به نام خدا

آیفون هوشمند

گردآوری

هادی رستمی سیاهگلی

کلمات کلیدی

میکرو کنترلر های ***AVR***، ارتباط سریال ***SPI***، نرم افزار ***BASCOM AVR***

چکیده

در مورد این پروژه باید گفت که سیستم یک آیفون هوشمند است که برای یک آپارتمان ۱۰ واحدی طراحی شده است . اصطلاح هوشمند بودن را به این علت به آن اختصاص داده ایم که قابلیت آنرا دارد که یک سری پیغامهای شخصی را در خود ذخیره کند .

فهرست مطالب

چکیده

در مورد این پروژه باید گفت که سیستم یک آیفون هوشمند است که برای یک آپارتمان ۱۰ واحدی طراحی شده است . اصطلاح هوشمند بودن را به این علت به آن اختصاص داده ایم که قابلیت آنرا دارد که یک سری پیغامهای شخصی را در خود ذخیره کند . البته این سیستم کاملاً ایده ال نیست و معایبی دارد که در آینده سعی میکنیم با کار بیشتر بر روی آن این معایب را رفع کرده و امکانات بیشتری را به آن اضافه کنیم تا بتوان آنرا برای واحدهای آپارتمانی بیشتر با آمکانات بیشتر و قابلیتهای ارزنده تر به کار برد . در مورد نرم افزار برنامه ریزی میکرو از نرم افزار ***bascom*** ویرایش ۱.۱۱.۷.۴ استفاده شده است که تمام میکروکنترولرهای ***avr*** را پشتیبانی کرده و برنامه نویسی آن به زبان ***basic*** صورت می گیرد و از نرم افزار های تحت ویندوز است و از میکروکنترولرهای ***avr*** نیز به عنوان قسمت اصلی سیستم استفاده شده است .

در فصل اول عملکرد کلی سیستم توضیح داده شده است . در فصل دوم به بررسی میکروهای ***avr*** پرداخته و مشخصات آنها را مورد توجه قرار داده ایم . در فصل سوم نرم افزار برنامه نویسی معرفی شده و منوها و امکانات آن گفته شده است . در فصل چهارم تجهیزات سخت افزاری سیستم پیکره بندی شده و ارتباط ***spi*** بررسی شده و دستورات مربوط به آنها معرفی شده است . در فصل آخر نیز برنامه های نوشته شده برای میکروها به زبان بیسیک ارائه شده و تعدادی از دستورات نیز توضیح داده شده و شمای کلی مدار نیز ارائه شده است .

مقدمه

الکترونیک از علوم بسیار گسترده است و روز به روز نیز در حال پیشرفت است . به نظر میرسد که این ترقی حد و مرزی نمی شناسد و همیشه سیر صعودی خود را طی می کند و ما نیز اگر می خواهیم با این پیشرفت جلو برویم باید با علم روز پیش برویم و از امکانات در دسترس حداکثر استفاده را ببریم .

یکی از امکانات خوبی که در اختیار دانشجویان الکترونیک و کامپیوتر وجود دارد میکروکنترولرها هستند که در انواع مختلف با ویژگی ها و امکانات مختلف در دسترس هستند . باید گفت ساخت بسیاری از پروژه های علمی و صنعتی آماتور و پیشرفته بدون استفاده از میکروکنترلرها غیر ممکن است و این به خاطر آن قدرت هوشمندی و برنامه پذیر بودن میکروکنترولر است .

میکروکنترولر را به کامپیوتر در یک تراشه تشبیه کرده اند و البته این تشبیه واقعاً درست است چون داخل یک تراشهء میکروکنترولر مانند یک میکروکامپیوتر ، واحد پردازندهء مرکزی ***(cpu)*** ، حافظهء برنامه ***(rom)*** ، حافظهء داده ***(ram)*** و واحدهای ورودی و خروجی برای ارتباط با دنیای خارج به عنوان بلوکهای اصلی وجود دارد ، یک سری قابلیت های دیگر نیز در میکروکنترولرها وجود دارد که بستگی به نوع میکرو و شرکتهای سازندهء آن دارد .

خاصیت اصلی میکروکنترولرها برنامه پذیر بودن آنها است و این گویای این واقیت است که باید تلفیقی از الکترونیک و کامپیوتر در کار باشد . بله درست است و این امکان را میدهد که سخت افزار را تا حد زیادی کاهش داده و بتوانیم پیچیده ترین سیستمها را پیاده کنیم . کاهش سخت افزار سیستم تنها اثر مطلوب نیست بلکه دهها خاصیت خوب دیگر نیز در کار است که کار با میکروکنترولر را برای هر دانشجوی الکترونیک و کامپیوتر جذاب و دوست داشتنی میکند .

میکروکنترولر چون مانند یک کامپیوتر عمل میکند بنابراین باید دارای قدرت مانور زیادی میباشد که این بستگی به علم و دانش استفاده کننده یا همان کاربر دارد که تاچه حد در این زمینه آگاهی دارد .

با توجه به برنامه پذیر بودن میکروکنترولر باید بتوان به نحوی با آن ارتباط برقرار کرد به عبارتی آن را برنامه ریزی کرد ، برای این کار زبان های برنامه نویسی استاندارد وجود دارد . بعد از طراحی و ساخت یک میکروکنترولر توسط شرکت سازنده نرم افزارهای برنامه نویسی مختلفی نیز برای آن طراحی میشود که اصولاً از کامپایلرهای استاندارد در این زمینه استفاده میشود مانند کامپایلرهای زبان ***c*** ، ***basic*** ، ***assebly*** و غیره .

حال با این تفاسیر سوال این است برای طراحی و ساخت یک پروژه بهترین میکروکنترولر کدام است و بهترین زبان برنامه نویسی برای آن کدام است ؟

باید گفت که این بستگی به نوع کار یا طرح مورد نظر دارد که تا چه حد پیچیده است و تا چه اندازه نیز با زبانهای برنامه نویسی آشنایی داریم و کدام کامپایلر بهتر نیازهای ما را برآورده میکند . انواع مختلفی از میکروکنترولرها در بازار وجود دارد مانند میکروهای سری ***8051*** ، میکروهای نوع ***avr*** شرکت ***atmel*** ، میکروهای ***pic*** و انواع دیگر.

یکی از انواع خوب آنها میکروهای ***avr*** میباشند که به علت داشتن کامپایلرهای قوی به زبان ***hll*** (زبانهای سطح بالا) مورد استفادهء دانشجویان قرار میگیرد .

ما با آگاهی هایی که در مورد میکروکنترولرها و برنامه ریزی آنها داشتیم و با علاقهء زیاد در این زمینه تصمیم به ساخت یک پروژهء جالب و عملی گرفتیم .

باید گفت که این پروژه ابتدا به صورت یک ایده مطرح شد و پس از بررسی جوانب و مراحل کار با پی ریزی اولیه بر مبنای میکروکنترولرهای ***avr*** کار را شروع کردیم .

۱. فصل اول، مشخصات کلی سیستم

در این قسمت می خواهیم عملکرد کلی مدار و قابلیتهای آنرا معرفی کنیم .

سیستم فوق برای یک آپارتمان ۱۰ واحدی طراحی شده است و بدین صورت عمل می کند که به عنوان مثال شخصی قصد مراجعه به یک واحد خاص را دارد ، برای کاربر یک صفحه کلید در نظر گرفته شده است که ارتباط بین او و سیستم است و از طرفی نیز نمایش دهنده ای ***(lcd)*** در نظر گرفته شده است . بر این اساس شخص مراجعه کننده می تواند اسامی تک تک واحدهای آپارتمانی را با زدن شماره آن واحد همراه با شماره واحد بر روی ***lcd*** مشاهده کند ، برای زنگ یک کلید بر روی صفحه کلید در نظر گرفته شده است که با زدن آن زنگ واحد مربوطه به صدا در می آید . اگر میزبان در منزل باشد مشخصا در را باز می کند اما اگر در منزل نباشد با زدن سه بار زنگ منزل شخص مراجعه کننده باید بتواند پیغامی مبتنی بر مراجعه خود در ساعت و تاریخ مشخص بگذارد .

برای اینکار سیستم به این صورت طراحی شده است که بعد از بار سوم زنگ زدن به طور اتوماتیک وارد قسمت پیغام گزاری می شود که در این مرحله شخص می تواند شماره تلفن خود را به عنوان شناسه خود وارد کند که البته حداکثر می تواند ۱۴ رقم باشد . برای این کار یک کلید ***delete*** نیز بر روی صفحه کلید در نظر گرفته شده است که اگر عددی اشتباه وارد شود می توان آنرا پاک کرد .

زمانی که اعداد بر روی ***lcd*** نشان داده می شوند همزمان نیز تک تک در حافظه ***eeprom*** نوشته می شود که با زدن کلید ***delete*** می توان تک تک این اعداد را پاک کرد و پس از ثبت کامل در حافظه ***eeprom*** داخلی میکرو و نمایش کامل بر روی ***lcd*** با زدن کلید ***enter*** به زیر برنامه پرش می شود که در آنجا اطلاعات مربوط به تاریخ و زمان این پیغام از میکرو دوم نمونه برداری می شود .

اینجا باید متذکر شویم که کل ظرفیت حافظه ***eeprom*** میکرو ***mega16*** ، ۵۱۲ بایت است که برای هر پیغام ۲۰ بایت در نظر گرفته شده است و از این ۲۰ بایت ،۱۴ بایت اول مربوط به شماره تلفن مورد نظر است و ۶ بایت بعدی برای اطلاعات تاریخ و زمان در نظر گرفته شده است و در کل می توان ۲۲ پیغام را وارد کرد که ادامه حافظه نیز برای متغیر های ***eeprom*** در نظر گرفته شده است .

بنابراین با زدن کلید ***enter*** ارتباط دو میکرو برقرار می شود و تاریخ وزمان پیغام مورد نظر نمونه برداری شده و در ۲۰ بایت مربوطه و در ادامه شماره وارده ذخیره میشود . اگر شخص قصد وارد کردن پیغامی را

نداشته باشد میتواند با زدن کلید ***exit*** از این قسمت خارج شود . در این حالت اطلاعاتی در حافظه ذخیره نمی شود .

بنابراین میکرو دوم نقش تنظیم کننده تاریخ و زمان را دارد که برای این کار برنامه ای نوشته شده است که تاریخ شمسی و زمان را تنظیم کرده و بر روی یک ***lcd*** نمایش می دهد که برای تنظیم زمان و تاریخ نیز سه عدد شستی در نظر گرفته شده است . برای ارتباط دو میکرو نیز از ارتباط ***spi*** استفاده شده است.

سیتم دارای دو عدد ***lcd*** است که یکی متصل به میکرو اول بوده و اطلاعات برنامه را نشان می دهد و دیگری در اختیار میکرو دوم بوده و تاریخ و زمان را نشان می دهد .

که با زدن زنگ مربوطه آن ***led*** روشن می شود .

برای نمایش زنگ هر واحد به صورت مجازی ۱۰ عدد ***led*** به پورتهای میکرو وصل شده است برای نمایش پیغامهای ذخیره شده نیز از دو کلید ***down*** و ***up*** استفاده شده است و میتوان کل پیغامها را مجزا مشاهده کرد و بدین صورت است که با زدن هر کدام از این کلیدها در هر مرحله ۲۰ بایت از حافظه ***eeprom*** خوانده می شود و به صورت صحیح بر روی ***lcd*** نمایش داده میشود .

در اینجا یک سوال مطرح می شود که تعداد ۲۲ پیغام تعداد کمی است و پس از پیغام آخر چه اتفاقی می افتد ؟

باید گفت که پس از تکمیل شدن آخرین پیغام ، پیغام بعدی جایگزین قدیمی ترین پیغام می شود و این روند به همین صورت ادامه می یابد ، یعنی همواره تعداد بیست و دو پیغام در حافظه ***eeprom*** به صورت ذخیره وجود دارد و به صورت چرخشی پیغامها جایگزین یکدیگر می شوند .

۲. فصل دوم، میکروکنترلرهای AVR

۲-۱. معرفی میکروکنترولرهای avr

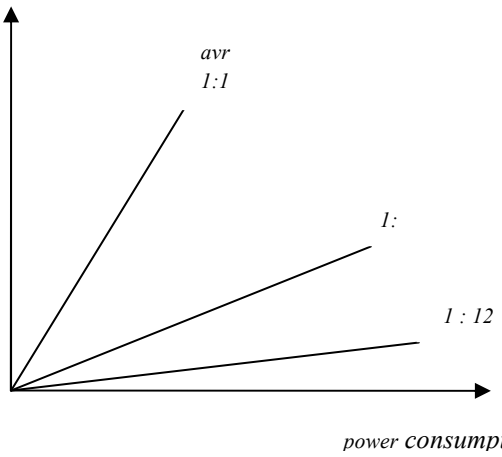
زبانهای سطح بالا یا همان ***hll (high level languages)*** به سرعت در حال تبدیل شدن به زبان برنامه نویسی استاندارد برای میکروکنترولرها حتی میکروهای کوچک ۸ بیتی هستند . زبانهای برنامه نویسی ***basic*** و ***c*** بیشترین استفاده را در برنامه نویسی میکروها دارند . شرکت ***atmel*** در تحولی جدید میکروکنترولرهای ***avr*** را روانهء بازار کرد که علاوه بر کاهش و بهینه سازی مقدار کدها به طور واقع عملیات را تنها در یک کلاک سیکل توسط معماری ***risc*** ، ***(reduced risc insteruction set computer)*** انجام میدهند و از ۳۲ رجیستر همه منظورهء داخلی استفاده می کنند که باعث شده ۴ تا ۱۲ بار سریعتر از میکروهای مورد استفادهء کنونی باشند .

تکنولوژی حافظهء کم مصرف غیر فرار شرکت ***atmel*** برای برنامه ریزی ***avr*** ها مورد استفاده قرار گرفته است در نتیجه حافظه های ***flash*** و ***eeprom*** در داخل مدار قابل برنامه ریزی ***(isp)*** هستند . ***avr*** ها به عنوان میکروهای ***rics*** با دستورات فراوان طراحی شده اند که باعث میشود کد تولید شده کم و سرعت بالاتری به دست آید .

باانجام تک سیکل دستورات ، کلاک اسیلاتور با کلاک داخلی سیستم یکی میشود . هیچ تقسیم کننده ای در داخل ***avr*** وجود ندارد که ایجاد اختلاف فاز کلاک کند . اکثر میکروها کلاک اسیلاتور به سیستم را به نسبت ۱:۴ یا ۱:۱۲ تقسیم می کنند که این خود باعث کاهش سرعت سیستم می شود ، بنابراین ***avr*** ها ۴ تا ۱۲ بار سریعتر و مصرف آنها نیز ۴ تا ۱۲ بار نسبت به میکروهای دیگر کمتر است زیرا در تکنولوژی ***cmos*** استفاده شده در میکروهای ***avr*** مصرف توان متناسب با فرکانس است .

نمودار زیر افزایش ***mips*** (میلیون دستور در ثانیه) را به علت انجام عملیات تک سیکل ***avr*** (نسبت ۱:۱) در مقایسه با نسبتهای ۱:۴ و ۱:۱۲ در دیگر میکروها نشان میدهد .

نمـــودار مقایســـه افـــزایش mips/power consumption avr بـــا دیگر میکروها



هدف ***atmel*** طراحی معماری بود که هم برای زبان اسمبلی و هم زبان ***hll*** مفید باشد . به طـور مثـال در زبانهای ***c*** و ***basic*** میتوان یک متغیر محلی به جای متغیر سراسری در داخل زیر برنامه تعریف کرد ، در این صورت فقط در زمان اجرای زیر برنامه مکانی از حافظهء ***ram*** برای متغیر اشغال می شود .

برای دسترسی به متغیرهای محلی و کاهش کد ، نیاز به رجیسترهای همه منظوره است . ***avr*** هـا دارای ۳۲ رجیستر همه منظوره هستند که مستقیماً به ***alu*** متصل شده اند و تنها در یک کلاک سیکل به ایـن واحد دسترسی پیدا می کنند . همه این رجیسترها ۸ بیتی هستند که سه جفت از ایـن رجیسـترها مـی توانند در کنار هم قرار گرفته و به عنوان رجیسترهای ۱۶ بیتی استفاده شوند .

از دیگر خصوصیات مهم میکروهای ***avr*** دارا بودن حافظهء ***eeprom*** داخلی برای استفادهء کاربر ، داشـتن مبدل آنالوگ به دیجیتال داخلی ، تایمر کانتر ، قابلت ارتباط با پروتکل های استاندارد و امکانات جـانبی دیگر که البته برای انواع میکروهای ***avr*** این امکانات و ویژگی ها اندکی فرق می کند .

میکروهای ***avr*** در چند سری به بازار عرضه شده اند که انواع اولیه میکروهای ***tiny avr*** بـوده کـه قابلیـت های کمی داشتند و برای پروژه های ساده تر استفاده میشوند ، سری بعدی میکروهـای ***avr*** بـا پیشـوند ***at90s*** هستند که دارای امکانات بیشتری نسبت به سری قبلی می باشند . نـوع پیشـرفته تـر میکروهـای ***avr*** نوع ***megaavr*** هستند که قابلیتهای خوبی دارند . این سری از میکروهـا در مـدلهای مختلـف تولیـد شده و با امکانات عالی قابـل تهیـه هسـتند . در ادامـه قصـد داریـم بـه معرفـی ، مشخصـات و امکانـات میکروهای ***atmega16*** و ***atmega8*** بپردازیم که در ساخت این پروژه از این دو میکرو استفاده شده است .

۲-۱-۱. میکروکنترولر atmega16

برای طراحی و ساخت قسمت اصلی پروژه از این میکرو استفاده شده است و سخت افزارهای اصلی سیستم به این میکرو وصل شده اند . در اینجا قصد داریم خصوصیات این نوع میکرو را بیان کنیم .

از معماری ***avr risc*** استفاده میکند

- کارایی بالا و توان مصرفی کم .
- دارای ۱۳۱ دستورالعمل با کارایی بالا که اکثراً دز یک کلاک سیکل اجرا می شوند .
- ۳۲ رجیستر کاربردی
- سرعتی تا ***16 mips*** در فرکانس ***16 mhz***
- حافظه ، برنامه وداده‌ء غیر فرار
- ***16kb*** حافظه‌ء ***flash*** داخلی قابل برنامه ریزی که قابلیت ۱۰۰۰۰ بار نوشتن و پاک کردن را دارد.
- ۱۰۲۴ بایت حافظه‌ء ***sram*** داخلی
- ۵۱۲ بایت حافظه‌ء ***eeprom*** داخلی قابل برنامه ریزی که قابلیت ۱۰۰۰۰۰ بار نوشتن و پاک کردن را دارد.
- قفل برنامه‌ء ***flash*** و حفاظت داده‌ء ***eerpom***
- قابلیت ارتباط ***jtag (ieee std.)***
- برنامه ریزی برنامه‌ء ***flash*** ، ***eeprom*** ، ***fues bits*** ، ***lock bits*** از طریق ارتباط ***jtag***

۲-۱-۱-۱. خصوصیات جانبی

- دو تایمر کانتر ***(timer/counter)*** ۸ بیتی با ***prescaler*** مجزا و مُد ***compare***
- یک تایمر کانتر ***(timer/counter)*** ۱۶ بیتی با ***prescaler*** مجزا و دارای مدهای ***compare*** و ***capture***
- ۴ کانال ***pwm***
- ۸ کانال مبدل آنالوگ به دیجیتال ***10*** بیتی
- کانال ***single-ended***
- دارای ۷ کانال تفاضلی در بسته بندی ***tqfp***
- دارای دو کانال تفاضلی با کنترل گین ***1x*** و ***10x*** و***200x***

- یک مقایسه گر آنالوگ داخلی
- ***watchdog*** قابل برنامه ریزی
- قابلیت ارتباط با پروتکل ارتباط دو سیمه ***(two-wire)***
- قابلیت ارتباط ***spi (serial peripheral interface)*** به صورت ***slave*** یا ***master***
- ***usarT*** سریال قابل برنامه ریزی

۲-۱-۱-۲. خصوصیات ویژهء میکروکنترولر

- ***power_ on reset circuit*** و ***brown_out*** قابل برنامه ریزی
- دارای اسیلاتور ***rc*** داخلی کالیبره شده
- دارای ۶ حالت ***sleep (adc noise reduction , extended standby , standby , power-save , idle , power down)***
- منابع وقفه ***(interrupts)*** داخلی و خارجی
- عملکرد کاملاً ثابت
- توان مصرفی پایین و سرعت بالا توسط تکنولوژی ***cmos***
- توان مصرفی در ***1mhz ، 3v ، 25˚c*** برای ***atmega16l***
- حالت فعال ***1.1 ma (active mode)***
- در حالت بیکاری ***0.35 ma (idle mode)***

در حالت ***power-down 1ua*** :

- ولتاژهای عملیاتی (کاری)
- ***2.7 v*** تا ***5.5 v*** برای ***atmega16 l***
- ***4.5 v*** تا ***5.5 v*** برای ***atmega16***

فرکانسهای کاری

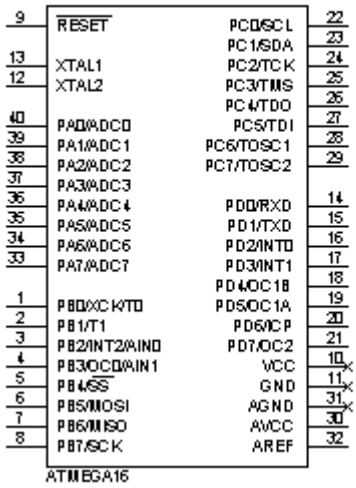
- ***0mhz*** تا ***8mhz*** برای ***atmega16l***
- ***0mhz*** تا ***16mhz*** برای ***atmega16***

خطوط ***i/o*** وانواع بسته بندی

- ۳۲ خط ورودی / خروجی ***(i/o)*** قابل برنامه ریزی
- ۴۰ پایه ***pdip***، ۴۴ پایه ***tqfp*** و ۴۴ پایه ***mlf***

ترکیب پایه ها ترکیب پایه های این میکرو در زیر نشان داده شده است:

این میکرو دارای ۴ پورت ۸ بیتی است که با نامهای ***PORTA*** ، ***PORTB*** ، ***PORTc*** و ***PORTD*** شـناخته می شوند . شمای کلی میکرو و شماره های پایه ها در شکل زیر مشخص شده است .



نمای پایه های میکرو ***MEGA16***

۲-۱-۱-۳. فیوز بیتهای *atmega16*

فیوز بیتها قسمتی از حافظهء ***flash*** هستند که امکاناتی را در اختیار کاربر قرار میدهند . بـه ایـن صـورت که تنظیماتی را می توان توسط آنها انجام داد مانند تنظیم کلاک سیستم و اینکه کلاک سیسـتم میکـرو چقدر باشد و از نوسان ساز داخلی استفاده کنیم یا خارجی ، تعیین عملکرد بعضـی از پایـه هـا در مـوارد خاص ، تنظیمات میکرو در ارتباط با سخت افزارهای دیگر و ... .فیوز بیت ها با پاک شدن میکرو ***(erase)*** پاک نمی شوند و حتی می توان آنها را توسط بیتهای قفل مربوطه قفل کرد .

میکرو ***mega16*** دارای دو بایت فیوز بیت است که هر کدام از این فیوز بیت ها کار خاصـی را انجـام مـی دهند و توسط خود محیط نرم افزار ***bascom*** میتوان فیوز بیتها را برنامه ریزی کرد . باید گفت که تغییـر فیوز بیتها باید با آگاهی کامل صورت گیرد چون با دستکاری بی مورد آنها ممکن است دیگر نتوان میکرو

را برنامه ریزی کرد که این خود میتواند به معنای سوختن میکرو باشد . توضیح کامل فیوز بیتها احتیاج به زمان بیشتری دارد و ار طرفی نیز کاربر در بیشتر مواقع نیازی به تنظیم فیوز بیتها ندارد زیرا تنظیم پیش فرض آنها صورت گرفته و نیازهای کاربر را برآورده می کند . منظور از برنامه ریزی فیوز بیتها ۰ شدن آنهاست زیرا اگر فیوزبیتی ۰ باشد یعنی برنامه ریزی شده است در غیر این صورت ۱ بودن آن به معنای برنامه ریزی نشدن است .

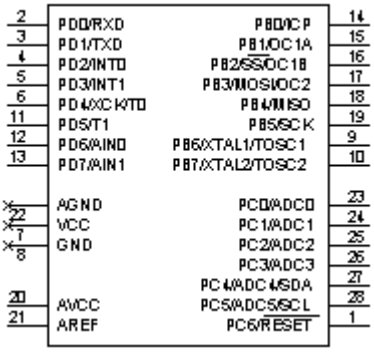
برای توضیح بیشتر در مورد فیوز بیتها و برای اطلاعات بیشتر می توانید به کاتالوگ ***(data sheet)*** خود میکرو مراجعه کنید .

۲-۲. میکروکنترولر ATMEGA8

این میکرو نیز از جهات بسیار زیادی شبیه به همان ***MEGA16*** است و فقط در یک سری مشخصات با آن فرق می کند . در اینجا از ذکر مواردی که با میکرو ***MEGA16*** یکی است خودداری میکنیم .

تفاوت های اصلی این میکرو در ***8KB*** حافظه ***FLASH*** و پورتهای کمتر است که دارای سه پورت به نامهای ***PORTB*** ، ***PORTC*** و ***PORTD*** است که پورتهای ***B*** و ***D*** ، ۸ بیتی و پورت ***C*** ، ۷ بیتی است .

این میکرو نیز دارای دو بایت فیوز بیت است . شمایی از این میکرو در زیر نشان داده شده است .



نمای پایه های میکرو ***MEGA8***

۳. فصل سوم، محیط برنامه نویسی bascom avr

در این بخش به بررسی کامپایلر ***bascom*** ویرایش ***1.11.7.4*** می پردازیم .

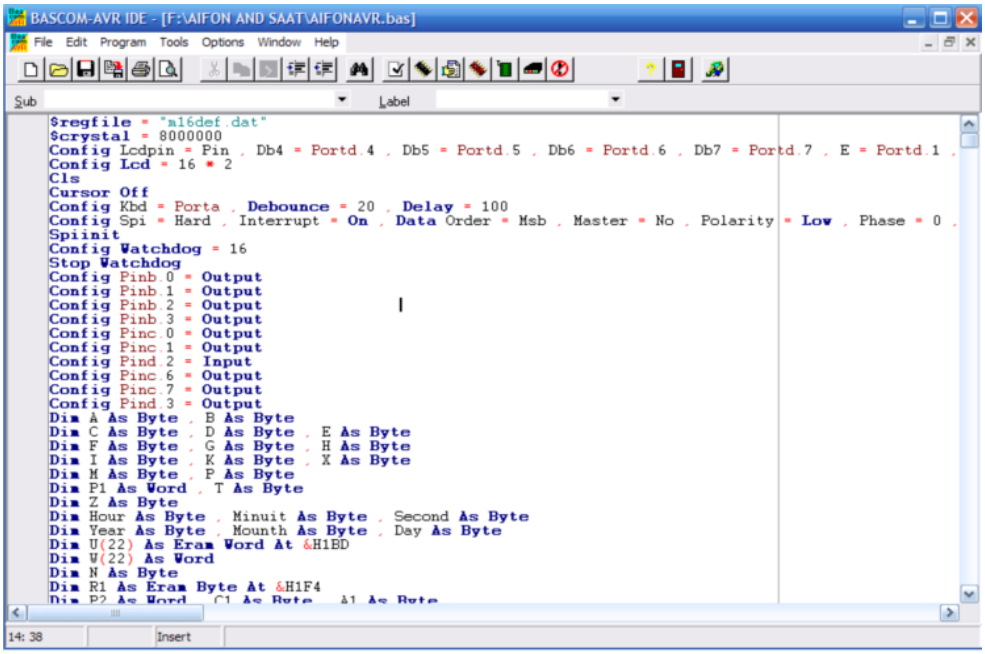
این نرم افزار کلیهء میکروهای ***avr*** را پشتیبانی می کند و برنامه نویسی آن به زبان بیسیک است . از قابلیت های بسیار خوب و ارزندهء این نرم افزار وجود تحلیل گر یا همان ***simulator*** داخلی است که کمک شایانی به یادگیری برنامه نویسان ***avr*** میکند . در اینجا قصد داریم نگاهی گذرا به چگونگی نصب وکار با این نرم افزار پرداخته و منوها و امکانات آنرا بررسی نماییم .

مراحل نصب این نرم افزار شبیه دیگر نرم افزار های تحت ویندوز است که به راحتی و در زمان کمی نصب می شود و البته کرک نیز دارد که کرک آن نیز باید به درستی اجرا شود . بنابراین به علت راحتی مراحل نصب از ارایهء جزییات بیشتر در این زمینه خود داری میکنیم .

محیط شبیه ساز این نرم افزار امکانات خوبی را در اختیار کاربر قرار می دهد مانند ورودی سیگنال آنالوگ برای ***adc*** ، مقایسه کنندهء آنالوگ ، ایجاد پالس بر روی پایه ای خاص از میکرو ، صفحه کلید ۴*۴ ، ***lcd*** ، ایجاد تمام وقفه ها به صورت اختیاری ، نوشتن و خواندن حافظهء ***eeprom*** و ***sram*** ، رویت تمام رجیستر ها و متغیر های محلی و سراسری ، اجرای برنامه به صورت خط به خط ، دیدن صفر و یک بودن پایه ها توسط ***led*** ، تغییر منطق پایه ای دلخواه و بسیاری امکانات دیگر .

۳-۱. معرفی منوهای محیط برنامه نویسی bascom

پس از اینکه برنامه را نصب کردید آنرا اجرا کرده و پنجرهء برنامه نویسی به صورت زیر ظاهر می شود .



۳-۱-۱. منوی file

ای منو در همهء برنامه های تحت ویندوز وجود دارد و امکاناتی از قبیل ایجاد فایل جدید***(file new)*** ، باز کردن فایل از قبل ذخیره شده ***(file open)*** ، بستن فایل ***(file close)***، ذخیرهء فایل ***(file save)*** ، نمـایش پرینت فایل ***(file print preview)*** ، پرینت فایل ***(file print)*** و خروج از فایل ***(file exit)*** را در اختیا کاربر قرار میدهد .

۳-۱-۲. منوی edit

گزینه های این منو به صورت زیر هستند :

edit undo

با این گزینه میتوان دستکاری اخیر در برنامه را از بین برد .

edit redo

با این گزینه میتوانید دستکاری اخیرتان را که از بین برده بودید برگردانید .

edit cut

با این گزینه شما میتوانید متن انتخاب شده را بریده و به محل جدیدی انتقال دهید .

edit copy

با این گزینه میتوان متن انتخاب شده را به محل جدیدی کپی کرد .

edit past

با این گزینه شما می توانید متنی را که قبلاً ***copy*** یا ***past*** کرده بودید در محل مورد نظر بچسبانید .

edit find

با این گزینه شما می توتنید متنی را در برنامه تان جستجو کنید .

edit replace

با این گزینه شما میتوانید متنی را جایگزین متن مورد نظر در برنامه کنید .

edit goto

با این گزینه شما می توانید 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 گزاری کرده و با دستور ***edit goto bookmark*** به آنها پرش نمایید.

۳-۱-۳. منوی program

این منو گزینه های مهمی را در بر دارد :

program compiLE

با این گزینه می توان برنامه را کامپایل کرد . کلید ***f7*** نیز همین کار را انجام میدهد . قبل از کامپایل شدن برنامه ذخیره می شود و فایلهای زیر بسته به انتخاب شما در ***option compiler setting*** تولید می شوند .

xx . bin فایل باینری که می تواند در میکروکنترولر ***program*** شود .

xx . dbg فایل ***debug*** که برای شبیه ساز ***bascom*** مورد نیاز است .

xx . obj فایل ***object*** که برای نرم افزار ***avr studio*** مورد نیاز است .

xx . rpt فایل گزارشی .

xx . hex فایل هگزا دسیمال اینتل که برای بعضی از انواع پروگرمرها مورد نیاز است .

باید متذکر شد که اگر خطایی در برنامه باشد شما پیغام خطا را در یک کادر محاوره دریافت خواهید کرد و کامپایلر متوقف خواهد شد . با کلیک روی آن می توان به خطی از برنامه که خطا در آن رخ داده است پرش کرد .

program syntax check

بوسیلهء این گزینه برنامهء شما برای نداشتن خطای املایی چک می شود . اگر خطایی وجود داشته مـی باشد هیچ فایلی ایجاد نخواهد شد .

program show result

از این گزینه برای دیدن نتیجهء کامپایل می توان استفاده کرد .

program simulator

با فشردن کلید ***f2*** یا این گزینه شبیه ساز داخلی استفاده می شود . در صورت تمایـل میتوانیـد از شـبیه ساز های دیگری نیز نظیر ***avr studio*** استفاده کنید .

send to chip

توسط این گزینه یا کلید ***f4*** پنجرهء محیط برنامه ریزی ظاهر می شود که در این پنجـره شـما میتوانیـد علاوه بر برنامه ریزی خود میکرو ، به فیوز بیتها و حافظهء فلـش میکـرو دسترسـی پیـدا کـرده و آنهـا را برنامه ریزی کنید .

۳-۱-۴. منوی tools

terminal emulator

توسط این گزینه یا کلیدهای ***ctrl+t*** با بالا آوردن ***terminal emulator*** مـی توانیـد از ایـن محـیط بـرای نمایش دادهء ارسالی و دریافتی در ارتباط ***rs – 232*** بین میکرو و کامپیوتر استفاده کنید .

lcd designer

توسط این گزینه می توانید کاراکترهای مورد نظر خود را طراحی کرده و بر روی ***lcd*** نمایش دهید .

graphic converter

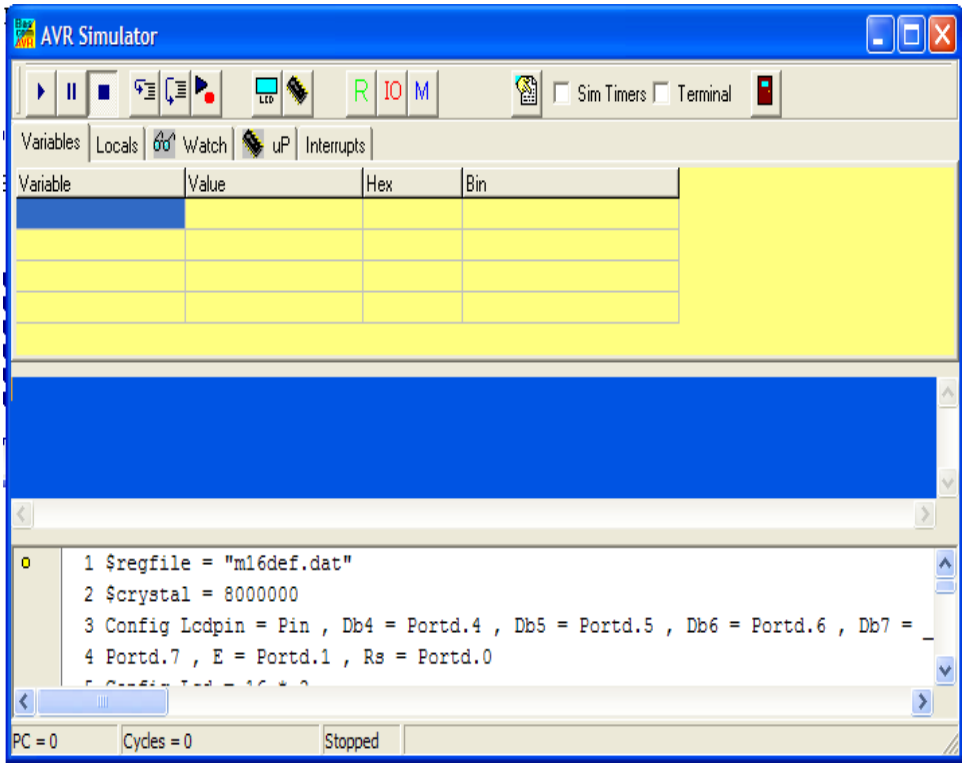
توسط این پنجره می توان عکس ها یا کاراکترهای مورد نظر را بر روی ***lcd*** گرافیکی نمایش داد .

۳-۱-۵. منوی option

در این منو یک سری تنظیمات انجام میگیرد که می توانید بـه راحتـی آنهـا را پیـدا کنیـد . در پنجـرهء ***option programmer*** می توانید نوع ***programmer*** خود را انتخاب کنید .

۳-۲. معرفی محیط شبیه سازی (simulator)

با ورود به محیط شبیه سازی پنجره زیر ظاهر می شود .



در ادامه به معرفی منوهای این محیط می پردازیم :

run

با فشردن این کلید شبیه سازی آغاز می شود .

pause

این دکمه باعث توقف موقت شبیه سازی می شود .

stop

این دکمه باعث توقف کامل شبیه سازی می شود .

step into code

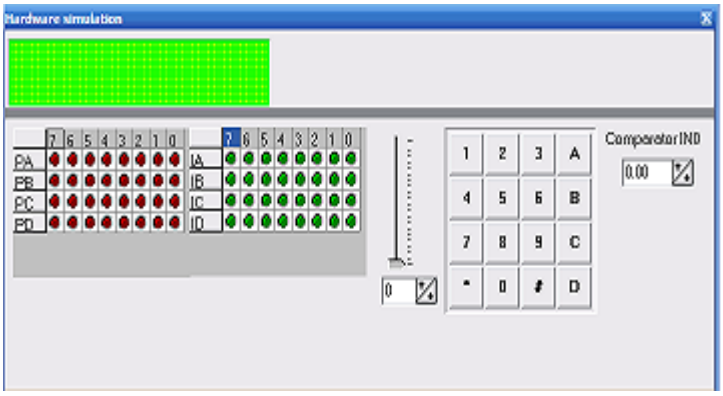
با فشردن این کلید یا کلید ***f4*** میتوان برنامه را خط به خط اجرا کرد و می توان هنگام فراخوانی توابع بـه داخل آنها رفته و مراحل اجرای آنها را بررسی کرد .

run to

این دکمه شبیه سازی را تا خط انتخاب شده انجام می دهد و سپس به حالت ***pause*** میرود .

شبیه ساز سخت افزاری ***(the hardware simulator)***

با کلیک روی آن پنجرهء زیر باز می شود .



قسمت بالایی پنجره یک ***lcd*** مجازی می باشد که برای نشان دادن داده های ارسالی به ***lcd*** استفاده مـی شود . نوار ***led*** های پایین خروجی پورت ها را نشان میدهد . با کلیک کردن بر روی هر یک از ***led*** ها که به عنوان ورودی هستند وضعیت آن معکوس می شود و روشن بودن ***led*** به منزلهء یک کردن پایهء پورت است . یک صفحه کلید نیز تعبیه شده است که با دسـتور ***getkbd()*** در برنامـه قابـل خوانـدن اسـت . در ضمن مقدار آنالوگ نیز هم برای مقایسه کنندهء آنالوگ و هم برای کانالهـای مختلـف ***adc*** قابـل اعمـال است .

registeres

این دکمه پنجرهء رجیسترها را با مقدار فعلی نمایش می دهد که میتوان مقدار رجیسترهای ***r0*** تا ***r31*** را به صورت هگزادسیمال مشاهده کرد .

i/o registeres

برای نمایش رجیسترهای ***i/o*** استفاده می شود .

vaeiables

شما قادر به انتخاب متغیر با دو بار کلیک کردن در ستون ***variables*** می باشید . با فشار دکمهء ***enter*** در هنگام اجرای برنامه قادر به مشاهدهء مقدار جدید متغیر در برنامه خواهید بود همچنین می توانید مقدار هر متغیر را توسط ***value*** تغییر دهید .

local

پنجره ***local*** متغیرهای محلی موجود در ***sub*** یا ***function*** را نشان میدهد . شما نمی توانیـد متغیرهـا را اضافه کنید .

watch

این گزینه برای وارد کردن وضعیتی که قرار است در خلال شبیه سازی ارزیابی شود مورد استفاده قرار می گیرد و هنگامی که وضعیت مورد نظر تصحیح شد شبیه سازی در حالت ***pause*** قرار خواهد گرفت .

up

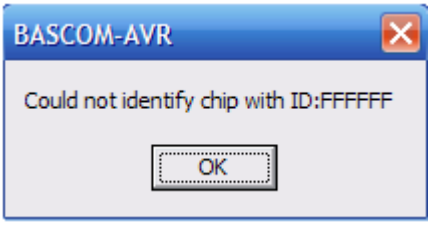
این گزینه وضعیت رجیستر وضعیت را نشان می دهد . ***flag*** ها را می توان توسط کلیک بر روی ***check box*** ها تغییر وضعیت داد .

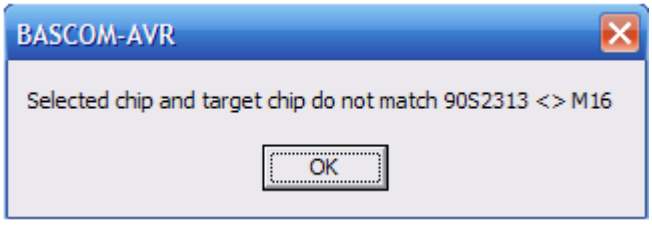
interrupts

این گزینه منابع وقفه را نشان می دهد . فعال بودن یا نبودن وقفه های داخلی و خارجی به راحتی در این پنجره قابل مشاهده است .

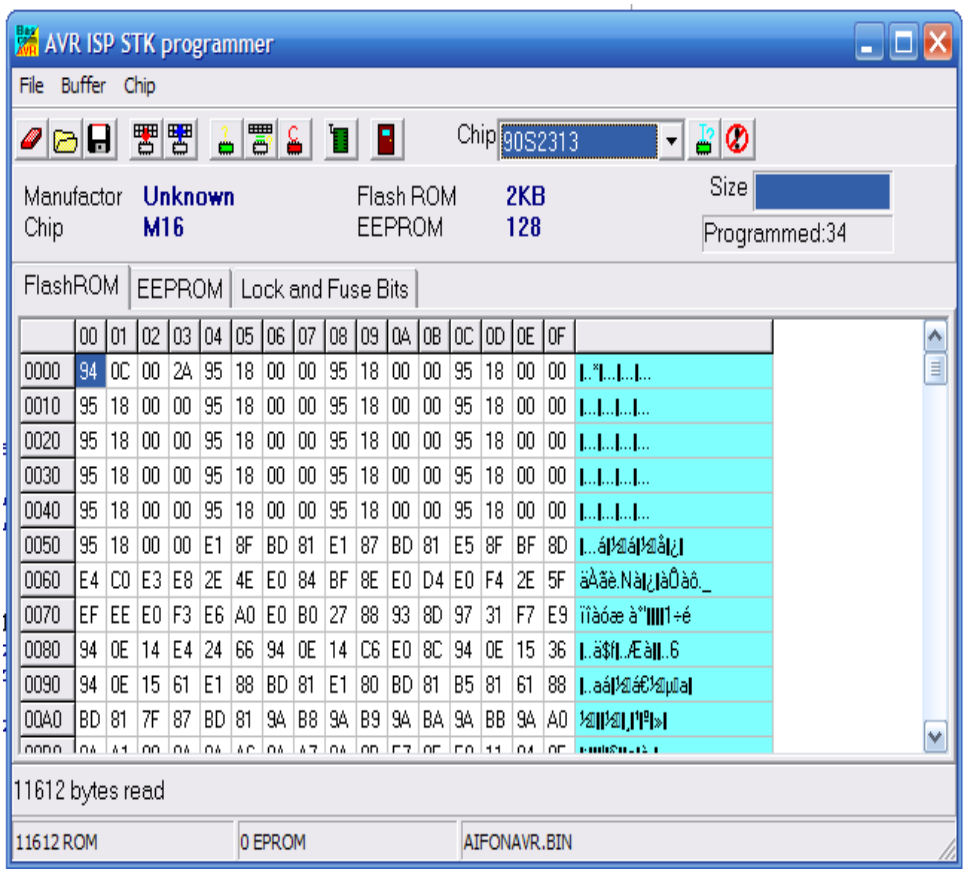
۳-۳. معرفی محیط برنامه ریزی میکرو

پنجرهء ارسال برنامه به میکرو هنگامی که ***programmer*** ***run*** انتخاب می شود ظاهر می گردد . با فشار کلید ***f4*** نیز می توان همین کار را کرد . در صورتی که کامپایلر نتواند میکرو متصل به ***programmer*** را شناسایی کند یا اینکه خود ***programmer*** مشکل داشته باشد در آن صورت یکی از دو پنجرهء زیر ظاهر میشود :





در شکل پنجرهء محیط برنامه ریزی نشان داده شده است .



منوهای اصلی این محیط عبارتند از :

منوی ***file***

test : با این گزینه شما می توانید پایه های پورت ***lpt*** را یک کنید . این گزینه فقط برای زمانی است که شما از ***sample electronic programmer*** استفاده می کنید .

منوی ***buffer***

buffer clear

این گزینه بافر را پاک می کند .

load from file

با این گزینه می توان بافر را با فایلی پر کرد و آنرا در حافظهء میکرو برنامه ریزی کرد .

save to file

توسط این گزینه می توان بافر را در فایلی ذخیره کرد .

منوی ***chip***

chip identify

با این گزینه می توان میکرو متصل به پروگرمر را شناسایی کرد .

write buffer to chip

توسط این گزینه می توان محتوای بافر را در حافظهء ***rom*** یا ***eeprom*** میکرو برنامه ریزی کرد .

raed clipcode into buffer

با این گزینه می توان حافظهء کدی میکرو را خواند .

blank check

خالی بودن حافظهء میکرو را مشخص می کند .

erase

این گزینه محتوای حافظه برنامه وداده ***eeprom*** را پاک می کند .

verify

این گزینه محتوای بافر و آنچه در میکرو برنامه ریزی شده است را مقایسه می کند و در صورت تساوی پیغام ***verify ok*** نمایش داده می شود .

auto program

میکرو را ***program*** میکند .

reset

این گزینه میکرو متصل به پروگرمر را ری ست می کند .

در همین محیط برنامه ریزی می توان فیوز بیتها را نیز برنامه ریزی کرد .

باید گفت که از توضیح تعدادی از منوها و پنجره ها نیز صرفنظر شده است و تا آنجا که ممکن بوده سعی شده است منوها وپنجره های اصلی توضیح داده شود تا محیط کلی نرم افزار تشریح شود و جزییات بیشتر را خود کاربر در زمان کار با نرم افزار متوجه خواهد شد . در هر صورت کار با این نرم افزار بسیار راحت است .

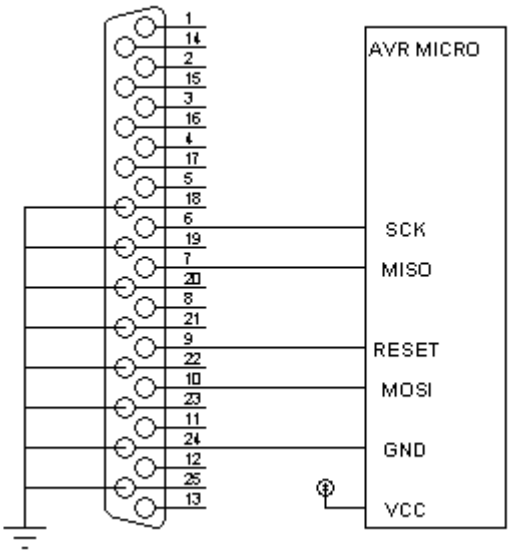
۳-۴. ساخت پروگرمر stk 200/300

باید به طریقی میکرو برنامه ریزی شود و به نظر می رسد که یک وسیلهء سخت افزاری باید در کار باشد تا ارتباط میکرو با کامپیوتر برقرار شود و از طریق یک پروتکل استاندارد میکرو برنامه ریزی شود ، به وسیله ای که این کار را انجام می دهد پروگرمر ***(programmer)*** می گویند و یکی از پروگرمرهای خوب در این زمینه پروگرمر ***stk 200/300*** است که از ارتباط استاندارد ***spi*** برای برنامه ریزی میکرو استفاده می کند .

نکته مهم در این مورد این است که اگر از این پروگرمر استفاده می کنید باید در منوی ***option*** و در گزینهء انتخاب ***programmer*** نوع پروگرمر را ***stk200/300*** انتخاب کنید .

این ***programmer*** را میتوان به چند روش ساخت .

یک روش استفاده از مدار زیر است که به پورت ***lpt*** کامپیوتر وصل می شود . مدار بهتر را می توانید با استفاده از بافر ***74HC244*** بسازید که مدار آن داخل بیشتر کتابهای آموزشی ***AVR*** وجود دارد .



مدار ***STK 200/300 PROGRAMER***

۴. فصل چهارم، سخت افزار سیستم و ارتباط SPI

۴-۱. ارتباط سریال SPI

در این پروژه با توجه به اینکه از دو میکرو استفاده شده است و این دو میکرو باید به نحوی با هم ارتباط بر قرار کنند بنابراین نیاز به یک پروتکل ارتباطی استاندارد است . برای اینکار از پروتکل ارتباط ***SPI*** استفاده شده است .

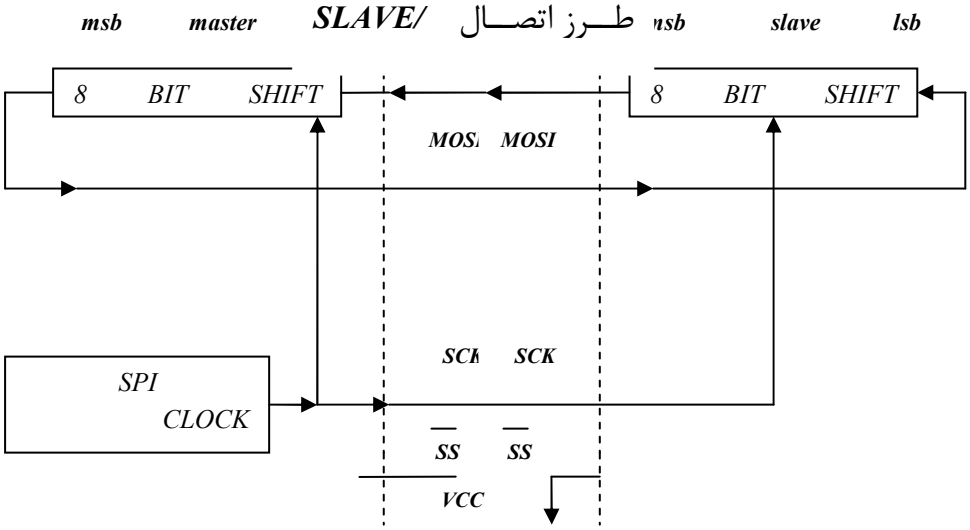
ارتباط سریال ***SPI (SERIAL PERIPHERAL INTERFACE)*** یک پروتکل ارتباطی سریال سنکرون با سرعت بالاست که میتواند برای ارتباط میکروهای ***AVR*** با یکدیگر و یا با وسیله های دیگر که قابلیت ارتباط با این نوع پروتکل را دارا هستند به کار برده می شود . از طرفی نیز این همان ارتباط استانداردی است که برای برنامه ریزی خود میکرو یا به عبارتی همان پروگرام کردن از آن استفاده میکنیم . رجیسترهای مربوط به این ارتباط در انواع میکروهای ***AVR*** یکسان است . در زیر به معرفی رجیسترهای مربوط به ***SPI*** دریک نمونه میکرو مثل ***MEGA16*** پرداخته ایم .

خصوصیات

- ***FULL-DUPLEX*** ارسال دادهء همزمان ***(SYNCHRONOUS)*** سه سیمه ***(3-WIRE)***
- ارتباط به صورتهای ***MASTER,*** و ***SLAVE***
- ارسال ابتدا ***MSB*** یا ***LSB***
- بیت های قابل برنامه ریزی برای سرعت
- پرچم وقفهء اتمام ارسال
- بیدار شدن از حالت بیکاری

با توجه به پایه های مورد نظر در ارتباط ***SPI*** ، پایهء ***SCK*** خروجی کلاک برای ***MASTER*** و ورودی کلاک برای ***SLAVE*** است . در حالت ***MASTER*** که میتوان گفت میکرو به عنوان ارسال کنندهء داده عمل می کند در این حالت دادهء مورد نظر برای ارسال داخل رجیستر دادهء ***SPI*** قرار گرفته و در این زمان ***CPU*** شروع به تولید کلاک ***SPI*** کرده و داده ها از پایهء ***MOSI (MATER OUTPUT SLAVE INPUT)*** خارج شده و به پایهء ***MOSI*** در ***SLAVE*** وارد می شود .

بعد از انتقال کامل داده توسط ***MASTER***، کلاک ***SPI*** قطع و پرچم وقفه پایان ارسال داده ***(SPIF)*** یک می شود و برنامهء وقفه اجرا می شود . دو شیفت رجیستر ۸ بیتی در ***MASTER*** و ***SLAVE*** را می تـوان به عنوان یک شیفت رجیستر چرخشی ۱۶ بیتی در نظر گرفت . این موضوع در شکل زیر دیده می شـود .زمانی که داده ای از ***MASTER*** به ***SLAVE*** ارسال می شود ، میتوان در همـان حـال در جهـت مخـالف داده ای از ***SLAVE*** به ***MASTER*** انتقـال یابـد . بـدین صـورت کـه در طـول ۸ کـلاک ***SPI*** داده هـای ***SLAVE*** و ***MASTER*** با هم عوض می شود.



۴-۱-۱. ارتباط SPI و رجیسترهای مربوطه

۴-۱-۱-۱. رجیستر کنترلی SPI

این رجیستر در اصل وظایف کنترلی ارتباط ***SPI*** را مشخص می کند و به نام ***SPCR*** شناخته می شـود . این رجیستر ۸ بیتی بوده که در زیر کار هر کدام از بیتهای آن مشخص شده است.

۷	۶	۵	۴	۳	۲	۱	۰
SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

SPIE- این بیت فعال کنندهء وقفهء ***SPI*** است . اگر در برنامهء اصلی ابتدا وقفهء سراسری فعال شده باشد و بعد از با فعال کردن وقفهء ***SPI*** این بیت ۱ خواهد شد .

SPE – برای فعال کردن ***SPI*** در حالت کلی باید این بیت ۱ باشد به عبارتی برای انجام هر گونه عملیات ***SPI*** باید این بیت ۱ باشد .

DORD – ***(DATA ORDER)*** این بیت مشخص کنندهء این است که ابتدا کم ارزش ترین بیت داده ارسال شود یا با ارزش ترین بیت آن ، به این صورت که اگر این بیت ۱ باشد ***LSB*** داده ابتدا فرستاده میشود و اگر صفر باشد ***MSB*** داده ابتدا فرستاده می شود .

MSTR – این بیت مشخص کنندهء ***SLAVE*** یا ***MASTER*** بودن ارتباط ***SPI*** برای میکرو است . اگر این بیت ۱ باشد ارتباط ***SPI*** در حالت ***MASTER*** و اگر صفر باشد ارتباط ***SPI*** در حالت ***SLAVE*** خواهد بود.

CPOL - ***(CLOCK POLARITY)*** در حالت بیکاری ***SPI*** اگر این بیت ۱ باشد پایهء ***SCK*** بالا خواهد بود و در غیر این صورت پایهء ***SCK*** پایین خواهد بود .

CPHA – ***(CLOCK PHASE)*** کار این بیت در ادامه توضیح داده شده است .

۴-۱-۱-۲. مدهای اطلاعاتی برای ارتباط *SPI (DATA MODE)*

چهار نوع ترکیب ***PHASE*** و ***POLARITY*** با توجه به دادهء سریال طبق جدول زیر موجود می باشد که توسط بیتهای کنترلی ***CPHA*** و ***CPOL*** مشخص می شوند . مدهای مختلف برای انتقال دادهء ***SPI*** در جدول زیر نشان داده شده است . بیت ***CPOL*** نقشی در نحوهء ارسال داده ندارد و فقط وضعیت پایهء ***SCK*** را در حالت بیکاری ***SPI*** مشخص می کند . در اینجا باید به یک نکتهء مهم توجه کنیم و آن این است که میکروها در حالت ***SLAVE*** و ***MASTER*** باید در یک مد پیکره بندی شوند .

SPI MODE	CPOL	CPHA	*SHIFT SCK EDGE*	*CAPTURESCK EDGE*
0	*0*	*0*	*FALLING*	**RISING**
1	*0*	*1*	*RISING*	**FALLING**
2	*1*	*0*	*RISING*	**FALLING**
3	**1**	**1**	**FALLING**	**RISING**

جدول انتخاب مُد های ارتباطی اطلاعاتی ***SPI***

بیتهای ***SPR1*** و ***SPR0*** – این دو بیت فرکانس کلاک ***SPI*** را برای ***MASTER*** تعیین می کنند . ایـن دو بیت تاثیری بر روی عملکرد ***SLAVE*** ندارند و در جدول زیر ارتباط بین ***SCK*** و فرکانس کلاک اسـیلاتور را مشاهده می کنید .

SPI2X	SPR1	SPR0	*SCK FREQUENCY*
0	*0*	*0*	**Fosc/4**
0	*0*	*1*	**Fosc/16**
0	*1*	*0*	**Fosc/64**
0	*1*	*1*	**Fosc/128**
1	*0*	*0*	**Fosc/2**
1	*0*	*1*	**Fosc/8**
1	*1*	*0*	**Fosc/32**
1	**1**	**1**	**Fosc/64**

جدول انتخاب فرکانس کلاک ***spi*** با توجه به فرکانس اسیلاتور

۴-۱-۱-۳. رجیستر وضعیت *SPI*

این رجیستر با نام ***SPSR (SPI STATUS REGISTER)***شناخته می شود کـه ۸ بیتـی بـوده و عملکـرد بیتهای آن به صورت زیر است :

۷	۶	۵	۴	۳	۲	۱	۰
SPIF	*WCOL*	-	-	-	-	-	*SPI2X*

بیت ***SPIF*** – زمانیکه ارسال داده تمام شد این بیت یک می شود در صورتی که بیت ***SPIE*** در رجیستر ***SPCR*** (رجیستر کنترلی ***SPI***) و وقفه سراسری فعال شده باشد .

بیت ***WCOL (WRITE COLLISION FLAG)*** – اگر در زمان انتقال داده در رجیستر ***SPDR*** نوشته شود بیت ***WCOL*** یک می شود . در زمان یک بودن این بیت اولین خواندن از رجیستر وضعیت باعث صفر شدن بیت ***WCOL*** و همچنین بیت ***SPIE*** می شود و سپس دسترسی به رجیستر داده ***(SPDR)*** انجام می گیرد .

بیتهای ***5 . . . 1*** – این بیتها جزء بیتهای رزرو شده هستند .

بیت ***SPI2X (DOUBLE SPI SPEED BIT)*** – همانطور که از نام این بیت مشخص است با یک شدن آن سرعت کلاک (فرکانس کلاک ***SPI***) در مُد ***MASTER*** دو برابر می شود و این بدین معنی است که کلاک ***SPI*** می تواند تا نصف کلاک سیستم افزایش یابد . زمانی که میکرو در مُد ***SLAVE*** قرار دارد همچنان بیشترین فرکانس کلاک ***SPI*** برابر ***FOSC/4*** است . باید گفت که این بیت در بعضی از میکروهای ***AVR*** وجود دارد .

۴-۱-۱-۴. رجیستر دادهء SPI

این رجیستر به نام ***SPDR (SPI DATA REGISTER)*** شناخته می شود که این رجیستر نیز مانند دیگر رجیسترها ۸ بیتی بوده و یک رجیستر خواندنی/نوشتنی است که برای ارسال یا دریافت دادهء ***SPI*** استفاده می شود . نوشتن در این رجیستر داده را به باس ***SPI*** ارسال می کند و خواندن از این رجیستر دادهء موجود در بافر دریافتی شیفت رجیستر خوانده می شود .

۴-۱-۲. پیکره بندی SPI در محیط BASCOM

در محیط این نرم افزار می توان پایه های ***SPI*** را به دو صورت نرم افزاری و سخت افزاری پیکره بندی کرد . زمانی که ارتباط به صورت سخت افزاری پیکره بندی می شود همان پایه های پیش فرض به کار میروند و نمی توان آنها را تغییر داد . اما توسط پیکره بندی نرم افزاری می توان هر یک از پایه های

میکرو را به عنوان پایه های ارتباط ***SPI*** تعریف کرد . در این پروژه ما ارتباط ***SPI*** را در مُد سخت افـزاری پیکره بندی کرده ایم که توسط دستور زیر ***SPI*** پیکره بندی می شود .

CONFIG SPI=HARD , INTERRUPT=ON|OFF , DATA ORDER =LSB|MSB , MASTER=YES|NO , POLARITY=HIGH|LOW , PHASE=0|1 , CLOCK RATE = 4|16|64|128 , NOSS=0|1

به طور کلی دستور ***CONFIG*** در ***BASCOM*** دستور پیکره بندی سخت افزاری است .

INTERRUPT=ON|OFF : در صورت استفاده از وقفه در ارتباط ***SPI*** از گزینه ***ON*** استفاده می شود .

DATA ORDER=LSB|MSB : در صورت انتخاب ***LSB*** ، ابتدا ***LSB*** و سـپس ***MSB*** داده ارسـال خواهـد شد و اگر ***MSB*** انتخاب شود ابتدا ***MSB*** داده ارسال می شود و سپس ***LSB***

MASTER=YES|NO : اگر میکرویی که در حال برنامه نویسی آن هستیم ***MASTER*** باشد کزینـه ***YES*** و اگر ***SLAVE*** باشد گزینه ***NO*** انتخاب می شود .

PHASE=0|1 : انتخاب صفر توصیه می شود .

POLARITY=HIGH|LOW : اگر بخواهیم زمانی که ***SPI*** در حالت بیکاری است پایه کـلاک بـالا باشـد گزینه ***HIGH*** انتخاب می شود .

CLOCK RATE : مشخص کننده فرکـانس کـلاک ***SPI*** کـه میتوانـد ***1/4 , 1/16 , 1/64, 1/12*** فرکـانس سیستم می باشد .

NOSS=0|1 : زمتنی که در حالت ***MASTER*** نمی خواهید سیگنال ***SS*** ایجاد شود ۱ را انتخـاب کنیـد . در این حالت باید کاربر به صورت نرم افزاری پایه ***SLAVE*** مورد نظر را پایین کند .

می توان پیکره بندی سخت افزاری را به این صورت نیز نوشت :

CONFIG SPI=HARD

که در این حالت به صـورت پـیش فـرض اول ***MSB*** فرسـتاده مـی شـود و ***POLARITY =HIGH*** ، ***MASTER =YES*** ، ***Phase =0*** ، ***clockrate=4*** در نظر گرفته می شوند .

۴-۱-۲-۱. دستور *spiinit*

توسط این دستور پایه های به کار برده شده در ارتباط ***initil, spi*** می شوند . این دسـتور بعـد از پیکـره بندی ***spi*** بایستی برای قرار گیری پایه استفاده شده در جهت مناسب نوشته شود .

۴-۱-۲-۲. دستور spiin

SPIIN VAR , BYTES

توسط این دستور به تعداد ***byte*** داده از باس ***spI*** دریافت شده و در داخل متغیر تعریف شده قرار می گیرد .

۴-۱-۲-۳. دستور SPIOUT

SPIOUT VAR , BYTES

توسط این دستور به تعداد ***BYTES*** داده مورد نظر به باس ***SPI*** ارسال می شود .

۴-۲. پیکره بندی تجهیزات سخت افزاری

۴-۲-۱. پیکره بندی lcd

برای آنکه ***lcd*** را به میکرو وصل کنیم باید ابتدا آنرا از لحاظ سخت افزاری پیکره بندی کنیم . پایه های ***lcd*** برای اتصال به میکرو طبق دستور زیر پیکره بندی می شوند .

config lcdpin=pin , db4=portd.4 , db5=portd.5 , db6=portd.6 , db7=_
portd.7 , e=portd.1 , rs=portd.0

که ***db4*** تا ***db7*** پورتهای ***lcd*** است که ***lcd***از نوع ***lcd***های کاراکتری است و از آن به صورت ۴ بیتی استفاده شده است و مشخص است که در مقابل نیز پورتهای میکرو قرار دارند که می توان هر پورت دلخواهی از میکرو را به ***lcd*** متصل کرد .

در اینجا باید متذکر شد که برای این برنامه ما از دو عدد ***Lcd*** استفاده کرده ایم که هر دو تا از نوع ***lcd*** های کاراکتری و از نوع ۲*۱۶ هستند که به این معنی است که دارای دو سطر و شانزده ستون است و طبق دستور زیر می توان این گزینه را مشخص کرد .

config lcd=16*2

برای کار با ***lcd*** دستوراتی در نظر گرفته شده است که در ادامه مهمترین این دستورات را معرفی می کنیم .

lcd

این دستور یک یا چند عبارت ثابت یا متغیر را بر روی ***lcd*** نمایش می دهد . فرم کلی این دستور به صورت زیر است .

lcd x نمایش متغیر بر روی ***LCD***

lcd "salam" نمایش یک یا چند کاراکتر

cls

این دستور کل صفحه نمایش را پاک می کند .

cursor

توسط این دستور می توان مکان نمای ***lcd*** را تنظیم کرد . مکان نما می تواند بـه صـورت روشـن (***on***) ، خاموش ***(off)*** ، چشمک زن ***(blink)*** ، یا چشمک نزن ***(no blink)*** باشد .

cursor on/off/blink/no blink

home

این دستور به تنهایی مکان نما را در سطر اول ستون اول قرار می دهد .

locate

توسط این دستور می توان به هر سطر و ستونی از ***lcd*** رفت و کاراکتر یا متغیر مورد نظر را نمـایش داد .

فرم کلی این دستور به صورت زیر است :

locate x,y

که ***x*** مشخص کننده سطر و ***y*** مشخص کننده ستون است .

shiftlcd

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

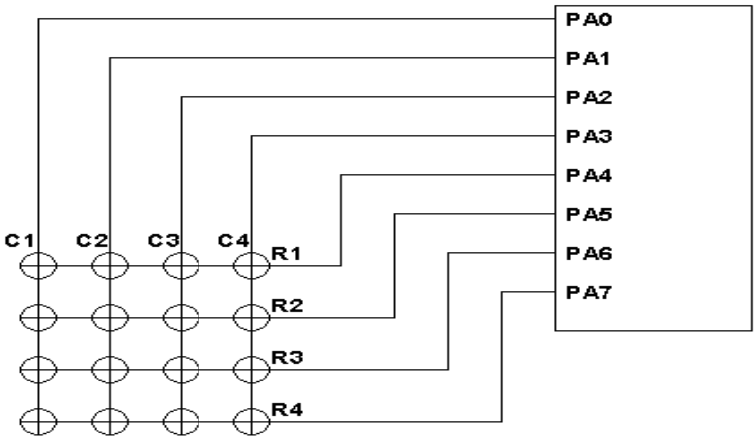
shiftlcd left/right

lowerline

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

پیکره بندی صفحه کلید ***4*4***

در این پروژه ما از یک صفحه کلید ۴*۴ استفاده کرده ایم که باید به عنوان یـک وسـیله سـخت افـزاری پیکره بندی شود . ابتدا صفحه کلید را طبق شکل زیر به میکرو وصل می کنیم :



نحوه اتصال صفحه کلید ماتریسی به میکرو

حال برای پیکره بندی سخت افزاری صفحه کلید از دستور زیر استفاده می کنیم :

CONFIG KBD = PORTX , DEBOUNCE = VALUE , DELAY = VALUE

PORTX مشخص کننده پورتی از میکرو است که صفحه کلید به آن وصل می شود . ***DEBOUNCE*** به صورت پیش فرض ۲۰ است و می تواند ماکزیمم مقدار ۲۵۵ را داشته باشد . ***DELAY*** نیز پارامتر اختیاری است و مشخص کننده تاخیری بر حسب میلی ثانیه است که در زمان خواندن کلید توسط دستور ***GETKBD ()*** ایجاد می شود .

مثال:

CONFIG KBD = PORTA , DEBOUNE=20 , DELAY=100

GETKBD()دستور خواندن از صفحه کلید:

توسط این دستور میکرو صفحه کلید را خوانده و عدد متناظر با کلید فشرده شده را در متغیر فرضی ***X*** قرار می دهد . این دستور زمانی که کلیدی فشرده نشده باشد عدد ۱۶ را بر میگرداند . متغیر مبدا باید از نوع ***BYTE*** تعریف شود . حال با توجه به شکل قبلی که صفحه کلید را به میکرو وصل کرده بودیم آنگاه عددهای متناظر با دستور ***GETKBD()*** به صورت زیر خواهند بود :

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

۵. فصل پنجم، نرم افزار سیستم

۵-۱. برنامه اصلی سیستم

برنامه اصلی سیستم داخل میکرو ***mega16*** قرار دارد و این میکرو نقش اصلی را بر عهده دارد. در اینجا می خواهیم در مورد این برنامه توضیح مختصری بدهیم .

فرم کلی هر برنامه در محیط ***bascom*** بدین صورت است که در ابتدا میکرو معرفی میشود :

$regfile = "mega16def.dat"

طبق این دستور نوع میکرو معرفی میشود که میتوان هر میکرو مشابه دیگری را نیز جایگزین نمود .

در ادامه فرکانس کلاک سیستم مشخص میشود که همان فرکانس کریستال است :

$crystal = 8000000

فرکانس نوشته شده بر حسب هرتز ***(hertz)*** است که میتوان هر فرکانس استاندارد دیگری را جایگزین نمود .

حال در ادامه تجهیزات سخت افزاری را پیکره بندی کرده ایم مانند ***lcd*** و صفحه کلید . سپس ارتباط ***spi*** پیکره بندی شده است .

گفتیم که یک کلید ***exit*** روی صفحه کلید در نظر گرفته شده است که با آن میتوان میکرو ***mega16*** را ***reset*** نرم افزاری کرد و باعث میشود که در هر قسمتی از برنامه هستیم از آنجا خارج شده و میکرو به حالت اولیه و کار عادی سیستم برگردد ، برای این کار از ***watchdog*** داخلی میکرو استفاده شده است و تایمر ***watchdog*** از اسیلاتور جداگانه داخلی کلاک دریافت میکند و با تنظیم آن پس از زمان مشخص شده سیستم ری ست میشود . ***watchdog*** ابتدا باید پیکره بندی شود :

config watchdog = 16

طبق این دستور مقدار زمان سپری شده برای ری ست شدن برابر ***16*** میلی ثانیه است که مقادیر معتبر دیگری را نیز می توان جایگزین نمود واین مقادیر عبارتند از : ۲۰۴۸ ، ۱۰۲۴، ۵۱۲ ، ۲۵۶ ، ۱۲۸ ، ۶۴ ، ۳۲ میلی ثانیه .

باید گفت با این دستور فقط ***watchdog*** پیکره بندی شده و شروع به کار نمیکند و برای شروع به کار ***watchdog*** باید دستور ***start watchdog*** نوشته شود و برای توقف آن دستور ***stop watchdog*** و با دستور ***reset watchdog*** تایمرآن ری ست میشود .

پورتهای میکرو یا به صورت ورودی هستند یا خروجی . بنابراین باید بتوان جهت پورت را مشخص کرد و با پیکره بندی هر یک از پورتها یا پینهای میکرو میتوان این کار را انجام داد :

config porta=input یا ***0***

config pinb.2=output یا ***1***

اگر توجه کنید میتوان هر یک از پینهای میکرو مانند ***pinb.2*** را نیز جداگانه به عنوان ورودی یا خروجی تعریف کرد ، پیداست وقتی میخواهیم مثلا ***led*** به میکرو وصل کنیم باید آن پین به عنوان خروجی تعریف شود .

میدانیم که مبنای اصلی هر برنامه متغیرهای آن هستند که مهمترین نقش برنامه بر عهده دارند و روند برنامه را آنطور که می خواهیم بر اساس آنها تنظیم میشود . برای معرفی یک متغیر در محیط ***bascom*** از دستور ***dim*** استفاده می شود و متغیرها میتوانند انواع مختلفی داشته باشند که این بستگی به مقداری دارد که متغیر به خود میگیرد مانند ***bit*** ، ***byte*** ، ***word*** ، ***integer*** ، ***long*** ، ***single*** ، ***string*** . حتی برای ***eeprom*** نیز میتوان متغیر تعریف کرد و آنرا در آدرس خاصی از حافظه ***eeprom*** ذخیره کرد .

```
byte  dim  a as
dim u(22) as eram word at &h1bd
```

متغیر ***a*** از نوع بایت معرفی شده است و متغیر ***u(22)*** یک رشته عددی است به طول ***44*** بایت (۲۲ تا ***word***) که برای ***eeprom*** معرفی شده است و در آدرس ***h1bd*** (هگزادسیمال) به بعد در خود حافظه ***eeprom*** ذخیره شده است .

زیر برنامه ها را توسط برچسبهای جداگانه با اسامی مختلف از هم تفکیک میکنند و توسط دستور ***gosub*** میتوان به آنها پرش کرد و برای برگشت از زیر برنامه نیز دستور ***return*** وجود دارد مانند :

```
gosub eprom_read
```

در ابتدای برنامه برچسب ***again (lable)*** وجود دارد که برچسب اصلی برنامه است و در اینجا ابتدا میکرو منتظر است تا عددی از صفحه کلید وارد شود و این عدد داخل متغیر ***a*** قرار میگیرد و چون میخواهیم عملکرد کلید های صفحه کلید را تغییر دهیم بنابراین از متغیر دیگری به نام ***b*** استفاده کرده ایم که مقدار متناسب با کلید فشرده شده بر اساس آن تنظیم میشود . بنابراین با زدن هر کدام از کلید واحدها طبق برنامه نام و شماره واحد بر روی ***lcd*** نمایش داده میشود . میدانید که مقدار هر متغیری قبل از

اینکه مقدار دهی شود صفر است و این یعنی که کلیه متغیرهای برنامه در ابتدا مقدار صفر را در خود دارند به جز متغیر هایی که مقدار دهی اولیه شده اند .

در اینجا برای مشخص کردن تعداد زنگهای زده شده هر یک از واحدها از متغیری به نام ***c*** استفاده شده است و تعداد زنگهای زده شده برای وارد شدن به قسمت پیغام گزاری را سه زنگ در نظر گرفته ایم که میتوان به طور دلخواه آنرا تغییر داد و زمان یک و صفر شدن پورتها را نیز میتوان به دلخواه تغییر داد .

در برچسبهای ***next1*** تا ***next10*** اعداد وارد شده از صفحه کلید در حافظه ***eeprom*** داخلی میکرو نوشته میشود و همزمان نیز تک تک بر روی ***lcd*** نمایش داده میشود . متغیر ***p*** یکی از متغیرهای مهم برنامه است که آدرس فعلی حافظه ***eeprom*** داخل آن قرار دارد و باید به درستی مدیریت شود .

بعد از اینکه سه بار زنگ واحد مربوطه زده شد به طور اتوماتیک وارد قسمت پیغام گزاری میشویم که ابتدا پیغام ***enter payam*** بر روی ***lcd*** نمایش داده میشود و از شخص مراجعه کننده میخواهد که پیغام خود را وارد کند . برنامه طوری تنظیم شده است که وارد کردن بیش از ۱۴ رقم مقدور نمی باشد و این تنظیمات توسط متغیرهای ***p1 , p2 , d*** صورت میگیرد .

متغیر ***z*** نیز ستون ***lcd*** را مشخص می کند و باید این متغیر در روند برنامه به صورت صحیحی تنظیم شود و این کار انجام شده و در ابتدا مقدار این متغیر را ۱۶ در نظر گرفته ایم .

برای ایجاد تاخیر در هر کجای برنامه در محیط ***bascom*** میتوان از دستور ساده ***wait*** استفاده کرد که تاخیرهایی را بر حسب میکروثانیه ، میلی ثانیه و ثانیه برای ما ایجاد میکند و فرم کلی این دستور به این صورت است :

```
waitus 10
waitms 10
wait 10
```

در اینجا به عنوان مثال تاخیرهای ***10us*** ، ***10ms*** و ***10s*** ایجاد شده است .

برای ایجاد دستورات شرطی برنامه بیشتر از دستور ***if*** استفاده شده که برای هر ***if*** باید یک ***end if*** به عنوان پایان این دستور شرطی وجود داشته باشد .

با زدن کلید ***enter*** به برچسب ***spi_inter*** پرش میشود و در این برچسب از برنامه ابتدا توسط ارتباط ***spi*** که از قبل پیکره بندی شده است بایتهای ارسالی از میکرو ***mega8*** که اطلاعات تاریخ و زمان را دربر دارند دریافت شده و داخل متغیرهای مربوطه قرار میگیرندو سپس داخل ***eerpom*** نوشته میشوند . سپس

از یک حلقه ***case*** استفاده شده است که طبق آن بیست بایت برای هر پیام در نظر گرفته میشود و در اینجا آدرس حافظه ***eeprom*** که داخل متغیر ***p*** قرار دارد به داخل آرایه ۲۲ بایتی ***u(22)*** کپی میشود و گفتیم که این آرایه از نوع خود حافظه ***eeprom*** است و بنابراین با قطع تغذیه اطلاعات آن حفظ شده و نگهداری میشود و این عمل باعث میشود که علاوه بر اینکه خود پیغامها در ***eeprom*** ذخیره میشوند بلکه آدرسها نیز در همین حافظه ذخیره شده و به صورت صحیح حتی پس از قطع تغذیه سیستم قابل دسترسی باشند . با این کار وقتی تغذیه سیستم به طور ناگهانی قطع میشود آخرین آدرس ***eeprom*** مشخص بوده و پیغامهای جدید بعد از این آدرس ذخیره میشوند و متغیر ***r*** نیز به روند اینکار کمک میکند . در ابتدای برنامه نیز یک دستور ***gosub eprom_read*** وجود دارد که باعث میشود به زیربرنامه ***eprom_read*** پرش شود و در آنجا با توجه به مقدار متغیر ***r*** آدرس تنظیم میشود .

برای کار با حافظه ***eeprom*** داخلی میکرو دو دستور اصلی وجود دارد که یکی برای نوشتن در حافظه است و دیگری برای خواندن . این دو دستور را با مثال توضیح میدهیم :

```
b , p  writeeeprom
readeeprom b , p
```

نوشتن متغیر ***b*** در آدرس ***p*** از حافظه ***eeprom***

خواندن خانه آدرس ***p*** از حافظه ***eeprom*** و قرار دادن محتوای آن در متغیر ***b***

جزییات دیگری نیز در برنامه وجود دارد که با بررسی برنامه میتوان آنها را متوجه شد . نمیتوان گفت که این برنامه یک برنامه صد در صد کامل است ولی از هر لحاظ سعی شده است که کلیه جوانب کار رعایت شود و شرایط مختلف در نظر گرفته شود . با این حال میتوان روی برنامه مانور داد و آنرا کاملتر کرد . در صفحه بعد برنامه اصلی سیستم به زبان بیسیک ارایه شده است .

```
$regfile = "m16def.dat"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = _
Portd.7 , E = Portd.1 , Rs = Portd.0
Config Lcd = 16 * 2
Cls
Cursor Off
Config Kbd = Porta , Debounce = 20 , Delay = 100
Config Spi = Hard , Interrupt = On , Data Order = Msb , Master = _
No , Polarity = Low , Phase = 0 , Clockrate = 16
Spiinit
Config Watchdog = 16
Stop Watchdog
Config Pinb.0 = Output
Config Pinb.1 = Output
Config Pinb.2 = Output
Config Pinb.3 = Output
Config Pind.2 = Output
Config Pind.3 = Output
Config Portc = Output
Dim A As Byte , B As Byte , C As Byte , Z As Byte
Dim X As Bit
Dim Hour As Byte , Minuit As Byte , Second As Byte
Dim Year As Byte , Mounth As Byte , Day As Byte
Dim U(22) As Eram Word At &H1BD
Dim Ep(6) As Byte
Dim N As Byte , D As Byte
Dim R As Eram Byte At &H1F4
Dim L1 As Word , L2 As Word
Dim P As Word , P2 As Word , P1 As Word
Lcd "BE APARTEMAN "
Lowerline
Lcd "KHOSH AMADID"
Z = 16
B = 10
Gosub Eprom_read

Again:
```

```
A = Getkbd()
If A > 15 Then Goto Again
If A = 0 Then
B = 7
If C = 3 Then Gosub Next7
If X = 0 Then
C = 0
Cls
Home
Lcd "ROOSTAMI"
Lowerline
Lcd "vahed 7"
End If
End If

If A = 1 Then
B = 8
If C = 3 Then Gosub Next8
If X = 0 Then
C = 0
Cls
Home
Lcd "BARATI"
Lowerline
Lcd "vahed 8"
End If
End If

If A = 2 Then
B = 9
If C = 3 Then Gosub Next9
If X = 0 Then
C = 0
Cls
Home
Lcd "GHOMAYSHI"
Lowerline
Lcd "vahed 9"
End If
End If
```

```
If A = 4 Then
B = 4
If C = 3 Then Gosub Next4
If X = 0 Then
C = 0
Cls
Home
Lcd "REZAEI"
Lowerline
Lcd "vahed 4"
End If
End If

If A = 5 Then
B = 5
If C = 3 Then Gosub Next5
If X = 0 Then
C = 0
Cls
Home
Lcd "SAHARDOOST"
Lowerline
Lcd "vahed 5"
End If
End If

If A = 6 Then
B = 6
If C = 3 Then Gosub Next6
If X = 0 Then
C = 0
Cls
Home
Lcd "AMJADI"
Lowerline
Lcd "vahed 6"
End If
End If
```

```
If A = 8 Then
B = 1
If C = 3 Then Gosub Next1
If X = 0 Then
C = 0
Cls
Home
Lcd "ASHORI"
Lowerline
Lcd "vahed 1"
End If
End If

If A = 9 Then
B = 2
If C = 3 Then Gosub Next2
If X = 0 Then
C = 0
Cls
Home
Lcd "HOSAINI"
Lowerline
Lcd "vahed 2"
End If
End If

If A = 10 Then
B = 3
If C = 3 Then Gosub Next3
If X = 0 Then
C = 0
Cls
Home
Lcd "NADERI"
Lowerline
Lcd "vahed 3"
End If
End If

If A = 13 Then
```

```
B = 0
If C = 3 Then Gosub Next10
If X = 0 Then
C = 0
Cls
Home
Lcd "AMIRI"
Lowerline
Lcd "vahed 10"
End If
End If

If A = 15 Then
If X <> 1 Then

If B = 1 Then
Set Portb.0
Waitms 500
Reset Portb.0
Incr C
End If

If B = 2 Then
Set Portb.1
Waitms 500
Reset Portb.1
Incr C
End If

If B = 3 Then
Set Portb.2
Waitms 500
Reset Portb.2
Incr C
End If

If B = 4 Then
Set Portb.3
Waitms 500
Reset Portb.3
```

```
Incr C
End If

If B = 5 Then
Set Portd.3
Waitms 500
Reset Portd.3
Incr C
End If

If B = 6 Then
Set Portc.0
Waitms 500
Reset Portc.0
Incr C
End If

If B = 7 Then
Set Portc.1
Waitms 500
Reset Portc.1
Incr C
End If

If B = 8 Then
Set Portd.2
Waitms 500
Reset Portd.2
Incr C
End If

If B = 9 Then
Set Portc.6
Waitms 500
Reset Portc.6
Incr C
End If

If B = 0 Then
```

```
Set Portc.7
Waitms 500
Reset Portc.7
Incr C
End If

If C = 3 Then
X = 1
P1 = P
Cls
Home
Lcd "enter payam"
Goto Again
End If

End If
End If

If A = 14 Then
If D > 0 Then
Shiftlcd Right
Waitms 100
Decr P
Decr Z
Decr D
End If
End If

If A = 12 Then
If D > 0 Then
B = 10
D = 0
X = 0
C = 0
Cls
Z = 16
Gosub Spi_inter
Waitms 150
End If
End If
```

```
If A = 7 Then
If X <> 1 Then
Incr N
If N = 23 Then N = 1
If N > 0 Then
If N < 23 Then
Gosub Ready_eprom
End If
End If
End If
End If

If A = 11 Then
If X <> 1 Then
Decr N
If N = 0 Then N = 22
If N = -1 Then N = 22
If N > 0 Then
If N < 23 Then
Gosub Ready_eprom
End If
End If
End If
End If

If A = 3 Then
If D = 0 Then
Reset Watchdog
Start Watchdog
End If
End If

Goto Again

End


Next7:
```

```
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Next8:
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Next9:
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
```

```
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Next4:
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Next5:
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Next6:
```

```
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Next1:
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Next2:
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
```

```
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Next3:
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Next10:
X = 1
P2 = P - P1
If P2 = 14 Then Return
Incr D
If P = P1 Then Cls
Shiftlcd Left
Waitms 100
Incr Z
Locate 1 , Z
Lcd B
Incr P
Writeeeprom B , P
Waitms 4
Return

Spi_inter:
```

```
Spiin Year , 1
Spiin Mounth , 1
Spiin Day , 1
Spiin Hour , 1
Spiin Minuit , 1
Spiin Second , 1

Select Case P
Case 1 To 20:
R = 1
U(1) = P
Case 21 To 40:
R = 2
U(2) = P
Case 41 To 60:
R = 3
U(3) = P
Case 61 To 80:
R = 4
U(4) = P
Case 81 To 100:
R = 5
U(5) = P
Case 101 To 120:
R = 6
U(6) = P
Case 121 To 140:
R = 7
U(7) = P
Case 141 To 160:
R = 8
U(8) = P
Case 161 To 180:
R = 9
U(9) = P
Case 181 To 200:
R = 10
U(10) = P
Case 201 To 220:
R = 11
U(11) = P
```

```
Case 221 To 240:
R = 12
U(12) = P
Case 241 To 260:
R = 13
U(13) = P
Case 261 To 280:
R = 14
U(14) = P
Case 281 To 300:
R = 15
U(15) = P
Case 301 To 320:
R = 16
U(16) = P
Case 321 To 340:
R = 17
U(17) = P
Case 341 To 360:
R = 18
U(18) = P
Case 361 To 380:
R = 19
U(19) = P
Case 381 To 400:
R = 20
U(20) = P
Case 401 To 420:
R = 21
U(21) = P
Case 421 To 440:
R = 22
U(22) = P
End Select

Incr P
Writeeeprom Mounth , P
Waitms 4
Incr P
Writeeeprom Day , P
Waitms 4
```

```
Incr P
Writeeeprom Hour , P
Waitms 4
Incr P
Writeeeprom Minuit , P
Waitms 4
Incr P
Writeeeprom Second , P
Waitms 4
Incr P
Writeeeprom Year , P
Waitms 4
Cls
Home
Lcd "OK"
Lowerline
Lcd "payam saved"
Wait 2
Cls
Lcd "BE APARTEMAN "
Lowerline
Lcd "KHOSH AMADID"
Gosub Eprom_read


Eprom_read:
If R = 1 Then P = 20
If R = 2 Then P = 40
If R = 3 Then P = 60
If R = 4 Then P = 80
If R = 5 Then P = 100
If R = 6 Then P = 120
If R = 7 Then P = 140
If R = 8 Then P = 160
If R = 9 Then P = 180
If R = 10 Then P = 200
If R = 11 Then P = 220
If R = 12 Then P = 240
If R = 13 Then P = 260
If R = 14 Then P = 280
```

```
If R = 15 Then P = 300
If R = 16 Then P = 320
If R = 17 Then P = 340
If R = 18 Then P = 360
If R = 19 Then P = 380
If R = 20 Then P = 400
If R = 21 Then P = 420
If R = 22 Then P = 0
Return

Ready_eprom:

Cls
B = 10
C = 0
L1 = U(n) + 1
L2 = L1 + 5
Z = 0
For P2 = L1 To L2
Incr Z
Readeeprom Ep(z) , P2
Next
Locate 2 , 9
Lcd Ep(4) ; ":" ; Ep(5) ; ":" ; Ep(6)
If Ep(5) < 10 Then
Cls
Locate 2 , 9
Lcd Ep(4) ; ":" ; "0" ; Ep(5) ; ":" ; Ep(6)
If Ep(6) < 10 Then
Cls
Locate 2 , 9
Lcd Ep(4) ; ":" ; "0" ; Ep(5) ; ":" ; "0" ; Ep(6)
End If
End If
If Ep(6) < 10 Then
Cls
Locate 2 , 9
Lcd Ep(4) ; ":" ; Ep(5) ; ":" ; "0" ; Ep(6)
If Ep(5) < 10 Then
Cls
```

```
Locate 2 , 9
Lcd Ep(4) ; ":" ; "0" ; Ep(5) ; ":" ; "0" ; Ep(6)
End If
End If
Locate 2 , 1
Lcd Ep(1) ; "/" ; Ep(2) ; "/" ; Ep(3)
Z = 0
L1 = N - 1
L2 = L1 * 20
Incr L2
For P2 = L2 To U(n)
Readeeprom Ep(1) , P2
Incr Z
Locate 1 , Z
Lcd Ep(1)
Next
Waitms 150
Z = 16
Return
```

۵-۲. تنظیم تاریخ و زمان

برای تنظیم تاریخ و زمان برنامه ای نوشته و آنرا داخل میکرو ***mega8*** پروگرم نمـوده ایـم . روش کـار بدین صورت بوده که از کریستال ساعت خارجی استفاده کرده ایم تا زمان دقیق یک ثانیـه را بـه عنـوان زمان مبنا تولید کنیم . تایمر / کانتر دو یا صفر در تعدادی از میکروهای ***avr*** میتواند به صورت آسـنکرون کار کند و به عنوان ساعت وقت واقعی یا ***rtc*** زمان وتقویم را حتی در حالت ***power save*** تنظیم کند . در این حالت کریستالی جدا از کریستال میکرو به مقدار ***32.768 khz*** در پایه های ***tosc1*** و ***tosc2*** برای تامین کلاک تایمر قرا میگیرد . در صورتی که ***prescaler = 128*** (تقسیم فرکانس) بـرای تـایمر در نظـر گرفتـه شود تایمر پس از زمان ***1s*** سرریز شده و برنامه ***rtc*** در زیر برنامه وقفـه ***(isr)*** را بـه روز مـی کنـد . بـرای کاهش مصرف تغذیه ، میکرو در حالت ***power save*** قرار گرفته و پس از یک شدن پرچم سرریزی تـایمر از مُد ***power save*** به حالت ***active*** رفتـه و ***isr*** مربوطـه را انجـام داده و دوبـاره بـه حالـت ***power save*** برمیگردد .

در این برنامه از دستور ***config clock*** استفاده شده است و این پیکره بندی توسط ***bascom*** برای ساده تر کردن کار با تایمر در مُد آسنکرون طراحی شده است . این دستور به طور خورکار تایمر را در مُد آسنکرون پیکره بندی کرده وزمان ***1s*** را تولید می کند :

config clock = soft , gosub sectic

بعد از سریزی تایمر بعد از یک ثانیه زیربرنامه اختیاری ***sectic*** اجرا میشود ، استفاده از این برچسب اختیاری است .

در برنامه شش متغیر برای مقادیر سال ، ماه ، روز، ساعت ، دقیقه و ثانیه در نظر گرفته شده است . برا ی تنظیمات دستی نیز ۳ عدد شستی به پینهای ***pind.0*** ، ***pind.1*** و ***pind.2*** متصل شده که طبق اتصال سخت افزار در حالت عادی مقادیر هر یک از این پینها منطقی ۱ میباشد و با زدن شستی مقدار صفر منطقی را به خود میگیرد .

در برچسب ***show*** مقادیر بر روی ***lcd*** نمایش داده میشوند و همزمان نیز به پورت ***spi*** اطلاعات ارسال میشود . پرش به این برچسب هر یک ثانیه یک بار صورت می گیرد . این میکرو در ارتباط ***spi*** نقش ***master*** یعنی ارسال کننده داده را بر عهده دارد . اگر ***pind.0=0*** آنگاه وارد بخش تنظیمات تاریخ و زمان میشویم که با صفر شدن ***pind.1*** میتوان هر یک از متغیرها را افزایش داد و این تغییرات نیز همزمان بر روی ***lcd*** نمایش داده میشوند . اگر ***pind.2=0*** آنگاه به روند عادی برنامه برگشته و تاریخ و زمان نمایش داده می شوند .

در صفحه بعد این برنامه به زبان بسسیک ارایه شده است .

```
$regfile = "m8def.dat"
$crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portc.3 , Db5 = Portc.2 , Db6 = Portc.1 , Db7 = _
Portc.0 , E = Portc.4 , Rs = Portc.5
Cls : Home : Cursor Off
Config Clock = Soft , Gosub = Sectic
Disable Interrupts
Config Portd = &B11111000
Dim Flag As Bit , S As Byte , M As Byte , H As Byte , Y As Word
Dim Mo As Byte , D As Byte , Slc As Byte , C As Byte
Config Spi = Hard , Interrupt = On , Data Order = Msb , Master = _
Yes , Polarity = Low , Phase = 0 , Clockrate = 16
Spiinit
Slc = 0
Y = 1385
Mo = 1
D = 1
H = 12
Flag = 0
Dim Ge As Byte

Scan:
If Pind.0 = 0 Then Slc = Slc + 1
If Slc = 5 Then Slc = 0
If Pind.2 = 0 Then
Flag = 0
Goto Waitls
End If
On Slc Goto Hour , Minute , Year , Month , Day
Goto Scan

Waitls:
Enable Interrupts
Do
Ge = 10
If Pind.0 = 0 Then
Flag = 1
Goto Scan
```

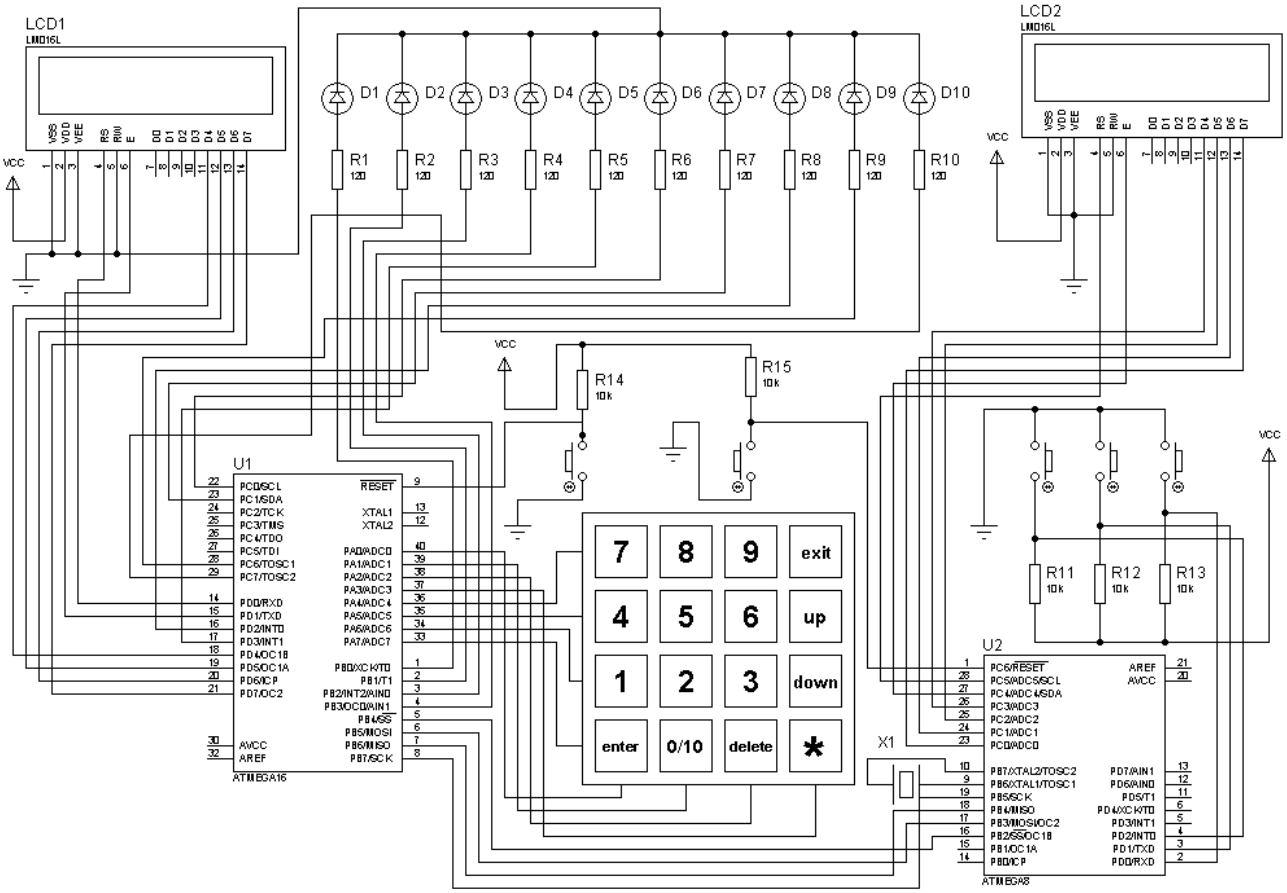
```
End If
Loop

Sectic:
If Flag = 0 Then Gosub Show
Incr S
If S > 59 Then
S = 0
Incr M
Cursor Off
Shiftcursor Left , 2
If M > 59 Then
Incr H
M = 0
If H > 23 Then
H = 1
Incr D
End If
End If
End If
If Mo < 7 Then Goto First
If Mo => 7 Then Goto Second

First:
If D > 31 Then
D = 1
Incr Mo
If Mo > 12 Then Mo = 1
End If

Second:
If D > 30 Then
D = 1
Incr Mo
If Mo > 12 Then Mo = 1
End If
Return

Show:
C = Y - 1300
```

```
Spiout C , 1
Spiout Mo , 1
Spiout D , 1
Spiout H , 1
Spiout M , 1
Spiout S , 1
Cls
Locate 2 , 1
Lcd "TIME:"
Locate 2 , 7
Lcd H ; ":" ; M ; ":" ; S
If M < 10 Then
Cls
Locate 2 , 1
Lcd "TIME:"
Locate 2 , 7
Lcd H ; ":" ; "0" ; M ; ":" ; S
If S < 10 Then
Cls
Locate 2 , 1
Lcd "TIME:"
Locate 2 , 7
Lcd H ; ":" ; "0" ; M ; ":" ; "0" ; S
End If
End If
If S < 10 Then
Cls
Locate 2 , 1
Lcd "TIME:"
Locate 2 , 7
Lcd H ; ":" ; M ; ":" ; "0" ; S
If M < 10 Then
Cls
Locate 2 , 1
Lcd "TIME:"
Locate 2 , 7
Lcd H ; ":" ; "0" ; M ; ":" ; "0" ; S
End If
End If
Locate 1 , 1
Lcd "DATE:"
```

```
Locate 1 , 7
Lcd Y ; "/" ; Mo ; "/" ; D
Return

End


Hour:
If Pind.1 = 0 Then H = H + 1
If H => 24 Then H = 0
Lcd "HOUR =" ; H
Spiout H , 1
Waitms 200
Cls : Home
Goto Scan

Minute:
If Pind.1 = 0 Then M = M + 1
If M => 60 Then M = 0
Lcd "MINUTE =" ; M
Spiout M , 1
Waitms 200
Cls : Home
Goto Scan

Year:
If Pind.1 = 0 Then Y = Y + 1
If Y = 1400 Then Y = 1385
C = Y - 1300
Lcd "YEAR =" ; Y
Spiout C , 1
Waitms 200
Cls : Home
Goto Scan

Month:
If Pind.1 = 0 Then Mo = Mo + 1
If Mo = 13 Then Mo = 1
Lcd "MOUNTH =" ; Mo
```

```
Spiout Mo , 1
Waitms 200
Cls : Home
Goto Scan

Day:
If Pind.1 = 0 Then D = D + 1
If D = 31 Then D = 1
Lcd "DAY =" ; D
Spiout D , 1
Waitms 200
Cls : Home
Goto Scan
```

۵-۳. شماتیک مدار



LCD1 , LCD2 = 2 * 16
OHM R1 . . . . . R10 = 120
R11 , R12 , R13 = 10 KOHM
U1= ATMEGA16
U2 = ATMEGA8
X1 = 32.768 KHZ
D1 . . . . . D10 = LED
VCC = +5V

۶. منابع

[1] میکروکنترولرهای ***avr*** ، مهندس علی کاهه ، ۱۳۸۴

[2] میکروکنترولرهای ***avr*** و کاربردهای آنها ، مهندس امیر ره افروز ، ۱۳۸۵

هادی رستمی سیاهگلی، فارغ التحصیل کارشناسی الکترونیک

(akharinrahgozar.h@gmail.com)