فهرست مطالب

فصل اول- توصیف کامل پروژه

۱-۱- ضرورت ها و نیازها

۱-۱-۲- مراحل انجام پروژه

فصل دوم تئوری مرتبط با پروژه

۲-۱- مقدمه

۲-۲- بررسی دستورات میکرو کنترلر

فصل سوم- بررسی سخت افزار سیستم

۳-۱- مقدمه

۳-۲- بلوک دیاگرام سیستم

۳-۳- توضیح بلوک دیاگرام سیستم

۳-٤-نقشه کامل مدار توضیحات مربوطه

فصل چهار - نرم افزار

٤-۱- مقدمه:بررسی نرم افزارمورداستفاده وکامپایلربکاررفته

٤-۲فلوچارت برنامه اصلی و فرعی

٤-۳- توضیحات نرم افزار

فصل پنجم - نکات پایانی

۵-۱- ضمایم

۵-۲- مراجع

فصل ۱ - توصیف کامل پروژه

۱-۱- ضرورت ها و نیاز ها :

در بسیاری از مدارات دیجیتالی (میکروکنترلی) که با برق و باطری کار می کنند با دو مشکل مواجه می باشیم :

۱- مشکل اول مربوط به عملکرد است. یعنی عملکرد نمایش LCD با ولتاژ ۵ ولت است در حالی که این مدارات در هنگام استفاده از خازن پشتیبان و باطری دارای ولتاژ ۳/٦ تا ۵ ولت هستند.

۲- مشکل دوم گرانی باطری و دشوار بودن تعویض آن است . در بعضی از این مدارات مثل کنتور دیجیتال تعویض باطری بسیار دشوار است در نتیجه باید تا حد امکان مصرف را در حالت استفاده از خازن و باطری کم کرد در حالی که دستگاه باید به کار خود ادامه دهد.

۱-۲- مراحل انجام پروژه :

در این پروژه جهت بررسی و حل این دو مشکل یک مدار میکرو کنترلی مثل ساعت دیجیتالی طراحی شده است و به حل مشکلات فوق پرداخته شده است این ساعت باید از نمایشگر ۵ ولت استفاده کند و بتواند حداکثر به مدت ۵۰۰۰ ساعت در صورت استفاده از باطری ۳/٦ ولتی و ۱/۲ آمپر ساعتی در حالت قطع برق به کار خود ادامه دهد.

در طراحی این مدار برای حل مشکل اول سعی شده است که برای تامین ولتاژ ۵ ولت LCD از تبدیل ولتاژ DC به کمک یک مبدل DC/DC استفاده کنیم تا زمانی که ولتاژ ما ۳/٦ ولت است با یک آ ی سی Admbbo ولتاژ ۷/۲ تولید کنیم (در ادامه خواهیم دید این آ ی سی یک دوبل کننده ولتاژ است) و بعد باکمک زنریک ولتاژ ۵ ولت تحویل LCD می دهیم و برای حل مشکل دوم سعی شده است که تا زمانی که LCD از تغذیه اصلی استفاده نمیکند از خازن پشتیبان تغذیه شود و بعد از مدت زمانی که تعریف می کنیم (در نرم افزار سیستم بررسی می شود) میکرو از طریق یک Mosfet از باطری استفاده میکند که توسط مبدل DC/DC و زنر به ولتاژ ۵ ولت تبدیل شده است.

فصل ۲ - تئوری ای مرتبط با پروژه

۲-۱- مقدمه :

تئوری هایی که در مورد پروژه وجود دارد به چندین بخش تقسیم می شوند یکی از این بخشها دستورات میکروکنترلر AVR است که بطور نسبتاً مشروح بیان شده اند البته در این قسمت توجه شود که از مبانی مربوط به تایمرها ، وقفه ها ، کار با پورتها و ... بیشتر استفاده شده است و برخی مسائل نیز جهت آشنایی کامل تر خواننده با این میکروکنترلر آورده شده است. در مورد تئوری های مربوط به نرم افزار سیستم نیز مطالبی به طور کلی در این فصل آورده شده است اما توضیحات دقیق تر مربوط به نرم افزار (صرف نظر از مطالب کلی) در جای خود در فصل های آینده بررسی شده است . خلاصه

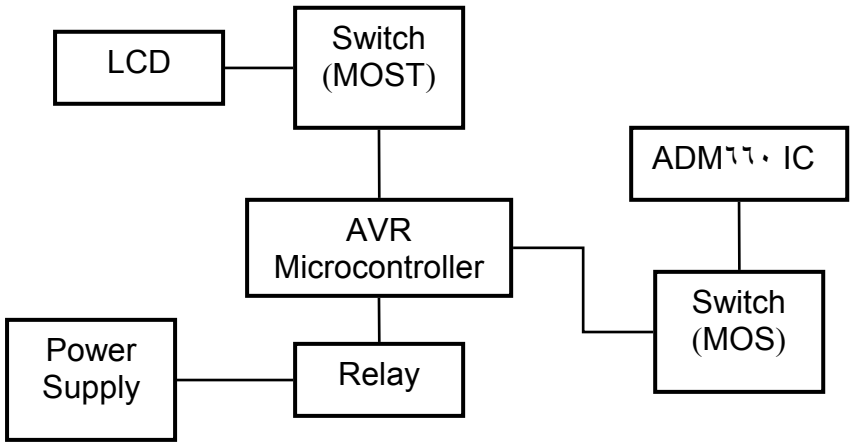
مطلب آنکه نکات تئوری مطرح شده در این فصل کلیات تئوری سیستم هستند و نکات ریز در فصل های مربوط به خود بحث شده اند.

فصل ۳- بررسی سخت افزار سیستم

۳-۱-مقدمه :

در این قسمت ابتدا یک بلوک دیاگرام کلی از سیستم ارائه می دهیم که در آن سخت افزار سیستم به طور کلی و به صورت بلوکی رسم شده است بعد از رسم این بلوک و توضیحات مربوط به آن نقشه کامل مدار و توضیحات مربوط به نقشه به طور کامل بیان شده است. در آن قسمت برخی جزئیات که در بلوک دیاگرام قابل مشاهده نمی باشد به طور کامل مورد بررسی قرار می گیرد . در این قسمت توضیحات مربوط شامل نیازهای مدار و دلایل به کار بردن هر قطعه می باشد . البته در لابلای توضیحات سخت افزار نکات دیگری که در هنگام بستن مدار مطرح شده بررسی می شود مثلا استفاده از زنر برای ساختن رگولاتور به جای آی سی رگولاتور و یا مثلا استفاده از یک مقاومت به خصوصی در مسیر مدار رگولاتور و مواردی از این قبیل که تا اندازه ای که باعث طولانی شدن مطلب نشود بحث و بررسی می شود.

۳-۲- بلوک دیاگرام سیستم :



۳-۳-توضیحات مربوط به دیاگرام :

۳-۳-۱- تغذیه :

قسمت تغذیه به منظور فراهم کردن ولتاژ۵ ولت DC می باشد که جهت راه اندازی IC میکروکنترلر و کلیه مدارات سازگار با TTL به کار کمی رود . ورودی قسمت تغذیه می تواند یک سیگنال AC یا DC باشد که الزاما به اندازه چند ولت از ۵+ بیشتر است. این ورودی توسط یک سوییچ ON-OFFقطع و وصل می شود. در حالتیکه سوییچ روشن است ، خازن مربوطه به علت ظرفیت بالایش باعث می شود که ولتاژی تقریبا صاف بدست آید که دارای اندکی رایپل است.برای ایجاد یک ولتاژ ۵ ولت کاملا DC از یک IC تنظیم کننده ولتاژ ۷۸۰S استفاده می کنیم که ورودی دارای اعواجاج را به یک ولتاژ کاملا مستقیم پنج ولت تبدیل می کند در خروجی تنظیم کننده ولتاژ یک خازن ۱۰ میکروفاراد قرار داده ایم که این خازن به اندازه ۵ ولت شارژ خواهد شد که نوسانات خروجی ناشی از تغییرات بار را کاهش می دهد.

۳-۳-۲- رله :

رله به کار رفته در مدار فقط برای این است که چک کنیم تغذیه است یا خیر هرگاه رله وصل باشد یعنی تغذیه وصل است و هرگاه قطع باشد یعنی تغذیه قطع است . اینجا ذکر یک نکته در مورد رله لازم استو آن اینکه رله چون جریان را از آداپتور می گیرد باعث نمی شود که بخاطر زیاد شدن مصرف جریان کاهش پیدا کند.

۳-۳-۳- میکروکنترلر AVR :

در مورد این بلوک مطالب فراوانی را می توان مورد بررسی قرار داد . در اینجا به ذکر برخی خصوصیات آن می پردازیم . لازم به ذکر است مطالب دقیق تر در قسمت تئوری های مربوط با پروژه در فصل ۲ بررسی شده است .

الف - استفاده از معماری AVR Risc

- کارایی بالا و توان مصرفی کم
- دارای ۱۳۱ دستور العمل که اکثرا در یک کلاک سیکل اجرا می شوند.
- ۸× ۳۲ رجیستر کاربردی
- سرعتی تا mps ۱٦ در فرکانس mhz ۱٦

ب- حافظه، برنامه و داده غیر فرار

- KB ۱٦ حافظه FLASH قابل برنامه ریزی داخلی
- ۱۲۰٤ بایت حافظه داخلی SRAM
- ۵۱۲ بایت حافظه EEPROM
- قفل برنامه FLASH و حفاظت داده EEPROM

ج - خصوصیات جانبی

- دو تایمر - کانتر ۸ بیتی با PRESCALER مجزا و مد COM PARE.
- یک مقایسه گر آنالوگ داخلی
- یک تایمر - کانتر ۱٦ بیتی با PRESCALER مجزا
- WATCH DOG قابل برنامه ریز با اسیلاتور داخلی

- وقفه در اثر تغییر وضعیت پایه

د- خصوصیات ویژه میکروکنترلر

- CIRCUTT BROWN - OUT - POWER - ON RESET برنامه ریزی .
- منابع وقفه (INTERRUPT) داخلی و خارجی .
- دارای ٦ حالت SLEEP .
- OWER – ON RESET CIRCUTT برای ۱۲ ATTING
- عملکرد کاملا ثابت
- توانت مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS.

ه - توان مصرفی در ۲۵،۳V.۱MHZ برای ATMEGAIGL

- حالت فعال ۱۰/MA
- در حالت بی کاری ۰,۳۵ MA
- در حالت POWER – DOWN : ۱ UA

و - ولتاژ عملیاتی یا کاری

- ۲,۷ V تا ۵,۵V برای ۱٦ L ATMEGA

- ٤,٥ V تا ٥,٥ V برای ATMEGA ١٦

خ - فرکانس های کاری

- ٠ MHZ تا ٨ MHZ برای Atmega ١٦L

- ٠ MHZ تا ١٦ MHZ برای Atmega

ج - انواع بسته بندی

- ٣٢ خط ورودی / خروجی قابل برنامه ریزی

- ٤٠ پایه PDIP , ٤٤ پایه TQFP و ٤٤ پایه MLF

٤-٣-٣ LCD :

LCD مورد استفاده در این پروژه ٢ × ١٦ بوده و به کارگیری در سیستم شامل استفاده از دستورات مختلفی می شود که دستورات و توابع مربوط به LCD نامیده می شود . همچنین پایه های LCD برای اتصال به پایه های میکرو به صورت زیر پیکره بندی می شوند :

CONFIG LCD PIN = PIN , D٨٤ = PN . DBS = PN .

DB٦ = PN , DB٧ = PN , E = PN , RS = PN

PN پایه ای دلخواه از میکرو است که پایه های LCD باید در یک خط نوشته شود و یا در ادامه آن با علامت Cunder Line – در خط بعد نوشته شود .

MOS Switch -۵-۳-۳

در این پروژه ۲ عدد MOS به عنوان کلید استفاده شده است یکی برای سوئیچ کردن خروجی ADM ٦٦٠ روی میکرو و دیگری برای LCD . توضیحات بیشتر در مورد عملکرد کلید ها در صفحات بعدی بررسی می شود .

ADM ٦٦٠ -٦-۳-۳

این آ ی سی یک مبدل DC/DC می باشد که ولتاژ ورودی خود را ۲ برابر می کند . ما در انجام این پروژه از آی سی برای تبدیل ولتاژ ۳/٦ ولت باطری به ۷/۲ ولت استفاده کردیم .

البته در مرحله بعد توسط یک رگلاتور زنری ما به ولتاژ ۵ ولت دست پیدا کردیم . دلیل استفاده از این آ ی سی در قسمت های بعدی مورد بررسی قرار می گیرد .

در مدار پیوست طریقه اتصالات پایه های آ ی سی مورد نظر را می بینید دقت کنید که ظریفت خازن های به کار رفته ۱۰ میکروفاراد می باشد.

٤-۳ نقشه کامل مدار و توضیحات مربوط به آن :

۳-٤-۱ مقدمه :

در بلوک دیاگرام مدار رسم شده در پیوست ، توضیحات هر بلوک نوشته شده و در این قسمت توضیحات دقیق تری در مورد مدار مربوطه نوشته می شود.

در این بخش ما قصد داریم در مورد نحوه و چگونگی ارتباط اجزا مختلف مدار با یکدیگر بررسی داشته باشیم . همچنین چنانچه نکاتی وجود داشته باشد که در هنگام انجام پروژه با آن برخورد داشته ایم (نکات خاص) آنها هم مورد بررسی قرار می گیرند .

در این قسمت ابتدا نقشه کلی مدار در پیوست ملاحظه می گردد و سپس در صفحات بعد توضیحات کامل مربوط به عناصر بررسی خواهند شد.

۳-٤-۲- توضیحات مربوط به نقشه کامل مدار :

همانطور که در نقشه پیوست می بینید اولین قسمت مدار یک منبع تغذیه است که شامل یک عدد پل دیود، یک آ ی سی رگولاتور و یک عدد خازن صافی می باشد. قسمت تغذیه به منظور فراهم کردن ولتاژ ۵ ولت DC می باشد که جهت راه اندازی IC میکروکنترلر و کلیه مدارات سازگار با TTL به کار می رود . ورودی قسمت تغذیه می تواند یک سیگنال AC یا DC باشد . این ورودی توسط یک سوئیچ ON – OFF قطع و وصل می شود . در حالتی که سوئیچ روشن است ورودی وارد یک پل دیود می شود و بعد به کمک یک خازن صاف می شود . این خازن به علت ظرفیت بالای خود باعث

می شود ولتاژی تقریباً DC به دست آید که دارای ریپل است برای ایجاد یک ولتاژ ۵+ کاملاً DC از یک آ ی سی تنظیم کننده ولتاژ ۷۸۰S استفاده شده است که ورودی دارای اعوجاج را به یک ولتاژ کاملاً مستقیم ۵ ولت تبدیل می کند .

در خروجی تنظیم کننده ولتاژ یک خازن ۱۰ میلی فاراد قرار داده شده است . که به اندازه ۵ ولت شارژ خواهد شد که نوسانات خروجی ناشی از تغییرات بار را کاهش می دهد.

بعد از این قسمت یک مقاوت ٤٧٠ اهم و یک عدد LED در مدار تعبیه شده است که LED برای نشان دادن این است که مدار تغذیه وصل است و کار می کند و مقاومت ٤٧٠ اهم برای محافظت LED است تا نسوزد.

بعد از مقاومت و LED و قبل از رله دیود داریم که این دیود هرز گرد رله است و المان بعدی رله است . رله به کار رفته در مدار برای این است که چک کنیم تغذیه وصل است یا خیر و این یعنی هرگاه رله وصل شود تغذیه برقرار است .

همانطور که در نقشه پیوست هم می بینید خروجی های رله به میکروکنترلر و یکی از سوئیچ ها وصل می شود. تغذیه میکرو از ۵ ولت منبع تغذیه است .

MOS های به کار رفته در مدار به عنوان Switch مورد استفاده قرار می گیرند . MOS وصل شده به ADMBBO وظیفه دارد تا ۵ ولت خروجی این آ ی سی را به میکرو اعمل کند .

همانطور که در قسمت نرم افزار هم توضیح خواهیم داد بعد از اینکه تغذیه اصلی قطع شود خازن پشتیبان شروع به تغذیه میکروکنترلر می کند (در مدتی که تغذیه اصلی وصل است خازن شارژ می شود) بعد از مدت زمان مشخصی که ما این مدت زمان را در نرم افزار سیستم معلوم می کنیم با فشار دادن پوش با تن وصل شده به پایه ۱۶ میکروکنترلر یک اینتراپت صادر می شود و میکرو از طریق کلید MOS مربوط به ADM ٦٦٠ وصل می شود و از آن تغذیه می کند . در مورد خازن پشتیبان ذکر این نکته لازم است که این خازن بعد از خاموش شدن تغذیه اصلی باید بتواند تا ۲٤ ساعت میکرو را تغذیه کند یعنی مدت زمان شارژ خازن باید تا ۲٤ ساعت طول بکشد . MOS ها چون دارای امپدانس ورودی بالایی هستند از میکرو جریانی نمی کشند.

پوش با تن متصل به پایه شماره ۹ میکرو برای Reset کردن آن است . در مورد LCD هم در قسمت قبل و در توضیحات مربوط به بلوک دیاگرام توضیحاتی کامل داده شده است. LCD هم با کمک یک کلید که به نام MOSFET LCD نامیده می شود کار می کند این MOS به PORT A.O وصل است. قطعه ۲ پایه نقره ای کنار میکرو هم کریستال خارجی آن است . دلیل استفاده از این کریستال آن است که ما برای ساعت نیاز به تایمر داریم و درمد Power save فقط تایمر ۲ است که در حالت آسنکرون کار می کند . فرکانس نوشته شده روی این کریستال ۳۲۷٦۸ است از طرفی ما در

برنامه نرم افزاری مربوطه Presale را ۱۲۸ تعریف کردیم و این یعنی اینکه اگر ۲۵٦ پالس روی میکرو بیاید می شود یک ثانیه و به این ترتیب یک تایمر برای ساعت سازی ایجاد می کنیم .

المان دیگر به کار رفته در مدار آ ی سی AMD ٦٦٠ می باشد . این آ ی سی یک مبدل ولتاژ DC به DC است. به طوری که ولتاژ ورودی خود را ۲ برابر می کند همانطور که در قسمت بلوک دیاگرام ها توضیح داده شد ما از این آ ی سی برای تبدیل ولتاژ ۳/٦ ولت به ولتاژ ۷/۲ ولت استفاده کردیم و بعد با کمک یک زنر ولتاژ ۵ ولت ثابت به میکرو تحویل دادیم .

نحوه اتصلات پایه های این آ ی سی در قسمت های قبلی بررسی شد .

۳-۵- چگونگی کارکرد مدار :

برای توضیح چگونگی کارکرد مدار بهتر است ابتدا بررسی کنیم هدف از انجام این پروژه چه بوده است و بعد بررسی کنیم که برای حل مشکلات مطرح شده، مدار چگونه کار می کند.

همانطور که در قسمت ضرورت ها و نیازها توضیح داده شد مشکل اول ما عملکرد سیستم بود یعنی LCD با ولتاژ ۵ ولت کار کار می کند اما مدارات ما هنگام استفاده از خازن پشتیبان و باطری دارای ولتاژ ۳/٦ تا ۵ ولت هستند و مشکل دوم گرانی باطری و دشواری تعویض آن بود که باعث می شد تا کاری کنیم که تا حد امکان مصرف در حالت استفاده از خازن و باطری کم شود.

برای این منظور در مدار طرح شده ابتدا با فشردن کلید اصلی سر راه منبع تغذیه LCD روشن شده و یک کاراکتری که به طور دلخواه در نرم افزار سیستم تعریف می شود را نشان می دهد در این مدت خازن پشتیبان شروع به شارژ شدن می کند .حال بعد از مدتی که این زمان را ما در نرم افزار سیستم تعریف می کنیم تغذیه میکرو و از خازن به روی باطری سوئیچ می شود . حال ما با فشار دادن پوش با تن مربوط که روی برد این کلید رنگی است می توانیم یک فرمان اینتراپت صادر کنیم و میکرو از این حالت به سوئیچ مربوطه که یک MOS است فرمان می دهد تا خروجی آ ی سی ADM٦٦٠ را روی میکرو بیاورد و ما باز هم کاراکتر تعریف شده در نرم افزار را روی صفحه نمایش LCD می بینیم . بعد از گذشت ٥ ثانیه (زمانی که نرم افزار سیستم تعریف می شود) نمایش کاراکتر روی LCD قطع شده و میکرو به Stand by می رود.

فصل ٤ - نرم افزار

٤-١- مقدمه ای بر نرم افزار مورد استفاده و کامپایلر به کار رفته

انواع متنوعی ازکامپایلرهای AVR عرضه شده اند که از بین آنها کامپایلرهای CODE VISION BASCOM , و FASTAVR استفاده کردیم . BASCOM تمام میکروهای AVR را حمایت کرده و از زبان BASIC برای برنامه نویسی AVRها استفاده می نماید. BASCOM دارای منوهای متعددی است که ما از تشریح آنها در اینجا خوداری می نماییم. از قابلیت های بسیار ارزنده

محیط BASCOM داشتن تحلیل گر یا به عبارتی SIMULATOR داخلی است که برای یادگیری برنامه های AVRبسیار مفید است.

```
$ Crystal= ۸۰۰۰۰۰۰

$ Regfile = "m ۱٦def.dat"

Config lcdpin = pin, Db۷ = Portb .۷, Db ٦ = Portb .٦, Db٥ = Portb .٥, Db٤ = portb.٤

,  Rs = Portb .۰, E = Portb.۱

Confing lcdmode = Port

Confing lcdbus =٤

Confing Watchdog= ۲۰٤۸
```

```
Confing Porta = Output

Confing Portb = Input

Confing Portd = Input

Confing Int۰ = Low Leve]

Confinga =۲۵۵

Confingd =۲۵۵

Confing Timer = Timer, Async = on. Prescale = ۱۲۸

Enable Interrupts

Stop Watchdog

Enable Timer ۲

Enable Int۰

On Int۰ Int۰ _ isr No save

Dim Ali as Word

Dim S as Byte

Dim M as Byte

Dim H as Byte

Dim H۲ as Byte

Bat Alias porta.۱

Lcdmos Alias Porta.۰

Power relay Alias Pind .٦
```

H۲ = ۰

Bat = ۱

Lcdmos = ۰

Start Timer۲

Cls

Cursor off

LCD "Timer ۲"

Do

Power save

If H۲ < ۱۲ Then

If Power relay = ۱ Then

Lcdmos = ۱

Portb = ۰

End If

Else

If Ali < ۵ Then

Lcdmos = ۰

Portb = ۲۵۵

Else

```
Lcdmos = ١

Portb = ٠

End If

If Power relay = ٠ Then

H٢ = ٠

Bat = ١

End If

End If

Loop

Timer٢_isr :

If Power relay = ٠ Then

Bat = ١

Ali = ٠

Portb = ٢٥٥

Initlcd

Cls

Cursor off

Lcd " Timer٢"

Locate ٢ , ١

Lcd H ; ":" ; M ; ":" ; S ; ":"
```

End If

Ali =Ali + ١

s = s + ١

If power relay = ١ Then

H٢ = H٢ + ١

End If

If H٢ > ١٢ Then

Bat = ٠

H٢ = ٢٠

Else

Nat = ١

End IF

If S= ٦٠ Then

S = ٠

M = M + ١

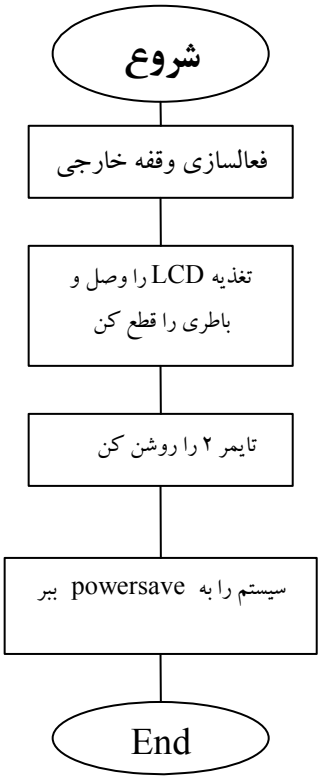
If M = ٦٠ Then

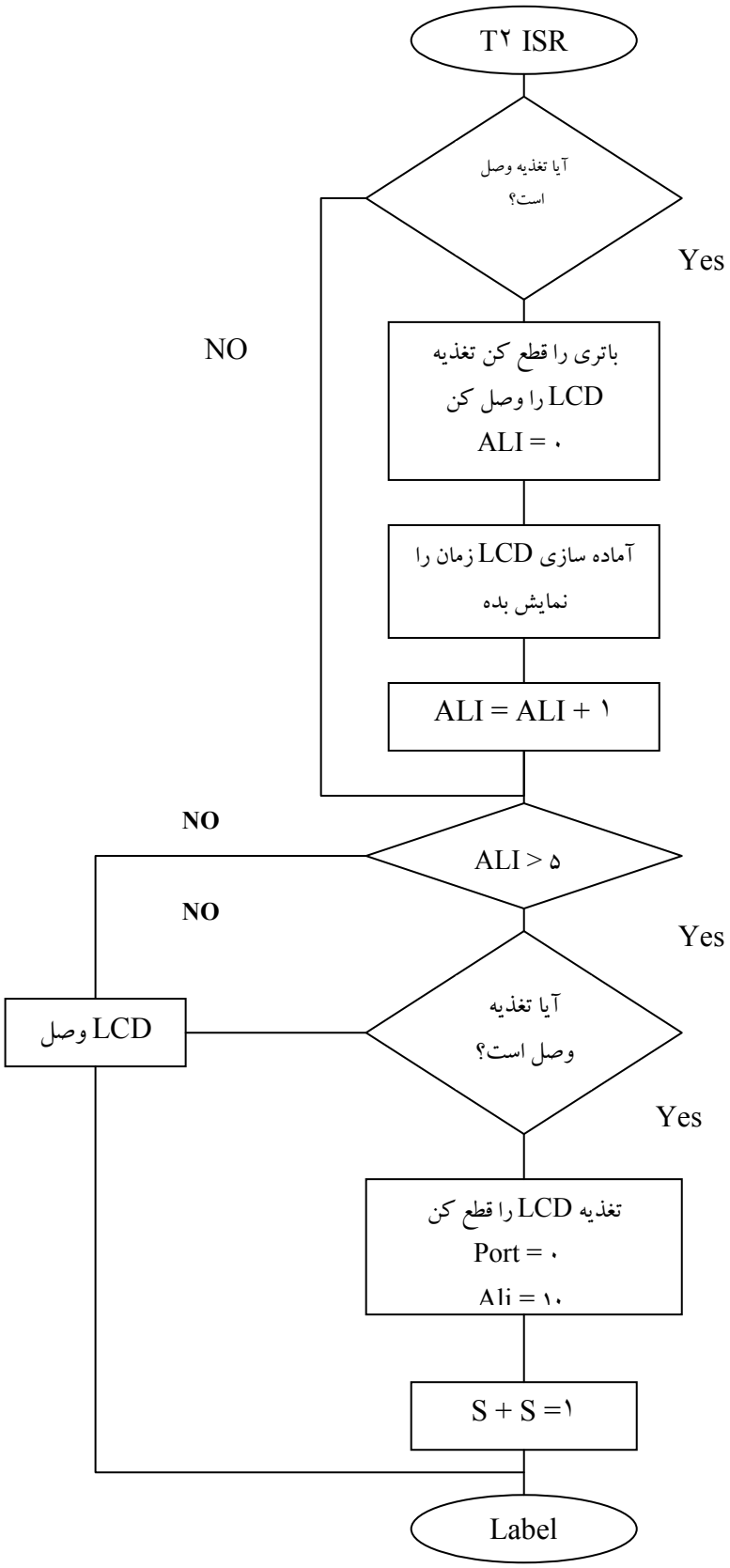
M = ٠

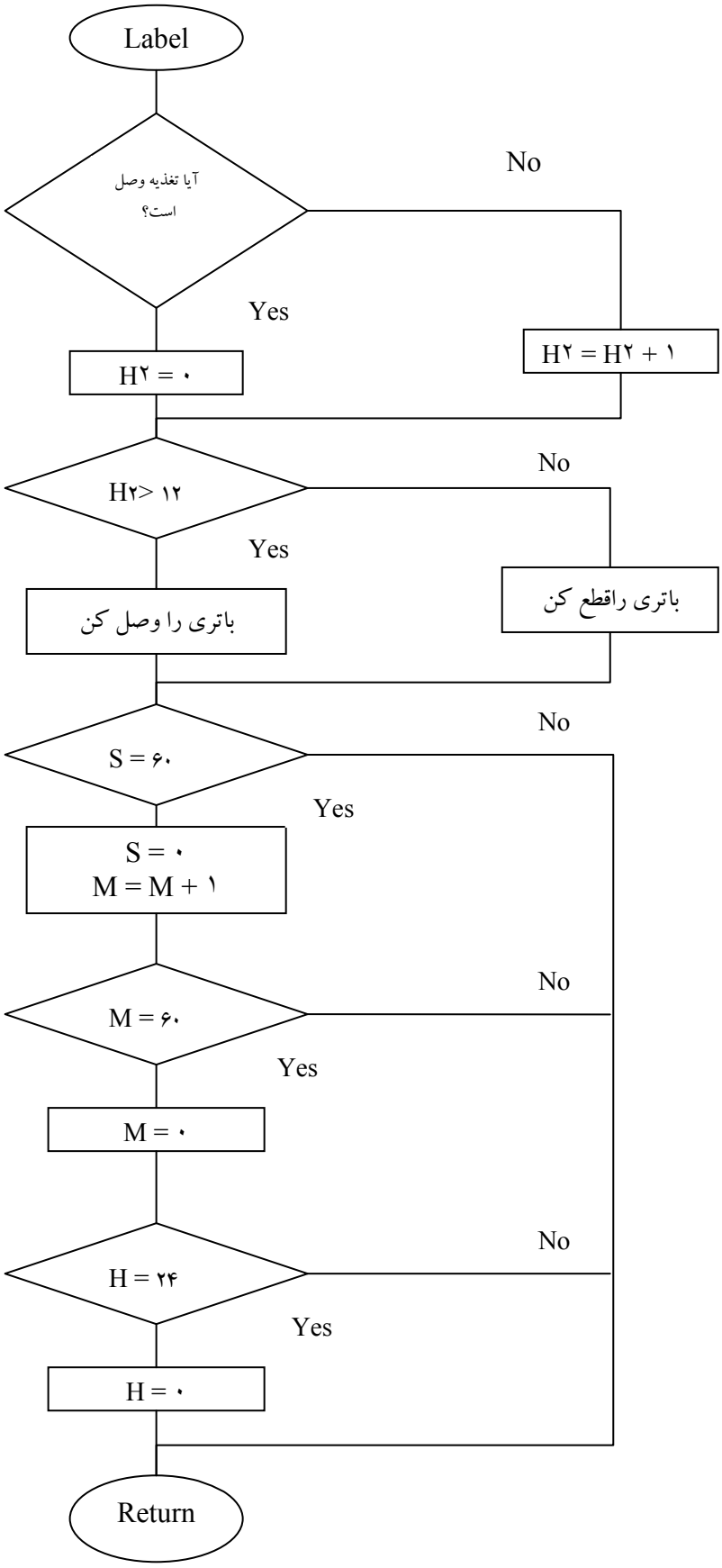
If H = ٢٤ Then

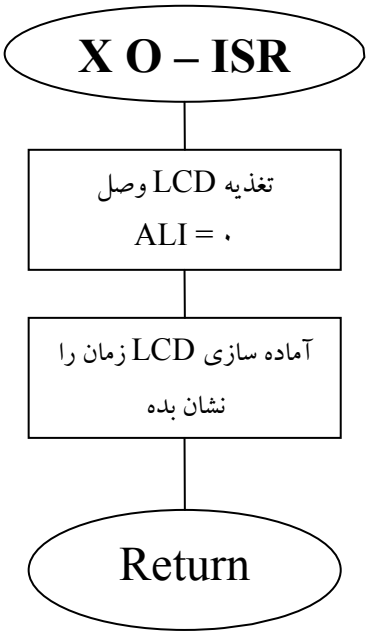
H = ٠

```
End If

End If

End If

Return

Int٠_isr:

Lcdmos = ٠

Ali = ٠

Initlcd

Cls

Cursor off

Lcd " Timer٢"

Local ٢ , ١

Lcd H ; ":" ; M ; ":" ; S ; ":"

Return

End
```


شروع
فعالسازی وقفه خارجی
تغذیه LCD را وصل و
باطری را قطع کن
تایمر ۲ را روشن کن
سیستم را به powersave ببر
End


T۲ ISR
آیا تغذیه وصل
است؟
Yes
NO
باتری را قطع کن تغذیه
LCD را وصل کن
ALI = ۰
آماده سازی LCD زمان را
نمایش بده
ALI = ALI + ۱
NO
ALI > ۵
NO
Yes
آیا تغذیه
وصل است؟
LCD وصل
Yes
تغذیه LCD را قطع کن
Port = ۰
Ali = ۱۰
S + S =۱
Label


Label
آیا تغذیه وصل است؟
No
Yes
H۲ = ۰
H۲ = H۲ + ۱
H۲> ۱۲
No
Yes
باتری را وصل کن
باتری راقطع کن
S = ۶۰
No
Yes
S = ۰
M = M + ۱
M = ۶۰
No
Yes
M = ۰
H = ۲۴
No
Yes
H = ۰
Return


X O – ISR
تغذیه LCD وصل
ALI = ۰
آماده سازی LCD زمان را
نشان بده
Return

٤-٣ - توضیحات نرم افزار برنامه

١- معرفی کریستال سیستم به عنوان کریستال هشت مگا هرتز به کامپایلر

٢- معرفی IC مورد استفاده به کامپایلر

٣- تنظیم LCD

٤- تنظیم LCD برای دو خط و ١٦ کاراکتر

٥- LCD بصورت پین های مجزا معرفی می گردد

٦- اطلاعات LCD ٤ بیتی

٧- تنظیم پورت A به عنوان خروجی

٨- تنظیم پورت B به عنوان ورودی

٩- تنظیم پورت D به عنوان ورودی

١٠-اینتراپت خارجی صفر حساس به لبه پایین رونده

١١- همه پین های پورت A یک شوند

١٢-همه پین های پورت B یک شوند

١٣-تنظیم تایمر دو به صورت آسنکرون با تقسیم کلاک ١٢٨

۱٤-فعال سازی عمومی وقفه ها

۱۵-فعال سازی وقفه تایمر ۲

۱٦-فعال سازی وقفه خارجی صفر

۱۷-معرفی ISR– Int۰ به عنوان روال سرویس وقفه خارجی صفر

۱۸-معرفی ISR – ۲ Timer به عنوان روال سرویس وقفه تایمر ۲

۱۹-معرفی ALI به صورت Word

۲۰-معرفی ساعت ، دقیقه و ثانیه و پارامتر H۲ به صورت ۱ بایتی

۲۱-نام گذاری پورت ۱. a با اسم BAT

۲۲-نام گذاری پورت a.۰ با اسم LCDDMOS

۲۳-ناگذاری پورت d.٦ با اسم Power relay

۲٤-BAT یک شود

۲۵-LCDMOS صفر شود

۲٦-تایمر ۲ را روشن کن

۲۷-صفحه نمایش را پاک کن

۲۸-مکان نما خاموش

۲۹- " ۲ timer" را نمایش بده

۳۰-برای همیشه سیستم را به Power save ببر

۳۱-پایان

۳۲-روال سرویس وقفه تایمر ۲

۳۳-اگر تغذیه وصل است

۳٤-باتری قطع شود

۳۵- LCD روشن شود

۳٦-آماده سازی LCD

۳۷-زمان را نمایش بده

۳۸-اتمام شرط

۳۹-اگر ALI > ۵ است

٤۰-LCD خاموش شود

٤۱-اگر ALI < ۵ است ، LCD روشن شود

٤۲-پایان شرط

٤۳-S را یکی اضافه کن

٤٤-اگر تغذیه وصل است

٤٥- H را صفر کن

٤٦-اگر وصل است ، H٢ را یکی اضافه کن و پایان شرط

٤٧-اگر H٢ > ١٢ است

٤٨-باتری وصل شود

٤٩-اگر H٢ < ١٢ است ، باتری قطع و پایان شرط

٥٠-اگر S = ٦٠ است

٥١-S را صفر کن و M را یکی اضافه کن

٥٢-اگر M = ٦٠ است ، M را صفر کن

٥٣- اگر H = ٢٤ است ، H را صفر کن

٥٤-پایان شرط

٥٥-برگشت از روال سرویس وقفه

٥٦-روال سرویس وقفه خارجی صفر شود

٥٧-LCD را روشن کن

٥٨- ALI را صفر کن

۵۹-LCD را آماده کن

۶۰-نمایش زمان

بررسی دستورات AVR

دستور Const

برای تعریف یک ثابت از این دستور استفاده می شود .

```
CONST SYMBOL = NUMCONST
CONST SYMBOL = STRINGCONST
CONST SYMBOL = EXPRESSIONVD
```

SYMBOL نـام ثابـت و NUMCONST رشـته انتـساب یافتـه بـه SYMBOL و EXPRESSION می تواند عبارتی باشد که نتیجه آن به SYMBOL انتساب یابد .

دستور ALIAS

از این دستور برای تغییر نام متغیر استفاده می شود .

```
DIRECTION ALIAS PORTB.۱
```

دستور CHR

از این دستور برای تبدیل متغیر عددی یا یک قابت به کاراکتر استفاده می شود . زمانی که شما قصد دارید یک کاراکتر بر روی LCD نمایش دهید از این دستور می توانید استفاده کنید . در صـورتی

که از این دستور بدین صورت استفاده نمایید PEINR CHR (VAR) کاراکتر اسکی VAR به پورت سریال فرستاده خواهد شد.

دستور INSTR

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

Var = Instr (Start , Start , Substr)

Var = Instr(String , Substr)

VAR عددی است که مشخص کننده محل SUNSTR 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 میکند . در صورتی که 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.

دستور INCR

این دستور یک واحد به متغیر عددی VAR می افزاید .

INCR var

دستور DECR

این دستور متغیر VAR را یک واحد کم می کند .

DECR var

دستور CHECKSUM

این دستور مجموع کد دسیمال اسکی رشته VAR را بر می گرداند که البته اگر مجموع کد اسکی رشته از عدد ۲۵۵ بیشتر شود مقدار ۲۵٦ از مجموع کم می شود

دستور LOW

این دستور LSB (least significant byle) یک متغیر را بر می گرداند .

Var = LOW (s)

LSB متغیر S در Var قرار می گیرد .

دستور HIGH

این دستور پرارزشترین بایت (MSB) یک متغیر را برمی گرداند

Var = HIGH (S)

مقدار MSB متغیر S در Var قرارمی گیرد .

دستور LCASE

این دستور تمام رشته مورد نظر را تبدیل به حروف گوچگ می کند .

Target = Lcase (source)

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

دستور RIGHT

با این دستور قسمتی از یک رشته را جدا می کنیم .

Var = RUGHT (var ۱ , n)

از سمت راست رشته var۱ به تعداد کاراکتر n ، رشته ای جدا شده و در رشته var قرار گیرد.

دستور LEFT

این دستور کاراکترهای سمت چپ یک رشته را به تعداد تعیین شده جدا می کند .

Var = LEFT (var۱, n)

رشته var۱ از سمت چپ به تعداد n کاراکتر جدا شده و در رشته var قرار می گیرد.

دستور LEN

این دستور طول یا به عبارتی تعداد کاراکترهای یک رشته را برمی گرداند .

Var = LEN (string)

طول رشته STRING در متغییر عددی var قرار می گیرد . رشته STRING نهایتا ۲۵۵ بایت می تواند طول داشته باشد. توجه داشته باشید که فضای خالی (Space bar) خود یک کاراکتر به حساب می آید.

دستور LTRIM

این دستور فضای خالی رشته را حذف می کند .

Var = LTRIM (org)

فضای خالی رشته org برداشته می شود (حذف می شود) و رشته بدون فضای خالی در متغییر رشته ای var قرار می گیرد .

دستور SWAP

با اجرای این دستور محتوای متغیر var۱ در متغیر var۲ و محتوای نتغیر var۲ در متغیر var۱ قرار می گیرد . var۱ و var ۲ می توانند داده هایی از نوع byte , integer , word , long , bit و یا string باشند

نکته : دو متغیر var۱ و var۲ بایستی از یک نوع داده باشند.

۱- Var = Mid (var , st [, L]

۲- Mid (var , st , [, L]

۱- قسمتی از رشته Var۱ با شروع از کارکتر st ام و طول L برداشته شده و در متغیر var قرا می گیرد.

۲- رشته var۱ در رشته var با شروع از کارکتر st ام و طول L قرار می گیرد.

در صورت قید نکردن گزینه اختیاری L بیشترین طول در نظر گرفته می شود.

دستور ROTATE

دستور زیر تمام بیت ها را به چپ یا راست منتقل می کند ولی تمام بیتها محفوظ هستند وهیچ بیتی بیرون فرستاده نمی شود.

ROTATE var , LEFT / RIGHT [, shifts]

Var می تواند داده ای از نوع word ، integer ، byte یا long باشد . Left / Right جهت چرخش بیتها و shift که اختیاری می باشد تعداد چرخش بیتها را مشخص می کند . در صورت قید نکردن shift به صورت پیش فرض مقدار یک در نظر گرفته می شوند.

دستور SPACE

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.

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.

تابع FUSING

این دستور برای رند 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 حتما با علامت # شروع شود و حداقل باید یکی از علامات

و & بعد از ممیز داشته باشد . علامات # و & با یکدیگر بعد از ممیز استفاده نمی شود . با استفاده از علامت # عدد روند می شود و در صورت استفاده از علامت & روندی صورت نمی گیرد .

جدول LOOKUP

توسط این جدول می توان مقدار دلخواهی را از جدولی برگرداند .

Var = lookup (value , label)

label برچسب جدول و value اندیس رشته دلخواه در جدول است .

رشته برگشتی از جدول در متغیر رشته ای var قرار می گیرد . Value = ۰ اولین رشته در جدول را باز میگرداند. تعداد اندیس ها نهایتا می توانند ۲۵۵ باشد

توابع ریاضی و محاسباتی

عملکردهای ریاضی

تابع ABS

Var = ABS (var۲)

این دستور به معنای ریاضی var = | var ۲| (قدر مطلق) است .

تابع EXP

Target = EXP (source)

Target برابر با e به توان source است . Target متغیری از نوع داده single است .

تابع LOG۱۰

Target = LOG ۱۰ (source)

لگاریتم پایه ۱۰ متغیر یا ثابت source از نوع داده single گرفته می شود و در متغیر target و source هر دو داده و نوع single هستند.

تابع LOG

این دستور لگاریتم طبیعی یک داده از نوع single را برمی گرداند .

Target = LOG (source)

لگاریتم ثابت یا متغیر source از نوع داده single قرار می گیرد . این تابع برای اجرا شدن وقت زیادی می برد مخصوصا زمانی که اعداد بزرگ استفاده می شود . همچنین با بزرگتر شدن اعداد دقت نیز پایین خواهد آمد.

تابع RND

این دستور یک عدد تصادفی را برمی گرداند .

Var = RND (limit)

عدد تصادفی بین ۰ و limit بدست آمده و در متغیر VAR قرار می گیرد. با هر بار استفاده از این دستور عدد مثبت تصادفی دیگر بدست خواهد آمد.

نکته : limit باید یک عدد مثبت باشد .

تابع SIN

Var = SIN(source)

این دستور (sine) سینوس 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.

تابع COS

Var = COS(source)

این دستور کسینوس (cosine) ثابت یا متغیر source را 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 .

تابع TAN

Var = TAN(source)

این دستور تانژانت (tangent) ثابت یا متغیر source از نوع داده single قرار می دهد . تمام دستورات مثلثاتی با رادیان کار می کنند و ورودی این دستور بایستی به رادیان باشد .

تابع SINH

این دستور sine hyperbole یک ثابت و متغیر از نوع داده single را به رادیان می دهد . سینوس هایپربولیک source در متغیر عددی var از نوع داده single قرار می گیرد . شما با دستورات RAD۲DEG و DEG۲RAD می توانید مقدار به دست آمده را به درجه تبدیل نمائید .

تابع COSH

این دستور cosine hyperbole یک ثابت یا یک متغیر را از نوع داده single را به رادیان می دهد.

Var = cosh (source)

کسینوس هایپربولیک source در متغیر عددی var از نوع داده single قرار می گیرد . شما با دستورات RAD۲DEG و DEG۲RAD می توانید مقدار به دست آمده را به درجه تبدیل کرد.

تابع TANH

این دستور tangent hyperbole یک ثابت یا متغیر از نوع داده single را به رادیان می دهد.

Var = TANH(source)

تانژانت هایپربولیک source درمتغیر عددی var از نوع داده single قرار می گیرد . شما توسط دستورات RAD۲DEG و DEG۲RAD می توانید مقدار به دست آمده را به درجه تبدیل نمائید.

تابع ASIN

این دستور ARCINE 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 ۱- و ۱+ باشد . نتیجه این محاسبه کخ بین II/۲ و II/۲– در متغیر var از نوع داده single قرار می گیرد . اگر عدد ورودی بیشتر ۱+ باشد II/۲ و اگر عدد ورودی کمتر از۱- باشد II/۲– بر گردانده می شود. شما با دستوراتRAD۲DEG و DEG۲RAD می توانید مقدار به دست آمده را به درجه تبدیل نمائید.

تابع ACOS

این دستور ARCOSINE یک ثابت یا متغیر نوع single را به رادیان بر می گرداند .

Var = ACOS(x)

متغیر ورودی x عددی از نوع دادهsingle است و می توانید بین ۱- تا ۱+ باشد . نتیجه این محاسبه که بین ۰ و II است در متغیر var از نوع داده single قرار می گیرد . اگر عدد ورودی بیشتر از ۱+ باشد صفر و اگر عدد ورودی کمتر ۱- باشد II برگردانده می شوند شما با دستوراتRAD۲DEG و DEG۲RAD می توانید مقدار بدست آمده را به درجه تبدیل نمائید .

تابع ATN

Var = ATN(x)

این دستور ARCTANGENT یک ثابت یا متغیر از نوع داده single را به رادیان می دهد . متغیر x می تواند عدد از نوع داده single باشد ، نتیجه این محاسبه که بین II/۲ و II/۲– است در متغیر var از نوع داده single قرار می گیرد .

تابع DEG۲RAD

برای تبدیل رادیال به درجه از این دستور استفاده می شود .

VAR=RAD۲DEG(SINGLE)

رادیان SINGLE به درجه تبدیل می شود و در متغیر VAR از نوع داده SINGLE قرار میگیرد.

تابع ROUND

متغیر یا داده Xاز نوع SINGLE روند شده ودر متغیرVARاز نوع داده SINGLE قرار میگیرد .

۵-٤ تبدیل کدها و متغیرها به یکدیگر

دستور ASC

این دستور اولین کاکتر یک متغیر از نوع داده string را به مقدار اسکی آن تبدیل می کند .

دستور HEX

این دستور یک داده از نوع word Integer ، Byte یا Long را به مقدار هگزادسیمال تبدیل می کند .

Var = HEX (x)

مقدار HEX متغیر یا ثابت x در نتغیر var جای می گیرد .

دستور HEXVAL

دستور HEXVAL یک داده هگزادسیمال را به مقدار عددی (دسیمال) تبدیل می کند .

Var = HEXVAL (x)

مقدار عددی داده هگزادسیمال x که می تواند Word , Integer , Byte یا Long باشد در متغیر var جای می گیرد .

دستور MAKEBCD

Var = MAKEBCD (x)

این دستور متغیر یا قابت var۲ را تبدیل به مقدار BCD می کند و در متغیر var۱ جای می دهد .

دستور MAKEDEC

Var۱ = MAKEBCD(var۲)

این دستور برای تبدیل یک داده BCB نوع Word , Byte یا Integer به مقدار Decimal از این دستور استفاده می شود . مقدار دسیمال متغیر دسیمال یا ثابت var۲ در متغیر var۱ قرار می گیرد.

دستور MAKENT

Varn = MAKENT (LSB,MSB)

این دستور دو بایت را به هم متصل می کند و یک داده نوع Word یا Integer می سازد که LSB بایت کم ارزش و MSB بایت پر ارزش متغیر دو بایتی varn را تشکیل می دهند .

Varn = (۲۵۶ *MSB)+ LSB

دستور STR

این دستور میتوان یک متغیر عددی (x) را به رشته (var) تبدیل کرد .

Var = str(x)

دستور VAL

Var = val (s)

این دستور می توان یک رشته (s) را به متغیر عددی (var) تبدیل کرد. این دستور دقیقاً عکس دستور STR عمل می کند .

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 = ۰ یک رشته به طول ۲۵۵ کاراکتر تولید می شود . همچنین قراردادن n = ۰ قابل قبول نیست .

تابع BIN۲GREY

متغیر var۲ که می تواند داده ای از نوع Byte , Integer , Word یا Long باشد به کد گری تبدیل شده و در متغیر var۱ قرار می گیرد .

تابع GRAY۲BIN

Var۱ = GRY۲BIN (var۲)

کد گری var۲ به مقدار باینری تبدیل شده و در متغیر avr۱ کخ می تواند داده ای از نوع Integer , Word , Byte یا Long باشد قرار می گیرد .

رجیسترها و آدرس های حافظه

دستور SET

توسط این دستور می توان یک بیت را یک کرد .

SET Bit / Pin
SET Var .X

Bit می تواند یک بیت یا یک SFR مانند PORTB.۱ باشد و var متغیر از نوع داده Integer , Word , Byte یا Long است . x برای Byte می تواند ۰ تا ۷ ، ۰ تا ۱۵ برای Word و برای Long می تواند ۰ تا ۳۱ باشد.

دستور TOGGLE

این دستور مقدار منطقی یک پایه با یک بیت را معکوس می کند .

TOGGLE Pin / Bit

Pin می تواند یک SFR مانند PORTB.۱ و یا یک بیت باشد.

دستور RESET

این دستور می توان یک بیت را صفر کرد .

RESET Bit/Pin

RESET Var .X

Bin می تواند یک بیت و یا یک SFR مانند PORTB.۱ باشد و Var متغییری از نوع داده Word Integer , Byte , یا Long است و X برای Byte می تواند ۰ تا ۷ ، ۰ تا ۱۵ برای Word و برای Long می تواند ۰ تا ۳۱ باشد.

دستور BITWAIT

BITWAIT X,SET / RESET

توسط این دستور اجرای این برنامه تا زمانی که بیت X ، SET (= ۱) یا RESET (=۰) شود و در خط جاری متوقف می ماند . در صورت True شدن شرایط ، اجرای برنامه از خط بعد ادامه میابد .

x می تواند یک بیت دجیستر داخلی مانند PORTB.Y باشد که Y می توانـد بـین اعـداد ۰ تـا ۷ تغییر کند .

دستور CPEEK

Var = CPEEK(address)

این دستور برای برگرداندن بایتی که در آدرسی از حافظه کدی ذخیره شده است استفاده می کنیم . بـا این دستور میتوانید به رجیسترهای داخلی نیز دسترسی پیداکنید. لازم به تذکر است کـه بـا ایـن دسـتور شما نمی توانید در حافظه داخلی بنویسید .

دستور CPEEKH

با این دستور می توانید بایت ذخیره شده در صفحه (page) بـالای حافظـه کـدی (FLASH MEM) میکرو ۱۰۳ Mega با دیگر میکروها که دارای ۱۲۵K حافظه است را خواند .

Var = CPEEKH (address)

Address آدرس حافظه و محتوای آدرس در متغیر یک بایت var مقدار می گیـرد . CPEEKH (۰) محتوای اولین بایت حافظه بالای ٦٤K را بر می گرداند .

دستور LOADADR

LOADADR var , reg

با این دستور می توانید آدرس یک متغیر را در یک جفت رجیستر ذخیره کنید . VAR متغیری اسـت که آدرس آن در متغیرهای دو بایتی X, Y و Z ذخیـره مـی شـود و REG رجیـسترهای X , Y و Z هستند . این دستور جزء دستور اسمبلی است و برای کمک به برنامه نویسان اضافه شده است .

دستور OUT

توسط این دستور می توان یک بایت به یک پورت سخت افزاری یا آدرس حافظه داخلی / خارجی ارسال کرد .

OUT address , value

Value به آدرس address که می تواند بین ۰H تا FFFFH باشد فرستاده می شود . دستور OUT می تواند در تمام مکانهای حافظه VAR بنویسد. توجه کنید که برای address یک Word تعریف شود . در ضمن برای نوشتن در مکان حافظه خارجی (X RAM) باید در محیط BASCOM در منوی (Option – Compiler –Chip) ، گزینه (External Access Enable) را فعال کنید .

دستور INP

توسط این دستور می توان یک بایت از پورت سخت افزاری یا آدرس حافظه داخلی / خارجی خواند .

Var = INP (address)

محتوای آدرس address که می تواند بین ۰H تا FFFFH باشد خوانده شده و رد متغیر var قرار می گیرد . دستور INP می تواند از تمام مکانهای حافظه var بخواند . در ضمن برای خواندن از مکان حافظه خارجی (X RAM) باید در محیط BASCOM و در منوی (Option – Compiler – Chip) گزینه (External Access Enable) را فعال کنید .

دستور PEEK

این دستور محتوای یک ریجیستر را بر می گرداند .

Var = PEEK (address)

Address آدرس رجیسترهای R۰ تا R۳۱ است بین ۰ تا ۳۱ می باشد . محتوای رجیستر در متغیر var جای می گیرد . دستور PEEK () فقط می تواند محتوای رجیسترها را بخواند ولی INP () از تمام مکان های حافظه توانایی خواندن را داراست .

دستور POKE

با این دستور می توانیم یک بایت داده را در یکی از رجیسترها تبنویسیم .

POKE address , Value

مقدار متغیر یا ثابت یک بایتی value در آدرس address که بین ۰ تا ۳۱ برای رجیسترهای R۰ – R۳۱ است نوشته می شود .

دستور VARPTR

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

Var = VARPTR(var ۲)

آدرس متغیر var۲ درمکانحافظه بدست آمده و در متغیر var قرار می گیرد .

دستور العمل های حلقه و پرش

دستور JMP , GOTO

GOTO label

JMP label

با این دستورات می توان به برچسب label پرش کرد . برچسب label باید با علامت : (colon) پایان باید و می تواند تا ۳۲ 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 .

دستور العمل DO – LOOP

فرم کلی دستور DO … LOOP به صورت زیر می باشد .

```
Do
        Statements
LOOP [ UNTIL expression ]
```

دستور العمل statements تا زمانی که expression دارای ارزش True یا غیر صفر است تکرار خواهد شد بنابراین این نوع حلقه حداقل یک بار تکرار می شود DO – LOOP به تنهایی یک حلقه بینهایتاست که با Exit Do می توان از درون حلقه خارج شد و اجرای برنامه در خط بعد از حلقه ادامه یابد.

دستور العمل FOR – NEXT

فرم کلی دستور FOR … NEXT به صورت زیر می باشد .

```
For var = start TO end [STEP value ]
            Statement
Next var
```

Var بعنوان یک کاراکتر 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 کمثبت یا منفی باشد . در صورت حذف کردن Step value کامپایلر به صورت پیش فرض مقدار یک را در نظر می گیرد .

نکته : نوشتن نام متغیر var بعد از next الزامی نیست .

دستور العمل WHILE _ WEND

دستورالعمل WHILE _ WEND تشکیل یک حلقه تکرار می دهد که تکرار این حلقه زمانی ادامه می یابد که عبارت به کار برده شده نتیجه False را حاصل کند و یا دارای مقدار صفر شود . دستور العمل While به صورت ورود به حلقه با شرط می باشد یعنی قبل از ورود به حلقه شرط حلقه تست می شود و در صورت True بودن کنترل اجرای برنامه به حلقه وارد می گردد . بنابراین حلقه While ممکن است در حالت های اصلا اجرا نشود یعنی حتی یک بار هم مراحل حلقه طی نشود .

```
While condition
        Statement
Wend
```

بخش statement می تواند یک دستور العمل ساده یا چند دستور العمل مرکب باشد.

حالت ۰ :

```
If Expression Then statement
```

دستور العمل statement زمانی اجرا می شود که عبارت expression دارای ارزش true باشد .

حالت ۱ :

```
if Expression Then
        Statement ۱
Else
        Statement ۲
End If
```

در صورتی که عبارت expression دارای ارزش true باشد دستور العمل statement اجرا می شود .

```
If Expression ۱ Then
```

Statement ۱

Else if [Expression ۲ Then]

Statement ۲

Else

Statement ۳

End If

در صورتی که عبارت ۱ expression دارای ارزش true باشد دستور العمل ۱ statement اجرا خواهد شد. در صورتی که عبارت ۱ expression دارای ارزش false ولی عبارت اختیاری ۲ expression اجرا خواهد شد و در غیر این صورت یعنی در حالتی که هر دو عبارت ۲ expression و ۱ expression دارای ارزش false باشند دستور العمل statement اجرا خواهد شد . همچنین با دستور IF می توان یک یا صفر بودن یک بیت را از یک متغیر را امتحان کرد.

If bit = ۱ Then or If bit = ۰ Then

یا

Dim var as byte, Idx as byte

Idx = ۱

If var. Idx = ۱ then If bit ۰ of var is ۱ then

Set PORTB. ۰

Else

دستور العمل 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 می توان یکی از چندین دستور را با توجه به مقـدار ورودی اجرا کرد .

Select Case var

Case test ۱: statement ۱

[Case test ۲ : statement ۲]

Case else : statement ۳

End select

اگر متغیر var یا مقدار test۱ برابر باشد statement۱ اجرا می شود و سپس اجرا برنامه بعد از End select ادامه می یابـد و نهایتـاً اگـر متغیـر var بـا هیچکـدام از مقـادیر ۱ test و test۲ برابـر نباشـد statement ۳ اجرا می شود و سپس اجرای برنامه بعد End select ادامه می یابد . شما مـی توانیـد به صورتهای زیر نیز متغیر را امتحان کنید:

اگر متغیر مورد نظر بزرگتر از دو باشد.

Case is > ۲ :

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

اگر متغیر مورد نظر بین ۲ تا ۵ باشد .

Case to ۵ :

دستور EXIT

با این دستور می توانید فقط از یک ساختار یا حلقه خارج شوید و ادامه برنامه را بعد از ساختار یا حلقـه 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

دستور العمل ON VALUE

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

On var [GOTO] [GOSUB] label ۱ [, label ۲]

Var متغیر مورد نظر برای امتحان شدن که می تواند یک SFR مانند PORTD باشد و Label ۱ , ... label ۲ برچسبهایی هستند که با توجه به مقدار var به آنها پرش می شود . توجه داشته باشید زمانی که var = ۰ باشد به اولین برچسب پرش می شود.

دستور DELAY

این دستور برای مدت کوتاهی به مقدار ۱۰۰۰ میکرو ثانیه در اجرای برنامه تاخیر ایجاد می کند .

Delay , whit for hardware to be ready

دستور WAITUS

برای ایجاد تاخیر در برنامه از این دستور استفاده می شود .

Waitus microseconds

اجرای برنامه به مدت microseconds میکرو ثانیه متوقف می شود پس از سپری شدن زمان مشخص شده اجرای برنامه از خط بعد ادامه پیدا می کند milliseconds می تواند عداد بین (۲۵۵ –۱) باشد .

نکته : دستورات تاخیری زمان دقیق را به شما نمی دهد . برای بدست آوردن زمان دقیق از تایمرها استفاده نمایید .

دستور WAITMS

برای ایجاد تاخیر در برنامه از این دستور استفاده می شود .

WAITMS milliseconds

اجرای برنامه به مدت milliseconds میلی ثانیه متوقف می شود . پس از سپری شدن زمان مشخص شده اجرای برنامه از خط بعد ادامه پیدا می کند . milliseconds می تواند عداد بین (۶۵۵۳۵-۱) باشد.

دستور WATH

برای ایجاد تاخیر در برنامه از این دستور استفاد می شود .

Wait seconds

اجرای برنامه به مدت seconds متوقف می شود ، پس از سپری شدن زمان مشخص شده اجرای برنامه از خط بعد ادامه پیدا می کند .

پیکره بندی LCD

اتصال پایه های LCD میکرو

پایه های LCDای اتصال به پایهای میکرو بصورت زیر پیکره بندی می شود .

,DB٤=PN,DB٥=PN,DB٦=PN,DB٧=PN,E=PN,RS=PN

CONFIGLCD=PIN

PN=پایه ای دلخواه از میکرو که پایه LCDبه ان اتصال می یابد به طور مثال

PORTB.٧

تعیین نوع :LCD

LCDTYPE می تواندانواع زیر باشد :

CONFIG LCD =LCD type

٤*٤٠= دارای ٤٠ ستون و ٤ سطر

la*١٦= دارای ١٦ ستون و١ سطر است . این نوع LCD نوع خاصی است که به صورت LCD٨*٢ استفاده می شود که دارای خط دومی در ستون نهم یا ادرس &HB است .

١٦*٢ = دارای ١٦ ستون و٢ سطر است که بصورت پیش فرض قرار می گیرد .

اگر ازاین LCD استفاده نیازی به تعیین LCD نیست .

ونیز میتواند از انواع ٤*١٦-٢*٢٠-٤*٢٠-١*١٦

باشد.

پیکره بندی باس LCD :

CONFIGLCDBUS= constant

در صورتی که بخواهیم از انتقال داده به LCD به صورت ٤ بیتی (پیش فرض) یا ٨ بیتی استفاده نماییم که CONSTANT می تواند عدد ٤ برای انتقال اطلاعات بصورت ٤ بیتی و عدد ٨ برای انتقال اطلاعات بصورت ٨ بیتی باشد . زمانی که از انتقال داده ٤ بیتی استفاده می نمایید نیازی به نوشتن این پیکره بندی نیست .

دستورات و توابع مربوط به LCD

دستور LCD

این دستور یک یا چند عبارت ثابت یا متغیر را برروی LCD نمایش می دهد .

```
LCD x
LCD "constant "
```

X متغیر و constant ثابتی است که نمایش داده می شود . برای نمایش چند عبارت پشت سر هم بین آنها علامت ; (semicolon) را قرار می دهیم .

LCD a;b۱ " constant "

دستور CLS

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

دستور DISPLY

توسط این دستور می توانید صفحه نمایش را روشن یا خاموش کنید .

دستور CURSOR

توسط این دستور می توان مکان نمای LCD را تنظیم کرد .

Cursor on / Off Blink / No Blink

شما می توانید روشن (ON) یا خاموش (OFF) و چشمک زدن (BLINK)یا چشمک نزدن (NO BLINK) . مکان نما را تنظیم کنید در حالت پیش فرض مکان نما در حالت روشن و چشمک نزدن است .

دستور HOME

این دستورات مکان نما را در اولین ستون سطر اول ، سطر دوم ، سطر سوم یا چهارم قرار می دهد .

HOME UPPER /LOWER / THERD / FOURD

دستورات فوق را بصورت ساده شده زیر نیز می توان نوشت :

HOME U / L /T/ F

اگر دستور HOME به تنهایی نوشته شود مکان نما در سطر و ستون اول قرار می گیرد.

دستور LOCATE

این دستور مکان نما را به مکان دلخواه در صفحه LCD می برد .

LOCATE X ,Y

X ثابت یا متغیری از (٤ – ١) مشخص کننده سطر و Y ثابت یا متغیر از ٦٤ – ١) که مـشخص کننـده ستون LCD اسست .

دستور SHIFT CURSOR

این دستور مکان نمای LCD را یک واحد به چپ یا راست انتقال می دهد .

SHIFT CURSORLEFT / RIGHT

دستور SHIFT LCD

SHIFT LCDEFT / RIGHT

این دستور کل صفحه نمایش LCD را یک واحد به چپ یا راست انتقال می دهد .

دستور LOWERLINE

LOWERLINE

این دستور مکان نما را به خط پایین تر می برد .

دستور UPPRILIN

UPPRILIN

این دستور مکان نما را به خط سوم می برد .

دستور FOURTH LINE

در صورت استفاده از LCD چهار سطر این دستور کرزر را به اول خط چهارم می برد این دستور فقط برای LCD های چهار خط معتبر است .

تابع DEFLCDHAR

با این دستور می توانید حرف یا علامتی که خودتان در منوی TOOLS و قسمت LCD DESIGNER محیط BACCOM طراحی نموده اید 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.

DEFLCDHAR ? , r۱ , r۲ ,r۳ ,r٤,r٥, r٦ , r۷ ,r۸

R۱ تا R۸ با توجه به طراحی ، توسط نرم افزار نوشته می شوند و شما می توانید به جای ؟ عددی بین ۰ تا ۷ قرار دهید . بدین صورت شما می توانید تا ۸ کارکتر را طراحی کنید و بر روی LCD نمایش دهید . نمایش کارکتر طراحی شده توسط دستور LCDCHR (?) بعد از دستور CLS انجام می گیرد .

ارسال داده سریال در حالت UART سخت افزاری

پیکره بندی SERIALOUT

زمانی که بخواهید از محیط TERMINAL EMULATOR برای نماتیش داده گرفته شده از پورت سریال استفاده نمایید ، می توانید از این پیکره بندی استفاده نمایید . توسط این دستور می توان برای داده های ارسالی به پورت سریال کامپیوتر بافری توسط UART سخت افزاری در نظر گرفت . حافظه بافر از حافظه ARAM تامین می شود .

CINFIG SERIALOUT = BUFFERD , SIZE , = size

Size مشخص کننده تعداد بایت بافر است .

دستور PRINT

این دستور داده به پورت سریال ارسال می کند .

PRINT var ; "constanst "

Var مقدار عددی و constans رشته اختیاری است که به پورت سریال RS-۲۳۲ فرستاده می شود . شما با استفاده از علامت ; می توانید در یک خط مقدارهای مختلفی را بفرستید . پایه های سریال RS-۲۳۲ میکرو می توانند به پورت سریال کامپیوتر وصل شود ، در این صورت شما می توانید از TERMINAL EMULATOR نرم افزار BASCOM به عنوان خروجی داده سریالی استفاهد نمایید . دستور print به تنهایی باعث ایجاد یک خط خالی TERMINAL EMULATOR می شود .

دستور PRINTBIN

با این دستور محتوای متغبر var و متغیر اختیاری varn به باینری تبدیل می شوند و به پورت سریال ارسال می شوند . همچنین چندین متغیر می تواند فرستاده شود که با علامت : از یکدیگر جدا می شوند.

دستور WAITKEY

این دستور تا زمانی که در بافر سریال (UART) کاراکتری دریافت شود منتظر می ماند ، پس از دریافت کاراکتر آن را در متغیر VAR می ریزد و برنامه ادامه می یابد .

دستور INKEY

این دستور مقدار اسکی اولین کاراکتر دریافت شده از پورت سریال (RS-۲۳۲) را بر می گرداند .

Var = inkey ()

Var می تواند Byte, Integer, Word و یا Long باشد.

نکته : در صورت خالی بودن بافر دستور Inkey عدد صفر را بر می گرداند .

دستور INPUT

زمانی از محیط Terminal Emulator استفاده می کنید این دستور به کار برده می شود . توسط این دستور می توانید زمانی که Terminal Emulator محیط BSCOM هستید از صفحه کلید کامپیوتر به عنوان ورودی استفاده کنید به این صورت که در زمان وارد کردن داده از طریق صفحه کلید در محیط Terminal داده از طریق پورت سریال به میکرو ارسال می شود . توسط دستور Input شما می توانید هر رشته ای را وارد کنید و

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

داده گرفته شده از صفحه کلید در متغیر var و متغیر اختیاری varn قرار می گیرد . به صورت پیش فرض داده در محیط Termina نمایش داده می شود ولی در صورت استفاده از دستور NOECHO داده گرفته نشده از صفحه کلید در محیط Terminal نمایش داده نخواهد شد .

دستور INPUTBIN

این دستور داده باینری را ازپورت سریال (RS-۲۳۲) می خواند .

INPUT BIN var ۱[,var۲]

زمانی که از محیط EMULATOR TERMINAL استفاده می کنید این دستور بکار برده می شود . توسط این دستور می توانید زمانی که در EMULATOR TERMINAL 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 var۱ ومتغیر اختیاری var۲ جای میگیرد که اگر به عنوان BYTE تعریف شده باشد یک بایت به ازاء INTEGER دو بایت و اگر یک آرایه تعریف شود به تعداد آرایه ها کاراکتر دریافت می کند . این دستور برای گرفتن تعداد بایتها درهمان خط می ایستد.

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 بکار 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 توانید بین ۹-۰ 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 هب صورت زیر تعیین می شود :

- اگر var یا varn بایت تعریف شده باشند ورودی هر یک نهایتاً ۲ کاراکتر می تواند طول داشته باشد.
- اگر var یا varn , Integer یا Word تعریف شده باشند ورودی هر یک نهایتاً ٤ کاراکتر می تواند طول داشته باشد .
- اگر var یا varn , Long تعریف شه باشند ، ورودی هر یک نهایتاً ۸ کاراکتر می تواند طول داشته باشد.

ارسال داده ها در حالت Uart نرم افزاری

دستور PRINT

این دستوز از طریق Uart نرم افزاری ، داده به پورت سریال RS- ۲۳۲ ارسال می کند .

Print # Channel , var

که var متغیر ، مقدار عددی یا رشته اختیاری است که به پورت سریال RS- ۲۳۲ ارسال می شود . پایه های تعیین شده برایUart نرم افزاری می تواند به پورت سریال کامپیوتر وصل شود ، در این صورت شما می توانید TERMINAL EMULATOR نرم افزار BASCOM به عنوان خروجی استفاده نمایید .

نکته : استفاده از Print var در کنار دستور var

Print # Channel ، به معنای ارسال داده از پورت سریال اصلی میکرو (پایه TXD) است.

دستور PRINTBIN

Printbin # channel , var [; varn]

Channel شماره کانال باز شده توسط Uart نرم افزاری است . محتوای متغیر avr و متغیر اختیاری varn به باینری تبدیل می شوند و پورت سریال نرم افزاری فرستاده می شوند . همچنین چندین متغیر می تواند فرستاده شود که با علامت : از یکدیگر جدا می شوند .

دریافت داده در حالت UART نرم افزاری

دستور WAITKEY

Var = Waitkey (# channel)

این دستور تا زمانی که دربافر UART نرم افزاری (Soft Uart) با شماره کانال (channel) کاراکتری ردیافت 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 .

دستور INKEY

این دستور مقدار اسکی 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 ، Word , Integer , Byte و یا Long باشد .

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

دستور Input

این دستور همانند Input در Uart سخت افزار 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 رد صورت وارد کردن داده توسط صفحه کلید کامپیوتر در محیط TERMINAL EMULATOR نمایش داده نمی شود.

دستور(Input (# Channel

این دسترو مقدار باینری را از پورت سریال (RS – ۲۳۲) می خواند .

Inputbin # Channel , var ۱ [, var]

زمانی که از محیط TERMINAL EMULATOR محیط BASCOM هستید از صفحه کلید به عنوان ورودی استفاده کنید بدین صورت که در زمان وارد 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 Channle میکرو ارسال می کند . مقدار باینری خوانده شده از پورت سریال نرم افزاری در متغیر var۱ و متغیر اختیاری var۲ 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 بایت ها در همان خط می ایستد .

دستور (Inputhex (# CHANNEL

دستور اجازه می دهد که در هنگام اجرای برنامه از صحفه 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 ۹-۰ و A_F باشد در نتغیر رشق و

متغیر اختیار varn قرار می دهد و داده از طریق پورت سریال به Uart نرم افزاری با شماره کانال Channel میکرو راسال می کند . این دستور NOECHO است و در صورت وارد کردن داده توسطصفحه کلید کامپیوتر داده ورودی در محیط TERMINAL EMULATOR نمایش داده نمی شود . تعداد بایت ورودی بستگی به تعریف متغیر های var و varn دارد که به صورت زیر تعیین می شوند:

- اگر var یا varn بایت تعریف شده باشند ورودی هر یک نهایتاً ۲ کاراکتر می تواند طول داشته باشد .
- اگر var یا Integer , varn یا Word تعریف شده باشند ورودی هر یک نهایتا ٤ کاراکتر می تواند طول داشته باشد .
- اگر var یا Long , varn تعریف شده باشند ، ورودی هر یک نهایتاً ۸ کاراکتر می تواند طول داشته باشد.

مبدل آنالوگ به دیجیتال

(ANALOG TO DIGITAL COVERTOR)

متدوال ترین انواع ADC ها به قرار زیر است :

۱- مبدل ADC نوع شمارشی

۲- مبدل ADC از نوع تقریب ها متوالی

۳- مبدل ADC با مقایسه موازی

٤- مبدل ADC دو شیبه

مبدل نوع SUCCESSIVE – APPROXIMATION

مبدل آنالوگ به دیجیتال داخلی میکرو های AVR که ADC دارند از این نوع است به همین دلیل قصد داریم در مورد این نوع ADC مختصری توضیح دهیم . به جای شمارنده در این طرح از یک میکرو کنترلر یا میکرو پروسسور استفاده می شوند . با برنامه ای MSB یک شده و در یک DAC به آنالوگ تبدیل شده خروجی DAC در مقایسه گر با سیگنال آنالوگ مقایسه می شود و اگر خروجی DAC بزرگتر باشد MSB صفر شده و MSB بعدی ۱ می شود و مقایسه می شود و اگر کوچکتر باشد MSB ۱ باقی می اند و MSB بعدی ۱ می شود و این عمل به همین ترتیب ادامه پیدا می کند تا سیگنال آنالوگ خروجی DAC با سیگنال آنالوگ حاضر در پایه ADC برابر شود .

مبدل آنالوگ به دیجیتال داخلی میکرو

خصوصیات مبدل آنالوگ به دجیتال داخلی AVR به شرح زیر است :

- وضوح ۱۰ بیتی
- صحت مطلق ±۲LSB
- زمان تبدیل (CONVERSIONTIME) ۲۶-۶۵ ms
- وضوح KSPS ۱۵ در بالاترین حد
- کانلهای مولتی پلکس شده
- مدهای تبدیل SINGLE و FREE
- ولتاژ ورودی از ۰V 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 پایه های پورت اجازه می دهدکه به عنوان یک ورودی مبدل آنالوگ به دیجیتال عمل نماید . مبدل داخلی میکرو دارای

وضوح ۱۰بیتی است و برای تبدیل با این وضوح نیاز به فرکانس کلاکی بین ۵۰ KHZ تا ۲۰۰ KHZ دارد و این کلاک را از تقسیم فرکانس کریستال تامین می کند . در صورت که نیاز به وضوح بالا (کمتر از ۱۰ بیت) نیست می توان کلاکی بالاتر از ۲۰۰ KHZ به آن اعمال کرد . ADC دارای یک SAMPLE AND HOLD است که باعث می شود ولتاژ ورودی ADC در زمان تبدیل در سطح ثابت نگه داشته شود تا عملیات تبدیل با دقت بیشتر انجام شود .

ADC دارای دو منبع ولتاژ آنالوگ مجزا است . AVCC و AGND که AGND بایستی به زمین یا ولتاژ آنالوگ متصل شود و AVCC نباید بیشتر از ۰,۳V ± نسبت به VCC اختلاف داشته باشد . ولتاژ مرجع (VOLTAGE REFRENCE) خارجی در صورت وجود باید به پایه AREF وصل شود که این ولتاژ بایستی بین ولتاژ موجود بر روی پایه های AGND – AVCC باشد . در غیر این صورت به VCC وصل می شود ADC مقدار آنالوگ ورودی را با تقریب متوالی به مقدار دیجیتال ۱۰ بتی تبدیل می کند . کمترین مقدرا ، نشان دهنده مقدار آنالوگ موجود در پایه AGND و بالاترین مقدار نشان دهنده ولتاژ پایه AEF منهای یک LSB است .

به طور مثال اگر پایه به ولتاژ AREF = ۳,۵ V و AGND = ۰V وصل شده ابشد ، مقدار دیجیتال شده ۱۰۲۳ نشان دهنده ولتاژ ۳,۵V و مقدار ۰ نشان دهنده ولتاژ ۰,۰V بر روی پایه مبدل ADC انتخاب شده است . ADC دارای دو مد تبدیل SINGLE و FREE است . مد SINGLE بایستی توسط کاربر پیکره بندی و کانال دلخواه برای نمونه برداری انتخاب شود . در مد FREE ، ADC با یک ثابت نمونه برداری ، رجیستر داده ADC را UPDATE می کند .

مدارات دجیتالی در داخل و خارج میکرو ایجاد نویز کرده و ممکن است در صحت اندازه گیری ADC تاثیر بگذارند . اگر صحت و دقت تبدیل قابل قبول نیست با استفاده از تکنیک های زیر می توان نویز را کاهش داد.

۱- بخش آنالوگ چیپ و تمام قسمت های آنالوگ باید دارای زمین جداگانه باشند . این زمین ها با زمین دیجیتال به وسیله یک مسیر به هم متصل می شوند

۲- مسیر های آنالوگ را می توانید کوتاه کنید و از تماس نداشتن مسیر های آنالوگ و زمین های آنالوگ اطمینان پیدا کنید و آنها را از مسیر های دیجیتال با فرکانس بالا دور کنید .

۳- پایه AVCC میکرو را به VCC با فیلتر پایین وصل می کنیم .

٤- از مدهای SLEEP و حالت ADC NOIS CANCELER برای کاهش نویز القاء شده توسط CPU استفاده کنید .

پیکره بندی ADC در محیط BASCOM

مبدل آنالوگ به دیجیتال باید برای میکروهایی که ADC دراند توسط این دستور پیکره بندی شود تا بتوان از آن استفاده نمود . مبدل داخلی میکرو دارای وضوح ۱۰ بیتی است و برای تبدیل با این وضوح نیاز به کلاکی بین ۵۰ KHZ تا ۲۰۰KHZ دارند و ایم کلاک را از تقسیم کریستال تامین می کند .

```
COMFIG ADC = SIGLE / FREE , PRESCCALER = AUTO
REFENCE = OPTIONAL
```

CONFIG ADC = SINGLE / FREE : برای تبدیل سیگنال آنالوگ خود به دیجیتال می توانید از دو مد FREE و SINGLE استفاده نمایید. زمانی که مد SINGLE را انتخاب می کنید باید از دستور ()GTADC استفاده کنید .

PRESCALER : ایـــن گزینـــه کـــلاک ADC را مـــشخص مـــی کنـــد . بـــا قـــرار دادن PRESCALER = AUTO کامپایلر با توجه به فرکانس اسیلاتور ، بهتـرین کـلاک را بـرای ADC تعیین می کند دیگر مقادیر معتبر ۲،٤،۸،۱۵،۳۲،٦٤ یا ۱۲۸ می باشد.

: REFERENCE = OPTIONAL

گزینه ای اختیاری برای ولتاژ مرجع (VOLTAGE REFERENC) در بعـضی از میکروهـا از جمله MEGA۸ است . OPTIONAL می تواندگزینه های زیر باشد :

OFF : برای خاموش کردن ولتاژ مرجع داخلی (Reference Turned Off Internal) و استفاده از ولتاژ موجود بر پایه AREF به عنوان ولتاژ مرجع انتخاب می شود .

AVCC : زمانی که ولتاژ پایه AVCC به عنوان ولتاژ مرجع در نظـر گرفتـه مـی شـود . در ایـن حالت اتصال پایه های AVCC و AREF به این ترتیب است .

: INTERNAL

زمانی که ولتاژ مرجع داخلی ۲,٥٦ V با خازن خارجی بر روی پایه ARFE استفاده شود . انتخاب این گزینه برای میکروهایی که ولتاژ مرجع داخلی ندارند ، هیچ تاثیری ندارد .

دستور GETADC

با این دستور سیگنال آنالوگ وارد شده به کنالهای (۷-۰) به مقدار دیجستال تبدیل می شود و در متغیر var از نوع داده Word قرار می گیرد .

```
Var = Getadc ( Channel )
```

Var نتیجه تبدیل و channel کانال مبدل آنالوگ به دیجیتال داخلی انتخاب شده است که می تواند بین ۰ تا ۷ باشد. این دستور فقط برای چیپهای avr که دارای مبدل آنالوگ به دیجیتال مـی

توانید به عنوان پورتهای ورودی و خروجی استفاده کنید وای هنگامی کـه پـورت بـه عنـوان adc پیکره بندی می شود دیگر نمی توانید از آن به عنوان I/O استفاده کنید.

دستورات Start و Stop

توسط دستئر ADC START , ADC شروع به نمونه برداری سیگنال آنالوگ کرده و توسط STOP ADC تغذیه را از ADC بایستی نوشته شود .

مثال

```
$ Regfile = " M۳۲ DEF . dat "

Config Ade = single , prescaler = Auto ' Now Give Power To  The Chip
Stop Ade
Start Ade        ' Wite Stop Ade , You Can Remove The Power From The Chip
Dim W As Word , Channel As Byte
Channel ۱ = ۰

Do
W = Getade (Channel )   ' Now Read  A / D     Value   from  Channel ۰ to ۷

Print " Channel " : Channel : " Value " :W
Incr Channel
If channel > ۷ Then Channel = ۰

Loop
End
```

کار با وقفه ADC

زمانی که کار نمونه برداری ADC از سیگنال آنالوگ به پایان رسید ADC ، پرچم اتمام تبدیل خود به نام ADC را یک می کند . با فعال کردن وقفه سراسری با دستور Enable Interrupts و فعال کردن وقفه ADC با دستور Enable ADC می توان به زیر برنامه وقفه ADC یا ISR مربوطه است . می توان برای کاهش نویز سیستم میکرو را در زمان نممونه برداری در مد Idle یا ADC Noise Reduction قرار داد و سپس میکرو خود کار با بالا رفتن پرچم وقفه با بالا رفتن پرچم وقفه اتمام تبدیل ADC از این مد بیدار شده و مقدار دیجیتال شده را در متغیر نوع Word قرار می دهد .

تغذیه :

قسمت تغذیه به منظور فراهم کردن ولتاژ پنج ولت DC می باشد که جهت زاه اندازی IC میکرو کنترلر و کلیه مدارات سازگار با TTL به کار می رود . ورودی قسمت تغذیه می تواند یک سیگنال AC یا DC باشد که الزاماً به اندازه چند ولت از ۵+ بیشتر است . این ورودی توسط یک سوئیچ ON- OFF قطع و وصل می شود . در حالتی که سوئیچ روشن است و ورودی وارد یک پل دیود شارژ می شود این خازن به علت ظرفیت بالای خود باعث می شود ولتاژ تقریباً DC به دست می آید . که دارای ریپل است . برای ایجاد یک ولتاژ ۵ + کاملاً DC از یک IC تنظیم کننده ولتاژ ۷۸۰۵ استفاده می کنیم که ورودی دارای اعواج را به یک ولتاژ کاملاً مستقیم ۵ ولت تبدیل می کند در خروجی تنظیم کننده ولتاژ یک خازن ۱۰ mf قرار داده شده که به اندازه ۵ ولت شارژ خواهد شد که نوسانات خروجی ناشی از تغییرات بار را کاهش می دهد . به منظور نشان دادن روشن بودن مدار تغذیه از یک LCD استفاده شده که توسط یک مقاومت در حدود ۳۳۰ اهم به خروجی تنظیم کننده ولتاژ وصل شده است .

دستورات کار با LCD

دستور CLS

این دستور تمام صفحه نمایش LCD چه قسمتی متنی و چه گرافیکی را پاک می کند .

دستور CLSGRAPH

این دستور فقط قسمت گرافیکی را پاک می کند .

دستور CLS TEXT

این دستور فقط قسمت متنی را پاک می کند .

دستور LCD

این دستور برای نوشتن متن بر روی LCD استفاده می شود. این دستور همانند دستور LCD برای LCD های ماتریسی عادی عمل می کنند .

دستور PSET X, Y, COLOR

این دستور یک PIXEL را در مختصات (X , Y) به ازای COLOR = ۰ خاموش و به ازای COLOR = ۱ روشن می کند . X از ۰-۲۳۹ و Y از ۰-۱۲۷ می تواند تغییر کند .

دستور LOCATE ROW ، COLUMN

این دستور مکان نما را در مکان سطر (ROW) و ستون (COLUMN) مشخص شده قرار می دهد . ROW می تواند از ۱ تا ۱۶ تغییر کند . تغییرات COLUMN بستگی به انتخاب MODE دارد که می تواند از ۱ تا ۴۰ تغییر کند .

دستور CULRSOR ON / OFF BLINK / NOBLINK

برای قسمت هایی متنی استفاده می شود . مکان نما می تواند در حالتهای ON یا OFF و چشمک زدن (BLINK) یا چمک نزدن (NOBLINK) باشد .

دستور LINE (X۰ , Y۰)- (X۱ , Y۱) , COLOR

با این دستور از PIXEL اول با مختصات (X۰ , Y۰) به PIXEL دوم با مختصات (X۱,Y۱) خطی با رنگ COLOR کشیده می شود . ۰= COLOR خط را پاک کرده و به ازای ۲۵۵ = COLOR خطی با رنگ سیاه رسم خواهد شد .

دستور CIRCLE (X۰,Y۰) , RADIUS , COLOR

این دستور دایره ای به مختصات مرکزیت (X۰,Y۰) و شعاع RADIUS و رنگ COLOR رسم خواهد کرد . ۰ = COLOR دایره را پاک کرده و به ازای ۲۵۵ = COLOR دایره با رنگ سیاه رسم خواهد شد .

دستور SHOWPIC X,Y , LABIE

برای نمایش عکسی که در منوی TOOLS و قسمت GRAPHIC CONVERTER ذخیره کردیده اید استفاده می شود . X مکان قرار گیرری افقی و Y مکان قرار گیری عمودی عکس را نشان می دهد . LABLE نام بر چسبی 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 فورمت BGF و با نام دلخواه FILE در کنار برنامه اصلی ذخیره شده است ، دارد .

دستورات مربوط به ارتباط SPI

دستور SPIINIT

توسط این دستور پایه های به کار برده شده در ارتباط SPI , INITIAL می شوند . این دستور بعد از پیکره بندی SPI بایستی برای قرار گیری پایه های استفاده شده در جهت مناسب نوشته شود .

دستور SPIIN

SPIIN var , bytes

توسط این دستور به تعداد bytes از باس SPI بایت دریافت می شود و در متغیر VAR قرار می گیرد .

دستور SPIOUT

SPIOUT var , bytes

با این دستور به تعداد bytes داده var به باس SPI ارسال خواهد شد .

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 می گیرد .

دستور I۲C SEND

این دستور داده را به خط یا باس I۲C (WIRF -۲) می فرستد .

I۲ csend Slave , Var

I۲csend Slave , Var , bytes

: SLAVE

عدد ثابت یا متغیر نوع Word , Integer , byte که حاوی آدرس slave است .

: VAR

داده ای قصد ارسال به خط یا باس ۱۲C را داریم که می توانید یک ، Word , Integer , Byte یا عدد ثابت باشد .

: Bytes

مشخص کننده تعداد بایت دلخواه برای ارسال به خط یا باس سریال ۱۲C است .

دستور ۱۲CSTART,۱۲CSTOP,۱۲CRBYTE,۱۲CWBYTE

۱۲cstart

۱۲cstop

۱۲crbyte Var , Ack / Nack

: ۱۲CSTART

که باعث ایجاد (START CONDITION) در پرتکل ارتباطی ۱۲C می شود .

: ۱۲CSTOP

که باعث ایجاد پایان (START CONDITION) در پرتکل ارتباطی ۱۲C می شود .

: ۱۲CRBYTE

یک بایت از خط یا باس ۱۲C می گیرد . var متغیری است که داده را از خط یا باس ۱۲C می گیرد و ack را برای دریافت بیش از یک بایت و NACK را زمانی که آخرین بایت را می خوانیم ، ایجاد می کنیم .

: ۱۲CWBYTE

یک بایت به خط یا باس ۱۲C می فرستد . var متغیر یا ثابتی است که به خط یا باس ۱۲C ارسال می شود. WATCHDOG اجرا شده و تایمر WATCHDOG مدام با عددی ۰۰$ ری ست می شود .

Next
End

وقفه ها

دستورات ENABLE , DISABLE

DISABLE interrupt
ENABLE interrupt

: DISABLE

این دستور برای غیر فعال کردن وقفه استفاده می شود .

: ENABLE

این دستور برای فعال کردن وقفه استفاده می شود .

: ENABLE

این دستور برای فعال کردن وقفه استفاده می شود .

نکته : باری فعال کردن هر یک از وقفه ها 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 وقفه ها را می دهد .

دستور ON INTERRUPT

زمانی که وقفه اتفاق بیافتد سیستم اجرای برنامه را متوقف می کند و به وقفه پاسخ می دهد و به بر چسبی که برای آن وقفه تعریف شده پرش می کند و بعد از برگشت اجرای برنامه ادامه پیدا می کند .

ON interrupt label [NOSAVE]

در دستور فوق label نام برچسبی است که به هنگام وقوع وقفه interrupt برنامه به آن پرش می کند و بقیه وقفه ها را غیر فعال می کند تا زمانی که از برنامه وقفه خارج شود . به کار بردن گزینه اختیاری no save باعث می شود که هیچ کدام از ریجیستر ها (رجیسترهای (R تا R۱ و R۱٦ تا R۳۱) ذخیره نشوند و درون برنامه وقفه تغییر یابند ولی در صورت استفاده نکردن از این گزینه تمام رجیسترهای استفاده شده برای کل برنامه تغییر نمی کند . در ضمن برای برگشت از وقفه نیاز به Return داریم ، اگر از چندین Return استفاده کنیم ، اولین Return که داخل شرط با حلقه نباشد به عنوان RET۱ (یعنی برگشت از وقفه استفاده می شود) و بقیه به عنوان Return استفاده می شود .

نکته :

امکان اینکه شما برای وقفه های الویت تعیین کنید نیست ، و هر وقفه ای که در آدرس پایین تر حافظه نوشهت شده باشد دارای الویت بالاتر است .

دستور WRITEEEPROM

WRITEEEPROM Var , ADDRESS

محتوای متغییر VAR در آدرس ADDRESS حافظه EEROM داخلی نوشته می شود . بعد از دستور WRITEEEPROM با توجه VCC بایستی ٤ ms – ۲,۵ تاخیر ایجاد کنید تا عملیات نوشتن شود . آدرس می تواند یک عدد ثابت یا متغیر بسته به حافظه از نوع داده Word یا Byte باشد.

- شما همچنین می توانید متغیرهای آرایه ای برای EEPROM استفاده کنید

Dim var As Eram var type

که varitype مـی توانـد داده هـای نـوع single , string , byte , word , integer , long باشد .

- شما همچنین می توانید متغیر های آرایه ای برای EEPROM استفاده کنید .

Dim ar (۱۰) as Eram Byte

در این حالت ۱۰ بایت اول از حافظه EEPROM برای متغیر آرایه ای () ar در نظر گرفته می شود .

- همچنین شما می توانید داده خود را در آدرسی دلخواه در EEPROM قرار دهید

محتوای متغیر Eb در آدرس ۱۳ از حافظه EEPROM قرار می گیرد .

نکته : در صورت مشخص نکردن آدرس حافظـه ، داده هـا بـه ترتیـب نوشـتن در برنامـه ، از آدرس ۰ شروع به جای گرفتن در حافظه می کنند.

نکته : LSB داده در حافظه پایین تر EEPROM قرار می گیرد .

دستور READEEPROM

READEEPROM VAR , ADDRESS

توسط این دستور محتـوای EEPROM از آدرس دلخـواه ADDRESS خوانـده مـی شـود و در متغیر VAR از نوع داده BYTE ذخیره می شود . آدرس می تواند یک عدد ثابت یا یک متغیر بسته به حافظه از نوع داده WORD یا BYTE باشد .

۵-۲- مراجع

۵-۲-۱- **میکرو کنترلر های AVR** ، علی کاهه

۵-۲-۲- **مبانی بیسیک** ، حمزه اصغری

۵-۲-۳- مبانی **الکترونیک** ، سید علی میر عشقی

۵-۲-٤- **تحلیل و طراحی سخت افزار** ، مسعود بهر الدین

۵-۲-۵-**مقدمه ای بر AVR** ، محمد گرزین