱	دستور FUSING, FORMAT و توابع
۱۴۲	جداول LOOKUP و LOOKUPSTR
۱۴۳	۳-۵ توابع ریاضی و محاسباتی
۱۴۳	عملگرهای ریاضی و منطقی و تابع ABS
۱۴۴	توابع LOG و LOG10, EXP
۱۴۵	توابع SIN و RND, TAN, COS
۱۴۶	توابع SINH و TANH, COSH
۱۴۷	توابع ACOS و ASIN, ATN, ATN2
۱۴۸	تابع DEG2RAD
۱۴۹	توابع ROUND و RAD2DEG
۱۴۹	۴-۵ توابع تبدیل کدها و متغیرها
۱۴۹	توابع HEX و ASC
۱۵۰	توابع MAKEINT, MAKEBCD, HEXVAL, MAKEDEC
۱۵۱	توابع BIN2GREY و GRAY2BIN, VAL, STR, STRING
۱۵۲	۵-۵ رجیسترها و آدرس‌های حافظه
۱۵۲	دستور RESET و TOGGLE, SET
۱۵۳	دستور LOADADR و CPEEK, CPEEKH, BITWAIT
۱۵۴	دستور INP و PEEK, OUT
۱۵۵	دستور VARPTR و POKE
۱۵۵	۶-۵ دستورالعمل‌های حلقه و پرش
۱۵۵	دستورالعملهای JMP و GOTO
۱۵۶	دستورالعملهای FOR-NEXT و WHILE-WEND, DO-LOOP
۱۵۷	دستورالعمل IF
۱۵۸	دستورالعمل CASE
۱۵۹	دستورالعمل ON VALUE و EXIT
۱۶۰	۷-۵ ایجاد تاخیر در برنامه
۱۶۰	دستورات WAITMS و WAIT, WAITUS, DELAY
۱۶۱	۸-۵ زیربرنامه و تابع
۱۶۱	معرفی تابع و زیربرنامه
۱۶۲	فراخوانی یا CALL
۱۶۳	بکارگیری متغیر محلی یا LOCAL
۱۶۴	پرش به زیربرنامه توسط دستور GOSUB

فصل ۶: پیکره‌بندی و کار با امکانات AVR در BASCOM

۱۶۶	۱-۶ پیکره‌بندی پورت‌ها
۱۶۶	بررسی پورت‌های میکرو ATMEGA32
۱۶۶	پورت A
۱۶۸	پورت B

۱۷۰	پورت C
۱۷۳	پورت D
۱۷۵	۲-۶ پیکره‌بندی صفحه‌کلید 4x4
۱۷۶	دستور GETKBD()
۱۷۷	۳-۶ پیکره‌بندی صفحه‌کلید کامپیوتر
۱۷۸	دستور GETATKBD
۱۷۹	۴-۶ پیکره‌بندی LCD
۱۷۹	تعیین نوع LCD
۱۸۰	تعیین باس LCD و دستور LCD
۱۸۱	دستورات CURSOR و DISPLAY, CLS
۱۸۲	دستورات SHIFTLCD و SHIFT CURSOR, HOME, LOCATE
۱۸۳	دستورات DEFLCDCHAR و تابع THIRDLINE, LOWERLINE, FOURTH LINE, UPPERLINE
۱۸۴	۵-۶ پیکره‌بندی تایمر/کانترها
۱۸۴	معرفی تایمر/کانتر صفر و رجیسترها
۱۸۵	پیکره‌بندی تایمر/کانتر صفر در محیط BASCOM
۱۸۷	معرفی تایمر/کانتر یک و رجیسترها
۱۹۳	پیکره‌بندی تایمر/کانتر یک در محیط BASCOM
۱۹۹	معرفی تایمر/کانتر دو رجیسترها
۲۰۲	پیکره‌بندی تایمر/کانتر دو در محیط BASCOM
۲۰۸	۶-۶ ارتباط با پورت سریال
۲۰۸	UART سخت‌افزاری
۲۰۸	تعیین میزان باود
۲۰۹	تغییر میزان باود
۲۰۹	ارسال داده در حالت UART سخت‌افزاری
۲۰۹	پیکره‌بندی SERIALOUT و دستور PRINT
۲۱۰	دستور PRINTBIN
۲۱۰	دریافت داده در حالت UART سخت‌افزاری
۲۱۰	پیکره‌بندی SERIALIN
۲۱۱	دستورات INPUT و INKEY, WAITKEY
۲۱۲	دستورات INPUTHEX و INPUTBIN
۲۱۳	UART نرم‌افزاری
۲۱۲	تعیین و تغییر میزان باود
۲۱۵	ارسال داده در حالت UART نرم‌افزاری
۲۱۵	دستورات PRINT و PRINTBIN
۲۱۶	دریافت داده در حالت UART نرم‌افزاری
۲۱۶	دستورات INPUT و INKEY, WAITKEY
۲۱۷	دستورات INPUTHEX و INPUTBIN
۲۱۸	اتصال AVR به RS232
۲۱۸	نراشه MAX232, MAX233 و مبدل نوانزیستوری منطق TTL و RS-232 به یکدیگر
۲۱۹	۷-۶ مبدل آنالوگ به دیجیتال (ANALOG TO DIGITAL CONVERTOR)
۲۲۱	پیکره‌بندی ADC در محیط BASCOM

فهرست مطالب ۱۳

۲۲۲ دستور GETADC
۲۲۳ کار با وقفه ADC
۲۲۴ ۸-۶ مقایسه کننده آنالوگ
۲۲۴ پیکره بندی مقایسه کننده آنالوگ در BASCOM
۲۲۵ ۹-۶ پیکره بندی GRAPHICAL LCD DISPLAY
۲۲۷ دستورات CURSOR , LOACTE , PSET , LCD , CLS TEXT , CLS GRAPH , CLS
۲۲۸ دستورات SBGF , SHOWPIC , CIRCLE , LINE و پرچسب
۲۲۹ ۱۰-۶ ارتباط سریال SPI
۲۳۱ ارتباط SPI و رجیسترهای مربوطه
۲۳۳ پیکره بندی SPI در محیط BASCOM
۲۳۴ پیکره بندی سخت افزاری و نرم افزاری SPI
۲۳۵ دستورات مربوط به ارتباط SPI
۲۳۵ دستورات SPIMOVE و SPIOUT , SPIIN , SPIINIT
۲۳۶ ۱۱-۶ پیکره بندی ارتباط سریال I2C (2-WIRE)
۲۳۶ تعیین کلاک I2C
۲۳۷ تعیین پایه SDA و SCL و دستور I2C RECEIVE
۲۳۸ دستورات I2CSTART , I2CSTOP , I2CRBYTE , I2CWBYTE , I2C SEND
۲۳۸ ۱۲-۶ پیکره بندی WATCHDOG
۲۳۹ ۱۳-۶ وقفه ها
۲۳۹ دستورات ENABLE , DISABLE
۲۴۰ دستور ON INTERRUPT و پیکره بندی وقفه های خارجی
۲۴۱ ۱۴-۶ حافظه EEPROM داخلی میکروهای AVR
۲۴۲ دستورات READEEPROM و WRITEEEPROM
۲۴۳ ۱۵-۶ مدهای SLEEP
۲۴۳ معرفی انواع مدهای SLEEP
۲۴۳ مدهای IDLE و ADC NOISE REDUCTION
۲۴۴ مدهای EXTENDED -STANDBY , STANDBY , POWER- SAVE , POWER-DOWN
۲۴۴ دستورات اجرای مدهای SLEEP در BASCOM
۲۴۶ ۱۶-۶ کار با حافظه BOOT (BOOT LOADER FLASH SECTION)
۲۴۶ وارد شدن به برنامه BOOTLOADER
۲۴۶ کار با BOOTLOADER در محیط BASCOM
۲۴۷ ۱۷-۶ دستورات و پیکره بندی های جانبی
۲۴۷ دستور DEBOUNCE
۲۴۸ دستور PULSEOUT
۲۴۹ دستور PULSEIN
۲۵۰ دستور SOUND

فصل ۷: حافظه های EEPROM سریال 2-WIRE

۲۵۱ ۱-۷ معرفی انواع حافظه های سری AT24
۲۵۳ ۲-۷ معرفی پایه ها

۲۵۴	۳-۷ طرز کار حافظه
۲۵۵	۴-۷ آدرس دهی سخت افزاری حافظه
۲۵۶	۵-۷ انواع عملیات نوشتن حافظه
۲۵۷	۶-۷ انواع عملیات خواندن حافظه

فصل ۸: پروژهای عملی

۲۶۰	۸-۱ اتصال کلید به میکرو توسط دستور DEBOUNCE
۲۶۱	۸-۲ اسکن صفحه کلید ۴×۴
۲۶۲	۸-۳ اسکن صفحه کلید ۴×۴ توسط انکدر MM74C922
۲۶۴	۸-۴ اسکن صفحه کلید کامپوتر
۲۶۶	۸-۵ برنامه ساعت
۲۶۷	۸-۶ تابلو روان توسط LCD
۲۶۹	۸-۷ فرکانس متر دیجیتال
۲۷۰	۸-۸ کار با محیط TERMINAL EMULATOR
۲۷۲	۸-۹ نمایش دما بر روی LCD توسط سنسور دمای LM35
۲۷۴	۸-۱۰ مانیتورینگ دما توسط نرم افزار VISUAL BASIC 6.0
۲۷۶	۸-۱۱ موتور پله ای STEPER MOTOR
۲۷۹	۸-۱۲ ارتباط سریال دو میکرو از طریق بوس SPI
۲۸۲	۸-۱۳ ارتباط با حافظه EEPROM سریال AT24C256
۲۸۶	۸-۱۴ ساخت پالس PWM توسط VISUAL BASIC
۲۸۷	۸-۱۵ مقایسه کننده آنالوگ و CAPTURE تایمر یک
۲۸۸	۸-۱۶ RTC (REAL TIME CLOCK)
۲۹۰	۸-۱۸ LCD گرافیکی

فصل ۹: پیوست

۲۹۲	خطاهای میکروکنترلرهای AVR
۳۰۹	AVR Instruction Set Table
۳۱۳	BASCOM Editor Keys
۳۱۴	reserved BASCOM statements or character
۳۱۵	Error Codes In bascom AVR
۳۱۶	چند راهنمای مهم
۳۲۰	غلطنامه



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 کاهش صرف وقت برای پروژه‌های کاربران بهینه سازی شده‌اند. در زیر به طور نمونه تعدادی از کاربردهای انواع خانواده TINYAVR را می‌بینید.

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 می کند.

- کارایی بالا و توان مصرفی کم.
- دارای 90 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می شوند.
- 32×8 رجیستر کاربردی.
- سرعتی تا 8MIPS در فرکانس 8MHZ

• حافظه، برنامه و داده غیر فرار

- 1K بایت حافظه FLASH قابل برنامه ریزی داخلی.
- پایداری حافظه FLASH: قابلیت 1000 بار نوشتن و پاک کردن (WRITE / ERASE).
- 64 بایت حافظه EEPROM داخلی قابل برنامه ریزی.
- پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE).
- قفل برنامه FLASH و حفاظت داده EEPROM

• خصوصیات جانبی

- یک تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا.
- یک مقایسه گر آنالوگ داخلی.
- WATCHDOG قابل برنامه ریزی با اسیلاتور داخلی.
- وقفه در اثر تغییر وضعیت پایه.

• خصوصیات ویژه میکروکنترلر

- تغذیه کم در مدهای IDLE و POWERDOWN.
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- ارتباط سریال SPI برای برنامه ریزی 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 25°C، 3V، 4MHZ

- حالت فعال (ACTIVE MODE) 2.2mA
- در حالت بی کاری (IDLE MODE) 0.5mA
- در حالت POWER - DOWN: $1\mu A >$

• ولتاژهای عملیاتی (کاری)

- 1.5V تا 5.5V برای (ATtiny12V-1)

۱۷ TINYAVR میکروکنترلرهای

— 2.7V تا 5.5V برای (ATtiny12L-4 و ATtiny11L-2)

— 4V تا 5.5V برای (ATtiny12-8 و ATtiny11-6)

• فرکانسهای کاری

— 0MHZ تا 1.2MHZ برای (ATtiny12V-1)

— 0MHZ تا 2MHZ برای (ATtiny11L-2)

— 0MHZ تا 4MHZ برای (ATtiny12L-4)

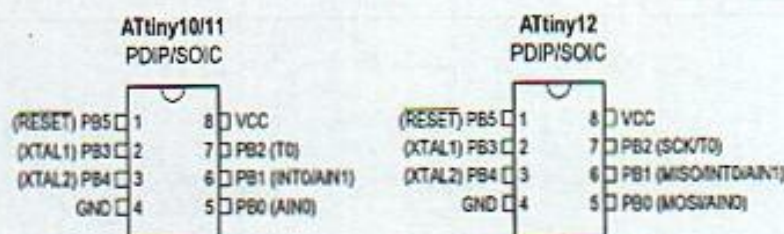
— 0MHZ تا 6MHZ برای (ATtiny11-6)

— 0MHZ تا 8MHZ برای (ATtiny12-8)

• انواع بسته‌بندی

— 8 پایه (PIN) در انواع PDIP و SOIC

• ترکیب بسته‌بندی



فیوز بیت‌های ATtiny12 و ATtiny11

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.

فیوز بیت‌های ATtiny11

این میکرو دارای 5 فیوز بیت (FSTRT , RSTDISBL , CKSEL2..0) به قرار زیر است:
FSTRT: این بیت با توجه به جدول زیر مشخص کننده زمان شروع (START-UP) از ریست یا بیدار شدن SLEEP است. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده (1) است.

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

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.

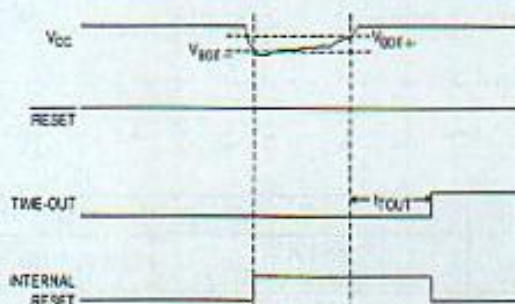
CKSEL2..0: با برنامه‌ریزی کردن این بیت‌ها کلاک سیستم را می‌توان با توجه به جدول زیر در مدهای مختلف تغییر داد. این بیت‌ها به صورت پیش‌فرض 100 هستند و میکرو با اسیلاتور RC داخلی ثابت با فرکانس 1.0MHZ کار می‌کند.

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

جدول انتخاب کلاک سیستم برای ATTINY11

فیوز بیت‌های ATTiny12

این میکرو دارای 8 فیوز بیت (**BODLEVEL**, **BODEN**, **SPIEN**, **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 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.



شکل ۱-۱

زمانبندی ری‌ست BROWN-OUT

BODEN: برای فعال کردن عملکرد مدار **BROWN - OUT** این بیت بایستی طبق جدول ۱-۱ 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.

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

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

CKSEL3..0 : با برنامه‌ریزی کردن این بیت‌ها و بیت BODLEVEL کلاک سیستم را می‌توان با توجه به جدول زیر در مدهای مختلف تغییر داد. این بیت‌ها به صورت پیش‌فرض 0010 هستند و میکرو با اسیلاتور RC داخلی ثابت 1.2MHZ کار می‌کند. کریستال خارجی بین پایه‌های XTAL1 و XTAL2 متصل می‌شود.

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
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

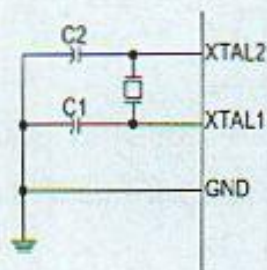
جدول انتخاب کلاک سیستم و زمان START-UP برای ATTINY12

منابع کلاک ATTINY11/12

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

اسیلاتور کریستالی (EXTERNAL CRYSTAL/CERAMIC RESONATOR)

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



شکل اتصال کریستال به میکرو در حالت
اسیلاتور کریستالی

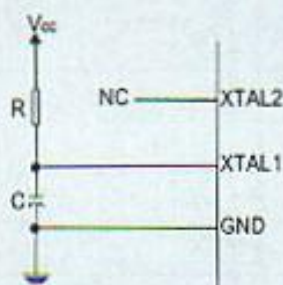
خازنهای C1 و C2 برای کریستال ها و نوسانگرها بایستی یک مقدار باشند. مقادیر خازنهای بستگی به کریستال ، نوسانگر و نویزهای الکترومغناطیسی محیط دارند که مقدار نامی 32P مناسب است.

اسیلاتور کریستالی فرکانس پایین (EXTERNAL LOW FREQ CRYSTAL)

برای استفاده از کریستال ساعت 32.768KHZ ، کریستال طبق شکل بالا به پایه های XTAL1 و XTAL2 متصل می شود.

اسیلاتور RC خارجی (EXT. RC OSCILLATOR)

اتصال RC به پایه های XTAL1 در شکل زیر آمده است. مقدار خازن بایستی حداقل 20PF و مقاومت باید در رنج 3K - 100K باشد. خازن و مقاومت سه فرکانس در جدول زیر آمده است.



شکل اتصال RC به میکرو در حالت
اسیلاتور RC خارجی

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

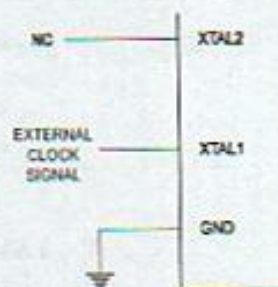
جدول مدهای کاری اسیلاتور RC خارجی

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 مد، کلاک خارجی به پایه XTAL1 طبق شکل زیر وصل می‌شود.



شکل اتصال کلاک خارجی به پایه میکرو
در حالت کلاک خارجی

۲-۱- 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 برنامه FLASH و حفاظت داده 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 تفاضلی ADC با کنترل گین 20x
- یک مقایسه‌گر آنالوگ داخلی
- WATCHDOG قابل برنامه‌ریزی با اسبیلانور داخلی

• خصوصیات ویژه میکروکنترلر

- تغذیه کم در مدهای 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
- امپلاتور 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 3V ، 1.6MHZ ، 25°C

- حالت فعال (ACTIVE MODE) 3mA
- در حالت بی‌کاری (IDLE MODE) 1mA
- در حالت POWER – DOWN : $1\mu A >$

• ولتاژهای عملیاتی (کاری)

— 2.7V تا 5.5V

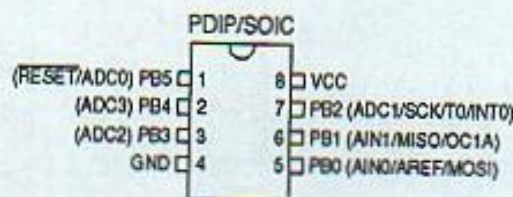
• فرکانسهای کاری

— 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



فیوز بیت‌های ATtiny15L

این میکرو دارای 6 فیوز بیت (BODLEVEL , BODEN , SPIEN , 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 سیستم را ری‌ست می‌کند ولی زمانی که بیت

۲۳ TINYAVR میکروکنترلرهای

برنامه‌ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین‌تر شود ری‌ست داخلی میکرو فعال شده و سیستم را طبق شکل ۱-۱ 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. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

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

جدول سطوح مختلف ولتاژ برای مدار BROWN-OUT

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 بیت‌ها کلاک سیستم را با توجه به جدول زیر می‌توان در مدهای مختلف تغییر داد. این بیت‌ها به صورت پیش‌فرض 00 هستند و میکرو با اسیلاتور RC داخلی 1.6MHz و زمان شروع طولانی $64\text{ ms} + 18\text{ CK}$ کار می‌کند.

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 μ s	18 CK + 8 μ s	BOD disabled
0	11	18 CK + 128 μ s	18 CK + 32 μ s	BOD enabled

انتخاب زمان RESET DELAY (X=DON'T CARE)

۳-۱ 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 داده غیر فرار

— 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 برنامه FLASH و حفاظت داده 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 (TIMER / COUNTER) 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.

— وقفه تغییر وضعیت بر روی 11 پایه.

• خصوصیات ویژه میکروکنترلر

— تغذیه کم در مدهای IDLE و POWERDOWN

— دارای مد 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 برای.

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• ولتاژهای عملیاتی (کاری)

— 2.7V تا 5.5V برای (ATtiny26L)

FUSE LOW BYTE			
FUSE HIGH 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)

ادامه جدول صفحه قبل

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.
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 سیستم را طبق شکل ۱-۱ 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 برنامه‌ریزی شده باشد. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است.

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

جدول سطوح مختلف ولتاژ برای مدار BROWN-OUT

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.

۲۷ میکروکنترلرهای TINYAVR

CKOPT: بیتی برای انتخاب کلاک که به صورت پیش فرض برنامه ریزی نشده است. عملکرد این بیت بستگی به بیت های CKSEL دارد که در بخش کلاک سیستم ATTINY26 آمده است.

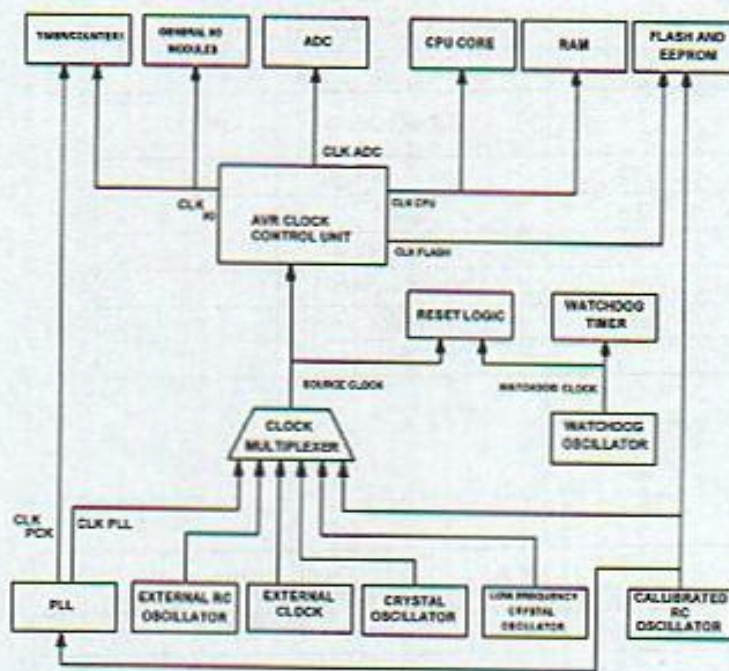
SUT1, SUT0: انتخاب زمان START-UP که عملکرد این دو بیت در بخش کلاک سیستم ATTINY26 کاملاً توضیح داده شده است.

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

کلاک سیستم ATTiny26

توزیع کلاک

کلاک در ATTINY26 طبق شکل زیر توزیع می شود.



شکل توزیع کلاک بخشهای مختلف در میکرو ATTINY26

کلاک CPU - CLKCPU

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

کلاک I/O - CLK I/O

این کلاک توسط بسیاری از ماژول های I/O به طور مثال تایمرها، کانترها و SPI استفاده می شود.

کلاک FLASH - CLKFLASH

این کلاک عملیات ارتباطی با حافظه FLASH را کنترل می کند. کلاک FLASH معمولاً با کلاک CPU فعال می شود.

کلاک ADC - CLKADC

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 (FLASH FUSE BITS) انواع آن را انتخاب کرد. کلاک انتخاب شده به عنوان ورودی کلاک AVR طبق جدول زیر در نظر گرفته شده و کلاک مناسب به هر قسمت سیستم داده می‌شود.

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

نکته

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

جدول انتخاب کلاک سیستم برای ATTINY26

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

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
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1100	XTAL1	XTAL2
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1101	XTAL1	XTAL2

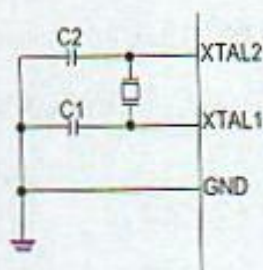
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1110	XTAL1	XTAL2
EXTERNAL CRYSTAL/RESONATOR OSCILLATOR	1	1111	XTAL1	XTAL2
PLL	0	0001	I/O	I/O

جدول انتخاب انواع کلاک سیستم و طرز کار پایه‌های PB4، PB5 برای ATTINY26 (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 زمانبندی‌های مذکور از اسیلاتور WATCHDOG استفاده می‌شود.

اسیلاتور کریستالی (EXTERNAL CRYSTAL/RESONATOR OSCILLATOR)

در این حالت کریستال یا نوسانگر سرامیکی (CERAMIC RESONATOR) یا کریستال کوارتز (QUARTZ CRYSTAL) همانطور که در شکل ۲-۱ نشان داده شده است به دو پایه XTAL1 و XTAL2 وصل می‌شود.



شکل ۲-۱ اتصال کریستال به میکرو
در حالت اسیلاتور کریستالی

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 زیر آمده است. برای نوسانگرهای سرامیکی مقدار خازن‌هایی که توسط کارخانه پیشنهاد می‌گردد بایستی استفاده شود. فیوز بیت CKOPT همیشه برنامه‌ریزی شده است.

CKOPT	CKSEL 3..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 - 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 مدهای عملیاتی اسیلاتور کریستالی

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

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 می‌تواند برای کریستال‌ها زمانی که در فرکانسهای پایین کار می‌کنند استفاده شوند.

جدول انتخاب زمان START-UP برای کلاک اسپلاتور کریستالی

اسپلاتور کریستالی فرکانس پایین (EXTERNAL LOW-FREQUENCY OSCILLATOR)

برای استفاده از کریستال ساعت 32.768KHZ، فیوز بیت‌های CKSEL با 1001 برنامه‌ریزی می‌شوند و کریستال طبق شکل ۱-۲ (صفحه قبل) 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 زیر قابل انتخاب است.

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		

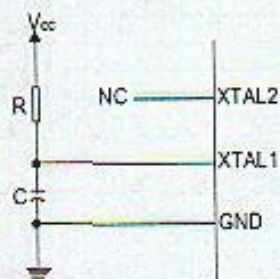
جدول انتخاب زمان START-UP برای کلاک اسپلاتور کریستالی فرکانس پایین

اسپلاتور RC خارجی (EXTERNAL RC OSCILLATOR)

اتصال RC به پایه‌های XTAL1 در شکل زیر آمده است. فرکانس تقریبی توسط معادله $f = 1 / (3RC)$

۳۱ میکروکنترلرهای TINYAVR

بدست می‌آید. مقدار خازن بایستی حداقل 22PF باشد. با برنامه‌ریزی کردن فیوز بیت CKOPT کاربر می‌تواند خازنهای داخلی 36PF را بین XTAL1 و GND و راه‌اندازی کند و در نتیجه دیگر نیازی به خازن خارجی نیست.



شکل 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 زیر قابل انتخاب است.

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 مُدهای عملیاتی اسیلاتور RC خارجی

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

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.

جدول انتخاب زمان START-UP برای کلاک اسیلاتور RC خارجی

اسیلاتور RC کالیبره شده داخلی (INTERNAL RC OSCILLATOR)

اسیلاتور RC کالیبره شده داخلی، کلاک‌های نامی داخلی 1، 2، 4 و 8MHZ را طبق جدول زیر در ولتاژ 5V و دمای 25°C 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. زمانی که از این مُد استفاده می‌گردد فیوز بیت CKOPT همیشه بایستی برنامه‌ریزی شده باشد.

حالت پیش فرض میکرو اسیلاتور RC داخلی 1MHz است.

نکته

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

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

جدول مدهای عملیاتی اسیلاتور RC کالیبره شده داخلی

هنگامی که فرکانس کاری انتخاب می شود، زمان شروع (START-UP) توسط فیوز بیت های SUT طبق جدول زیر قابل انتخاب است.

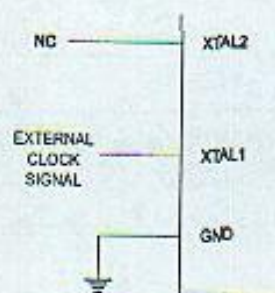
SUT 1..0	START-UP TIME FROM POWER-DOWN	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10 ⁽¹⁾	6 CK	65 ms	SLOWLY RISING POWER
11	RESERVED		

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

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

کلاک خارجی (EXTERNAL CLOCK)

برای راه اندازی میکرو در این مُد، کلاک خارجی به پایه XTAL1 بایستی طبق زیر وصل شود. برای کار در این مُد بیت های CKSEL با 0000 برنامه ریزی می شوند. با برنامه ریزی کردن فیوز بیت CKOPT خازن داخلی 36PF بین پایه های XTAL1 و GND فعال می شود.



شکل اتصال کلاک خارجی به پایه میکرو در حالت کلاک خارجی

هنگامی که این نوع کلاک انتخاب می شود، زمان شروع (START-UP) توسط فیوز بیت های SUT طبق جدول ۱-۱ (صفحه بعد) 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

۳۳ TINYAVR میکروکنترلرهای

است باعث رفتارهای غیرقابل انتظار میکرو شود. زمانی که قصد تغییر فرکانس کلاک را دارید بایستی میکرو در حالت RESET نگه داشته شود.

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 ۱-۱ انتخاب زمان START-UP برای کلاک خارجی

کلاک فرکانس بالای PLL

مدار PLL داخلی در این مُد، فرکانس 64MHZ را تولید می‌کند که می‌تواند بعنوان کلاک تایمرکنترلر ۱ و تقسیمی از آن بعنوان کلاک سیستم استفاده شود. در صورت برنامه‌ریزی فیوز بیت PLLCK و فیوز بیت‌های CKSEL فرکانس PLL بر 4 تقسیم شده و بعنوان 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 زیر تنظیم می‌شود.

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

انتخاب زمان START-UP برای کلاک PLL داخلی

۴-۱ 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 داده غیر فرار
 - 2K بایت حافظه FLASH قابل برنامه‌ریزی داخلی.
 - قفل برنامه FLASH.

• خصوصیات جانبی

- یک تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا.
- یک مقایسه گر آنالوگ داخلی.
- WATCHDOG قابل برنامه ریزی با اسیلاتور داخلی.

• خصوصیات ویژه میکروکنترلر

- تغذیه کم در مدهای 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 2V، 1MHz، 25°C

- حالت فعال (ACTIVE MODE) 3mA
- در حالت بی‌کاری (IDLE MODE) 1.2mA
- در حالت POWER - DOWN : $1\mu A >$

• ولتاژهای عملیاتی (کاری)

- 1.8 V تا 5.5V برای (ATtiny28V)
- 2.7V تا 5.5V برای (ATtiny28L)

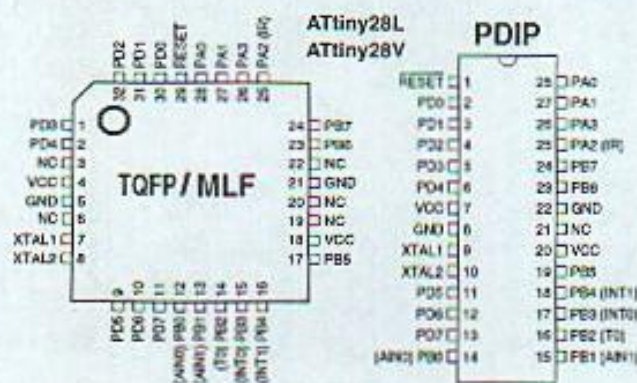
• فرکانسهای کاری

- 0MHz تا 1.2MHz برای (ATtiny28V)
- 0MHz تا 4MHz برای (ATtiny28L)

• خطوط I/O و انواع بسته‌بندی

- 11 خط ورودی / خروجی (I/O) قابل برنامه ریزی.
- 28 پایه (PIN) نوع PDIP، 32 پایه نوع TQFP و 32 پایه نوع MLF.

• ترکیب پایه‌ها



فیوز بیت‌های ATTiny28

ATTINY28 دارای ۵ فیوزبیت است. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تأثیری نمی‌بینند. در تمام توضیحات زیر ۰ به معنای برنامه‌ریزی شدن و ۱ 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 استفاده می‌شود. این بیت‌ها به صورت پیش‌فرض ۰۰۱۰ هستند و میکرو با اسیلاتور RC داخلی با مقدار نامی ۱.۲MHz و زمان شروع طولانی کار می‌کند.

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

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

منابع کلاک ATTINY28

اسیلاتور کریستالی (EXTERNAL CRYSTAL/CERAMIC RESONATOR)

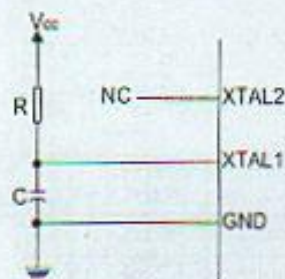
در این حالت کریستال یا نوسانگر سرامیکی (CERAMIC RESONATOR) یا کریستال کوارتز (QUARTZ CRYSTAL) همانطور که در شکل ۱-۲ نشان داده شده است به دو پایه XTAL1 و XTAL2 وصل می‌شود. خازنهای C1 و C2 برای کریستال‌ها و نوسانگرها بایستی یک مقدار باشند. مقادیر خازنها بستگی به کریستال، نوسانگر و نویزهای الکترومغناطیسی محیط دارند که مقدار نامی 32PF مناسب است.

اسیلاتور کریستالی فرکانس پایین (EXTERNAL LOW FREQ CRYSTAL)

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

اسیلاتور RC خارجی (EXT. RC OSCILLATOR)

اتصال RC به پایه‌های XTAL1 در شکل زیر کشیده شده است. مقدار خازن بایستی حداقل 20PF باشد و مقاومت باید در محدوده 3K - 100K باشد. خازن و مقاومت سه فرکانس در جدول زیر آمده است.



شکل اتصال RC به میکرو در حالت
اسیلاتور RC خارجی

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

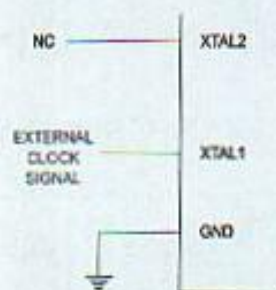
جدول مدهای کاری اسیلاتور RC خارجی

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 زیر بایستی وصل شود.



شکل اتصال کلاک خارجی به پایه
میکرو در حالت کلاک خارجی



میکروکنترلرهای AVR

این 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
۴. توانایی برنامه‌ریزی فیوز بیت‌ها برای تعیین کلاک سیستم دلخواه

— 0MHz تا 4MHz پر ای (AT90S1200-4)

(AT90S1200-12) 12MHZ 0MHZ —

— 15 خط ورودی / خروجی (I/O) قابل برنامه ریزی.

— 20 پایه (PIN) در انواع PDIP ، SOIC و SSOP

AT90S1200

POIPISOMISSOP



این میکرو دارای 2 فیوز بیت (RCEN , SPIEN) است. فیوز بیت ها با پای کردن (ERASE) میکرو اثری نمی بینند.

در تمام توضیحات زیر 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت است.

فیوزیت‌های AT90S1200 در زمان برنامه‌ریزی به صورت سریال قابل دسترس نمی‌باشند.

نکتہ

SPIEN: در حالت بشرفض 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 به کر یستال خارجی نیست. این بیت به صورت پیش فرض برنامه‌ریزی نشده است.

• از معماری AVR RISC استفاده می‌کند.

— کارایی بالا و توان مصرفی کم.

— دارای ۱۱۸ دسته و اعمال با کارهای بالا که اکثر آنها در یک کلاهی سبکی اجرا می‌شوند.

— 32*8 رجیستر کاربیدی.

— سرعتی تا 10MIPS در فرکانس 10MHZ

۴۰ میکروکنترلرهای AVR

• حافظه، برنامه و داده 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 برنامه FLASH و حفاظت داده 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 مدهای CAPTURE، COMPARE و PWM، 8، 9 یا 10 بیتی.
- یک مقایسه‌کننده آنالوگ داخلی.
- WATCHDOG قابل برنامه‌ریزی با اسپلاتور داخلی.
- ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING).
- UART دوطرفه (FULL DUPLEX).

• خصوصیات ویژه میکروکنترلر

- تغذیه کم در مدهای IDLE و POWERDOWN.
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- عملکرد کاملاً ثابت.
- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• توان مصرفی در 25°C، 3V، 4MHZ

- حالت فعال (ACTIVE) 2.8mA.
- در حالت بی‌کاری (IDLE) 0.8mA.
- در حالت POWER - DOWN : $1\mu A >$

• 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

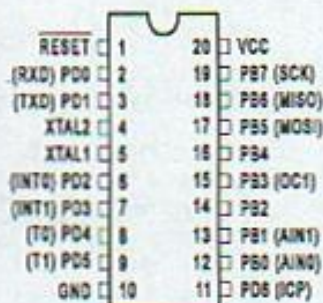
— 15 خط ورودی / خروجی (I/O) قابل برنامه‌ریزی.

— 20 پایه (PIN) در انواع PDIP، SOIC

• ترکیب پایه‌ها

AT90S2313

PDIP/SOIC



فیوز بیت‌های AT90S2313

این میکرو دارای 2 فیوز بیت (RCEN، 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.

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 مدهای SLEEP در نظر گرفته می‌شود. این بیت به صورت پیش‌فرض برنامه‌ریزی نشده است و طولانی‌ترین زمان در نظر گرفته شده است. جدول زیر مشخص کننده این زمان است.

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

جدول Reset Characteristics برای AT90S2313

۳-۲ 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 داده غیر فرار

- 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 برنامه FLASH و حفاظت داده 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 مدهای 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 3V ، 4MHZ ، 25°C

- حالت فعال (ACTIVE) 2.4mA
- در حالت بی‌کاری (IDLE) 0.5mA
- در حالت POWER - DOWN : $1\mu A >$

• 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



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 زیر مشخص کننده این زمان است.

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

جدول Reset Characteristics برای AT90S/LS2323

فیوز بیت‌های AT90S/LS2343

این میکرو دارای 2 فیوز بیت (SPIEN, RCEN) است. فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری

۴۴ میکروکنترلرهای AVR

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 اسلاتور 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 داده غیر فرار

— 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 برنامه FLASH و حفاظت داده EEPROM

• خصوصیات دیگر

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

— یک تایمر / کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مدهای

CAPTURE، COMPARE و PWM، 8، 9 یا 10 بیتی.

— یک مقایسه‌کننده آنالوگ داخلی.

— WATCHDOG قابل برنامه‌ریزی با یک اسلاتور داخلی.

— UART دوطرفه (FULL DUPLEX)

— 6 کانال مبدل آنالوگ به دیجیتال 10 بیتی ADC.

— 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 درمدهای IDLE و POWERDOWN.
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- عملکرد کاملاً ثابت.
- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• توان مصرفی در 25°C ، 3V ، 4MHZ

- حالت فعال 3.4mA (ACTIVE).
- درحالت بی‌کاری 1.4mA (IDLE).
- در حالت POWER - DOWN : $I_{\mu A} >$

• ولتاژهای عملیاتی (کاری)

- 2.7V تا 6V برای (AT90LS2333/AT90LS4433)
- 4V تا 6V برای (AT90S2333/AT90S4433)

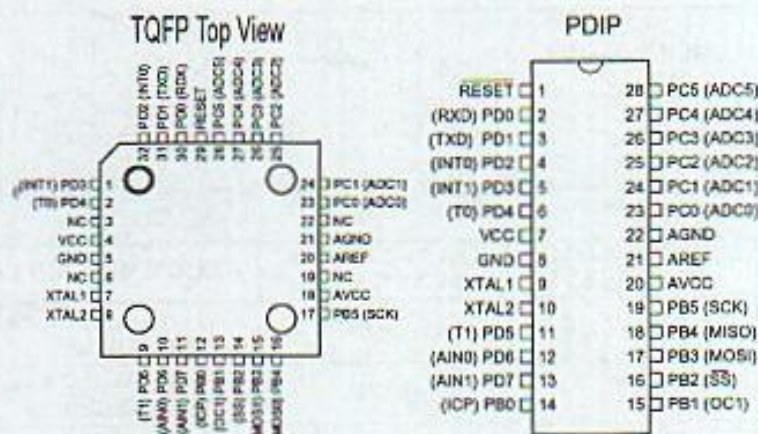
• فرکانسهای کاری

- 0MHZ تا 4MHZ برای (AT90LS2333/AT90LS4433)
- 0MHZ تا 8MHZ برای (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



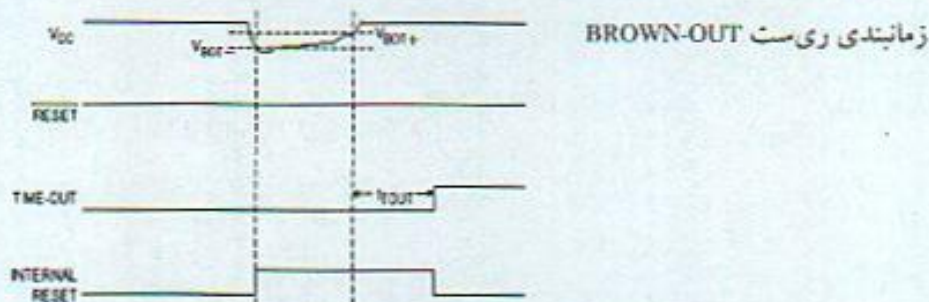
فیوز بیت‌های AT90S2333/4433

این میکروها دارای 6 فیوز بیت (BODLEVEL , BODEN , SPIEN , 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 پیش فرض برنامه ریزی شده و میکرو از طریق سریال SPI برنامه ریزی می شود. این بیت در زمان برنامه ریزی سریال قابل دسترس نمی باشد.

BODLEVEL: زمانی که این بیت برنامه ریزی نشده (پیش فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین تر شود ریست داخلی میکرو فعال شده و سیستم را ریست می کند و زمانی که این بیت برنامه ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین تر شود ریست داخلی میکرو فعال شده و طبق شکل زیر سیستم را ریست می کند.

لازم به تذکر است که این بیت به همراه بیت های CKSEL2..0 زمان شروع (START UP) میکرو را نیز تعیین می کند.



BODEN: برای فعال کردن عملکرد مدار BROWN-OUT این بیت بایستی برنامه ریزی شده باشد. این بیت به صورت پیش فرض برنامه ریزی نشده است.

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

جدول سطوح مختلف ولتاژ برای مدار BROWN-OUT

CKSEL2..0: با برنامه ریزی کردن این بیت ها کلاک سیستم را می توان با توجه به جدول زیر در مدهای مختلف تغییر داد. این بیت ها به صورت پیش فرض 010 هستند.

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

جدول انتخاب زمان RESET DELAY (ادامہ جدول صفحہ قبل)

۵-۲ 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 داده غیر فرار
 - 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 برنامه FLASH و حفاظت داده 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 مدهای CAPTURE، COMPARE 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 مدهای IDLE و POWERDOWN.
 - منابع وقفه (INTERRUPT) داخلی و خارجی.
 - عملکرد کاملاً ثابت.
 - توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS
- توان مصرفی در 25°C، 3V، 4MHZ
 - حالت فعال (ACTIVE MODE) 3.0mA

— در حالت بی کاری 1mA (IDLE MODE).

— در حالت POWER - DOWN : $1\mu A >$

• ولتاژهای عملیاتی (کاری)

— 2.7V تا 6V برای (AT90S8515-4)

— 4V تا 6V برای (AT90S8515-8)

• فرکانسهای کاری

— 0MHZ تا 4MHZ برای (AT90S8515-4)

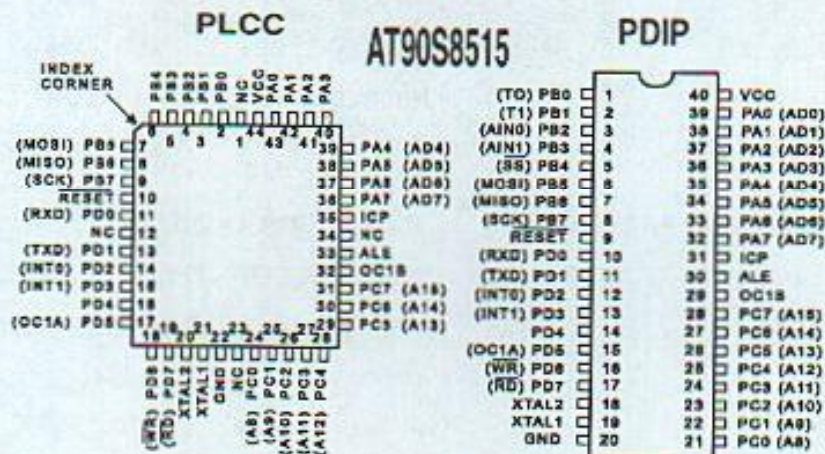
— 0MHZ تا 8MHZ برای (AT90S8515-8)

• خطوط I/O و انواع بسته بندی

— 32 خط ورودی / خروجی (I/O) قابل برنامه ریزی.

— 40 پایه (PIN) نوع PDIP، 44 پایه نوع PLCC و TQFP

• ترکیب پایه ها



فیوز بیت های AT90S8515

این میکرو دارای 2 فیوز بیت (FSTRT , SPIEN) است. فیوز بیت ها با پاک کردن (ERASE) میکرو تاثیری نمی بینند. در تمام توضیحات زیر 0 به معنای برنامه ریزی شدن و 1 به معنای برنامه ریزی نشدن بیت است.

فیوز بیت های AT90S8515 در زمان برنامه ریزی به صورت سریال قابل دسترس نمی باشند.

نکته

SPIEN : در حالت پیش فرض برنامه ریزی شده و میکرو از طریق سریال SPI برنامه ریزی می شود.
FSTRT : با برنامه ریزی کردن این بیت کوتاه ترین زمان شروع (START UP) از ریست و مدهای SLEEP در نظر گرفته می شود، این بیت به صورت پیش فرض برنامه ریزی نشده و طولانی ترین زمان در

نظر گرفته شده است. جدول زیر مشخص کننده این زمان است.

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

جدول انتخاب زمان RESET DELAY

۶-۲-۳-۴-۵-۶-۷-۸-۹-۱۰-۱۱-۱۲-۱۳-۱۴-۱۵-۱۶-۱۷-۱۸-۱۹-۲۰-۲۱-۲۲-۲۳-۲۴-۲۵-۲۶-۲۷-۲۸-۲۹-۳۰-۳۱-۳۲-۳۳-۳۴-۳۵-۳۶-۳۷-۳۸-۳۹-۴۰-۴۱-۴۲-۴۳-۴۴-۴۵-۴۶-۴۷-۴۸-۴۹-۵۰-۵۱-۵۲-۵۳-۵۴-۵۵-۵۶-۵۷-۵۸-۵۹-۶۰-۶۱-۶۲-۶۳-۶۴-۶۵-۶۶-۶۷-۶۸-۶۹-۷۰-۷۱-۷۲-۷۳-۷۴-۷۵-۷۶-۷۷-۷۸-۷۹-۸۰-۸۱-۸۲-۸۳-۸۴-۸۵-۸۶-۸۷-۸۸-۸۹-۹۰-۹۱-۹۲-۹۳-۹۴-۹۵-۹۶-۹۷-۹۸-۹۹-۱۰۰-۱۰۱-۱۰۲-۱۰۳-۱۰۴-۱۰۵-۱۰۶-۱۰۷-۱۰۸-۱۰۹-۱۱۰-۱۱۱-۱۱۲-۱۱۳-۱۱۴-۱۱۵-۱۱۶-۱۱۷-۱۱۸-۱۱۹-۱۲۰-۱۲۱-۱۲۲-۱۲۳-۱۲۴-۱۲۵-۱۲۶-۱۲۷-۱۲۸-۱۲۹-۱۳۰-۱۳۱-۱۳۲-۱۳۳-۱۳۴-۱۳۵-۱۳۶-۱۳۷-۱۳۸-۱۳۹-۱۴۰-۱۴۱-۱۴۲-۱۴۳-۱۴۴-۱۴۵-۱۴۶-۱۴۷-۱۴۸-۱۴۹-۱۵۰-۱۵۱-۱۵۲-۱۵۳-۱۵۴-۱۵۵-۱۵۶-۱۵۷-۱۵۸-۱۵۹-۱۶۰-۱۶۱-۱۶۲-۱۶۳-۱۶۴-۱۶۵-۱۶۶-۱۶۷-۱۶۸-۱۶۹-۱۷۰-۱۷۱-۱۷۲-۱۷۳-۱۷۴-۱۷۵-۱۷۶-۱۷۷-۱۷۸-۱۷۹-۱۸۰-۱۸۱-۱۸۲-۱۸۳-۱۸۴-۱۸۵-۱۸۶-۱۸۷-۱۸۸-۱۸۹-۱۹۰-۱۹۱-۱۹۲-۱۹۳-۱۹۴-۱۹۵-۱۹۶-۱۹۷-۱۹۸-۱۹۹-۲۰۰-۲۰۱-۲۰۲-۲۰۳-۲۰۴-۲۰۵-۲۰۶-۲۰۷-۲۰۸-۲۰۹-۲۱۰-۲۱۱-۲۱۲-۲۱۳-۲۱۴-۲۱۵-۲۱۶-۲۱۷-۲۱۸-۲۱۹-۲۲۰-۲۲۱-۲۲۲-۲۲۳-۲۲۴-۲۲۵-۲۲۶-۲۲۷-۲۲۸-۲۲۹-۲۳۰-۲۳۱-۲۳۲-۲۳۳-۲۳۴-۲۳۵-۲۳۶-۲۳۷-۲۳۸-۲۳۹-۲۴۰-۲۴۱-۲۴۲-۲۴۳-۲۴۴-۲۴۵-۲۴۶-۲۴۷-۲۴۸-۲۴۹-۲۵۰-۲۵۱-۲۵۲-۲۵۳-۲۵۴-۲۵۵-۲۵۶-۲۵۷-۲۵۸-۲۵۹-۲۶۰-۲۶۱-۲۶۲-۲۶۳-۲۶۴-۲۶۵-۲۶۶-۲۶۷-۲۶۸-۲۶۹-۲۷۰-۲۷۱-۲۷۲-۲۷۳-۲۷۴-۲۷۵-۲۷۶-۲۷۷-۲۷۸-۲۷۹-۲۸۰-۲۸۱-۲۸۲-۲۸۳-۲۸۴-۲۸۵-۲۸۶-۲۸۷-۲۸۸-۲۸۹-۲۹۰-۲۹۱-۲۹۲-۲۹۳-۲۹۴-۲۹۵-۲۹۶-۲۹۷-۲۹۸-۲۹۹-۳۰۰-۳۰۱-۳۰۲-۳۰۳-۳۰۴-۳۰۵-۳۰۶-۳۰۷-۳۰۸-۳۰۹-۳۱۰-۳۱۱-۳۱۲-۳۱۳-۳۱۴-۳۱۵-۳۱۶-۳۱۷-۳۱۸-۳۱۹-۳۲۰-۳۲۱-۳۲۲-۳۲۳-۳۲۴-۳۲۵-۳۲۶-۳۲۷-۳۲۸-۳۲۹-۳۳۰-۳۳۱-۳۳۲-۳۳۳-۳۳۴-۳۳۵-۳۳۶-۳۳۷-۳۳۸-۳۳۹-۳۴۰-۳۴۱-۳۴۲-۳۴۳-۳۴۴-۳۴۵-۳۴۶-۳۴۷-۳۴۸-۳۴۹-۳۵۰-۳۵۱-۳۵۲-۳۵۳-۳۵۴-۳۵۵-۳۵۶-۳۵۷-۳۵۸-۳۵۹-۳۶۰-۳۶۱-۳۶۲-۳۶۳-۳۶۴-۳۶۵-۳۶۶-۳۶۷-۳۶۸-۳۶۹-۳۷۰-۳۷۱-۳۷۲-۳۷۳-۳۷۴-۳۷۵-۳۷۶-۳۷۷-۳۷۸-۳۷۹-۳۸۰-۳۸۱-۳۸۲-۳۸۳-۳۸۴-۳۸۵-۳۸۶-۳۸۷-۳۸۸-۳۸۹-۳۹۰-۳۹۱-۳۹۲-۳۹۳-۳۹۴-۳۹۵-۳۹۶-۳۹۷-۳۹۸-۳۹۹-۴۰۰-۴۰۱-۴۰۲-۴۰۳-۴۰۴-۴۰۵-۴۰۶-۴۰۷-۴۰۸-۴۰۹-۴۱۰-۴۱۱-۴۱۲-۴۱۳-۴۱۴-۴۱۵-۴۱۶-۴۱۷-۴۱۸-۴۱۹-۴۲۰-۴۲۱-۴۲۲-۴۲۳-۴۲۴-۴۲۵-۴۲۶-۴۲۷-۴۲۸-۴۲۹-۴۳۰-۴۳۱-۴۳۲-۴۳۳-۴۳۴-۴۳۵-۴۳۶-۴۳۷-۴۳۸-۴۳۹-۴۴۰-۴۴۱-۴۴۲-۴۴۳-۴۴۴-۴۴۵-۴۴۶-۴۴۷-۴۴۸-۴۴۹-۴۵۰-۴۵۱-۴۵۲-۴۵۳-۴۵۴-۴۵۵-۴۵۶-۴۵۷-۴۵۸-۴۵۹-۴۶۰-۴۶۱-۴۶۲-۴۶۳-۴۶۴-۴۶۵-۴۶۶-۴۶۷-۴۶۸-۴۶۹-۴۷۰-۴۷۱-۴۷۲-۴۷۳-۴۷۴-۴۷۵-۴۷۶-۴۷۷-۴۷۸-۴۷۹-۴۸۰-۴۸۱-۴۸۲-۴۸۳-۴۸۴-۴۸۵-۴۸۶-۴۸۷-۴۸۸-۴۸۹-۴۹۰-۴۹۱-۴۹۲-۴۹۳-۴۹۴-۴۹۵-۴۹۶-۴۹۷-۴۹۸-۴۹۹-۵۰۰-۵۰۱-۵۰۲-۵۰۳-۵۰۴-۵۰۵-۵۰۶-۵۰۷-۵۰۸-۵۰۹-۵۱۰-۵۱۱-۵۱۲-۵۱۳-۵۱۴-۵۱۵-۵۱۶-۵۱۷-۵۱۸-۵۱۹-۵۲۰-۵۲۱-۵۲۲-۵۲۳-۵۲۴-۵۲۵-۵۲۶-۵۲۷-۵۲۸-۵۲۹-۵۳۰-۵۳۱-۵۳۲-۵۳۳-۵۳۴-۵۳۵-۵۳۶-۵۳۷-۵۳۸-۵۳۹-۵۴۰-۵۴۱-۵۴۲-۵۴۳-۵۴۴-۵۴۵-۵۴۶-۵۴۷-۵۴۸-۵۴۹-۵۵۰-۵۵۱-۵۵۲-۵۵۳-۵۵۴-۵۵۵-۵۵۶-۵۵۷-۵۵۸-۵۵۹-۵۶۰-۵۶۱-۵۶۲-۵۶۳-۵۶۴-۵۶۵-۵۶۶-۵۶۷-۵۶۸-۵۶۹-۵۷۰-۵۷۱-۵۷۲-۵۷۳-۵۷۴-۵۷۵-۵۷۶-۵۷۷-۵۷۸-۵۷۹-۵۸۰-۵۸۱-۵۸۲-۵۸۳-۵۸۴-۵۸۵-۵۸۶-۵۸۷-۵۸۸-۵۸۹-۵۹۰-۵۹۱-۵۹۲-۵۹۳-۵۹۴-۵۹۵-۵۹۶-۵۹۷-۵۹۸-۵۹۹-۶۰۰-۶۰۱-۶۰۲-۶۰۳-۶۰۴-۶۰۵-۶۰۶-۶۰۷-۶۰۸-۶۰۹-۶۱۰-۶۱۱-۶۱۲-۶۱۳-۶۱۴-۶۱۵-۶۱۶-۶۱۷-۶۱۸-۶۱۹-۶۲۰-۶۲۱-۶۲۲-۶۲۳-۶۲۴-۶۲۵-۶۲۶-۶۲۷-۶۲۸-۶۲۹-۶۳۰-۶۳۱-۶۳۲-۶۳۳-۶۳۴-۶۳۵-۶۳۶-۶۳۷-۶۳۸-۶۳۹-۶۴۰-۶۴۱-۶۴۲-۶۴۳-۶۴۴-۶۴۵-۶۴۶-۶۴۷-۶۴۸-۶۴۹-۶۵۰-۶۵۱-۶۵۲-۶۵۳-۶۵۴-۶۵۵-۶۵۶-۶۵۷-۶۵۸-۶۵۹-۶۶۰-۶۶۱-۶۶۲-۶۶۳-۶۶۴-۶۶۵-۶۶۶-۶۶۷-۶۶۸-۶۶۹-۶۷۰-۶۷۱-۶۷۲-۶۷۳-۶۷۴-۶۷۵-۶۷۶-۶۷۷-۶۷۸-۶۷۹-۶۸۰-۶۸۱-۶۸۲-۶۸۳-۶۸۴-۶۸۵-۶۸۶-۶۸۷-۶۸۸-۶۸۹-۶۹۰-۶۹۱-۶۹۲-۶۹۳-۶۹۴-۶۹۵-۶۹۶-۶۹۷-۶۹۸-۶۹۹-۷۰۰-۷۰۱-۷۰۲-۷۰۳-۷۰۴-۷۰۵-۷۰۶-۷۰۷-۷۰۸-۷۰۹-۷۱۰-۷۱۱-۷۱۲-۷۱۳-۷۱۴-۷۱۵-۷۱۶-۷۱۷-۷۱۸-۷۱۹-۷۲۰-۷۲۱-۷۲۲-۷۲۳-۷۲۴-۷۲۵-۷۲۶-۷۲۷-۷۲۸-۷۲۹-۷۳۰-۷۳۱-۷۳۲-۷۳۳-۷۳۴-۷۳۵-۷۳۶-۷۳۷-۷۳۸-۷۳۹-۷۴۰-۷۴۱-۷۴۲-۷۴۳-۷۴۴-۷۴۵-۷۴۶-۷۴۷-۷۴۸-۷۴۹-۷۵۰-۷۵۱-۷۵۲-۷۵۳-۷۵۴-۷۵۵-۷۵۶-۷۵۷-۷۵۸-۷۵۹-۷۶۰-۷۶۱-۷۶۲-۷۶۳-۷۶۴-۷۶۵-۷۶۶-۷۶۷-۷۶۸-۷۶۹-۷۷۰-۷۷۱-۷۷۲-۷۷۳-۷۷۴-۷۷۵-۷۷۶-۷۷۷-۷۷۸-۷۷۹-۷۸۰-۷۸۱-۷۸۲-۷۸۳-۷۸۴-۷۸۵-۷۸۶-۷۸۷-۷۸۸-۷۸۹-۷۹۰-۷۹۱-۷۹۲-۷۹۳-۷۹۴-۷۹۵-۷۹۶-۷۹۷-۷۹۸-۷۹۹-۸۰۰-۸۰۱-۸۰۲-۸۰۳-۸۰۴-۸۰۵-۸۰۶-۸۰۷-۸۰۸-۸۰۹-۸۱۰-۸۱۱-۸۱۲-۸۱۳-۸۱۴-۸۱۵-۸۱۶-۸۱۷-۸۱۸-۸۱۹-۸۲۰-۸۲۱-۸۲۲-۸۲۳-۸۲۴-۸۲۵-۸۲۶-۸۲۷-۸۲۸-۸۲۹-۸۳۰-۸۳۱-۸۳۲-۸۳۳-۸۳۴-۸۳۵-۸۳۶-۸۳۷-۸۳۸-۸۳۹-۸۴۰-۸۴۱-۸۴۲-۸۴۳-۸۴۴-۸۴۵-۸۴۶-۸۴۷-۸۴۸-۸۴۹-۸۵۰-۸۵۱-۸۵۲-۸۵۳-۸۵۴-۸۵۵-۸۵۶-۸۵۷-۸۵۸-۸۵۹-۸۶۰-۸۶۱-۸۶۲-۸۶۳-۸۶۴-۸۶۵-۸۶۶-۸۶۷-۸۶۸-۸۶۹-۸۷۰-۸۷۱-۸۷۲-۸۷۳-۸۷۴-۸۷۵-۸۷۶-۸۷۷-۸۷۸-۸۷۹-۸۸۰-۸۸۱-۸۸۲-۸۸۳-۸۸۴-۸۸۵-۸۸۶-۸۸۷-۸۸۸-۸۸۹-۸۹۰-۸۹۱-۸۹۲-۸۹۳-۸۹۴-۸۹۵-۸۹۶-۸۹۷-۸۹۸-۸۹۹-۹۰۰-۹۰۱-۹۰۲-۹۰۳-۹۰۴-۹۰۵-۹۰۶-۹۰۷-۹۰۸-۹۰۹-۹۱۰-۹۱۱-۹۱۲-۹۱۳-۹۱۴-۹۱۵-۹۱۶-۹۱۷-۹۱۸-۹۱۹-۹۲۰-۹۲۱-۹۲۲-۹۲۳-۹۲۴-۹۲۵-۹۲۶-۹۲۷-۹۲۸-۹۲۹-۹۳۰-۹۳۱-۹۳۲-۹۳۳-۹۳۴-۹۳۵-۹۳۶-۹۳۷-۹۳۸-۹۳۹-۹۴۰-۹۴۱-۹۴۲-۹۴۳-۹۴۴-۹۴۵-۹۴۶-۹۴۷-۹۴۸-۹۴۹-۹۵۰-۹۵۱-۹۵۲-۹۵۳-۹۵۴-۹۵۵-۹۵۶-۹۵۷-۹۵۸-۹۵۹-۹۶۰-۹۶۱-۹۶۲-۹۶۳-۹۶۴-۹۶۵-۹۶۶-۹۶۷-۹۶۸-۹۶۹-۹۷۰-۹۷۱-۹۷۲-۹۷۳-۹۷۴-۹۷۵-۹۷۶-۹۷۷-۹۷۸-۹۷۹-۹۸۰-۹۸۱-۹۸۲-۹۸۳-۹۸۴-۹۸۵-۹۸۶-۹۸۷-۹۸۸-۹۸۹-۹۹۰-۹۹۱-۹۹۲-۹۹۳-۹۹۴-۹۹۵-۹۹۶-۹۹۷-۹۹۸-۹۹۹-۱۰۰۰

• از معماری AVR RISC استفاده می کند.

- کارایی بالا و توان مصرفی کم.
- دارای 118 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می شوند.
- 32*8 رجیستر کاربردی.

• حافظه، برنامه و داده غیر فرار

- 8K بایت حافظه FLASH قابل برنامه ریزی داخلی.
- پایداری حافظه FLASH: قابلیت 1000 بار نوشتن و پاک کردن (WRITE / ERASE)
- 512 بایت حافظه SRAM
- 512 بایت حافظه EEPROM داخلی قابل برنامه ریزی.
- پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)
- قفل برنامه FLASH و حفاظت داده EEPROM

• خصوصیات جانبی

- دو تایمر/کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE
- یک تایمر/کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مُدهای CAPTURE، COMPARE و دو خروجی PWM، 8، 9 یا 10 بیتی.
- 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
- یک مقایسه کننده آنالوگ داخلی.
- UART سریال قابل برنامه ریزی
- WATCHDOG قابل برنامه ریزی با اسیلاتور داخلی.
- ارتباط سریال SPI برای برنامه ریزی 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) با سیلاتور مجزا.
- تغذیه کم در مدهای IDLE و POWERDOWN.
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- دارای سه مد IDLE : SLEEP ، POWER-DOWN و POWER-SAVE.
- عملکرد کاملاً ثابت.
- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• توان مصرفی در 20°C ، 3V ، 4MHz

— حالت فعال (ACTIVE MODE) 6.4mA

— در حالت بی‌کاری (IDLE MODE) 1.9mA

— در حالت POWER - DOWN : $1\mu\text{A} >$

• ولتاژهای عملیاتی (کاری)

— 2.7V تا 6V برای (AT90LS8535)

— 4V تا 6V برای (AT90S8535)

• فرکانسهای کاری

— 0MHz تا 4MHz برای (AT90LS8535)

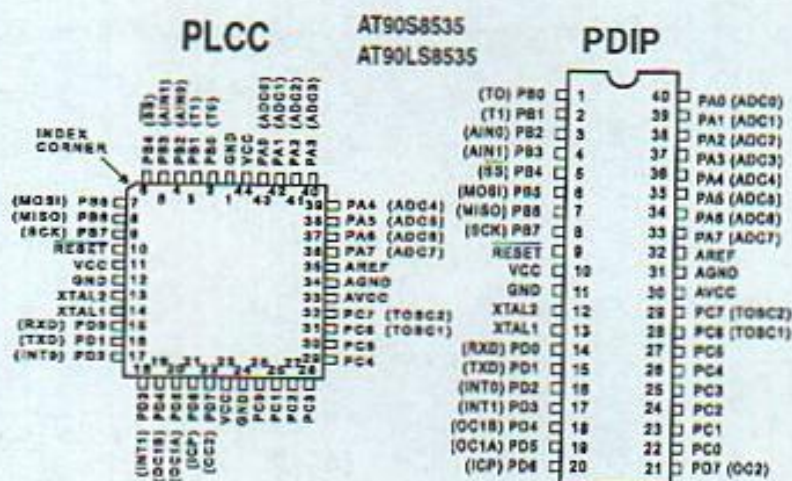
— 0MHz تا 8MHz برای (AT90S8535)

• 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



فیوز بیت های AT90S8535

این میکرو دارای 2 فیوز بیت (FSTRT , SPIEN) است. فیوز بیت ها با پاک کردن (ERASE) میکرو تأثیری نمی بینند.

در تمام توضیحات زیر 0 به معنای برنامه ریزی شدن و 1 به معنای برنامه ریزی نشدن بیت است.
 SPIEN : در حالت پیش فرض برنامه ریزی شده و میکرو از طریق سریال SPI برنامه ریزی می شود.
 FSTRT : با برنامه ریزی کردن این بیت کوتاه ترین زمان شروع (START UP) از ریست و مدهای SLEEP در نظر گرفته می شود. این بیت به صورت پیش فرض برنامه ریزی نشده است و طولانی ترین زمان در نظر گرفته شده است. جدول زیر مشخص کننده این زمان است.

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

جدول انتخاب زمان Reset Characteristics



میکروکنترلرهای MEGA AVR

در این فصل به معرفی میکروکنترلرهای نوع MEGA AVR از سری میکروکنترلرهای AVR شرکت ATMEGA می‌پردازیم. میکروهای MEGA نسبت به نوع قبلی (AT90S, TINY) دارای قابلیت بیشتری هستند. خصوصیات و قابلیت‌های هر یک از میکروهای نوع MEGA AVR 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 (1) و (2) به معرفی انواع کلاک سیستم میکروهای ارجاع شده به این دو بخش پرداخته است.

اهداف

۱. آشنایی کامل با انواع میکروهای MEGA AVR
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

۱-۳-۵۴ 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 داده غیرفرار

- 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 (WRITE / ERASE).
- قفل برنامه FLASH و حفاظت داده EEPROM .

• قابلیت ارتباط JTAG (IEEE Std.)

— برنامه‌ریزی برنامه 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 (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مُدهای CAPTURE و COMPARE .
- 4 کانال PWM
- 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
- یک مقایسه‌کننده آنالوگ داخلی.
- WATCHDOG قابل برنامه‌ریزی با اسپلاتور داخلی.
- ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING) .
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) 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 , IDLE , POWER - SAVE , STANDBY , EXTENDED STANDBY و ADC NOISE REDUCTION)
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- دارای اسلاتور RC داخلی کالیبره شده
- عملکرد کاملاً ثابت.
- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• ولتاژهای عملیاتی (کاری)

- 2.7V تا 5.5V 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 بسته بندی

- 32 خط ورودی / خروجی (I/O) قابل برنامه ریزی.
- 40 پایه PDIP و 44 پایه TQFP.

• ترکیب پایه ها



فیوز بیت های ATMEGA323

ATMEGA323 دارای دو بایت فیوز بیت طبق جدول های زیر می باشد :

FUSE HIGHT BYTE

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 , JTAG 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)

ادامه جدول صفحه قبل

FUSE LOW BYTE

FUSE HIGH 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)

OC DEN : 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 قسمتهای از میکرو امکان می‌دهد که در مدهای **SLEEP** کار کنند که این خود باعث افزایش مصرف سیستم میگردد. 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 **PC5..2** در این ارتباط استفاده می‌شود.

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** پاک می‌شود ولی در صورتی که برنامه‌ریزی شود محتویات **EEPROM** در زمان پاک شدن میکرو محفوظ می‌ماند.

BOOTSZ0 , BOOTSZ1 : برای انتخاب مقدار حافظه **BOOT** طبق جدول زیر برنامه‌ریزی می‌شوند و در صورت برنامه‌ریزی شدن فیوز بیت **BOOTRST** اجرای برنامه از آدرس حافظه **BOOT** آغاز خواهد شد.

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

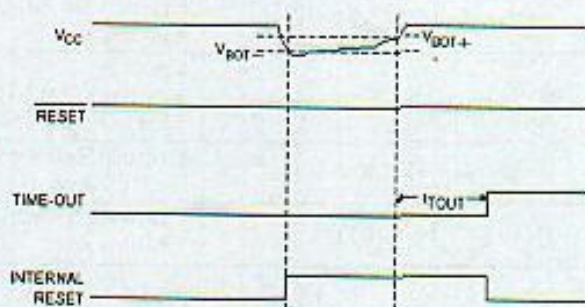
جدول انتخاب مقدار حافظه BOOT توسط فیوز بیت های 1,0 BOOTSZ0

BOOTRST: بیتی برای انتخاب بردار ریست BOOT که در حالت پیش فرض برنامه ریزی نشده و آدرس بردار ریست \$0000 است و در صورت برنامه ریزی آدرس بردار ریست طبق جدول زیر به آدرسی که فیوز بیت های 0,1 BOOTSZ0 و 1,0 BOOTSZ1 مشخص کرده اند تغییر می یابد.

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

جدول انتخاب آدرس بردار ریست توسط فیوز بیت BOOTRST

BODLEVEL: زمانی که این بیت برنامه ریزی نشده (پیش فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین تر شود ریست داخلی میکرو فعال شده و سیستم را ریست می کند. زمانی که این بیت برنامه ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین تر شود ریست داخلی میکرو فعال شده و سیستم را طبق شکل ۱-۳ ریست می کند.



شکل ۱-۳

زمانبندی ریست BROWN-OUT

BODEN: برای فعال کردن عملکرد مدار BROWN - OUT این بیت بایستی برنامه ریزی شده باشد. این بیت به صورت پیش فرض برنامه ریزی نشده است.

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

جدول سطوح مختلف ولتاژ برای مدار BROWN-OUT

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

CKSEL 3.0	Start-up Time, VCC=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 μ s + 6 CK	10 μ s + 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 μ s + 6 CK(4)	10 μ s + 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 μ s + 6 CK(4)	10 μ s + 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 μ s + 16K CK(4)	10 μ s + 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 μ s + 1K CK(4)	10 μ s + 1K CK(5)	Ceramic Resonator, BOD Enabled

جدول انتخاب کلاک سیستم و زمان START-UP برای ATMEGA323

۲-۳ 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 داده غیرفرار

- 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 (WRITE / ERASE).
- قفل برنامه FLASH و حفاظت داده EEPROM

• قابلیت ارتباط JTAG (IEEE Std.)

- برنامه‌ریزی برنامه 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 (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مُدهای CAPTURE ، COMPARE
- 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) با اسیلاتور مجزا.
- WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی.
- ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING).
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE
- قابلیت ارتباط با پروتکل سریال دوسیمه (TWO - WIRE)
- USART سریال قابل برنامه‌ریزی

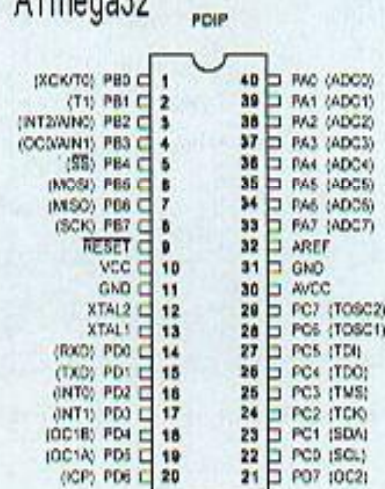
• خصوصیات ویژه میکروکنترلر

- POWER - ON RESET CIRCUIT
- BROWN-OUT DETECTION قابل برنامه‌ریزی
- دارای 6 حالت (SLEEP ، POWER - DOWN ، IDLE ، POWER - SAVE ، STANDBY)
- (ADC NOISE REDUCTION و EXTENDED STANDBY)
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- دارای اسیلاتور RC داخلی کالیبره شده

۶۰ میکروکنترلرهای AVR

- عملکرد کاملاً ثابت.
- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS
- ولتاژهای عملیاتی (کاری)
 - 2.7V تا 5.5V 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

ATmega32



فیوز بیت‌های ATMEGA32

ATMEGA32 دارای دو بایت فیوز بیت طبق جدول‌های زیر می‌باشد :

FUSE HIGT BYTE

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)

ادامه جدول صفحه قبل

FUSE LOW BYTE

FUSE HIGH 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)

فیوز بیت‌ها با پاک کردن (ERASE) میکرو تأثیری نمی‌بینند ولی می‌توانند با برنامه‌ریزی بیت LB1 قفل شوند. منطق 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت 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 مدهای 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 پیش فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید. پایه‌های PC5..2 در این ارتباط استفاده می‌شود.

SPIEN: در حالت پیش فرض برنامه‌ریزی شده و میکرو از طریق سریال SPI برنامه‌ریزی می‌شود.
CKOPT: انتخاب کلاک که به صورت پیش فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در قسمت کلاک سیستم (1) در انتهای همین فصل آمده است.
EESAVE: در حالت پیش فرض 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 , BOOTSZ1: برای انتخاب مقدار حافظه BOOT طبق جدول زیر برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت **BOOTRST** اجرای برنامه از این آدرس حافظه BOOT آغاز خواهد شد.

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 انتخاب مقدار حافظه BOOT توسط فیز بیت های 1,0 BOOTSZ0

BOOTRST: انتخاب بردار ریست BOOT که در حالت پیش فرض برنامه ریزی نشده و آدرس بردار ریست \$0000 است و در صورت برنامه ریزی آدرس بردار ریست طبق جدول زیر به آدرسی که فیز بیت های 0,1 BOOTSZ0 و BOOTSZ1 مشخص کرده اند تغییر می یابد.

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

جدول انتخاب آدرس بردار ریست توسط فیز بیت BOOTRST

BODLEVEL: زمانی که این بیت برنامه ریزی نشده (پیش فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین تر شود ریست داخلی میکرو فعال شده و سیستم را ریست می کند. زمانی که این بیت برنامه ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین تر شود ریست داخلی میکرو فعال شده و میکرو را طبق شکل ۳-۱ ریست می کند.

BODEN: برای فعال کردن عملکرد مدار BROWN-OUT این بیت بایستی برنامه ریزی شده باشد. این بیت به صورت پیش فرض برنامه ریزی نشده است.

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

جدول سطوح مختلف ولتاژ برای مدار BROWN-OUT

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

CKSEL0 ... CKSEL3: عملکرد این بیت ها در بخش ۳-۱۴ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش فرض INTERNAL RC OSCILLATOR @ 1MHZ است.

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 داده غیرفرار
 - 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 آدرس دهی 64K بایت حافظه خارجی
 - 4K بایت حافظه EEPROM داخلی قابل برنامه‌ریزی.
 - پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE).
 - قفل برنامه FLASH و حفاظت داده EEPROM
- قابلیت ارتباط JTAG (IEEE Std.)
 - برنامه‌ریزی برنامه 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
 - دو تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مُدهای CAPTURE ، COMPARE
 - 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 و 200x
 - یک مقایسه‌کننده آنالوگ داخلی.
 - WATCHDOG قابل برنامه‌ریزی با امپلاتور داخلی.
 - ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING)
 - قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE

- قابلیت ارتباط با پروتکل سریال دوسیمه (TWO - WIRE)
- دو USART سریال قابل برنامه‌ریزی
(UNIVERSAL SYNCHRONOUS AND ASYNCHRONOUS RECEIVER AND TRANSMITTER)

• خصوصیات ویژه میکروکنترلر

- POWER - ON RESET CIRCUIT
- BROWN-OUT DETECTION قابل برنامه‌ریزی
- انتخاب نرم‌افزاری فرکانس کلاک سیستم
- دارای 6 حالت (SLEEP , POWER - DOWN , IDLE , POWER - SAVE , 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 5.5V 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

فیوز بیت‌های ATMEGA128

ATMEGA128 دارای سه بایت فیوز بیت طبق جدول‌های زیر می‌باشد :

EXTENDED FUSE BYTE

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)

ادامه جدول صفحه قبل

FUSE HIGH BYTE

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)

FUSE LOW BYTE

FUSE HIGH 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)

فیوز بیت‌ها با پاک کردن (ERASE) میکرو تاثیری نمی‌بینند ولی می‌توانند با برنامه‌ریزی بیت LBI قفل شوند.

M103C: دو میکرو MEGA103 و MEGA128 بسیار باهم 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 پیش فرض WATCHDOG غیرفعال و کاربر 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 قسمتهای از میکرو امکان می‌دهد که در مدهای **SLEEP** کار کنند که این خود باعث افزایش مصرف سیستم می‌گردد. 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 پیش فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید.

SPIEN: در حالت پیش فرض برنامه‌ریزی شده و میکرو از طریق سریال **SPI** برنامه‌ریزی می‌شود.

CKOPT: انتخاب کلاک که به صورت پیش فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های **CKSEL** دارد که در بخش کلاک سیستم (1) آمده است.

EESAVE: در حالت پیش فرض 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, BOOTSZ1: بیت‌هایی برای انتخاب مقدار حافظه **BOOT** که طبق جدول زیر برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت **BOTRST** اجرای برنامه از آدرس حافظه **BOOT** آغاز خواهد شد.

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 انتخاب مقدار حافظه **BOOT** توسط فیوز بیت‌های **BOOTSZ0,1**

BOTRST: بیتی برای انتخاب بردار ریست **BOOT** که در حالت پیش فرض برنامه‌ریزی نشده و آدرس بردار ریست **\$0000** است و در صورت برنامه‌ریزی آدرس بردار ریست طبق جدول زیر به آدرسی که فیوز بیت‌های **BOOTSZ0** و **BOOTSZ1** مشخص کرده‌اند تغییر می‌یابد.

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

جدول انتخاب آدرس بردار ریست توسط فیوز بیت **BOTRST**

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

7V میکروکنترلرهای MEGA AVR

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.

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

جدول سطوح مختلف ولتاژ برای مدار BROWN-OUT

SUT1, SUT0: این دو بیت برای انتخاب زمان START -UP بکار برده می‌شوند. عملکرد آنها در بخش کلاک سیستم (1) در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL3 ... CKSEL0: عملکرد این بیت‌ها در بخش 3-14 در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض INTERNAL RC OSCILLATOR @ 1MHZ است.

۴-۳ 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 داده غیرفرار
 - 16K بایت حافظه FLASH داخلی قابل برنامه‌ریزی .
 - پایداری حافظه FLASH : قابلیت 1000 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 برنامه FLASH و حفاظت داده EEPROM
- خصوصیات جانبی
 - دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE

- یک تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مدهای CAPTURE و COMPARE
- 3 کانال PWM
- دارای RTC (REAL-TIME CLOCK) با اسیلاتور مجزا.
- 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
- یک مقایسه کننده آنالوگ داخلی.
- WATCHDOG قابل برنامه ریزی با اسیلاتور داخلی.
- ارتباط سریال SPI برای برنامه ریزی داخل مدار (IN-SYSTEM PROGRAMMING)
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE
- قابلیت ارتباط با پروتکل سریال دو سیمه (TWO-WIRE)
- UART سریال قابل برنامه ریزی

• خصوصیات ویژه میکروکنترلر

- مدار POWER-ON RESET داخلی.
- مدار BROWN-OUT DETECTION قابل برنامه ریزی
- دارای 4 حالت SLEEP (POWER-SAVE, POWER-DOWN, 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°C

- حالت فعال (ACTIVE MODE) 5.0mA
- در حالت بی کاری (IDLE MODE) 1.9mA
- در حالت POWER-DOWN: $1\mu A >$

• 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 بسته بندی

- 32 خط ورودی / خروجی (I/O) قابل برنامه ریزی.

— 40 پای PDIP و 44 پایه TQFP

• ترکیب پایه‌ها

ATmega163



فیوز بیت‌های ATMEGA163

ATMEGA163 دارای دو بایت فیوز بیت طبق جدول‌های زیر می‌باشد :

FUSE HIGH BYTE

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)

FUSE LOW BYTE

FUSE HIGH 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)

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

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

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	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 انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های **BOOTSZ0,1**

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 زیر به آدرسی که فیوز بیت‌های **BOOTSZ0** و **BOOTSZ1** مشخص کرده‌اند تغییر می‌یابد.

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

جدول انتخاب آدرس بردار ریست توسط فیوز بیت **BOOTRST**

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.

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

جدول سطوح مختلف ولتاژ برای مدار **BROWN-OUT**

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

۷۱ میکروکنترلرهای MEGA AVR

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

CKSEL3..0	Start-up Time, VCC=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 μ s + 6 CK	10 μ s + 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 μ s + 6 CK	10 μ s + 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 μ s + 6 CK	10 μ s + 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 μ s + 16K CK	10 μ s + 16K CK	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 μ s + 1K CK	10 μ s + 1K CK	Ceramic Resonator, BOD Enabled

جدول انتخاب کلاک سیستم و زمان START-UP برای ATMEGA163

۵-۳ 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 داده غیرفرار
- 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 (WRITE / ERASE)
- قفل برنامه FLASH و حفاظت داده EEPROM

• خصوصیات جانبی

- دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE.
- یک تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مُدهای CAPTURE و COMPARE
- 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 های PDIP
- 4 کانال با دقت 10 بیتی
- 2 کانال با دقت 8 بیتی
- دارای RTC (REAL-TIME CLOCK) با اسلاتور مجزا.
- یک مقایسه‌کننده آنالوگ داخلی.
- USART سریال قابل برنامه‌ریزی
- WATCHDOG قابل برنامه‌ریزی با اسلاتور داخلی.
- ارتباط سریال SPI برای برنامه‌ریزی داخل مدار (IN - SYSTEM PROGRAMMING)
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) 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 ,
- (STANDBY و IDLE ,
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- دارای اسلاتور RC داخلی کالیبره شده
- عملکرد کاملاً ثابت.

— توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• توان مصرفی در 25°C ، 3V ، 4MHz

— حالت فعال (ACTIVE MODE) 3.6 mA

— در حالت بی کاری (IDLE MODE) 1.0mA

— در حالت POWER – DOWN : $5\mu\text{A} >$

• ولتاژهای عملیاتی (کاری)

— 2.7V تا 5.5V 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 بسته بندی

— 23 خط ورودی / خروجی (I/O) قابل برنامه ریزی

— 28 پایه PDIP و 32 پایه TQFP و MLF

• ترکیب پایه ها

ATmega8

PDIP



فیوز بیت های ATMEGA8

ATMEGA8 دارای دو بایت فیوز بیت است که در دو جدول زیر نشان داده شده اند. منطق 0 به معنای برنامه ریزی شدن و 1 به معنای برنامه ریزی نشدن بیت است.

FUSE HIGH BYTE

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)

ادامه جدول صفحه قبل

FUSE LOW BYTE

FUSE HIGH 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 پیش فرض PC6 پایه ریست است. با برنامه ریزی این بیت، پایه PC6 به عنوان پایه I/O استفاده می شود.

WDTON : در حالت پیش فرض WATCHDOG غیرفعال و کاربر بایستی نرم افزاری WATCHDOG را راه اندازی کند ولی زمانی که این بیت برنامه ریزی شود WATCHDOG همیشه روشن است.

SPIEN : در حالت پیش فرض برنامه ریزی شده و میکرو از طریق سریال SPI برنامه ریزی می شود. این بیت در مُد برنامه ریزی سریال قابل دسترس نمی باشد.

CKOPT : بیت انتخاب کلاک که به صورت پیش فرض برنامه ریزی نشده است. عملکرد این بیت بستگی به بیت های CKSEL دارد که در بخش ۳-۱۴ در انتهای همین فصل آمده است.

EESAVE : در حالت پیش فرض برنامه ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM پاک می شود ولی در صورتی که برنامه ریزی شود محتویات EEPROM در زمان پاک شدن میکرو، محفوظ می ماند.

BOOTSZ0 , BOOTSZ1 : برای انتخاب مقدار حافظه BOOT طبق جدول زیر برنامه ریزی می شوند و در زمان برنامه ریزی شدن فیوز بیت **BOOTRST** اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	4	0x000 - 0xF7F	0xF80 - 0xFFFF	0xF80
1	0	256 words	8	0x000 - 0xEF7F	0xF80 - 0xFFFF	0xF80
0	1	512 words	16	0x000 - 0xDFFF	0xE00 - 0xFFFF	0xE00
0	0	1024 words	32	0x000 - 0xBFFF	0xC00 - 0xFFFF	0xC00

جدول انتخاب مقدار حافظه BOOT توسط فیزیت های 1,0 BOOTSZ0

BOTRST: بیتی انتخاب بردار ریست BOOT که در حالت پیش فرض برنامه ریزی نشده و آدرس بردار ریست 0000\$ است و در صورت برنامه ریزی آدرس بردار ریست به آدرسی که فیزیت های 0 BOOTSZ0 و 1 BOOTSZ1 مشخص کرده اند تغییر می یابد.

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

جدول انتخاب آدرس بردار ریست توسط فیزیت BOTRST

BODLEVEL: زمانی که این بیت برنامه ریزی نشده (پیش فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین تر شود ریست داخلی میکرو فعال شده و سیستم را ریست می کند. زمانی که این بیت برنامه ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین تر شود ریست داخلی میکرو فعال شده و میکرو را طبق شکل 3-1 ریست می کند.

BODEN: برای فعال کردن عملکرد مدار BROWN-OUT این بیت بایستی برنامه ریزی شده باشد. این بیت به صورت پیش فرض برنامه ریزی نشده است.

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

جدول سطوح مختلف ولتاژ برای مدار BROWN-OUT

SUT0, SUT1: برای انتخاب زمان START-UP بکار برده می شوند. عملکرد این دو بیت در بخش 3-14 در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL0 CKSEL3: عملکرد این بیت ها در بخش 3-14 در انتهای همین فصل کاملاً توضیح داده شده است.

۳-۶ خصوصیات ATMEGA8515L, ATMEGA8515

• 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.
- 16MIPS در فرکانس 16MHZ

• حافظه، برنامه و داده غیرفرار

- 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 (WRITE / ERASE).
- قفل برنامه FLASH و حفاظت داده EEPROM

• خصوصیات جانبی

- یک تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و مد مقایسه‌ای.
- یک تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مدهای CAPTURE, COMPARE
- 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 مدهای IDLE و POWERDOWN.
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- دارای 3 مد SLEEP (IDLE, STANDBY, POWER - DOWN)
- 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 به ای (ATMEGA8515)

• فرکانسهای کاری

— 0MHz تا 8MHz پر ای (ATMEGA8515L)

— 0MHZ تا 16MHZ بر ای (ATMEGA8515)

• خطوط I/O و انواع بسته بندی

35 - خط ورودی / خروجی (I/O) قابل برنامه ریزی.

— 40 پایه (PIN) نوع PDIP، 44 پایه نوع PLCC، TQFP و MLF

• ترکیب یایدها



فیوز بیت‌های ATMEGA8515

ATMEGA8515 دارای دو بایت فیوز بیت طبق جدول‌های زیر است:

FUSE HIGT BYTE

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)

FUSE LOW BYTE

FUSE HIGH 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)

فیوز بیت‌ها با پاک کردن (ERASE) میکرو تأثیری نمی‌بینند ولی می‌توانند با برنامه‌ریزی بیت LBI قفل شوند.

AT90S8515 : ATMEGA8515 دارای تمام امکانات AT90S8515 است بعلاوه امکانات دیگری که به این میکرو اضافه شده 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.

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 بخش ۳-۱۴ 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 , BOOTSZ1 : برای انتخاب مقدار حافظه BOOT طبق جدول زیر برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

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

جدول انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

۷۹ میکروکنترلرهای MEGA AVR

BOOTRST: بیتی برای انتخاب بردار ریست BOOT است که در حالت پیش فرض برنامه ریزی نشده و آدرس بردار ریست \$0000 است و در صورت برنامه ریزی آدرس بردار ریست به آدرسی که فیوز بیت های **BOOTSZ0** و **BOOTSZ1** مشخص کرده اند تغییر می یابد.

BODLEVEL: زمانی که این بیت برنامه ریزی نشده (پیش فرض) باشد اگر ولتاژ پایه VCC از 2.7V پایین تر شود ریست داخلی میکرو فعال شده و سیستم را ریست می کند. زمانی که این بیت برنامه ریزی شده باشد اگر ولتاژ پایه VCC از 4V پایین تر شود ریست داخلی میکرو فعال شده و میکرو را ریست می کند.

BODEN: برای فعال کردن عملکرد مدار **BROWN-OUT** این بیت بایستی برنامه ریزی شده باشد. این بیت به صورت پیش فرض برنامه ریزی نشده است.

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

جدول سطوح مختلف ولتاژ برای مدار BROWN-OUT

SUT1 , SUT0: برای انتخاب زمان **START-UP** بکار برده می شوند. عملکرد این دو بیت در بخش ۳-۱۴ در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL3 ... CKSEL0: عملکرد این بیت ها در بخش ۳-۱۴ در انتهای همین فصل به طور کامل توضیح داده شده است.

۷-۳-۱ ATMEGA8535L , ATMEGA8535L خصوصیات

- 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 می شوند.
 - 32*8 رجیستر کاربردی.
- حافظه ، برنامه و داده غیر فرار
 - 8K بایت حافظه FLASH قابل برنامه ریزی داخلی.
 - پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن (WRITE / ERASE).
 - 512 بایت حافظه SRAM
 - 512 بایت حافظه EEPROM داخلی قابل برنامه ریزی.
 - پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE).
 - قفل برنامه FLASH و حفاظت داده EEPROM.

• خصوصیات جانبی

- دو تایمر - کانتر (TIMER/COUNTER) 8 بیتی با PRESCALER مجزا و دارای مُد COMPARE.
- یک تایمر - کانتر (TIMER/COUNTER) 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 و 200x در بسته‌بندی TQFP
- یک مقایسه‌کننده آنالوگ داخلی.
- USART سریال قابل برنامه‌ریزی
- WATCHDOG قابل برنامه‌ریزی با اسپلاتور 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 اسپلاتور RC داخلی کالیبره شده
- دارای RTC (REAL - TIME CLOCK) با اسپلاتور 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)

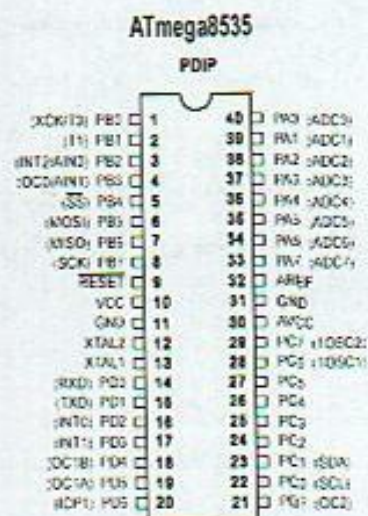
- 2.7V تا 5.5V برای (ATMEGA8535L)
- 4.5V تا 5.5V برای (ATMEGA8535)

• 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



فیوز بیت‌های ATMEG8535

این میکرو دارای و بایت فیوز بیت طبق جدول‌های زیر می‌باشد :

FUSE HIGH BYTE

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)

FUSE LOW BYTE

FUSE HIGH 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)

AT90S8535 از ATMEGA8535: S8535C دارای تمام امکانات AT90S8535 است. بعلاوه 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: انتخاب کلاک که به صورت پیش‌فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که 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.

BOOTSZ0, BOOTSZ1: برای انتخاب مقدار حافظه BOOT طبق جدول زیر برنامه‌ریزی می‌شوند.

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

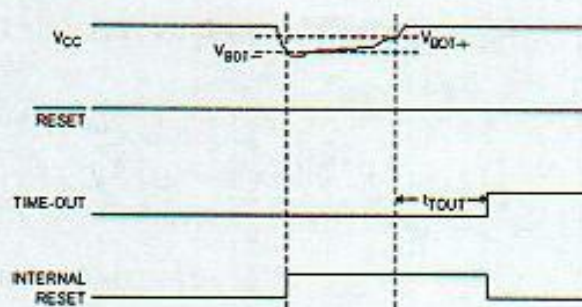
جدول انتخاب مقدار حافظه BOOT توسط فیز بیت‌های BOOTSZ0,1

BOOTRST: انتخاب بردار ریست BOOT است که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ریست 50000 است و در صورت برنامه‌ریزی آدرس بردار ریست به آدرسی که فیز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

BOOTRST	RESET ADDRESS
1(UNPROGRAMMED)	RESET VECTOR = APPLICATION RESET (ADDRESS 50000)
0(PROGRAMMED)	RESET VECTOR = BOOT LOADER RESET

جدول انتخاب آدرس بردار ریست توسط فیز بیت BOOTRST

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 طبق شکل ۲-۳ ریست می‌کند.



شکل ۲-۳

زمانبندی ریست BROWN-OUT

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

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

جدول سطوح مختلف ولتاژ برای مدار BROWN-OUT

SUT1 , SUT0: برای انتخاب زمان START -UP بکار برده می‌شوند. عملکرد این دو بیت در بخش ۱۴-۳ در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL3 CKSEL0: عملکرد این بیت‌ها در بخش ۱۴-۳ در انتهای همین فصل کاملاً توضیح داده شده است.

۸-۳ Atmega161L , Atmega161 خصوصیات

• 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 داده غیرفرار

- 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 آدرس دهی 64K بایت حافظه خارجی
- 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی.
- پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE).

— قفل برنامه FLASH و حفاظت داده EEPROM

• خصوصیات جانبی

- دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و PWM
- یک تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مدهای CAPTURE, COMPARE و دو خروجی PWM 8, 9 یا 10 بیتی
- یک مقایسه کننده آنالوگ داخلی.
- WATCHDOG قابل برنامه ریزی با اسیلاتور داخلی.
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE
- دارای RTC (REAL-TIME CLOCK) با اسیلاتور مجزا.
- دو UART سریال قابل برنامه ریزی

• خصوصیات ویژه میکروکنترلر

- POWER - ON RESET CIRCUIT.
- دارای 3 حالت SLEEP (IDLE, POWER - SAVE, POWER - DOWN)
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- عملکرد کاملاً ثابت.
- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

• توان مصرفی در 25°C , 3V, 4MHZ

- حالت فعال (ACTIVE MODE) 3.0mA
- در حالت بی کاری (IDLE MODE) 1.2 mA
- در حالت POWER - DOWN : $1\mu\text{A} >$

• ولتاژهای عملیاتی (کاری)

- 2.7V تا 5.5V 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 بسته بندی

- 35 خط ورودی / خروجی (I/O) قابل برنامه ریزی.
- 40 پایه PDIP و 44 پایه TQFP.

• ترکیب پایه‌ها

ATmega161



فیوز بیت‌های ATMEGA161

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

FUSE BITS

FUSE HIGH 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 مد برنامه‌ریزی سریال قابل دسترس نمی‌باشد.

BOOTRST: انتخاب بردار ریست 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 زیر تغییر داده می‌شود و در حالت پیش‌فرض برنامه‌ریزی نشده (1) است.

CKSEL2, CKSEL1, CKSEL0: کلاک سیستم توسط این سه بیت طبق جدول زیر تعیین می‌شود که پیش‌فرض CKSEL0..2 = 010 هستند.

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	CERAMIC RESONATOR/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

جدول انتخاب کلاک سیستم و زمان START-UP برای میکرو

۹-۳ خصوصیات Atmega162V, Atmega162

• از معماری AVR RISC استفاده می کند.

- کارایی بالا و توان مصرفی کم.
- دارای 131 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می شوند.
- 32*8 رجیستر کاربردی.
- سرعتی تا 16MIPS در فرکانس 16MHZ

• حافظه، برنامه و داده غیرفرار

- 16K بایت حافظه FLASH داخلی قابل برنامه ریزی.
- پایداری حافظه FLASH: قابلیت 10,000 بار نوشتن و پاک کردن (WRITE/ERASE)
- 1024 بایت حافظه داخلی SRAM
- 512 بایت حافظه EEPROM داخلی قابل برنامه ریزی.
- پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن (WRITE/ERASE)
- قفل برنامه FLASH و حفاظت داده EEPROM

• قابلیت ارتباط JTAG (IEEE Std.)

- برنامه ریزی برنامه FLASH، EEPROM، FUSE BITS و LOCK BITS از طریق ارتباط JTAG

• خصوصیات جانبی

- دو تایمر - کانتر (TIMER/COUNTER) 8 بیتی با PRESCALER مجزا و مد COMPARE
- دو تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مدهای CAPTURE، COMPARE
- 6 کانال PWM
- یک مقایسه کننده آنالوگ داخلی.

- WATCHDOG قابل برنامه‌ریزی با اسیلاتور داخلی.
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE.
- دارای RTC (REAL-TIME CLOCK) با اسیلاتور مجزا.
- دو UART سریال قابل برنامه‌ریزی.

• خصوصیات ویژه میکروکنترلر

- POWER - ON RESET CIRCUIT و مدار BROWN-OUT قابل برنامه‌ریزی.
- دارای اسیلاتور RC داخلی کالیبره شده.
- دارای ۵ 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 5.5V برای (Atmega162V)
- 2.7V تا 5.5V برای (Atmega162)

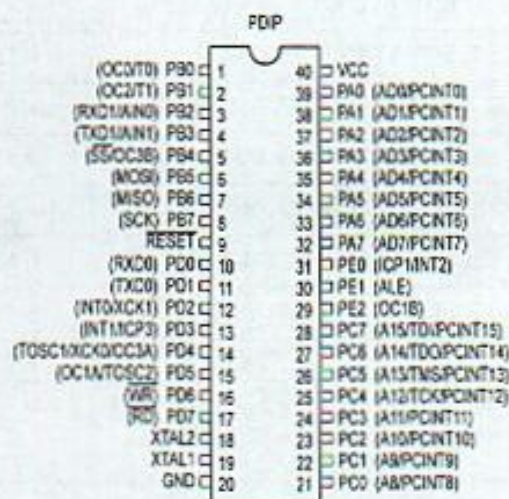
• فرکانسهای کاری

- 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 (PDIP)



فیوز بیت های ATMEGA162

ATMEGA162 دارای سه بایت فیوز بیت طبق جدول های زیر می باشد :

EXTENDED FUSE BYTE

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)

FUSE HIGH BYTE

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1 (UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0 (PROGRAMMED , JTAG 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)
BOTRST	0	SELECT RESET VECTOR	1 (UNPROGRAMMED)

FUSE LOW BYTE

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)

فیوز بیت ها با پاک کردن (ERASE) میکرو تاثیری نمی بینند ولی می توانند با برنامه ریزی بیت LB1 قفل شوند.

نکته

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.

BODLEVEL 0, 1, 2 : میکرو ATMEGA162 دارای مدار BROWN-OUT است که ولتاژ تریگ آن توسط فیوز بیت‌های BODLEVEL طبق جدول زیر قابل تنظیم می‌باشد.

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

جدول انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

OCDEN : زمانی که این بیت به همراه بیت JTAGEN برنامه‌ریزی شده باشند سیستم ON CHIP DEBUG فعال است اگر بیت‌های قفل برنامه‌ریزی نشده باشند. برنامه‌ریزی شدن این بیت به قسمتهای از میکرو امکان می‌دهد که در مدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می‌گردد. 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 پیش فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید.

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

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

EESAVE : در حالت پیش فرض 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, BOOTSZ1 : برای انتخاب مقدار حافظه BOOT طبق جدول زیر برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

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 انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTRST : انتخاب بردار ریست BOOT است که در حالت پیش فرض برنامه ریزی نشده و آدرس بردار ریست \$0000 است و در صورت برنامه ریزی آدرس بردار ریست به آدرسی که فیوز بیت های **BOOTSZ0** و **BOOTSZ1** مشخص کرده اند تغییر می یابد.

CKDIV8 : در حالت پیش فرض این فیوز برنامه ریزی شده است و کلاک سیستم بر 8 و زمانی که برنامه ریزی نشده باشد کلاک سیستم بر یک تقسیم می شود.

CKOUT : در صورت برنامه ریزی کردن این فیوز بیت کلاک سیستم در تمام مدها، در پایه خروجی **PORTB.0** ایجاد می گردد که برای راه اندازی مدارات خارجی مناسب می باشد.

SUT0 ، **SUT1** : بیت هایی 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 بیت ها در بخش 3-15 در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش فرض **INTERNAL RC OSCILLATOR @ 8MHZ** است.

۱۰-۳ خصوصیات Atmega16 , Atmega16L

• از معماری AVR RISC استفاده می کند.

- کارایی بالا و توان مصرفی کم.
- دارای 131 دستورالعمل با کارایی بالا که اکثراً تنها در یک کلاک سیکل اجرا می شوند.
- 32*8 رجیستر کاربردی.
- سرعتی تا 16 MIPS در فرکانس 16 MHZ

• حافظه ، برنامه و داده غیر فرار

- 16K بایت حافظه FLASH داخلی قابل برنامه ریزی .
- پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن (WRITE / ERASE)
- 1024 بایت حافظه داخلی SRAM
- 512 بایت حافظه EEPROM داخلی قابل برنامه ریزی.
- پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)
- قفل برنامه FLASH و حفاظت داده EEPROM

• قابلیت ارتباط JTAG (IEEE Std.)

- برنامه ریزی برنامه FLASH ، EEPROM ، FUSE BITS و LOCK BITS از طریق ارتباط JTAG

• خصوصیات جانبی

- دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و مد COMPARE
- یک تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مدهای CAPTURE و COMPARE

- 4 کانال PWM
- 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
- 8 کانال SINGLE-ENDED
- دارای 7 کانال تفاضلی در بسته بندی TQFP
- دارای دو کانال تفاضلی با کنترل گین 1x ، 10x و 200x
- یک مقایسه کننده آنالوگ داخلی.
- WATCHDOG قابل برنامه ریزی یا اسلایاتور داخلی.
- قابلیت ارتباط با پروتکل سریال دو سیمه (TWO - WIRE)
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE
- USART سریال قابل برنامه ریزی

• خصوصیات ویژه میکروکنترلر

- POWER - ON RESET CIRCUIT و BROWN-OUT قابل برنامه ریزی.
- دارای اسلایاتور 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 3V ، 1MHZ ، 25°C برای ATMEGA16L

- حالت فعال (ACTIVE MODE) 1.1mA
- در حالت بی کاری (IDLE MODE) 0.35mA
- در حالت POWER - DOWN : $1\mu A >$

• ولتاژهای عملیاتی (کاری)

- 2.7V تا 5.5V برای (Atmega16L)
- 4.5V تا 5.5V برای (Atmega16)

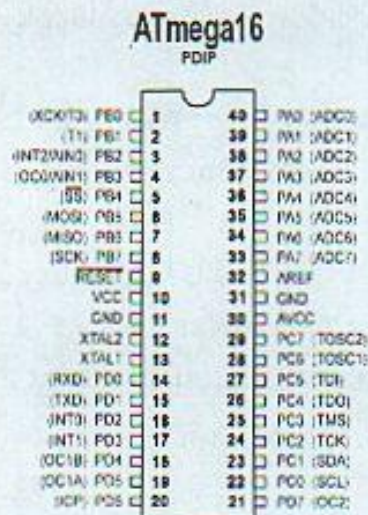
• فرکانسهای کاری

- 0MHZ تا 8MHZ برای (Atmega16L)
- 0MHZ تا 16MHZ برای (Atmega16)

• خطوط I/O و انواع بسته بندی

- 32 خط ورودی / خروجی (I/O) قابل برنامه ریزی.
- 40 پایه PDIP ، 44 پایه TQFP و 44 پایه MLF

• ترکیب پایه‌ها



فیوز بیت‌های ATMEGA16

ATMEGA16 دارای دو بایت فیوز بیت طبق جدول‌های زیر می‌باشد :

FUSE HIGH BYTE

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)
BOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

FUSE LOW BYTE

FUSE HIGH 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)

فیوز بیت‌ها با پاک کردن (ERASE) میکرو تأثیری نمی‌بینند ولی می‌توانند با برنامه‌ریزی بیت LB1 قفل شوند. منطق 0 به معنای برنامه‌ریزی شدن و 1 به معنای برنامه‌ریزی نشدن بیت 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 قسمتهای از میکرو امکان می‌دهد که در مدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می‌گردد. 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 پیش فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید.

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

CKOPT: انتخاب کلاک که به صورت پیش فرض 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 پیش فرض 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, BOOTSZ1: برای انتخاب مقدار حافظه BOOT طبق جدول زیر برنامه‌ریزی می‌شوند و در زمان برنامه‌ریزی شدن فیوز بیت BOOTRST اجرای برنامه از آدرس حافظه BOOT آغاز خواهد شد.

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 انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های BOOTSZ0,1

BOOTRST: بیتی برای انتخاب بردار ریست BOOT که در حالت پیش فرض برنامه‌ریزی نشده و آدرس بردار ریست S0000 است و در صورت برنامه‌ریزی آدرس بردار ریست به آدرسی که فیوز بیت‌های BOOTSZ0 و BOOTSZ1 مشخص کرده‌اند تغییر می‌یابد.

BOOTRST	RESET ADDRESS
1(UNPROGRAMMED)	RESET VECTOR = APPLICATION RESET (ADDRESS S0000)
0(PROGRAMMED)	RESET VECTOR = BOOT LOADER RESET

جدول انتخاب آدرس بردار ریست توسط فیوز بیت BOOTRST

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 طبق شکل ۲-۳ ری‌ست می‌کند.

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

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

جدول انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

SUT1, SUT0: عملکرد این دو بیت برای انتخاب زمان START-UP در بخش ۳-۱۴ در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL3 ... CKSEL0: عملکرد این بیت‌ها در بخش ۳-۱۴ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض INTERNAL RC OSCILLATOR @ 1MHZ است.

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 میکل اجرا می‌شوند.
- 32*8 رجیستر کاربردی.
- سرعتی تا 6 MIPS در فرکانس 6 MHZ

• حافظه، برنامه و داده غیرفرار

- 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 برنامه FLASH و حفاظت داده EEPROM

• خصوصیات جانبی

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

- یک تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مدهای COMPARE و CAPTURE و دو خروجی PWM 8، 9 یا 10 بیتی
- 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
- یک مقایسه کننده آنالوگ داخلی.
- دارای RTC (REAL-TIME CLOCK) با اسیلاتور مجزا.
- WATCHDOG قابل برنامه ریزی با اسیلاتور داخلی.
- ارتباط سریال SPI برای برنامه ریزی داخل مدار (IN - SYSTEM PROGRAMMING)
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE.
- UART سریال قابل برنامه ریزی

• خصوصیات ویژه میکروکنترلر

- انتخاب نرم افزاری فرکانس کلاک
- دارای 3 حالت SLEEP (IDLE ، POWER - SAVE ، POWER - DOWN)
- 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°C

- حالت فعال (ACTIVE MODE). 5.5mA
- در حالت بی کاری (IDLE MODE) 1.6mA
- در حالت POWER - DOWN : $1\mu A >$

• 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 بسته بندی

- 32 خط ورودی / خروجی (I/O) قابل برنامه ریزی.
- 64 پایه TQFP

فیوز بیت های ATMEGA103

ATMEGA103 دارای 4 عدد فیوز بیت است.

FUSE BITS

HIGH 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 پیش فرض برنامه ریزی شده و میکرو از طریق سریال SPI برنامه ریزی می شود.

EESAVE: در حالت پیش فرض برنامه ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM ریست می شود ولی در صورتی که برنامه ریزی شود محتویات EEPROM در زمان پاک شدن میکرو محفوظ می ماند.

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

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	

جدول انتخاب زمان RESET DELAY به ازاء VCC=5V

۱۲-۳ خصوصیات Atmega169 , Atmega169L , Atmega169V

- 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 می شوند.
 - 32*8 رجیستر کاربردی.
 - سرعتی تا 16 MIPS در فرکانس 16 MHZ
- حافظه ، برنامه و داده غیر فرار
 - 16K بایت حافظه FLASH داخلی قابل برنامه ریزی .
 - پایداری حافظه FLASH : قابلیت 10,000 بار نوشتن و پاک کردن (WRITE / ERASE)
 - 1024 بایت حافظه داخلی SRAM
 - 512 بایت حافظه EEPROM داخلی قابل برنامه ریزی.
 - پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)

— قفل برنامه FLASH و حفاظت داده EEPROM.

• قابلیت ارتباط JTAG (IEEE Std.)

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

• خصوصیات جانبی

— دارای راه انداز 4x25 SEGMENT LCD

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

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

CAPTURE و COMPARE

— 4 کانال PWM

— 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی

— یک مقایسه‌کننده آنالوگ داخلی.

— WATCHDOG قابل برنامه‌ریزی با اسپلاتور داخلی.

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

SLAVE.

— قابلیت ارتباط سریال USI (UNIVERSAL SERIAL INTERFACE)

— USART سریال قابل برنامه‌ریزی

— دارای RTC (REAL-TIME CLOCK) با اسپلاتور مجزا.

— وقفه در اثر تغییر وضعیت پایه

• خصوصیات ویژه میکروکنترلر

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

— دارای اسپلاتور RC داخلی کالیبره شده

— دارای 5 حالت SLEEP (SLEEP - 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 : 400 μ A

در 32KHZ و ولتاژ 1.8V : 20 μ A (با اسپلاتور)

در 32KHZ و ولتاژ 1.8V : 40 μ A (با اسپلاتور و LCD)

— در حالت POWER - DOWN و ولتاژ 1.8V : 0.5 μ A

• خطوط I/O و انواع بسته‌بندی

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

— پایه TQFP و MLF

• ولتاژهای عملیاتی (کاری)

— 1.8 V تا 5.5 V 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)

فیوز بیت‌های ATMEGA169

ATMEGA169 دارای سه بایت فیوز بیت طبق جدول‌های زیر می‌باشد :

EXTENDED FUSE BYTE

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

FUSE HIGH BYTE

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)

FUSE LOW BYTE

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)

BODLEVEL 2, 1, 0: میکرو ATMEGA169 دارای مدار BROWN-OUT است که ولتاژ تریگ آن

توسط فیوز بیت‌های BODLEVEL قابل تنظیم می‌باشد.

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

جدول انتخاب سطح ولتاژ برای مدار BROWN-OUT داخلی

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 قسمتهای از میکرو امکان می‌دهد که در مدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می‌گردد. 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 پیش فرض فعال است و میکرو می‌تواند از این ارتباط برای برنامه‌ریزی خود استفاده نماید.

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

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

EESAVE: در حالت پیش فرض 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 پیش فرض برنامه‌ریزی نشده است. عملکرد این بیت بستگی به بیت‌های CKSEL دارد که در بخش ۳-۱۵ آمده است.

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

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 انتخاب مقدار حافظه BOOT توسط فیوز بیت‌های **BOOTSZ0,1**

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 حالت پیش‌فرض این فیوز بیت 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 مدها، در پایه خروجی **PORTC.7** ایجاد می‌شود که برای راه‌اندازی مدارات خارجی مناسب می‌باشد.

SUT0, SUT1: عملکرد این دو بیت برای انتخاب زمان **START-UP** در بخش ۳-۱۵ کاملاً توضیح داده شده است.

CKSEL0 ... CKSEL3: عملکرد این بیت‌ها در بخش ۳-۱۵ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض **INTERNAL RC OSCILLATOR @ 8MHZ** است.

۱۳-۳ 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 داده غیرفرار

— 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 آدرس دهی 64K بایت حافظه خارجی
- 2K بایت حافظه EEPROM داخلی قابل برنامه ریزی.
- پایداری حافظه EEPROM: قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)
- قفل برنامه FLASH و حفاظت داده EEPROM.

• خصوصیات جانبی

- دو تایمر - کانتر (TIMER / COUNTER) 8 بیتی با PRESCALER مجزا و مد COMPARE
- دو تایمر - کانتر (TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای مدهای CAPTURE ، COMPARE
- 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 و 200x
- یک مقایسه کننده آنالوگ داخلی.
- WATCHDOG قابل برنامه ریزی با اسپلاتور داخلی.
- قابلیت ارتباط با پروتکل سریال دوسیمه (TWO - WIRE)
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE
- دو USART سریال قابل برنامه ریزی
- دارای RTC (REAL-TIME CLOCK) با اسپلاتور مجزا.

• خصوصیات ویژه میکروکنترلر

- POWER - ON RESET CIRCUIT و BROWN-OUT قابل برنامه ریزی.
- دارای اسپلاتور RC داخلی کالیبره شده
- دارای 6 حالت SLEEP (POWER - DOWN ، IDLE ، POWER - SAVE ، STANDBY ،
- EXTENDED STANDBY و NOISE ADC REDUCTION)
- منابع وقفه (INTERRUPT) داخلی و خارجی.
- انتخاب نرم افزاری فرکانس کلاک
- مطابقت با میکرو ATMEGA103 توسط انتخاب فیوز بیت
- غیرفعال کردن سراسری مقاومت های 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

• ولتاژهای عملیاتی (کاری)

— 2.7 V تا 5.5 V 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 زیر می‌باشد :

EXTENDED FUSE BYTE

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)

FUSE HIGH BYTE

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTAG 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)

FUSE LOW BYTE

FUSE HIGH 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 و کاربر بایستی نرم‌افزاری WATCHDOG را راه‌اندازی کند ولی زمانی که این بیت برنامه‌ریزی شود WATCHDOG همیشه روشن است.

ODEN: 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 قسمتهای از میکرو امکان می‌دهد که در مدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می‌گردد. این بیت به صورت پیش‌فرض 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 ۳-۱۴ 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 پاک شدن محفوظ می‌ماند.

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

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

جدول انتخاب مقدار حافظه BOOT توسط بیت‌های BOOTSZ0,1

BOOTRST: بیتی برای انتخاب بردار ریست BOOT که در حالت پیش‌فرض برنامه‌ریزی نشده و آدرس بردار ریست S0000 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 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 ۳-۱۴ در انتهای همین فصل کاملاً توضیح داده شده است.

CKSEL2, CKSEL1, CKSEL0: عملکرد این بیت‌ها در بخش ۳-۱۴ در انتهای همین فصل کاملاً توضیح داده شده است. مقدار پیش‌فرض INTERNAL RC OSCILLATOR @ 1MHz است.

۳-۱۴ کلاک سیستم (۱)

توزیع کلاک

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

کلاک CPU - CLKCPU

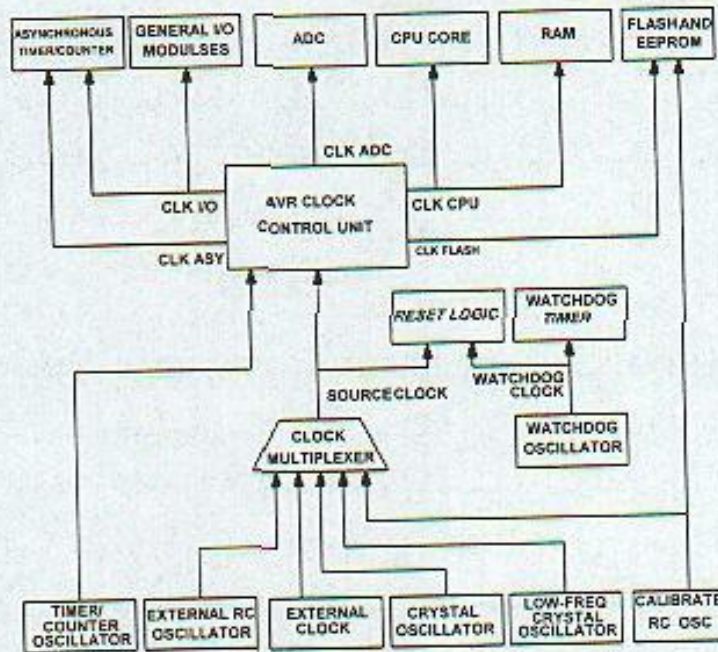
این کلاک برای انجام عملیات AVR به طور مثال رجیسترها استفاده می‌شود. توقف و به مکث بردن این کلاک باعث می‌شود که عملیات و محاسبات AVR انجام نگیرد.

کلاک I/O - CLK I/O

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

کلاک FLASH - CLKFLASH

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



شکل ۳-۳
نمودار توزیع کلاک سیستم میکرو

کلاک غیرهمزمان تایمر - CLK ASY

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

کلاک ADC - CLKADC

ADC از یک کلاک جداگانه حساس استفاده می کند که باعث می شود 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.

منابع کلاک (CLOCK SOURCE)

میکرو دارای انواع منابع کلاک اختیاری است که می توان انواع آن را به وسیله بیت های قابل برنامه ریزی FLASH (FLASH FUSE BITS) انتخاب کرد. کلاک انتخاب شده به عنوان ورودی کلاک AVR طبق جدول زیر در نظر گرفته شده و کلاک مناسب به هر قسمت سیستم داده می شود.

نکته
در تمام جداول فیوز بیت ها، 0 به معنای بیت برنامه ریزی شده (PROGRAMMED) و 1 به معنای بیت برنامه ریزی نشده (UNPROGRAMMED) است.

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

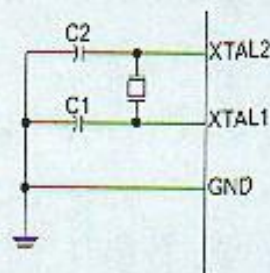
جدول انتخاب انواع کلاک سیستم میکرو

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 می‌گردد.

اسیلاتور کریستالی (EXTERNAL CRYSTAL/CERAMIC RESONATOR)

در این حالت کریستال یا نوسانگر سرامیکی (CERAMIC RESONATOR) یا کریستال کوارتز (QUARTZ CRYSTAL) همانطور که در شکل زیر 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 زیر آمده است. برای نوسانگرهای سرامیکی بایستی مقدار خازنهایی که توسط کارخانه پیشنهاد می‌گردد استفاده شود.



شکل اتصال کریستال به میکرو در حالت 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.

جدول مُدهای عملیاتی اسیلاتور کریستالی

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

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 گزینه زمانی که سیستم در فرکانسهای بالا کار نمی کند استفاده می گردد . انتخاب این گزینه ها برای کریستال ها مناسب نیست.

⁽²⁾ - این گزینه ها برای نوسانگرهای سرامیکی استفاده می شود . همچنین می توانند برای کریستال ها زمانی که در فرکانسهای پایین کار می کنند استفاده شوند .

جدول انتخاب زمان START-UP برای کلاک اسلاتور کریستالی

اسلاتور کریستالی فرکانس پایین (EXTERNAL - LOW FREQUENCY CRYSTAL)

برای استفاده از کریستال ساعت 32.768KHZ ، فیوز بیت های CKSEL با 1001 برنامه ریزی می شوند و کریستال طبق شکل صفحه قبل به پایه های XTAL1 و XTAL2 متصل می شود . با برنامه ریزی کردن CKOPT می توان خازنهای داخلی را فعال نمود و در نتیجه خازنهای خارجی را برداشت . مقدار نامی خازنهای داخلی 36PF است .

هنگامی که این نوع کریستال انتخاب می شود ، زمان شروع (START-UP) توسط فیوز بیت های SUT طبق جدول زیر قابل انتخاب است .

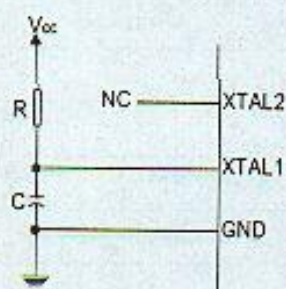
SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	1K CK	4.1 ms	FAST RISING POWER OR BOD ENABLE
01	1K CK	65 ms	SLOWLY RISING POWER
10	32K CK	65 ms	STABLE FREQUENCY AT START -UP
11	RESERVED		

جدول انتخاب زمان START-UP برای کلاک اسلاتور کریستالی فرکانس پایین

اسلاتور RC خارجی (EXTERNAL RC OSCILLATOR)

انصال RC به پایه های XTAL1 در شکل صفحه بعد آمده است . فرکانس تقریبی توسط معادله $f = 1 / (3RC)$ بدست می آید . مقدار خازن بایستی حداقل 22PF باشد . با برنامه ریزی کردن فیوز بیت

CKOPT کاربر می تواند خازنهای داخلی 36PF را بین XTAL1 و GND راه اندازی کند و در نتیجه دیگر نیازی به خازن خارجی نیست.



شکل اتصال RC به میکرو در حالت
اسیلاتور RC خارجی

اسیلاتور می تواند در 4 مُد فرکانسی کار کند که این فرکانس ها طبق فیوز بیت های CKSEL3..0 طبق جدول زیر قابل انتخاب است.

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

جدول مُدهای عملیاتی اسیلاتور RC خارجی

هنگامی که فرکانس کاری انتخاب می شود، زمان شروع (START-UP) توسط فیوز بیت های SUT طبق جدول زیر قابل انتخاب است.

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.

جدول انتخاب زمان START-UP برای کلاک اسیلاتور RC خارجی

اسیلاتور RC کالیبره شده داخلی (CALIBRATED INTERNAL RC SCILLATOR)

اسیلاتور RC کالیبره شده داخلی ، کلاک های نامی داخلی 1 ، 2 ، 4 و 8MHZ را در ولتاژ 5V و 25°C تولید می کند. این کلاک با برنامه ریزی کردن بیت های CKSEL می تواند به عنوان کلاک سیستم استفاده گردد که در این حالت نیازی به مدار خارجی نیست. زمانی که از این مُد استفاده می گردد فیوز بیت CKOPT همیشه بایستی برنامه ریزی شده باشد.

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

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

جدول مدهای عملیاتی اسپلاتور RC کالیبره شده داخلی

هنگامی که فرکانس کاری انتخاب می شود، زمان شروع (START-UP) توسط فیوز بیت های SUT طبق جدول زیر قابل انتخاب است.

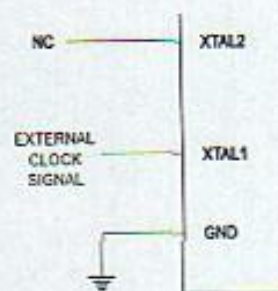
SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10 ⁽¹⁾	6 CK	65 ms	SLOWLY RISING POWER
11	RESERVED		

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

جدول انتخاب زمان START-UP برای کلاک اسپلاتور RC کالیبره شده داخلی

کلاک خارجی (EXTERNAL CLOCK)

برای راه اندازی میکرو توسط کلاک خارجی پایه XTAL1 طبق شکل زیر بایستی وصل شود. برای کار در این مدهای CKSEL با 0000 برنامه ریزی می شوند. با برنامه ریزی کردن فیوز بیت CKOPT خازن داخلی 36PF بین پایه های XTAL1 و GND فعال می شود.



شکل اتصال کلاک خارجی به پایه میکرو در حالت کلاک خارجی

هنگامی که این نوع کلاک انتخاب می شود، زمان شروع (START-UP) توسط فیوز بیت های SUT طبق جدول زیر قابل انتخاب است.

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		

جدول انتخاب زمان START-UP برای کلاک خارجی (ادامه جدول صفحه قبل)

در این مد 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.

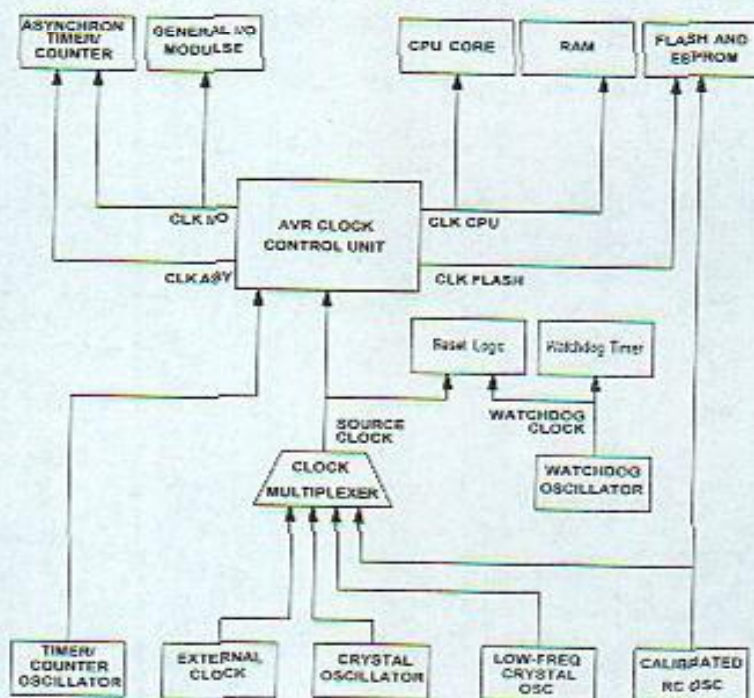
اسیلاتور تایمر/کانتر

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.

۱۵-۳ کلاک سیستم (۲)

توزیع کلاک

کلاک سیستم میکرو طبق شکل زیر توزیع شده است.



شکل ۴-۳ نمودار توزیع کلاک سیستم میکرو

کلاک CPU - CLKCPU

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 I/O - CLK I/O

این کلاک توسط بسیاری از ماژول های I/O به طور مثال تایمرها، کانترها، SPI و USART استفاده می گردد.

کلاک FLASH - CLKFLASH

این کلاک عملیات ارتباطی با حافظه FLASH را کنترل می کند. کلاک FLASH معمولاً با کلاک CPU فعال می شود.

کلاک غیرهمزمان تایمر - CLK ASY

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

منابع کلاک (CLOCK SOURCE)

میکرو دارای انواع منابع کلاک اختیاری است که می توان انواع آن را به وسیله بیت های قابل برنامه ریزی FLASH (FLASH FUSE BITS) انتخاب کرد. کلاک انتخاب شده به عنوان ورودی کلاک AVR طبق جدول زیر در نظر گرفته شده و کلاک مربوطه به هر قسمت سیستم داده می شود.

در تمام جداول فیوز بیت ها، 0 به معنای بیت برنامه ریزی شده (PROGRAMMED) و 1 به معنای بیت برنامه ریزی نشده (UNPROGRAMMED) است.

نکته

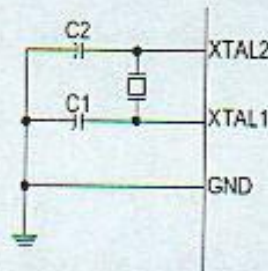
DEVICE CLOCKING OPTION	CKSEL3...0
EXTERNAL CRYSTAL/CERAMIC RESONATOR	1111 - 1000
EXTERNAL - LOW FREQUENCY CRYSTAL	0111 - 0100
CALIBRATED INTERNAL RC SCILLATOR	0010
EXTERNAL CLOCK	0000
RESERVED	0011, 0001

جدول انتخاب انواع کلاک سیستم میکرو

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

اسیلاتور کریستالی (EXTERNAL CRYSTAL/CERAMIC RESONATOR)

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



شکل اتصال کریستال به میکرو در حالت اسیلاتور کریستالی

خازنهای C 1 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 زیر آمده است. برای نوسانگرهای سرامیکی مقدار خازنهایی که توسط کارخانه پیشنهاد می‌گردد بایستی استفاده گردد.

CKSEL3..1	FREQUENCY RANGE (MHZ)	RECOMMENDED RANGE FOR CAPACITORS C1 AND C2 FOR USE WITH CRYSTAL (PF)
100 ^(۱)	0.4 – 0.9	-
101	0.9 – 3.0	12 – 22
110	3.0 – 8.0	12 – 22
111	8.0 ≤	12 – 22

^(۱) - این انتخاب برای نوسانگر سرامیکی استفاده می‌شود و نباید آن را برای کریستال به کار برد.

جدول مدهای عملیاتی اسیلاتور کریستالی

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

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 ^(۱)	4.1ms	CERAMIC RESONATOR , FAST RISING POWER
0	01	258 CK ^(۱)	65 ms	CERAMIC RESONATOR , SLOWLY RISING POWER
0	10	1K CK ^(۲)	-	CERAMIC RESONATOR , BOD ENABLE
0	11	1K CK ^(۳)	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

(۱) - این گزینه‌ها زمانی که سیستم در فرکانسهای بالا کار نمی‌کند استفاده می‌گردد. انتخاب این گزینه‌ها برای کریستال‌ها مناسب نیست.

(۲) - این گزینه‌ها برای نوسانگرهای سرامیکی استفاده می‌شود. همچنین می‌تواند برای کریستال‌ها زمانی که در فرکانسهای پایین کار می‌کنند استفاده شوند.

جدول انتخاب زمان START-UP برای کلاک اسپلاتور کریستالی

اسپلاتور 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 اسپلاتور کریستالی به پایه‌های XTAL1 و XTAL2 متصل می‌شود. با برنامه‌ریزی کردن CKSEL با مقادیر 0110 یا 0111 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 زیر قابل انتخاب است.

CKSEL3..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 انتخاب زمان START-UP برای کلاک اسپلاتور 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	

جدول انتخاب زمان RESET DELAY برای کلاک اسپلاتور کریستالی فرکانس پایین

اسپلاتور RC کالیبره شده داخلی (CALIBRATED INTERNAL RC SCILLATOR)

اسپلاتور RC کالیبره شده داخلی، کلاک‌های نامی داخلی 1، 2، 4 و 8MHZ را در ولتاژ 5V و 25°C 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. پیش فرض میکرو اسیلاتور RC داخلی 8MHZ است. هنگامی که فرکانس کاری انتخاب می شود، زمان شروع (START-UP) توسط فیوز بیت های SUT طبق جدول زیر قابل انتخاب است.

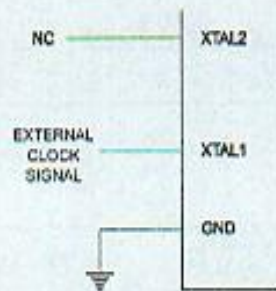
SUT1..0	START-UP TIME FROM POWER-DOWN AND POWER-SAVE	ADDITIONAL DELAY FROM RESET (VCC = 5.0V)	RECOMMENDED USAGE
00	6 CK	-	BOD ENABLE
01	6 CK	4.1ms	FAST RISING POWER
10 ⁽¹⁾	6 CK	65 ms	SLOWLY RISING POWER
11	RESERVED		

⁽¹⁾ - میکرو به صورت پیش فرض این گزینه انتخاب شده است.

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

کلاک خارجی (EXTERNAL CLOCK)

برای راه اندازی چیپ توسط کلاک خارجی پایه XTAL1 طبق شکل زیر بایستی وصل شود. برای کار در این مد بیت های CKSEL با 0000 برنامه ریزی می شوند. هنگامی که این نوع کلاک انتخاب می شود، زمان شروع (START-UP) توسط فیوز بیت های SUT طبق جدول زیر قابل انتخاب است.



شکل اتصال کلاک خارجی به پایه میکرو در حالت 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		

جدول انتخاب زمان START-UP برای 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 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 میکروکنترلرهای که دارای پایه TOSC1 و TOSC2 هستند، کریستال ساعت 32.768KHZ مستقیماً بین دو پایه قرار می گیرد. برای میکرو ATMEGA169 پایه های XTAL1 و TOSC1، XTAL2 و TOSC2 مشترک هستند و این بدان معناست که زمانی که سیستم با اسیلاتور RC داخلی کار می کند تایمر می تواند از این دو پایه کلاک دریافت کند.



محیط برنامه نویسی BASCOMAVR

انواع متنوعی از کامپایلرهای AVR عرضه شده‌اند که در این میان کامپایلرهای BASCOM ، CODEVISION و FASTAVR از اهمیت و اعتبار بیشتری برخوردار هستند. در این فصل قصد داریم به معرفی یکی از قویترین 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 4x4 ، LCD ، ایجاد تمام وقفه‌ها به صورت اختیاری، نوشتن و خواندن حافظه EEPROM و SRAM ، رویت تمام رجیسترها و متغیرهای 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 دیگر را که در عرض مدت بسیار 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

۱-۴ معرفی منوهای محیط BASCOM

پس از اجرای برنامه BASCOM پنجره محیط برنامه نویسی BASCOM ظاهر خواهد شد. اگر اولین راه اندازی شما باشد پنجره خالی خواهد بود در غیر این صورت آخرین فایلی که باز شده بود ظاهر می شود.

▪ منوی FILE

ایجاد فایل جدید (FILE NEW)



با انتخاب این گزینه یک پنجره جدید که شما قادر به نوشتن برنامه در آن هستید ایجاد می شود.

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 می کند. بنابراین شما می توانید 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 می بندد. اگر در فایل تغییری ایجاد کرده اید 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 می توانید 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 برنامه تان تغییری داده اید و آن را ذخیره نکرده اید، 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 قبلاً 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. یعنی در قسمت TEXT TO FIND متن مورد جستجو که باید توسط متن دیگری جایگزین شود را تایپ کنید و در قسمت REPLACE WITH متنی را که باید جایگزین شود تایپ کنید.

EDIT GOTO



با این گزینه شما می‌توانید مستقیماً و به سرعت به خط دلخواهی بروید.

EDIT TOGGLE BOOKMARK

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 قبلاً 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 OPTION COMPILER SETTING ایجاد خواهند شد.

فایل باینری که می‌تواند در میکروکنترلر PROGRAM شود.	XX. BIN
فایل DEBUG که برای نرم افزار شبیه ساز BASCOM مورد نیاز است.	XX. DBG
فایل OBJECT که برای نرم افزار AVR STUDIO مورد نیاز است.	XX. OBJ
فایل گزارشی	XX. RPT
فایل هگزاسیمال اینتل که برای بعضی از انواع PROGRAMMER ها مورد نیاز است.	XX. HEX
فایل خطا که فقط در هنگام بروز خطا ایجاد می‌شود.	XX. ERR
داده‌هایی که باید در EEPROM برنامه ریزی شوند در این فایل نگهداری می‌گردند.	XX. EEP

اگر خطایی در برنامه موجود باشد شما پیغام خطا را در یک کادر محاوره‌ای دریافت خواهید کرد و COMPILE متوقف می‌شود. با کلیک به روی هر کدام از آنها به خطی که خطا در آن رخ داده است پرسش 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 می‌توان استفاده کرد. گزینه OPTION COMPILER OUTPUT را برای تعیین اینکه کدام فایل‌ها باید ایجاد شوند را ببینید. فایل‌هایی که محتوای آنها قابل مشاهده اند REPORT, ERROR می‌باشند.

PROGRAM SIMULATOR

با فشردن کلید F2 یا این گزینه از منو PROGRAM شبیه ساز داخلی فعال خواهد شد. شما در برنامه با نوشتن کلمه کلیدی \$SIM قادر به شبیه‌سازی سریعتر برنامه می‌باشید. در صورت تمایل شما می‌توانید از شبیه‌سازهای دیگری مانند AVR STUDIO نیز استفاده کنید. برای شبیه‌سازی فایل‌های OBJ و DBG باید ایجاد شده باشند. فایل OBJ در برنامه شبیه‌سازی AVR STUDIO و فایل DBG برای شبیه ساز داخلی مورد استفاده قرار می‌گیرد.

فایل HEX ایجاد شده در صورت وجود دستور \$SIM در متن برنامه معتبر نمی‌باشد.

نکته

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

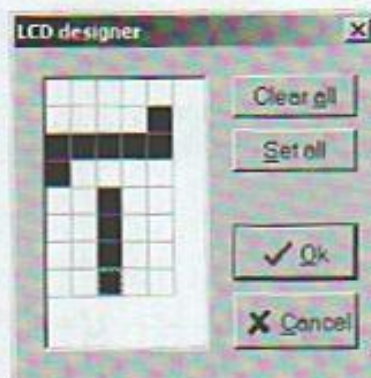


توسط این گزینه یا کلیدهای CTRL+T با بالا آوردن TERMINAL EMULATOR می‌توانید از این محیط برای نمایش داده ارسالی و دریافتی در ارتباط سریال RS-232 بین میکرو و کامپیوتر استفاده نمایید.

LCD DESIGNER



توسط این گزینه می‌توانید کاراکترهای دلخواه خود را طراحی نماید و بر روی LCD نمایش دهید.



پنجره محیط LCD DESIGNER

ماتریس LCD دارای 5x7 پیکسل میباشد که شما می‌توانید با کلیک چپ هر کدام از مربع‌ها را انتخاب و با کلیک دوباره آن را از حالت انتخاب خارج کنید. دکمه SET ALL همه نقاط را انتخاب و CLEAR 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 میگردد.

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

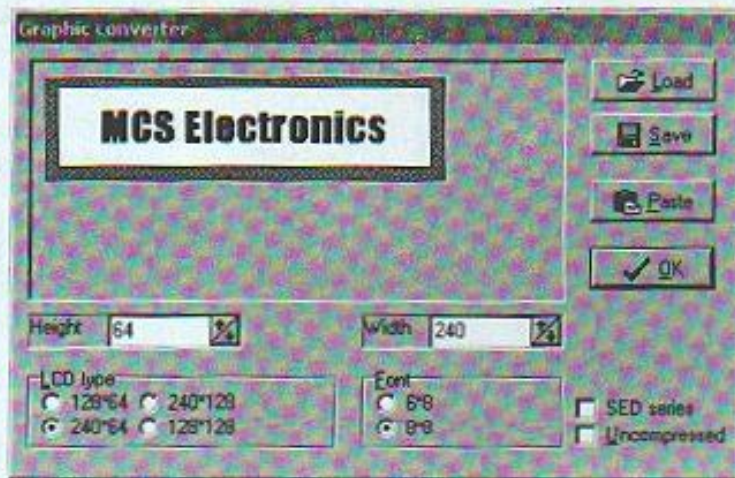


با کلیک بر روی این منو پنجره محیط GRAPHIC CONVERTOR نشان داده شده در صفحه بعد برای تبدیل تصویر با پسوند BMP* به تصویری با پسوند BGF* که قابل نمایش بر روی GRAPHIC LCD است ظاهر می‌شود.

فایل دلخواه خود را با پسوند BMP* 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*8 یا 6*8 پیکسل باشد.

نکته

رنگ‌هایی غیر از رنگ سفید در طی تبدیل به رنگ سیاه تبدیل خواهند شد.



پنجره 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 SREGFILE در برنامه استفاده کرده‌اید دیگر نیازی به انتخاب میکرو توسط این گزینه نیست.

OPTION COMPILER OUTPUT

با این گزینه می‌توان فایل‌هایی که مایل به ایجاد آنها پس از کامپایل هستیم را انتخاب کرد. با انتخاب گزینه SIZE WARNING زمانی که حجم CODE از مقدار حافظه FLASH ROM تجاوز کرد کامپایلر تولید WARNING می‌کند.

OPTION COMPILER COMMUNICATION

نرخ انتقال (BAUD RATE) ارتباط سریال توسط این گزینه تعیین می‌شود که می‌توان یک نرخ جدید نیز تایپ کرد. گزینه FREQUENCY انتخاب فرکانس کریستال استفاده شده است که می‌تواند فرکانس اختیاری نیز باشد. در صورت پیکره‌بندی هر یک از امکانات فوق در برنامه نیازی به تنظیم کردن آنها در این منو نیست.

OPTION COMPILER I2C, SPI, I2WIRE

توسط گزینه OPTION COMPILER I2C, SPI, I2WIRE می‌توان پایه‌های مربوط به ارتباطات I2C SPI و I2WIRE را تعیین کرد. در صورت پیکره‌بندی هر یک از امکانات فوق در برنامه نیازی به تنظیم کردن آنها در این منو نیست.

OPTION COMPILER LCD

گزینه OPTION COMPILER LCD نیز دارای قابلیت‌های زیر است:

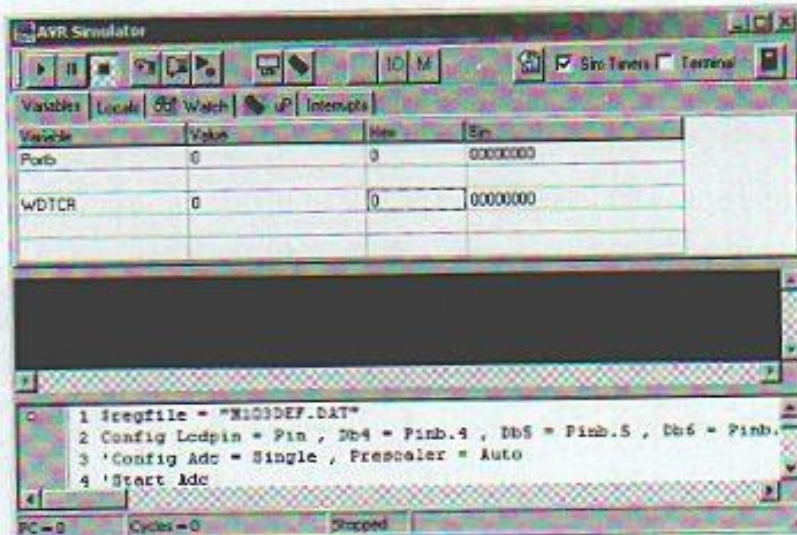
در قسمت LCD TYPE نوع LCD را مشخص می‌کنیم. گزینه BUS MODE مشخص می‌کند LCD به صورت 8 بیتی یا 4 بیتی کار کند. توسط گزینه DATA MODE تعیین می‌کنیم LCD به صورت PIN کار کند یا BUS و گزینه LCD ADDRESS مشخص‌کننده آدرس LCD در مَد BUS است. در صورت پیکره‌بندی هر یک از امکانات فوق در برنامه نیازی به تنظیم کردن آنها در این منو نیست.

OPTIONS PROGRAMMER

در این منو شما می‌توانید PROGRAMMER موردنظر خود را انتخاب نمایید.

۲-۴ معرفی محیط شبیه سازی (SIMULATOR)

با اجرای محیط شبیه سازی پنجره زیر ظاهر خواهد شد. نوار ابزار شامل دکمه هایی است که با فشار هر کدام عمل خاصی انجام می شود که قصد داریم در زیر به معرفی هر یک بپردازیم.



پنجره محیط شبیه سازی

RUN: نام این کلید RUN میباشد. با فشار دادن این دکمه شبیه سازی آغاز می شود.

PAUSE: این دکمه PAUSE میباشد، که باعث توقف موقت شبیه سازی می شود و با فشردن دکمه RUN شبیه سازی ادامه پیدا می کند.

STOP: این دکمه باعث توقف کامل شبیه سازی برنامه جاری می شود.

STEP INTO CODE: این دکمه STEP INTO CODE نام دارد و به این معناست که شما می توانید برنامه را خط به خط اجرا کنید و می توان هنگام فراخوانی توابع به داخل آنها رفته و مراحل اجرای آنها را نیز بررسی کرد. این کار را با فشردن کلید F8 نیز می توانید انجام دهید. بعد از هر بار اجرای این دستور شبیه سازی به حالت PAUSE میرود.

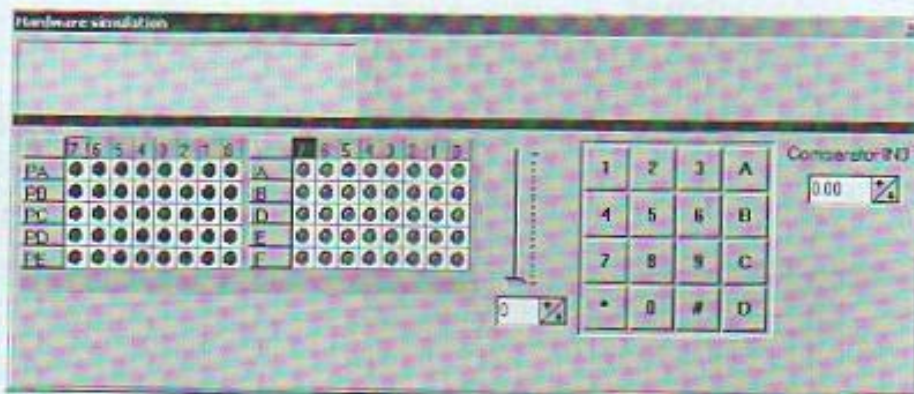
STEP OVER: این دکمه شبیه دکمه قبلی است با این تفاوت که در هنگام فراخوانی توابع به داخل SUB ROUTINE نخواهید رفت. این کار را می توانید با فشردن کلید SHIFT F8 نیز انجام دهید.

RUN TO: دکمه RUN TO شبیه سازی را تا خط انتخاب شده انجام میدهد و سپس به حالت PAUSE میرود (خط جاری باید شامل کدهای قابل اجرا باشد).

شبیه ساز سخت افزاری THE HARDWARE SIMULATOR

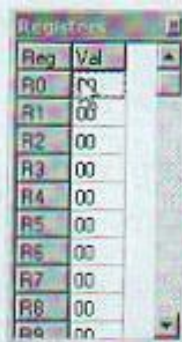
با کلیک روی این گزینه پنجره زیر نمایش داده می شود.

پنجره محیط شبیه سازی سخت افزاری



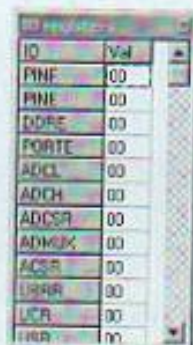
قسمت بالایی یک LCD مجازی میباشد که برای نشان دادن داده های فرستاده شده به LCD استفاده می شود. نوار LED های قرمز رنگ پایین خروجی پورتها را نشان میدهد. با کلیک روی هر یک از LED های سبز رنگ که به عنوان ورودی هستند وضعیت آن معکوس می شود و روشن شدن LED به منزله یک کردن پایه پورت است. یک صفحه کلید نیز تعبیه شده است که با دستور GETKBD() در برنامه قابل خواندن می باشد. در ضمن مقدار آنالوگ نیز هم برای مقایسه کننده آنالوگ و هم برای کانال های مختلف ADC قابل اعمال است.

REGISTERS: این دکمه پنجره رجیسترها را با مقادیر فعلی نمایش میدهد. مقادیرهای نشان داده شده در این پنجره هگزادسیمال میباشد که برای تغییر هر کدام از آنها روی خانه مربوطه کلیک کرده و مقدار جدید را وارد کنید (شکل زیر).



پنجره نمایش رجیسترهای R0 - R31

I/O REGISTERS: دکمه IO که برای نمایش رجیسترهای I/O استفاده می شود که مانند R قابل مقادیردهی است (شکل زیر).



پنجره نمایش رجیسترهای I/O

گزینه VARIABLES

شما قادر به انتخاب متغیر با دوبار کلیک کردن در ستون VARIABLES می‌باشید. با فشار دکمه ENTER در هنگام اجرای برنامه قادر به مشاهده مقدار جدید متغیر در برنامه خواهید بود. همچنین می‌توانید مقدار هر متغیر را توسط VALUE تغییر دهید. برای نمایش 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 فشار دهید.

Variables	Locals	Alt Watch	uP	Interrupts
Variable	Value	Hex	Bin	
Portb	0	0	00000000	
WDTCR	0	0	00000000	

پنجره نمایش
VARIABLES

گزینه LOCAL

پنجره LOCAL متغیرهای محلی موجود در SUB یا FUNCTION را نشان می‌دهد. شما نمی‌توانید متغیرها را اضافه نمایید.

Variables	Locals	Alt Watch	uP	Interrupts
Variable	Value	Hex	Bin	

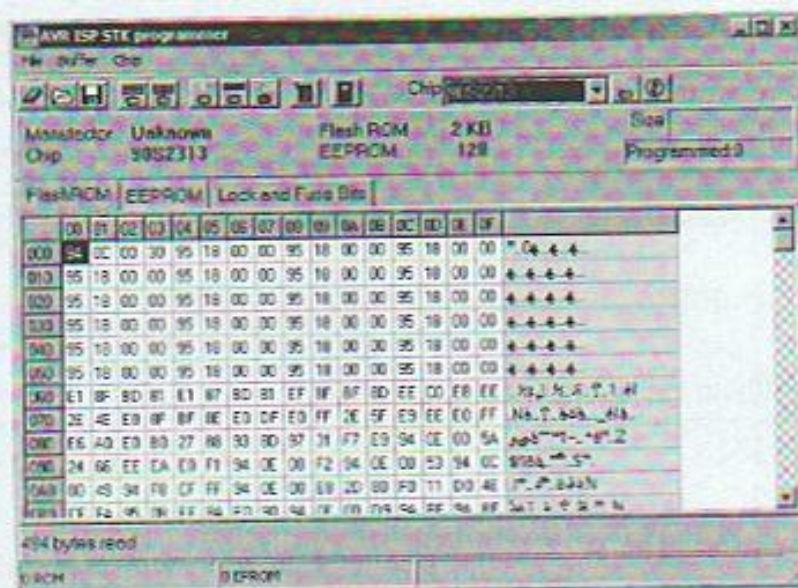
پنجره نمایش
متغیرهای محلی

گزینه WATCH

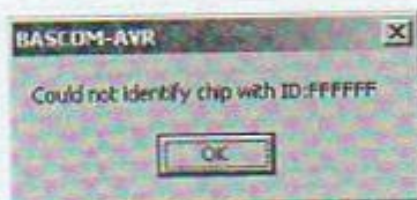
گزینه WATCH می‌تواند برای وارد کردن وضعیتی که قرار است در خلال شبیه‌سازی ارزیابی شود مورد استفاده قرار گیرد و هنگامی که وضعیت مورد نظر صحیح شد شبیه‌سازی در حالت PAUSE قرار خواهد رفت. حالت مورد نظر را در مکان متن تایپ نموده و دکمه ADD-BUTTON را فشار دهید. هنگامی که دکمه MODIFY-BUTTON فشار داده شود، وضعیت 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 فشار دهید.

Variables	Locals	Alt Watch	uP	Interrupts
		Add	Modify	
		Remove		

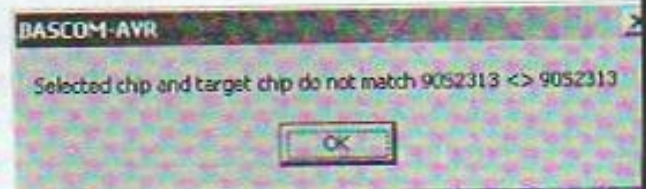
پنجره گزینه
WATCH



شکل ۱-۴
پنجره محیط برنامه ریزی



شکل ۲-۲
شکل ۳-۲





• منوی BUFFER


- BUFFER CLEAR**: این گزینه بافر را پاک می کند.
- LOAD FROM FILE**: با این گزینه می توان بافر را با فایل پی آر آر در حافظه میکرو برنامه ریزی کرد.
- SAVE TO FILE**: توسط این گزینه می توان بافر را در فایل ذخیره کرد. بافر می تواند محتوای حافظه یک میکرو باشد.

• منوی CHIP

- CHIP IDENTIFY**: با این گزینه می توان میکرو متصل شده به PROGRAMMER را شناسایی کرد.
- WRITE BUFFER TO CHIP**: توسط این گزینه می توان محتوای بافر را در حافظه ROM یا EEPROM میکرو برنامه ریزی کرد.
- READ CLIPCODE INTO BUFFER**: با این گزینه می توان داده حافظه کدی میکرو را خواند.
- BLANK CHECK**: خالی بودن حافظه میکرو را مشخص می کند.
- ERASE**: این گزینه محتوای حافظه برنامه و داده EEPROM را پاک می کند.

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. 

گزینه‌های زیر نیز به ترتیب حافظه FLASHROM ، EEPROM ، بیت‌های LOCK AND FUSE را برنامه‌ریزی می‌کنند.


FlashROM | EEPROM | Lock and Fuse Bits |

FLASHROM : با انتخاب این گزینه حافظه ROM با فایل HEX برنامه PROGRAM می‌شود.

EEPROM : حافظه EEPROM توسط این گزینه برنامه‌ریزی می‌شود.

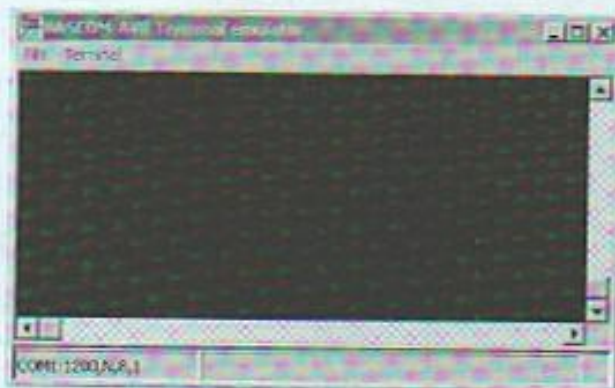
LOCK AND FUSE BITS : با این گزینه شما می‌توانید در صورت سالم بودن میکرو بیت‌های قفل و فیوز بیت‌ها را برنامه‌ریزی کنید. توسط دکمه فرمان WRITE LB می‌توان LOCK BITS را برنامه‌ریزی کرد. توسط کلیدهای WRITE FS ، WRITE FSH ، WRITE FSL و WRITE FSE به ترتیب می‌توان FUSE BITS ، FUSE BITS LOW BYTE ، FUSE BITS EXTENDED BYTE و FUSE BITS HIGH BYTE 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 نمایید. 

پنجره محیط

TERMINAL EMULATOR



اطلاعاتی که شما در این محیط تایپ می‌کنید 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 می‌کنید، می‌توانید پس از PROGRAM کردن برنامه درون میکرو و اتصال آن به پورت سریال PC ، داده‌های ارسالی توسط UART میکرو به بیرون را دریافت کرده و نمایش داد و از صحت و سقم آنها اطلاع یافت. همچنین اگر از دستوری مانند INKEY در برنامه استفاده کرده‌اید می‌توانید داده 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 میکنید میباید در گزینه COMMUNICATION SETTING نیز BAUD برابر با 9600 را انتخاب کنید. همچنین نرخ انتقال را میتوان در فایل REPORT نیز مشاهده کرد.

▪ منوهای محیط TERMINAL EMULATOR

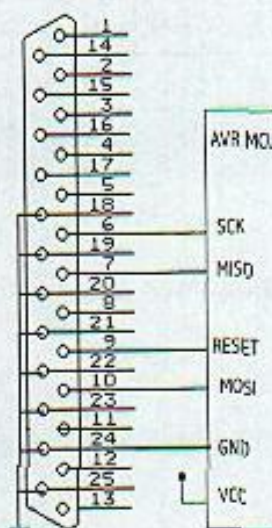
FILE UPLOAD: برنامه جاری در فرمت HEX را UPLOAD می کند.
FILE ESCAPE: صرف نظر کردن از UPLOAD کردن فایل.
FILE EXIT: خروج از برنامه EMULATOR.
TERMINAL CLEAR: پنجره ترمینال را پاک می کند.
TERMINAL OPEN LOG: فایل LOG را باز یا بسته می کند. هنگامی که فایل LOG وجود نداشته باشد از شما درخواست نامی برای فایل گزارش می کند. تمام اطلاعاتی که در پنجره TERMINAL پرینت می شود، داخل فایل LOG ثبت می شود.
SETTING: توسط این منو می توانید تنظیمات پورت COM و دیگر OPTION ها را انجام دهید.

۵-۴ ساخت STK200/300 PROGRAMMER

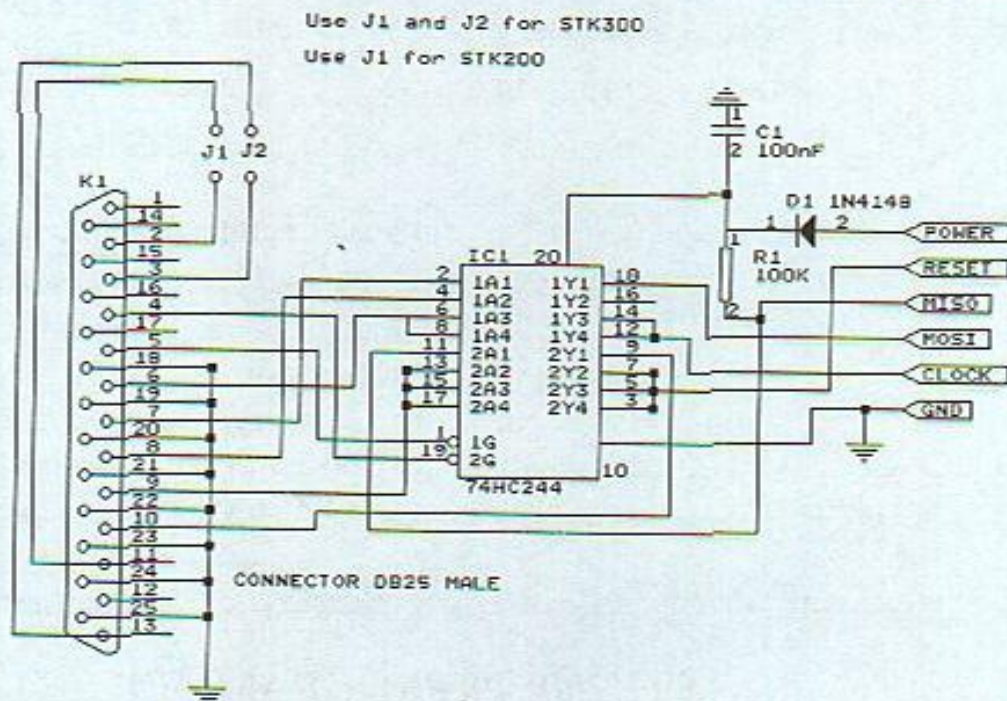
در این بخش قصد داریم به ساخت STK200/300 PROGRAMMER در چند نوع توسط بافر 74HC244 بپردازیم. در صورت استفاده از START KIT 200/300 از اتصالات J1, J2 / J1 طبق مدار شکل ۴-۴ صفحه بعد استفاده نمایید. این نوع PROGRAMMER از ارتباط SPI برای برنامه ریزی میکرو استفاده می کند در نتیجه میکروهایی که قابلیت ارتباط SPI را دارا هستند، می توان با آن برنامه ریزی کرد. همانطور که در شکل ۴-۴ نمایش داده شده است خروجی بافر به پایه های (POWER), VCC, RESET, MISO, MOSI, SCK(CLOCK) و GND از میکرو 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.

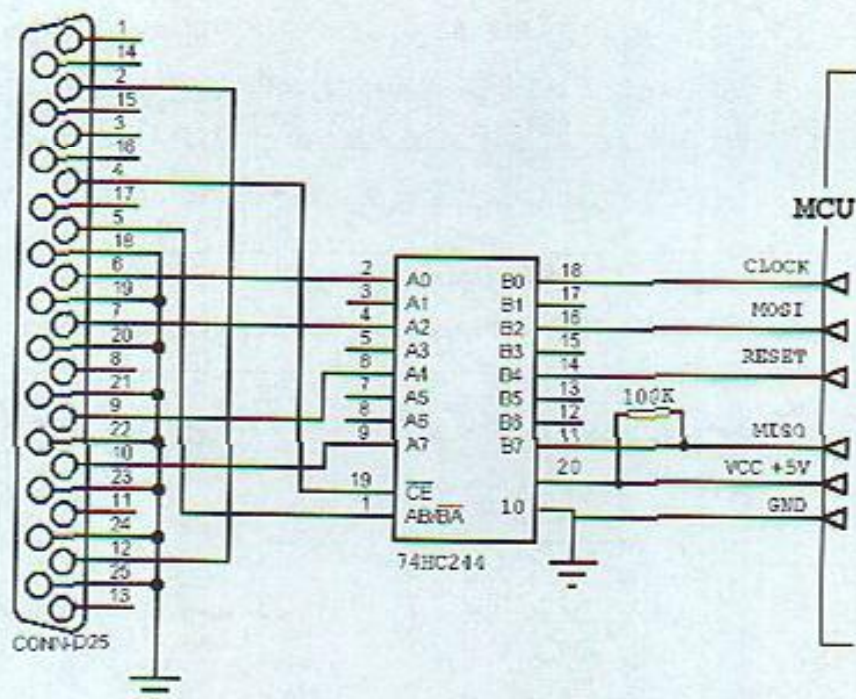
PROGRAMMER فوق را می توان به صورت مدار شکل ۵-۴ بهینه و به صورت مدار شکل ۶-۴ ساده کرد.



شکل ۶-۴
مدار STK200/300 PROGRAMMER ساده شده



شکل ۴-۴ مدار STK200/300 PROGRAMER



شکل ۵-۴ مدار STK200/300 PROGRAMER بهینه شده



دستورات و توابع محیط برنامه‌نویسی BASCOM

کامپایلر BASCOM دارای دستورات فراوانی است که در این فصل قصد داریم تمام 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 از معرفی شده‌اند. در بخش دوم با دستوراتی که با اعداد و متغیرها سروکار دارند آشنا می‌شوید. در مورد توابع ریاضی و محاسباتی در بخش سوم توضیح داده شده است. بخش چهارم شامل معرفی تبدیل کدها و متغیرها به یکدیگر است. رجیستر و خانه‌های حافظه در بخش پنجم معرفی شده‌اند. بخش ششم شامل معرفی دستورات عملیاتی حلقه و پرش است. در بخش هفتم دستوراتی که برای ایجاد تاخیر در برنامه در یک برنامه مورد نیاز از معرفی شده‌اند و بالاخره در بخش هشتم کار با زیربرنامه و تابع معرفی شده‌اند.

اهداف

۱. آشنایی کامل با تمام دستورات BASCOM
۲. تحلیل کردن برنامه‌های نوشته شده توسط SIMULATOR داخلی BASCOM

۱-۵ بدنه یک برنامه در محیط BASCOM

بدنه یک برنامه بیسیک در محیط BASCOM شامل تعیین میکرو مورد استفاده، کریستال و پایان و گزینه‌های اختیاری دیگری است که در زیر معرفی شده‌اند.

معرفی میکرو

\$REGFILE = VAR

برای شروع یک برنامه در محیط BASCOM ابتدا بایستی میکرو موردنظر تعریف گردد. VAR نام چپ مورد استفاده است که می‌تواند یکی از موارد زیر باشد.

\$regfile = "At12def.dat"	'ATTiny12 MCU
\$regfile = "At15def.dat"	'ATTiny15 MCU
\$regfile = "At22def.dat"	'ATTiny22 MCU
\$regfile = "At26def.dat"	'ATTiny26 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"	'AT90S8515 MCU
\$regfile = "8535def.dat"	'AT90S8535 MCU
\$regfile = "M8535.dat"	'MEGA 8535 MCU
\$regfile = "M8515.dat"	'MEGA8515 MCU
\$regfile = "M8def.dat"	'MEGA 8 MCU
\$regfile = "M103def.dat"	'MEGA103 MCU
\$regfile = "M16def.dat"	'MEGA16 MCU
\$regfile = "M163def.dat"	'MEGA 163 MCU
\$regfile = "M161def.dat"	'MEGA161 MCU
\$regfile = "M32def.dat"	'MEGA 32 MCU
\$regfile = "M323def.dat"	'MEGA323 MCU
\$regfile = "M603def.dat"	'MEGA603 MCU
\$regfile = "M64def.dat"	'MEGA64 MCU
\$regfile = "M128def.dat"	'MEGA 128 MCU
\$regfile = "M128103.dat"	'MEGA128 IN MEGA103 MODE MCU

کریستال

برای مشخص کردن فرکانس کریستال استفاده شده بر حسب هرتز از دستور زیر استفاده می‌نماییم.

SCRISTAL=X

X فرکانس کریستال استفاده شده بر حسب هرتز است.

این دستور را حتی برای زمانی که با اسپلاتور داخلی میکرو کار می‌کنید بنویسید.

نکته

• مثال

SCRISTAL = 14000000	'14MHZ external osc
SCRISTAL = 8000000	'8MHZ external osc
SCRISTAL = 1000000	'1MHZ internal osc

اسمبلی و بیسیک (اختیاری)

در صورت نیاز برای نوشتن برنامه اسمبلی در بین برنامه بیسیک از دستور زیر استفاده می‌نماییم.

\$ASM

ASSEMBLY PROGRAMME

\$ENDASM

با دستور \$ASM می‌توان در برنامه شروع به نوشتن برنامه مورد نظر اسمبلی کرده و پس از اتمام

برنامه اسمبلی با دستور SENDASM برنامه اسمبلی را به پایان رساند و به نوشتن ادامه برنامه پرداخت.

• مثال

```

Dim C As Byte
Loadaddr 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 یادشتهایی 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 و همچنین به رنگ سبز در می‌آیند.

شما می‌توانید از دو علامت برای شروع (' و ') برای 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 یادداشتی
  This will not be compiled
' ) end block comment        پایان متن یادداشتی

```

• 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.

\$FROMSTART = ADDRESS

ADDRESS مکانی از حافظه است که برنامه HEX از این آدرس در حافظه میکروکنترلر، شروع به نوشته شدن می‌شود. در صورتی که از این دستور استفاده نشود کامپایلر به طور خودکار آدرس 0000H 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 نرم افزاری می توان کلاک سیستم را تغییر داد. تقسیم کلاک به طور مثال برای کاهش مصرف تغذیه استفاده می شود.

```
CLOCKDIVISION = var
```

Var مقادیر معتبر بین اعداد 2 تا 128 می تواند باشد.

نکته

اگر شما از این دستور استفاده نمایید، دستورانی که مستقیماً با کلاک سیستم کار می کنند ممکن است درست کار نکنند. بطور مثال `WAITms var` دو برابر طول می کشد زمانی که از `var = 2` استفاده می نماید.

• مثال

```
Sbaud = 2400
ClockDivision = 2
Print "Hello"
End
```

پایان برنامه

```
END
```

این دستور در انتهای برنامه قرار می گیرد و اجرای برنامه را متوقف می کند. با دستور END تمام وقفه ها غیر فعال شده و یک حلقه بی نهایت تولید و برنامه خاتمه می یابد.

• مثال

```
PRINT "Hello"          'print this
END                    'end program execution and disable all interrupts
```

۲-۵ اعداد و متغیرها و جداول LOOKUP

دیمانسیون متغیر

این دستور بُعد یک متغیر را نشان می دهد. با این دستور می توانید متغیرهایی که در برنامه به کار برده می شوند تعریف کنید.

```
DIM var AS [ XRAM/SRAM/ERAM ] data type [AT location] [OVERLAY]
```

VAR نام متغیری که در برنامه به کار برده می شود. در صورت استفاده از حافظه جانبی آن را با XRAM مشخص کنید و SRAM را زمانی اختیار کنید که می خواهید متغیرها را در حافظه SRAM قرار دهید و ERAM متغیر مورد نظر را در EEROM داخلی جای می دهد. data type نوع داده است که می تواند طبق جدول زیر BIT، BYPE، INTEGER، LONG، WORD، STRING یا SINGLE باشد.

در صورت استفاده از متغیر STRING، بیشترین طول آن نیز باید نوشته شود. گزینه اختیاری OVERLAY متغیر تعریف شده را به صورت POINTER در نظر میگیرد و فضایی را برای متغیر 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	1.5×10^{-45} to 3.4×10^{38}
STRING	0 - 254 BYTES	-

جدول انتخاب نوع داده

DIM S AS STRING*10

متغیر STRING به نهایت طول 10 کاراکتر

AT LOCATION به شما اجازه می دهد که متغیرتان را در آدرسی که می خواهید در حافظه ذخیره کنید. زمانی که محل آدرس دهی اشغال باشد، اولین جای خالی در حافظه استفاده می شود.

عدد HEX را با علامت &H و عدد BINARY را با علامت &B مشخص می کنیم.

نکته

• مثال

```
A=&H01DE      'HEX NUM
B=&B01011011  'BIN NUM
```

می توان به جای استفاده از دستور DIM از دستور DEFxxx استفاده نمایید.

نکته

```
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 X As Integer At 120 'You Can Specify The Address Of The Variable
Dim Kk As Integer   'The Next Dimensioned Variable Will Be Placed After
'Variable S
B1 = 1
Set B1
A = 12
A = A + 1
C = -12
'Assign Bits
'is equal with B1 = 1
'Assign Bytes
'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"
CONST A = 5          'DECLARE A AS A CONSTANT
CONST B1 = & B1001  'OR USE AN EXPRESSION TO ASSIGN A CONSTANT
CONST X = (B1 * 3) + 2
CONST SSINGLE = SIN(1)

```

دستور ALIAS

از این دستور برای تغییر نام متغیر استفاده می‌شود.

• مثال

```
DIRECTION ALIAS PORTB.1
```

حال شما می‌توانید به جای PORTB.1 از متغیر DIRECTION استفاده نمایید.

```
SET DIRECTION          ' is equal with set portb.1
```

دستور CHR

از این دستور برای تبدیل متغیر عددی یا یک ثابت به کاراکتر استفاده می‌شود. زمانی که شما قصد دارید یک کاراکتر بر روی LCD نمایش دهید از این دستور می‌توانید استفاده نمایید. در صورتی که از این دستور بدین صورت استفاده نمایید (PRINT CHR (VAR) کاراکتر اسکی VAR به پورت سریال فرستاده خواهد شد.

• مثال

```

Dim a As Byte          'dim variable
a=65                   'assign variable
Print a                'print value (65)
PrintHEX(a)            'print hex value (41)
PrintChr(a)             'print ASCII character 65 (A)
End

```


دستور INSTR

این دستور محل و موقعیت یک زیر رشته را در رشته دیگر مشخص می کند.

```
Var = Instr( start , String , Substr )
```

```
Var = Instr( string , Substr )
```

VAR عددی است که مشخص کننده محل SUBSTR در رشته اصلی STRING می باشد و زمانی که

زیر رشته مشخص شده در رشته اصلی نباشد صفر برگردانده می شود. START نیز عددی دلخواه است

که مکان شروع جستجو زیر رشته در رشته اصلی را مشخص می کند. در صورتیکه START قید نشود

تمام رشته از ابتدا جستجو می شود. رشته اصلی تنها باید از نوع رشته باشد ولی زیر رشته

(SUBSTR) می تواند 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 متغیر عددی var می افزاید.

• مثال

```
Do
```

```
Incr A
```

```
Print A
```

```
Loop Until A > 10
```

```
'start loop
```

```
'increment a by 1 A=A+1
```

```
'print a
```

```
'repeat until a is greater than 10
```

دستور DECR

DECR var

این دستور متغیر VAR را یک واحد کم می کند.

• مثال

```
Dim A As Byte
```

```
A = 5
```

```
Decr A
```

```
Print A
```

```
End
```

```
'assign value to a
```

```
'decr by one A=A-1
```

```
'print it A=4
```

دستور CHECKSUM

این دستور مجموع کد دسیمال اسکی رشته VAR را بر می گرداند که البته اگر مجموع کد اسکی رشته

از عدد 255 بیشتر شود مقدار 256 از مجموع کم می شود.

• مثال

```
Dim S As String * 10
```

```
S="test"
```

```
PrintChecksum(s)
```

```
S = "testnext"
```

```
Print Checksum(s)
```

```
End
```

```
'Dim Variable
```

```
'Assign Variable
```

```
'Print Value(192)
```

```
'Assign Variable
```

```
'Print Value 127 (127=383-256)
```


دستور LOW

این دستور LSB (least significant byte) یک متغیر را برمی گرداند.

Var = LOW(s)

LSB متغیر S در Var قرار می گیرد.

• مثال

```
Dim I As Integer , Z As Byte
I = &H1001
Z = Low(i)           ' is 1
End
```

دستور HIGH

این دستور پرارزش ترین بایت (MSB) یک متغیر را برمی گرداند.

var = HIGH(s)

مقدار MSB متغیر S در var جای می گیرد.

• 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 می کند.

Target = Lcase(source)

تمام حروف رشته source کوچک شده و در رشته Target جای داده می شود.

• 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 می کند.

Target = Ucase(source)

تمام حروف رشته source بزرگ شده و در رشته Target جای داده می شود.

• 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 می کنیم.

۱۳۹ دستورات محیط BASCOM

`var = RIGHT(var1 , n)`

از سمت راست رشته `var1` به تعداد کاراکتر `n`، رشته‌ای جدا شده و در رشته `var` قرار می‌گیرد.

• مثال

```
Dim S As String * 15 , Z As String * 15
S = "ABCDEFGH"
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)`

رشته `var1` از سمت چپ به تعداد `n` کاراکتر جدا شده و در رشته `var` قرار می‌گیرد.

• مثال

```
Dim S As String * 15 , Z As String * 15
S = "ABCDEFGH"
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` در متغیر عددی `VAR` قرار می‌گیرد. رشته `STRING` نهایتاً ۲۵۵ 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 متغیر

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
```

دستور SWAP

SWAP var1, var2

با اجرای این دستور محتوای متغیر var1 در متغیر var2 و محتوای متغیر var2 در متغیر var1 قرار می‌گیرد. Var1 و var2 می‌توانند داده‌هایی از نوع BIT, BYTE, INTEGER, WORD, LONG یا STRING باشند.

دو متغیر Var1 و var2 بایستی از یک نوع داده باشند.

نکته

• مثال

```
Dim A As Integer, B1 As Integer
A = 1 : B1 = 2
Swap A, B1
Print A
Print B1
End
```

'assign two integers
'swap them
'prints 2
'prints 1

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 متغیر 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 = "ABCDEFGH"
Z = Mid(s, 2, 3)
Print Z
Z = Mid(s, 2, 1)
Print Z
Z = "12345"
Mid(s, 2, 2) = Z
Print S
End
```

'BCD
'B
'A12DEFG

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 BIT, BYTE, INTEGER, WORD یا LONG باشد. LEFT/RIGHT جهت

۱۴۱ دستورات محیط BASCOM

چرخش بیتها و SHIFT که اختیاری می باشد تعداد چرخش بیتها را مشخص می کند. در صورت قید نکردن shifts به صورت پیش فرض مقدار یک در نظر گرفته می شود.

• مثال

```
Dim A As Byte
a = 128
Rotate A , Left , 2
Print A
```

'A=2

SPACE دستور

برای ایجاد فضای خالی از این دستور استفاده می شود.

var = SPACE(x)

x تعداد فضای خالی است که به عنوان رشته در متغیر رشته ای var جای می گیرد.

• مثال

```
Dim S As String * 15 , Z As String * 15
S = Space(5)
Print "{* ; S ; *}"
Print "{* ; Space(6) ; *}"
```

' { } 5 space
' { } 6 space

تابع FORMAT

این دستور یک رشته عددی را شکل دهی (FORMAT) می کند.

target = Format (source , "mask")

source رشته ای است که شکل دهی شود و نتایج در target قرار می گیرد. mask نوع شکل دهی 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 , " ")
Print S
S = "12345"
S = Format(S , " + ")
Print S
S = "123"
S = Format(s , "00000")
Print S
S = "12345"
S = Format(s , "000.000")
Print S
S = "12345"
S = Format(s , "000.00")
Print S
S = "12345"
S = Format(s , " +000.00")
Print S
End
End
```

'5 space
'S= "123" 2 space first , then 123
'S= +12345
'S=00123
'S=012.345
'S=123.45
'S=+123.45

تابع FUSING

از این دستور برای روند کردن رشته های عددی استفاده می شود.

target = Fusing(source , "mask")

SOURCE رشته مورد نظر برای شکل دهی و MASK نوع شکل دهی است سپس نتایج این

شکل دهی در TARGET قرار می گیرد. عمل MASK حتماً باید با علامت # شروع شود و حداقل باید

یکی از علامات # و & را بعد از ممیز داشته باشد. علامات # و & با یکدیگر بعد از ممیز استفاده نمی‌شود. با استفاده از علامت # عدد روند می‌شود و در صورت استفاده از علامت & 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

متغیر var قرار می‌گیرد. Value = 0 اولین داده در جدول را برمی‌گرداند. تعداد اندیس‌ها و مقدار داده برگشتی به ترتیب نهایتاً می‌توانند 255 و 65535 باشند.

داده دو بایتی داخل جدول بایستی با علامت % پایان یابد.

نکته

• مثال

```
Dim B1 As Byte , I As Integer
B1 = Lookup(2 , Dta)
Print B1 ' Prints 2 (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

متغیر 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 راحت‌تر انجام دهید.

علامت	نماد
علامت ضرب	* Asterisks (multiplication symbol)
علامت جمع	+ Plus sign
علامت تفریق	- Minus sign
علامت ممیز	. Period (decimal point)
علامت تقسیم	/ Slash (division symbol)
علامت کوچکتر از	> Less than
علامت تساوی	= Equal sign
علامت بزرگتر از	< Greater than
علامت بتوان	^ Exponent
علامت کوچکتر یا مساوی با	=> Less than or equal to
علامت بزرگتر یا مساوی با	>= Greater than or equal to
علامت مخالف	<> Inequality

جدول عملگرهای ریاضی محیط BASCOM

عملگرهای منطقی

عملگرهای منطقی در BASCOM به قرار زیر است :

معرفی	نماد عملگرهای منطقی در BASCOM
Conjunction	AND
Disjunction	OR
Exclusive or	XOR
Logical complement	NOT

جدول عملگرهای منطقی محیط BASCOM

• مثال

```
A = 63 And 19      '19 PRINTS
PRINT A
A = 10 Or 9         '11 PRITNS
PRINT A
```

تابع ABS

VAR = ABS(VAR2)

این دستور به معنای ریاضی $VAR = |VAR2|$ (قدر مطلق) است.

• مثال

```
Dim A As Integer , C As Integer
```



```

a = -1000
c = Abs(a)
Print c
End

```

'c= | a |
'c=1000

تابع EXP

Target = Exp(source)

Target برابر با c به توان source است. Target متغیری از نوع داده SINGLE است.

• مثال

```

Dim X As Single
X = Exp(1.1)
Print X
X = 1.1
X = Exp(x)
Print X

```

'Prints 3.004166124
'prints 3.004164931

تابع LOG10

Target = Log10(source)

لگاریتم پایه 10 متغیر یا ثابت source در متغیر 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 متغیر یا ثابت source از نوع داده SINGLE گرفته می‌شود و در متغیر 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
X = 100
X = Log( x )
Print X
X = Log( 1.1 )
Print X
X = 1.1
X = Log( x )
Print X

```

'prints 4.605170
'Prints 4.605098
'prints 0.095310147
'prints 0.095310147
'So a smaller number is calculated faster

تابع RND

این دستور یک عدد تصادفی برمی گرداند.

var = RND(limit)

عددی تصادفی بین 0 و limit بدست آمده و در متغیر VAR قرار می گیرد. با هر بار استفاده از این دستور عدد مثبت تصادفی دیگری بدست خواهد آمد.

limit باید یک عدد مثبت باشد.

نکته

• 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 متغیر source را در متغیر var از نوع داده SINGLE قرار می دهد. تمام دستورات مثلثاتی با رادیان کار می کنند 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)
Print "VSIN="; Vsin
End
' VSIN= SIN(P/2)
'VSIN=0.99999332 WILL PRINT
'end program
```

تابع COS

var = COS (source)

این دستور کسینوس (COSINE) ثابت یا متغیر source از نوع داده SINGLE را در متغیر var از نوع داده SINGLE قرار می دهد. تمام دستورات مثلثاتی با رادیان کار می کنند 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)
Print "VCOS="; Vcos
End
' Vcos= COS(P/2)
'VCOS= -0.000006613 WILL PRINT
'end program
```

تابع TAN

var = TAN (source)

این دستور تانژانت (TANGENT) ثابت یا متغیر source از نوع داده SINGLE را در متغیر 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)
Print "VTAN=" ; Vtan
End
' Vtan= TAN(P/2)
'VTAN= -0.000000357 WILL PRINT
'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 متغیر 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 متغیر 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 متغیر عددی var از نوع داده SINGLE قرار می‌گیرد. 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 متغیر نوع SINGLE را به رادیان بر می گرداند.

```
var = ASIN ( x )
```

متغیر ورودی X عددی از نوع SINGLE است و می تواند بین -1 تا +1 باشد. نتیجه این محاسبه که بین $\pi/2$ و $-\pi/2$ در متغیر VAR از نوع داده SINGLE قرار می گیرد. اگر عدد ورودی بیشتر از +1 باشد ، $\pi/2$ و اگر عدد ورودی کمتر از -1 باشد ، $-\pi/2$ برگردانده می شود. شما با دستورات RAD2DEG و DEG2RAD می توانید مقدار بدست آمده را به درجه تبدیل نمایید.

• مثال

```
Dim S As Single , X As Single      '0.523595867 WILL PRINT
X = 0.5 : S = Asin(x)
Print S
End                                'end program
```

تابع ACOS

این دستور ARCCOSINE یک ثابت یا متغیر نوع SINGLE را به رادیان بر می گرداند.

```
var = ACOS ( x )
```

متغیر ورودی X عددی از نوع داده SINGLE است و می تواند بین -1 تا +1 باشد. نتیجه این محاسبه که بین 0 و π است در متغیر VAR از نوع داده SINGLE قرار می گیرد. اگر عدد ورودی بیشتر از +1 باشد ، صفر و اگر عدد ورودی کمتر از -1 باشد ، π برگردانده می شود. شما با دستورات RAD2DEG و DEG2RAD می توانید 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 متغیر از نوع داده SINGLE را به رادیان بر می گرداند.

```
var = ATN ( x )
```

متغیر ورودی X می تواند عددی از نوع داده SINGLE باشد. نتیجه این محاسبه که بین $\pi/2$ و $-\pi/2$ است در متغیر VAR از نوع داده SINGLE قرار می گیرد. شما با دستورات RAD2DEG و DEG2RAD می توانید مقدار بدست آمده را به درجه تبدیل نمایید.

• مثال

```
Dim S As Single
S = Atn(1) * 4                     ' prints 3.141593
Print S
```

تابع ATN2

این دستور ARCTANGENT دو متغیر X , Y از نوع داده SINGLE را در هر چهار ربع دایره مختصات

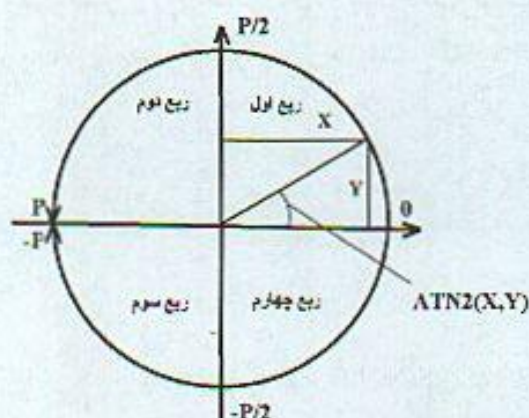
۱۴۸ میکروکنترلرهای AVR

به رادیان بر می گرداند. دستور ATN خروجی بین $\pi/2$ و $-\pi/2$ دارد در حالی که این دستور تمام مختصات دایره را از π تا $-\pi$ پوشش می دهد.

$\text{var} = \text{ATN2}(x, y)$

متغیرهای ورودی X و Y می توانند اعدادی از نوع داده SINGLE باشند که نمایانگر مختصات (X,Y) هستند. نتیجه این محاسبه در متغیر VAR از نوع داده SINGLE قرار می گیرد.

شکل خروجی دستور ATN2



QUADRANT	SING Y	SING X	ATN2
1	+	+	0 TO $\pi/2$
2	+	-	$\pi/2$ TO π
3	-	-	$-\pi/2$ TO $-\pi$
4	-	+	0 TO $-\pi/2$

جدول خروجی ATN2 به ازای علامات مختلف X و Y

شما با دستورات RAD2DEG و DEG2RAD می توانید 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 می شود.

$\text{VAR} = \text{DEG2RAD}(\text{SINGLE})$

زاویه SINGLE به رادیان تبدیل می شود و در متغیر VAR از نوع داده SINGLE قرار می گیرد.

• 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
End
```

'179.999984736 WILL PRINT
'end program

تابع ROUND

var = ROUND (x)

متغیر یا داده X از نوع SINGLE روند شده و در متغیر var از نوع داده SINGLE قرار می‌گیرد.

```
Round(2.3) = 2 , Round(2.8) = 3
Round(-2.3) = -2 , Round(-2.8) = -3
```

• مثال

```
Dim S As 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 متغیرها به یکدیگر

دستور ASC

Var = ASC(string)

این دستور اولین کاراکتر یک متغیر 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 , WORD یا LONG را به مقدار هگزادسیمال تبدیل می‌کند.

var = Hex (x)

مقدار HEX متغیر یا ثابت X در متغیر 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 هگزادسیمال را به مقدار عددی (دسیمال) تبدیل می‌کند.

`var = HEXVAL(x)`

مقدار عددی داده هگزادسیمال x که می‌تواند BYTE, INTEGER, WORD یا LONG باشد در متغیر 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 متغیر یا ثابت var2 را تبدیل به مقدار BCD می‌کند و در متغیر 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 متغیر یا ثابت VAR2 در متغیر 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 متغیر 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 m را با تعداد تکرار n تبدیل به رشته کرده و در متغیر 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 )
```

متغیر var2 که می‌تواند داده‌ای از نوع BYTE , INTEGER , WORD یا LONG باشد به گذگری تبدیل شده و در متغیر 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 )
```


کدگری var2، به مقدار باینری تبدیل شده و در متغیر 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 دستورانی که با رجیسترهای میکرو و آدرس‌های حافظه کار می‌کند بپردازیم. تمام میکروهای AVR دارای ۳۲ رجیستر ۸ بیتی (R0-R31) همه‌منظوره در CPU خود هستند. رجیسترهای R31(MSB) تا R30 (LSB)، R29(MSB) تا R28 (LSB) و R27(MSB) تا R26 (LSB) تشکیل سه رجیستر ۱۶ 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 متغیری از نوع داده BYTE، INTEGER، WORD یا LONG است. X برای BYTE می‌تواند ۰ تا ۷، ۰ تا ۱۵ برای WORD و برای LONG می‌تواند ۰ تا ۳۱ 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.

دستورات محیط BASCOM ۱۵۳

Reset Bit/pin
Reset Var.x

Bit می‌تواند یک بیت و یا یک SFR مانند PORTB.1 باشد و Var متغییری از نوع داده WORD, INTEGER, BYTE, 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 تغییر کند.

• 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 کدی (FLASH MEM) میکرو MEGA103 یا دیگر میکروها که دارای 128K حافظه است را خواند.

var = CPEEKH(address)

address آدرس حافظه و محتوای آدرس در متغیر 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 متغیر را در یک جفت رجیستر ذخیره کنید. VAR متغیری است که آدرس آن در متغیرهای دو بیتی 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 فرستاده 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 متغیر var قرار می‌گیرد. دستور INP می‌تواند از تمام مکانهای حافظه AVR بخواند. در ضمن 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 )
```


۱۰۰ دستورات محیط BASCOM

address آدرس رجیسترهای R0 تا R31 است که بین 0 تا 31 می باشد. محتوای رجیستر در متغیر var جای می گیرد. دستور (PEEK) فقط می تواند 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 می توانیم یک بایت داده را در یکی از رجیسترها بنویسیم.

POKE address , value

مقدار متغیر یا ثابت یک بایتی value در آدرس address که بین 0 تا 31 برای رجیسترهای R0-R31 است نوشته می شود.

• مثال

```
Poke 1 , 5      'write 5 to R1
End
```

دستور VARPTR

این دستور آدرس یک متغیر را در مکان حافظه بر می گرداند.

var = VARPTR(var2)

آدرس متغیر var2 در مکان حافظه بدست آمده و در متغیر var قرار می گیرد.

• مثال

```
Dim B As Xram Byte At &H300 , I As Integer , W As Word
W = Varptr(b)
Print Hex(w)      'Print &H0300
End
```

۵-۶ دستورات عملهای حلقه و پرش

دستور JMP و GOTO

GOTO label

JMP lable

با این دستورات می توان به برچسب label پرش کرد. برچسب label باید با علامت : (collon) پایان یابد و می تواند تا 32 کاراکتر طول داشته باشد. به خاطر داشته باشید زمانی که از دو label هم نام استفاده شود کامپایلر به شما هشدار (Warning) می دهد. دستور RETURN برای برگشت از برچسب 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 می باشد.

```
DO
  statements
LOOP [ UNTIL expression ]
```

دستورالعمل Statements تا زمانی که expression دارای ارزش TRUE یا غیر صفر است تکرار خواهد شد بنابراین این نوع حلقه، حداقل یکبار تکرار شود. DO - LOOP به تنهایی یک حلقه بینهایت است که با EXIT DO می توان از درون حلقه خارج شد و اجرای برنامه در خط بعد از حلقه 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 می باشد.

```
FOR var = start TO end [ STEP value ]
  statements
Next var
```

VAR بعنوان یک کانتر عمل می کند که START مقدار اولیه و END مقدار پایانی است و هر دو می توانند یک ثابت عددی یا متغیر عددی باشند. VALUE مقدار عددی STEP (قدمها) را نشان می دهد که می تواند مثبت یا منفی باشد. در صورت حذف کردن STEP VALUE کامپایلر بصورت پیش فرض مقدار یک را در نظر می گیرد.

نوشتن نام متغیر var بعد از NEXT الزامی نیست.

نکته

• 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 می دهد که تکرار این حلقه زمانی ادامه می یابد

که عبارت بکار برده شده نتیجه FALSE را حاصل کند و یا دارای مقدار صفر شود. 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 دستورالعمل ساده یا چند دستورالعمل مرکب 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 expression1 دارای ارزش TRUE باشد دستورالعمل statement1 اجرا خواهد 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 این صورت یعنی در حالتی که هر دو عبارت expression1 و expression2 باشند دستورالعمل statement3 اجرا خواهد شد.

همچنین با دستور IF می‌توان یک یا صفر بودن یک بیت از یک متغیر 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 0 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 دستورالعمل های انشعاب یا پرش جهت کنترل اجرای دستورات یک برنامه را تغییر داد. یکی از این دستورات SELECT - CASE است که میتوان 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
```

اگر متغیر Var با مقدار test1 برابر باشد statement1 اجرا می شود و سپس اجرا برنامه بعد از End Select ادامه می یابد در غیر اینصورت اگر متغیر Var با مقدار test1 برابر نباشد ولی با مقدار test2 برابر باشد statement2 اجرا می شود و سپس اجرا برنامه بعد از End Select ادامه می یابد و نهایتاً اگر متغیر Var با هیچکدام از مقادیر test1 و test2 برابر نباشد، statement3 اجرا می شود و سپس اجرای برنامه بعد از End select ادامه می یابد.

شما می توانید به صورت های زیر نیز متغیر را امتحان کنید :

CASE IS > 2 : اگر متغیر مورد نظر بزرگتر از دو باشد.

و یا می توان محدوده ای را برای امتحان کردن در نظر گرفت :

CASE 2 TO 5 : اگر متغیر مورد نظر بین 2 تا 5 باشد.

• مثال

```
Dim X As Byte
Do
Input "X ? " , X
Select Case 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
```


159 دستورات محیط BASCOM

```
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 ادامه برنامه را بعد از ساختار یا 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
```

دستور العمل ON VALUE

با این دستور با توجه به مقدار متغیر می‌توان به توابع یا برچسب‌های مختلفی پرش کرد.

ON var [GOTO][GOSUB] label1 [,label2]

var متغیر مورد نظر برای امتحان شدن که میتواند 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.

زمانی که GOSUB استفاده می‌نماید باید با دستور RETURN از برنامه برگردد.

نکته

• مثال

```
Dim X As Byte
X = 1
On X GOSUB Lbl2 , Lbl3          'jump to sub lbl3
X = 0                           'jump to label lbl1
On X GOTO Lbl1 , Lbl4
Lbl4:
Print X
Lbl1:
Incr X                          'PRINTS 1
Print X
End

Lbl2:
Print " lbl3 "
Return

Lbl3:
Decr X                          'PRINTS 0
Print X
Return
```


۷-۵ 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 (1 - 255) باشد.

دستورات تأخیری زمان دقیق را به شما نمی‌دهد. برای بدست آوردن زمان دقیق از تایمرها استفاده نمایید.

نکته

• مثال

```
Waitus 10          'wait for 10 us
Print "bascom"
End
```

دستور WAITms

برای ایجاد تأخیر در برنامه از این دستور استفاده می‌شود.

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 (1 - 65535) باشد.

• مثال

```
Waitms 10          'wait for 10 ms
Print "bascom"
End
```

دستور WAIT

برای ایجاد تأخیر در برنامه از این دستور استفاده می‌شود.

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"
```


۵-۸ 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 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 متغیر به تابع فرستاده شود و در محتوای آن هیچ تغییری ایجاد نشود ولی در حالت BYREF آدرس متغیر ارسال و تغییرات در آن تاثیر می‌گذارد و داده برگشتی در صورت انجام عملیات بر روی آن با مقدار اولیه خود برابر نخواهد بود. در صورت عدم استفاده از گزینه [BYREF/BYVAL] بصورت پیش فرض، 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 متغیر T قرار می‌گیرد. شما می‌توانید در محدوده تابع یک متغیر محلی معرفی نمایید.

```
Declare Function Myfunction(byval I As Integer , S As String) As Integer
Dim K As Integer
Dim Z As String * 10
Dim T As Integer          'assign the values
K = 5 : Z = "123":
T = Myfunction(K , Z)
LcdT                      ' T= 25
LcdZ                      ' Z= Bascom
Lcd K                     ' K= 5
End

Function Myfunction(byval I As Integer , S As String) As Integer
Local P As Integer      'you can also use local variables in subs and
functions
P = I * 5
I = 5
S = "Bascom"
T = P
Myfunction = T
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 بر نمی‌گرداند. در زمان ارسال داده به صورت BYREF (پیش فرض 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 متغیر 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
```

(CALL) فراخوانی

توسط این دستور زیربرنامه یا تابعی را فراخوانی (CALL) می‌کنیم.

CALL TEST(VAR1 , VAR2 , ...)

VAR1 و VAR2 متغیرهایی 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
Calltest(a,5)                  'call test with parameter A and constant
testa,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 متغیر محلی یا LOCAL

از این دستور برای تعریف متغیر محلی در زیر تابع یا زیربرنامه استفاده می‌نماییم.

LOCAL var As Type

VAR نام متغیر و TYPE نوع داده است که می‌توانند BYTE, INTEGER, WORD, STRING

SINGLE یا LONG باشد. نوع داده‌های XRAM, SRAM, ERAM و آرایه‌ها نمی‌توانند محلی تعریف

شوند. یک متغیر محلی یک متغیر موقت است که فقط در هنگام فراخوانی زیربرنامه مربوطه برای آن فضا در نظر گرفته می‌شود و با برگشت از زیربرنامه عمر متغیر (LIFE TIME) به اتمام می‌رسد.

متغیرهای بیتی نمی‌توانند بصورت محلی تعریف شوند.

نکته

• 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

این دستور به زیربرنامه پرش می‌کند و اجرای برنامه را از آدرس برجسب ادامه می‌دهد.

GOSUB label
 این دستور به زیربرنامه پرش می‌کند و اجرای برنامه را از آدرس برجسب ادامه می‌دهد.
 نام LABEL برجسبی زیربرنامه است که به آن پرش می‌شود. توسط دستور RETURN می‌توان از
 SUB برگشت کرد و اجرای برنامه بعد از دستور GO SUB ادامه یابد.

• مثال

```

Dim X as byte
Gosub Routine
Print "Hello"
End

Routine:
X = X + 2
Print X
Return

'jump to Routine
'After come back from Routin print "Hello"
'Terminate Program

'This Is A Subroutine
'Perform Some Math
' Print Result 2
'Return

```




پیکره‌بندی و کار با امکانات AVR در BASCOM

امکانات تمام میکروهای AVR قبل از استفاده 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 و سپس نحوه پیکره‌بندی و دستورات مربوط به وسیله کاملاً تشریح شده است 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 است که 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
۳. راه‌اندازی وسیله دلخواه و تحلیل نرم‌افزاری توسط تحلیلگر داخلی BASCOM
۴. راه‌اندازی وسیله دلخواه و تحلیل نرم‌افزاری توسط تحلیلگر PROTEUS

۱-۶ 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

X و Y بسته به میکرو می‌توانند به ترتیب پایه‌های 0 تا 7 پورت‌های A,B,C,D,E,F 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 پورتهای پورتی بخوانید بایستی از رجیستر PIN پورت مربوطه استفاده کنید و در هنگام نوشتن در پورت بایستی در رجیستر PORT بنویسید.

• مثال

```
Dim A As Byte , Count As Byte
'configure PORT D for input mode
Config Portd = Input
A = Pind
A = A And Portd
Print A
Bitwait Pind. 7 , Reset
'We will use port B for output
Config Portb = Output
'assign value
Portb = 10
Portb = Portb And 2
Set Portb. 0
Incr Portb
Count = 0
Do
    Incr Count
    Portb = 1
    For A = 1 To 8
        Rotate Portb , Left
        Wait 1
    Next
Loop Until Count = 10
End
```

```
'Read Data On Portd
'A=A & PORTD
'print it
'wait until bit is low

'set port B to 10
'set bit 0 of port B to 1

'rotate bits left
```

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

PINA، آدرس پایه های ورودی پورت A فقط قابل خواندن است در صورتیکه رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند. تمام پایه های پورت دارای مقاومت Pull-Up مجزا هستند. بافر خروجی پورت A می تواند تا 20mA را Sink کند و در نتیجه LED را مستقیماً راه اندازی می کند. هنگامیکه که پایه های PA0-PA7 با مقاومت های Pull-Down خارجی، خروجی استفاده می شوند، آنها SOURCE جریان می شوند زمانی که مقاومت های 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	DDA5	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 پایه های ورودی پورت A - PINA [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

PINA یک رجیستر نیست. این آدرس دسترسی به مقدار فیزیکی بر روی هریک از پایه های پورت A را ممکن می سازد. زمانیکه پورت A (PORTA) خوانده می شود، داده لچ (Latch) پورت A خوانده می شود و زمانیکه از PINA خوانده می شود مقدار منطقی که بر روی پایه ها موجود است خوانده می شود.

استفاده از پورت A بعنوان یک I/O عمومی دیجیتال

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

جدول تاثیر تغییرات DDAn بر روی پایه های PORTA

تمام 8 پایه موجود زمانیکه بعنوان پایه های I/O دیجیتال استفاده می شوند دارای عملکرد مساوی هستند. PAn، پایه I/O عمومی: بیت DDAn در رجیستر DDRA مشخص کننده جهت پایه است. اگر

دیگر کاربردهای پورت A

پورت B

رجیسترهای پورت B

رجیسترهای پورت B عبارتند از:

[PORT B DATA REGISTER] PORTB -B رجیستر داده پورت

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

رجسٹر جهت داده پورت B - DDRB [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

بابت آدرس پايه‌های ورودی پورت B [PORT B INPUT PINS ADDRESS] PINB-B

[illegible]

PINB یک رجیستر نیست. این آدرس دسترسی به مقدار فیزیکی بر روی هریک از پایه‌های پورت B را ممکن می‌سازد. زمانیکه پورت B (PORTB) خوانده می‌شود، داده لچ (Latch) پورت B خوانده و زمانیکه از PINB خوانده می‌شود مقدار منطقی که بر روی پایه‌ها موجود است خوانده می‌شود.

استفاده از پورت B بعنوان یک I/O عمومی دیجیتال

تمام ۸ پایه موجود زمانیکه 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 بعنوان یک پایه خروجی مورد استفاده قرار می‌گیرد و اگر DDBn صفر باشد، PBN بعنوان یک پایه ورودی در نظر گرفته می‌شود. 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 بعنوان خروجی تعریف شود. پایه‌های پورت زمانیکه ریست اتفاق می‌افتد به حالت Tri-state می‌روند.

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

جدول تاثیر تغییرات DDAn بر روی پایه‌های PORTA

دیگر کاربردهای پورت B

Port Pin	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	MOSI (SPI Bus Master Input/Slave Output)
PB7	SCK (SPI Bus Serial Clock)

جدول 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 و در 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 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 DDD7 می‌تواند برای خروجی مُد مقایسه‌ای Timer/Counter0 شکل‌دهی شود.

▪ PORTB.2 - INT2 , AIN0

AIN0: ورودی مثبت مقایسه‌کننده آنالوگ است.
INT2: دیگر کاربرد این پایه بعنوان منبع وقفه خارجی دو است. پایه PB2 می‌تواند بعنوان منبع وقفه خارجی برای میکرو (MCU) 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

PINC است. آدرس پایه‌های ورودی پورت C فقط قابل خواندن است در صورتیکه 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. هنگامیکه 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	PINC7	PINC7	PINC7	PINC7	PINC7	PINC7	PINC7
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

PINC یک رجیستر نیست. این آدرس دسترسی به مقدار فیزیکی بر روی هریک از پایه‌های پورت C را ممکن می‌سازد. زمانیکه پورت C (PORTC) خوانده می‌شود، داده لچ (Latch) پورت C خوانده می‌شود و زمانیکه از PINC خوانده می‌شود مقدار منطقی که بر روی پایه‌ها موجود است خوانده می‌شود.

استفاده از پورت C بعنوان یک I/O عمومی دیجیتال

تمام 8 پایه موجود زمانیکه بعنوان پایه‌های I/O دیجیتال استفاده می‌شوند دارای عملکرد مساوی هستند. PCn، پایه عمومی I/O: بیت DDcn در رجیستر DDRC مشخص کننده جهت پایه است. 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 PCn یک باشد هنگامیکه پایه بعنوان ورودی تعریف شود، مقاومت Pull-Up فعال می‌شود برای خاموش کردن مقاومت Pull-Up باید Port Cn صفر شود یا اینکه پایه بعنوان خروجی تعریف شود. پایه‌های پورت در زمانیکه ریست اتفاق می‌افتد به حالت Tri-State می‌روند.

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

جدول تاثیر تغییرات DDAn بر روی پایه‌های PORTA

دیگر کاربردهای پورت C

Port Pin	Alternate Functions
PC0	TOSC2 (Timer Oscillator Pin 2)
PC1	TOSC1 (Timer Oscillator Pin 1)
PC2	TDI (JTAG Test Data In)
PC3	TDO (JTAG Test Data Out)
PC4	TMS (JTAG Test Mode Select)
PC5	TCK (JTAG Test Clock)
PC6	SDA (Two-Wire Serial Bus Data Input/Output Line)
PC7	SCL (Two-Wire Serial Bus Clock Line)

جدول دیگر کاربردهای پورت C

PORTC.7 - TOSC2 ▪

TOSC2: زمانی که تایمر/کانتر 2 در مد آسنکرون کار می‌کند به این پایه و پایه TOSC1 کریستال ساعت متصل می‌شود. در این حالت دیگر نمی‌توان این پایه را بعنوان I/O استفاده نمود.

PORTC.6 - TOSC1 ▪

TOSC1: زمانی که تایمر/کانتر 2 در مد آسنکرون کار می‌کند به این پایه و پایه TOSC2 کریستال متصل می‌شود. در این حالت دیگر نمی‌توان این پایه را بعنوان I/O استفاده نمود.

PORTC.5 - TDI ▪

TDI: در زمان ارتباط JTAG بعنوان ورودی داده سریال عمل می‌کند و دیگر نمی‌توان از این پایه بعنوان I/O استفاده نمود.

PORTC.4 - TDO ▪

TDO: در زمان ارتباط JTAG بعنوان خروجی داده سریال عمل می‌کند و دیگر نمی‌توان از این پایه بعنوان I/O استفاده نمود.

PORTC.3 - TMS ▪

TMS: در زمان ارتباط JTAG استفاده می‌شود و دیگر نمی‌توان از این پایه بعنوان I/O استفاده نمود.

PORTC.2 - TCK ▪

TCK: در زمان ارتباط JTAG استفاده می‌شود و دیگر نمی‌توان از این پایه بعنوان I/O استفاده نمود.

PORTC.1 – SDA ▪

SDA : در زمان ارتباط WIRE – 2 بعنوان خط داده استفاده می‌شود.

PORTC.0 – SCL ▪

SCL : در زمان ارتباط WIRE – 2 بعنوان 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 در صورتیکه 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. هنگامیکه که PD0-PD7 با مقاومت‌های Pull-Down، 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 – PORTD [PORT D DATA REGISTER]

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 – DDRD [PORT D DATA DIRECTION REGISTER]

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 – PIND [PORT D INPUT PINS ADDRESS]

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

PIND یک رجیستر نیست. این آدرس دسترسی به مقدار فیزیکی بر روی هریک از پایه‌های پورت D را ممکن می‌سازد. زمانیکه پورت D (PORTD) خوانده می‌شود، داده لچ (Latch) پورت D خوانده می‌شود و زمانیکه از PIND خوانده می‌شود مقدار منطقی که بر روی پایه‌ها موجود است خوانده می‌شود.

استفاده از پورت D بعنوان یک I/O عمومی دیجیتال :

تمام ۸ پایه موجود زمانیکه بعنوان 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. 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

PORT PIN	ALTERNATE FUNCTION
PD0	RXD (UART INPUT LINE)
PD1	TXD (UART OUTPUT LINE)
PD2	INT0 (EXTERNAL INTERRUPT 0 INPUT)
PD3	INT1 (EXTERNAL INTERRUPT 1 INPUT)
PD4	OC1B (T/C 1 OUTPUT COMPAREB MATCH OUTPUT)
PD5	OC1A (T/C 1 OUTPUT COMPAREA MATCH OUTPUT)
PD6	ICP (T/C1 INPUT CAPTURE PIN)
PD7	OC2 (T/C2 OUTPUT COMPARE MATCH OUTPUT)

جدول دیگر کاربردهای پورت D

PORTD.7-OC2 ▪

OC2 : خروجی مُد مقایسه‌ای تایمر/کانتر ۲. 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/کانتر ۱ 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 تایمر ۱ 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.

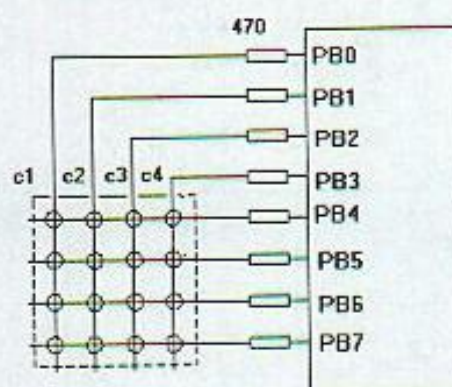
PORTD.0- RXD

RXD: دریافت داده (پایه ورودی داده برای USART)

زمانیکه دریافت USART فعال می‌شود پایه با توجه به DDD0 بعنوان ورودی شکل‌دهی می‌شود.

۲-۶ پیکره‌بندی صفحه کلید 4x4

اسکن صفحه کلید در BASCOM کار ساده است و تنها کافی است صفحه کلید خود را طبق شکل زیر به یکی از پورت‌های میکرو وصل نمایید و توسط دستور CONFIG KBD آن را پیکره‌بندی کنید.



شکل اتصال صفحه کلید 4x4 به پورت میکرو

صفحه کلید توسط دستور زیر پیکره‌بندی می‌شود.

CONFIG KBD = PORTx, DEBOUNCE = value [, DELAY = value]

PORTx مشخص کننده پورتهی 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 تاخیری 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 تاخیر در زمان اسکن صفحه کلید از دستورات تاخیر 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 متغیر 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 صفحه قبل عدد متناظر با کلید فشرده شده توسط دستور GETKBD() به صورت زیر دریافت خواهد شد.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

عدد متناظر با کلید فشرده شده
توسط دستور GETKBD()

• مثال

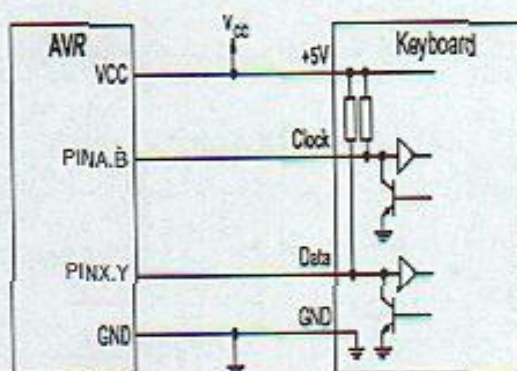
```
$regfile = "m32def.dat"
$crystal = 8000000
Config Lcdpin = Pin, Db4 = Pinb.4, Db5 = Pinb.5, Db6 = Pinb.6, Db7_
= Pinb.7, Rs = Pinb.2, E = Pinb.3
Config Lcd = 16 * 2
Config Kbd = Portc, Debounce = 50, Delay = 100
Dim A As Byte
Main:
  A = Getkbd()
  ' Wait 100ms By Compiler Because Of Delay=100
  If A > 15 Then Goto Main
  Home
  Lcd A
  JMP Main
End
```


۳-۶ پیکره‌بندی صفحه‌کلید کامپیوتر

در زمان اتصال صفحه‌کلید کامپیوتر به پایه‌های میکرو از پیکره‌بندی زیر استفاده می‌نماییم که PINA.B به پایه کلاک و PINX.Y به پایه DATA از صفحه‌کلید وصل می‌شود و از جدول KEYDATA برای تبدیل کلید فشرده شده به کد اسکی استفاده می‌شود زیرا کدهای گرفته شده از صفحه‌کلید اسکی نیستند.

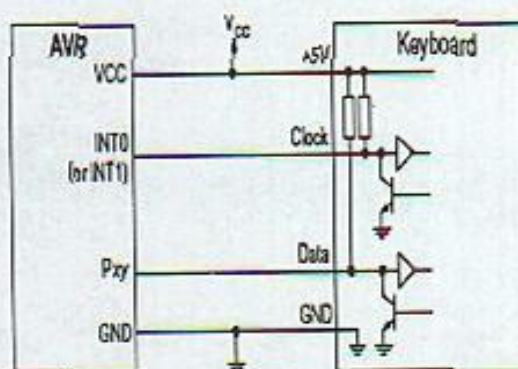
CONFIG KEYBOARD = PINA.B, DATA = PINX.y, KEYDATA = table

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 زیر اتصال یابد.



شکل اتصال صفحه‌کلید به دو پایه I/O از میکرو

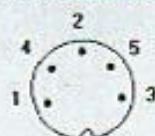
می‌توان صفحه‌کلید را طبق شکل زیر بصورت وقفه‌ای اتصال داد و پایه کلاک را به یکی از پایه‌های وقفه خارجی وصل کرد و زمانی که کلیدی فشرده شود وقفه خارجی فعال شده و نرم افزار در زیر برنامه وقفه، DATA را خوانده و مقدار را می‌گیرد.



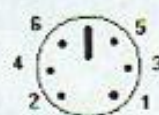
شکل اتصال صفحه‌کلید بصورت وقفه‌ای

ترکیب پایه صفحه‌کلید در پشت کامپیوتر

DIN41524 FEMALE
AT COMPUTER, 5-PIN DIN



6-PIN MINI DIN PS2 STYLE AT
COMPUTER




```
'shifted keys UPPER case
```

```
Data 0 , 0 , 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
```

۴-۶ 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 خط نوشته شود و یا ادامه آن با علامت _

(UNDER LINE) در خط بعد نوشته شود.

پایه‌های LCD (PIN) با توجه به پیکره‌بندی بالا باید به صورت زیر متصل شود.

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	3	GROUND
VDD	2	VCC
VSS	1	GROUND

جدول ترکیب پایه‌های LCD

تعیین نوع LCD

```
CONFIG LCD = LCDtype
```

LCDTYPE می‌تواند انواع زیر باشد:

40*4 = دارای 40 ستون و 4 سطر

16*1a = دارای 16 ستون و 1 سطر است. این نوع LCD، نوع ویژه‌ای است که بصورت LCD 2*8

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 نیازی به تعیین نوع LCD نیست.

و نیز می‌تواند از انواع 16*4 - 20*2 - 20*4 - 16*1 باشد.

• مثال

```

CONFIG LCD = 40 * 4
LCD "Hello"
FOURTHLINE
LCD "4"
END
' display on LCD
' select line 4
' display 4

```

پیکره‌بندی باس LCD

CONFIG LCDBUS = constant

در صورتی که بخواهیم از انتقال داده به LCD بصورت 4 بیتی (پیش فرض) یا 8 بیتی استفاده نماییم از این دستور استفاده می‌نماییم 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
LCD "hello"
' 4 bit mode

```

دستورات و توابع مربوط به LCD

دستور LCD

این دستور یک یا چند عبارت ثابت یا متغیر را بر روی LCD نمایش می‌دهد.

```

LCD x
LCD "constant"

```

x متغیر و constant ثابتی است که نمایش داده می‌شود.

برای نمایش چند عبارت پشت سر هم بین آنها علامت (semicolon) را قرار می‌دهیم.

```

LCD a ; b1 ; "constant"

```

• مثال

```

Dim A As Byte
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portb.1 , Db5 = Portb.2 , Db6 = Portb.3 , Db7 = _
Portb.4 , E = Portb.5 , Rs = Portb.6
Cls
Lcd "Hello world."
Wait 1
Lowerline
Wait 1
Lcd "Shift this."
Wait 1
For A = 1 To 10
ShiftLcd Right
Wait 1
Next
For A = 1 To 10
ShiftLcd Left
Wait 1
Next
Locate 2 , 1
Lcd "*"
Wait 1
Shiftcursor Right
Lcd "@"
' configure lcd screen
' clear the LCD display
' display this at the top line
' select the lower line
' display this at the lower line
' shift the text to the right
' wait 1s
' shift the text to the left
' wait 1s
' set cursor position
' display this
' wait 1s
' shift the cursor
' display this

```


۱۸۱ پیکره‌بندی و کار با امکانات AVR در BASCOM

```

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                           'first letter also works
Home F
Locate 4 , 1 : Lcd "Line 4"
End

```

دستور CLS

این دستور مخفف CLEAR SCREEN است که باعث می‌شود تمام صفحه نمایش LCD پاک شود.

• 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
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 چشمک‌نزدن است.

• مثال

```

Dim a As Byte
a = 255
LCD a
CURSOR OFF                          'hide cursor
Wait 1                              'wait 1 second

```



```
CURSOR BLINK      'blink cursor
End
```

دستور HOME

این دستورات مکان‌نما را به ترتیب در اولین ستون سطر اول ، سطر دوم ، سطر سوم یا سطر چهارم قرار می‌دهد.

HOME UPPER / LOWER / THIRD / FOURTH

دستورات فوق را به صورت ساده شده زیر نیز می‌توان نوشت :

HOME U / L / T / F

اگر دستور HOME بتنهای 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
```

دستور SHIFTLCD

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

۱۸۳ پیکره‌بندی و کار با امکانات AVR در BASCOM

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 دستور کرزر 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 بر روی صفحه LCD نمایش دهید. بعد از طراحی حرف یا علامت

دلخواه در LCD DESIGNER و کلیک کردن بر روی 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 می شوند و شما می توانید به جای ؟ عددی بین 0 تا 7 قرار دهید. بدین صورت شما می توانید تا 8 کاراکتر را طراحی کنید و بر روی LCD نمایش دهید. نمایش کاراکتر طراحی شده توسط دستور LCD CHR(?) بعد از دستور CLS انجام می گیرد.

• مثال

Deflcdchar 0 , 32 , 32 , 17 , 17 , 31 , 32 , 4 , 32

Cls

LCD Chr (0) 'show the character " ب "

End

۵-۶ پیکره بندی تایمر/کانترها

AVR ها (بجز MEGA128 که 4 تایمر دارد) 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 پیکره بندی آنها در محیط BASCOM پردازیم. در صورت وجود هر یک از تایمرها در میکرویی که شما با آن کار می کنید، می توانید به راحتی آن را در محیط BASCOM پیکره بندی و با آن کار کنید.

تایمر/کانتر صفر

معرفی تایمر/کانتر صفر و رجیسترهای مربوطه

تایمر/کانتر هشت بیتی صفر می تواند کلاک خود را از سیستم، تقسیمی از کلاک سیستم و یا از پایه خروجی تامین کند. تایمر/کانتر صفر توسط رجیستر کنترلی TCCR0 می تواند متوقف شود. وقفه های تایمر/کانتر توسط رجیستر TIMSK (TIMER/COUNTER1 INTERRUPT MASK ENABLE) می توانند فعال/ غیرفعال شود. پرچم سرریزی (OVER FLOW) در رجیستر TIFR موجود می باشد.

زمانی که تایمر/کانتر از پایه خروجی کلاک می خورد، سیگنال خروجی با فرکانس اسبلا تور CPU سنکرون (SYNCHRONIZE) می شود. بنابراین برای اطمینان از نمونه برداری مناسب، بایستی زمان بین دو کلاک خروجی حداقل برابر یک دوره تناوب کلاک CPU داخلی باشد. کلاک خروجی در لبه بالارونده کلاک داخلی CPU نمونه برداری می شود.

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 ۷..۳: بیت‌های رزرو شده

بیت‌های ۲، ۱، ۰ - CS00، CS01، CS02: انتخاب کلاک تایمر/کانتر صفر

این بیت‌ها طبق جدول زیر مشخص کننده PRESCALE برای TIMRE/COUNTER0 یا به عبارتی کلاک تایمر/کانتر صفر هستند.

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 TO FALLING ADGE
1	1	1	EXTERNAL PIN TO RISING ADGE

جدول مشخص کننده PRESCALE برای TIMRE/COUNTER0

رجیستر تایمر / کانتر ۰: TCNT0 [TIMER/CONTER0]

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 ۸ 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 مد تایمر با فرکانسهای سیستم تقسیم بر ۱، ۸، ۶۴، ۲۵۶ و ۱۰۲۴ کار می‌کند. با دستور START TIMER تایمر را 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 مقدار اولیه‌ای را می‌توان در تایمر صفر قرار داد و نیز محتوای تایمر/کانتر صفر را می‌توان با دستور VAR = TIMER0 خواند که VAR متغیری از نوع BYTE است.

تایمر پس از شمردن تا مقدار \$FF پرچم سرریزی خود را با نام OVFO یک می‌کند. در صورتی که وقفه سرریزی با دستور ENABLE OVFO و وقفه سراسری با ENABLE INTERRUPTS فعال شده باشند می‌توان در زمان سرریزی تایمر با دستور ON OVFO LABEL یا ON TIMER0 LABEL به LABEL پرش کرد و ISR سرریزی را اجرا کرد.

برگشت از وقفه سرریزی با دستور RETURN انجام می‌گیرد.

نکته

پیکره‌بندی به صورت کانتر

CONFIG TIMER0 = COUNTER, EDGE = RISING/FALLING

در این پیکره‌بندی تایمر/کانتر صفر بصورت کانتر استفاده شده است و میتوان شمارش آن را با لبه بالارونده یا پایین‌رونده فعال کرد. با انتخاب $EDGE = RISING$ ، اعمال هر لبه بالارونده به پایه T_0 باعث می‌شود که محتوای رجیستر $TCNT0$ یا متغیر $COUNTER0$ یک واحد افزایش یابد و همچنین با انتخاب $EDGE = FALLING$ ، اعمال هر لبه پایین‌رونده به پایه T_0 باعث می‌شود که محتوای رجیستر $TCNT0$ یا $COUNTER0$ یک واحد افزایش یابد.

کانتر پس از شمردن تا مقدار SFF و به تعداد $SFF+1$ 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 $ON\ OVF0\ LABEL$ یا $ON\ COUNTER0\ LABEL$ به پرش کرد و ISR سرریزی را اجرا کرد. محتوای تایمر/کانتر صفر را می‌توان با دستور $VAR = COUNTER0$ خواند که VAR متغیری 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

در این 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 ۸، ۱۶، ۲۵۶، ۱۰۲۴ می‌باشند. به طور مثال با کریستال ۸MHz و $PRESCALER = 1024$ ، تایمر با فرکانس $8MHz / 1024 = 7.8125KHz$ کار خواهد کرد.

```
$crystal = 8000000
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 صفر را نشان می‌دهد. زمانی که تایمر سرریز شود وقفه سرریزی رخ می‌دهد و زیر برنامه وقفه اجرا می‌شود.

```
Config Timer0 = Timer, Prescale = 1024
Enable Interrupts      'Global interrupt should Enable ,when using any
interrupt
```


۱۸۷ پیکره‌بندی و کار با امکانات AVR در BASCOM

```
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 تأمین کند. تایمر/کانتر 1 توسط رجیستر کنترلی TCCR1A و TCCR1B می‌تواند متوقف شود. وقفه‌های تایمر/کانتر توسط رجیستر TIMSK (TIMER/COUNTER1 INTERRUPT MASK ENABLE) می‌تواند فعال/غیرفعال شوند.

زمانی که تایمر/کانتر از پایه خروجی کلاک دریافت میکند، سیگنال خروجی با فرکانس اسپلاتور 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 و محتوای تایمر/کانتر وضعیت پایه‌های خروجی مُد مقایسه‌ای OC1A و OC1B (OUTPUT COMPARE PINS) می‌تواند تغییر یابند.

تایمر/کانتر همچنین می‌تواند بعنوان PWM (PULSE WIDTH MODULATOR) 9، 8 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 A تایمر/کانتر [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 - COM1A0, COM1A1 : COMPARE OUTPUT MODE 1 A

این دو بیت عملکرد پایه خروجی مُد مقایسه‌ای A را در زمان تساوی محتوای رجیستر مقایسه‌ای و محتوای تایمر/ کانتر را طبق جدول زیر نشان می‌دهد. پایه OC1A (OUTPUT COMPARE PIN1) خروجی مُد مقایسه A است که باید بعنوان خروجی تعریف شود.

بیت‌های 4, 5 - COM1B0, COM1B1 : COMPARE OUTPUT MODE 1 B

این دو بیت عملکرد پایه خروجی مُد مقایسه‌ای B را در زمان تساوی محتوای رجیستر مقایسه‌ای و محتوای تایمر/ کانتر را طبق جدول زیر نشان می‌دهد. پایه OC1B (OUTPUT COMPARE PIN1) خروجی مُد مقایسه‌ای B است که باید بعنوان خروجی تعریف شود.

COM1X1	COM1X0	DESCRIPTION
0	0	TIMER/COUNTER DISCONNECTED FROM OUTPUT PIN OC1X
0	1	TOGGLE THE OC1X OUTPUT LINE
1	0	CLEAR THE OC1X OUTPUT LINE (TO ZERO)
1	1	SET THE OC1X OUTPUT LINE (TO ONE)

جدول انتخاب عملکرد پایه‌های خروجی مُد مقایسه‌ای 1 (X = A OR B)

این دو بیتها (COM1X1, COM1X0) در حالت PWM دارای عملکرد متفاوتی هستند.

نکته

بیت‌های 2, 3 - بیت‌های رزرو شده**بیت‌های 0, 1 - PWM11, PWM10 : PULSE WIDTH MODULATOR SELECT BITS**

این دو بیت تایمر/ کانتر را بعنوان PWM با توجه به جدول زیر به کار می‌برند :

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

جدول انتخاب مُدهای PWM

رجیستر کنترلی B تایمر/ کانتر 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 ۶ - ICES1 : انتخاب لبه ورودی CAPTURE1 EDGE SELECT- INPUT CAPTURE1

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 ۴، ۵ - بیت‌های رزرو شده

بیت ۳ - 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 S0000 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, 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|....

این بیت در مد PWM تأثیری ندارد.

نکته

بیت‌های ۰، ۱، ۲ - CS12, CS11, CS10 : انتخاب کلاک 1 TIMER/COUNTER

این بیت‌ها طبق جدول زیر $PRESCALE$ تایمر/کانتر و یا به عبارتی فرکانس کاری تایمر/کانتر را با توجه به فرکانس اسیلاتور مشخص می‌سازند.

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

جدول انتخاب کلاک تایمر/کانتر ۱

زمانی که تایمر/کانتر از پایه خروجی کلاک دریافت می‌کند، بایستی تنظیمات مربوطه با توجه به جدول فوق صورت گیرد.

رجیستر تایمر / کانتر 1 - TCNT1H AND TCNT1L [TIMER/COUNTER1]

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 16 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.

• نوشتن TCNT1

زمانی که CPU در بایت بالا (TCNT1H) می‌نویسد، داده در رجیستر موقت TEMP (TEMPORARY) قرار می‌گیرد و سپس زمانی که CPU در بایت پایین (TCNT1L) می‌نویسد، این بایت با بایت نوشته در رجیستر TEMP ترکیب شده و تمام 16 بیت یکجا در رجیستر TCNT1 نوشته می‌شود. بنابراین برای نوشتن 16 بیت، ابتدا TCNT1H بایستی نوشته شود.

• خواندن TCNT1

زمانی که CPU بایت پایین (TCNT1L) را می‌خواند، محتوای بایت پایین TCNT1L به CPU ارسال می‌شود و محتوای بایت بالا (TCNT1H) در رجیستر موقت TEMP (TEMPORARY) قرار می‌گیرد و سپس زمانی که CPU بایت بالا را بخواند محتوای رجیستر TEMP به CPU ارسال می‌شود. بنابراین برای خواندن 16 بیت، ابتدا TCNT1L بایستی خوانده شود.

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 FLAG) یک می‌شود.

رجیستر ورودی CAPTURE - ICR1H AND ICR1L

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 ورودی CAPTURE (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 OCR1B، OCR1A، TCNT1 نیز استفاده شد.

تایمر/کانتر یک 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 یک در حالت PWM استفاده می‌شود، رجیستر مقایسه A (OCR1A) و رجیستر مقایسه‌ای B (OCR1B) در حالت‌های 8، 9 یا 10 بیتی برای تولید پالس PWM در پایه‌های OC1A و OCR1B استفاده می‌شوند.

تایمر/کانتر یک در مد PWM به صورت UP/DOWN COUNTER کار می‌کند. تایمر/کانتر یک در زمان UP-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 زیر بدست می‌آید که FTCK1 در جدول زیر به معنای فرکانس کاری تایمر/کانتر یک است.

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 فرکانسهای مختلف پالس PWM

با تغییر بیت‌های COM1X0 و COM1X1 می‌توان مدهای مختلف PWM را طبق جدول زیر انتخاب کرد.

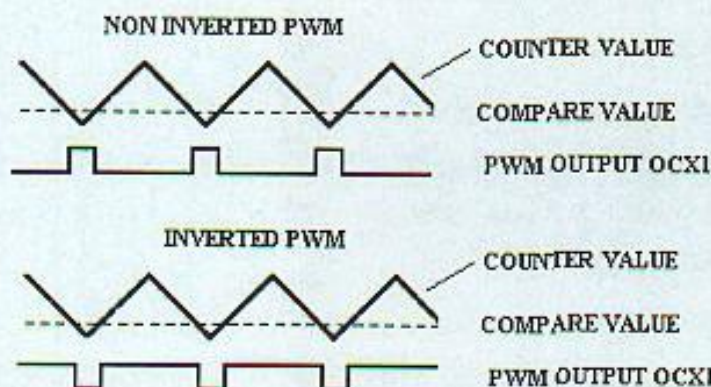
COM1X1	COM1X0	EFFECT ON OCX1
0	0	NOT CONNECTED
0	1	NOT CONNECTED
1	0	CLEAR ON COMPARE MATCH, UP-COUNTING. SET ON COMPARE MATCH, DOWN COUNTING (NON-INVERTED PWM)
1	1	CLEAR ON COMPARE MATCH, DOWN-COUNTING. SET ON COMPARE MATCH, UP COUNTING (INVERTED PWM)

جدول انتخاب مدهای مختلف پالس PWM

برای درک بیشتر تفاوت INVERTED PWM و NON-INVERTED PWM به جدول و شکل زیر توجه کنید.

COM1X1	COM1X0	OCR1X	OUTPUT OC1X
1	0	\$0000	L
1	0	TOP	H
1	1	\$0000	H
1	1	TOP	L

جدول خروجی پالس PWM به ازاء مقادیر مختلف OCR1X

شکل خروجی پالس
NON INVERTED PWMخروجی 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 OVFI یک می‌کند. در صورتی که وقفه سرریزی با دستور ENABLE OVFI و وقفه سراسری با ENABLE INTERRUPTS فعال شده باشند در زمان سرریزی تایمر می‌توان با دستور ON OVFI LABEL یا ON TIMER1 LABEL به LABEL پرش کرد و سرریزی را اجرا کرد. با دستور VAR = TIMER1 می‌توان محتوای تایمر/کانتر1 خواند که VAR متغیری 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 با 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 اسیلاتور RC 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 روئین وقفه OVFI 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 روئین سرریزی با دستور RETURN برگشت می‌کنیم. با دستور TIMER1= INITIAL VALUE می‌توان مقدار اولیه‌ای را در تایمر یک قرار داد.

```
$regfile = "M8535.DAT"
' INTERNAL RC OSC IS DEFAULT AND IF WE WORK WITH IT
Config Timer1 = Timer , Prescale = 1
Enable Interrupts
Enable Timer1
Enable Ovf1
On Ovf1 Gvflroutin
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 $\text{Prescale} = 8$ قرار می‌دهیم. فرکانس کلاک تایمر تغییری نمی‌کند و همان 1 MHz 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 Ovfl
On Ovfl Ovflroutin
Timer1 = 64536
Start Timer1
Do
A = Timer1
Print A          'or      Print Timer1
Loop

Ovflroutin:
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

با می‌توان چنین نوشت :

Config Timer1 = Counter , Eedge = Rising | Falling

تایمر/ کانتر 1 می‌تواند در مد 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 $\text{VAR} = \text{COUNTER1}$ خواند و با دستور $\text{COUNTER1} = \text{VAR}$ در محتوای کانتر نوشت. در هر دو حالت VAR متغیر از نوع داده WORD است. کانتر بعد از شمردن تعداد $\text{SFFFF} + 1$ پالس، سرریز می‌شود و سپس پرچم سرریزی خود را با نام OVFI یک می‌کند. در صورتی که وقفه سرریزی با دستور ENABLE OVFI و وقفه سراسری با ON OVFI LABEL فعال شده باشند می‌توان در زمان سرریزی کانتر با دستور ON OVFI LABEL یا ON COUNTER1 LABEL به پرش کرد و ISR سرریزی را اجرا کرد.

• مثال

پس از شمردن 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 بیتی 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 می‌تواند 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 متغیر نوع BYTE ، WORD یا INTEGER با مقادیر مثبت است. از این رجیسترها می‌توان با دستور VAR = COMPARE1A | B خواند که VAR متغیر 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) و یا ارتباط پایه با کانتر 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.

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 تأمین کند. تایمر با فرکانس تعیین شده همانطور که در قسمت پیکره‌بندی 1 TIMER / COUNTER 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 می‌تواند 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 متغیر نوع BYTE، WORD یا INTEGER با مقادیر مثبت باشد. از این رجیستر می‌توان با دستور $VAR = COMPARE1A | B$ خواند که VAR متغیر نوع WORD است.

OC1A : $Compare\ A = Clear | Set | Toggle | Disconnect$ در زمان تطابق مقایسه پایه خروجی OC1A می‌تواند یک (SET)، صفر (CLEAR)، معکوس (TOGGLE) و یا ارتباط پایه با تایمر قطع (DISCONNECT) شود.

OC1B : $Compare\ B = Clear | Set | Toggle | Disconnect$ 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 50000 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 روئین وقفه تطابق مقایسه‌ای A | B از دستور $ON\ OC1A|B\ LABEL$ استفاده می‌کنیم. زمانی که محتوای رجیسترهای مقایسه‌ای A یا B با محتوای تایمر یا کانتر برابر شود، زیر برنامه وقفه LABEL اجرا خواهد شد.

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.

نکته

• مثال

```
$regfile = "8535DEF.DAT
S crystal = 8000000
Config Timer1 = Counter , Edge = Falling , Compare A = Set , Compare B = _
Toggle , Prescale = 1
Enable interrupts
Enable Ocla
On Ocla 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 بالارونده (Capture Edge = Falling) استفاده می‌کنید اعمال یک لبه پایین‌رونده به پایه ICP باعث می‌شود که محتوای تایمر / کانتر 1 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 متغیری 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 ICPI همراه دستور ENABLE INTERRUPTS می‌توان وقفه CAPTURE را فعال کرد و با دستور ON ICPI LABEL در زمان رخ داد CAPTURE به زیربرنامه وقفه LABEL 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 Icpl
On Icpl 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 ، ۸ ، ۹ و ۱۰ 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 ۸ ، ۹ و ۱۰ بیتی باشد که در مُد ۸ ، ۹ و ۱۰ 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 نطابق 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 $PWM1A = VAR$ ، $PWM1B = VAR$ و یا همچنین با دستورات $COMPARE1A = VAR$ ، $COMPARE1B = VAR$ بنویسید که VAR می‌تواند ثابت یا متغیری ۱ یا ۲ بیتی باشد.

فرکانس PWM با توجه به معادله‌های زیر بدست می‌آید که F_{osc} فرکانس کلاک سیستم است :

$$PWM \text{ FREQUENCY} = F_{osc} / (510 * Prescale) \quad : \text{ PWM , 8 بیتی}$$

$$PWM \text{ FREQUENCY} = F_{osc} / (1022 * Prescale) \quad : \text{ PWM , 9 بیتی}$$

$$PWM \text{ FREQUENCY} = F_{osc} / (2046 * Prescale) \quad : \text{ PWM , 10 بیتی}$$

• مثال

فرکانس پالس PWM تولید شده برای خروجی A و B در مثال زیر $8000000/8*2046= 488.7585\text{HZ}$ است.

```
$regfile = *8515DEF.DAT
$CRYSTAL = 8000000
Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Up , Compare B Pwm_
=Clear Down , Prescale = 8
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 (TIMER/COUNTER MASK REGISTER) 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 S00 ریست شود.

نکته

در حالت تطابق مقایسه، نوشتن بر روی پایه OC2 نمی‌تواند وضعیت این پایه را تغییر دهد.

تایمر/کانتر 2 همچنین به عنوان PWM (PULSE WIDTH MODULATOR) 8 بیتی استفاده می‌شود.

رجیستر کنترلی تایمر/کانتر 2 - 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 PWM - PULSE WIDTH MODULATOR ENABLE

این بیت زمانی که یک است، مُد PWM برای TIMER/COUNTER2 فعال است.

بیت 5, 4, COM21, COM20: خروجی مُد مقایسه‌ای - COMPARE OUTPUT MODE

این دو بیت مشخص کننده وضعیت پایه OC2 (OUTPUT COMPARE) در زمان تطابق مقایسه هستند.

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)

جدول انتخاب عملکرد پایه‌های خروجی مُد مقایسه‌ای 2

بیت 3 - CTC2 : صفر شدن محتوای تایمر / کانتر در زمان تطابق مقایسه در زمان تطابق مقایسه محتوای تایمر / کانتر با عدد \$00 ریست می‌شود اگر این بیت یک باشد 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,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|.....

بیت CTC2 در مد PWM تأثیری ندارد.

نکته

بیت‌های 0, 1, 2 : CS20, CS21, CS22 : انتخاب کلاک

این بیت‌ها طبق جدول زیر مشخص کننده PRESCALE برای TIMRE/COUNTER2 یا به عبارتی کلاک تایمر / کانتر 2 هستند.

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

جدول انتخاب کلاک تایمر / کانتر 2

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.

[TIMER/CONTR2 OUPUT COMPARE] OCR2 - تایمر / کانتر

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 میدهد. محتوای این رجیستر مدام با محتوای تایمر/کانتر (TCNT2) مقایسه می‌شود و تغییراتی که در زمان تطابق مقایسه یعنی زمانی که محتوای OCR2 با TCNT2 یکی شود بر روی پایه OC2 می‌دهد در رجیستر TCCR2 مشخص شده است.

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.

نکته

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 زیر بدست می‌آید. FTCK2 فرکانس کلاک تایمر/کانتر دو است.

PWM RESOLUTION	TIMER TOP VALUE	FREQUENCY
8-BIT	\$FF(255)	FTCK2/510

جدول فرکانسهای مختلف پالس PWM

می‌توان با تغییر بیت‌های COM1X0 و COM1X1 مدهای مختلف PWM را طبق جدول زیر انتخاب کرد.

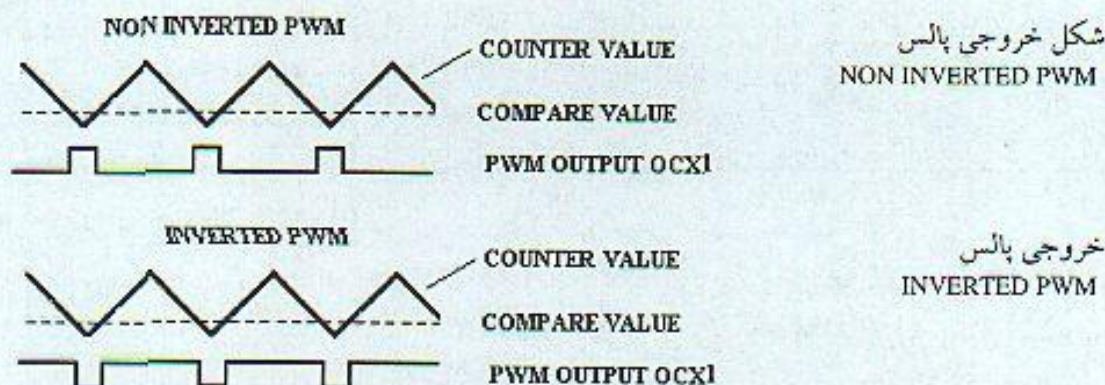
COM21	COM20	EFFECT ON OC2
0	0	NOT CONNECTED
0	1	NOT CONNECTED
1	0	CLEAR ON COMPARE MATCH, UP-COUNTING, SET ON COMPARE MATCH, DOWN COUNTING (NON-INVERTED PWM)
1	1	CLEAR ON COMPARE MATCH, DOWN-COUNTING, SET ON COMPARE MATCH, UP COUNTING (INVERTED PWM)

جدول انتخاب مدهای مختلف پالس PWM

برای درک بیشتر تفاوت INVERTED PWM و NON-INVERTED PWM به جدول و شکل‌های زیر توجه کنید.

COM21	COM20	OCR2	OUTPUT OC2
1	0	\$00	L
1	0	\$FF	H
1	1	\$00	H
1	1	\$FF	L

جدول خروجی پالس PWM به ازاء مقادیر مختلف OCR2



پیکره‌بندی تایمر/کانتر دو در محیط BASCOM

تایمر/کانتر 8 بیتی دو بسته به نوع میکرو می‌تواند در مدهای TIMER، COUNTER، COMPARE و PWM می‌تواند 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 مد کانتر عمل نماید به همین منظور پیکره‌بندی تایمر/کانتر به دو حالت یک و دو تقسیم‌بندی شده است. حالت یک برای میکروهایی است که T/C2 نمی‌تواند به صورت کانتر عمل نمایند و حالت دو برای میکروهایی است که T/C2 دارای مد کانتر نیز هست.

پیکره‌بندی تایمر/کانتر دو (حالت یک)

پیکره‌بندی تایمر/کانتر دو در مد تایمر

CONFIG TIMER2 = TIMER, ASYNC = ON | OFF, PRESCALE = 1 | 8 | 32 | 64 | 128 | 256 | 1024

تایمر UP - COUNTER دو در مد 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 LABEL در زمان سرریزی تایمر می‌توان به LABEL 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 متغیری از نوع BYTE است. ASYNC زمانی ON انتخاب می‌شود که تایمر

۲۰۳ پیکره‌بندی و کار با امکانات AVR در BASCOM

به صورت آستکرون از پایه‌های TOSC1 و TOSC2 با کریستال 32768HZ کلاک دریافت می‌کند. در این حالت با $PRESCALE=128$ دقیقاً تایمر بعد از یک ثانیه OVERFLOW می‌دهد ($128 \times 256 / 32768 = 1s$) 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 با 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 دستور START 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 = 128
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

Ovflroutin:
Stop Timer2
Print "OVERFLOW OCCURES"
Timer2 = 6
Start Timer2
Return

```

پیکره‌بندی تایمر/کانتر دو در مُد مقایسه‌ای (COMPARE)

تایمر دو و مُد مقایسه‌ای

CONFIG TIMER1 = TIMER , COMPARE = CLEAR | SET | TOGGLE | DISCONNECT_ , PRESCALE = 1 | 8 | 32 | 64 | 128 | 256 | 1024 , CLEAR TIMER = 1|0

نوسط 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 2 TIMER / COUNTER 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 دستور OCR2 = VAR قابل تغییر است که در آن VAR می‌تواند یک عدد ثابت یا یک متغیر نوع WORD , BYTE , INTEGER با مقادیر مثبت باشد. از این رجیستر می‌توان با دستور VAR = COMPARE خواند که VAR متغیر نوع WORD است.

OC2 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 گزینه 1 ، محتوای تایمر/کانتر دو در زمان تطابق مقایسه‌ای ریست و یا به عبارتی \$00 خواهد شد.

طرز کار با وقفه تطابق مقایسه (COMPARE MATCH)

برچم 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.

برای اجرا شدن وقفه تطابق مقایسه بایستی وقفه تطابق با دستور ENABLE OC2 و وقفه سراسری با دستور ENABLE INTERRUPTS فعال شده باشند.

نکته

• مثال

```

$regfile = "M8DEF.DAT"          'MEGA8 MCU
$crystal = 8000000
Config Timer2 = Timer , Compare = Set , Prescale = 8
Enable Interrupts

```


۲۰۵ پیکره‌بندی و کار با امکانات AVR در BASCOM

```
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 متغیری 1 بیتی است.

فرکانس PWM با توجه به معادله زیر بدست می‌آید که F_{osc} فرکانس کلاک سیستم است :

$$PWM \text{ FREQUENCY} = F_{osc} / (510 * Prescale) \quad : \text{ PWM , 8 بیتی}$$

پیکره‌بندی تایمر/کانتر دو (حالت دو)

پیکره‌بندی تایمر/کانتر دو در مُد تایمر

تنها تفاوت پیکره‌بندی تایمر در این حالت با حالت اول تنها در PRESCALE است. در این حالت مقادارهای 32 و 128 موجود نمی‌باشند.

CONFIG TIMER2 = TIMER , PRESCALE = 1 | 8 | 64 | 256 | 1024

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 / کانتر 2 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 متغیر WORD است. کانتر بعد از شمردن تعداد SFF+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, PRESCALE = 1 | 8 | 64 | 256 | 1024, COMPARE = CLEAR | SET_ | TOGGLE | DISCONNECT

با این دستور تایمر / کانتر دو در حالت تایمر استفاده شده و می‌تواند فرکانس کلاک خود را از فرکانس سیستم بخش بر 1, 8, 64, 256, 1024 تامین کند. تایمر با فرکانس تعیین شده همانطور که در بخش پیکره‌بندی 2 / COUNTER 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 را می‌توان با دستور OCR2 = VAR تغییر داد که در آن VAR می‌تواند یک عدد ثابت یا یک متغیر نوع BYTE, WORD یا INTEGER با مقادیر مثبت باشد. از این رجیستر می‌توان با دستور VAR = OCR2 خواند که VAR متغیر نوع WORD است.

OC2 پایه خروجی مقایسه تطابق : Compare = Clear | Set | Toggle | Disconnect

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 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 رجیستر مقایسه‌ای 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 $OCR2 \approx VAR$ تغییر داد که در آن VAR یک عدد ثابت یا یک متغیر نوع BYTE، WORD یا INTEGER با مقادیر مثبت می‌باشد. از این رجیستر می‌توان با دستور $VAR = OCR2$ خواند که VAR متغیر نوع WORD است.

OC2 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 گزینه 1، محتوای تایمر/ کانتر 2 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 LABEL استفاده می‌کنیم. زمانی که محتوای رجیستر مقایسه‌ای با محتوای تایمر برابر شود، زیر برنامه LABEL 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

```
Sregfile = *M32DEF.DAT
Scrystal = 8000000
Config Timer2 = Timer, Compare = Set, Prescale = 1
Enable Interrupts
Enable Oc2
On Oc2 Comparematch
OCR2 = 100
Do
' 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 2 دارای خروجی PWM، 8 بیت نیز می‌باشد. در این حالت پایه‌های OC2 به عنوان خروجی پالس PWM عمل می‌نمایند.

CONFIG TIMER2 = PWM, PRESCALE = 1 | 8 | 64 | 256 | 1024, PWM = ON | OFF
COMPARE PWM = CLEAR UP | CLEAR DOWN | DISCONNECT

Prescale : 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 پالس PWM به صورت INVERTED در پایه خروجی OC2 ایجاد می‌شود.

Clear Down: در صورت استفاده از این گزینه پالس PWM به صورت NON - INVERTED در پایه خروجی OC2 ایجاد می‌شود.

Disconnect: در صورت استفاده از این گزینه پالس PWM در زمان تطابق مقایسه از پایه خروجی OC2 قطع می‌شود.

برای تولید PWM می‌توانید در رجیستر PWM که همان رجیسترهای مقایسه‌ای است با دستورات $OCR2 = VAR$ بنویسید که VAR ثابت یا متغیری ۱ بایستی است.

فرکانس PWM با توجه به معادله زیر بدست می‌آید که F_{osc} فرکانس کلاک سیستم است:

$$PWM \text{ FREQUENCY} = F_{osc} / (510 * Prescale)$$

PWM، ۸ بیتی:

• مثال

```
$regfile = "M103DEF.DAT"
$CRYSTAL = 8000000
Config Timer2 = Pwm , Prescale = 8 , Pwm=on, Compare Pwm = Clear Up
Do
OCR2 = 100
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 و TXD 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 می‌دهیم.

UART سخت‌افزاری

در صورت استفاده از این حالت از دستورات زیر می‌توانید برای ارتباط استفاده نمایید.

تعیین میزان باود

$SBAUD = X$

X مقدار باودی است که در ارتباط سریال استفاده می‌شود. این مقدار با فرکانس کریستال رابطه مستقیم دارد و خطای حاصل از آن در ارتباط سریال با مقدار بیش از ۱٪ 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
$crystal = 8000000
Print "Hello"
End
'8 MHz crystal
'send "hello" to serial Port
```

تغییر میزان باود در برنامه

BAUD = VAR2

این دستور برای تغییر میزان باود در برنامه اضافه می‌شود. مقدار اولیه باود با دستور SBAUD = X داده شده است.

• مثال

```
$Baud = 2400
$crystal = 14000000
Print "Hello"
Baud = 9600
Print "Hello"
End
'14 MHz crystal
'send to serial port 2400 bps
'changing baud rate
'send to serial port 9600 bps
```

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 می‌شود.

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 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 شود، در این صورت شما می‌توانید از TERMINAL EMULATOR نرم‌افزار BASCOM به عنوان خروجی داده سریال استفاده نمایید. دستور PRINT به‌تنهایی باعث ایجاد یک خط خالی در TERMINAL EMULATOR می‌شود.

• 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 = 13
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 متغیر VAR و متغیر اختیاری VARN به باینری تبدیل می‌شوند و به پورت سریال ارسال می‌شوند. همچنین چندین متغیر 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

پیکره‌بندی CONFIG SERIALIN

زمانی که بخواهید از محیط TERMINAL EMULATOR برای ارسال داده به میکرو استفاده نمایید، 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

```
Scrystal = 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
```


۲۱۱ پیگره‌بندی و کار با امکانات AVR در BASCOM

```
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 متغیر 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.

در صورت خالی بودن بافر دستور INKEY عدد صفر را برمی‌گرداند.

نکته

• 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 متغیر var و متغیر 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) می خواند.

INPUTBIN var1 [,var2]

زمانی که از محیط TERMINAL EMULATOR استفاده می کنید این دستور به کار برده می شود. توسط این دستور می توانید 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 می شود. داده گرفته شده از صفحه کلید در متغیر var1 و متغیر اختیاری var2 جای می گیرد که اگر به عنوان BYTE تعریف شده باشد یک بایت، به ازاء INTEGER دو بایت و اگر یک آرایه تعریف شود به تعداد آرایه ها کاراکتر دریافت می کند. این دستور برای گرفتن تعداد بایتها در همان خط می ایستد.

• مثال

```

Dim A As Byte , C As Integer
Inputbin A , C 'Wait For 3 Characters
End

```

دستور INPUTHEX

این دستور اجازه می دهد که در هنگام اجرای برنامه از صفحه کلید ورودی دریافت کنیم.

INPUTHEX [" prompt"] , var [, varn] [NOECHO]

زمانی که از محیط TERMINAL EMULATOR استفاده می کنید این دستور به کار برده می شود. توسط این دستور می توانید 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 می تواند بین 0-9 و A-F باشد در متغیر var و متغیر اختیاری varn قرار می دهد و داده از طریق پورت سریال به میکرو ارسال می شود. PROMPT یک رشته ثابت دلخواه و اختیاری که می خواهید قبل از کاراکتر وارد شده در محیط TERMINAL نوشته شود. NOECHO نیز باعث می شود که مقدار وارد شده از صفحه کلید در محیط TERMINAL نمایش داده نشود. تعداد بایت ورودی بستگی به تعریف متغیرهای var و varn دارد که به صورت زیر تعیین می شوند :

۲۱۳ پیکره‌بندی و کار با امکانات AVR در BASCOM

- اگر var یا varn بایت تعریف شده باشند ورودی هر یک نهایتاً ۲ کاراکتر می‌تواند طول داشته باشد.
- اگر var یا varn ، INTEGER یا WORD تعریف شده باشند، ورودی هر یک نهایتاً ۴ کاراکتر می‌تواند طول داشته باشد.
- اگر var یا varn ، LONG تعریف شده باشند، ورودی هر یک نهایتاً ۸ 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 (4 bytes) Enter hex number در Terminal Emulator نوشته می‌شود و دستور Inputhex داده را از صفحه کلید می‌گیرند 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
```

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 پریته (parity) می‌تواند N برای NONE، O برای ODD (فرد) و E برای EVEN (زوج) استفاده شود. پارامتر اختیاری INVERTED منطق RS-232 را معکوس می‌کند.

MODE: می‌تواند گزینه INPUT برای بکارگیری pin بعنوان پایه RXD مجازی و یا گزینه OUTPUT 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 9600,8,N,2 COMB.0 از پایه صفر پورت B با نرخ انتقال (baud rate) 9600 و دو بیت STOP BIT و هشت بیت داده برای تبادل سریال توسط UART نرم‌افزاری استفاده

می‌شود. دستورات PRINT ، INPUT ، INPUTEX ، INKEY ، WAITKEY کانالهای فوق را نیز پوشش می‌دهند. هر کدام از کانالهای باز شده بعد از استفاده با دستور CLOSE #CHANNEL 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
```

تعیین میزان باود

\$BAUD #CHANNEL , X

X مقدار باودی است که در ارتباط سریال استفاده می‌شود. این مقدار با فرکانس کریستال رابطه 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
Print #1, "Hello"
Baud #1, 9600
Print #1, "Hello"
End
End

```

'8 Mhz crystal
'send to serial port 2400 bps
'changing baud rate
'send to serial port 9600 bps

ارسال داده در حالت UART نرم‌افزاری

دستور PRINT

این دستور از طریق UART نرم‌افزاری، داده به پورت سریال RS-232 ارسال می‌کند.

```
PRINT #CHANNEL, var
```

که VAR متغیر، مقدار عددی یا رشته اختیاری است که به پورت سریال RS-232 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.

استفاده از PRINT var در کنار دستور PRINT #CHANNEL, var به معنای ارسال داده از پورت سریال اصلی میکرو (پایه TXD) است.

نکته

• 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."
Do
B = Inkey(# 2 )
'when the value > 0 we got something
If B > 0 Then
Print # 1 , Chr ( b )
End If
Loop Until B = 27
Close # 2
Close # 1
'use normal hardware UART for printing
Print B
End

```

'constant to print
'print the character

دستور PRINTBIN

```
Printbin #channel, Var [ ; Varn]
```

channel شماره کانال باز شده توسط UART نرم‌افزاری است. محتوای متغیر VAR و متغیر اختیاری VARN به باینری تبدیل می‌شوند و به پورت سریال نرم‌افزاری فرستاده می‌شوند. همچنین چندین متغیر 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 نرم افزاری

دستور WAITKEY

```
Var = Waitkey(#Channel)
```

این دستور تا زمانی که در بافر UART نرم افزاری (SOFT UART) با شماره کانال Channel، کاراکتری دریافت شود منتظر می ماند پس از دریافت کاراکتر آن را در متغیر var قرار می دهد و اجرای برنامه از خط بعد ادامه پیدا می کند.

• 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 نرم افزاری با شماره کانال Channel را در متغیر VAR قرار می دهد و اجرای برنامه از خط بعد ادامه پیدا می کند.

```
Var = Inkey(#Channel)
```

VAR می تواند BYTE، INTEGER، WORD و یا LONG باشد.

در صورت خالی بودن بافر دستور Inkey(#Channel) عدد صفر را بر می گرداند.

نکته

• مثال

```

Open "comd.2:9600,8,n,1,inverted" For Input As #2
Do
A = Inkey(#2)      'Start Loop
If A > 0 Then      'Look For Character
Print A           'Is Variable > 0
End If             'Send A to hardware Uart (TXD PIN)
Loop               'Loop Forever

```

دستور INPUT

```
INPUT #Channel, var
```

این دستور همانند دستور INPUT در UART سخت افزاری عمل می نماید که #Channel شماره کانال UART نرم افزاری است. با این دستور می توان هر نوع داده ای 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 نمی شود.

• مثال

```

$regfile = "M32DEF.dat"
$crystal = 8000000

```


۲۱۷ پیکره بندی و کار با امکانات AVR در BASCOM

```
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) می خواند.

```
INPUTBIN #channel, var1 [,var2]
```

زمانی که از محیط TERMINAL EMULATOR استفاده می کنید این دستور به کار برده می شود. توسط این دستور می توانید 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 نرم افزاری با شماره کانال Channel میکرو ارسال می کند. مقدار باینری خوانده شده از پورت سریال نرم افزاری در متغیر var1 و متغیر اختیاری var2 جای می گیرد که اگر به عنوان BYTE تعریف شده باشند یک بایت، به ازاء INTEGER دو بایت و اگر یک آرایه تعریف شوند به تعداد آرایه ها کاراکتر دریافت می کنند. این دستور برای گرفتن تعداد بایتها در همان خط می ایستد.

• 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 می دهد که در هنگام اجرای برنامه از صفحه کلید ورودی دریافت کنیم.

```
INPUTHEX #Channel, var [, varn]
```

زمانی که از محیط TERMINAL EMULATOR استفاده می کنید این دستور به کار برده می شود. توسط این دستور می توانید 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 می تواند بین 0-9 و A-F باشد در متغیر var و متغیر اختیاری varn قرار می دهد و داده از طریق پورت سریال به UART نرم افزاری با شماره کانال Channel میکرو ارسال می کند.

این دستور NOECHO است و در صورت وارد کردن داده توسط صفحه کلید کامپیوتر داده ورودی در محیط TERMINAL EMULATOR نمایش داده نمی شود. تعداد بایت ورودی بستگی به تعریف متغیرهای var و varn دارد که به صورت زیر تعیین می شوند:

- اگر var یا varn بایت تعریف شده باشند ورودی هر یک نهایتاً 2 کاراکتر می تواند طول داشته باشد.
- اگر var یا varn INTEGER یا WORD تعریف شده باشند، ورودی هر یک نهایتاً 4 کاراکتر می تواند طول داشته باشد.
- اگر var یا varn LONG تعریف شده باشند، ورودی هر یک نهایتاً 8 کاراکتر می تواند 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

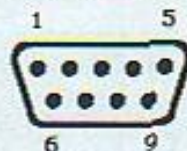
```

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.

DB9 CONNECTOR



- 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 می‌شود. 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 خازن 22µF است.

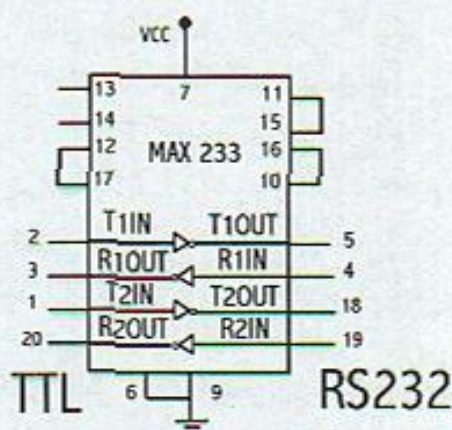
تراشه MAX233

تراشه MAX233 هم مانند MAX232 عمل می‌نماید با این تفاوت که دیگر نیازی به خازن‌های خارجی نیست.

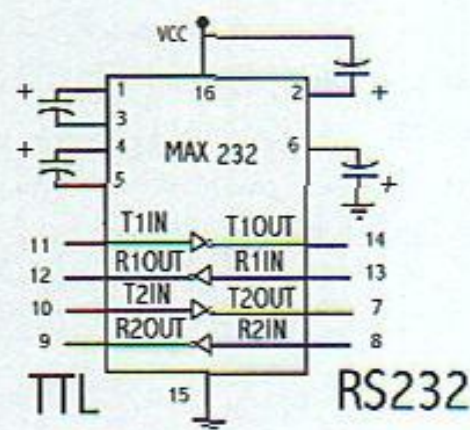
مبدل ترانزیستوری منطق TTL و RS-232 به یکدیگر

از مدارهای ساده صفحه بعد می‌توانید برای فاصله‌های نه‌چندان طولانی به جای تراشه‌های MAX استفاده نمایید. از ترانزیستور BC107 می‌توانید به جای BC547 استفاده نمایید.

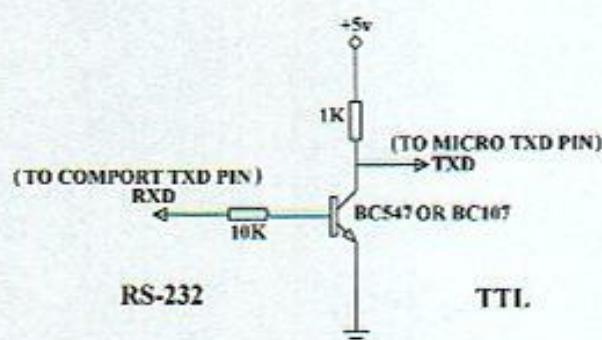
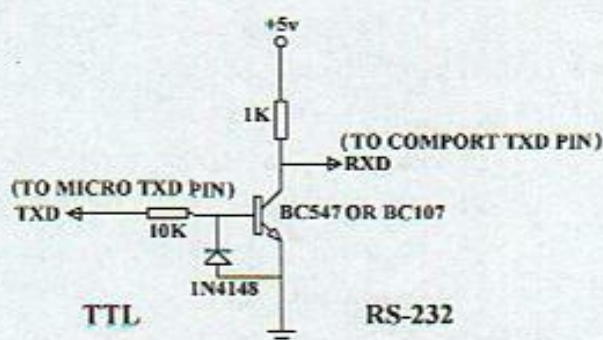
۲۱۹ پیکره‌بندی و کار با امکانات AVR در BASCOM



شکل پایه های تراشه MAX233



شکل پایه های تراشه MAX232



مبدل ترانزیستوری منطق RS-232 و TTL به یکدیگر

BC547 , BC557



نمای گرافیکی

ترانزیستور BC547

۷-۶ 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)
۴. مبدل ADC دو شیبی (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 و MSB 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 $\pm 1.5\text{LSB}$
- زمان تبدیل (CONVERSION TIME) $65 - 260 \mu\text{s}$
- وضوح 15 KSPS در بالاترین حد
- کانالهای مولتی پلکس شده
- مدهای تبدیل SINGLE و FREE
- ولتاژ ورودی از 0V 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 از $\pm 0.3\text{V}$ 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 پایه‌های AVCC - AGND باشد. در غیر اینصورت 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.

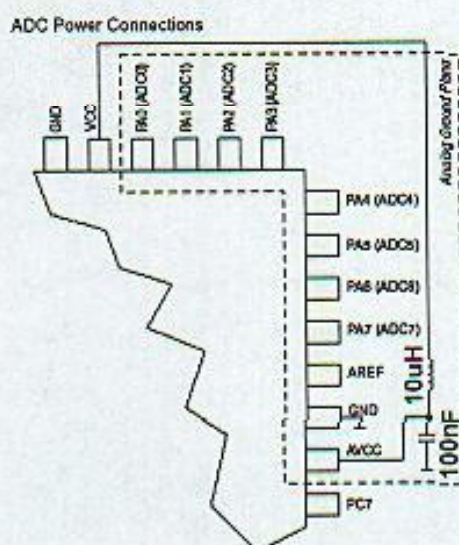
به طور مثال اگر پایه به ولتاژ $\text{AREF} = 3.5\text{V}$ و $\text{AGND} = 0\text{V}$ وصل شده باشد، مقدار دیجیتال شده 1023 نشان دهنده ولتاژ 3.5V و مقدار 0 نشان دهنده ولتاژ 0.0V بر روی پایه مبدل ADC انتخاب شده است.

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 پایین‌گذر نشان داده شده در شکل زیر وصل نمایید.
۴. از مُدهای SLEEP و حالت ADC NOISE CANCELER برای کاهش نویز القاء شده توسط CPU استفاده کنید.



طرز اتصال پایه AVCC به پایه VCC با فیلتر پایین‌گذر برای کاهش نویز

پیکره‌بندی ADC در محیط BASCOM

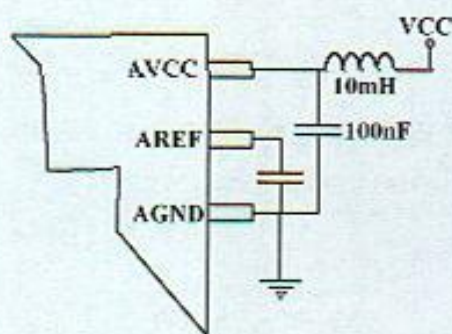
مبدل آنالوگ به دیجیتال باید برای میکروهایی که ADC دارند، توسط این دستور پیکره‌بندی شود تا بتوان از آن استفاده نمود. مبدل داخلی میکرو دارای وضوح 10 بیتی است و برای تبدیل با این وضوح، نیاز به کلاکی بین 50KHZ تا 200KHZ دارند و این کلاک را از تقسیم کریستال تامین می‌کند.

CONFIG ADC = SINGLE|FREE, PRESCALER = AUTO, REFERENCE = OPTIONAL

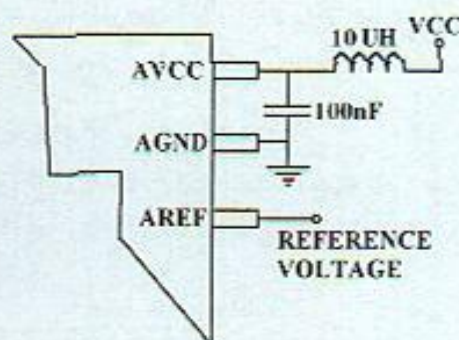
CONFIG ADC = SINGLE | FREE : برای تبدیل سیگنال آنالوگ خود به دیجیتال می‌توانید از دو مد SINGLE و FREE استفاده نمایید. زمانی که مد SINGLE را انتخاب می‌کنید، باید از دستور GETADC() استفاده کنید.

PRESCALER = AUTO : 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، 15، 32، 64 یا 128 می‌باشند.

REFERENCE = OPTIONAL : گزینه‌ای اختیاری برای ولتاژ مرجع (VOLTAGE REFERENCE) در بعضی از میکروها از جمله MEGA8 است. OPTIONAL می‌تواند گزینه‌های زیر باشد :
OFF : برای خاموش کردن ولتاژ مرجع داخلی (INTERNAL REFERENCE TURNED OFF) و استفاده از ولتاژ موجود بر روی پایه AREF بعنوان ولتاژ مرجع انتخاب می‌شود. در این حالت اتصال پایه‌های AVCC و AREF به صورت شکل ۱-۶ است.



شکل ۲-۶



شکل ۱-۶

AVCC : زمانی که ولتاژ پایه AVCC به عنوان ولتاژ مرجع در نظر گرفته می‌شود. در این حالت اتصال پایه‌های AVCC و AREF به صورت شکل ۲-۶ است.
INTERNAL : زمانی که ولتاژ مرجع داخلی 2.56V با خازن خارجی بر روی پایه AREF استفاده شود. در این حالت اتصال پایه‌های AVCC و AREF به صورت شکل ۲-۶ 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 (0-7) 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 بیکره‌بندی می‌شود دیگر نمی‌توانید از آن به عنوان I/O استفاده کنید.

دستورات STOP و START

توسط دستور 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 LABEL ON ADC پرش کرد که LABEL 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 یا بالا رفتن پرچم وقفه اتمام تبدیل ADC از این مُد بیدار شده و مقدار دیجیتال شده را در متغیر نوع WORD قرار می‌دهد.

• مثال

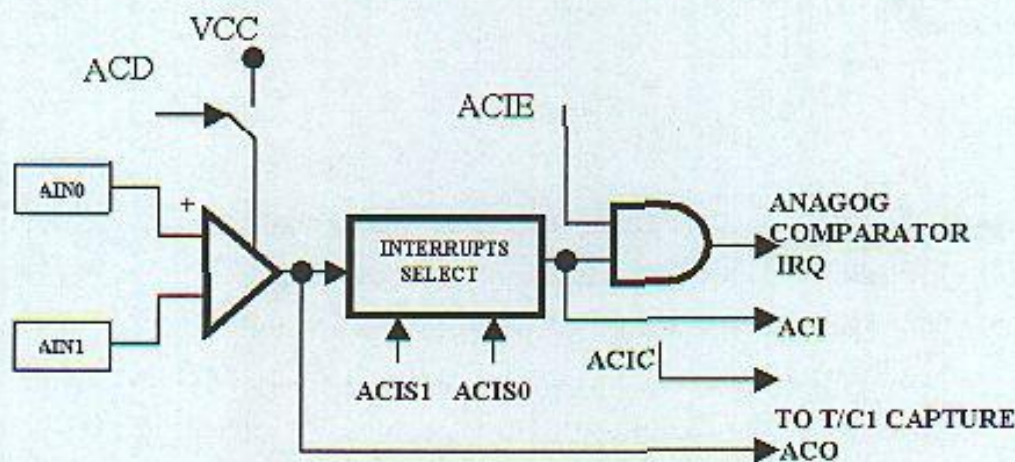
```
$regfile = '4433def.dat'
$crystal = 4000000
$baud = 9600
'configure single mode and auto prescaler setting
'The AUTO feature, will select the highest clockrate possible
'the ADC needs a clock from 50-200 KHz
Config Adc = Single , Prescaler = Auto
On Adc Adc_isr
Enable Adc
Enable Interrupts
Dim W 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
```


۸-۶ مقایسه کننده آنالوگ

مقایسه کننده آنالوگ و رجیسترها

مقایسه کننده آنالوگ (ANALOG COMPARATOR) مقادیر آنالوگ در دو ورودی پایه مثبت (AIN0) و پایه منفی (AIN1) را با هم مقایسه می کند. زمانی که ولتاژ موجود در ورودی مثبت (AIN0) بالاتر از ولتاژ موجود در ورودی منفی (AIN1) باشد، خروجی مقایسه کننده (ACO) یک می شود. مقایسه کننده همچنین دارای یک پرچم وقفه مجزا است. کاربر می تواند نحوه تریگر شدن وقفه خروجی مقایسه کننده را در لبه بالارونده، پایین رونده یا TOGGLE انتخاب کند. خروجی مقایسه کننده می تواند به عنوان تریگر ورودی CAPTURE تایمر/کانتریک نیز استفاده شود.



بلوک دیاگرام مقایسه کننده آنالوگ

پیکره بندی مقایسه کننده آنالوگ در BASCOM

مقایسه کننده آنالوگ باید توسط دستور زیر پیکره بندی شود تا بتوان از آن استفاده نمود. مقایسه کننده آنالوگ، ولتاژ موجود در دو پایه مثبت (AIN0) و پایه منفی (AIN1) را با هم مقایسه می کند.

CONFIG ACI = ON|OFF, COMPARE = ON|OFF, TRIGGER = TOGGLE|RISING|FALLING

CONFIG ACI = ON|OFF : باید در زمان استفاده از مقایسه کننده ON باشد.

COMPARE = ON|OFF : در صورت انتخاب ON، خروجی مقایسه کننده آنالوگ (ACO) مستقیماً به

ورودی CAPTURE تایمر/کانتریک وصل است و در صورت انتخاب OFF این اتصال برقرار نیست.

TRIGGER = TOGGLE|RISING|FALLING : این گزینه نحوه روی دادن وقفه مقایسه کننده

را نشان می دهد.

در صورت انتخاب FALLING، یک لبه پایین رونده در خروجی مقایسه کننده باعث یک شدن

پرچم وقفه مقایسه کننده و اجرا شدن برنامه وقفه خواهد شد.

در صورت انتخاب RISING، یک بالارونده در خروجی مقایسه کننده باعث یک شدن پرچم وقفه

مقایسه کننده 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 فعال‌کننده وقفه سراسری ENABLE INTERRUPTS فعال می‌شود. توسط دستور ON ACI LABEL می‌توان در زمان رویداد وقفه مقایسه‌کننده به LABEL 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

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 128*64 باشد. برای LCD های نوع SED به طور مثال از 128*64 SED استفاده نمایید.

DATAPORT: مشخص‌کننده پورتهی 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.

CONTROLPORT: مشخص‌کننده پورتهی 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 $\text{CONTROLPORT} = \text{PORTC}$ باشد، $\text{CE} = 0$ 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 $\text{CONTROLPORT} = \text{PORTC}$ باشد، $\text{CD} = 1$ 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 $\text{CONTROLPORT} = \text{PORTC}$ باشد، $\text{WR} = 2$ 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 $\text{CONTROLPORT} = \text{PORTC}$ باشد، $\text{RD} = 3$ 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 $\text{CONTROLPORT} = \text{PORTC}$ باشد، $\text{FS} = 4$ 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 $\text{CONTROLPORT} = \text{PORTC}$ باشد، $\text{RESET} = 5$ به معنای اتصال PORTC.5 به پایه RESET از LCD می‌باشد.

MODE : مشخص کننده تعداد ستون متنی LCD است که می‌تواند 8 یا 6 باشد. زمانی که از عدد 6 استفاده می‌نمایید نهایتاً $X\text{-PIXEL} / 6$ ستون متنی خواهید داشت و زمانی که از عدد 8 استفاده نمایید نهایتاً $X\text{-PIXEL} / 8$ ستون متنی خواهید داشت. این گزینه همچنین مشخص کننده نوع فونت است. به طور مثال اگر تعداد $(X\text{-PIXEL}) = 24$ باشد با $\text{MODE} = 8$ ، تعداد ستونها برابر 30 و در صورت استفاده از $\text{MODE} = 6$ ، تعداد ستونها برابر 40 است.

طبق مثال صفحه قبل طرز اتصال پایه‌های LCD به میکرو در جدول زیر آمده است.

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/E	PORTC.3
9	NC	NC
10	RESET	PORTC.4
11 - 18	D0 - D7	PORTA0-PORTA.7
19	FS	PORTC.5
20	NC	NC

جدول پایه های GRAPHICAL LCD

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 "1234567890123456789012345678901234567890"
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 0 - 239 و Y از 0 - 127 می‌تواند تغییر کند.

• مثال

```
FOR X = 0 TO 140
  PSET X , 20 , 255 ' SET THE PIXEL
NEXT
FOR X = 0 TO 140
  PSET X , 127 , 255 ' 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

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.

• 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. COLOR = 0 دایره را پاک کرده و به ازای COLOR = 255 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, LABEL

برای نمایش عکسی که در منوی TOOLS و قسمت GRAPHIC CONVERTER ذخیره کرده‌اید استفاده می‌شود. X مکان قرارگیری افقی و Y مکان قرارگیری عمودی عکس را نشان می‌دهد. LABEL نام برجسبی است که اطلاعات عکس مورد نظر در آن قرار دارد.

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
This label holds the image data
Plaatje:
'$BGF will put the bitmap into the program at this location
$bgf "mcs.bgf"
```

'end program

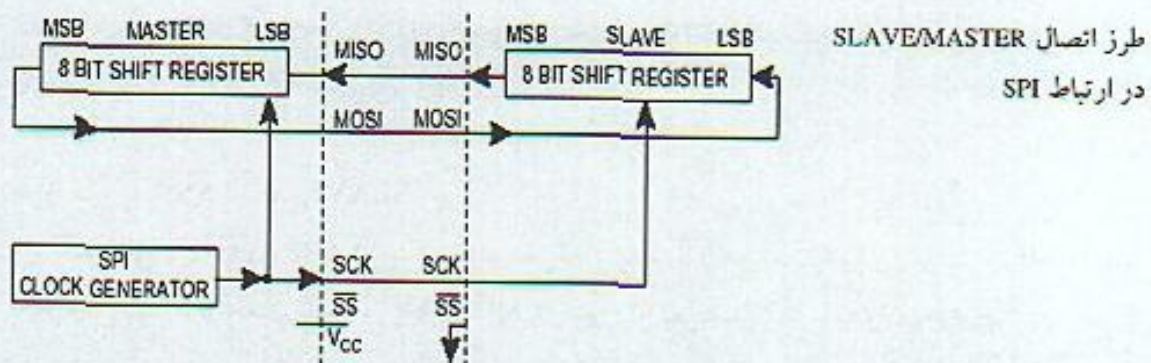
۱۰-۶ ارتباط سریال SPI

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 بخشی 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 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 دادها 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 زیر 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، داده‌ای MASTER و SLAVE با هم عوض شود.



طرز اتصال MASTER/ SLAVE

زمانیکه SPI فعال شده باشد جهت پایه‌های \overline{SS} ، SCK، MISO، MOSI با توجه به جدول زیر 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.

PIN	DIRECTION, MASTER SPI	DIRECTION, SLAVE SPI
MOSI	USER DEFINED	INPUT
MISO	INPUT	USER DEFINED
SCK	USER DEFINED	INPUT
SS	USER DEFINED	INPUT

جدول نحوه اتصال پایه‌های ارتباط SPI (SPI PIN OVERRIDES)

طرز کار پایه \overline{SS} در مُد MASTER

زمانیکه SPI بعنوان 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 بعنوان پایه خروجی عادی استفاده می‌شود بدینصورت که هیچ تأثیری در ارتباط SPI ندارد و می‌تواند بعنوان انتخاب SLAVE و یا بعنوان 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 زمانیکه بعنوان ورودی تعریف شده است بوسیله مدار جانبی دیگر پایین (صفر) شود در حالتیکه SPI در مُد MASTER پیکره‌بندی شده است سیستم آنرا بعنوان 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. هنگامیکه بیت MSTR بعلت انتخاب MASTER دیگر صفر شده باشد کاربر باید آنرا یک کند تا مُد MASTER را دوباره فعال کند.

طرز کار پایه \overline{SS} در مُد SLAVE

زمانیکه SPI در حالت SLAVE پیکره‌بندی می‌شود پایه \overline{SS} همیشه ورودی است. هنگامیکه پایه \overline{SS} پایین می‌شود SPI فعال شده و پایه MISO در صورت تعریف کاربر، بعنوان خروجی در نظر گرفته می‌شود. پایه‌های دیگر همه ورودی هستند.

زمانیکه پایه \overline{SS} بالا می‌شود تمام پایه‌ها ورودی هستند و SPI بیکار است. بدین معنی که هیچگونه داده‌ای را دریافت و یا ارسال نمی‌کند. اگر پایه \overline{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 : فعال کننده SPI

زمانیکه 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.

بیت 3-CPOL : CLOCK POLARITY

در حالت بیکاری SPI، اگر این بیت یک باشد پایه SCK بالا خواهد بود در غیر اینصورت پایه SCK پایین خواهد بود.

بیت 2-CPHA : CLOCK PHASE

مُد های اطلاعاتی (DATA MODE)

چهار نوع ترکیب PHASE و POLARITY با توجه به داده سریال طبق جدول زیر موجود می‌باشد که توسط بیت‌های کنترلی CPHA و CPOL مشخص می‌شوند. مُدهای انتقال داده SPI در شکلها و جدول‌های زیر مشخص شده‌اند. CPOL (CLOCK POLARITY) نقشی در نحوه ارسال داده ندارد فقط وضعیت پایه SCK را در حالت بیکاری SPI مشخص می‌کند.

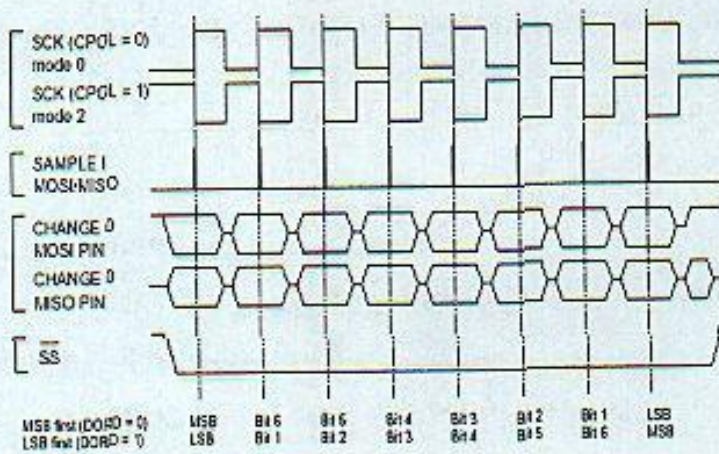
MASTER و SLAVE باید در یک مُد پیکره‌بندی شوند.

نکته

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

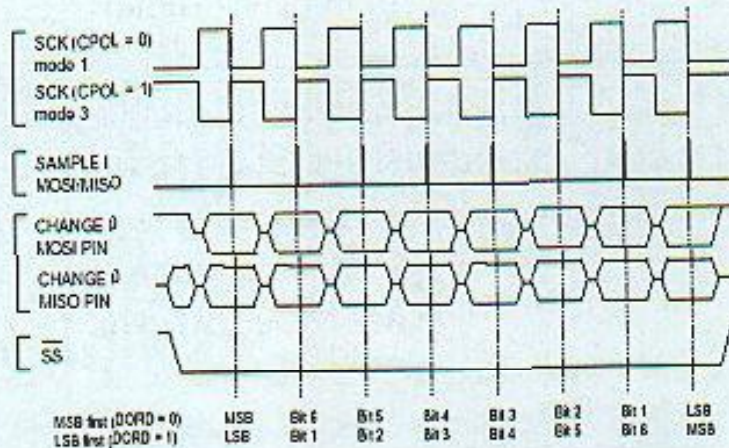
جدول انتخاب مُدهای ارتباطی اطلاعاتی SPI

دو نمودارهای زمانی زیر ارتباط MASTER/SLAVE را در مدهای اطلاعاتی مختلف نشان می‌دهد.



SPI TRANSFER MODE
SELECTION WITH CPHA = 0

نمودارهای زمانی مدهای 0 و 2



SPI TRANSFER MODE
SELECTION WITH CPHA = 1

نمودارهای زمانی مدهای 1 و 3

بیت 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 f_{osc} در جدول زیر آمده است.

SPI2X	SPR1	SPR0	SCK FREQUENCY
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

جدول انتخاب فرکانس کلاک SPI با توجه به فرکانس OSC

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

بیت 7-SPIF : پرچم وقفه

زمانیکه ارسال داده تکمیل شد بیت SPIF یک می‌شود در صورتی که بیت SPIE در رجیستر SPCR و وقفه سراسری (1) فعال شده باشند.

بیت 6-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

بیت 0-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 $F_{OSC}/4$ 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 \overline{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 و سپس 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 1/4 , 1/16 , 1/64 , 1/128

فرکانس سیستم باشد.

NOSS=0/1 : زمانی که در حالت MASTER نمی‌خواهید سیگنال \overline{SS} ایجاد شود، 1 را انتخاب کنید.

در این حالت کاربر بایستی نرم‌افزاری پایه SLAVE موردنظر را پایین کند.

پیکره‌بندی سخت‌افزاری را می‌توان نیز بصورت دستور ساده زیر نوشت.

CONFIG SPI = HARD

POLARITY= HIGH که در این حالت بصورت پیش فرض اول MSB فرستاده می‌شود و

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 در نمی‌خواهید سیگنال

\overline{SS} داشته باشید از گزینه 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 متغیر 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 = P1.2 , 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)

متغیر یا ثابت byte به باس SPI ارسال شده و همزمان داده دریافت شده از باس SPI در متغیر var جای می‌گیرد.

• مثال

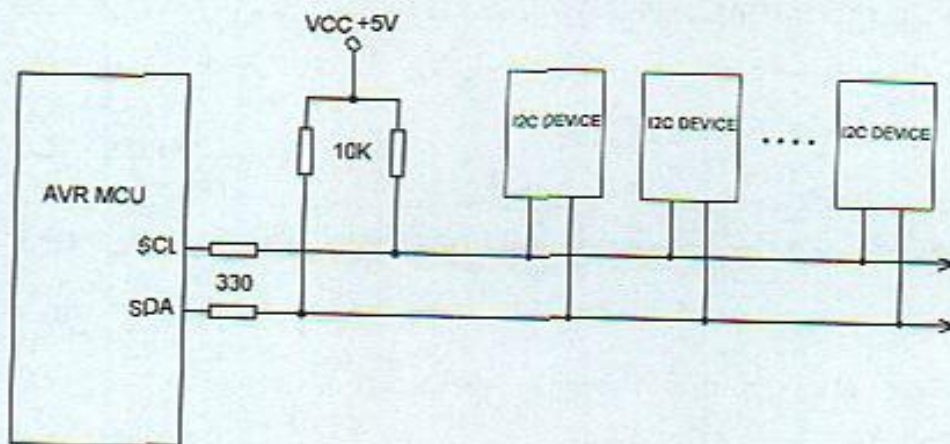
```

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

```

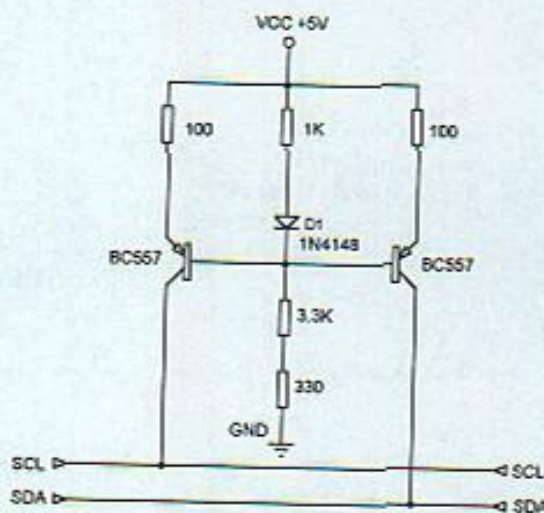

۱۱-۶ پیکره‌بندی ارتباط سریال I2C (2-WIRE)

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 زیر طرز اتصال وسایل I2C در یک باس را نشان می‌دهد.



شکل اتصال
وسایل I2C در
یک باس

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 زیر در انتهای خطوط I2C قرار می‌گیرد.



شکل مدار ارتباطی وسایل I2C
برای فواصل دور

تعیین کلاک I2C

کلاک ارتباط I2C بستگی به کلاک سیستم دارد. برای رفع این مشکل از پیکره‌بندی زیر برای تعیین فرکانس کاری ارتباط I2C استفاده می‌نماییم.

۲۳۷ پیکره‌بندی و کار با امکانات AVR در BASCOM

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 100KZ 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      ' PORTB.7 is the SDA line
```

تعیین پایه SCL

CONFIG SCL = pin

برای ارتباط میکرو با وسایل جانبی دیگر از طریق پروتکل I2C (2-wire)، با این دستور پایه SCL (CLOCK) پیکره‌بندی می‌شود.

• مثال

```
CONFIG SCL = PORTC.6      ' PORTB.7 is the SDA 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 متغیر نوع BYTE، INTEGER، WORD یا عدد ثابت که حاوی آدرس Slave است.

VAR: یک متغیر 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
I2creceive Slave, X
Print X
```

```
'Slave Address
'Get The Value
'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 متغیر نوع **WORD**، **INTEGER**، **BYTE**، که حاوی آدرس Slave است.

VAR: داده‌ای که قصد ارسال به خط یا باس I2C را داریم که می‌تواند یک **INTEGER**، **BYTE**، **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 اسلاتور جداگانه داخلی کلاک دریافت می‌کند. با کنترل کلاک WATCHDOG و منبع تغذیه میکرو بعد از سپری شدن زمان مشخص شده سیستم ریست خواهد شد. ۸ 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

۲۳۹ پیکره‌بندی و کار با امکانات AVR در BASCOM

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.

با دستور **CONFIG WATCHDOG**، **WATCHDOG** شروع به کار نمی‌کند 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.

نکته

• مثال

```
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 000 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.

برای فعال کردن هر یک از وقفه‌ها علاوه بر اینکه وقفه مربوطه بایستی توسط دستورات فوق فعال شده باشد، وقفه سراسری نیز باید توسط دستور **ENABLE INTERRUPTS** فعال شده باشد. این دستور اجازه استفاده از همه وقفه‌ها را می‌دهد.

نکته

دستور ON INTERRUPT

زمانی که وقفه اتفاق بیافتد سیستم اجرای برنامه را متوقف کرده و به وقفه پاسخ می‌دهد و به برجسی که برای آن وقفه تعریف شده پرش می‌کند و بعد از برگشت اجرای برنامه ادامه پیدا می‌کند.

ON interrupt label [NOSAVE]

در دستور فوق label نام برجسی 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.

امکان اینکه شما برای وقفه‌ها اولویت تعیین کنید نیست، و هر وقفه‌ای که در آدرس پایین‌تر حافظه نوشته شده باشد دارای اولویت بالاتر است.

نکته

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 لبه پایین 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.

جدول انواع وقفه میکروهای AVR و نام آنها در محیط BASCOM

EXTERNAL INTERRUPTS		INTERRUPT NAME
وقفه خارجی صفر	External Interrupt 0	INT0
وقفه خارجی یک	External Interrupt 1	INT1
وقفه‌های خارجی 2 تا 7 برای بعضی از میکروها	External Interrupt for some chips	INT2 - INT7

TIMER/COUNTER0 INTERRUPTS		
TIMER0,COUNTER0	Enable TIMER/COUNTER0 interrupts	فعال / غیرفعال کننده تمام وقفه‌های موجود برای T/C0
OVF0	TIMER0 overflow interrupt	وقفه سرریزی 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 سرریزی TIMER / COUNTER1
CAPTURE1 , IC1	INPUT CAPTURE TIMER1 interrupt	وقفه ورودی CAPTURE برای T/C1
COMPARE1A,OC1A	OUTPUT COMPARE A interrupt	وقفه تطابق مقایسه مد A برای T/C1
COMPARE1B,OC1B	OUTPUT interrupt COMPARE B	وقفه تطابق مقایسه مد 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 سرریزی 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
On Int0 Label2 Nosave
Do
Loop
End

Label2 :
Dim A As Byte
If A > 1 Then
Return
End If
Return
Return

```

'enable the interrupt
'jump to label2 on INT0
'endless loop

'generates a RET because it is inside a condition
'generates a RETI because it is the first
'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 متغیر VAR در آدرس ADDRESS حافظه EEPROM داخلی نوشته می‌شود. بعد از دستور WRITEEEPROM با توجه به VCC بایستی 4ms - 2.5 تاخیر ایجاد کنید تا عملیات نوشتن تکمیل شود. آدرس می‌تواند یک عدد ثابت یا یک متغیر بسته به حافظه از نوع داده WORD یا BYTE باشد.

- شما همچنین می‌توانید متغیر برای EEPROM تعریف کنید.

Dim var As Eram var type

که var type می‌تواند داده‌های نوع single, long, integer, word, byte و string باشد.

- شما همچنین می‌توانید متغیرهای آرایه‌ای برای EEPROM استفاده کنید.

Dim ar(10) as Eram Byte

در این حالت 10 بایت اول از حافظه EEPROM برای متغیر 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 متغیر Eb در آدرس 13 از حافظه EEPROM قرار می‌گیرد.

در صورت مشخص نکردن آدرس حافظه، داده‌ها به ترتیب نوشتن در برنامه، از آدرس 0 شروع به جای گرفتن در حافظه می‌کنند.

نکته**• 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)

LSB داده در حافظه پایین‌تر EEPROM قرار می‌گیرد.

نکته

```
$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 متغیر VAR از نوع داده BYTE ذخیره می‌شود. آدرس می‌تواند یک عدد ثابت یا یک متغیر بسته به حافظه از نوع داده WORD یا BYTE باشد.

• مثال

```
Dim B As Byte
Writeeprom B, 0           'store at first position
Waitms 10
Readeprom B, 0           'read byte back
```

۱۵-۶ مدهای SLEEP

معرفی انواع مدهای SLEEP

مدهای SLEEP برای متوقف کردن (SHUT DOWN) قسمت‌ها و امکانات استفاده نشده میکرو و همچنین کاهش و صرفه‌جویی در توان مصرفی به کاربرده می‌شوند. AVRها مدهای مختلفی از SLEEP را برای استفاده کاربر مهیا ساخته‌اند. بیشترین تعداد مدهای SLEEP، ۶ حالت می‌باشد. بعلاوه وجود تمام این حالات در میکرو نمونه ATMEGA32 در این بخش قصد داریم به بررسی انواع مدها 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، WATCHDOG، TIMER/COUNTERS، ارتباط سریال TWO-WIRE و وقفه‌های سیستم اجازه می‌دهد که کار کنند (اگر فعال شده باشند). به طور مثال اگر تریگر به پایه ایتراپت 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، WATCHDOG، TIMER/COUNTER2، ارتباط سریال TWO-WIRE و وقفه‌های سیستم اجازه می‌دهد که کار کنند (اگر فعال شده باشند). به طور مثال اگر تریگر به پایه ایتراپت 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 می‌شود. 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 سطح INT0، 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 و INT1 و وقفه خارجی INT2 می‌توانند میکرو را از مُد POWER-DOWN بیدار کنند. تاخیری 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
```


۲۴۰ BASCOM در AVR با امکانات

```

$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.

۱۶-۶ 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 میشود. اندازه حافظه BOOTLOADER در این میکروها با دو فیز بیت BOOTSZ_{1,2} قابل انتخاب است.

نکته

در قسمت حافظه BOOT فقط قادر به نوشتن برنامه به زبان ASSEMBLY هستید.

وارد شدن به برنامه BOOTLOADER

از فیز 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 فیز بیت های BOOTSZ تعیین کرده اند تغییر می یابد.

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

جدول انتخاب آدرس بردار ری ست توسط فیز بیت BOOTRST

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 میتوان با دستور JMP \$BOOTADDRESS در هر جای برنامه APPLICATION به برنامه BOOTLOADER پرش کرد. در صورتی که فیز بیت BOOTRST برنامه ریزی شود برنامه BOOTLOADER پس از ری ست اجرا میشود. لازم به ذکر است که توسط دستور JMP \$0000 در داخل برنامه BOOTLOADER میتوان به آدرس \$0000 حافظه FLASH پرش کرد.

• مثال

```

Sregfile = "M8DEF.DAT"
S crystal = 8000000
S baud = 9600
Dim A As Byte
Do
  For A = 0 To 10
    Print A
  Next
  JMP $F80
Loop
End                                     'end program

```


۲۴۷ پیکره‌بندی و کار با امکانات AVR در BASCOM

```
$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 میشود و در صورتی که پایه بالا باشد دوباره حلقه $MAIN$ اجرا میشود 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 میکند.

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 میکند و پایه $PINB.1$ امتحان میشود و در صورتی که پایه بالا باشد دوباره حلقه $MAIN$ اجرا میشود در غیر اینصورت یعنی زمانی که پایه زمین باشد اجرای برنامه از آدرس ابتدای حافظه $FLASH$ ($\$0000$) دوباره آغاز خواهد شد.

آدرس شروع حافظه $BOOT$ طبق جدول زیر و با توجه به نحوه برنامه ریزی فیوزهای $BOOTSZ1,2$ در نظر گرفته میشود.

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

جدول انتخاب مقدار حافظه $BOOT$ توسط فیوز بیت‌های $BOOTSZ0,1$ برای $ATMEGA8$

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 اختیاری SUB را برای زمانی که می‌خواهید به جای پرش به برجسب lable به یک زیربرنامه با نام برجسب 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 (پیش فرض) تاخیر ایجاد می‌شود و دوباره پایه خوانده می‌شود و اگر هنوز وضعیت پایه تغییر نکرده باشد آن را به منزله فشردن کلید در نظر می‌گیرد و به lable پرش می‌کند. زمانی که دستور DEBOUNCE بخواند دوباره اجرا شود بایستی وضعیت کلید به حالت قبل تغییر کرده باشد.

دستور DEBOUNCE منتظر می‌ماند که حالت state اتفاق بیفتد.

نکته

این پیکره‌بندی تاخیری که در هنگام استفاده از دستور DEBOUNCE ایجاد می‌شود را مشخص می‌کند. CONFIG DEBOUNCE = VAR
VAR مدت زمانی تاخیر بر حسب میلی ثانیه است. زمانی که DEBOUNCE پیکره‌بندی نمی‌شود مقدار ثابت 25ms به صورت پیش فرض 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 متغیر یا ثابتی که نشان دهند شماره یکی از پایه‌های پورت است به طور مثال PORT=PORTA و PIN=0 به معنای PORTA.0 است. PERIOD متغیر 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

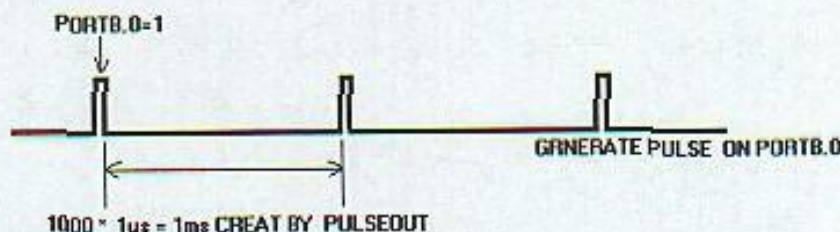
پایه استفاده شده بایستی خروجی تعریف شده باشد.

• مثال

در صورتی که دستور Pulseout در حلقه قرار نگیرد فقط یک پالس ایجاد می‌شود. شکل پالس خروجی پایه portb.0 در صفحه بعد آمده است.

```
$crystal = 4000000
Config Portb = Output
Portb = 1
Do
    Pulseout Portb , 0 , 1000
Loop
End

'PORTB all output pins
'PORTB.0=1
'generate a pulse
'loop for ever
```



شکل 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 بالعکس آشکار کرد.

PULSEIN var , PINX , PIN , STATE

Var متغیری از نوع داده WORD است که مدت زمان مذکور را در خوی جای می‌دهد. PINX و PIN نیز مشخص کننده پایه موردنظر برای امتحان کردن هستند. به طور مثال PINX=PIN 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 متغیر خطا با نام ERR یک می‌شود. شما می‌توانید با تست کردن این متغیر از ایجاد خطا در اندازه‌گیری زمان استفاده کنید.

• مثال

```
Dim W As Long
Baud = 9600
```



```

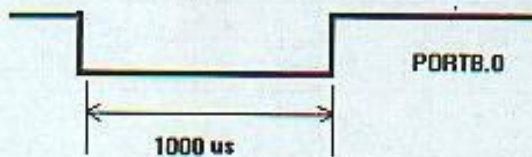
Config PortB = Input
Pulsein W, PinB, 0, 0
Print W
Loop
End

```

```

'detect time from 0 to 1
100 WILL PRINT 1000us = 100*10us

```



شکل 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 متغیری است که مشخص کننده تعداد پالس ارسالی است و ثابت یا متغیر pulses نمایانگر مدت زمان بالا و پایین بودن پایه موردنظر است. از این دستور بیشتر برای ایجاد صدا زمانهای که بلندگو یا BUZZER به یکی از پایه متصل است استفاده می‌شود.

این دستور برای ایجاد فرکانس‌های دقیق توصیه نمی‌شود. برای ایجاد فرکانس‌های دقیق از تایمرها استفاده نمایید.

pulses و duration بایستی بین اعداد 1 تا 65535 باشند.

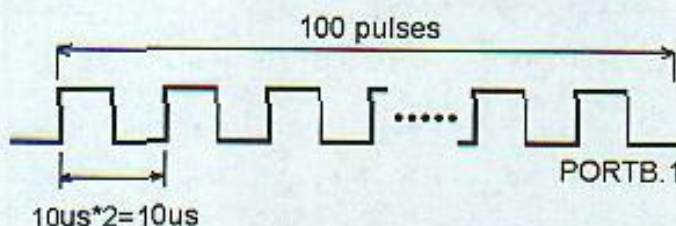
نکته

• مثال

```

$CRYSTAL = 8000000
CONFIG PORTB=OUTPUT
SOUND PORTB.1, 100, 10 'LIKE BEEP SOUND
End

```



شکل پالس ایجاد شده توسط دستور SOUND



حافظه‌های EEPROM

سریال 2-WIRE

۱-۷ 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 می‌گردد. 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 با قابلیت ارتباط بصورت (2- WIRE) I2C را از سری AT24 به بازار عرضه کرده است. انواع آنها در جدول زیر آمده است.

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

جدول انواع حافظه‌های سریال 2-WIRE (ادامه جدول صفحه قبل)

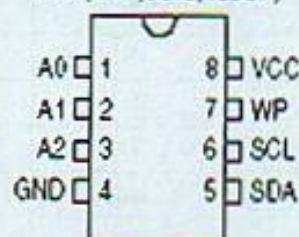
معرفی نوع های AT2402A/04A/08A

در این بخش قصد داریم به طور نمونه انواع حافظه‌های AT24C02A/04A/08A را انتخاب کرده و در مورد ترکیب پایه‌ها، 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
 $(VCC = 4.5V \text{ to } 5.5V)$ 5.0
 $(VCC = 2.7V \text{ to } 5.5V)$ 2.7
 $(VCC = 2.5V \text{ to } 5.5V)$ 2.5
 $(VCC = 1.8V \text{ to } 5.5V)$ 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 2.5 - 1.7
- دارای پایه حفاظت از نوشتن (WRITE PROTECT) WP برای استفاده نرم‌افزاری یا سخت‌افزاری
- مد 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 بسته‌بندی

AT24C02A/04A/08A
8PIN(PDIP,SOIC,TSSOP)



ترکیب پایه های حافظه‌های

AT24C02A/04A/08A

Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect

جدول عملکرد پایه های حافظه سریال

AT24C02A/04A/08A دارای مقدار 8192/4096/2048 بیت حافظه EEPROM به صورت 1024 / 512 / 256 کلمه 8 بیتی هستند. این وسایل برای کار در محیط های تجاری و صنعتی و در مکان هایی که استفاده از ولتاژ و توان های کم احتیاج باشد بهینه سازی شده اند. محدوده های کاری حافظه های در جدول زیر آمده اند.

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 پایه ها

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 EEPROM ها استفاده میشود.

پایه داده سریال (SDA)

SDA یک پایه دو طرفه برای ارتباط داده ها است.

پایه های A0-A1-A2 آدرس سخت افزاری EEPROM

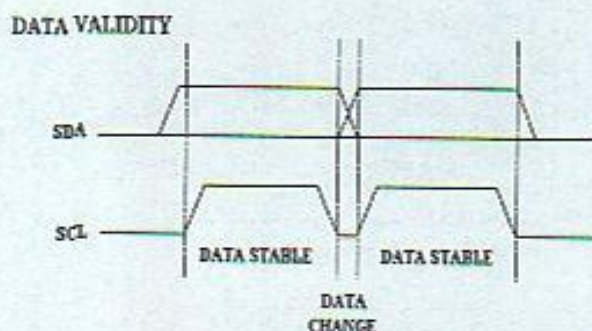
پایه های ورودی A0-A1-A2 مشخص کننده آدرس EEPROM است. در صورتی که به صورت سخت افزاری بخواهیم وسیله را آدرس دهی بکنیم بسته به نوع حافظه تا هشت EEPROM می توانند در یک BUS قرار گیرند.

پایه محافظت از نوشتن (WP)

این پایه ورودی زمانی که زمین یا بدون اتصال باشد نوشتن به صورت عادی انجام می گیرد. اگر این پایه یک شود از نوشتن در EEPROM جلوگیری میشود.

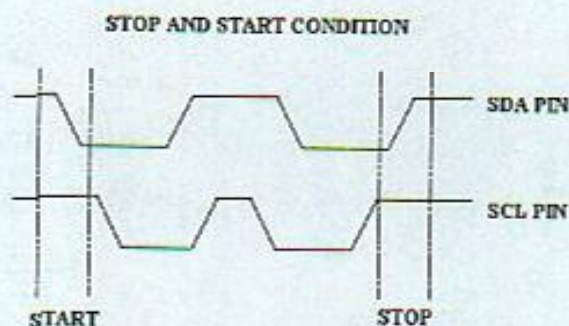
۳-۷ طرز کار حافظه

انتقال کلاک و داده : پایه SDA به طور معمول توسط وسیله خارجی بالا نگه داشته میشود. داده بر روی SDA زمانی که SCL پایین است تغییر می یابد (با توجه به شکل ۱-۷). تغییر SDA در زمان بالا بودن SCL به منزله ایجاد یک START CONDITION یا STOP CONDITION است.



شکل ۱-۷

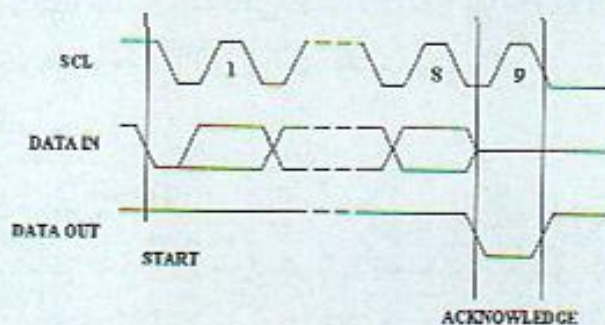
ایجاد START CONDITION : یک به صفر (HIGH TO LOW) کردن SDA در زمان بالا بودن SCL به منزله ایجاد یک START CONDITION است (با توجه به شکل ۲-۷).
ایجاد STOP CONDITION : صفر به یک (LOW TO HIGH) کردن SDA در زمان بالا بودن SCL به منزله ایجاد یک STOP CONDITION است (با توجه به شکل ۲-۷).



شکل ۲-۷

دیاگرام زمانی ایجاد
شرایط START و STOP

شناسایی (ACKNOWLEDGE) : تمام آدرس ها و داده ها در قالب WORD به صورت هشت بیتی از EEPROM به خارج یا از خارج به EEPROM انتقال می یابد. EEPROM با فرستادن صفر در کلاک نهم مشخص میکند که هر یک از کلمات (WORD) را طبق دیاگرام زمانی شکل ۳-۷ دریافت کرده است.



شکل ۳-۷

دیاگرام زمانی ارسال ACK
توسط EEPROM

RESET شدن حافظه: برای ریست شدن حافظه مراحل زیر بایستی طی شود:

۱. بالا بودن کلاک برای ۹ کلاک سیکل
۲. بالا بودن SDA در هر یک از سیکل ها هنگامی که SCL یک است.
۳. ایجاد START CONDITON در زمان بالا بودن SDA

۴-۷ آدرس دهی سخت افزاری حافظه

منظور از آدرس دهی سخت افزاری، آدرسی که توسط پایه های A0، A1 و A2 برای انتخاب چیپ دلخواه در باس ایجاد می شود، است. این نوع از حافظه ها نیاز به ۸ بیت آدرس سخت افزاری دارند.

2K	1	0	1	0	A ₂	A ₁	A ₀	R/W
	MSB							LSB
4K	1	0	1	0	A ₂	A ₁	P0	R/W
8K	1	0	1	0	A ₂	P1	P0	R/W

بایت آدرس دهی سخت افزاری برای 2402/04/08

پایه های A0/A1/A2 برای آدرس دهی نوع AT2402A استفاده می شوند پس بنابراین نهایتاً ۸ وسیله می توانند در یک باس قرار گیرند. بیت اول نیز برای خواندن و نوشتن مورد استفاده می شوند. زمانی که بخواهید در حافظه بنویسید بیت اول را صفر و زمانی که بخواهید از حافظه بخوانید بیت اول را یک می کنید.

• مثال

آدرس سخت افزاری چیپ AT2402A جهت خواندن از این وسیله زمانی که پایه های A0، A1 و A2 همگی به زمین وصل شده اند برابر $161 = B10100001$ است.

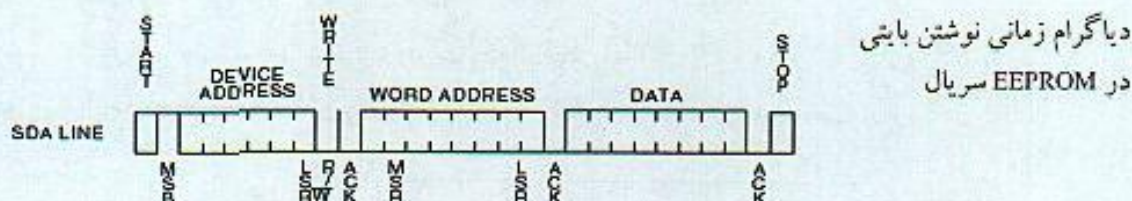
آدرس سخت افزاری چیپ AT2402A جهت نوشتن در این وسیله زمانی که پایه های A0، A1 و A2 همگی به یک وصل شده اند برابر $174 = B10101110$ است.

بیت های A1/A2 برای آدرس دهی نوع AT2404A استفاده می شوند. بیت اول نیز برای خواندن و نوشتن مورد استفاده می شود. 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 پایه های A1/A2 استفاده می شود و پایه A0 آزاد می باشد. پس بنابراین نهایتاً ۴ وسیله می توانند در یک باس مشترک استفاده شوند.

بیت A2 برای آدرس دهی نوع AT2408A استفاده می گردد. بیت اول نیز برای خواندن و نوشتن استفاده می شود. 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 پایه های A2 استفاده می شود. پایه A0/A1 آزاد می باشد پس بنابراین نهایتاً ۲ وسیله می توانند 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 زیر انجام می‌گیرد.



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.

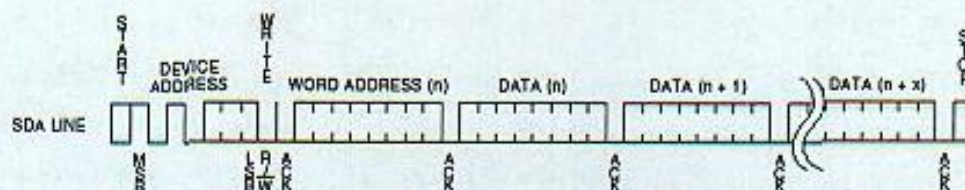
ACK : در تمام موارد مورد نیاز این سیگنال توسط کامپایلر به صورت خودکار ایجاد شده و نیازی به نوشتن برنامه اضافه برای آن نیست.

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 می‌گردد تا در خانه حافظه حفظ شود.

STOP : ایجاد STOP CONDITION که توسط دستور I2CSTOP در محیط BASCOM به خط یا باس SDA ارسال می‌گردد.

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، ۸ بایت و برای AT2404/08، ۱۶ بایت است. به طور مثال برای AT2404 بعد از نوشتن اولین بایت در حافظه با ارسال سیگنال ACK می‌توان تا ۸ بایت را به همین منوال نوشت. برای درک بیشتر به دیاگرام زمانی زیر توجه نمایید.



$X=15$, $X=7$ AND For AT2404/08, For AT2404

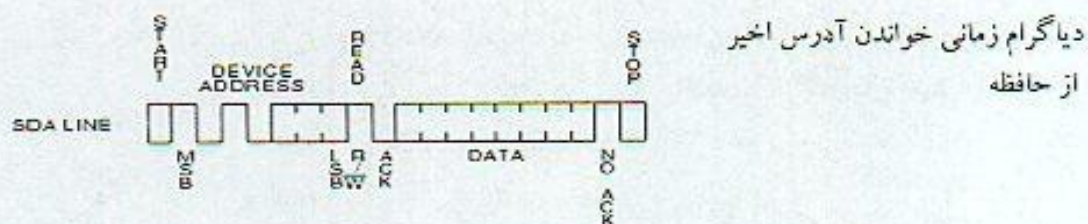
دیاگرام زمانی نوشتن صفحه‌ای

۶-۷ انواع عملیات خواندن حافظه

خواندن از حافظه شبیه نوشتن بر روی آن است با این تفاوت که در تنظیم بایت آدرس دهی سخت افزاری بیت W/R به یک تغییر می کند. سه نوع مختلف نوشتن موجود می باشد که در زیر آمده است.

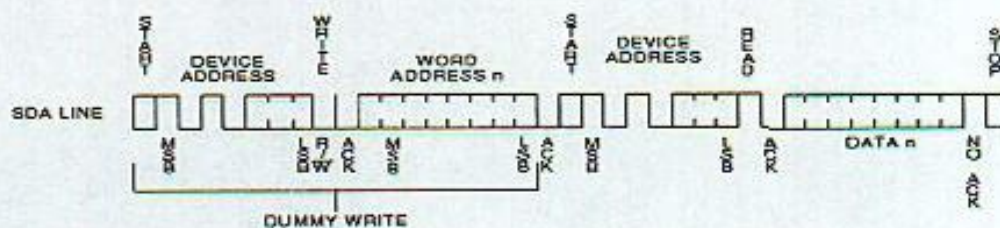
۱. خواندن آدرس اخیر (CURRENT ADDRESS READ) :

آدرس آخرین خانه حافظه چه از آن خوانده شده یا در آن نوشته شده باشد در کانتور داخلی حافظه EEPROM موجود می باشد. 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 می گردد و در زمان خواندن از آخرین خانه حافظه صفحه آخر به اولین خانه صفحه اول حافظه بر می گردد. زمانی که آدرس سخت افزاری EEPROM با بیت $R/W=1$ به چیپ ارسال می شود، داده خانه حافظه اخیر توسط EEPROM به بیرون ارسال می شود. برای درک بیشتر به دیاگرام زمانی زیر توجه کنید.



۲. خواندن از آدرس دلخواه (RANDOM READ) :

را از آدرس دلخواه خواند. برای درک طرز عملکرد به دیاگرام و توضیحات زیر توجه نمایید.



دیاگرام زمانی خواندن از آدرس دلخواه حافظه

START : ایجاد START COMDITION که می توان با دستور I2CSTART آن را در محیط BASCOM

ایجاد کرد.

DEVICE ADDRESS : منظور همان آدرس دهی سخت افزاری است که توسط بایت آدرس دهی

سخت افزاری برای نوشتن بدست آمده است که با دستور I2CWBYTE address می توان 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 I2CWRITE address به خط یا باس SDA ارسال می‌گردد.

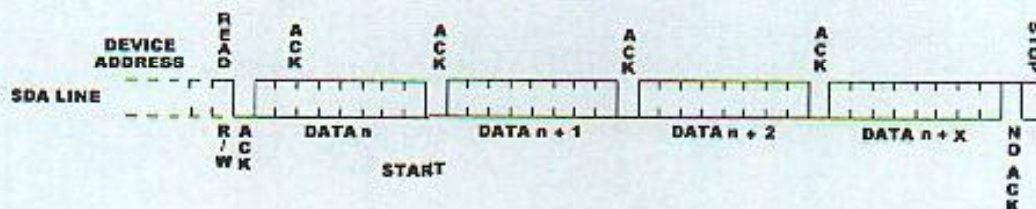
DEVICE ADDRESS: منظور همان آدرس دهی سخت‌افزاری است که توسط بایت آدرس‌دهی سخت‌افزاری برای خواندن بدست آمده است و با دستور I2CWRITE address می‌توان آن را به خط یا باس SDA ارسال کرد. توجه کنید که بیت $R/W = 1$ است.

DATA: داده دلخواه از خانه حافظه WORD ADDRESS توسط دستور I2CWRITE data,nack/ack خوانده شده و در متغیر بایت data قرار می‌گیرد.

STOP: ایجاد STOP CONDITION که توسط دستور I2CSTOP در محیط BASCOM 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. فل



دیاگرام زمانی خواندن دنباله‌ای از حافظه



پروژه‌های عملی

پروژه‌های عملی در این فصل برای تسلط و یادگیری بیشتر امکانات میکروکنترلرهای AVR ارائه شده است. مدارات موجود در این فصل بسته و تست شده‌اند و توسط نرم‌افزار PROTEUS 6.0 کشیده و در صورت امکان تحلیل شده‌اند. در مدارات بسته شده اکثراً از میکروکنترلرهای نوع ATMEGA8535 ، ATMEGA32 ، AT90S8535 و AT90S8515 استفاده شده است ولی کاربران عزیز می‌توانند از دیگر میکروهای AVR که دارای امکانات لازم هستند استفاده کنند. در شکل مدارات هر یک از برنامه‌ها، کریستال خارجی کشیده نشده است ولی دانشجویان در صورت از کریستال خارجی برای میکرو دلخواه خود بایستی کریستال را در بین دو پایه XTAL1 و XTAL2 قرار دهند و در صورت انتخاب RC خارجی به عنوان کلاک سیستم بایستی RC متناسب با فرکانس را بدست آورده و در پایه قرار دهند.

اهداف

۱. تحلیل کامل برنامه‌های نوشته شده و بستن مدار مربوطه
۲. یادگیری کامل برنامه‌نویسی و کار با تمام امکانات AVR ها
۳. تحلیل کردن برنامه‌های نوشته شده توسط SIMULATOR داخلی BASCOM
۴. تحلیل برنامه‌های نوشته شده توسط نرم افزار PROTEUS 6.0 تا حد امکان

۸-۱ 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 PORTB.0 و دیگری به PORTB.1 طبق شکل مدار صفحه بعد متصل شده است. مدت زمان تاخیر DEBOUNCE در حالت پیش فرض 25ms است که توسط دستور CONFIG DEBOUNCE قابل تغییر است.

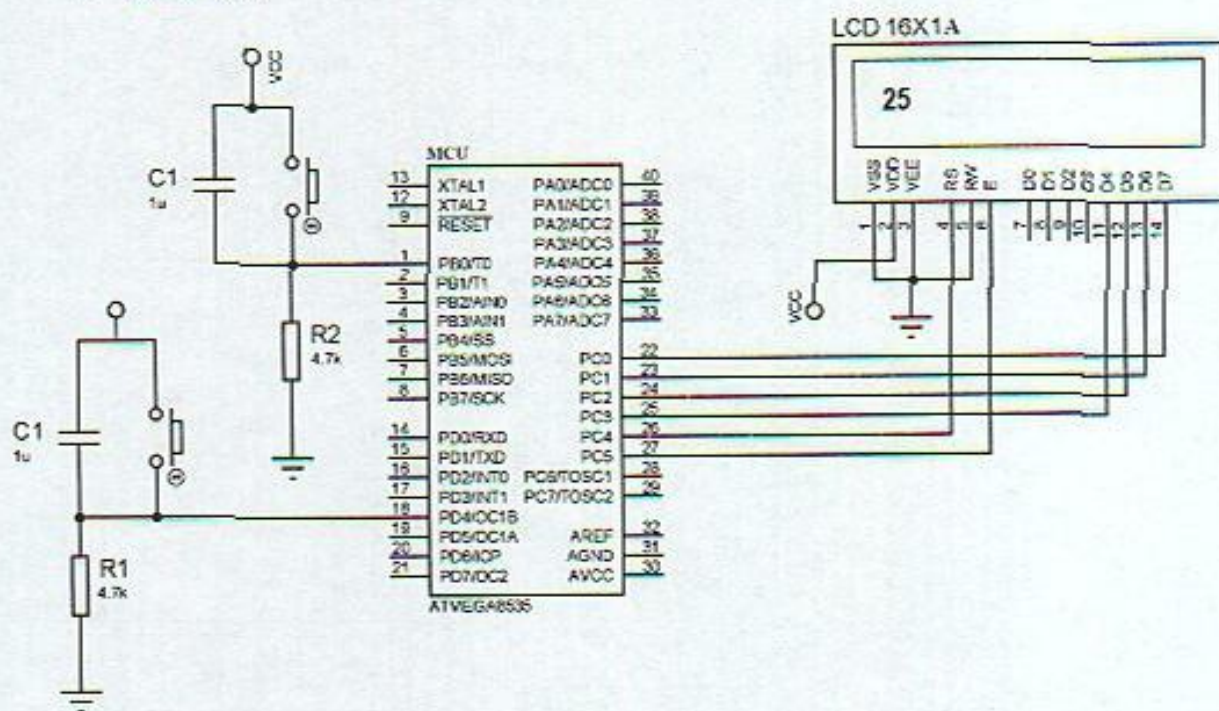
در این برنامه از میکرو ATMEGA8535 و فرکانس اسیلاتور داخلی 8MHZ استفاده شده است. LCD نوع 16 * 1A است و پایه متصل شده به کلیدها ورودی تعریف شده‌اند. دو زیر تابع INCREMENT و DECREMENT توسط دستور DECLARE تعریف شده‌اند. اجرای برنامه داخل حلقه DO - LOOP انجام شده و ابتدا پایه PINB.0 امتحان می‌شود اگر پایه یک باشد برای مدت زمان 100ms تاخیر ایجاد می‌شود، پس از سپری شدن زمان اگر هنوز پایه بالا باشد آن را به منزله فشردن کلید در نظر می‌گیرد و زیر تابع DECREMENT اجرا می‌شود در غیر اینصورت خط بعد اجرا می‌شود و اگر پایه در زمان امتحان شدن صفر باشد تاخیری ایجاد نشده و خط بعد اجرا می‌شود. پایه PINB.1 هم به همین منوال امتحان می‌شود.

در زیر برنامه DECREMENT عدد A یک واحد کم می‌شود و در زیر برنامه INCRMENT متغیر یک واحد افزایش می‌یابد و زمانی که $A > 30$ شود متغیر A با عدد 0 ریست می‌شود.

```
$regfile = "M8535.dat" ' we use the MEGA8535
'internal osc 8MHZ for atmega8535
Config Lcdpin = Pin , Db4 = Pinc.3 , Db5 = Pinc.2 , Db6 = Pinc.1 , Db7 = _
Pinc.0 , E = Pinc.5 , Rs = Pinc.4
Config Lcd = 16 * 1a
Config Pinb.0 = Input
Config Pind.4 = Input
Config Debounce = 100
Dim A As Byte
Declare Sub Increment
Declare Sub Decrement
Cursor Off
A = 0
Do
    Debounce Pinb.0 , 1 , Decrement , Sub
    Debounce Pind.4 , 1 , Increment , Sub
Loop
End 'end program

Sub Increment
    Cls
    Incr A
    If A > 30 Then A = 0
    Home
    Lcd A
    Return
End Sub Increment

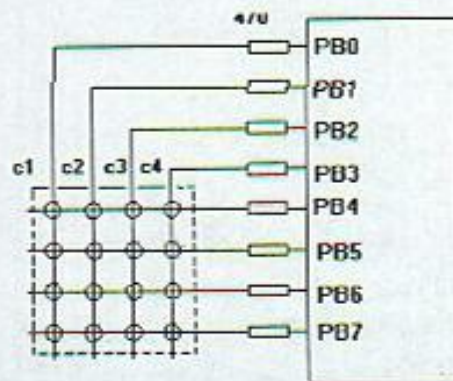
Sub Decrement
    Cls
    Decr A
    Home
    Lcd A
    Return
End Sub Decrement
```

مدار بسته شده برای برنامه اتصال کلید به میکرو توسط دستور DEBOUNCE

۲-۸ اسکن صفحه کلید ۴×۴

اسکن صفحه کلید در BASCOM کار ساده است و تنها کافی است صفحه کلید خود را طبق شکل زیر به یکی از پورت های میکرو وصل نمایید و توسط دستور CONFIG KBD آن را پیکره بندی کنید .



طرز اتصال صفحه کلید
به یک پورت میکرو

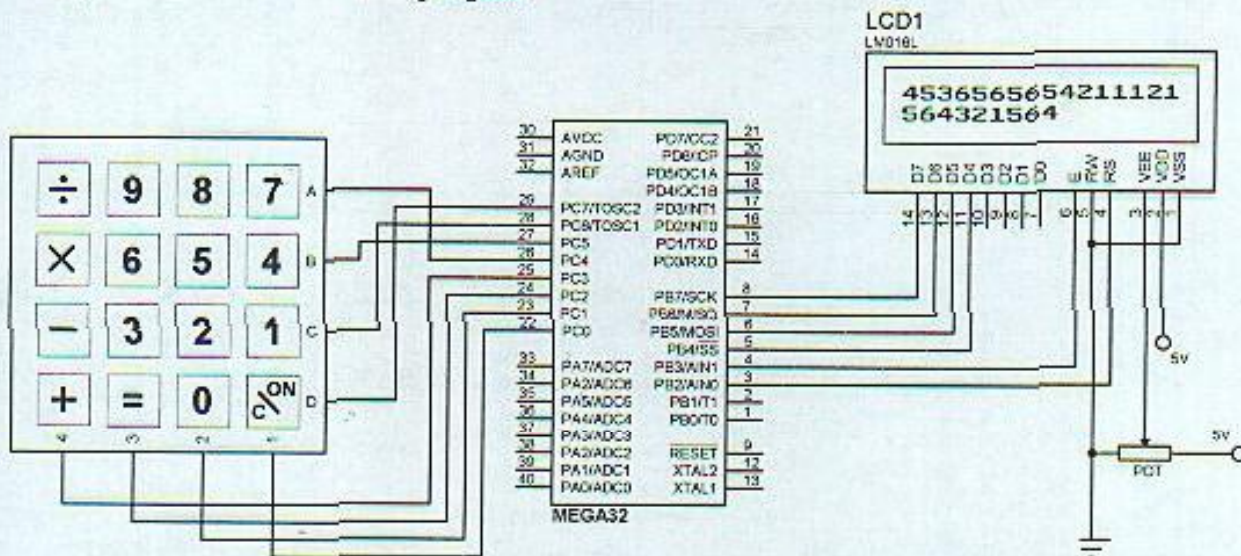
برنامه اسکن صفحه کلید ۴×۴ در محیط BASCOM

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 ۱۶×۲ استفاده شده است. صفحه کلید خوانده می شود و اگر کلید فشرده شده باشد و یا به عبارتی GETKBD() عددی کوچکتر یا مساوی ۱۵ را برگرداند عدد خوانده شده بر روی LCD نمایش داده می شود. زمانی که سطر بالای LCD نوشته شود ادامه آن در خط پایین LCD ادامه می یابد .


```

Sregfile = "m32def.dat"
Socrystal = 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
Dim Row As Byte , Column As Byte , A As Byte
Column = 1 : Row = 1
Main:
A = Getkbd()
If A > 15 Then Goto Main
Locate Row , Column
Lcd A
Waitms 500
Incr Column
If Column > 15 Then
Column = 1 : Incr Row
End If
If Row > 2 Then
Cls : Row = 1
End If
jmp main
End
'end program

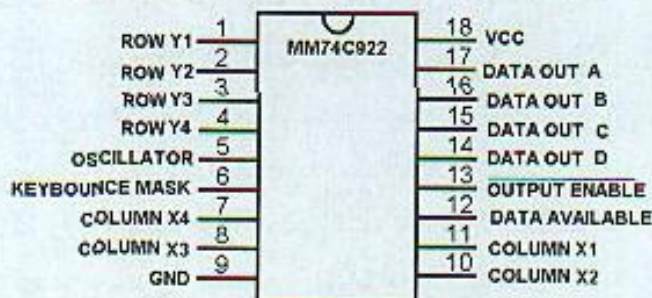
```



شکل مدار اسکن صفحه کلید ۴×۴

۳-۸ اسکن صفحه کلید ۴×۴ 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 را به کار برده‌ایم. 74C922 انکدر 16 کلیدی و 74C923 انکدر 20 کلیدی است. برای اسکن کی‌بورد 4x4 از MM74C922 استفاده می‌کنیم.



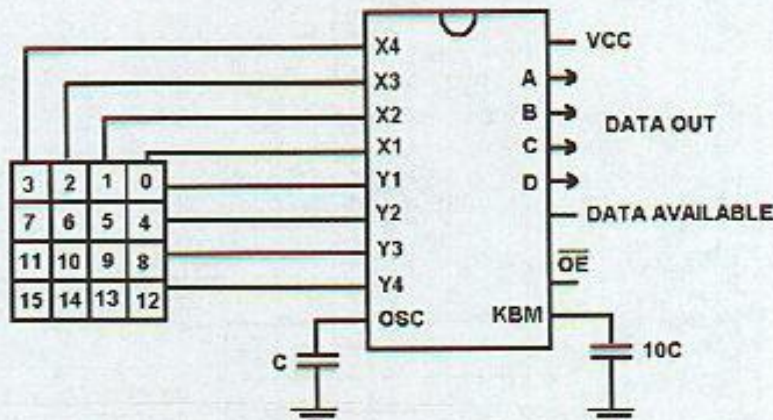
ترکیب پایه‌های انکدر MM74C922

۲۶۳ پروژه‌های عملی

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.

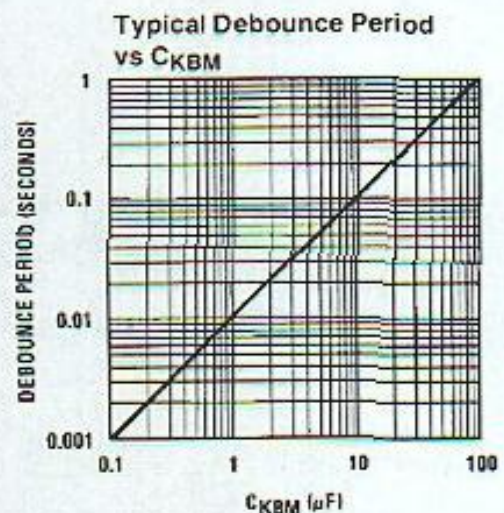
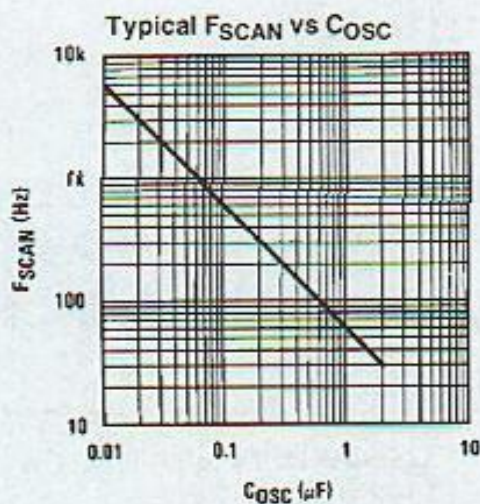
مقدار خازن $10\text{COSC} = \text{CKBM}$ است.

نکته



شکل اتصال صفحه کلید به
انکدر MM74C922

برای بدست آوردن خازن متصل بین زمین و پایه‌های OSCILLATOR، KEYBOUNCE MASK، یا به عبارتی برای تعیین زمان DEBOUNCE و فرکانس اسکن صفحه کلید از دو نمودار زیر استفاده می‌شود. در این پروژه $\text{COSC} = 1\mu\text{f}$ و $\text{CKBM} = 10\mu\text{f}$ است.



نمودارهای تعیین فرکانس اسکن و زمان DEBOUNCE برای اسکن صفحه کلید

```
$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 Portc = Input
```



```

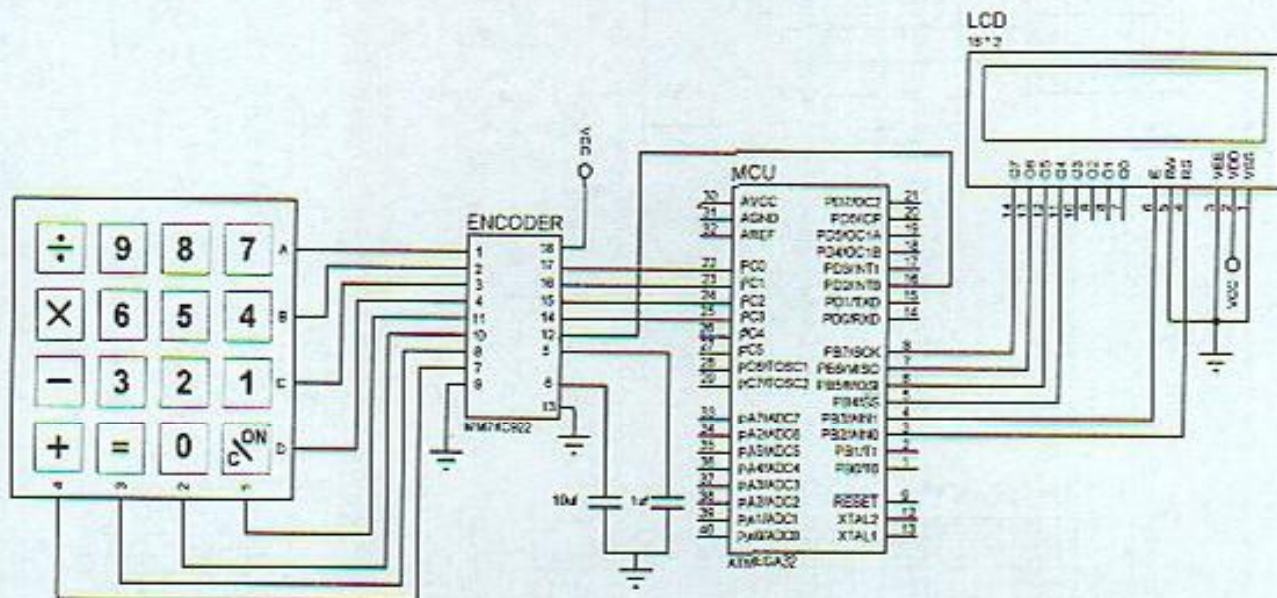
Config Portb = Output
Dim Asd As Byte
Dim A As Byte
Wait 1
Cls
Enable Interrupts
Enable Int0
Config Int0 = Rising
On Int0 Int11
Do

Loop
End

'end program

Int11:
A = Pinc
A = A And &B00001111
Home
Lcd A
Return

```



شکل مدار بسته شده برای پروژه اسکن صفحه کلید ۴×۴ توسط اتمکدر MM74C922

۴-۸ اسکن صفحه کلید کامپیوتر

صفحه کلید توسط دستور CONFIG KEYBOARD پیکره بندی شده است. پایه CLOCK صفحه کلید به پایه PINC.1 و پایه DATA صفحه کلید به پایه PINC.0 متصل شده اند. دستور GETATKBD() مقدار اسکی کلیدها را بر نمی گرداند 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 صفحه کلید، مقدار اسکی معادل آن از جدول برگردانده می شود. کلید فشرده شده از صفحه کلید توسط دستور GETATKBD() خوانده شده و در متغیر A قرار می گیرد.

در این برنامه با فشردن کلیدهای ۱، ۲، ۳، ۴ به ترتیب مکان نمای LCD به خط پایین، ستون راست

۲۶۵ پروژهای عملی

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 و مکان‌نما در آن مکان می‌ماند. با فشردن کلید ENTER (13) مکان‌نما به خط پایین پرش می‌کند و در نهایت با فشردن دکمه TAB (9) مکان‌نما 4 ستون به راست جهش می‌کند.

```
$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 Keyboard = Pinc.0 , Data = Pinc.1 , 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
```

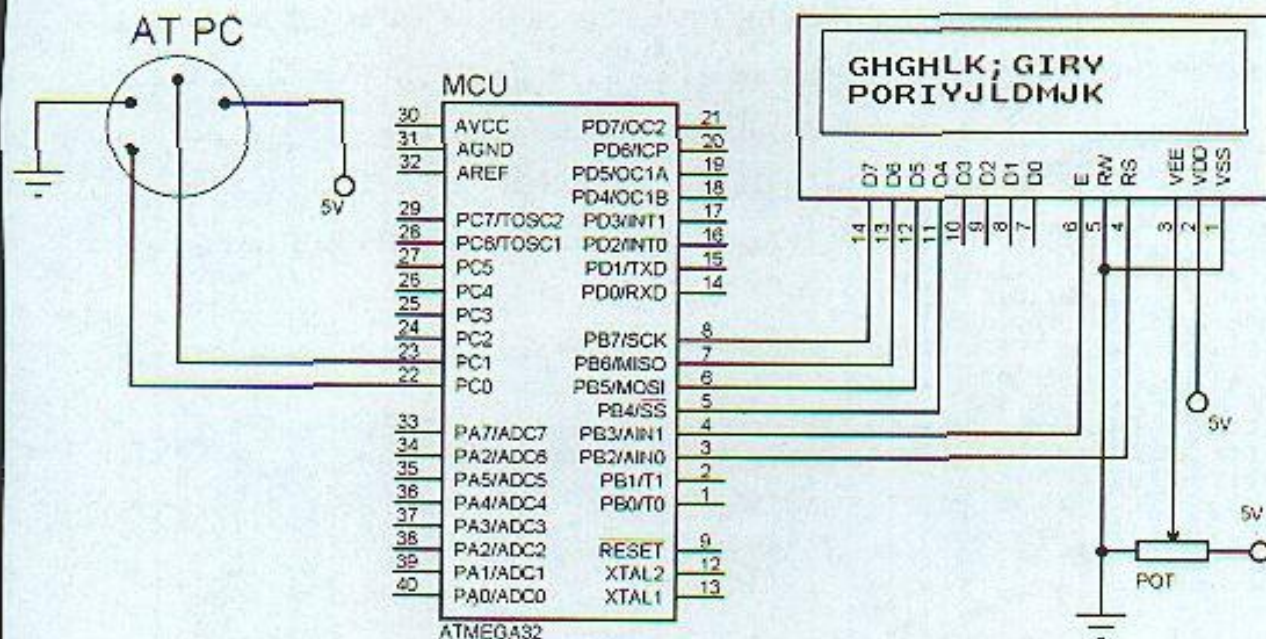
Keydata:

'normal keys lower case

```
Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 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 مدار اسکن صفحه کلید کامپیوتر

۵-۸ برنامه ساعت

در این برنامه از میکرو نوع ATMEGA32 و کریستال 8MHZ استفاده شده است. LCD با پایه‌های پورت B پیکره‌بندی شده است و پایه‌های PINC.0 و PINC.1 که کلیدهای تنظیم ساعت و دقیقه به آنها متصل شده است به عنوان ورودی پیکره‌بندی شده‌اند. 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.

```
$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 Pinc.1 = Input
Config Pinc.0 = Input
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 = 1 Then Call Incr_h
If Pinc.1 = 1 Then Call Incr_m
Locate 2 , 1
Lcd " " ; H ; ":" ; M ; ":" ; S
Waitms 995
Incr S
If S > 59 Then
S = 0
Incr M
```


۲۶۷ پروژه‌های عملی

```

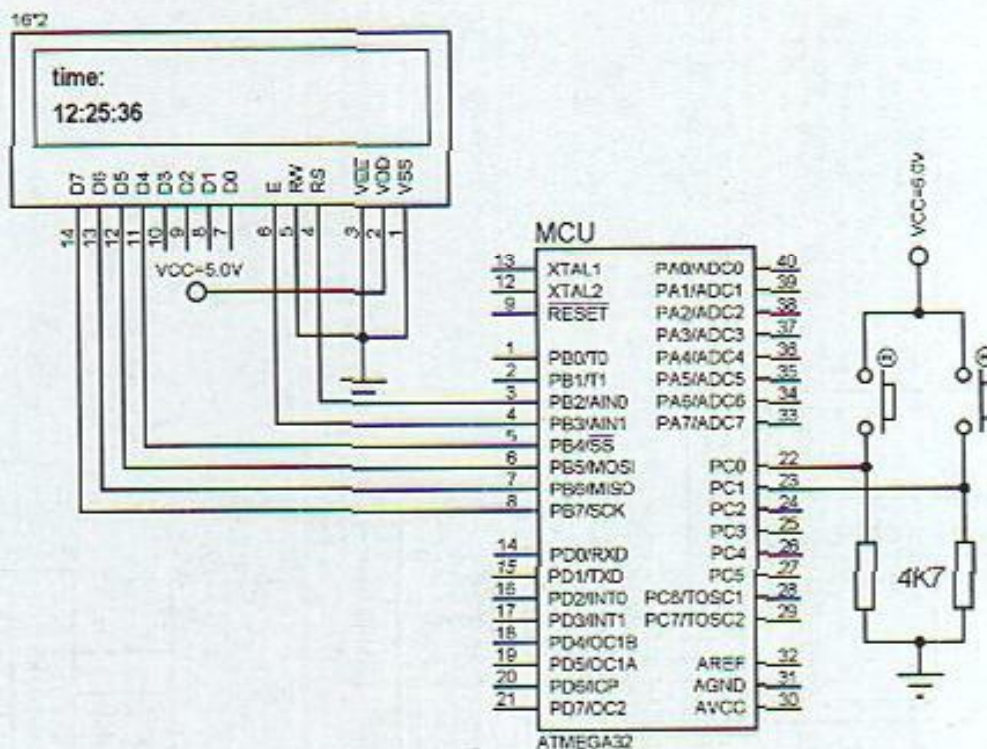
Shiftcursor Left , 2
Lcd " "
If M > 59 Then
    Incr H
    M = 0
If H > 12 Then
    jmp main
End If
End If
End If

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

```



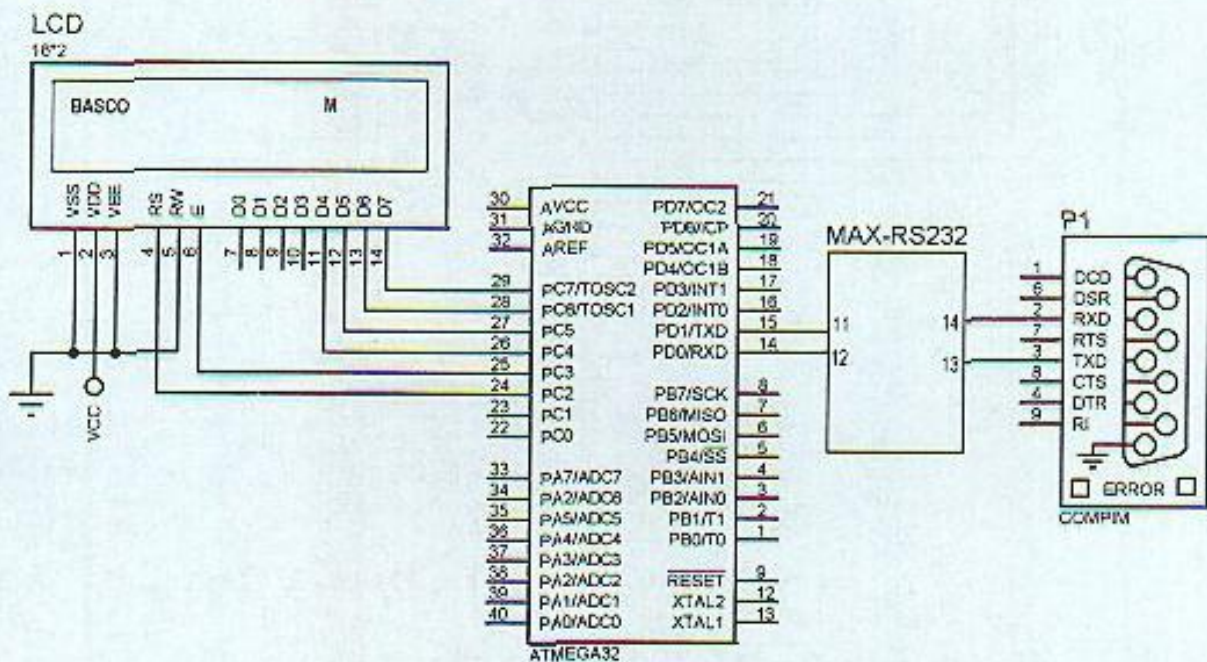
شکل مدار بسته شده برای برنامه ساعت

۶-۸ 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 : شما با نوشتن متن مورد نظر که نهایتاً می‌تواند 10 کاراکتر طول داشته باشد و فشار دادن کلید ENTER برنامه را اجرا کرده‌اید. این برنامه به طور کامل توسط SIMULATOR داخلی BASCOM قابل تحلیل است. توسط SSIM تحلیل سریعتر انجام می‌گیرد.

```
$regfile = "M32def.dat"
$crystal = 8000000
$baud = 9600
'$sim
Config Lcdpin = Pin , Db4 = Pinc.4 , Db5 = Pinc.5 , Db6 = Pinc.6 , Db7 = _
Pinc.7 , Rs = Pinc.2 , E = Pinc.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 مدار بسته شده برای برنامه تابلو روان توسط LCD

۷-۸ فرکانس متر 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 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 8,000,000HZ است.

$$256 \times 1024 \times 30 / 8000000 \approx 0.983040s \quad \text{تولید 1s توسط T/C0}$$

تایمر یک به صورت کانتر عمل می‌کند و در صورت سرریز 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 شمرد شده در زیربرنامه وقفه تایمر صفر مشخص شده و در LCD نمایش داده می‌شود.

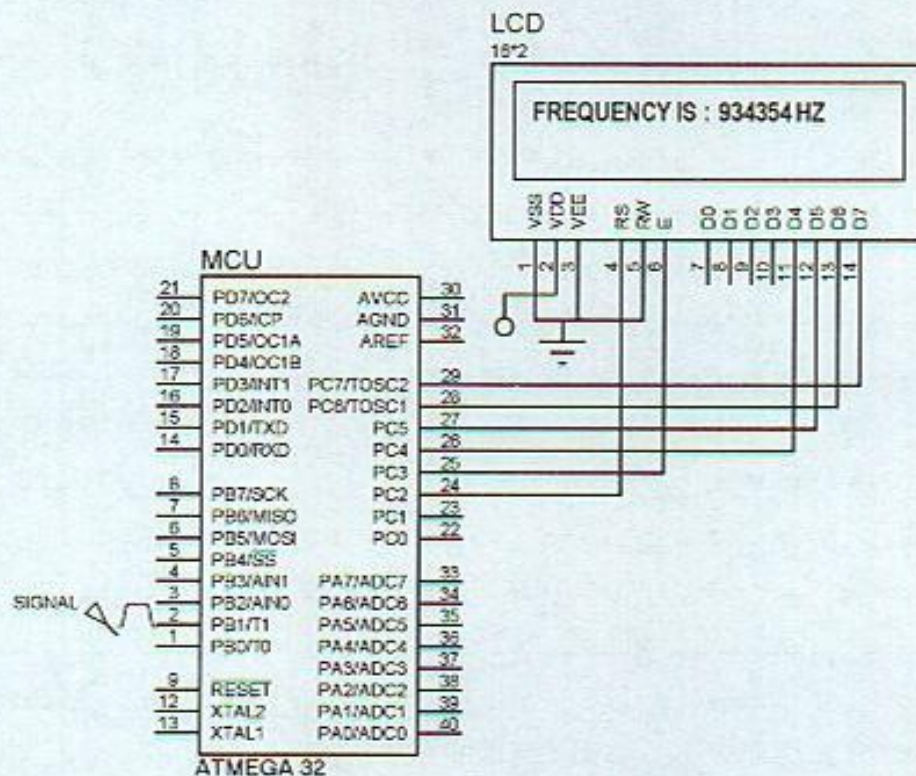
در برنامه زیر از میکرو نوع ATMEGA32 با کلاک داخلی در فرکانس 8MHZ استفاده شده است و LCD به پورت C متصل شده است.

```
$regfile = "M32DEF.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 Timer1 = Counter , Edge = Rising
Config Timer0 = Timer , Prescale = 1024
Enable Counter1
Enable Interrupts
Enable Timer0
Enable Timer1
On Ovfl Pulsecount
On Ovfo Ovfooccures
Dim A As Long , I As Long , B As Byte
B = 0
Cls
Start Timer0
Do

Loop
End 'end program

Ovfooccures:
Incr I
If I > 30 Then
Stop Timer0
Cls : Home
A = B * 65536
A = A + Counter1
Lcd "FREQUENCY IS : " ; A ; "HZ"
B = 0
I = 0 : Counter1 = 0
Start Timer0
End If
Return

Pulsecount:
Incr B
Counter1 = 0
Return
```

شکل مدار بسته شده
برای برنامه
فرکانس متر دیجیتال

۸-۸ کار با محیط TERMINAL EMULATOR

کار با محیط TERMINAL EMULATOR توسط UART سخت افزاری

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 می کنید، از این محیط می توانید برای نمایش داده ارسالی به کامپیوتر استفاده نمایید. این محیط تا حدی شبیه HYPER TERMINAL ویندوز است. توسط دستورات INPUT می توان داده تایپ شده در محیط TERMINAL را به میکرو ارسال کرد. برنامه زیر نحوه کار با محیط TERMINAL را نشان می دهد. زمانی که برنامه زیر را در میکرو برنامه ریزی کردید، پس از ارتباط سخت افزاری بین میکرو و PC محیط TERMINAL را با CTRL+T بالا بیاورید. بعد از ریست میکرو، رشته INPUT PASSWORD در محیط ظاهر می گردد. شما می توانید رشته ای را تا 10 کاراکتر تایپ کنید. رشته تایپ شده در متغیر S قرار می گیرد. در صورتی که رشته وارد شده BASCOM باشد، جمله VALID PASSWORD در محیط نوشته می شود و برنامه نام شما را درخواست می کند در غیر این صورت یعنی زمانی که PASSWORD وارد شده غیرمعتبر باشد برنامه با جملات END PROGRAM و INVALID PASSWORD پایان می یابد.

```
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 9600
Dim S As String * 10
```

```
Do
  Input "INPUT PASSWORD " , S
```


۲۷۱ پروژه‌های عملی

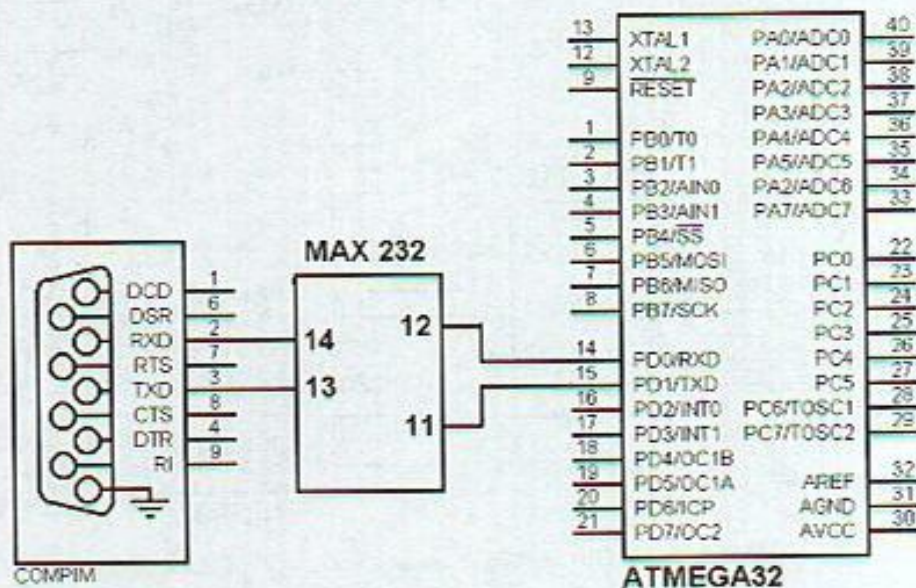
```

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

```



مدار بسته شده برای برنامه کار با محیط TERMINAL

کار با محیط TERMINAL EMULATOR توسط UART نرم‌افزاری

برنامه فوق در حالت UART سخت‌افزاری نوشته شده ولی می‌توان آن را به برنامه زیر با UART نرم‌افزاری تغییر داد. در برنامه زیر تمام دستورات INPUT به صورت NO ECHO هستند یعنی همانطور که در بخش ارتباط سریال فصل ششم گفته شده زمانی که شما در محیط TERMINAL تایپ می‌کنید داده وارد شده در محیط نمایش نداده خواهد شد. در زیر PORTB.1 بعنوان پایه RXD و PORTB.0 بعنوان پایه TXD مجازی استفاده شده‌اند.

```

$regfile = "M32DEF.dat"
$crystal = 8000000          '11059200
Open "COMB.1:9600,8,N,1" For Input As #1
Open "COMB.0:9600,8,N,1" For Output As #2
Dim S As String * 10
Do
    Print #2 , "INPUT PASSWORD"
    Input #1 , S
    Print #2 , S

    If S <> "BASCOM" Then
        Print #2 , "INVALIDE"
        Exit Do
    End If

```


MAX 232

ATMEGA32

Pin	Function	Pin	Function
1	PA0/ADC0	14	MAX232 Pin 14
2	PA1/ADC1	13	MAX232 Pin 13
3	PA2/ADC2	12	MAX232 Pin 12
4	PA3/ADC3	11	MAX232 Pin 11
5	PA4/ADC4		
6	PA5/ADC5		
7	PA6/ADC6		
8	PA7/ADC7		
9	PC0		
10	PC1		
11	PC2		
12	PC3		
13	PC4		
14	PC5		
15	PC6/TOSC1		
16	PC7/TOSC2		
17	AREF		
18	AGND		
19	AVCC		
20			
21			
22			
23			
24			
25			
26			
27			
28			

۹-۸ نمایش دما بر روی LCD توسط سنسور دمای LM35

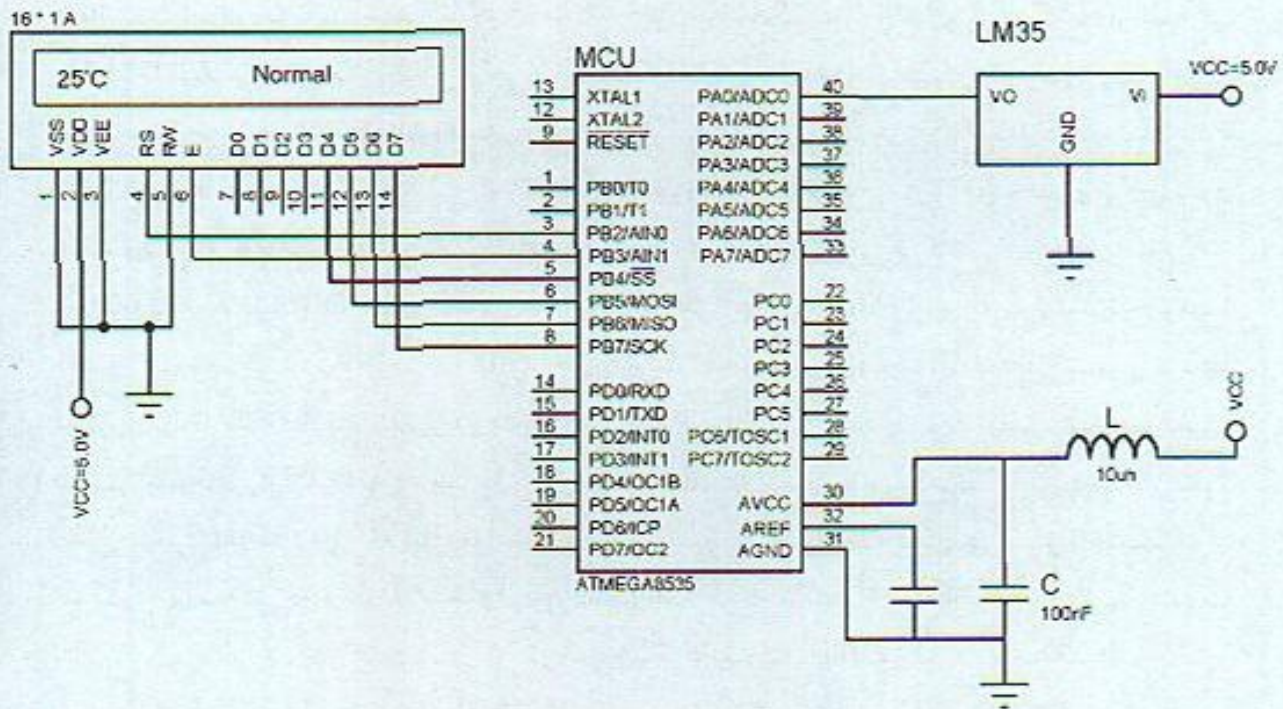
DEVICE	TEMP RANGE	ACCURECY	OUTPUT SCALE
LM35A	-55 C TO +150 C	+1. 0C	10mv/c
LM35	-55 C TO +150 C	+1. 5C	10mv/c
LM35CA	-40 C TO +110 C	+1. 0C	10mv/c
LM35C	-40 C TO +110 C	+1. 5C	10mv/c
LM35D	0 C TO +100 C	+2. 0C	10mv/c

به ازاء ولتاژ ورودی 0v توسط LM35 به کانال ADC داخلی میکرو، عدد 0 در متغیر W و در ازاء ولتاژ 1.0v مقدار دیجیتال شده 204 در متغیر W قرار می گیرد. عدد 0 نمایانگر 0°C و 204 نمایانگر 100°C

100 است. بنابراین با تقسیم مقدار دیجیتال به 2 می‌توان به سادگی دما را اندازه گرفت. شما از مدارات ZERO-SPAN می‌توانید برای تبدیل رنج 0 - 1.0V به 0 - 5.0V استفاده نمایید و یا ولتاژ +1.0V را به عنوان ولتاژ مرجع به پایه AREF اعمال کرد. در برنامه زیر از میکرو ATMEGA8535 و LCD نوع 16×1 استفاده شده است. این نوع از LCD ها همانطور که گفته شده است دارای خط دومی در ستون هشتم هستند و با نوشتن دستور HOME L مکان نما به ستون هشتم پرش می‌کند.

ADC به مد SINGLE استفاده شده و کلاک آن به طور بهینه (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 W نوشته می‌شود. 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 و در 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. میکرو انتخاب شده برای این برنامه ATMEGA8535 است. برنامه زیر توسط SIMULATOR داخلی BASCOM به طور کامل قابل تحلیل است.

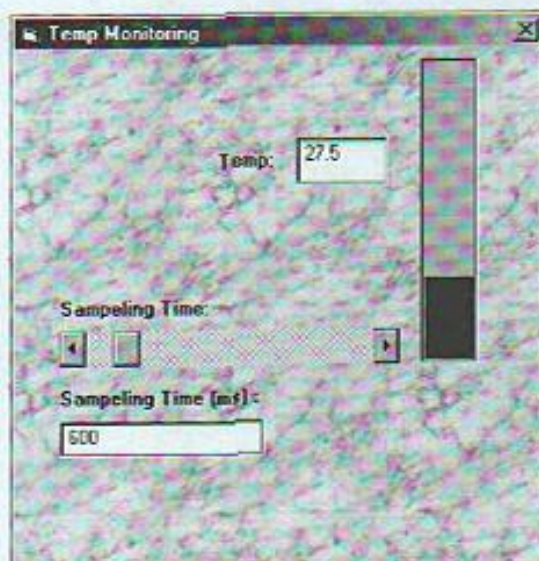
```
$regfile = "m8535.dat"
'we use internal osc 8000000 for atmega8535
$crystal=8000000
Config Lcdpin = Pin , Db4 = Pinc.3 , Db5 = Pinc.2 , Db6 = Pinc.1 , Db7 = _
Pinc.0 , E = Pinc.5 , Rs = Pinc.4
config Lcd = 16 * 1a
Config Adc = Single , Prescaler = Auto
Deflcdchar 0 , 24,24,32,32,32,32,32,32 'we replace ? with number 0
Dim W As Word
Start adc
cursor off
Do
    W = getadc(0)
    W = W / 2
    HOME
    lcd w ; chr(0) ; "C"
    WAITms 100
    select case W
        case is <= 10 :
            CLS
            HOME L
            lcd " Low"
        case 11 to 34 :
            CLS
            HOME L
            lcd " Normal"
        case is >= 35
            CLS
            HOME L
            lcd " High"
    end select
Loop
end 'end program
```

شکل مدار بسته شده برای برنامه نمایش دما بر روی LCD

۸-۱۰ مانیتورینگ دما توسط نرم افزار VISUAL BASIC 6.0

در این برنامه ولتاژ آنالوگ پایه خروجی سنسور LM35 متناظر با دمای محیط توسط مبدل آنالوگ به دیجیتال داخلی به عدد دیجیتال تبدیل و به پورت سریال فرستاده و سپس توسط برنامه VISUAL BASIC دما نمایش داده می شود. این برنامه امکانات دیگری مانند تغییر زمان نمونه برداری از دمای محیط را توسط SCROLLBAR در اختیار کاربر می گذارد. نمای گرافیکی نمایش دما در محیط VISUAL BASIC در زیر آمده است.



نمای گرافیکی نمایش دما در محیط VISUAL BASIC

برنامه VB

در این برنامه از یک تایمر برای تعیین زمان نمونه‌برداری استفاده شده است که در حالت پیش‌فرض زمان نمونه‌برداری 100ms قرار داده شده است. 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 شده، در عدد 0.49 ضرب شده و دمای بدست آمده در TEXT BOX و PROGRESSBAR نمایش داده می‌شود. با تغییر SCROLLBAR زمان نمونه‌برداری از دما، می‌تواند تغییر کند.

```
Dim A As String
Option Explicit

-----

Private Sub Form_load()
    Mscomm1.comport = 1
    Mscomm1.settings = "9600,n,8,1"
    Mscomm1.portopen = True
    Timer1.interval = 100
End Sub

-----

Private Sub Hscroll1_change()
    Timer1.interval = Hscroll1.value
    Text2.text = Hscroll1.value
End Sub

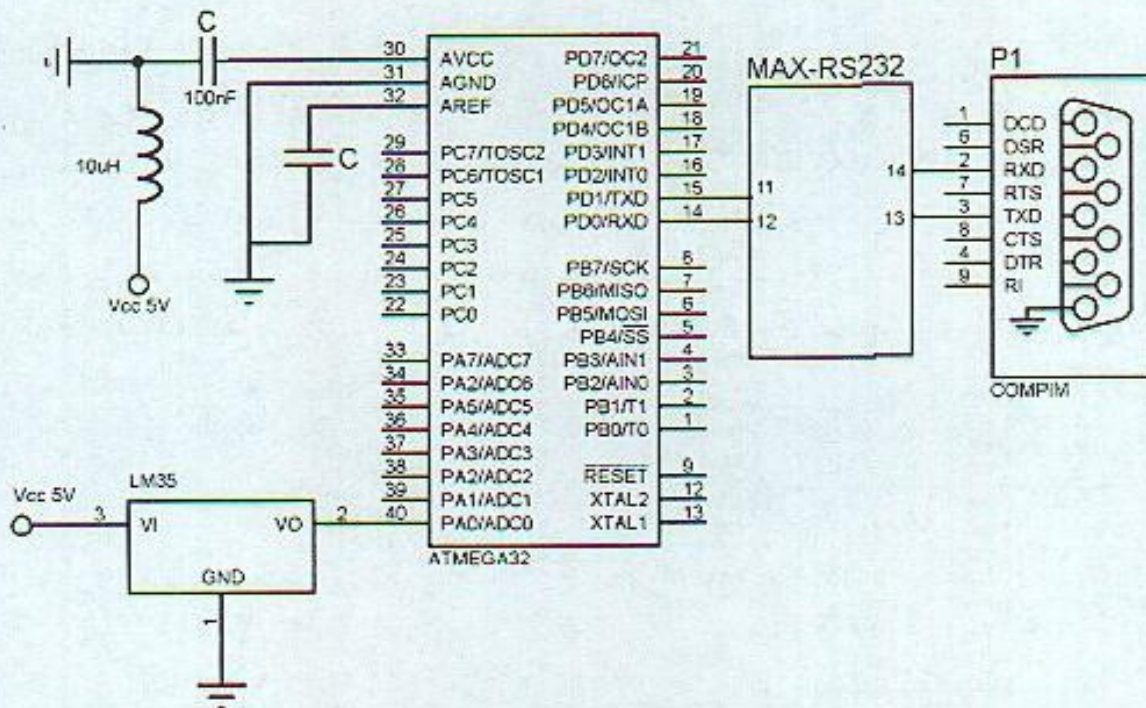
-----

Private Sub Timer1_timer()
    A = Mscomm1.input
    Text1 = Val(a) * 0.49 & " °C"
    A = Val(a) * 0.49
    Progressbar1.value = Val(a)
End Sub
```

برنامه BASCOM

در این برنامه از میکرو ATMEGA32 با کریستال داخلی 8MHZ استفاده شده است. سیگنال آنالوگ خارج شده از LM35 توسط ADC میکرو به دیجیتال تبدیل شده و با باودی مساوی با باود برنامه VB به پورت سریال فرستاده می‌شود.

```
$regfile = "M32def.dat"
$crystal = 8000000 'you can use 11059200 for less errors
$baud = 9600
Config Serialout = Buffered , Size = 10
Config Adc = Single , Prescaler = Auto
Start Adc
Dim C As Word
Do
    C = Getadc(0)
    Waitms 10
    Print C
Loop
End 'end program
```

شکل مدار بسته شده برای برنامه ماینورینگ دما توسط VB

۱۱-۸ موتور پله‌ای STEP 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 دائمی 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 ۵ سیم است: ۴ 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 زیر یک رشته چهار پله نرمال را نشان می‌دهد.

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	

جدول رشته چهار پله

باید توجه داشت که اگر چه می‌توان با هر یک از رشته‌های موجود در جدول فوق آغاز کرد ولی بمحض شروع باید ترکیب رعایت شود.

زاویه پله

زاویه پله مشخصه موتور را نشان می‌دهد که به ازاء هر پله موتور چند درجه چرخش دارد. جدول زیر بعضی از زوایا را برای انواع موتورها نشان می‌دهد.

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

جدول زاویه پله برای انواع موتورها

پله در ثانیه و دور در دقیقه (RPM)

رابطه بین RPM یا دور در دقیقه، پله در دور (SPT) و پله در ثانیه (SPS) از رابطه زیر بدست 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 چرخش نیازمند 200 پالس می‌باشد ($360 / 1.8 = 200$). با شمارش پالسها، میکروکنترلر AVR می‌تواند موقعیت موتور 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 رشته نیم پله صفحه بعد منطبق مورد نیاز برای برنامه را نشان می‌دهد. وقتی به انتهای جدول رسیدید، این روند را تکرار کرده و دوباره از ابتدای جدول شروع کنید.

برنامه موتور پله‌ای

در این پروژه برای راه‌اندازی موتور پله‌ای از راه‌انداز ULN2003 استفاده شده است. این درایور 16 پایه دارای 7 پایه راه‌انداز کلکتور باز است و بیشترین جریان خروجی آن برای هر پایه 500mA می‌باشد. جدول زیر چهار نوع از خانواده ULN را نشان می‌دهد.

ULN2001A	GENERAL PURPOSE , DTL , TTL , PMOS , CMOS
ULN2002A	14 – 25V PMOS
ULN2003A	5V TTL , CMOS
ULN2004A	6- 15V ,CMOS , PMOS

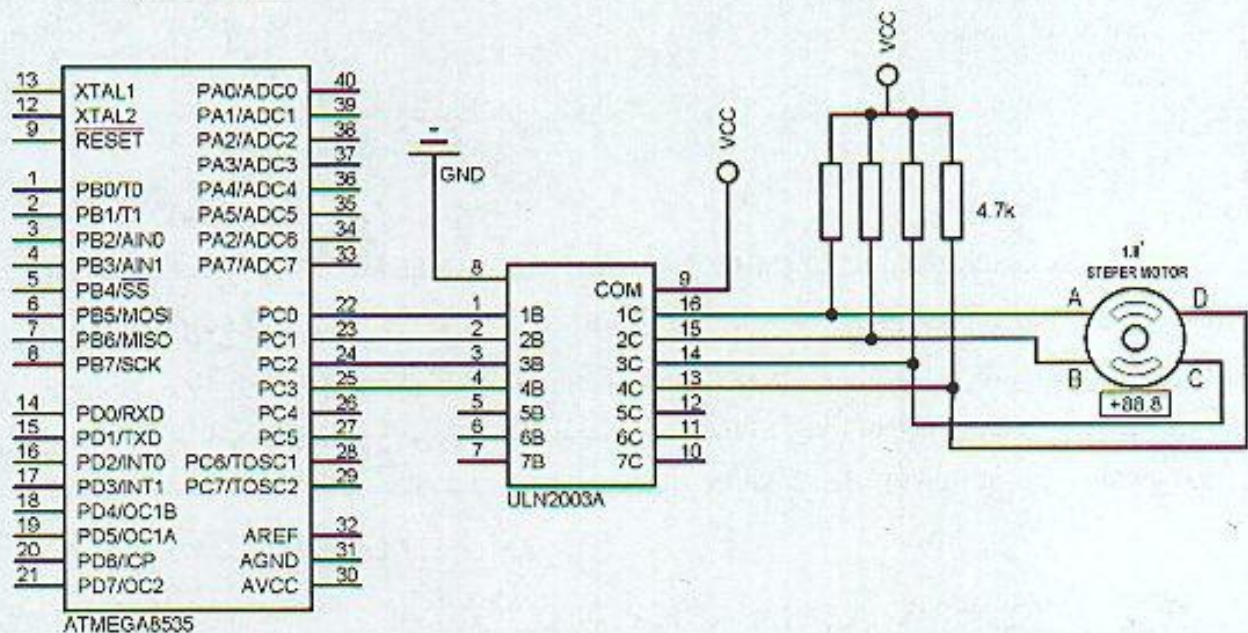
جدول چهار نوع از خانواده ULN

در جهت ساعت	#پله	سیم بیچ A	سیم بیچ B	سیم بیچ C	سیم بیچ 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	

جدول رشته نیم پله

این برنامه یک موتور پله‌ای با دقت 1.8 درجه را مدام یک بار پله کامل و بار دیگر به صورت نیم پله به اندازه 360 درجه در جهت ساعت می‌چرخاند.

```
$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
```

شکل مدار بسته شده برای برنامه موتور پله ای

۱۲-۸ 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 فشردن کلید عدد متناسب با کلید فشرده شده از طریق خط SPI به SLAVE انتقال داده می‌شود 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.

```
Sregfile = "m32def.dat"
Scrystal = 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

MASTER میکرو نوع ATMEGA8535 با کریستال داخلی 8MHZ است. ارتباط SPI به صورت سخت افزاری و با پایه‌های پیش فرض که در پورت B قرار دارد انجام می‌گیرد. 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 بایس SPI دریافت می‌گردد و بر روی LCD نمایش داده می‌شود.

```

$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 = On , 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 فشردن کلید عدد متناسب با کلید فشرده شده از طریق خط SPI به SLAVE انتقال داده می‌شود 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 بایس SPI 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 SPIOU به بایس SPI ارسال می‌شود.

۲۸۱ پروژه‌های عملی

```
$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

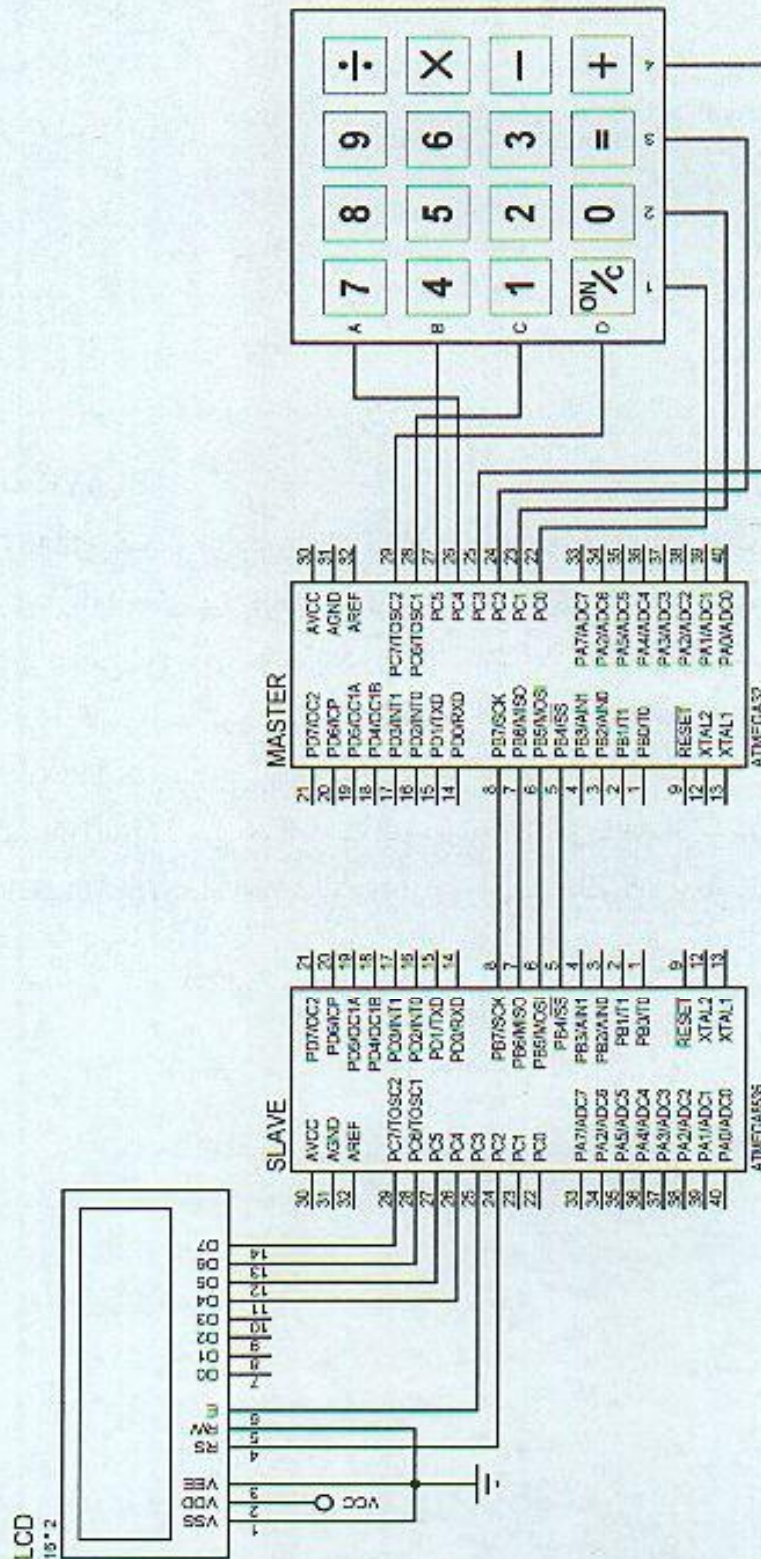
MASTER میکرو نوع ATMEGA8535 با کریستال داخلی 8MHZ است. ارتباط SPI به صورت سخت افزاری و با پایه‌های پیش فرض که در پورت B قرار دارد انجام می‌گیرد. وقفه فعال است و LSB داده ابتدا گرفته می‌شود. در حالت بیکاری پایه کلاک بالا نگه داشته می‌شود. فرکانس انتقال داده برابر با (128 / فرکانس اسپلاتور) می‌باشد. توسط 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 و محتوای رجیستر داده SPI با نام SPDR بر روی LCD نمایش داده می‌شود.

```
$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 مدار بسته شده
برای دو برنامه ارتباط
SPI با وقفه وبدون
وقفه



۸-۱۳ ارتباط با حافظه EEPROM سریال AT24C256

حافظ EEPROM سریال AT24C256 دارای 256KBIT حافظه است که قابلیت ارتباط با پروتکل سریال (I2C) 2-WIRE را داراست. آدرس سخت افزاری آن طبق بایت زیر بدست می آید.

1	0	1	0	0	A1	A0	R/W
MSB					LSB		

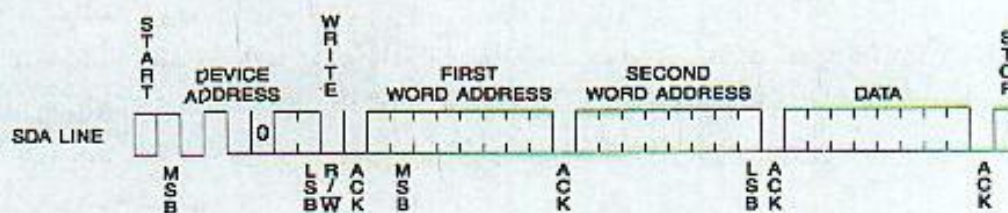
(بایت آدرس‌دهی سخت‌افزاری)

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 شکل زیر):

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.
۴. ارسال ابتدای بایت بالای آدرس حافظه دلخواه توسط دستور I2CWBYTE. این بایت می‌تواند توسط دستور HIGH از یک متغیر دو بایتی بدست آید.
۵. ایجاد ACK که کامپایلر خودکار آن را ایجاد کرده و دیگر نیازی نیست برنامه‌ای برای آن نوشته شود.
۶. ارسال بایت پایین آدرس حافظه دلخواه توسط دستور I2CWBYTE. این بایت می‌تواند توسط دستور LOW از یک متغیر 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



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 متغیر address از نوع داده WORD جدا شده‌اند.

```
$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 Writeepromserial
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 Writeepromserial
Incr Address
wend

End                                     'end program

Sub Writeepromserial
I2cstart
I2cwbyte Eewrite
i2cwbyte a
I2cwbyte b
i2cwbyte 7
I2cstop
Waitms 10
End Sub Writeepromserial
```

برنامه خواندن از حافظه

برای خواندن از حافظه به صورت بایتی بایستی مراحل زیر بر روی پایه SDA طی شود (طبق شکل صفحه بعد):

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.
۴. ارسال ابتدای بایت بالای آدرس حافظه دلخواه توسط دستور I2CWBYTE. این بایت می‌تواند توسط دستور HIGH از یک متغیر دو بایتی بدست آید.
۵. ایجاد ACK که کامپایلر خودکار آن را ایجاد کرده و دیگر نیازی نیست برنامه‌ای برای آن نوشته شود.
۶. ارسال بایت پایین آدرس حافظه دلخواه توسط دستور I2CWBYTE. این بایت می‌تواند توسط دستور LOW از یک متغیر دو بایتی بدست آید.

۲۸۵ پروژه‌های عملی

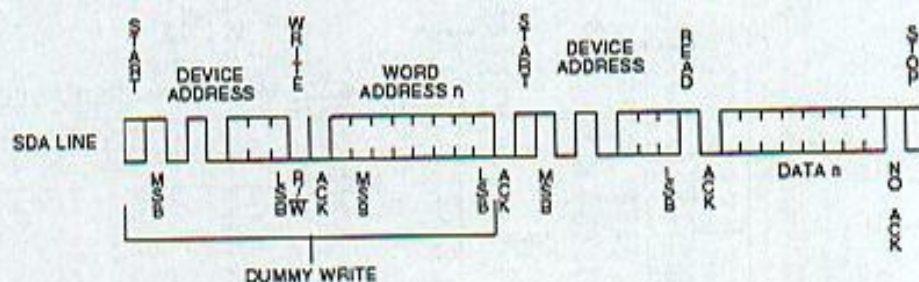
۷. ایجاد ACK که کامپایلر خودکار آن را ایجاد کرده و دیگر نیازی نیست برنامه‌ای برای آن نوشته شود.

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



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 متغیر address از نوع داده WORD جدا می‌شوند.

```
$regfile = "M32def.dat"
Scrystal = 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

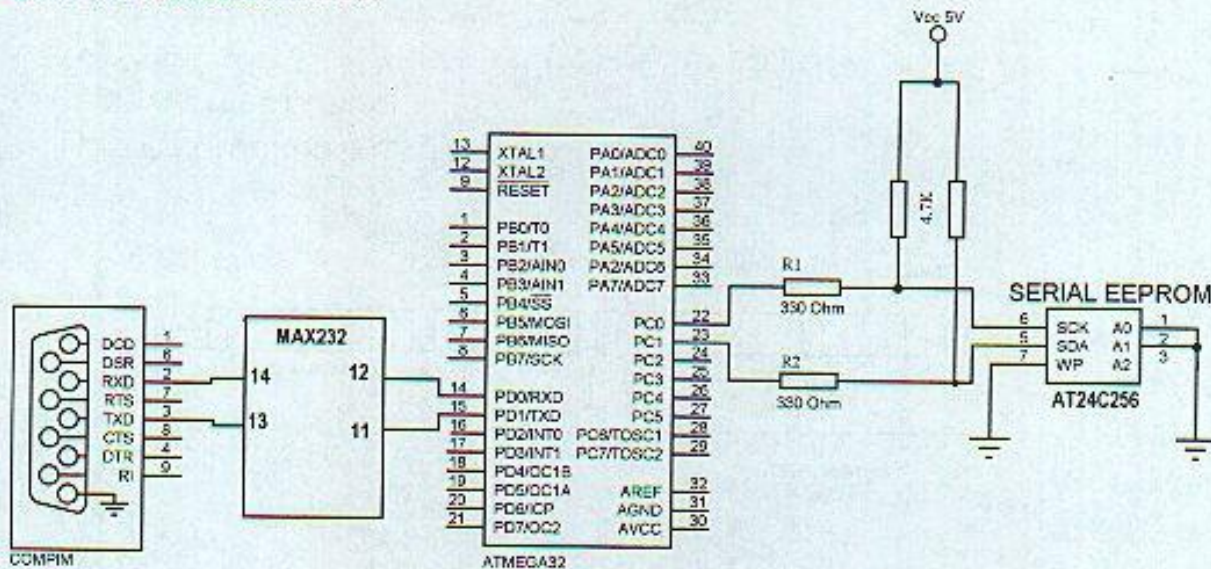
Sub Readeepromserial
    I2cstart
    I2cwbyte Eewrite
    I2cwbyte a
```



```

i2cwrite b
I2cstart
I2cwrite Eeread
I2cwrite A , Nack
I2cstop
print a
Incr Address
End Sub ReadEEPROMserial

```



شکل مدار بسته شده برای برنامه ارتباط با حافظه سریال

۱۴-۸ ساخت پالس PWM توسط VISUAL BASIC

در این پروژه قصد داریم از طریق برنامه ویژوال بیسیک عددی را به پورت سریال ارسال کرده و میکرو متصل شده به پورت سریال عدد را دریافت کند و پالس PWM متناسب با آن را در پایه خروجی OCR1A ایجاد کند. به همین منظور بایستی دو برنامه یکی برای ویژوال و دیگری برای میکرو نوشته شود.

برنامه VB

محیط گرافیکی برنامه به صورت شکل صفحه بعد است. عددی از 1 تا 255 از طریق تغییر SCROLL BAR به پورت سریال ارسال می‌شود. برای ارتباط با پورت سریال در VB از MSCOMM استفاده شده است.

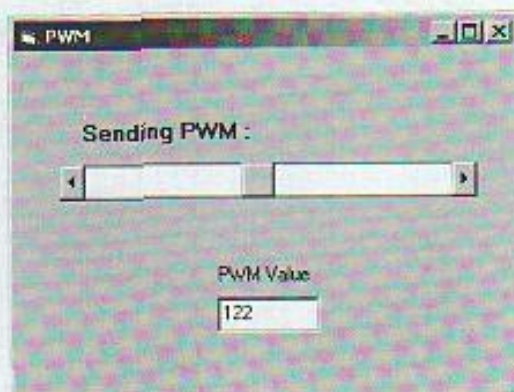
```

Dim a As String
Option Explicit

Private Sub Form_Load()
    MSCOMM1.CommPort = 1
    MSCOMM1.Settings = "9600,n,8,1"
    MSCOMM1.PortOpen = True
End Sub

Private Sub HScroll11_Change()
    a = HScroll11.Value
    MSCOMM1.Output = Chr$(a)
    Debug.Print a
    txtPwm = a
End Sub

```

نمای گرافیکی برنامه ساخت پالس PWM توسط VB

برنامه BASCOM برای میکرو

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 $8000000 / (2046 * 8) = 488 \text{ Hz}$ است. با تغییر PRESCALE و OSC شما به راحتی می‌توانید فرکانس‌های مختلف برای PWM ده بیتی ایجاد کنید. خروجی پالس PWM پایه OCR1A است. مدار بسته شده برای این پروژه ارتباط سریال بین PC و MCU است.

```
$regfile = "m8535.dat"
$crystal = 8000000
$baud = 9600
Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down , Prescale = 8
Dim A As Byte
Dim B As Word
Do
Main:
B = Inkey()
If B = 0 Then Goto Main
B = B * 4
Pwm1a = B
Loop
End
```

۱۵-۸ 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 می‌شود. 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 می‌توانید از محیط TERMINAL EMULATOR برای اجرای برنامه استفاده نمایید.

زمانی که خروجی مقایسه‌کننده آنالوگ معکوس شود زیر برنامه Analogcominit اجرا می‌شود ولی زمانی که خروجی مقایسه‌کننده آنالوگ یک خروجی با لبه پایین رونده داشته باشد، CAPTURE تایمر یک تریگ شده و زیر برنامه Capturetimer1 اجرا می‌شود.

```
$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 Icpl
Enable Aci
On Icpl Capturetimer1
On Aci Analogcominit
Start Timer1
Do
Loop
End
'end program

```

```

Capturetimer1:
Print "timer1" ; Timer1
Return

```

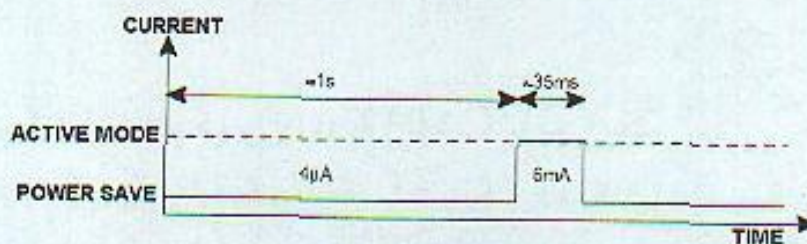
```

Analogcominit:
Print "on aci"
Return

```

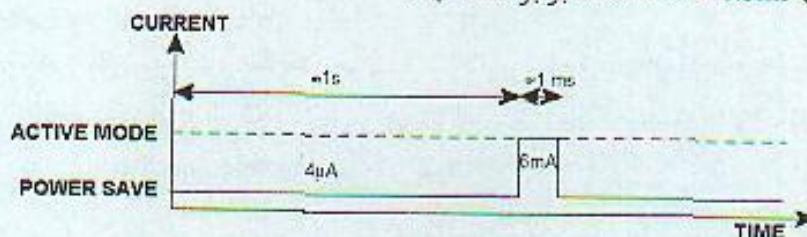
۱۶-۸ RTC (REAL TIME CLOCK)

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 تأمین 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 مد POWER-SAVE به حالت ACTIVE رفته و ISR مربوطه را انجام داده و سپس به حالت POWER-SAVE برمیگردد. تمام جریان مصرفی در یک ثانیه برای کریستال بین پایه‌های XTAL 1,2 با زمان $STARTUP=35ms$ برابر است با:



$$= (1s * 4uA) + (35ms * 6mA) = 4uAs + 210uAs = 214uAs$$

تمام جریان مصرفی در یک ثانیه برای نوسانگر سرامیکی (CERAMIC RESONATOR) بین پایه‌های XTAL 1,2 با زمان $STARTUP=0.5ms$ برابر است با:



$$= (1s * 4uA) + (0.5ms * 6mA) = 4uAs + 6uAs = 10uAs$$

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

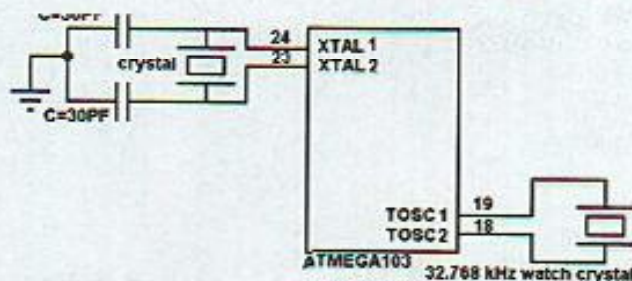
جدول مقدار جریان مصرفی در مدهای مختلف برای میکروکنترلرهای AVR

زمانهای مختلفی که می‌توان با کریستال 32768HZ توسط تایمر صفر یا دو ساخت در جدول زیر آمده است.

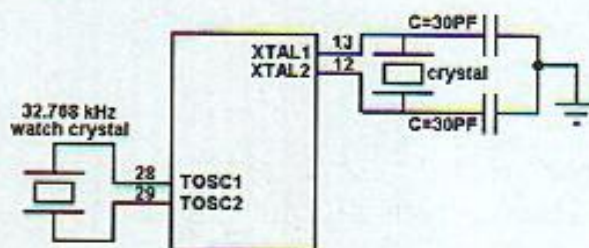
DESCRIPTION	OVERFLOW PERIOD
STOP, TIMER/COUNTER IS STOPED	-
CK2,0	1/64 S
CK2,0 /8	1/8 S
CK2,0 /32	1/4 S
CK2,0 /64	1/2 S
CK2,0 /128	1 S
CK2,0 /256	2 S
CK2,0 /1024	8 S

جدول انتخاب کلاک تایمر/کانتر 0 یا 2 به ازاء CK2,0=32,768 HZ در مدهای آسنکرون
[Note: CK2,0 = 32,768 Hz]

طرز اتصال کریستال ساعت به پایه‌های TOSC در شکل‌های زیر آمده است.



شکل اتصال کریستال ساعت به پایه‌های
TOSC1,2 در مدهای آسنکرون
تایمر/کانتر صفر برای ATMEGA103



شکل اتصال کریستال ساعت به پایه‌های
TOSC1,2 در مدهای آسنکرون تایمر/کانتر صفر

در برنامه زیر از CONFIG CLOCK استفاده شده است. این پیکره‌بندی توسط BASCOM برای ساده‌تر کردن کار با تایمر در مدهای آسنکرون طراحی شده است و برای تایمر دو در AT90S8535 و تایمر صفر در ATMEGA103 قابل اجرا است. این دستور خودکار، تایمر را در مدهای آسنکرون پیکره‌بندی کرده و زمان 1s را تولید می‌کند.

Config Clock = Soft [, Gosub = Sectic]

بعد از سرریزی تایمر (Is) زیربرنامه اختیاری Sectic اجرا می‌شود. نوشتن و استفاده کردن از این برچسب اختیاری است. زمانی که از این پیکره‌بندی استفاده می‌نمایید متغیرهای TIME\$ (hh:mm:ss) و DATES (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 متغیرهای DATE\$ و TIME\$ نسبت داد.

```
$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 :

```
$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.

۸-۱۸ LCD گرافیکی

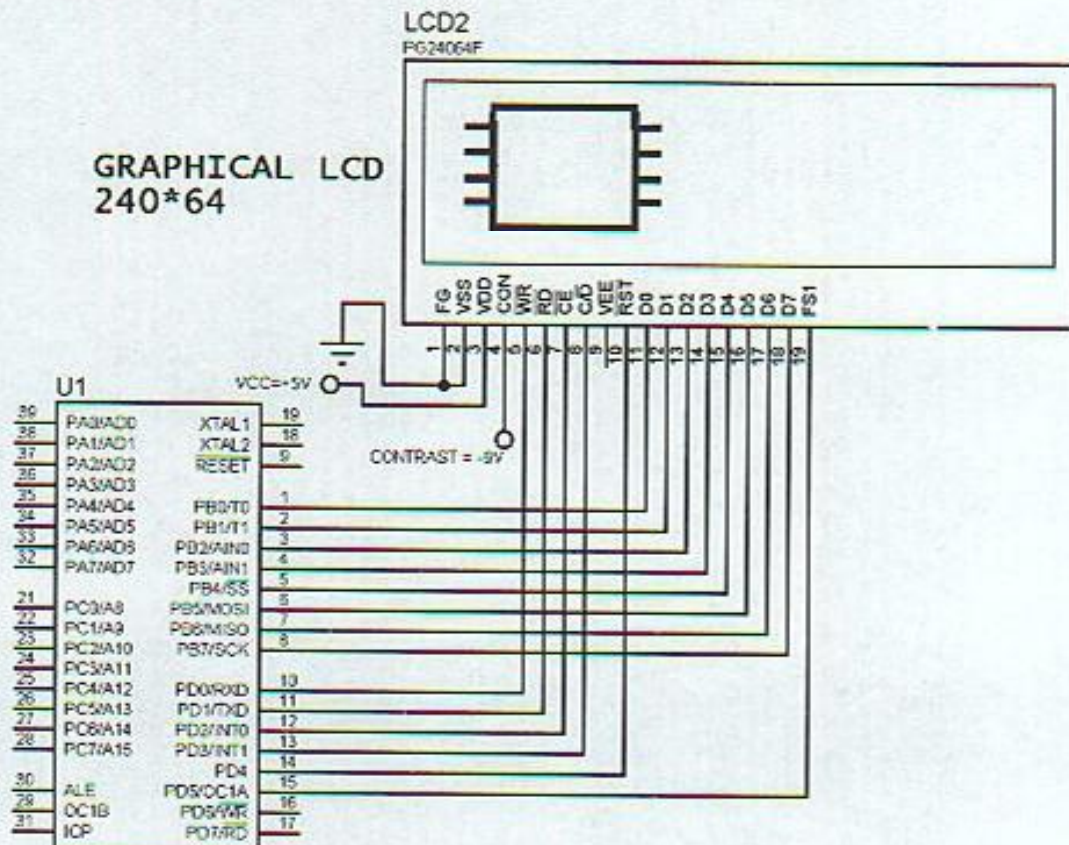
در محیط BASCOM ، LCD گرافیکی را پیکره‌بندی می‌کنیم. پیکره‌بندی، دستورات و طرز اتصال پایه‌های LCD به میکرو در فصل ششم آمده است. در این برنامه عکسی با پسوند *.BMP در محیط PAINT ویندوز کشیده شده است و سپس با باز کردن محیط GRAPHIC CONVERTOR، توسط کلید LOAD عکس کشیده شده را به این محیط وارد می‌کنیم. سپس آن 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 Is ، با دستور CLS GRAPHI عکس کشیده شاه پاک می‌شود. با بردن مکان‌نما به موقعیت (1,1)

۲۹۱ پروژه‌های عملی

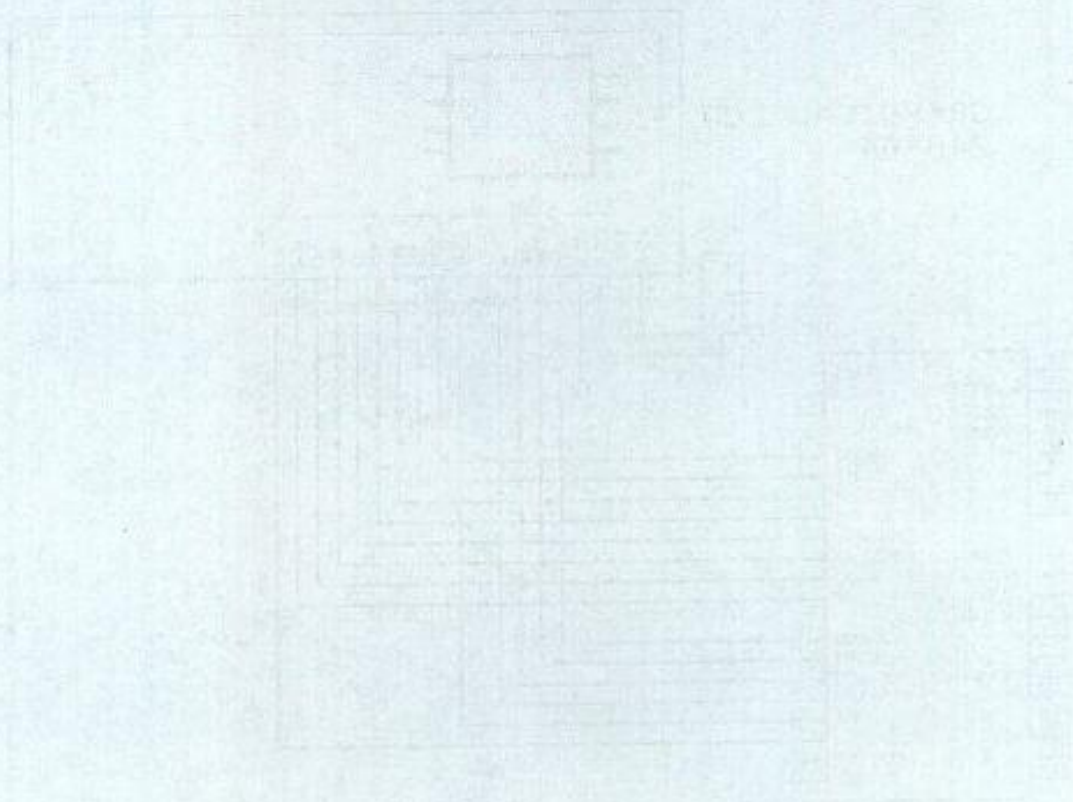
متن MCS ELECTRONIC توسط دستور LCD نمایش داده می‌شود و بعد از 1s متن نوشته شده را با دستور CLS TEXT پاک می‌شود. توسط دستور CIRCLE دایره‌ای با قطر 10 پیکسل به رنگ مشکی در موقعیت (60,30) می‌شود.

در این برنامه از LCD با اندازه 240*64 پیکسل استفاده شده است. شما می‌توانید از نوع‌های دیگر نیز استفاده نمایید.

```
$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 مدار بسته شده برای برنامه LCD گرافیکی



۹

پیوست: خطاهای 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 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.

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, the clearing 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 "lsl".

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 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 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 3.0V**
- **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 "lsl".

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 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/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 any VCC.

11. Error in Half Carry Flag

The half carry flag is undefined after executing the commands "ror", "asr" and "lsl".

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 during 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 μ A at 6V, rather than the specified 50 μ A. 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 μ A and sink up to 6 μ A 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 300K per pin. Or use devices with date codes later than 9836.

AT90S/LS8535Rev. 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 "lsl".

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 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.

AT90S4414Rev. 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 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 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 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.

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 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.

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.

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 μ A 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

**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 UBRRH 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 UBBRH updates baud rate generator for both UART0 and UART1. The baud rate generator's counter is updated each time either UBRR or UBBRH are written. Since the UBBRH register is shared by UART0 and UART1, changing the baud rate for one UART will affect the operation of the other UART.

Problem Fix/Workaround

Do not update UBBRH 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 two-word 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 SF8 (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 NACK 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 UBRL 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 UBRL incorrectly clears the UBRRH setting.

Problem Fix/Workaround

UBRRH must be written after UBRL because setting UBRL 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 compatibility with corrected devices.

```
ldi r17, HIGH(baud)
ldi r16, LOW(baud)
out UBRRH, r17 ; Added for upward compatibility
out UBRL, r16 ; Set new UBRL, 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 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.

ATmega128 Rev. F

There are no errata for this revision of ATmega128.

ATmega128 Rev. G

There are no errata for this revision of ATmega128.

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: $N \oplus V$, 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.

۳۰۹ خطاهای میکروکنترلرهای AVR

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
$Rd > Rr$	$Z \leftarrow (N \oplus V) = 0$	BRLT ⁽¹⁾	$Rd \leftarrow Rr$	$Z \leftarrow (N \oplus V) = 1$	BRGE*	Signed
$Rd \geq Rr$	$(N \oplus V) = 0$	BRGE	$Rd < Rr$	$(N \oplus V) = 1$	BRLT	Signed
$Rd = Rr$	$Z = 1$	BREQ	$Rd \leftarrow Rr$	$Z = 0$	BRNE	Signed
$Rd \leq Rr$	$Z \vee (N \oplus V) = 1$	BRGE ⁽¹⁾	$Rd > Rr$	$Z \wedge (N \oplus V) = 0$	BRLT*	Signed
$Rd < Rr$	$(N \oplus V) = 1$	BRLT	$Rd \leftarrow Rr$	$(N \oplus V) = 0$	BRGE	Signed
$Rd > Rr$	$C + Z = 0$	BRLO ⁽¹⁾	$Rd \leftarrow Rr$	$C + Z = 1$	BRSH*	Unsigned
$Rd \geq Rr$	$C = 0$	BRSH/BRCC	$Rd < Rr$	$C = 1$	BRLO/BRCS	Unsigned
$Rd = Rr$	$Z = 1$	BREQ	$Rd \leftarrow Rr$	$Z = 0$	BRNE	Unsigned
$Rd \leq Rr$	$C + Z = 1$	BRSH ⁽¹⁾	$Rd > Rr$	$C + Z = 0$	BRLO*	Unsigned
$Rd < Rr$	$C = 1$	BRLO/BRCS	$Rd \leftarrow Rr$	$C = 0$	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 \leftarrow 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	$Rd \leftarrow Rd + Rr$	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,S,H	1
ADIW	Rd, K	Add Immediate to Word	$Rd+1:Rd \leftarrow Rd+1:Rd + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract without Carry	$Rd \leftarrow Rd - Rr$	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	$Rd \leftarrow Rd - K$	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	$Rd \leftarrow Rd - K - C$	Z,C,N,V,S,H	1

SBRW	Rd, K	Subtract Immediate from Word	$Rd+1:Rd \leftarrow Rd+1:Rd - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND	$Rd \leftarrow Rd \cdot Rr$	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	$Rd \leftarrow Rd \cdot K$	Z,N,V,S	1
OR	Rd, Rr	Logical OR	$Rd \leftarrow Rd \vee Rr$	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	$Rd \leftarrow Rd \vee K$	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	$Rd \leftarrow Rd \oplus Rr$	Z,N,V,S	1
COM	Rd	One's Complement	$Rd \leftarrow \text{SFF} - Rd$	Z,C,N,V,S	1
NEG	Rd	Two's Complement	$Rd \leftarrow \$00 - Rd$	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \cdot (\text{SFFh} - K)$	Z,N,V,S	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V,S	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \cdot Rd$	Z,N,V,S	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V,S	1
SER	Rd	Set Register	$Rd \leftarrow \text{SFF}$	None	1
MUL	Rd,Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr \text{ (UU)}$	Z,C	2
MULS	Rd,Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr \text{ (SS)}$	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr \text{ (SU)}$	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1 \text{ (UU)}$	Z,C	2
FMULS	Rd,Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \ll 1 \text{ (SS)}$	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1 \text{ (SU)}$	Z,C	2

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
Branch Instructions					
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP		Indirect Jump to (Z)	$PC(15:0) \leftarrow Z$	None	2
EIJMP		Extended Indirect Jump to(Z)	$PC(15:0) \leftarrow Z$	None	2
JMP	k	Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Call Subroutine	$PC \leftarrow PC + k + 1$	None	3 / 4
ICALL		Indirect Call to (Z)	$PC(15:0) \leftarrow Z$	None	3 / 4
EICALL		Extended Indirect Call to (Z)	$PC(15:0) \leftarrow Z$	None	4
CALL	k	Call Subroutine	$PC \leftarrow k$	None	4 / 5
RET		Subroutine Return	$PC \leftarrow \text{STACK}$	None	4 / 5
RETI		Interrupt Return	$PC \leftarrow \text{STACK}$	1	4 / 5
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) $PC \leftarrow PC + 2 \text{ or } 3$	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) $PC \leftarrow PC + 2 \text{ or } 3$	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b)=1) $PC \leftarrow PC + 2 \text{ or } 3$	None	1 / 2 / 3
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b)=0) $PC \leftarrow PC + 2 \text{ or } 3$	None	1 / 2 / 3
SBIS	A, b	Skip if Bit in I/O Register Set	if (I/O(A,b)=1) $PC \leftarrow PC + 2 \text{ or } 3$	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then $PC \leftarrow PC + k + 1$	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then $PC \leftarrow PC + k + 1$	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then $PC \leftarrow PC + k + 1$	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then $PC \leftarrow PC + k + 1$	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then $PC \leftarrow PC + k + 1$	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then $PC \leftarrow PC + k + 1$	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then $PC \leftarrow PC + k + 1$	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then $PC \leftarrow PC + k + 1$	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then $PC \leftarrow PC + k + 1$	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then $PC \leftarrow PC + k + 1$	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N \oplus V = 0) then $PC \leftarrow PC + k + 1$	None	1 / 2

BRLT	k	Branch if Less Than, Signed	if $(N \oplus V = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if $(H = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if $(H = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRTS	k	Branch if T Flag Set	if $(T = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if $(T = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if $(V = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if $(V = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if $(I = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if $(I = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2

Data Transfer Instructions

MOV	Rd, Rr	Copy Register	$Rd \leftarrow Rr$	None	1
MOVW	Rd, Rr	Copy Register Pair	$Rd+1:Rd \leftarrow Rr+1:Rr$	None	1
LDI	Rd, K	Load Immediate	$Rd \leftarrow K$	None	1
LDS	Rd, k	Load Direct from data space	$Rd \leftarrow (k)$	None	2
LD	Rd, X	Load Indirect	$Rd \leftarrow (X)$	None	2
LD	Rd, X+	Load Indirect and Post-Increment	$Rd \leftarrow (X), X \leftarrow X + 1$	None	2
LD	Rd, -X	Load Indirect and Pre-Decrement	$X \leftarrow X - 1, Rd \leftarrow (X)$	None	2
LD	Rd, Y	Load Indirect	$Rd \leftarrow (Y)$	None	2
LD	Rd, Y+	Load Indirect and Post-Increment	$Rd \leftarrow (Y), Y \leftarrow Y + 1$	None	2
LD	Rd, -Y	Load Indirect and Pre-Decrement	$Y \leftarrow Y - 1, Rd \leftarrow (Y)$	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	$Rd \leftarrow (Y + q)$	None	2
LD	Rd, Z	Load Indirect	$Rd \leftarrow (Z)$	None	2
LD	Rd, Z+	Load Indirect and Post-Increment	$Rd \leftarrow (Z), Z \leftarrow Z + 1$	None	2
LD	Rd, -Z	Load Indirect and Pre-Decrement	$Z \leftarrow Z - 1, Rd \leftarrow (Z)$	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	$Rd \leftarrow (Z + q)$	None	2
STS	k, Rr	Store Direct to data space	$Rd \leftarrow (k)$	None	2
ST	X, Rr	Store Indirect	$(X) \leftarrow Rr$	None	2
ST	X+, Rr	Store Indirect and Post-Increment	$(X) \leftarrow Rr, X \leftarrow X + 1$	None	2
ST	-X, Rr	Store Indirect and Pre-Decrement	$X \leftarrow X - 1, (X) \leftarrow Rr$	None	2
ST	Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	None	2
ST	Y+, Rr	Store Indirect and Post-Increment	$(Y) \leftarrow Rr, Y \leftarrow Y + 1$	None	2
ST	-Y, Rr	Store Indirect and Pre-Decrement	$Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2
STD	Y+q, Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	None	2
ST	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	2
ST	Z+, Rr	Store Indirect and Post-Increment	$(Z) \leftarrow Rr, Z \leftarrow Z + 1$	None	2

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
ST	-Z, Rr	Store Indirect and Pre-Decrement	$Z \leftarrow Z - 1, (Z) \leftarrow Rr$	None	2
STD	Z+q, Rr	Store Indirect with Displacement	$(Z + q) \leftarrow Rr$	None	2
LPM		Load Program Memory	$R0 \leftarrow (Z)$	None	3
LPM	Rd, Z	Load Program Memory	$Rd \leftarrow (Z)$	None	3
LPM	Rd, Z+	Load Program Memory and Post-Increment	$Rd \leftarrow (Z), Z \leftarrow Z + 1$	None	3
ELPM		Extended Load Program Memory	$R0 \leftarrow (RAMPZ:Z)$	None	3
ELPM	Rd, Z	Extended Load Program Memory	$Rd \leftarrow (RAMPZ:Z)$	None	3
ELPM	Rd, Z+	Extended Load Program Memory and Post-Increment	$Rd \leftarrow (RAMPZ:Z), Z \leftarrow Z + 1$	None	3
SPM		Store Program Memory	$(Z) \leftarrow R1:R0$	None	-
ESPM		Extended Store Program Memory	$(RAMPZ:Z) \leftarrow R1:R0$	None	-
IN	Rd, A	In From I/O Location	$Rd \leftarrow I/O(A)$	None	1
OUT	A, Rr	Out To I/O Location	$I/O(A) \leftarrow Rr$	None	1
PUSH	Rr	Push Register on Stack	$STACK \leftarrow Rr$	None	2
POP	Rd	Pop Register from Stack	$Rd \leftarrow STACK$	None	2

Bit and Bit-test Instructions

LSL	Rd	Logical Shift Left	$Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0, C \leftarrow Rd(7)$	Z, C, N, V, H	1
LSR	Rd	Logical Shift Right	$Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0, C \leftarrow Rd(0)$	Z, C, N, V	1
ROL	Rd	Rotate Left Through Carry	$Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)$	Z, C, N, V, H	1
ROR	Rd	Rotate Right Through Carry	$Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$	Z, C, N, V	1
ASR	Rd	Arithmetic Shift Right	$Rd(n) \leftarrow Rd(n+1), n=0..6$	Z, C, N, V	1
SWAP	Rd	Swap Nibbles	$Rd(3..0) \leftarrow Rd(7..4)$	None	1

BSET	s	Flag Set	$SREG(s) \leftarrow 1$	$SREG(s)$	1
BCLR	s	Flag Clear	$SREG(s) \leftarrow 0$	$SREG(s)$	1
SBI	A, b	Set Bit in I/O Register	$I/O(A, b) \leftarrow 1$	None	2
CBI	A, b	Clear Bit in I/O Register	$I/O(A, b) \leftarrow 0$	None	2
BST	Rr, b	Bit Store from Register to T	$T \leftarrow Rr(b)$	T	1
BLD	Rd, b	Bit load from T to Register	$Rd(b) \leftarrow T$	None	1
SEC		Set Carry	$C \leftarrow 1$	C	1
CLC		Clear Carry	$C \leftarrow 0$	C	1
SEN		Set Negative Flag	$N \leftarrow 1$	N	1
CLN		Clear Negative Flag	$N \leftarrow 0$	N	1
SEZ		Set Zero Flag	$Z \leftarrow 1$	Z	1
CLZ		Clear Zero Flag	$Z \leftarrow 0$	Z	1
SEI		Global Interrupt Enable	$I \leftarrow 1$	I	1
CLI		Global Interrupt Disable	$I \leftarrow 0$	I	1
SES		Set Signed Test Flag	$S \leftarrow 1$	S	1
CLS		Clear Signed Test Flag	$S \leftarrow 0$	S	1
SEV		Set Two's Complement Overflow	$V \leftarrow 1$	V	1
CLV		Clear Two's Complement Overflow	$V \leftarrow 0$	V	1
SET		Set T in SREG	$T \leftarrow 1$	T	1
CLT		Clear T in SREG	$T \leftarrow 0$	T	1
SEH		Set Half Carry Flag in SREG	$H \leftarrow 1$	H	1
CLH		Clear Half Carry Flag in SREG	$H \leftarrow 0$	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

۳۱۳ خطاهای میکروکنترلرهای AVR

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.

LOWER	FOR	CALL	^
LOWERCASE	FOURTH	CAPTURE1	!
MAKEBCD()	FOURTHLINE	CASE	:
MAKEDEC()	FUNCTION	CHR()	\$BAUD
MAKEINT()		CLS	\$CRYSTAL
MID()	GATE	CLOSE	\$DATA
MOD	GETAD()		
MODE	GETRC5()	Compare1a	\$DEFAULT
	GOSUB	COMPARE1B	\$END
	GOTO		\$EEPROM
NACK		Config	\$EXTERNAL
NEXT	HEXVAL()	CONST	\$INCLUDE
NOBLINK	HIGH()	COUNTER	\$LCD
NOSAVE	HOME	COUNTER0	\$LCDRS
NOT		COUNTER1	\$LCDPUTCTRL
	I2CRECEIVE	COUNTER2	\$LCDPUTDATA
OFF	I2CSEND	CPEEK()	\$LIB
ON	I2CSTART	CPEEKH()	\$REGFILE
OR	I2CSTOP	CRYSTAL	\$SERIALINPUT
OUT	I2CRBYTE	CURSOR	\$SERIALINPUT2LC
OUTPUT	I2CWBYTE		D
	IDLE	DATA	
PEEK()		DATES	
POKE	If	DEBOUNCE	
PORTA	INCR	DECR	\$SERIALOUTPUT
PORTB	INKEY	DECLARE	\$WAITSTATE
PORTC	INP()	DEFBIT	\$XRAMSIZE
PORTD	INPUT	DEFBYTE	\$XRAMSTART
PORTE	INPUTBIN	DEFLNG	
PORTF	INPUTHEX	DEFWORD	
POWERDOWN	INT0	DEGSNG	\$WRESET
PRINT	INT1	DEFLCDCHAR	\$WREAD
PRINTBIN	INTEGER	DEFINT	\$WWRITE
	INTERNAL	DEFWORD	
Pulseout	INSTR	DELAY	ACK
PWM1A	IS	DIM	ABS()
PWM1B		DISABLE	ALIAS
	LCASE()	DISPLAY	AND
READ	LCD	DO	AS
READEEPROM	LEFT	DOWNT0	ASC()
REM	LEFT()		AT
RESET	LEN()	ELSE	
RESTORE	LOAD	ELSEIF	BAUD
RETURN	LOCAL	ENABLE	BCD()
RIGHT	LOCATE	END	BIT
RIGHT()	LONG	ERAM	BITWAIT
ROTATE	LOOKUP()	ERASE	BLINK
RTRIM()	LOOKUPSTR()	ERR	BOOLEAN
	LOOP	EXIT	BYTE
SELECT	LTRIM()	EXTERNAL	BYVAL
SERIAL	LOW()		

WAIT	TIMER1	STEP	SET
WAITKEY()	TIMER2	STR()	SHIFT
WAITMS	TO	STRING()	SHIFTLCD
WAITUS	TRIM()	STOP	SHIFTCURSOR
WATCHDOG		STOP TIMER	SHIFTIN
WRITEEEPROM	UCASE()	SUB	SHIFTOUT
WEND	UNTIL	SWAP	SOUND
WHILE	UPPER		SPACE()
WORD	UPPERLINE	THEN	SPIINIT
		TIMES	SPIIN
XOR	VAL()	THIRD	SPIMOVE
	VARPTR()	THIRDLIN	SPIOUT
XRAM		TIMER0	START

Error Codes In bascom AVR

The following table lists errors that can occur.

Error	Description	Error	Description
40	Variable can not be used with RESET	1	Unknown statement
41	Variable can not be used with SET	2	Unknown structure EXIT statement
42	Numeric parameter expected	3	WHILE expected
43	File not found	4	No more space for IRAM BIT
44	2 variables expected	5	No more space for BIT
45	DO expected	6	. expected in filename
46	Assignment error	7	IF THEN expected
47	UNTIL expected	8	BASIC source file not found
50	Value doesn't fit into INTEGER	9	Maximum 128 aliases allowed
51	Value doesn't fit into WORD	10	Unknown LCD type
52	Value doesn't fit into LONG	11	INPUT, OUTPUT, 0 or 1 expected
60	Duplicate label	12	Unknown CONFIG parameter
61	Label not found	13	CONST already specified
62	SUB or FUNCTION expected first	14	Only IRAM bytes supported
63	Integer or Long expected for ABS()	15	Wrong data type
64	, expected	16	Unknown Definition
65	device was not OPEN	17	9 parameters expected
66	device already OPENED	18	BIT only allowed with IRAM or SRAM
68	channel expected	19	STRING length expected (DIM S AS STRING * 12 for example)
70	BAUD rate not possible	20	Unknown DATA TYPE
71	Different parameter type passed then declared	21	Out of IRAM space
72	Getclass error. This is an internal error.	22	Out of SRAM space
73	Printing this FUNCTION not yet supported	23	Out of XRAM space
74	3 parameters expected	24	Out of EPROM space
80	Code does not fit into target chip	25	Variable already dimensioned
81	Use HEX(var) instead of PRINTHEX	26	AS expected
82	Use HEX(var) instead of LCDHEX	27	parameter expected
85	Unknown interrupt source	28	IF THEN expected
86	Invalid parameter for TIMER configuration	29	SELECT CASE expected
87	ALIAS already used	30	BIT's are GLOBAL and can not be erased
88	0 or 1 expected	31	Invalid data type
89	Out of range : must be 1-4	32	Variable not dimensioned
90	Address out of bounds	33	GLOBAL variable can not be ERASED
91	INPUT, OUTPUT, BINARY, or RANDOM expected	34	Invalid number of parameters
92	LEFT or RIGHT expected	35	3 parameters expected
93	Variable not dimensioned	36	THEN expected
94	Too many bits specified	37	Invalid comparison operator
95	Falling or rising expected for edge	38	Operation not possible on BITS
		39	FOR expected

۳۱۵ خطاهای میکروکنترلرهای AVR

Error Description

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
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
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
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
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

Error Description

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
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

چند راهنمایی مهم

- ۱- از پورت میکروهای که پایه های آن برای ارتباط با پروتکل JTAG به کار برده میشوند، نمی توان استفاده کرد.
 - فیوز بینهای مربوط به ارتباط JTAG را توسط programmer محیط Bascom را طبق زیر برنامه ریزی کنید:
 - فیوز بیت JTAG ENABLE را به گزینه JTAG DISABLE تغییر دهید.
 - فیوز بیت OCD ENABLE را به گزینه OCD DISABLE تغییر دهید.
- ۲- پایه خروجی پالس PWM خروجی نمی دهد.
 - پیکره بندی تایمر/کانتر مربوطه درست انجام نشده است.
 - پایه خروجی پالس PWM به عنوان خروجی تعریف نشده و بایستی خروجی تعریف شود.
- ۳- مفادیر ارسالی به پورت سریال و یا ارتباط سریال مشکل دارد.
 - نرخ باود در هر دو وسیله یکسان بایستی باشد.
 - مقدار فرکانس کریستال میکرو (انواع مختلف کلاک) توسط دستور \$CRYSTAL درست تعریف و مشخص شده باشد.
 - دقت کنید که جهت پایه های مربوط به ارتباط سریال را در جایی از برنامه به جهت نامناسب تغییر نداده باشید.
- ۴- دستورات بیتی یا بایستی بر روی پایه یا پورت کار نمی کنند.
 - دقت شود که با توجه به کاربرد پایه یا پورت، درست خروجی یا ورودی تعریف شده باشند.
- ۵- PROGRAMMER محیط Bascom میکرو را شناسایی نمی کند.
 - VCC مدار PROGRAMMER و میکرو بایستی برقرار باشد.
 - CRYSTAL برای میکروهای که با اسپلاتور داخلی آن کار می کنند بین دو پایه XTAL1,2 قرار گرفته باشد.
 - دقت کنید در منوی OPTION، گزینه PROGRAMMER از محیط BASCOM نوع STK200/300 برای برنامه ریز تعیین شده باشد.
 - بایستی آدرس پورت موازی در منوی OPTION، گزینه PROGRAMMER درست انتخاب شده باشد.
 - ممکن است پورت LPT سالم نباشد.
- ۶- دستورات تأخیری (WAIT, WAITms, WAITus) بیشتر یا کمتر از مقدار تعریف شده تأخیر ایجاد می کنند.
 - علت آن یکی نبودن فرکانس کریستال برنامه و فرکانس کاری میکرو است. در این حالت دقت کنید که میکرو با فرکانس کریستال برابری با فرکانس تعریف شده برنامه کار می کند.
- ۷- ارتباط SPI بین دو میکرو کار نمی کند.
 - دقت کنید که در این ارتباط بایستی یکی از میکروها MASTER و دیگری SLAVE باشد.
 - DDRX (X پورت مربوط به ارتباط SPI) درست پیکره بندی شده باشند.
 - گزینه های polarity, phase و clockrate در هر دو میکرو بایستی یکسان باشند.
- ۸- میکرو ری ست میشود.
 - احتمال وجود نویز
 - ممکن است فیوز بیت WDTON برنامه ریزی شده باشد و مدام میکرو ری ست شود.
 - WATCHDOG با توجه به برنامه نوشته شده با مقدار نامناسب پیکره بندی شده است و یا اینکه دستور RESET WATCHDOG در جاهای مناسبی از برنامه استفاده نشده است.
- ۹- چگونه از تایمر/کانتر ۳ (که فقط در میکروهای ATMEGA64, ATMEGA128 و ATMEGA162 موجود است) استفاده نمایم.
 - شما می توانید تایمر/کانتر ۳ را همانند تایمر/کانتر ۱ با تغییرات زیر در محیط Bascom پیکره بندی کنید:

پیکره بندی تایمر/کانتر سه در محیط BASCOM

پیکره بندی تایمر/کانتر سه در حالت تایمر

Config Timer3 = Timer, PRESCALE = 1[8]64[256]1024

پیوست ۳۱۷

تایمر UP = COUNTER سه در مُد TIMER به کار برده شده و می‌تواند فرکانس کلاک خود را از فرکانس اسیلاتور بخش بر ۱، ۸، ۶۴، ۲۵۶، ۱۰۲۴ تا ۱۰۰۰۰۰۰۰ (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 LABEL یا ON TIMER3 LABEL به LABEL پرش کرد و ISR سرریزی را اجرا کرد. با دستور VAR = TIMER3 می‌توان محتوای تایمر/کانتر ۳ خواند که VAR متغیری 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 تایمر

```
Sregfile = "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 OvfIroutin
Start Timer3
Do
Print Timer3
Loop
```

**** T/C3 OVER FLOW INTERRUPT SERVICE ROUTIN****

```
OvfIroutin:
Print "OVERFLOW OCCURES"
Return
```

پیکره‌بندی تایمر/کانتر سه در حالت کانتر

```
Config Timer3 = Counter , Edge = Rising | Falling , Prescale = 1|8|64|256|1024
```

یا می‌توان چنین نوشت :

```
Config Timer3 = Counter , Edge = Rising | Falling
```

تایمر/کانتر ۳ 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 متغیری از نوع داده WORD است. کانتر بعد از شمردن تعداد ۱ + 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 دستور ON OVF3 LABEL یا ON COUNTER3 LABEL به LABEL پرش کرد و ISR سرریزی را اجرا کرد.

• مثال

پس از شمردن شدن ۱۰ 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 بیتی A، B و C (که bascom حمایت نمی‌کند) است 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 می‌تواند 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 متغیر 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 متغیر 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) و یا ارتباط پایه با کانتر قطع (DISCONNECT) شود.
Clear Timer = 1j0 : با انتخاب گزینه 1 ، محتوای تایمر/کانتر سه در زمان تطابق مقایسه‌ای ریست و یا به عبارتی S0000 خواهد شد.

تایمر سه و مُد مقایسه‌ای

Config Timer3 = Timer , Compare A = Clear | Set | Toggle | Disconnect , Compare B = Clear | Set | Toggle | Disconnect , Prescale = 1j8j64j256j1024

طبق این پیکره‌بندی تایمر/کانتر در حالت تایمر استفاده می‌شود و می‌تواند فرکانس کلاک خود را از فرکانس اسلایاتور بخش بر 1 ، 8 ، 64 ، 256 ، 1024 تأمین کند. تایمر با فرکانس تعیین شده همانطور که در قسمت پیکره‌بندی 3 TIMER / COUNTER در حالت TIMER 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 می‌تواند 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 متغیر نوع WORD ، BYTE ، INTEGER یا مقادیر مثبت باشد. از این رجیستر می‌توان با دستور VAR = COMPARE3A | B (یا VAR = COMPARE3A | B) خواند که VAR متغیر 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 = 1j0 : 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 روئین وقفه تطابق مقایسه‌ای A | B از دستور ON OC3A|B LABEL استفاده می‌کنیم. زمانی که محتوای رجیسترهای مقایسه‌ای A یا B با محتوای تایمر یا کانتر سه برابر شود، زیر برنامه وقفه LABEL 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 OCCURS"
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 بایستی CAPTURE3 جای می‌گیرد و پرچم وقفه CAPTURE یک می‌شود و در صورت فعال بودن وقفه مربوطه، زیر برنامه وقفه اجرا می‌شود.

کانتر سه و مُد CAPTURE

Config Timer3 = Counter , Edge = Falling | Rising , Capture Edge = Falling | Rising , Noise Cancel = 1 | 0 , Prescale = 1j8j64j256j1024

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 CAPTURE3 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 می‌کنند. اعمال یک لبه پایین‌رونده به پایه IC3 باعث می‌شود که محتوای تایمر/ کانتر در همان لحظه در رجیستر CAPTURE3 قرار گیرد. محتوای رجیستر CAPTURE3 را می‌توان با دستور VAR = CAPTURE3 خواند و با دستور CAPTURE3 = VAR می‌توان در این رجیستر نوشت که VAR ثابت یا متغیری دو بیتی است.

طرز کار با وقفه CAPTURE

در صورت اعمال پالس مطلوب به پایه IC3 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 LABEL در زمان رخ داد CAPTURE به زیر برنامه وقفه LABEL پرش و ISR مربوطه را اجرا کرد.

• مثال

```
Sregfile = "M64DEF.DAT"
$CRYSTAL=8000000
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 مولاسیون 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 SFF و S1FF 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 متغیری 1 یا 2 بیتی باشد.

فرکانس PWM با توجه به معادله‌های زیر بدست می‌آید که فرکانس کلایک سیستم است :

$$\text{PWM FREQUENCY} = \text{Fosc} / (510 * \text{Prescale}) \quad \text{PWM , 8 بیتی}$$

$$\text{PWM FREQUENCY} = \text{Fosc} / (1022 * \text{Prescale}) \quad \text{PWM , 9 بیتی}$$

$$\text{PWM FREQUENCY} = \text{Fosc} / (2046 * \text{Prescale}) \quad \text{PWM , 10 بیتی}$$

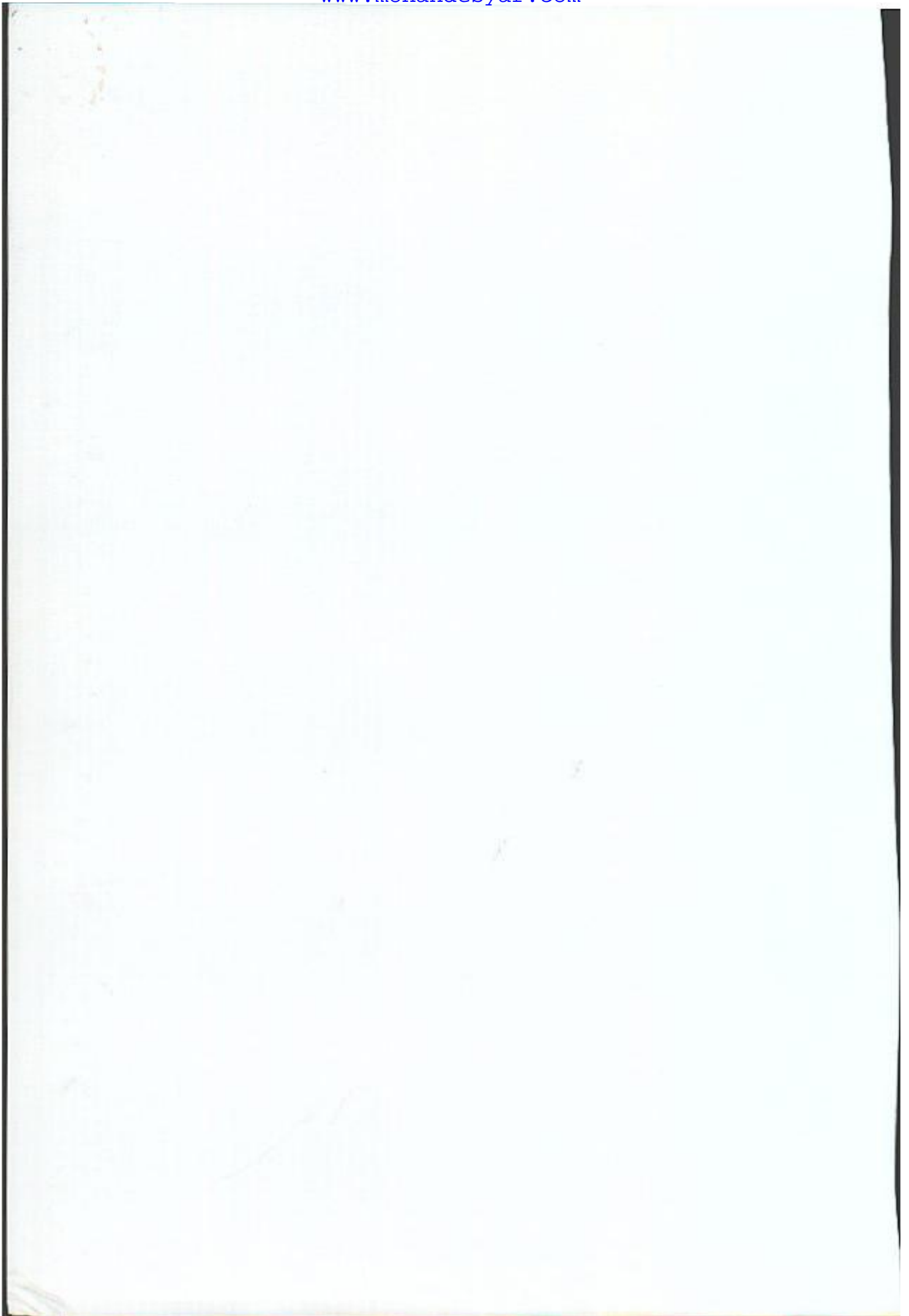
• مثال

فرکانس پالس PWM تولید شده برای خروجی A و B در مثال زیر $488.75851\text{Hz} = 8000000/8*2046$ است.

```
Sregfile = "M128DEF.DAT"
$CRYSTAL = 8000000
Config Timer3 = Pwm , Pwm = 10 , Compare A Pwm = Clear Up , Compare B Pwm = Clear Down ,
Prescale = 8
COMPIG PORTE = OUTPUT
Do
Compare3a = 100
Compare3b = 200
Loop
End
```


غلطنامه

صفحه	غلط	صحیح
۱۲۹ (شکل ۶-۴)		
۱۳۵ (بالای صفحه)	میگیرد	برچسب
۱۶۲ (دستور gosub)	برچسبی	میگیرد
۱۶۹ (توضیح جدول بالای صفحه)	جدول تاثیر تغییرات DDAn بر روی پایه های PORTA	جدول تاثیر تغییرات DDBn بر روی پایه های PORTB
۱۷۲ (توضیح جدول بالای صفحه)	جدول تاثیر تغییرات DDAn بر روی پایه های PORTA	جدول تاثیر تغییرات DDCn بر روی پایه های PORTC
۱۸۴ (بخش ۵-۶)	AVR ها بجز MEGA128 که چهار تایمر دارد	AVR ها بجز MEGA128 و MEGA64 که چهار تایمر دارند
۱۹۷	رجیستر CAPTURE	رجیستر CAPTURE به CAPTUREI تغییر نام داده شود.
صفحات ۲۰۲ تا ۲۰۸	تمام متغیرهایی که با رجیسترهای تایمر/کانتر ۲ ارتباط دارند از نوع word تعریف شده اند.	تمام متغیرهایی که با رجیسترهای تایمر/کانتر ۲ ارتباط دارند علاوه بر اینکه میتوانند از نوع word باشند به علت هشت بیتی بودن تایمر از نوع byte نیز تعریف می شوند.
۲۰۳ (مثال تایمر)	Config timer2=timer,prescale=128	Config timer2=timer,prescale=1
۲۰۴ (تایمر دو و مد مقایسه ای)	Config timer1=timer,compare=...	Config timer2=timer,compare=...
۲۲۲ (شکل ۲-۶)	L=10mH	L=10uH
۲۴۹ (خط چهارم دستور pulsein)	خوی	خود
۲۵۰ (خط هفتم دستور sound)	فرکانس هاس	فرکانس های
۲۵۵ (پاراگراف دوم از پایین صفحه)	مورد استفاده می شود.	مورد استفاده قرار می گیرد.



میکرو کنترلرهای

AVR

عناوین اصلی کتاب عبارتند از:

- معرفی انواع میکرو کنترلرهای AVR شامل AT90S، TINYAVR و MEGA AVR به همراه فیوز بیت ها و کلاک سیستم هر یک
- آشنایی و معرفی محیط کامپایلر BASCOM برای برنامه نویسی و برنامه ریزی میکروهای AVR به همراه مثال های متعدد
- دستورات و نحوه استفاده از امکانات AVR در محیط BASCOM با مثال های متعدد
- کار با حافظه های EEPROM سریال 2-Wire
- ۱۸ پروژه عملی در انتهای فصل کتاب
- خطاهای میکرو کنترلر AVR و نحوه برطرف کردن آنها

CD همراه شامل



نرم افزار 1.11.7.4 BASCOM، CodeVision، FastAVR

Proteus 6.0 و برگه های اطلاعاتی تمام میکروهای AVR



تهران - میدان انقلاب خ اردیبهشت پست مین شماره ۲۲۷
تلفن: ۶۹۵۳۸۸۳ تلفن فکس: ۶۴۱۲۳۸۵ ص.ب. ۱۳۱۴۵-۸۶۳