به نام خدا

گروه برق دانشکده فنی و حرفه ایی

شهید چمران گرگان

خـــودآمـــوز کاربـــردی میـــکروکنتـــرلرهای

زبان بیسیک (BascomAVR)

همراه با بیش از ۵۰ مثال و پروژه های کاربردی

ویژه :

دانشجویان

دانش آموزان

و سایر علاقه مندان رشته برق و الکترونیک

با مطالعه کامل و انجام مثال ها و آزمایش های این مجموعه، قادر به طراحی و ساخت پروژه های زیر خواهید بود:

فلاشر، چراغ راهنما، ارگ الکترونیکی ، اجرای انیمیشن روی LCD کاراکتری، ساعت دیجیتال، تایمر، دماسنج، ترموستات، اهم متر، فرکانس متر، ولت متر، ماشین حساب، تابلو روان، ارسال SMS و ذخیره سازی آن در حافظه EEPROM میکروکنترلر، راه اندازی استپر موتور، کنترل دور موتور DC با PWM، ولوم دیجیتال، دیمر دیجیتال، قفل رمز الکترونیکی ۸ رقمی، ارتباط میکروکنترلر با کامپیوتر، ربات مسیریاب و ...

برنامه نویسی و شبیه سازی تمامی آزمایش ها و پروژه ها بر مبنای آی سی میکروکنترلر AVR سری Mega16 انجام شده ، چون این سری از خانواده AVR دارای امکانات بسیار خوبی بوده و برای آموزش و شروع کار کردن با میکروکنترلر بسیار مناسب است.

فایل های شبیه سازی شده و سورس کامل کلیه آزمایش ها و پروژه ها به همراه تصویر مدارها و نیز دیتاشیت قطعات معرفی شده در ضمیمه همراه این مجموعه قرار داده شده است.

گردآوری و تنظیم: امیر کشیری **پاییز ۱۳۹۱**

Amir.Kashiri@Yahoo.com

فهرســت

عنوان **صفحه**

مقدمـــه

با توجه به فراگیر شدن روزافزون استفاده از میکروکنترلرها در اکثر سیستم های الکترونیکـی و نیـاز دانـشجویان بـه فراگیری آن ، این مجموعه را برای یادگیری این قشر و همینطور سایر علاقه منـدان در سـطح مقـدماتی و متوسـط تهیه کردیم. از مشخصات بارز این مجموعه می توان به سادگی و کاربردی بودن مطالب آن و ارائه پـروژه بـه همـراه توضیحات کامل در پایان هر بخش آن اشاره کرد. البته این نکته لازم به ذکر است که اکثر مطالب و پروژه های ایـن مجموعه را از کتابهای معروف موجود در بازار و همینطور مقاله های موجود در سایت های اینترنتی گـردآوری کـرده ایم ، بطوری که از میان آن ها بهترین و کاربردی ترین مباحث برگزیده شده ، تا شخصی که می خواهد برای اولـین بار کار کردن با میکروکنترلر AVR را تجربه کند دچار سردرگمی و درگیر مباحث پیچیده نشود.

لازم به ذکر است افرادی که آشنایی کافی با قطعات و مدارهای الکترونیکی را ندارند می بایست قبل از مطالعـه ایـن مجموعه به ضمیمه های ۲ و ۴ مراجعه کنند تا در هنگام انجام آزمایش ها و پروژه ها به مشکلی برنخورند.

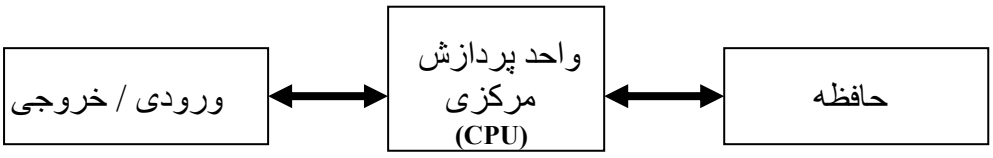
فصل ۱

تعاریف مقدماتی

۱ – ۱ تعریف کامپیوتر:

کامپیوتر به معنای محاسبه کردن می باشد و دستگاهی ست که با توجه به نیاز انسان قابل برنامه ریزی است و توانایی انجام محاسبه و نگهداری نتایج محاسبه را در اختیار دارد. به زبان ساده تر یک کامپیوتر اطلاعات (داده یا دیتا) را از ورودی ها گرفته ، پس از پردازش (انجام محاسبات روی داده ها) و ذخیره آنها در صورت نیاز بر روی حافظه ، فرمان هایی را به خروجی ها اعمال می کند.

بلوک دیاگرام بسیار ساده از یک سیستم کامپیوتر:



۱ – ۲ تعریف میکروپروسسور:

تراشه ایی که شامل یک واحد پردازشگر مرکزی به نام CPU (Central Processing Unit) می باشد. خود CPU نیز از چندین قسمت تشکیل شده است.

۱ – ۳ وظایف CPU به شرح زیر است:

۱- انجام محاسبات ریاضی ، منطقی و بیتی

۲- انجام دادن دستورالعمل ها

۳- ارتباط با حافظه

۴- کنترل تجهیزات جانبی

۵- پاسخ دادن به وقفه ها

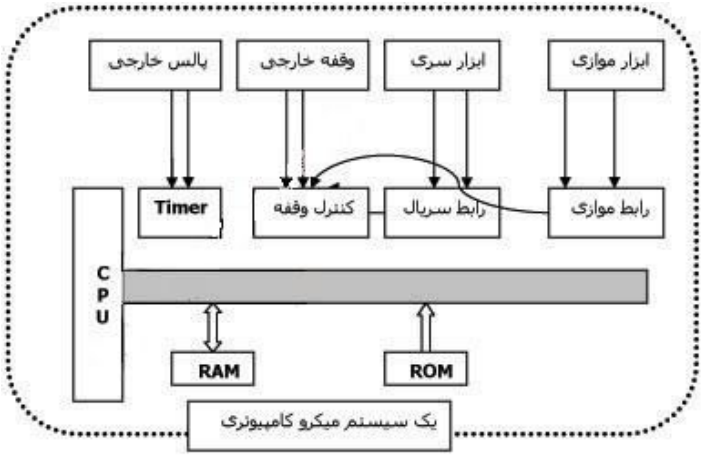
۴ – ۱ تعریف میکروکامپیوتر:

عبارتست از یک میکروپروسسور ، مدارها و قطعات ورودی – خروجی جانبی و حافظه ها که در کنار یکدیگر قرار گرفته اند تا کامپیوتری کوچک را به منظور تحلیل اطلاعات و کاربردهای کنترلی شکل دهند.

۵ – ۱ تعریف میکروکنترلر:

هنگامی که قطعات سازنده یک میکروکامپیوتر در یک تراشه سیلیکونی و در کنار یکدیگر قرار گیرند ، یک میکروکنترلر به وجود می آید. در واقع آی سی میکروکنترلراز یک CPU به همراه مقدار ثابتی از RAM ، ROM ، تایمر، رجیستر ، وقفه ها ، کلاک پالس ، پورت های ورودی / خروجی (I / O) ، رابط سریال و موازی و ... تشکیل شده است.

به عبارت دیگر میکروکنترلر یک تراشه الکترونیکی برنامه پذیر است که استفاده از آن باعث افزایش سرعت و کارآیی مدار در مقابل حجم و هزینه کمتر می شود.



شکل ۱-۱ بلوک دیاگرام کلی میکروکنترلرها

کاربرد میکروپروسسورها و میکروکنترلرها بطور گسترده در تولید سیستم های تک منظوره می باشد. مانند پرینتر ، ماوس ، انواع ربات ها ، سنجش دما و رطوبت و ...

نکته) هدف از سیستم های تک منظوره ، کاهش توان مصرفی ، هزینه و فضای اشغالی آن است.

۶ – ۱ انواع میکروکنترلرها:

میکروکنترلرهای سری 8048 و 8051 محصول شرکت اینتل

میکروکنترلرهای سری 6811 محصول شرکت موتورولا

میکروکنترلرهای سری Z8 محصول شرکت زایلوگ

میکروکنترلرهای سری H8 محصول شرکت هیتاچی

میکروکنترلرهای سری PIC محصول شرکت میکروچیپ

میکروکنترلرهای سری 89C51 ، AVR ، ARM محصول شرکت اتمل

۷ – ۱ انواع معماری پردازشگرهای میکروکنترلرها:

معماری Cisc (Compex Instruction Set Computer)

معماری Risc (Reduced Instruction Set Computer)

نکته) از مزیت های معماری Risc می توان به سرعت بالا ، دستورالعمل های ساده ، کم و سازگاری با زبان های سطح بالا (HLL) در مقایسه با معماری Cisc که قدیمی تر نیز هست اشاره کرد. در واقع دلایل زیر باعث شده در بسیاری از موارد در طراحی پردازشگرهای امروزی ، ساختار Risc بر معماری Cisc ترجیح داده شود:

۱- سرعت پاسخ دهی بالا به وقایع پیش بینی نشده

۲- پهنای باند بالا

۳- توان های مصرفی کم تر

۴ – سخت افزار داخلی کم

و ...

نکته) در معماری Risc از باس های سه گانه و مجزای **آدرس** ، **داده** و **کنترل** برای حافظه برنامه استفاده می شود.

۸ – ۱ انواع زبان های برنامه نویسی در میکروکنترلرها:

اسمبلی

C و C##

بیسیک

پاسکال

و...

نکته) هر کدام از این زبان ها نرم افزار کامپایلر (Compiler یا مترجم) مخصوص به خود را دارند که برنامه را در محیط آن نرم افزارها نوشته و در نهایت خروجی همگی آن ها فایل هایی با پسوند هگزادسیمال (Hex.) و باینری می باشد که برای میکروکنترلرها قابل استفاده است.

مثلا کامپایلر AVR Studio مخصوص برنامه به زبان اسمبلی و C برای AVR است. کامپایلر Codevision ویژه برنامه به زبان C برای AVR است. یا کامپایلر Bascom AVR برای برنامه به زبان بیسیک می باشد و ...

نکته) کلمه بیسیک (Basic) مخفف عبارت زیر می باشد:

Beginners All-Purpose Symbolie Instruction Code

که معنی آن می شود: زبان برنامه نویسی نسبتا ابتدایی ، اما انعطاف پذیر و ساده.

نکته) برای انتقال فایل های هگزادسیمال به داخل میکروکنترلر از وسیله ایی به نام پروگرامر استفاده می کنیم که خود آن استانداردها و انواع مختلفی دارد ، مثلا STK 200/300 یا STK 500 .

برای آشنایی بیشتر با پروگرامر و ساخت آن به ضمیمه ۱ مراجعه نمایید.

۹ – ۱ بررسی اعداد در مبناهای مختلف:

همانطور که در بالا اشاره شد ، زبان مورد استفاده برای هر آی سی میکروکنترلر ، زبان باینری یا هگزا دسیمال می باشد. در واقع چون زبان قابل درک برای ماشین های دیجیتالی (کامپیوترها) فقط ۰ و ۱ است ، لذا می بایست همه اعداد و دستورات به اعداد باینری (۰ و ۱) تبدیل شوند. (یک یعنی وصل ، صفر یعنی قطع).

۱) اعداد دهی دهی یا دسیمال (Decimal):

این اعداد همان اعداد معمولی و مورد استفاده در ریاضیات می باشد که شامل ده رقم از صفر تا ۹ می باشند.

۲) اعداد دودویی یا باینری (Binary):

این اعداد فقط به شکل 0 و 1 یا (2^N) نمایش داده می شوند که جای N تعداد بیت ها را قرار می دهیم.

۳) اعداد شانزده تایی یا هگزادسیمال (Hexadechmal):

چون کار کردن و تسلط بر اعداد بانری مشکل و احتمال خطا کردن در آن زیاد است ، از مبنای هگزادسیمال استفاده می شود تا کار خلاصه تر و کم حجم شود.

جدول تبدیل مبنای اعداد:

دسیمال	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
باینری	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
هگزادسیمال	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

جدول ۱-۱ تبدیل مبنای اعداد

بعنوان مثال داریم: $(E7BD)_{16} = (1110\ 0111\ 1011\ 1101)_2$

فصل ۲

بررسی میکروکنترلرهای AVR

میکروکنترلرهای AVR محصول شرکت اتمل (Atmel) بوده که ساختار اصلی آن در دانشگاه هاروارد و توسط دو دانشجوی نروژی به نام های Alf Egil Bogen و Vegard Wollan طراحی شده است. اولین قطعات از آنها در سال 1993 میلادی روانه بازار شد و به سرعت جای خود را در قلب طراحان مدارات میکروکنترلری باز کرد.

شکل ۱-۲ طراحان میکروکنترلر AVR

سادگی ، امکانات گسترده ، قیمت ارزان ، مصرف توان کم ، زبان های برنامه نویسی متعدد ، فناوری حافظه پیشرفته با ظرفیت بالا ، دستورالعمل های قوی و توانایی های دیگر خانواده AVR ، آن ها را به یکی از پر کاربردترین میکروکنترلرها تبدیل کرده است و شامل طیف وسیعی از تراشه های ساخت شرکت اتمل است که بر پایه فناوری CMOS ساخته شده اند. این میکروکنترلرهای ۸ بیتی از معماری کاهش دستورالعمل های کامپیوتر(Risc) و نیز زبان های برنامه نویسی سطح بالا (HLL) پیروی می کنند.

۱ – ۲ کاربردهای میکروکنترلر AVR :

این خانواده طیف وسیعی از کاربردهای آموزشی ، تجاری و صنعتی را پوشش می دهند. بعنوان مثال می توان به کاربردهای زیر اشاره نمود:

کنترل صنعتی ، کنترل از راه دور ، حسگرها ، ربات ها ، کارت خوان ها ، امکانات جانبی و خط تلفن ، تجهیزات مخابرات ، مودم ، سرورهای شبکه کامپیوتری ، کاربردهای بیسیم (RF) ، تحلیل سیگنال ، مبدل پروتکل ، موقعیت یاب جهانی (GPS) ، هشدار دهنده ها ، واقعه نگار داده ، خودروهای موتوری ، اسباب بازی ها ، کنترل زمان ، کنترل موتور، مدیریت توان ، شارژ باتری ، UPS و...

۲ – ۲ تقسیم بندی میکروکنترلرهای AVR :

۱- AT90S) اولین نوع تولید شده که نسبتا قدیمی هستند و امکانات کمی دارند.

۲- Attiny) دارای حجم کوچک و پایه های کم هستند.

۳- Atmega) دارای امکانات وسیع و دستورالعمل های قوی می باشند.

۴- CAN AVR) توانایی پشتیبانی از پروتکل CAN را دارند. (کاربرد صنعتی)

۵- LCD AVR) برای راه اندازی LCD ها هستند و دارای درایور مخصوص با کنترل وضوح تصویر می باشند.

نکته) پروتکل CAN (Control Area Network) یک ارتباط صنعتی است و فاصله طولانی را پشتیبانی می کند که نسبت به نویز مصون تر است.

۳ – ۲ بسته بندی میکروکنترلرهای AVR :

P : بسته بندی PDIP (Plastic Dual Inline Package)

S : بسته بندی SOIC (Small Outline Integrated Circuit)

M : بسته بندی MLF (Micro Lead Frame)

A : بسته بندی TQFP (Thin Quad Flat Package)

۴ – ۲ بررسی قسمت های مختلف میکروکنترلر:

۱- واحد پردازش مرکزی (CPU) ; مهمترین قسمت میکروکنترلر می باشد که فعالیت همه قسمت های میکرو را تحت کنترل دارد ، و نیز Objectcode های یک برنامه برای اجرای ترتیبی به این قسمت فرا خوانده می شوند . CPU از چند قسمت مهم زیر تشکیل شده :

ALU (Arithmetic Logic Unit) واحد محاسبه و منطق ریاضی مانند مدارات جمع و تفریق کننده می باشد. ALU به طور مستقیم با رجیسترهای R0 – R31 در ارتباط بوده و قادر به انجام سه نوع عملیات ریاضی ، منطقی و بیتی است.

CLU (Control Logic Unit) واحد کنترل بوده که ممکن است به صورت سخت افزاری یا نرم افزاری طراحی شود و مسئول رمزگشایی و تعیین نوع عملیاتی است که ALU باید انجام دهد.

رجیسترها (یا ثبات ها) که جهت ذخیره موقت داده ها قبل از رفتن به ALU و همچنین نگهداری نتایج پردازش به کار می رود.

PC یا **شمارنده برنامه** که آدرس دستورالعمل بعدی که CPU باید از حافظه بخواند را در خود نگه می دارد.

IR یا **ثبات دستورالعمل** که مسئول ذخیره قسمت عملیاتی دستورالعمل فعلی می باشد.

دکودر دستورالعمل

مولد پالس ساعت

ورودی/خروجی

۲- حافظه RAM یا SRAM ; حافظه موقت می باشد و در طی اجرای برنامه اطلاعات ورودی سیستم و خروجی آن و نیز تمام مقادیری که قرار است روی آن پردازش صورت گیرد و نیز برنامه ها در ابتدا وارد این قسمت شده و سپس در CPU پردازش می شوند. سرعت RAM بسیار زیاد است و اطلاعات وارد شده به این حافظه ممکن است بصورت تصادفی در هر قسمت از آدرس ها قرار گیرد. اطلاعات قرار گرفته در حاظه RAM با قطع منبع تغذیه از بین می روند.

۳- حافظه ROM ; حافظه ایی از نوع فقط خواندنی که قابلیت ذخیره اطلاعات را بصورت دائمی دارد و با قطع منبع تغذیه اطلاعات ذخیره شده در آن از بین نمی رود.

۴- حافظه FLASH ; مانند حافظه ROM بوده که برنامه های کاربردی (فایل های Hex) توسط پروگرامر در این حافظه ریخته شده و سپس توسط میکرو اجرا می شود. اطلاعات داخلی این حافظه فقط با برنامه نویسی و پروگرامر قابل تغییر و پاک شدن است و با قطع منبع تغذیه این اطلاعات از بین نمی رود.

حافظه Flash به دو قسمت Application و Bootloader تقسیم می شود. قسمت Application مخصوص قرارگیری برنامه های کاربردی است که توسط پراگرامر در این قسمت قرار می گیرند. ولی قسمت Bootloader دارای قابلیت خود برنامه ریزی بوده و میکروکنترلر می تواند با اجرای برنامه Bootloader برنامه کاربردی که در قسمت Application نوشته شده است را تغییر دهد. در واقع برنامه Bootloader برنامه ایی ست که قادر بوده با دنیای خارج از طریق پورت های میکرو و رابط هایی نظیر Rs232 ، UART ، I2C ، SPI ارتباط برقرار کرده و میکرو با این برنامه می تواند بدون استفاده از پروگرامر ، دیتاهای مورد نیاز خود را دریافت نموده م محتوای قسمت Application یا حتی خود Bootloader را تغییر دهد. برای برنامه نویسی در قسمت Bootloader ، فقط با زبان اسمبلی می توان کار کرد.

۵- حافظه EEPROM ; حافظه ایی با قابلیت برنامه ریزی و پاک شدن توسط جریان الکتریکی می باشد. اطلاعات این حافظه بر خلاف حافظه FLASH می تواند در طی اجرای برنامه توسط خود میکرو ذخیره ، پاک و یا تغییر کند.

۶- ثبات ها (Register) ; حافظه هایی ۸ و ۱۶ بیتی از نوع حافظه RAM بوده که برای نگهداری موقت اطلاعات مورد پردازش یا اطلاعات ورودی و خروجی استفاده می شود. ثبات ها در واقع نوعی دفترچه یادداشت برای میکرو می باشند.

۷- تایمر / کانترها (Timer/Counter) ; شمارنده های داخلی میکروکنترلر بوده که بر حسب نوع برنامه ریزی ممکن است پالس های داخلی (مد تایمر) یا پالس های خارجی (مد کانتر) را بشمارد.

۸- پورت های ورودی و خروجی (I/O) ; در میکروکنترلر رجیسترهایی به پایه های خروجی متصل می باشد که با کمک این پورت ها (I/O) بصورت سخت افزاری به وسایل مختلف متصل می شوند. در واقع این واحد مسئول ارتباط میکرو با دنیای خارج است.

۹- اوسیلاتور (OSC) ; وظیفه تامین کلاک پالس ساعت میکروکنترلر را بر عهده دارد تا قسمت های مختلف میکرو را با یکدیگر هماهنگ کند.

۱۰- واحد آنالوگ به دیجیتال (ADC) ; این واحد مقادیر پیوسته (آنالوگ) را به سیگنال های گسسته (دیجیتال) تبدیل می کند.

۱۱- واحد وقفه (Interrupts) ; ایجاد وقفه سخت افزاری در اجرای برنامه برای انجام کاری خاص در هنگام وقوع وقفه.

۱۲- مدار ریست (Reset) ; این مدار در ابتدای کار ، تمام طبقات و رجیسترهای سیستم را در وضعیت مقرر قرار می دهد.

۱۳- تایمر Watchdog ; اصلی ترین وجود آن این است که با امکاناتی که در اختیار دارد از تخریب های نرم افزاری جلوگیری می کند. این تایمر از نوع هشت بیتی بوده و منبع پالس ساعت مستقلی از سیستم دارد. در مواقعی که میکروکنترلر به علت هنگ (Hang) کردن قادر به اجرای برنامه نیست ، این تایمر با در نظر گرفتن زمان اجرای برنامه به علت توقف زیاد آن ، میکرو را ریست می کند.

۱۴- واحد RTC ; یک ساعت زمانی واقعی (Real Time Clock) است که وظیفه نگهداری تاریخ و زمان را بر عهده دارد که می تواند جهت یادآوری زمان های مختلف به کار برود.

۱۵- گذرگاه (Bus) ; مجموعه ایی از سیم ها که اطلاعات را با یک هدف مشترک حمل می کنند. در معماری کامپیوتر از سه باس می توان نام برد:

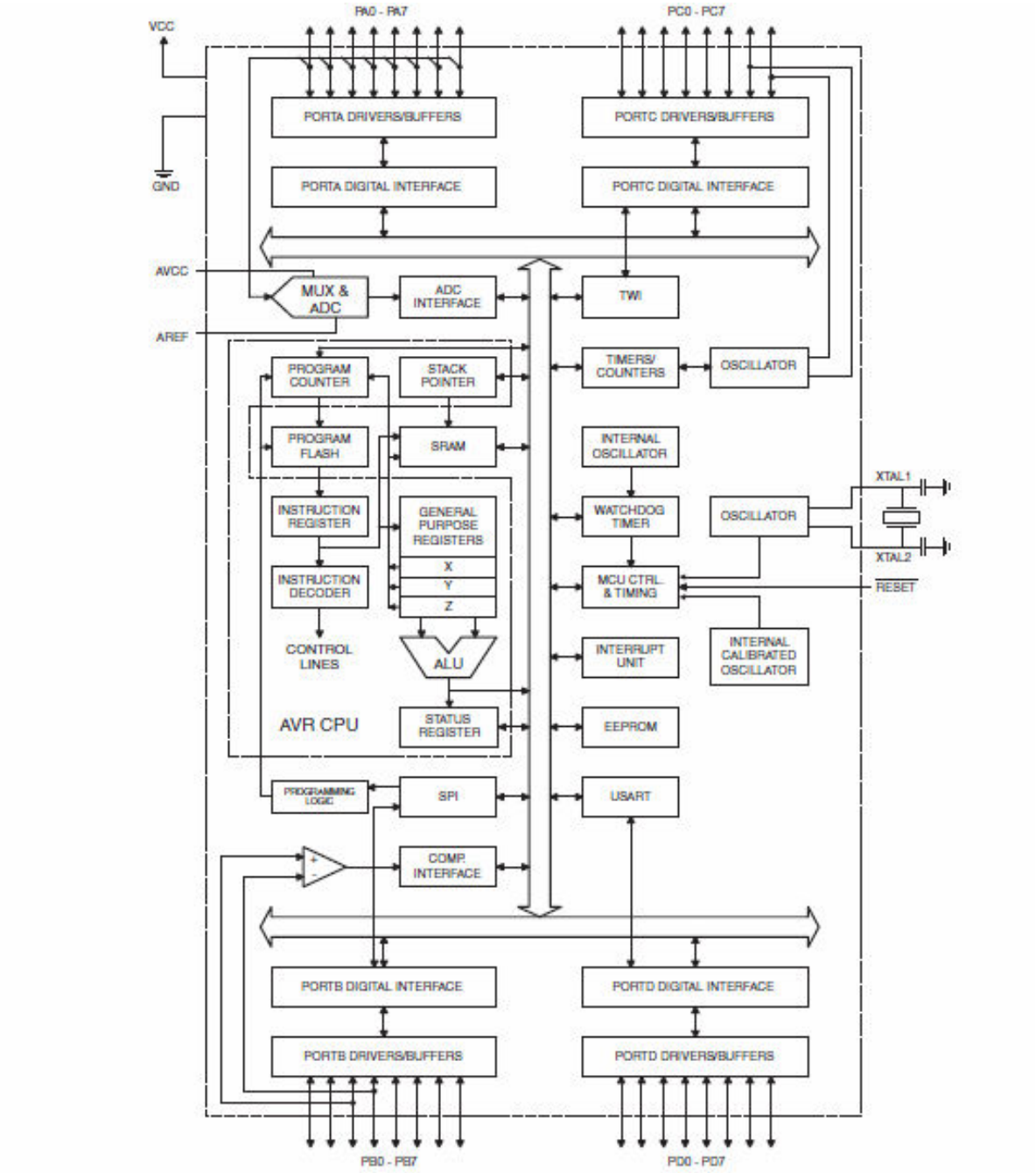
الف- گذرگاه آدرس (Address Bus) ; اطلاعات ذخیره شده در حافظه میکرو دارای آدرس های مشخص می باشند که برای خواندن یا نوشتن روی حافظه به آن ها نیاز است. در واقع برای هر عمل خواندن یا نوشتن CPU آدرس (موقعیت) داده با قرار دادن یک آدرس روی باس آدرس به حافظه ارسال می شود.

ب- گذرگاه داده (Data Bus) ; داده های یک برنامه (اعداد ، حروف ، اعمال منطقی) با دادن آدرس آن ها ، محتوایشان در گذرگاه داده قرار می گیرد. یعنی برای عملیات خواندن یک بایت داده بر روی گذرگاه داده توسط حافظه قرار داده می شود و برای عمل نوشتن یک بایت داده توسط CPU بر روی گذرگاه داده قرار می گیرد. علاوه بر سیگنال های کنترلی Read و Write سیگنال کنترلی دیگری هم با نام Clock وجود دارد که برای همزمان کردن واحدها به کار می رود.

ج- گذرگاه کنترل (Control Bus) ; تمام قسمت های یک میکروکنترلر نیاز به کنترل دارد که بوسیله این قسمت کنترل می شود. به عبارت دیگر با قرار دادن آدرس بر روی باس آدرس یک سیگنال کنترلی بر روی گذرگاه کنترل قرار میگیرد که مشخص می کند عملیات مورد نظر خواندن است یا نوشتن در حافظه.

نکته) واحد MCU (Memory Centeral Unit)، همان هسته مرکزی میکروکنترلر می باشد که قسمت هایی همچون CPU و حافظه ها را در بر می گیرد.

۵ – ۲ بلوک دیاگرام میکروکنترلر AVR :



شکل ۲-۲ بلوک دیاگرام میکروکنترلر AVR

۶ – ۲ بررسی خصوصیات و امکانات میکروکنترلر AVR :

الف – بررسی سری Mega8 :

۱- میکروکنترلری ۸ بیتی با کارایی بالا و توان مصرفی کم ، مبتنی بر معماری Risc .

۲- دارای ۲۸ پایه (نوع PDIP) و ۳۲ پایه (نوع TQFP و MLF).

۳- دارای ۲۳ خط ورودی / خروجی (I/O) قابل برنامه ریزی.

۴- ولتاژ کاری 4.7 تا 5.5 ولت.

۵- دارای ۱۳۰ دستورالعمل با کارایی بالا.

۶- ۸*۳۲ رجیستر کاربردی (۸ بیتی).

۷- اجرای ۱۶ میلیون دستورالعمل در ثانیه (MIPS) در فرکانس 16MHZ .

۸- دارای ۸ کیلوبایت حافظه فلش داخلی قابل برنامه ریزی (FLASH).

۹- قابلیت ۱۰۰۰۰ بار نوشتن و پاک کردن حافظه فلش.

۱۰- دارای ۱۰۲۴ بایت حافظه داخلی SRAM .

۱۱- دارای ۵۱۲ بایت حافظه EEPROM داخلی قابل برنامه ریزی.

۱۲- قابلیت ۱۰۰۰۰۰ بار نوشتن و پاک کردن حافظه EEPROM .

۱۳- قفل برنامه FLASH و حفاظت داده EEPROM .

۱۴- دارای ۳ تایمر (۲ تایمر ۸ بیتی و یک تایمر ۱۶ بیتی).

۱۵- دارای ۳ کانال PWM .

۱۶- دارای ۸ کانال ADC (TQFP و MLF) / ۶ کانال ADC (PDIP) .

۱۷- دارای RTC با اوسیلاتور مجزا.

۱۸- دارای یک مقایسه کننده آنالوگ داخلی.

۱۹- USART سریال قابل برنامه ریزی.

۲۰- WATCHDOG قابل برنامه ریزی با اوسیلاتور داخلی.

۲۱- ارتباط سریال SPI برای برنامه ریزی داخلی مدار.

۲۲- قابلیت ارتباط سریال SPI به صورت Master یا Slave .

۲۳- قابلیت ارتباط با پروتکل سریال دو سیمه (Two Wire).

۲۴- دارای ۵ حالت Sleep .

۲۵- منابع وقفه (Interrupt) داخلی و خارجی.

۲۶- دارای اوسیلاتور RC داخلی کالیبره شده.

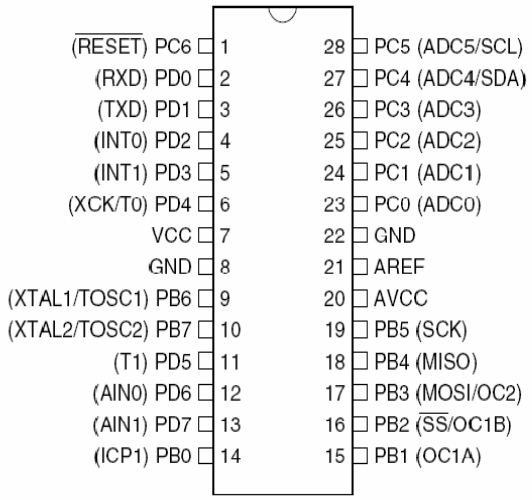
۲۷- Power – On Reset Circuit

۲۸- عملکرد کاملا ثابت.

۲۹- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS .

۳۰- برنامه ریزی از طریق ارتباط JTAG .

ترتیب پایه های Mega8 :



شکل ۳-۲ ترتیب پایه های Mega8

ب – بررسی سری Mega16 و Mega32 :

۱- میکروکنترلری ۸ بیتی با کارایی بالا و توان مصرفی کم ، مبتنی بر معماری Risc.

۲- دارای ۴۰ پایه (نوع PDIP) و ۴۴ پایه (نوع TQFP و MLF).

۳- دارای ۳۲ خط ورودی / خروجی (I/O) قابل برنامه ریزی.

۴- ولتاژ کاری ۴,۷ تا ۵,۵ ولت.

۵- دارای ۱۳۱ دستورالعمل با کارایی بالا.

۶- ۸*۳۲ رجیستر کاربردی (۸ بیتی)

۷- اجرای ۱۶ میلیون دستورالعمل در ثانیه (MIPS) در فرکانس 16MHZ .

۸- دارای ۱۶ کیلوبایت حافظه فلش داخلی قابل برنامه ریزی (برای Mega16) ، و همینطور ۳۲ کیلوبایت حافظه فلش داخلی قابل برنامه ریزی (برای Mega32).

۹- قابلیت ۱۰۰۰۰ بار نوشتن و پاک کردن حافظه فلش.

۱۰- دارای ۱۰۲۴ بایت حافظه داخلی SRAM (برای Mega16) ، ۲ کیلوبایت حافظه داخلی SRAM (برای Mega32) .

۱۱- دارای ۵۱۲ بایت حافظه EEPROM داخلی قابل برنامه ریزی (برای Mega16) ، و همینطور ۱۰۲۴ بایت حافظه EEPROM داخلی قابل برنامه ریزی (برای Mega32) .

۱۲- قابلیت ۱۰۰۰۰۰ بار نوشتن و پاک کردن حافظه EEPROM .

۱۳- قفل برنامه FLASH و حفاظت داده EEPROM .

۱۴- دارای ۳ تایمر (۲ تایمر ۸ بیتی و یک تایمر ۱۶ بیتی).

۱۵- دارای ۴ کانال PWM .

۱۶- دارای ۸ کانال ADC ۱۰ بیتی.

۱۷- دارای RTC با اوسیلاتور مجزا.

۱۸- دارای یک مقایسه کننده آنالوگ داخلی.

۱۹- USART سریال قابل برنامه ریزی.

۲۰- WATCHDOG قابل برنامه ریزی با اوسیلاتور داخلی.

۲۱- ارتباط سریال SPI برای برنامه ریزی داخلی مدار.

۲۲- قابلیت ارتباط سریال SPI به صورت Master یا Slave .

۲۳- قابلیت ارتباط با پروتکل سریال دو سیمه (Two Wire).

۲۴- دارای ۶ حالت Sleep .

۲۵- منابع وقفه (Interrupt) داخلی و خارجی.

۲۶- دارای اوسیلاتور RC داخلی کالیبره شده.

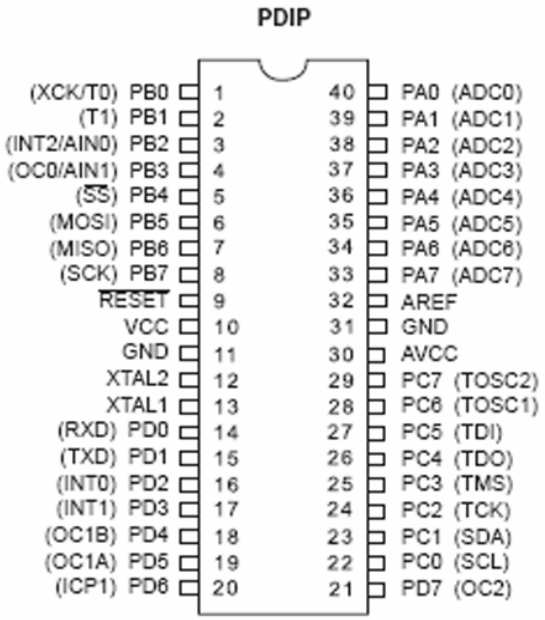
۲۷- Power – On Reset Circuit

۲۸- عملکرد کاملا ثابت.

۲۹- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS .

۳۰- برنامه ریزی از طریق ارتباط JTAG .

ترتیب پایه های Mega16 و Mega32 :



شکل ۴-۲ ترتیب پایه های Mega16 و Mega32

۲ – ۷ بررسی پایه های میکروکنترلر AVR (سری Mega16 و Mega32) :

RESET پایه ریست بوده که برای راه اندازی مجدد میکرو به کار می رود.

VCC پایه تغذیه میکرو می باشد که می بایست قطب مثبت تغذیه ۵ ولت به آن متصل شود.

GND پایه اتصال به زمین میکرو می باشد که می بایست به قطب منفی تغذیه وصل شود.

XTAL 1 , 2 پایه های دریافت کلاک ساعت میکرو می باشد که می بایست به کریستال با فرکانس کاری مورد نظر که در برنامه تعیین می شود وصل می شود.

AREF پایه مربوط به اعمال ولتاژ مرجع به میکرو در هنگام استفاده از ADC .

AVCC پایه مربوط به تغذیه ولتاژ مثبت در هنگام استفاده از مبدل ADC .

PA (0-7) پورت A دارای هشت پایه با قابلیت ورودی و خروجی (I/O).

PB (0-7) پورت B دارای هشت پایه با قابلیت ورودی و خروجی (I/O).

PC (0-7) پورت C دارای هشت پایه با قابلیت ورودی و خروجی (I/O).

PD (0-7) پورت D دارای هشت پایه با قابلیت ورودی و خروجی (I/O).

نکته) بافر خروجی کلیه پورت ها جریانی تا ۲۰ میلی آمپر را عبور می دهد که در نتیجه مـی توانـد یـک LED را مستقیما راه اندازی کند.

نکته) پایه (پین) های کلیه پورت های ورودی / خروجی (I / O) میکروکنترلر AVR داری کاربردهای دیگری هم هستند که به بررسی آن می پردازیم:

PORTA.0 – ADC0 کانال مربوط به مبدل آنالوگ به دیجیتال می باشد (کانال صفر).

PORTA.1 – ADC1 کانال مربوط به مبدل آنالوگ به دیجیتال می باشد (کانال یک).

PORTA.2 – ADC2 کانال مربوط به مبدل آنالوگ به دیجیتال می باشد (کانال دو).

PORTA.3 – ADC3 کانال مربوط به مبدل آنالوگ به دیجیتال می باشد (کانال سه).

PORTA.4 – ADC4 کانال مربوط به مبدل آنالوگ به دیجیتال می باشد (کانال چهار).

PORTA.5 – ADC5 کانال مربوط به مبدل آنالوگ به دیجیتال می باشد (کانال پنج).

PORTA.6 – ADC6 کانال مربوط به مبدل آنالوگ به دیجیتال می باشد (کانال شش).

PORTA.7 – ADC7 کانال مربوط به مبدل آنالوگ به دیجیتال می باشد (کانال هفت).

PORTB.0 – XCK , T0 ورودی کلاک برای تایمر/کانتر صفر اسـت (T0). و نیـز مـی توانـد بعنـوان کـلاک خارجی USART استفاده شود (XCK).

PORTB.1 – T1 ورودی کلاک برای تایمر/کانتر یک.

PORTB.2 – INT2 , AIN0 ورودی مثبت مقایسه کننده آنالوگ است (AIN0) و دیگر کـاربرد آن بعنـوان منبع وقفه خارجی دو است (T1).

PORTB.3 – OC0 , AIN1 ورودی منفی مقایسه کننده آنالوگ است (AIN1). و دیگـر کـاربرد آن بعنـوان خروجی مد مقایسه ای تایمر/کانتر صفر است (OC0).

PORTB.4 – SS زمانی که SPI بعنوان Slave پیکره بندی شود ، این پایه با توجه به این کـه ورودی تعریـف می شود ، در حالت Slave با Low شدن این پایه ، SPI فعال می شود. ایـن پایـه در حالـت Master می تواند ورودی یا خروجی تعریف شود.

PORTB.5 – MOSI در ارتباط SPI برای حالت Master خروجی داده و در حالت Slave ورودی داده مـی باشد.

PORTB.6 – MISO در ارتباط SPI برای حالت Master ورودی داده و در حالت Slave خروجـی داده مـی باشد.

PORTB.7 – SCK کلاک خروجی Master و کلاک ورودی Slave برای ارتباط SPI است.

PORTC.0 – SCL در زمان ارتباط I2C به عنوان خط کلاک استفاده می شود.

PORTC.1 – SDA در زمان ارتباط I2C به عنوان خط داده استفاده می شود.

PORTC.2 – TCK در زمان ارتباط JTAG استفاده می شود و دیگر نمی تـوان از ایـن پایـه بـه عنـوان I/O استفاده نمود.

PORTC.3 – TMS در ارتباط استفاده می شود و دیگر نمی توان از این پایه به عنوان I/O استفاده نمود.

PORTC.4 – TDO در زمان ارتباط JTAG ، به عنوان خروجی داده سریال عمل می کند و دیگر نمی توان از این پایه به عنوان I/O استفاده نمود.

PORTC.5 – TDI در زمان ارتباط JTAG ، به عنوان ورودی داده سریال عمل می کند و دیگر نمـی تـوان از این پایه به عنوان I/O استفاده نمود.

PORTC.6 – TOSC1 زمانی که تایمر/ کانتر ۲ در مد آسنکرون کار می کند ، به این پایـه و پایـه TOSC2 کریستال متصل می شود. در این حالت دیگر نمی توان از این پایه به عنوان I/O استفاده نمود.

PORTC.7 – TOSC2 زمانی که تایمر/ کانتر ۲ در مد آسنکرون کار می کند ، به این پایه و پایه TOSC1 کریستال متصل می شود. در این حالت دیگر نمی توان از این پایه به عنوان I/O استفاده نمود.

PORTD.0 – RDX پایه ورودی داده برای USART می باشد که مـی بایـست بعنـوان ورودی پیکـره بنـدی شود.

PORTD.1 – TXD پایه خروجی داده برای USART می باشد که می بایست به عنوان خروجی پیکره بندی شود.

PORTD.2 – INT0 پایه مربوط به دریافت منبع وقفه خارجی صفر است.

PORTD.3 – INT1 پایه مربوط به دریافت منبع وقفه خارجی یک است.

PORTD.4 – OC1B خروجی مد مقایسه ایی تایمر/ کانتر یک و نیز خروجی PWM تایمر یک است. در این حالت این پایه می بایست به عنوان خروجی پیکره بندی شود.

PORTD.5 – OC1A خروجی مد مقایسه ایی تایمر/ کانتر یک و نیز خروجی PWM تایمر یک است. در این حالت این پایه می بایست به عنوان خروجی پیکره بندی شود.

PORTD.6 – ICP بعنوان پایه ورودی CAPTURE تایمر/ کانتر یک عمل می کند.

PORTD.7 – OC2 خروجی مد مقایسه ایی تایمر/ کانتر ۲ و نیز خروجی PWM تایمر ۲ است. در این حالت این پایه می بایست به عنوان خروجی پیکره بندی شود.

نکته) در حالت عادی ، از پایه های C.2 تا C.5 نمی توان به عنوان I/O استفاده کرد و برای استفاده از آن ها حتما می بایست فیوز بیت JTAG آن ها یک شود.

۸ – ۲ فیوز بیت ها در میکروکنترلر AVR :

فیوز بیت ها قسمتی از حافظه Flash میکروکنترلر AVR هستند و بنا به برنامه ایی که به آن ها داده می شود ، امکاناتی (نظیر تنظیم دقیق کریستال ساعت ، تغییر کارآیی پورت ها و...) را در اختیار کاربر قرار می دهند که با پاک کردن برنامه میکروکنترلر ، مقدار آن ها تغیر نمی کند. در AVR حداکثر سه بایت فضا برای فیوز بیت ها در نظر گرفته شده است که شامل موارد زیر است:

۱- High Byte

۲- Low Byte

۳- فیوز بیت های توسعه یافته

تعداد فیوز بیت هایی که AVR پشتیبانی می کند ، به مدل آن بستگی دارد. منطق 0 به معنی برنامه ریزی شده و منطق 1 به معنای برنامه ریزی نشده است.

تعدادی از این فیوزبیت ها که نقش مهمتری دارند ، به شرح زیر معرفی می شوند:

۱- **فیوز بیت EESAVE :** مقدار پیش فرض آن ۱ بوده و برای این که آیا در هنگام پاک کردن میکرو حافظه EEPROM پاک شود یا نه ، از این فیوز بیت استفاده می شود. اما زمانی که مقدار این بیت را صفر کنیم ، محتویات EEPROM در هنگام پاک کردن میکرو حفظ می شود.

۲- **فیوز بیت CKOPT و CKSEL 0 ... 3 :** فیوز بیت CKOPT می تواند برای دو حالت مختلف استفاده شود. یعنی زمانی که محیط بسیار پر نویز باشد ، این بیت برنامه ریزی می شود (0) و رنج وسیعی از فرکانس ها را شامل می شود ، چون فرکانس داخلی میکروکنترلر AVR از ۸ مگاهرتز بیشتر نیست و فرکانس های بالاتر که توسط کریستال یا مدار RC خارجی ایجاد می شود را پوشش می دهد.

نحوه تعیین فرکانس	مقادیر CKSEL0 تا CKSEL3
کلاک خارجی	0
اسیلاتور RC داخلی	1,2,3,4
اسیلاتور RC خارجی	5,6,7,8
اسیلاتور کریستال فرکانس پایین خارجی	9
اسیلاتور کریستال خارجی	10,11,12,13,14,15

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

جدول ۱-۲ تنظیمات فیوزبیت برای انتخاب نوع اسیلاتور میکروکنترلر

تنظیم فیوز بیت برای استفاده از اوسیلاتور RC داخلی میکرو بصورت زیر است: (دراین حالت CKOPT برنامه ریزی نمی شود)

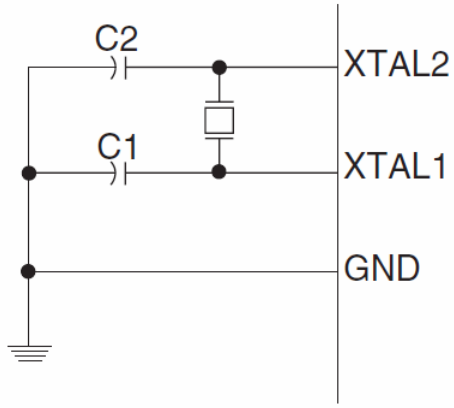
CKSEL3..0	Nominal Frequency (MHz)
0001[1]	1.0
0010	2.0
0011	4.0
0100	8.0

جدول ۲-۲ تنظیمات فیوزبیت برای انتخاب کریستال داخلی

تنظیم فیوز بیت برای استفاده از کریستال خارجی برای میکرو بصورت زیر است :

CKOPT	CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101[1]	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

جدول ۲-۳ تنظیمات فیوزبیت برای انتخاب کریستال خارجی



شکل ۲-۵ اتصال کریستال خارجی به میکروکنترلر

۳- فیوز بیت JTAG : این فیوز بیت که به صورت پیش فرض در حالت برنامه ریزی شده می باشد ، برای ارتباط میکرو با پورت موازی کامپیوتر (Parallel) تعبیه شده است. و این پیش فرض بودن آن باعث شده تا پورت C میکرو برای ارتباط I / O در دسترس نباشد و اگر نیاز باشد تا از این پورت استفاده گردد می بایست این فیوز بیت را از حالت برنامه ریزی خارج نمود.

نکته) کلیه تغییرات در فیوزبیت ها را در هنگام پروگرام کردن برنامه می توان اعمال کرد.

فصل ۳

شروع کار با میکروکنترلر AVR

۳ – ۱ طریقه آماده به کار شدن میکروکنترلر AVR :

با اتصال این سه پین به نقاط مورد نظر میکروکنترلر فعال می شود.

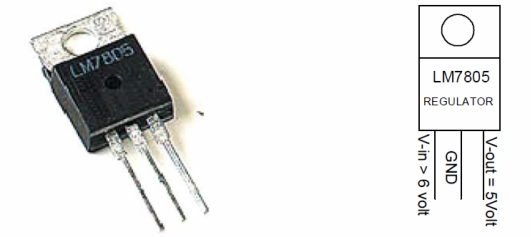
۱- اتصال پین VCC به قطب مثبت منبع تغذیه ۵ ولت.

۲- اتصال پین GND به خط زمین (قطب منفی) منبع تغذیه.

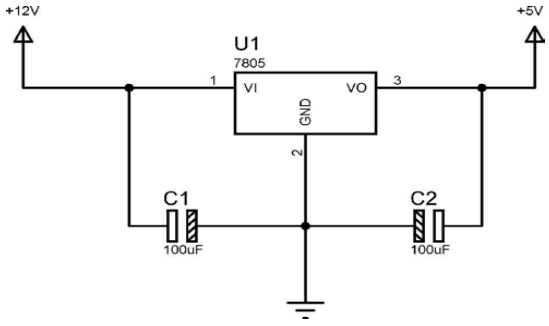
۳- اتصال پین Reset به مدار مربوط به آن. (یا اتصال این پین به قطب مثبت تغذیه به واسطه یک عدد مقاومت ۴,۷ کیلو اهمی)

۳ – ۲ مدار منبع تغذیه ۵ ولت:

این مدار از یک عدد آی سی رگولاتور به شماره 7805 و یک خازن به ظرفیت ۱۰۰ میکروفاراد تشکیل شده که با اعمال هر ولتاژ بیشتر از ۵ ولت به ورودی رگولاتور ، در خروجی آن یک ولتاژ ۵ ولت صاف شده ایی را تحویل می گیریم.

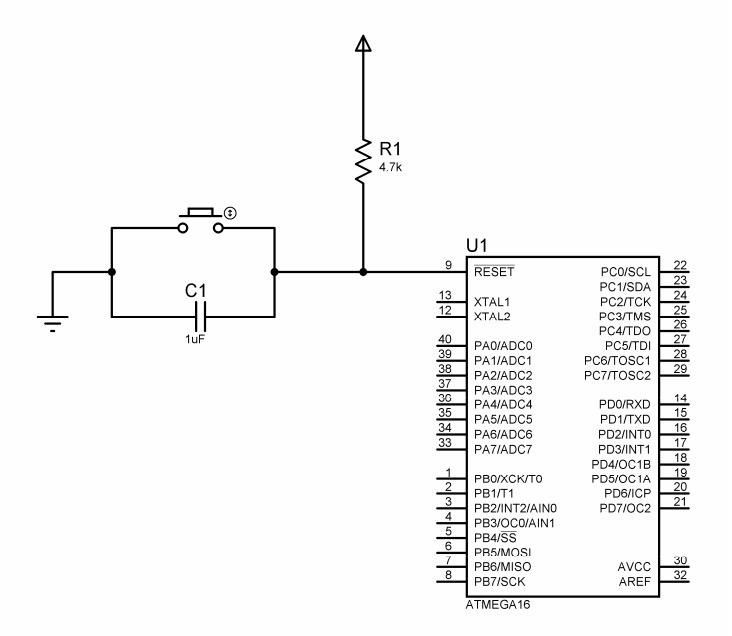


شکل ۳-۱ آی سی رگولاتور



شکل ۳-۲ نحوه اتصال آی سی رگولاتور به تغذیه

۳ – ۳ مدار ریست (Reset) :



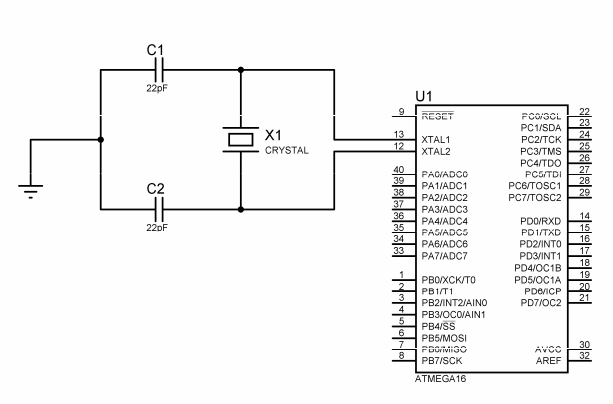
شکل ۳-۳ مدار Reset

۴ – ۳ اتصال کریستال به میکروکنترلر:

برای کار کردن هر میکروکنترلری می بایست یک کریستال با فرکانس کاری مشخص (که در برنامه نویسی تعیین می شود) به پین های مختص آن در میکرو اتصال یابد.

شکل ۴-۳ کریستال

در میکروکنترلرهای AVR این کریستال در داخل خود آی سی میکروکنترلر وجود دارد (نوسان ساز RC) که در فرکانس های ۱ ، ۲ ، ۴ و ۸ مگاهرتز به خوبی کار می کند. ولی برای فرکانس های بالاتر از ۸ مگاهرتز حتما می بایست یک کریستال خارجی به پین های XTAL1 و XTAL2 متصل شود.



شکل ۵-۳ اتصال کریستال خارجی به میکروکنترلر

به این نکته هم باید توجه شود که برای تنظیم دقیق کریستال در فرکانس های بالا می بایست فیوزبیت های میکروکنترلر را تغییر داد.

۵ – ۳ الگوریتم برنامه نویسی:

به روش حل یک مساله الگوریتم گفته می شود و اینکه چه راه حلی برای یک مساله بدست آوریم تا نسبت به بقیه راه حل ها بهینه باشد. الگوریتم برنامه نویسی برای میکروکنترلر AVR به زبان بیسیک به روش زیر است:

۱- معرفی نوع میکروکنترلر	$Regfile = "M16def.dat"
۲- تعیین فرکانس (هرتز)	$Crystal = 1000000
۳- پیکره بندی ها	Config …
۴- تعریف متغییرها	Dim … as …
۵- برنامه اصلی	Do
۶- پایان برنامه	END
۷- زیر برنامه ها (در صورت نیاز)	Label:

نکته) رعایت ترتیب الگوریتم در برنامه نویسی پروژه ها بسیار مهم می باشد.

۶ – ۳ انواع داده برای متغییرهای حافظه:

متغییرها در واقع بخشی از رجیسترهای (دفترچه یادداشت) میکروکنترلر بوده کـه داده هـای ورودی بـرای انجـام محاسبات ریاضی و پردازش بصورت موقت در داخل آن قـرار مـی گیرنـد. نتیجـه محاسـبات نیـز در داخـل همـین متغییرها قرار می گیرد تا در نهایت به خروجی میکروکنترلر ارسال شوند.

۱- Bit) تک بیتی (0 یا 1)

۲- Byte) هشت بیتی (0 تا 255)

۳- Word) شانزده بیتی (0 تا 65535)

۴- Integer) شانزده بیتی (32767 – تا 32767)

۵- Long) سی و دو بیتی (214783647 – تا 214783647)

۶- Single) سی و دو بیتی ($1.5*10^{-45}$ تا $3.4*10^{38}$)

۷- String) متغییر رشته ایی بایت (0 تا 254)

۸- Double) اعداد صحیح و اعشاری را شامل می شود ($5*10^{-324}$ تا $1.7*10^{308}$)

نکته) برای تعریف هر متغییر در برنامه می توان از یک حرف یا کلمه دلخواه در برنامه استفاده کرد.

نکته) زمانی که به تعداد زیادی متغییر با یک نوع داده نیاز داشتیم ، می توان از متغییر آرایـه ایـی کـه بـه صـورت Name (Var) است استفاده کنیم.

بعنوان مثال داریم:

```
Config  A(5)  as Byte
```

در این حالت ۵ متغییر بصورت زیر تعریف می شود:

A(0) , A(1) , A(2) , A(3) , A(4)

فصل ۴

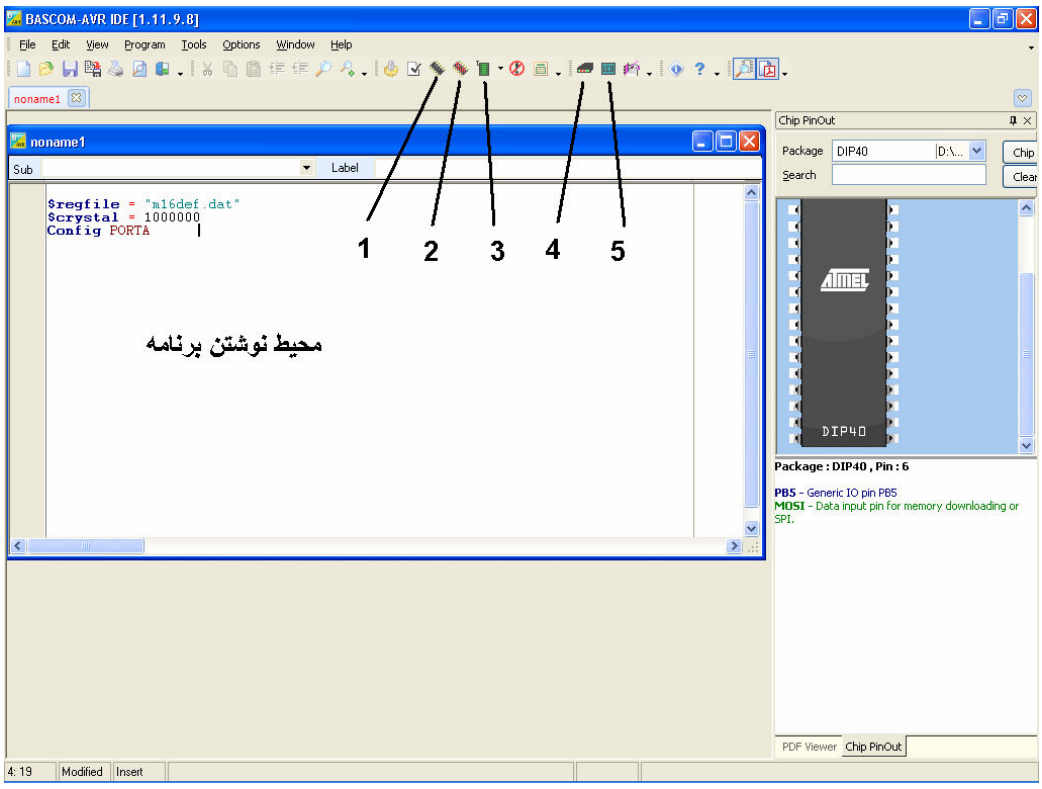
بررسی نرم افزارهای کاربردی در AVR

۴ – ۱ بررسی مختصر نرم افزار Bascom AVR :

از میان انواع مختلف کامپایلرهایی که تا به حال برای میکروکنترلر AVR عرضه شده است ، نرم افزار معروف و قدرتمند Bascom AVR را مورد بررسی قرار می دهیم.

این نرم افزار تمام میکروهای AVR را پشتیبانی کرده و از زبان Basic برای برنامه نویسی استفاده می کند. علاوه بر محیط ساده برنامه نویسی و محیط مخصوص برای پروگرام کردن برنامه ، از قابلیت های بسیار ارزنده آن می توان به داشتن تحلیلگر یا به عبارتی Simulator داخلی اشاره کرد که برای یادگیری برنامه نویسی بسیار کارآمد است. همچنین محیطی به نام Terminal Emulator در این نرم افزار وجود دارد که برای برقراری ارتباط و نمایش داده های ارسالی و دریافتی بین میکروکنترلر و کامپیوتر در ارتباط سریال Rs – 232 استفاده می شود.

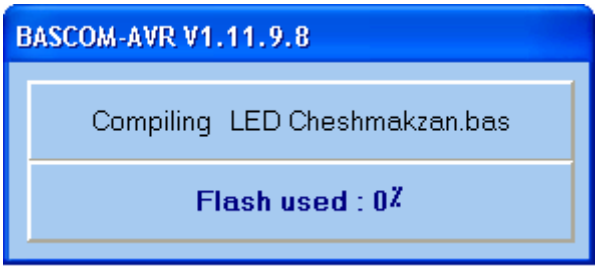
در شکل زیر محیط نرم افزار Bascom AVR را مشاهده می کنید که در ادامه به مهمترین قسمت های آن می پردازیم :



شکل ۱-۴ محیط نرم افزار بیسکام

۱- Compile Program : پس از اتمام برنامه نویسی ، با این گزینه (یا فشردن کلید F7) شما قادر بـه ترجمـه برنامه به زبان ماشین خواهید بود که به اصطلاح کامپایل هم گفته می شود.

البته پیش از کامپایل کردن می بایست برنامه را در مکانی از هارد دیسک کامپیوتر ذخیره کرد.



شکل ۲-۴ مرحله کامپایل شدن برنامه توسط نرم افزار بیسکام

پس از کامپایل کردن فایل هایی با پسوند های مختلف تولید می شود.

XX . Bin فایل باینری که می تواند در میکرو کنترلر Program شود.

XX . Dbg فایل Debug که برای نرم افزار شبیه ساز Bascom مورد نیاز است.

XX . Obj فایل Object که برای نرم افزار Avr Studio مورد نیاز است.

XX . Rpt فایل گزارشی.

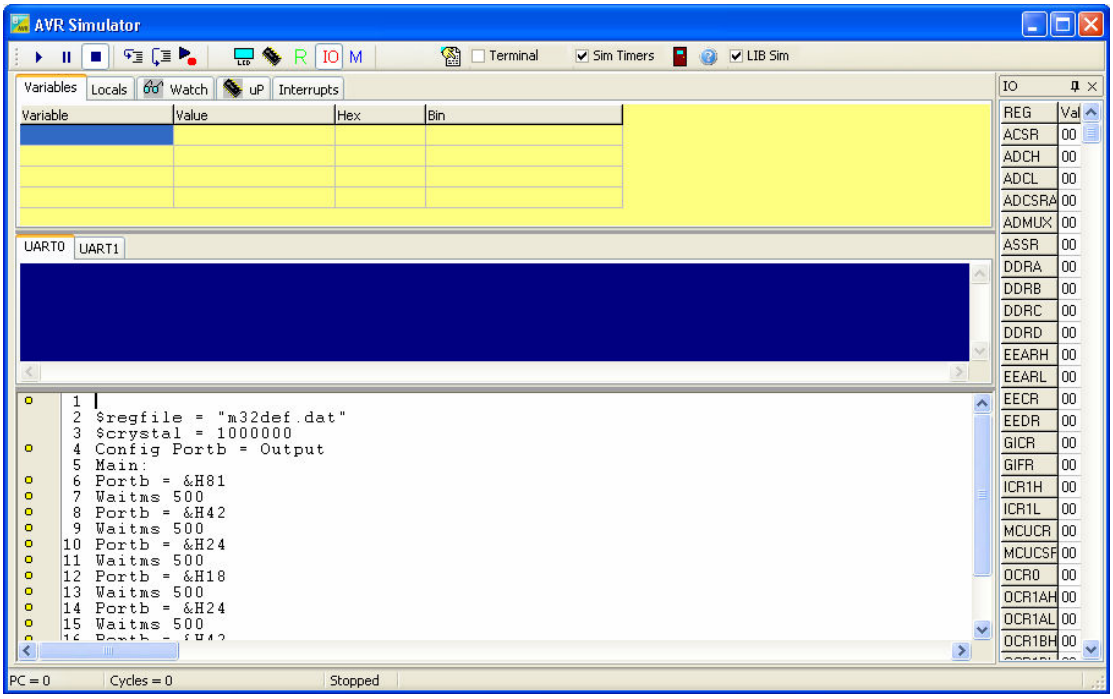
XX . Hex فایل هگزادسیمال اینتل که برای بعضی از انواع پروگرامرها مورد نیاز است.

XX . Err فایل فایل خطا که فقط در هنگام بروز خطا ایجاد می شود.

XX . Eep داده هایی که باید در EEPROM برنامه ریزی شوند در این فایل نگهداری می گردند.

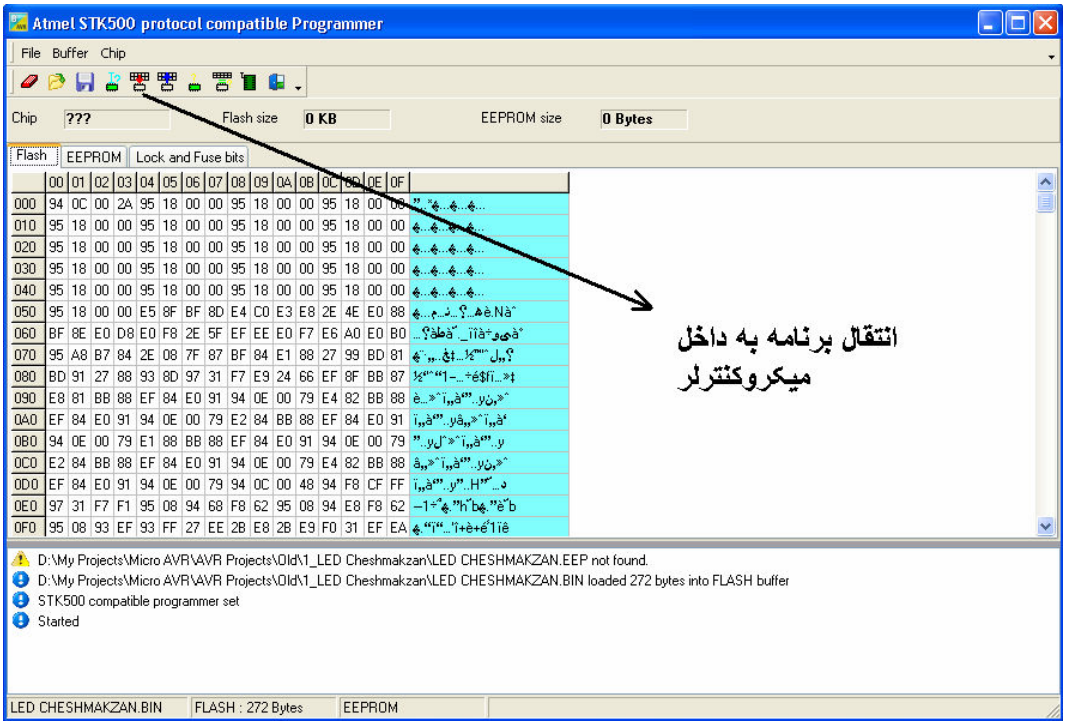
نکته مهم: اگر خطایی در برنامه نویسی موجود باشد ، آن خطا را در یک کادر محاوره ایی در پایین نرم افزار بیسکام دریافت خواهید کرد و عملیات کامپایل متوقف می شود. با دو بار کلیک به روی هر کـدام از آن خطاهـا ، برنامـه بـه خطی که خطا در آن رخ داده پرش می کند که با رنگ قرمز مشخص می شـود و حتمـا مـی بایـست آن را برطـرف نمود.

۲- Simulate Program : با این گزینه (یا فشردن کلید F2) شبیه ساز داخلی نرم افزار بیـسکام فعـال خواهـد شد و می توان برنامه را در این محیط اجرا کرد. همچنین در ایـن محـیط امکـان کـار بـا LCD ، کـی پـد ۴*۴ ، ADC ، خواندن و نوشتن حافظه های EEPROM و SRAM و... وجود دارد.



شکل ۳-۴ محیط شبیه سازی نرم افزار بیسکام

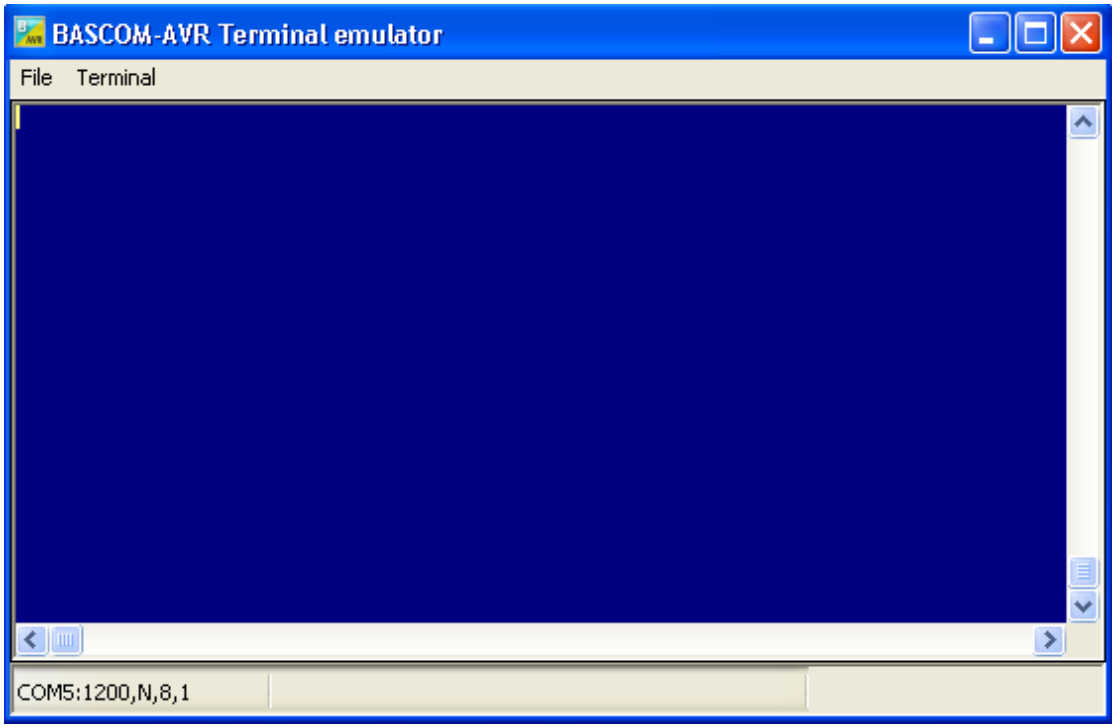
۳- Program Chip : توسط این گزینه (یا فشردن کلید F4) ، محیط برنامه ریزی یا پروگرام ظاهر می شود که می توان از طریق همین محیط ، برنامه مورد نظر را پس از کامپایل کردن و گرفتن فایل خروجی با پسوند Hex. توسط دستگاه پروگرامر به آی سی میکروکنترلر انتقال داد.



شکل ۴-۴ محیط پروگرام نرم افزار بیسکام

نکته: برای پاک کردن برنامه داخل میکروکنترلر ، به منوی Chip رفته و سپس گزینه Erase را انتخاب می کنیم.

۴- Terminal Emulator : از این گزینه می توان برای نمایش داده ارسالی و دریافتی در ارتباط سریال Rs – 232 بین میکروکنترلر و کامپیوتر استفاده کرد. در استفاده از این محیط باید به این نکته توجه داشت که حتما می بایست از نرخ انتقال Baud مشابه در میکرو و کامپیوتر استفاده شود. مثلا اگر از Baud برابر با 9600 استفاده می کنیم باید در گزینه Communication Setting نیز Baud برابر با 9600 را انتخاب کنیم.



شکل ۵-۴ محیط Terminal Emulator

منوهای محیط Terminal Emulator عبارتند از :

File Upload : برنامه جاری در فرمت Hex را Upload می کند.

File Escape : صرف نظر کردن از Upload کردن فایل.

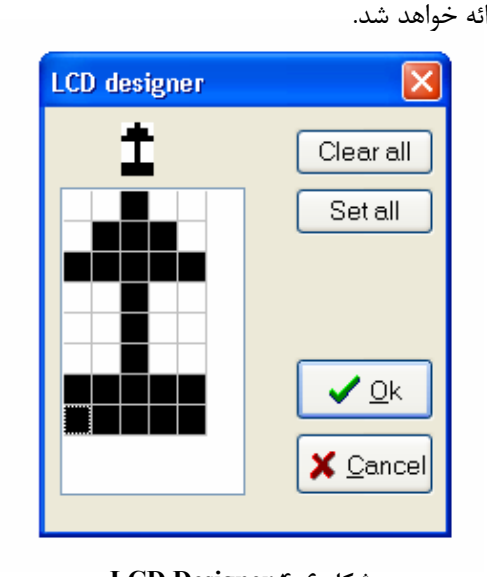
File Exit : خروج از برنامه Terminal Emulator

Terminal Clear : محتوای پنجره Terminal Emulator را پاک می کند.

Terminal Open Log : فایل Log را باز یا بسته می کند. هنگامی که فایل Log وجود نداشته باشد از شما درخواست نامی برای باز کردن فایل گزارش می کند. تمام اطلاعاتی که در پنجره Terminal Emulator پرینت می شود داخل فایل Log ثبت می شود.

Setting : توسط این منو می توانید تنظیمات پورت Com و دیگر Option ها را انجام دهید.

۵- LCD Designer : با رفتن به این گزینه ، پنجره ایی به شکل زیر باز می شود. که برای طراحی اشکال دلخواه یا درست کردن کاراکترهای فارسی و .. می باشد. با کلیک کردن روی هر کدام از خانه ها می توان طرحی را ایجاد کرد و با فشردن دکمه OK در صفحه برنامه نویسی کدهای مختلف آن را ایجاد می کند. در قسمت های بعدی توضیحات بیشتر در این رابطه ارائه خواهد شد.



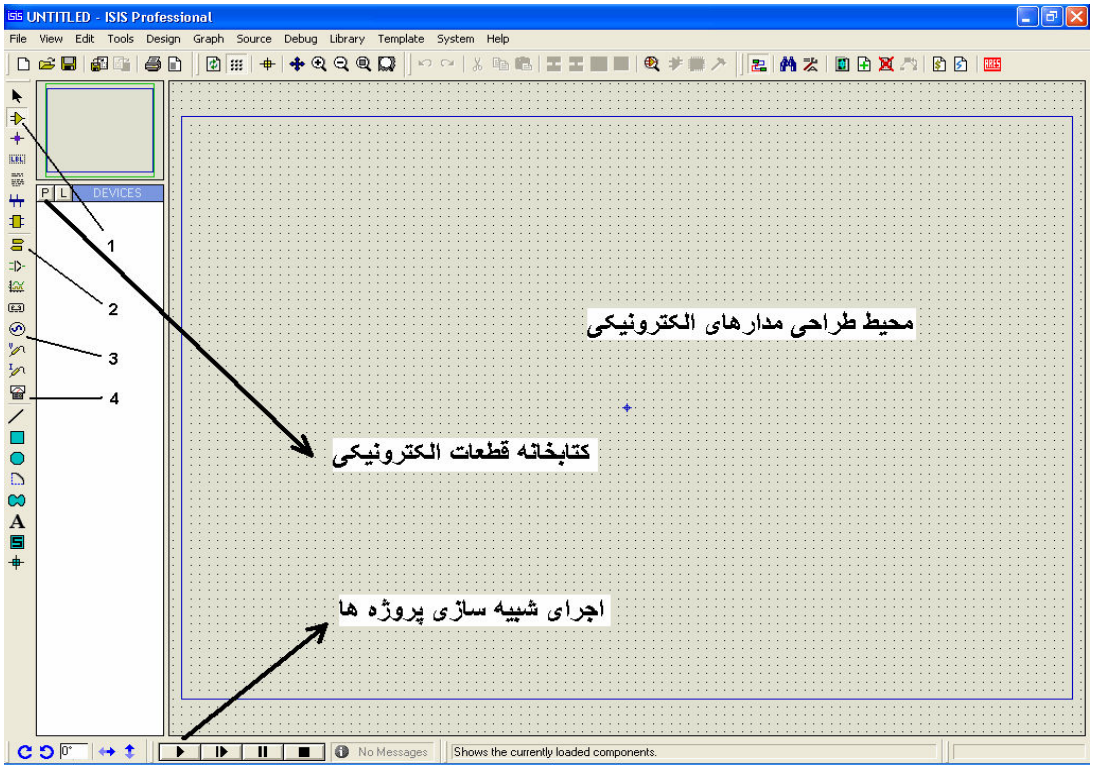
شکل ۴-۶ LCD Designer

در فصل های بعدی به بررسی دستورات کاربردی و طریقه پیکره بندی امکانات میکروکنترلر AVR توسط نرم افزار بیسکام می پردازیم.

۴ – ۲ بررسی مختصر نرم افزار Proteus :

نرم افزار Proteus یکی از قوی ترین نرم افزارها در بحث شبیه سازی مدارات الکترونیکی بوده و دارای قابلیت های زیادی می باشد. این نرم افزار به دو بخش ISIS و ARES تقسیم می شود. از محیط ISIS برای طراحی و تست مدار (شبیه سازی) استفاده می شود. از محیط ARES برای تهیه نقشه PCB مدارات الکترونیکی استفاده می شود. از خصوصیات مهم و قابل توجه نرم افزار پروتئوس می توان به قابلیت شبیه سازی و پروگرام کردن مجازی انواع میکروکنترلرها اشاره کرد که به صورت Runtime این کار را انجام می دهد.

در شکل زیر محیط نرم افزار Proteus را مشاهده می کنید که در ادامه به مهمترین قسمت های آن می پردازیم :



شکل ۷-۴ محیط نرم افزار پروتئوس

۱- Component Mode : لیستی از کلیه قطعات الکترونیکی انتخاب شده را در اختیارمان قرار می دهد که برای طراحی می بایست هر کدام از آنها را با موس به محل مورد نظر کشیده شود.

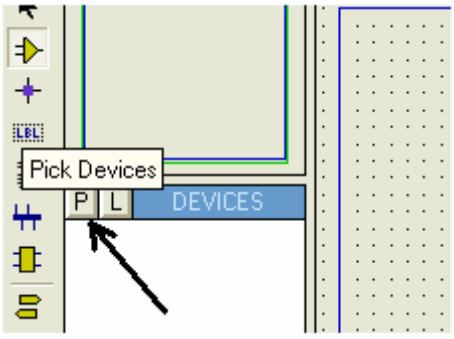
۲- Terminals Mode : برای ایجاد انواع خروجی ها و نیز VCC و GND به کار می رود و با ایـن ابزارهـا مـی توانیم به کاهش حجم سیم کشی برای جلوگیری از شلوغی مدار استفاده کنیم.

۳- Generator Mode : برای ایجاد انواع منابع تولید سیگنال از قبیل موج سینوسی ، پالس های مربعی و ... می باشد.

۴- Virtual Instruments Mode : برای دسترسی به دستگاه های اندازه گیری نظیر اوسیلوسکوپ ، مولتی متر ، وات متر و ... می باشد.

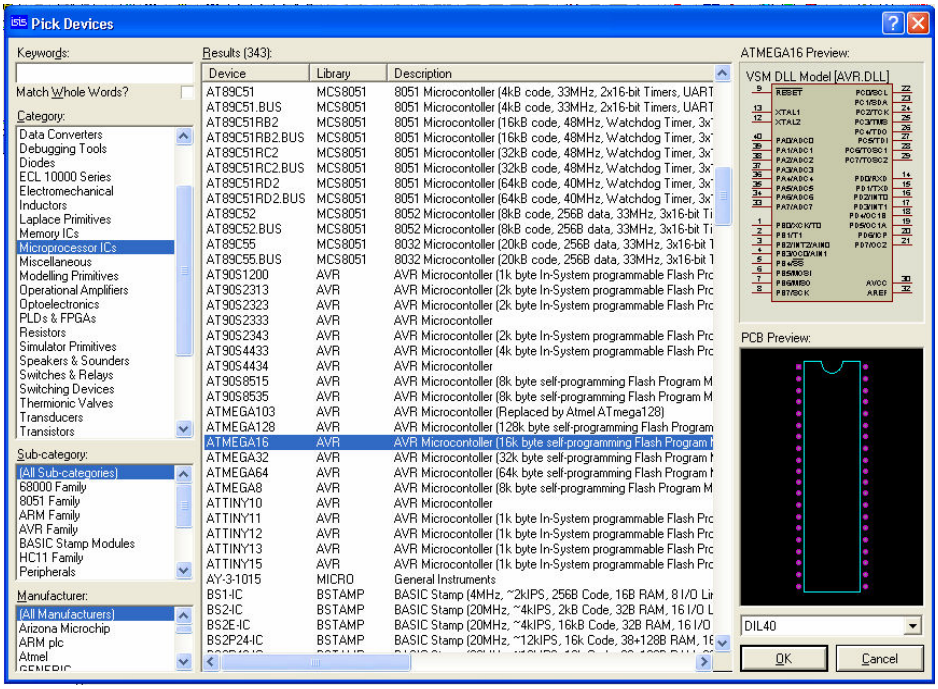
برای رسم یک مدار الکترونیکی به ترتیب زیر عمل می کنیم:

۱- ابتدا با کلیک کردن روی Pick Devices که در شکل زیر مشخص شده به کتابخانه قطعات الکترونیکـی دسترسی پیدا می کنیم.



شکل ۸-۴ دسترسی به کتابخانه قطعات الکترونیکی

پنجره زیر باز می شود:



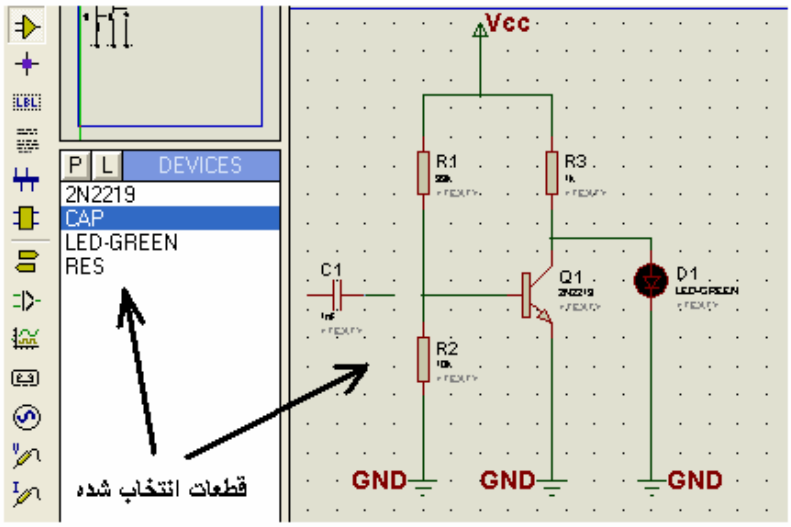
شکل ۹-۴ لیست قطعات الکترونیکی

همانطور که مشاهده می شود ، در ستون سمت چپ می توانیم نوع قطعات را از قبیل مقاومت ، خازن ، ترانزیستور ، انواع آی سی ها ، میکروکنترلر و ... مشخص کنیم. و سپس در قسمت وسط این پنجره به شماره مورد نظر همان قطعه دسترسی پیدا کنیم ، آنگاه روی OK کلیک کرده تا قطعه مورد نظر به لیست قطعات مورد استفاده نرم افزار افزوده شود. در اینجا بعنوان نمونه میکرکنترلر AVR به شماره ATMEGA16 مشخص شده است.

نکته: یک راه ساده تر برای یافتن قطعه مورد نظر این است که در پنجره بالا و در قسمت Keywords فقط نام یا شماره قطعه را تایپ کنیم ، در صورت وجود پیدا می شود.

۲- پس از انتخاب همه قطعاتی که برای طراحی یک مدار نیاز داریم حالا می بایست آن ها را بصورت تک به تک به محل طراحی نرم افزار وارد کنیم که این کار با انتخاب هر قطعه به وسیله ماوس و سپس کلیک چپ در صفحه محیط طراحی صورت می گیرد. برای چرخاندن و یا تغییرات در قطعه مثل تغییر نام و شماره و ... کافیست روی هر کدام از آن ها راست کلیک کنیم و به منوی مربوطه وارد شویم. برای حذف هر قطعه یا مسیر سیم کشی بین دو قطعه ، کافی ست روی آن دو بار کلیک راست کنیم.

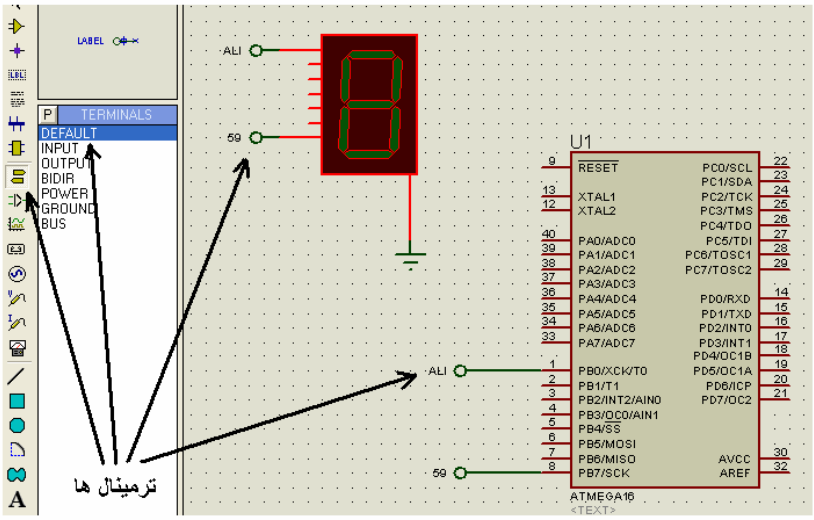
۳- پس چیدن قطعات در جای مناسب و اعمال تغییرات روی آن ها می بایست برای برقراری ارتباط بین آن ها ، سیم کشی کنیم. برای سیم کشی کافیست نشانگر ماوس را به پایه هر قطعه نزدیک کرده و سپس با کشیدن ماوس به سمت پایه ایی از قطعه دیگر ، ارتباط بین دو قطعه برقرار می شود.



شکل ۱۰-۴ نحوه رسم مدار در محیط نرم افزار پروتئوس

نکته: گاهی اوقات در مدارات شلوغ ، برای کاهش حجم سیم کشی می توان از ترمینال ها استفاده کرد که دیگر نیازی به سیم کشی نیست. و برای این کارکافیست نام یا شماره ترمینال های متصل به پایه قطعاتی که قرار است با هم در ارتباط باشند دقیقا یکی باشد.(هم نام یا هم شماره). برای نامگذاری روی هر یک از ترمینال ها ، می بایست روی هر یک از آن ها دوبار کلیک چپ کرده و در پنجره ایی که باز می شود ، در کادر مقابل عبارت String نام یا شماره دلخواه خود را تایپ کنیم.

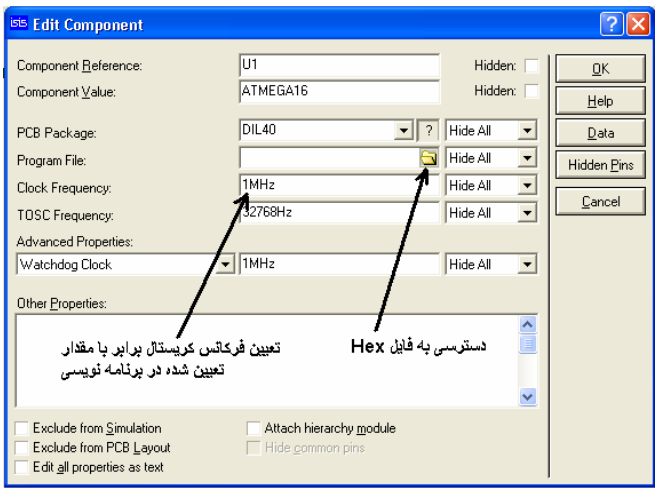
در شکل زیر پایه هایی از میکرو و سون سگمنت که به ترمینال هایی با نام دلخواه ALI متصل هستند به هم ارتباط دارند و پایه هایی که به ترمینال شماره دلخواه 59 متصل هستند نیز همینطور است. بقیه پایه ها به هم راه ندارند و ارتباطی بین آنها برقرار نیست.



شکل ۱۱-۴ ایجاد ترمینال بین قطعات

طریقه پروگرام کردن مجازی میکروکنترلرها در نرم افزار پروتئوس:

برای شبیه سازی مدارهای میکروکنترلری می بایست در ابتدا فایل Hex را در نرم افزار پروتئوس به میکروکنترلر معرفی نماییم و برای این کار روی میکروکنترلری که به صفحه طراحی پروتئوس وارد کرده ایم دو بار کلیک چپ می کنیم ، پنجره زیر باز می شود ، برای دسترسی به فایل هگزی که توسط کامپایلر ایجاد کرده ایم روی قسمت نشان داده شده در شکل بالا کلیک کرده و فایل ذخیره شده را انتخاب می کنیم. و نیز مقدار کریستال تعیین شده در برنامه را در قسمت مشخص شده وارد می کنیم (به مگاهرتز) ، سپس روی دکمه OK کلیک می کنیم.



شکل ۱۲-۴ پروگرام کردن مجازی میکروکنترلر در پروتئوس

فصل ۵

دستورات کاربردی در نرم افزار Bascom – AVR

نرم افزار بیسکام شامل بیش از ۵۰۰ دستورالعمل برای برنامه نویسی بوده که در اینجا پرکاربردترین دستوراتی که در مثال ها و پروژه های این مجموعه استفاده شده اند گردآوری شده است.

به این نکته هم توجه شود که به هیچ عنوان نیازی به حفظ کردن این دستورات نداریم ، بلکه با رجوع به همین قسمت ، یا سایر کتاب های مرجع مربوط به AVR و یا استفاده از Help خود نرم افزار بیسکام می توان نیازهای خود را بر طرف نمود.

۱- دستور Set) یک بیت را یک (روشن) می کند.

```
Set PortB.1
```

۲- دستور Reset) یک بیت را صفر(خاموش) می کند.

```
Reset PortB.2
```

۳- دستور Toggle) مقدار منطقی یک پایه یا یک بیت را معکوس می کند.

```
Toggle PortA.6
```

۴- دستور Bitwait) باعث توقف اجرای برنامه در همان خط می شود تا زمانی که بیت مورد نظر Set یا Reset شود.

```
Bitwait PortD.3 , Reset
```

۵- دستور Alias) برای تغییر نام متغییر استفاده می شود.

```
LED Alias PortC.1
```

۶- دستور Swap) محتوای دو متغییر مختلف جا به جا می شود.

Swap A , B

۷- دستور Incr) یک واحد به متغییر عددی Var می افزاید.

Incr A

۸- دستور Decr) یک واحد از متغییر عددی Var کم می کند.

Decr B

۹- دستور Rotate) تمام بیت ها را به سمت چپ یا راست منتقل می کند.

Rotate B , Left , 2

۱۰- Lookup) توسط این جدول می توان مقدار داخواهی (عدد) را از جدولی برگرداند.

B = Lookup (A , Dta)

۱۱- Lookupstr) توسط این جدول می توان رشته دلخواهی (مثل حروف) را از جدولی برگرداند.

B = lookupstr (A , Dta)

۱۲- دستور Len) طول یا به عبارتی تعداد کاراکترهای یک رشته را بر می گرداند.

A = "Book"

B = Len (A)

(عدد ۴ را که با تعداد کاراکترهای کلمه Book است ، در متغییر B ریخته می شود)

۱۳- دستور Mid) قسمتی از یک رشته را با قسمتی از رشته دیگر عوض می کند.

B = Mid (A , 2 , 3)

(کاراکتر دوم به طول سه کاراکتر از متغییر A را در B می ریزد)

۱۴- دستور Mod) باقیمانده تقسیم A/B را در متغییر C قرار میگیرد.

C = A Mod B

۱۵- دستور Str) متغییر عددی B را به رشته A تبدیل می کند.

A = Str (B)

۱۶- دستور Val) رشته A را به متغییر عددی B تبدیل می کند.(این دستور درست عکس Str عمل می کند)

B = Val (A)

۱۷- دستور Delay) برای مدت کوتاهی در اجرای برنامه تاخیر ایجاد می کند.(یک ثانیه)

۱۸- دستور Wait) برای ایجاد تاخیر به مقدار دلخواه در برنامه می باشد.

Wait = (تاخیر بر حسب ثانیه)

Waitms = (تاخیر بر حسب میلی ثانیه)

Waitus = (تاخیر بر حسب میکرو ثانیه)

۱۹- دستور Goto و Jmp) برای پرش به لیبل مورد نظر می باشد.

۲۰- دستور Do _ Loop) برای ایجاد یک حلقه بی نهایت تکرار می باشد.

البته با دستور Until که در کنار Loop درج می شود ، می توان حلقه را از حالت بی نهایت به حالت تعداد تکرار مورد نظر تبدیل کرد.

Do

Loop Until B = 26

(توقف تکرار حلقه در عدد ۲۶)

۲۱- دستور For _ Next) برای ایجاد یک حلقه با تعداد تکرار و پله تعیین شده می باشد.

For A = 0 To 10 Step 2

(متغییر A از عدد صفر تا عدد ۱۰ شروع به شمردن می کند ولی با ۲ پله یعنی فقط اعداد زوج را در نظر می گیرد)

۲۲- دستور While _ Wend) برای ایجاد حلقه شرطی به کار می رود و در صورتی که شرط برقرار شـود برنامـه داخل حلقه While افتاده و دیگر از آن خارج نمی شود.

```
While  A = 10

 Wend
```

۲۳- دستور If) دستوری شرطی که در صورت برقرار شدن شرط کار خاصی را انجام می دهد یا به لیبل مورد نظر پرش پیدا می کند.

```
If  A = 6  Then
Set PortB.1
 Else
Reset PortB.1
End if
```

۲۴- Select _ Case) برای بررسی کردن چندین شرط روی یک مورد از این دستور استفاده می کنیم.

```
Select Case A
Case is = 1
Set PortD.1
Case is = 2
Set PortD.2
End Select
```

۲۵- دستور Exit) برای خارج شدن از کلیه حلقه استفاده می شود.

```
Exit While
```

۲۶- دستور Declare Sub) برای معرفی زیر برنامه ای (لیبل) که قصد فراخوانی آن را داریم (Call) استفاده می شود.

```
Declare Sub Book
```

۲۷- دستور Call) برای فراخوانی زیر برنامه (لیبل) استفاده می شود.

۲۸- دستور Gosub) این دستور برای پرش به زیر برنامه (لیبل) مورد نظر و اجرای برنامـه را از آدرس آن لیبـل انجام می دهد و سپس به برنامه اصلی باز می گردد.

در آخر زیر برنامه (لیبل) حتما باید برای بازگشت به خط بعد Gosub از دستور Return استفاده شود.

Gosub Book

۲۹- دستور Return) برای بازگشت از یک زیربرنامه از این دستور استفاده می شود.

۳۰- دستور Const) برای تعریف یک عدد ثابت از این دستور استفاده می شود.

Constsymbol = Numconst / String / Expression

۳۱- دستور Debounce) برای اتصال میکروسوئیچ به یکی از پین های میکروکنترلر می باشد.

Debounce PinC.2 , 1 , Book

به این نکته توجه شود ، زمانی که از دستور Debounce استفاده می کنیم ، حتمـا مـی بایـست Pull Up (بـالا کش) میکرو فعال شود. Pull Up در واقع یک مقاومت (4.7 K) بوده که بین VCC و پین مربوطه قرار گرفتـه و با برقراری جریان بسیار کم ، باعث پایداری سیستم میکروکنترلر می شود و از وجود نویزهـای ناخواسـته جلـوگیری می کند. لازم به ذکر است که در این حالت فقط در پایه حساس به یک به کار می رود.

DDR B = 0 : Port B = 255

۳۲- دستور Pulseout) برای ایجاد پالس بر روی پایه ایی دلخواه می باشد.

Pulseout Port A , 1 , 1000

(ایجاد پالسی به فرکانس ۱۰۰۰ هرتز بر روی پین A.1)

۳۳- دستور Pulsein) مدت زمان بین تغییر وضعیت پایه دلخواه را از منطق یک به صفر یـا بلعکـس آشـکار مـی کند.(برای اندازه گیری پالس ورودی)

Pulsein A , PinB , 1 , 0

۳۴- دستور Sound) برای ایجاد پالس های صوتی در پین دلخواه است.

Sound PortB.2 , 100 , 10

۳۵- دستور Dtmfout) تولید سیگنال های DTMF برای عمل شماره گیری تلفن در حالت تن می باشد.

Dtmfout A , 100

(در این حال عدد موجود در متغییر A که می بایست از نوع رشته ایی یا String باشد با تاخیر ۱۰۰ میلی ثانیه به خط تلفن ارسال می شود تا عمل شماره گیری انجام شود.)

۳۶- دستور Day Of Week) این دستور روزهای هفته را نشان می دهد.

Target = Day Of Week () / (Bday Month Year) / (Str Data)

Target متغییری است که روز هفته به آن ارجاع داده می شود.

Bday Month Year متغییری است که تاریخ ، روز ، ماه و سال در آن قرار می گیرد.

Str Data رشته ای است که مقدار رشته ای تاریخ را طبق فرمت تعیین شده در دستور Config Date نگه می دارد.

۳۷- دستور Encoder) این دستور برای خواندن پالس های خروجی یک روتور انکدر (شفت انکدر) می باشد.

Var = Encoder (Pin1 , Pin2 , Left Lable , Right Lable , Wait)

Var متغییر مقدار نتیجه شمارش پالس های ورودی است.

Pin1 , Pin2 نام پایه هایی است که قرار است از میکرو AVR به خروجی شفت انکدر متصل شوند. (هر دو پایه حتما می بایست به یک پورت متصل شوند)

Left Lable نام برچسبی ست که هر وقت شفت به سمت چپ گردش کرد ، برنامه به آن لیبل پرش می کند.

Right Lable نام برچسبی ست که هر وقت شفت به سمت راست گردش کرد ، برنامه به آن لیبل پرش می کند.

Wait ، اگر این مقدار 0 انتخاب شود ، فقط چرخش / فاز آن را بررسی می کند و اگر 1 انتخاب شود ، تا زمانی که شفت بچرخد ، مکث می کند.

استفاده از مقدار 1 باعث توقف برنامه می شود.

۳۸- دستور Sonysend) برای ارسال کدهای مادون قرمز در ریموت کنترل می باشد.(مارک سونی)

Sonysend addres [,bits]

و نیز می توان با دستور زیر دیتاهای مادون قرمز اکثر دستگاه های صوتی و تصویری را تولید کرد.

Rc6send Togglebit , Address , Command

۳۹- دستورات ریاضی و منطقی)

معرفی	نماد
CONJUNCTION	AND
DISJUNCTION	OR
EXCLUSIVE OR	XOR
COMPLIMENT	NOT

علامت	نماد
علامت ضرب	*
علامت جمع	+
علامت تفریق	–
علامت ممیز	.
علامت تقسیم	/
علامت کوچکتر از	<
علامت تساوی	=
علامت بزرگتر از	>
علامت بتوان	^
علامت کوچکتر یا مساوی	=>
علامت بزرگتر یا مساوی	<=
علامت مخالف	<>

جدول ۱-۵ دستورات ریاضی و منطقی

دستور Sqr) جذر متغییر Source را حساب کرده و در متغییر Var قرار می دهد.

Var = Sqr (Source)

دستور Rnd) یک عدد تصادفی را در بازه ای که تعریف می کنیم ، بدست می آورد.

Var = Rnd (Limit)

Limit عدد ثابت یا متغییر است که تعیین می کند از صفر تا چه عددی ، عدد تصادفی را انتخاب کند.

فصل ۶

کار با انواع پیکره بندی ها

برای استفاده از تمامی امکانات میکروکنترلر AVR (و یا هر میکروکنترلـر دیگـری) ، مـی بایـست در ابتـدای هـر برنامه آن را آماده سازی یا پیکره بندی (Config) نماییم. که در این قسمت به پرکاربرد ترین آن ها می پردازیم.

۱ – ۶ پیکره بندی پورت ها:

برای تعیین ورودی یا خروجی بودن هر یک از پایه های میکروکنترلر استفاده می شود.

```
Config Port A = Input

Config Port B = Output
```

البته می توان برای نشان دادن ورودی و خروجی به ترتیب از عدد صفر و یک نیز استفاده کرد.
بعنوان مثال در پیکره بندی حالت زیر پین های D.0 تا D3 بعنوان خروجی و پین های D4 تا D7 بعنـوان ورودی تعریف شده اند.

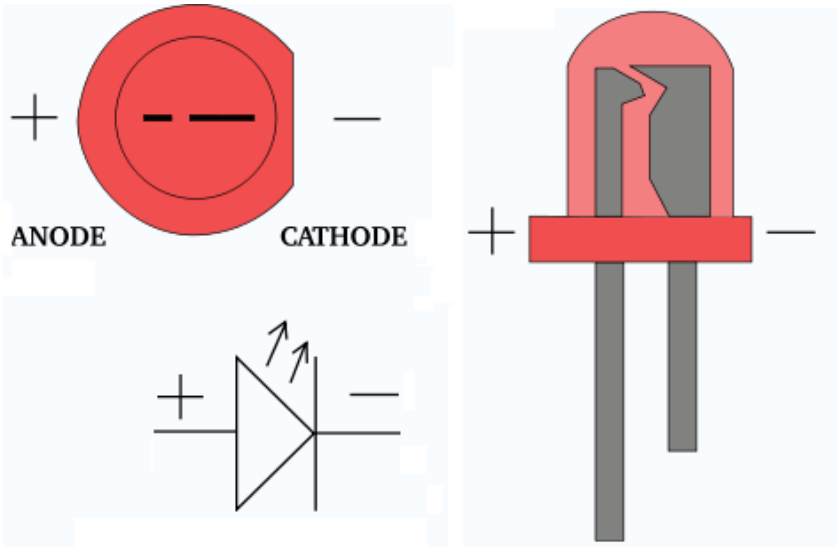
```
Config PortD = &B00001111
```

با پیکره بندی پورت ها می توان انواع LED ها و 7Segment هـا را بعنـوان خروجـی و انـواع میکروسـوئچ هـا و سنسورها را بعنوان ورودی به کار برد.

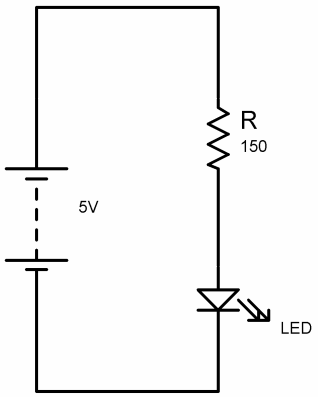
الف) راه اندازی LED)

یک LED در واقع دیودی است که با عبور جریان از خود نور ساطع می کند. در این حالت ، اخـتلاف پتانـسیلی در حدود ۰,۷ ولت بین دو سر آند و کاتد آن مشاهده می گردد.

برای روشن شدن یک LED ، نیاز به ولتاژی در حدود ۲ ولت با جریان ۲۰ میلی آمپر داریم.



شکل ۱-۶ دیود نورانی



شکل ۲-۶ مدار راه اندازی دیود نورانی

رابطه محاسبه مقدار مقاومت مناسب برای راه اندازی یک LED :

$$R = (VCC - V_{LED}) / I_{LED}$$

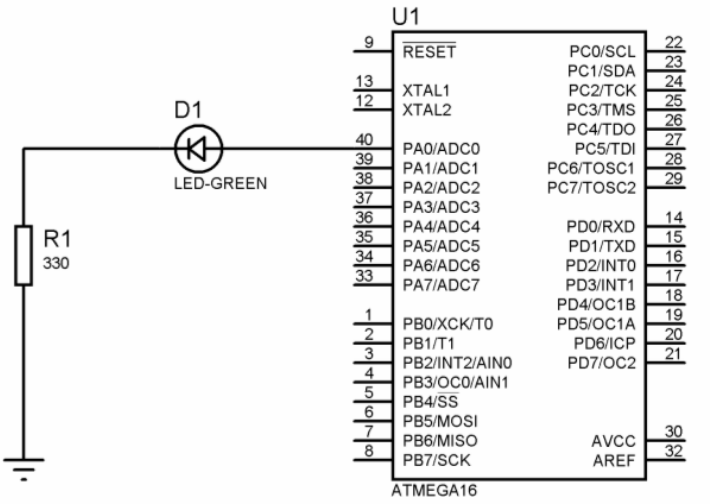
$$(5 - 2) / 20\ m = 150\ R$$

آزمایش ۱) چشمکزن ساده (A) :

دستورات جدید) **حلقه** Do – Loop ، Config Port **بعنوان خروجی** ، Set ، Reset ، Wait

هدف) در این آزمایش به وسیله برنامه نویسی در محیط نرم افزار بیسکام و انجام شبیه سازی مدار آن در نرم افـزار پروتئوس ، مشاهده می شود که دیود نورانی (LED) بطور متناوب روشن و خاموش می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Output
Do
Set Porta.0
Wait 1
Reset Porta.0
Wait 1
Loop
End
```



شرح برنامه) در خط اول برنامه به معرفی نوع میکروکنترلر AVR پرداخته ایم که از سری Mega16 اسـت. و در همه پروژه های این مجموعـه از آن اسـتفاده شـده اسـت. در خـط دوم برنامـه ، مقـدار کریـستال (کـلاک سـاعت) میکروکنترلر تعیین شده است که ما در همه پروژه های این مجموعه مقدار 1MHz را در نظر گرفتـه ایـم. بـه ایـن نکته توجه شود که در تمامی پروژه های میکروکنترلر AVR نوشتن این دو خط ، در اول برنامه الزامیست. در خـط سوم پورت A (پین های A0 تا A7) میکروکنترلر را خروجی تعریف کرده ایم. در خط چهارم دسـتور Do وجـود دارد که شروع بدنه اصلی برنامه بوده و در واقع با این دستور به همراه دستور Loop یـک حلقـه بـی نهایـت تکـرار ایجاد می کنیم. در خط پنجم پایه شماره صفر پورت A را روشن می کنیم که به دنبال آن LED مدار هـم روشـن می شود. در خط ششم یک ثانیه توقف برنامه را داریم یعنی LED یک ثانیه روشن می ماند. در خـط هفـتم همـان پایه را خاموش می کنیم که به دنبال آن LED مدار هم خاموش می شود. در خط هشتم دوبـاره یـک ثانیـه توقـف برنامه را داریم که باعث می شود LED به مدت یک ثانیه خاموش باقی بماند. در خـط نهـم برنامـه بـا رسـیدن بـه دستور Loop ایجاد یک حلقه بی نهایت تکرار را می دهد یعنی دوباره به دستور Do پـرش کـرده و همـان اعمـال قبلی تکرار می شود و در نتیجه شاهد چشمک زدن دیود نورانی بطور متناوب هستیم. در خط دهم نیز دستور پایان برنامه را به میکرو می دهیم ولی چون حلقه بی نهایت داریم ، روند اجرای برنامه هیچ وقت به پایان نمی رسـد مگـر

اینکه به واسطه ریست کردن میکروکنترلر بتوان این کار را انجام داد. فایل های برنامه نویسی شده و شبیه سازی این آزمایش و همینطور سایر آزمایش ها در فولدر Projects ، قسمت ۱ ، قرار داده شده است.

آزمایش ۲) چشمکزن ساده (B) :

دستورات جدید) Toggle

هدف) عملکرد این آزمایش دقیقا شبیه آزمایش قبلی ست ولی دستورات کمی متفاوت است.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Output
Do
Toggle Porta.0
Wait 1
Loop
End
```

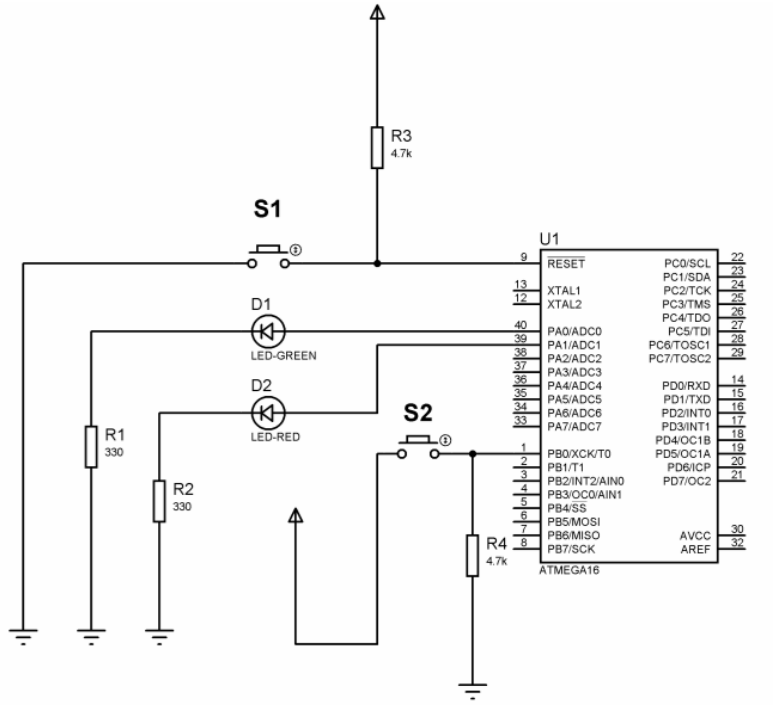
شرح برنامه) خط اول تا چهارم این برنامه دقیقا شبیه برنامه بالا می باشد ، ولی در خط پنجم دستور Toggle را داریم که حالت پین A.0 را معکوس می کند یعنی اگر خاموش بود آن را روشن و اگر روشن بود آن را خاموش می کند که تداوم این کار موجب چشمک زدن دیود نورانی می شود (یک ثانیه روشن و یک ثانیه خاموش) و نیز با توجه به وجود حلقه Do و Loop در برنامه این روند تا بی نهایت ادامه می یابد.

آزمایش ۳) چشمکزن دو لامپی با کلید راه انداز :

دستورات جدید) Alias ، **شرط** IF ، Config Port **بعنوان ورودی** ، Jmp ، Goto ، **ایجاد زیربرنامه (یا لیبل)**

هدف) در این آزمایش با فشردن کلید S2 ، دیودهای نورانی به صورت یکی در میان روشن و خاموش می شوند و زمانی که کلید S1 (کلید ریست) فشرده شود این عمل متوقف می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Output
Config Portb = Input
Main1 Alias Porta.0
Main2 Alias Porta.1
Key:
If Pinb.0 = 1 Then Jmp Main
Goto Key
Main:
Set Main1
Reset Main2
Wait 1
Reset Main1
Set Main2
Wait 1
Goto Main
End
```

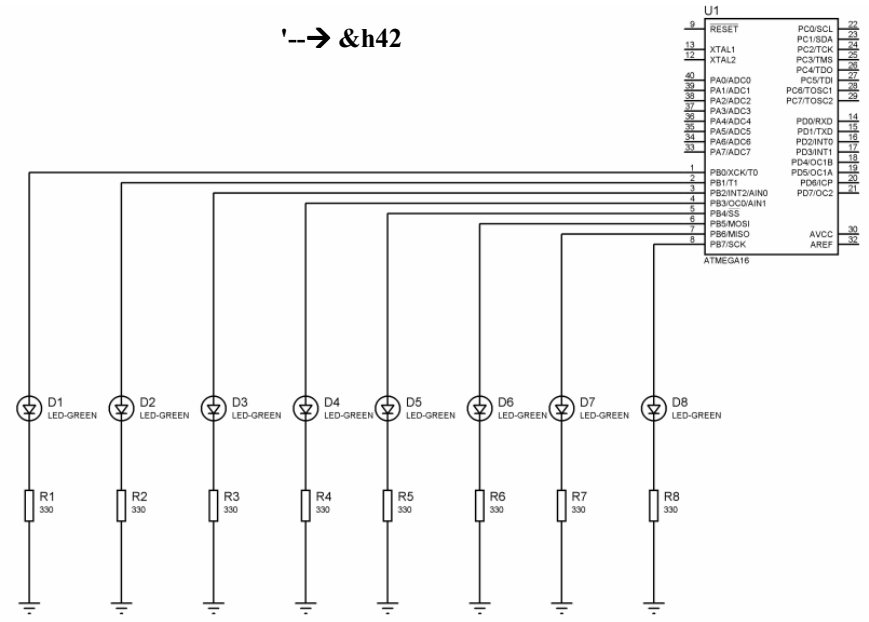


شرح برنامه) در خط سوم این برنامه پورت A را برای روشن کردن LED ها ، خروجی تعریف کردیم و در خط چهارم پورت B را ورودی تعریف کردیم برای اینکه فرمان کلید S2 را به میکرو ارسال کنیم. در خط پنجم و ششم از یک کلمه دلخواه (مثلا Main) به جای استفاده از Porta.0 و Porta.1 بکار برده ایم. مثلا هر جا که لازم بود به Porta.1 فرمان بدهیم ، نام دلخواه Main2 را استفاده می کنیم. لازم به ذکر است که این کار اختیاری ست و برای تسهیل در امر برنامه نویسی به کار برده می شود. در خط هفتم یک زیربرنامه یا لیبل به کار برده ایم که می تواند هر نام دلخواهی (به غیر از دستورات بیسکام) با علامت " : " در جلوی آن باشد. در ادامه لیبل هر برنامه ایی که نوشته می شود به عنوان زیر برنامه محسوب شده و هر عملیات خاصی را می توان در آنجا تعریف کرد. در خط هشتم که به عنوان زیربرنامه Key نیز محسوب می شود از دستور شرطی If استفاده شده است. یعنی اگر کلید S2 که به پین B0 متصل است یک شود یا به اصطلاح به منبع ۵ ولتی وصل شود برنامه با دستور Jmp به زیربرنامه Main پرش می کند ولی اگر کلید S2 فشرده نشود با توجه به دستور خط نهم (Goto Key) به لیبل Key پرش کرده و یک حلقه بینهایت ایجاد می شود. در خط دهم یا زیر برنامه Main یک حالت روشن و خاموش برای LED ها تعریف کرده ایم که شبیه برنامه های قبلی می باشد.

آزمایش ۴) چشمکزن راه رونده (رقص نور) :

هدف) در این آزمایش دیودهای نورانی بصورت ترتیبی روشن و خاموش می شوند.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Portb = Output
Do
Portb = &B10000001              '--→ &h81
Waitms 500
Portb = &B01000010              '--→ &h42
Waitms 500
Portb = &B00100100              '--→ &h24
Waitms 500
Portb = &B00011000              '--→ &h18
Waitms 500
Portb = &B00100100              '--→ &h24
Waitms 500
Portb = &B01000010              '--→ &h42
Waitms 500
Loop
End
```



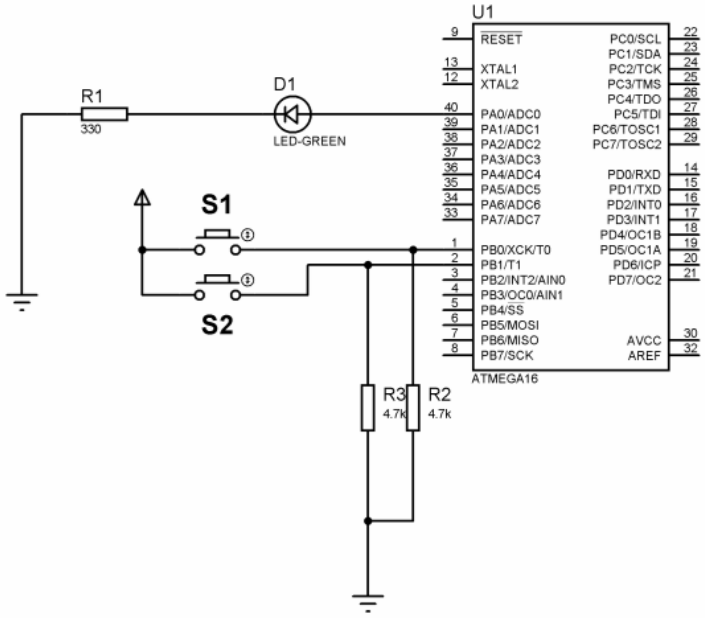
شرح برنامه) با توجه به مقدار دهی باینری که در جلوی هر یک از عبارات Portb در برنامه قرار داده شده این کار صورت می گیرد و مدل رقص نور بوجود می آید. وجود عدد 1 یعنی LED مربوط به آن روشن می شود و 0 به منزله خاموش بودن LED هاست. البته به جای مقدار باینری از مقدار هگزادسیمال هم می توان استفاده کرد که در جلوی هر خط از برنامه آورده شده است.

آزمایش ۵) کنترل یک لامپ به وسیله دو کلید:

دستورات جدید) Bitwait

هدف) در این آزمایش با استفاده از دو کلید (یا میکرو سوئیچ) در دو مکان متفاوت می توان روشن و خاموش کردن هر وسیله ایی (مانند دیود نورانی در این آزمایش) را کنترل کرد. (عملکردی شبیه به کلید دوپل)

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Output
Config Portb = Input
Do
If Pinb.0 = 1 Then
Toggle Porta.0
Bitwait Pinb.0 , Reset
End If
If Pinb.1 = 1 Then
Toggle Porta.0
Bitwait Pinb.1 , Reset
End If
Loop
End
```



شرح برنامه) در این برنامه برای هر یک از کلیدها ، دستور Toggle یا حالت معکوس را تعریف کرده ایم که به خروجی میکرو اعمال می شود. در خط هشتم و دوازدهم از دستور Bitwait استفاده شده که باعث توقف اجرای برنامه در همان خط می شود تا زمانی که بیت مورد نظر Set یا Reset شود.

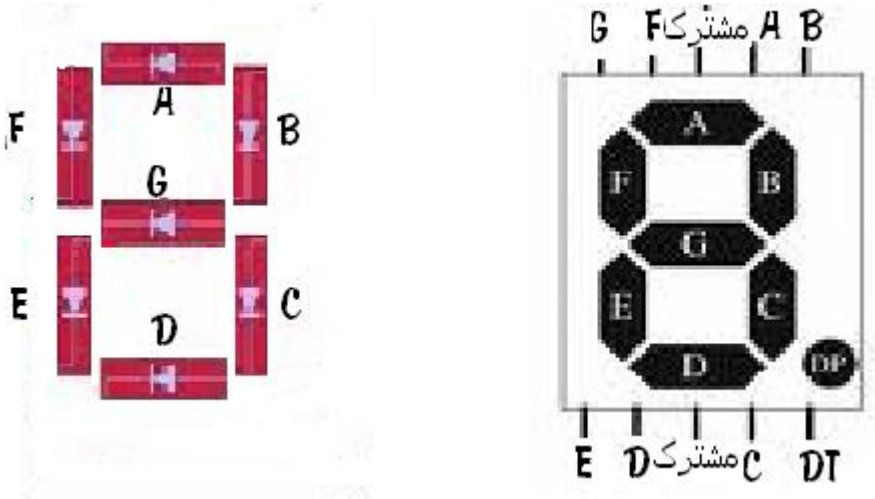
ب) راه اندازی 7Segment)

این قطعه از قطعات کاربردی در نمایشگرها (بخصوص نمایش اعداد) بوده و از ترکیب ۱ + ۷ عدد LED تشکیل شده است که یکی از آنها بعنوان ممیز به کار می رود. هر سون سگمنت دارای ۱۰ پایه می باشد.

سون سگمنت ها به دو دسته کاتد مشترک و آند مشترک تقسیم می شوند. در نوع کاتد مشترک پایه کاتد همه LED ها به هم وصل بوده که در نهایت به قطب منفی تغذیه متصل می شود ولی در آند مشترک برعکس این حالت است.

شکل ۳-۶ سون سگمنت

ترتیب اتصال پایه های هر سون سگمنت بصورت زیر است:



شکل ۴-۶ ترتیب LED های سون سگمنت

توجه) ما در این مجموعه ، 7Seg نوع کاتد مشترک را مورد بررسی قرار می دهیم. (عملکرد آند مشترک نیز شبیه کاتد مشترک است)

جدول حالت های مختلف در 7Seg نوع کاتد مشترک:

عدد	A	B	C	D	E	F	G	DP	کد هگز
0	1	1	1	1	1	1	0	0	3F
1	0	1	1	0	0	0	0	0	06
2	1	1	0	1	1	0	1	0	5B
3	1	1	1	1	0	0	1	0	4F
4	0	1	1	0	0	1	1	0	66
5	1	0	1	1	0	1	1	0	6D
6	1	0	1	1	1	1	1	0	7D
7	1	1	1	0	0	0	0	0	07
8	1	1	1	1	1	1	1	0	7F
9	1	1	1	1	0	1	1	0	6F

جدول ۱-۶ نمایش اعداد مختلف با توجه به مقدار دهی پایه های سون سگمنت

نکته) برای 7Seg های نوع آند مشترک در جدول فوق ، می بایست جای صفر ها و یک ها با هم عوض شوند.

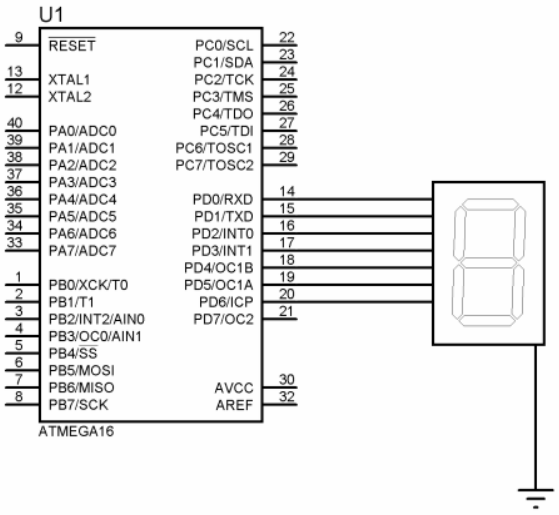
آزمایش ۶) شمارنده یک رقمی با 7Seg :

دستورات جدید) حلقه For – Next ، Dim ، **متغییرهای** Bit **و** Byte ، **جدول** Lookup

هدف) در این آزمایش به وسیله یک عدد سون سگمنت ، شمارنده ایی تک رقمی طراحی کرده ایم که از عدد 0 تا 9 را می شمارد و این کار بطور مداوم تکرار می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
```

```
Config Portd = Output
Dim W0 As Byte
Dim W1 As Byte
Do
For W0 = 0 To 9
W1 = Lookup(w0 , Main)
Portd = W1
Waitms 500
Next W0
Loop
End
Main:
Data &H3F , &H06 , &H5B , &H4F , &H66 , &H6D , &H7D , &H07 , &H7F , &H6F
```



شرح برنامه) در خط چهارم متغییری هشت بیتی (حافظه) با نام دلخواه W0 را برای شمارش 0 تا 9 تعریف کـرده ایم و نیز در خط پنجم متغییری دیگر از همان نوع هشت بیتی بـا نـام W1 را بـرای اسـتفاده از جـدول Lookup تعریف کرده ایم که در اینجا مختص اعداد سون سگمنت است. عملکرد حلقه For در خط هفتم به این صورت است که اعداد 0 تا 9 را به ترتیب شمرده و در متغییر W0 می ریزد و سپس متناسب با عدد قرار گرفته در این متغییر با استفاده از دستور Lookup ، مقدار هگز عدد قابل نمایش برای سون سگمنت از زیر برنامـه Main فراخوانـده مـی شود و در متغیر دیگری به نام W1 قرار داده می شود. این روند برای تک تک اعـداد بـا توجـه بـه دسـتور Next بطور مدام ادامه دارد و بصورت پیوسته تکرار می شود. در زیر برنامه Main هم مقدار هگزادسیمال اعداد به ترتیـب 0 تا 9 در مقابل دستور Data وجود دارد.

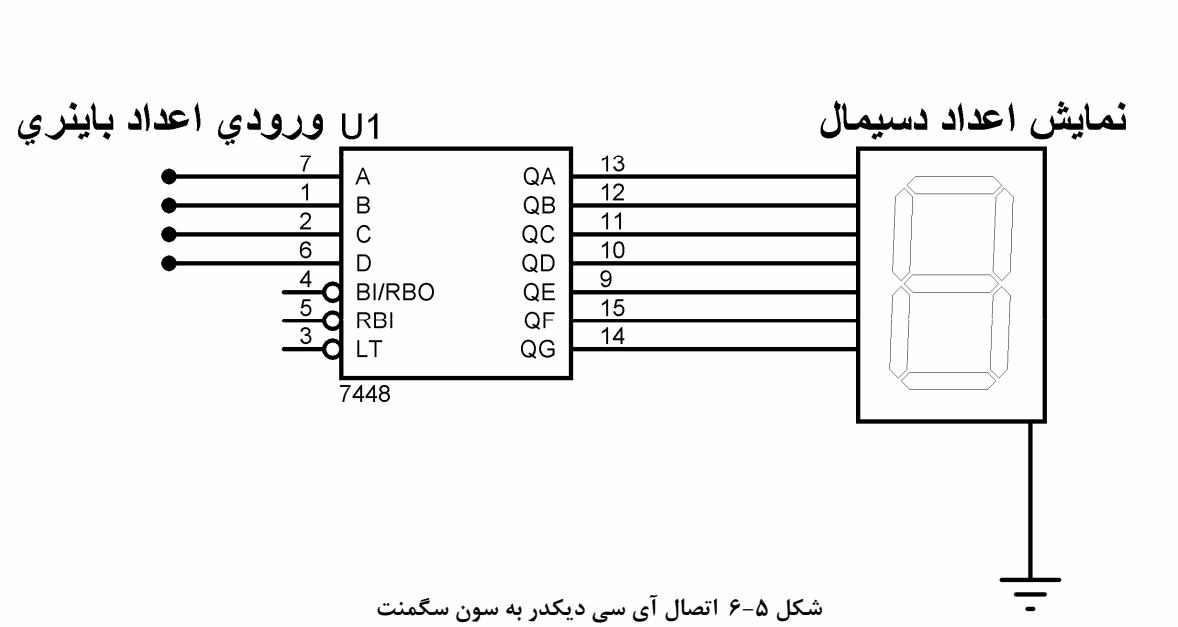
ج) راه اندازی 7Seg با استفاده از آی سی دیکدر (BCD) :

آی سی دیکدر (Decoder) کدهای باینری (صفر و یک) که از پایه های ورودی خود دریافت می کند را به مقـدار دسیمال یا همان اعداد صفر تا ۹ تبدیل می کند. و در واقع با استفاده از این قطعه در مدار مـورد نظـر ، کـار برنامـه نویسی بسیار آسان تر می شود و در تعداد پایه های مورد استفاده میکروکنترلر نیز صرفه جویی می شود.

آی سی های دیکدر از نوع TTL بوده (فقط با ولتاژ ۵ ولت کار می کند) و به شماره های زیر معروف است:

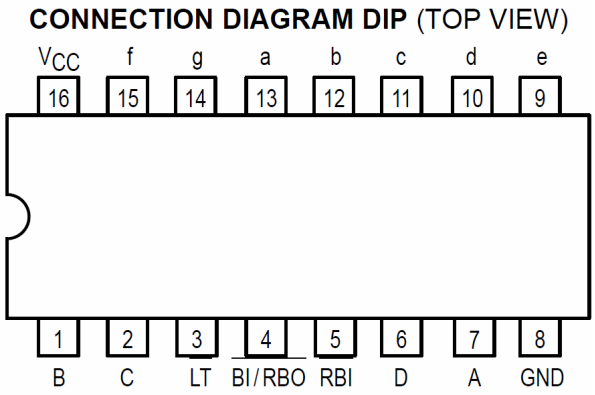
7446 و 7447 برای راه اندازی 7Seg های آند مشترک.

7448 برای راه اندازی 7Seg های کاتد مشترک.



شکل ۵-۶ اتصال آی سی دیکدر به سون سگمنت

شکل پایه های آی سی دیکدر (7447 و 7448):



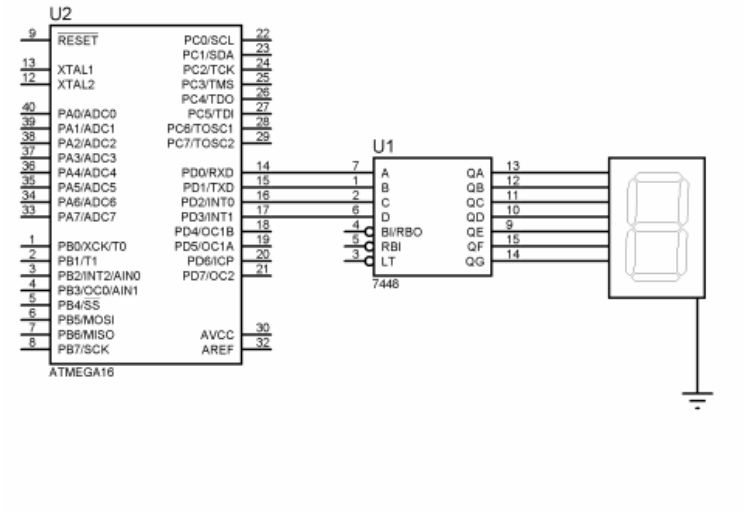
شکل ۶-۶ ترتیب پایه های آی سی دیکدر

برای اتصال ورودی و خروجی پایه ها دقیقا مانند نقشه قبلی عمل شود. پایه های 16 و 8 به ترتیب بـه قطـب هـای مثبت و منفی منبع تغذیه ۵ ولت وصل شود. مابقی پایه ها به جایی متصل نمی شوند.

آزمایش ۷) شمارنده یک رقمی با آی سی دیکدر و 7Seg :

هدف) عملکرد این آزمایش دقیقا مشابه آزمایش قبلی ست.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Portd = Output
Dim A As Byte
Do
For A = 0 To 9
Portd = A
Waitms 500
Next A
Loop
End
```



شرح برنامه) در این برنامه چون از آی سی دیکدر استفاده شده ، دیگر برای نمایش اعداد ، به جدول Lookup و زیر برنامه Data مربوط به مقدار هگزادسیمال اعداد نیازی نداریم که باعث ساده تر شدن برنامه نویسی می شود. در این روش فقط کافی ست مقدار هر عدد را در متغییر مربوطه (در این برنامه متغییر A) ریخته و سپس آن را در خروجی یکی از پورت های میکرو قرار دهیم و در آخر طبق مدار پورت را به آی سی دیکدر متصل کنیم تا عدد مربوطه رمزگشایی شده و برای سون سگمنت قابل نمایش باشد.

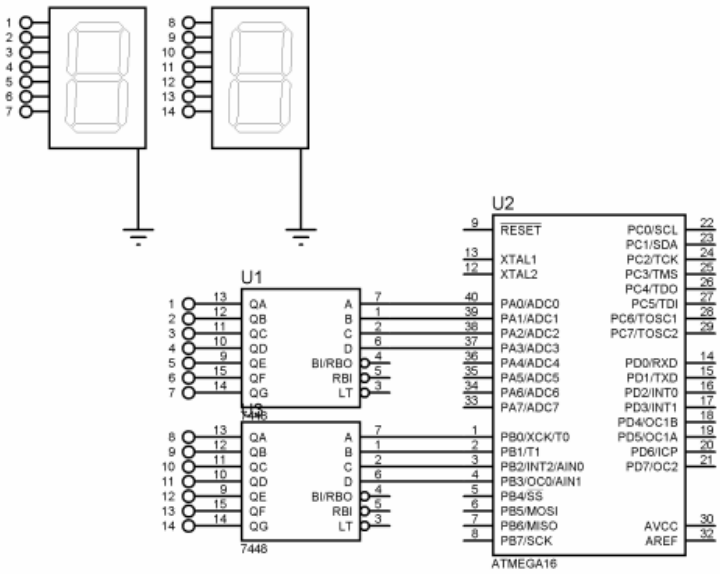
آزمایش ۸) شمارنده دو رقمی با آی سی دیکدر و 7Seg :

دستورات جدید) Return ، Gosub

هدف) نمایش اعداد دو رقمی به ترتیب از 0 تا 99 .

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Output
Config Portb = Output
Dim A As Byte
```

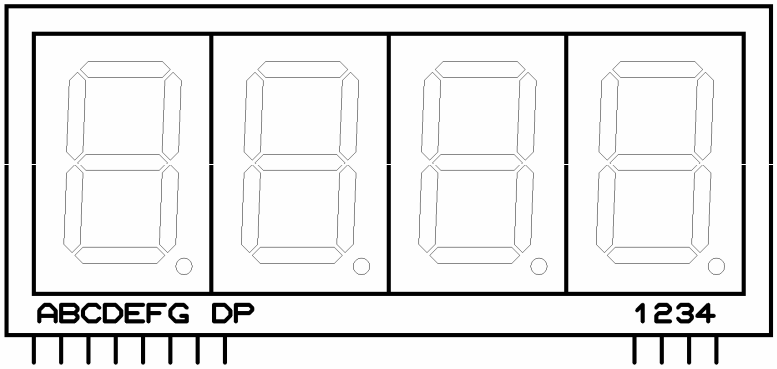
```
Dim B As Byte
Do
For A = 0 To 9
Porta = A
Gosub Yekan
Next A
Loop
Yekan:
For B = 0 To 9
Portb = B
Waitms 500
Next B
Return
End
```



شرح برنامه) در این برنامه برای اعداد یکان و دهگان دو متغییر مختلف با نام های دلخواه تعریف کرده ایم. متغییر A مربوط به اعداد دهگان و متغییر B مربوط به اعداد یکان می باشد. سپس برای هر یک از آن ها حلقه For – Next ایجاد می کنیم که از 0 تا 9 را بشمارد و در دو پورت جداگانه میکرو قرار می گیرد. ابتدا برای رقم دهگان یک شمارنده قرار می دهیم که از صفر شروع به شمردن می کند سپس در خط دهم از دستور Gosub استفاده شده که برنامه را به زیر برنامه Yekan پرش می دهد تا در آنجا شمارش اعداد برای رقم یکان صورت پذیرد مثلا در هر مرحله عدد دهگان رقمی ثابت بوده با دستور Gosub و لیبل Yekan رقم یکان به سرعت از 0 تا 9 شمرده می شود و پس از اتمام شمارش با رسیدن به دستور Return به ادامه برنامه یعنی خط یازدهم می رود که روند آن بصورت نمایش اعداد دو رقمی است.

در مدار این پروژه توجه شود که برای کاهش حجم سیم کشی بین آی سی های دیکدر و سون سگمنت ها از ترمینال استفاده شده است. به ترتیب شماره گذاری آن دقت شود !

نکته) غیر از سون سگمنت های معمولی ، سون سگمنت های مولتی پلکس هم در بازار موجود است که بصورت دوتایی یا چند سون سگمنت در کنار هم قرار دارند و برای صرفه جویی در تعداد پایه های مورد استفاده میکروکنترلر ، به روش اسکن (ماتریسی) برنامه ریزی می شوند.



۶-۷ سون سگمنت مولتی پلکس

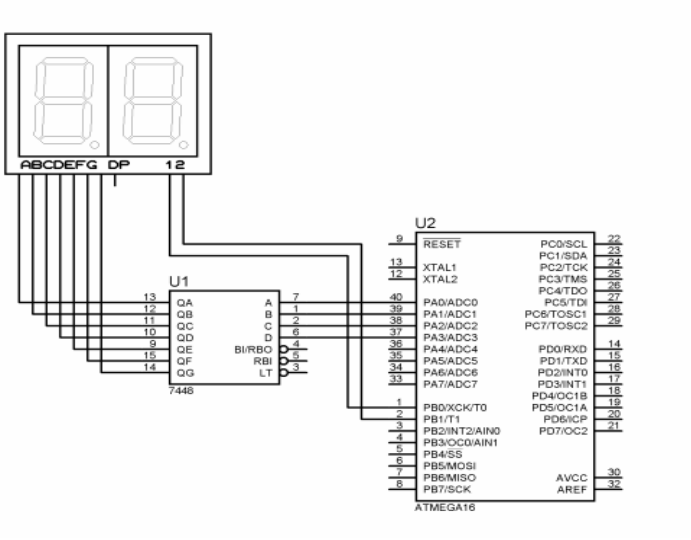
در شکل بالا اگر به پایه های مربوط به سون سگمنت ها کد هگزادسیمال اعداد را بدهیم (مثلا عدد 5) و در صورت اتصال همه پایه های مشترک به زمین مشاهده می کنیم که همه سون سگمنت ها همان عدد را نشان می دهند (مثلا عدد 5555 را مشاهده می کنیم). برای اینکه بتوانیم عدد چهار رقمی مختلفی را روی اینگونه سون سگمنت ها ببینیم می بایست از روش اسکن (ماتریسی) که در بالا اشاره شده استفاده کنیم. یعنی مثلا برای نشان دادن عدد 1390 می بایست در ابتدا و به ترتیب کد تک تک اعداد 1 و 3 و9 و 0 را به پایه های مربوط به سون سگمنت ها داده و برای نمایش هر عدد یکی از پایه های مشترک را به زمین وصل کنیم که حتما باید این کار با سرعت خیلی زیاد و در کسری از ثانیه صورت پذیرد (در حد چند هزارم ثانیه) تا با استفاده از خطای چشم انسان بتوانیم آن را بصورت یک عدد چهار رقمی (یا رقم های بیشتر و کمتر) ببینیم. در صورت پایین بودن سرعت اسکن در مدار ، اعداد را بصورت چشمکزن و یا حالتی به هم ریخته مشاهده می کنیم. برای این روش حتما می بایست از میکروکنترلر و ترانزیستور با سرعت سوئیچ زنی بالا استفاده شود. ولی در شبیه سازی نیازی به استفاده از ترانزیستور نداریم.

آزمایش ۹) نمایش عدد دو رقمی روی 7Seg مولتی پلکس:

هدف) در این آزمایش عدد ثابت 25 را روی سون سگمنت مولتی پلکس دو رقمی نمایش می دهیم.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Output
Config Portb = Output
Do
```

```
Portb = &B00000010
Porta = 2
Waitms 1
Portb = &B00000001
Porta = 5
Waitms 1
Loop
End
```



شرح برنامه) در این برنامه پورت A را مختص ارسال دیتای اعداد و پورت B را برای کنترل پایه های مشترک (اسکن این دو پایه) قرار داده ایم. طبق خط ششم و هفتم برنامه ، در ابتدا عدد 2 را به سون سگمنت ها ارسال می کنیم و در همان لحظه پایه مشترک شماره 1 (که مربوط به سون سگمنت دهگان است) را صفر می کنیم یعنی با این کار این پایه به زمین وصل شده جریان برقرار می شود که طبق آن سون سگمنت دهگان روشن و عدد 2 را نمایش می دهد. دقت شود زمان تاخیر این کار باید کمتر از ۱۰ میلی ثانیه باشد که ما در این آزمایش مدت زمان یک میلی ثانیه را در نظر گرفته ایم. در مرحله بعد همین کار برای ارسال عدد 5 صورت می گیرد و مهمترین تفاوت آن با حالت قبلی این است که طبق خط نهم برنامه زمین را به پایه شماره 2 سون سگمنت مولتی پلکس اختصاص می دهیم و همزمان عدد 5 را ارسال می کنیم. کل این اعمال را در حلقه بی نهایت Do – Loop قرار می دهیم که نتیجه آن مشاهده عدد ثابت 25 بر روی سون سگمنت مولتی پلکس است. همانطور که قبلا هم اشاره شد مهمترین مزیت این روش ، صرفه جویی در تعداد پورت های استفاده شده میکروکنترلر در نمایش اعداد چند رقمی می باشد.

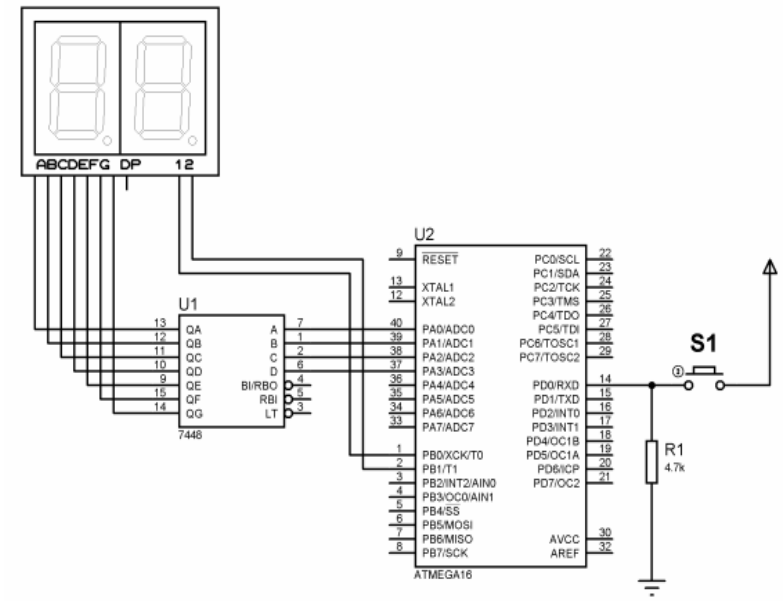
آزمایش ۱۰) معکوس شمار با کلید کنترلی:

دستورات جدید) Mod ، Decr ، دستورات ریاضی

هدف) در این آزمایش شمارنده ایی طراحی کرده ایم که در ابتداد عدد 99 را نشان می دهد و با هر بار فشردن کلید S1 ، یک رقم از آن کم می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Output
```

```
Config Portb = Output
Config Portd = Input
Dim A As Byte
Dim B As Byte
Dim C As Byte
A = 99
Do
If Pind.0 = 1 Then Decr A
If A = -1 Then A = 99
B = A / 10
C = A Mod 10
Portb = &B00000010
Porta = B
Waitms 50
Portb = &B00000001
Porta = C
Waitms 50
Loop
End
```



شرح برنامه) عملکرد این برنامه تقریبا شبیه برنامه قبلی ست ، با این تفاوت که در اینجا مقدار عدد نمایشی را می توانیم عوض کنیم. در ابتدا سه متغییر با نام های دلخواه A و B و C با ظرفیت بایت ایجاد می کنیم. در متغییر A عدد 99 را قرار می دهیم که همان عدد نمایشی ابتدای برنامه است. متغییر B مربوط به رقم دهگان و متغییر C مربوط به رقم یکان است. در خط یازدهم برای کلید S1 که به پین D.0 میکرو متصل است ، شرط حساس به یک ایجاد می کنیم. یعنی به محض اینکه پین D.0 برای یک لحظه کوتاه به ولتاژ 5 ولت وصل شد با توجه به دستور Decr یک رقم از عدد داخل متغییر A کم می شود. در خط دوازدهم نیز شرطی دیگر لحاظ می کنیم که اگر مقدار متغییر A کوچکتر ار عدد صفر شد ، مقدار متغییر A دوباره 99 شود و شمارش از نو شروع شود. با توجه به خط سیزدهم برای اینکه بتوانیم رقم یکان و دهگان هر عدد دو رقمی را از هم تفکیک کرده و به دو متغییر مختص آن یعنی C و B انتقال دهیم (برای نمایش روی سون سگمنت ها) می بایست ابتدا آن عدد دو رقمی را بر عدد ۱۰ تقسیم کرده ، رقم خارج قسمت آن را (که یک عدد صحیح است) در متغییر B می ریزیم. در خط چهاردهم نیز باقیمانده این تقسیم با توجه به دستور Mod به متغییر C منتقل می شود. مثلا در تقسیم 95 بر عدد 10 جواب آن عدد 9 و باقی مانده آن 5 می شود که برای نمایش آن کافیست عدد 9 را در متغییر B ریخته و عدد 5 را هم در متغییر C قرار می دهیم. عملکرد مابقی برنامه ، دقیقا شبیه به برنامه قبلی ست.

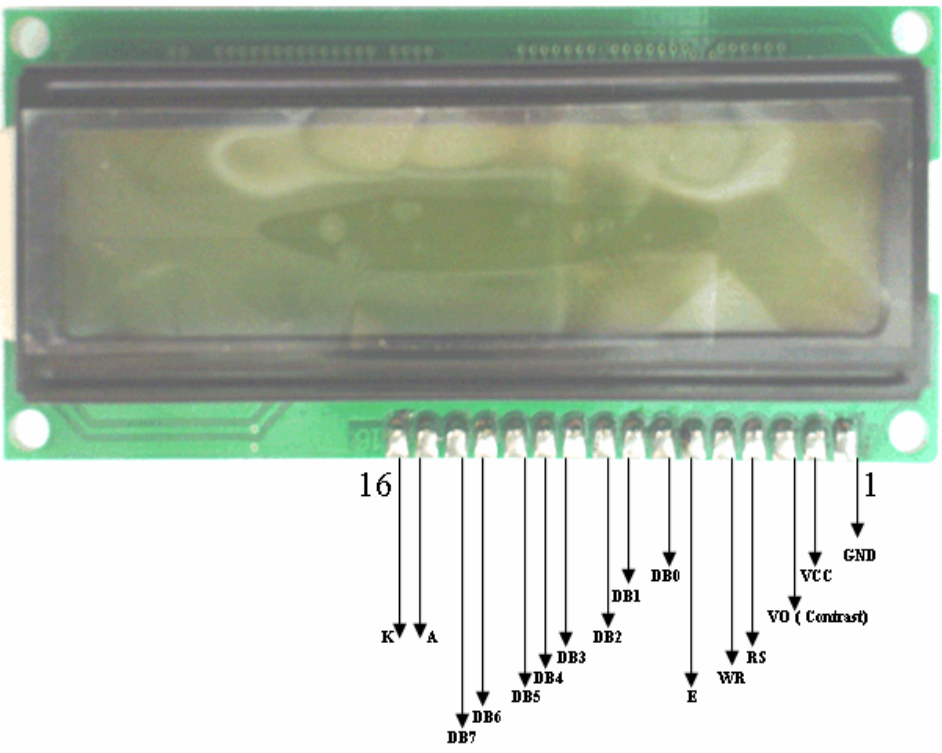
۲ – ۶ معرفی و پیکره بندی LCD کاراکتری:

LCD مخفف عبارت Liquid Crystal Display به معنای صفحه کریستال مایع بوده و امروزه در کاربردهای مختلفی برای نمایش اعداد ، کاراکترها ، تصاویر و... به عنوان نمایشگرهای کم مصرف ، ساده و با وضوح بالا استفاده می شود. به علت دارا بودن LCD به CPU و سایر مدارات جانبی دیگر بصورت مستقل ، کنترل آن کار بسیار ساده می باشد.

LCD ها از نظر ساختار به دو دسته کاراکتری و گرافیکی تقسیم بندی می شوند. خود LCD کاراکتری و همینطور گرافیکی نیز به چند دسته مختلف تقسیم می شوند.

مثلا LCD کاراکتری 2*16 (دارای ۱۶ ستون و ۲ سطر) و یا 4*16 و ...

نمای یک LCD کاراکتری:



شکل ۸-۶ معرفی پایه های LCD کاراکتری

شرح پایه های LCD کاراکتری:

۱- **VSS (GND):** پایه زمین LCD بوده که باید به قطب منفی تغذیه وصل شود.

۲- **VDD (VCC):** پایه تغذیه LCD بوده و باید به قطب مثبت منبع تغذیه (۵ ولت) وصل شود.

۳- **VO (Contrast):** پایه مربوط به کنتراست نمایش می باشد.(به پتانسیومتر 1K متصل می شود)

۴- **RS (Register Select):** انتخاب کننده ثبات LCD می باشد و شامل دو فرمان زیر می شود:

ارسال فرمان به LCD (کد صفر)

ارسال داده به LCD (کد یک)

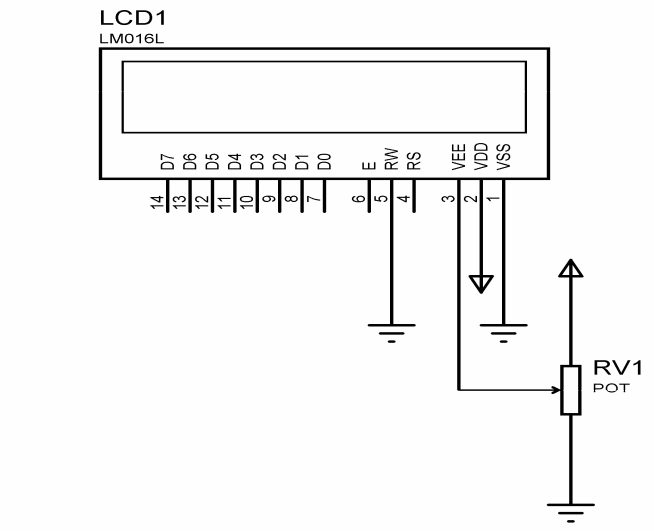
۵- **RW (Read / Write):** برای تعیین نوع عمل درخواستی از LCD است. در عمل خوندن این پایه برابر یک و در عمل نوشتن روی LCD این پایه برابر صفر میگردد. معمولا چون روی LCD ها عمل نوشتن صورت می گیرد ، این پایه را به زمین (صفر) وصل می کنیم.

۶- **E (Enable):** با ایجاد پالس مثبت یا منفی بر روی این پایه ، LCD عمل درخواستی را با توجه بـه وضـعیت RS و R/W انجام می دهد.

۷- پایه های **DB0** تا **DB7** برای ارتباط و تبادل داده یا فرمان از میکرو به LCD استفاده می شود.

۸- پایه های **A** (شماره ۱۵) و **K** (شماره ۱۶) نیز ، مربوط به LED داخلی صفحه LCD بوده که با اتصال آنها به منبع تغذیه ، کل صفحه LCD روشن می شود. (برای دید بهتر در تاریکی).

توجه) برای راه اندازی اولیه یک LCD کاراکتری طبق شکل زیر عمل می کنیم:



شکل ۹-۶ نحوه اتصال LCD به منبع تغذیه

برای پیکره بندی کلیه LCD های کاراکتری داریم:

```
Config LCD PIN = PIN , DB4 = PortB.4 , DB5 = PortB.5 , DB6 = PortB.6 , _
DB7 = PortB.7 , E = PortB.3 , RS = PortB.2

Config LCD = 16 * 2
```

دستورات مربوط به LCD :

۹- **دستور LCD)** برای نمایش مقدار متغییر یا عبارت (کاراکتر) به کار می رود.

```
LCD  W          (نمایش محتوای درون متغییر W)
LCD  "W"        (نمایش حرف W)
```

نکته) برای نمایش چند کاراکتر در یک خط و پشت سر هم ، بین آنها از علامت ; را قرار می دهیم.

۲- دستور CLS) برای پاک کردن صفحه نمایش LCD به کار می رود.

۳- دستور Display) برای خاموش یا روشن کردن صفحه نمایش است.

```
Display  ON / OFF
```

۴- دستور Cursor) برای تنظیم مکان نمای LCD است.

```
Cursor  ON / OFF / BLINK / NOBLINK
```

Blink برای چشمک زدن و Noblik برای حالت بدون چشمک به کار می رود.

۵- دستور Home) مکان نما را در اولین ستون و سطر قرار می دهد.

```
Home  UPPER / LOWER / THIRD / FOURTH
```

۶- دستور Locate) مکان نما را به مکان دلخواه می برد.

```
Locate  X , Y                (Y ستون و X سطر)
```

۷- دستور Shift Cursor) مکان نما را یک واحد به سمت چپ یا راست انتقال می دهد.

ShiftCursor LEFT / RIGHT

۸- دستور Shift LCD) کل صفحه نمایش را یک واحد به سمت چپ یا راست منتقل می کند.

ShiftLcd LEFT / RIGHT

۹- دستور Lowerline) مکان نما را به خط پایین تر می برد.

۱۰- دستور Upperline) مکان نما را به خط بالاتر می برد.

۱۱- دستور Thirdline) مکان نما را به خط سوم می برد.

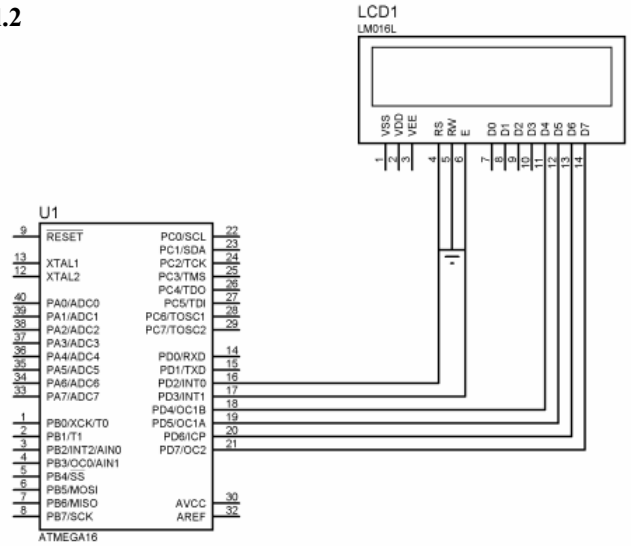
۱۲- دستور Fourthline) مکان نما را به خط چهارم می برد.

آزمایش ۱۱) نمایش کاراکتر (کلمه) و شمارنده اعداد سه رقمی :

دستورات جدید) Config LCD ، **متغییر** Word ، CLS ، Home ، "_" LCD

هدف) در این آزمایش عبارت " Value " و همینطور یک شمارنده سه رقمی (0 تا 999) روی صفحه LCD کاراکتری نمایش داده می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , _
Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Dim A As Word
Do
For A = 0 To 999 Step 1
Cls
Home
Lcd  "VALUE= " ; A
Waitms 500
Next A
Loop
End
```



شرح برنامه) در خط سوم برنامه LCD را پیکره بندی کرده ایم و در خط پنجم نوع LCD کاراکتری را از نظر تعداد سطر و ستون مشخص می کنیم. در خط ششم متغییری را با نام دلخواه A با ظرفیت Word (برای شمارنده سه رقمی) تعریف کرده ایم. در خط هشتم با ایجاد حلقه For ، یک شمارنده درست می کنیم که از 0 تا 999 بصورت تک پله (به ترتیب اعداد) می شمارد و این امر به خاطر وجود عدد 1 در مقابل دستور Step است. البته اگر مقدار عدد جلوی دستور Step را تغییر دهیم به همان مقدار عدد قرار داده شده پله ها تغییر می کند. مثلا اگر عدد 2 را جلوی دستور Step قرار دهیم ، شمارش بصورت دو تایی خواهد بود و فقط اعداد زوج را روی صفحه LCD مشاهده می کنیم. برای حالت معکوس شمار هم کافیست به اینصورت عمل کنیم:

For A = 999 To 0 Step -1

در خط نهم برنامه دستور CLS را قرار می دهیم تا کل صفحه LCD را پاک نماید و نوشتن آن قبل از سایر دستورات LCD در هر برنامه لازم است. در خط دهم دستور Home باعث می شود که همه کاراکترهایی که قرار است روی LCD نمایش داده شوند از ابتدای صفحه یعنی سطر و ستون اول نشان داده شوند. در خط یازدهم با استفاده از دستور LCD می توانیم کلمه دلخواه و یا محتوای متغییر موجود در برنامه را نمایش دهیم. یعنی اگر هر کلمه ایی را در جلوی دستور LCD و در داخل " ...." قرار بگیرد ، عینا روی صفحه LCD مشاهده می شود که در این آزمایش عبارت **"VALUE= "** قرار گرفته و نیز اگر هر متغییری که در جلوی دستور LCD قرار بگیرد محتوای درون آن روی صفحه LCD مشاهده می شود که در این برنامه شمارنده عدد سه رقمی 0 تا 999 است. البته برای جدا سازی بین کلمه مورد نظر و متغییر A ، علامت (**;**) را قرار می دهیم.

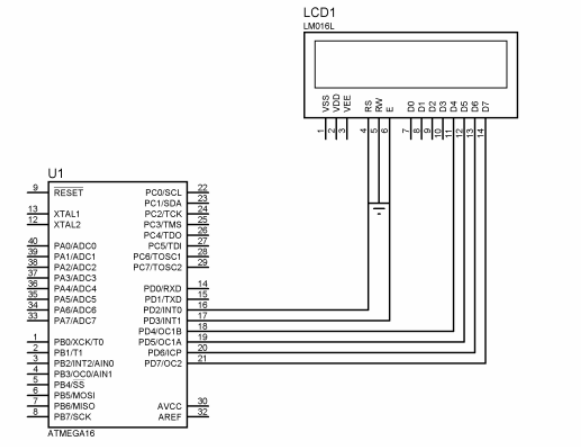
آزمایش ۱۲) ایجاد متن متحرک روی LCD :

دستورات جدید) Shiftlcd Right ، Locate ، Cursor Off

هدف) در این آزمایش ، عبارت " My Project " از سمت راست به سمت چپ صفحه LCD حرکت می کند.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , _
Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Dim A As Word
Do
Cls
Cursor Off
```

```
Locate 1 , 17
Lcd   "My Projects"
For A = 1 To 25
Shiftlcd Right
Waitms 300
Next
Loop
End
```



شرح برنامه) پیکره بندی و عملکرد این برنامه شبیه برنامه فبلی ست با این تفاوت که در ابتدا می بایست عبارت مورد نظر " My Project " را در آخرین سطر سمت راست قرار دهیم و چون LCD مورد استفاده 16 کاراکتر در سطر دارد پس عبارت را در آخرین سطر قرار می دهیم طوری که در ابتدای راه اندازی اصلا دیده نشود. این کار را در خط دهم و با استفاده از دستور Locate انجام داده ایم. حالا برای ایجاد حرکت به سمت چپ ابتدا یک حلقه For برای شمارش اعداد ایجاد می کنیم که این اعداد در واقع همان سطرهای LCD هستند که عبارت مورد نظر با استفاده از همین شمارنده موجود در حلقه For و نیز دستور Shiftlcd Right (خط سیزدهم) به سمت چپ حرکت می کند.

توجه) گاهی اوقات حرکت مشاهده شده روی LCD (بخصوص در حرکت به سمت راست) در شبیه سازی با اجرای واقعی که روی برد برد مونتاژ می شود متفاوت است.

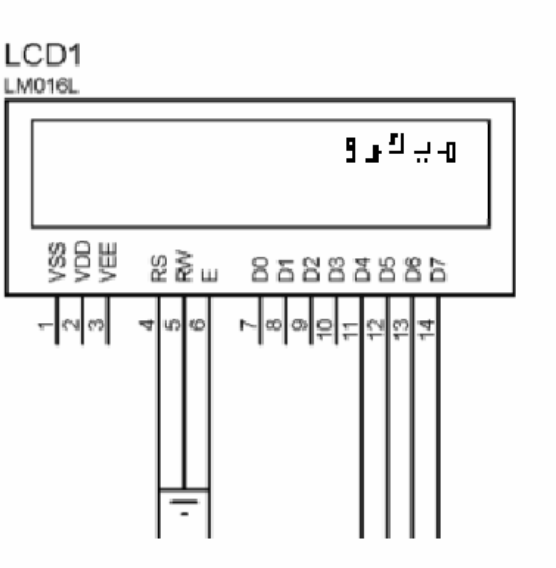
آزمایش ۱۳) ایجاد متن فارسی روی LCD :

دستورات جدید) Lcd Chr( ) ، Deflcdchar

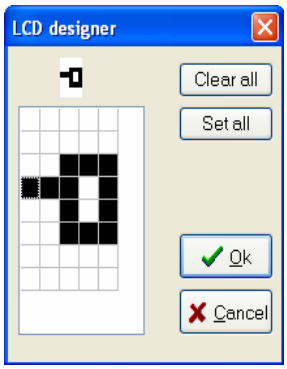
هدف) در این آزمایش کلمه " میکرو" را روی صفحه LCD مشاهده می کنیم.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , _
Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
```

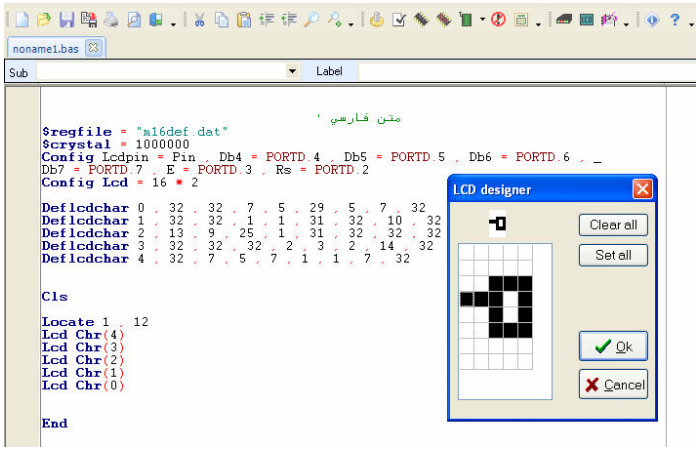
```
Deflcdchar 0 , 32 , 32 , 7 , 5 , 29 , 5 , 7 , 32
Deflcdchar 1 , 32 , 32 , 1 , 1 , 31 , 32 , 10 , 32
Deflcdchar 2 , 13 , 9 , 25 , 1 , 31 , 32 , 32 , 32
Deflcdchar 3 , 32 , 32 , 32 , 2 , 3 , 2 , 14 , 32
Deflcdchar 4 , 32 , 7 , 5 , 7 , 1 , 1 , 7 , 32
Cls
Locate 1 , 12
Lcd Chr(4)
Lcd Chr(3)
Lcd Chr(2)
Lcd Chr(1)
Lcd Chr(0)
End
```



شرح برنامه) عملکرد کلی این برنامه نیز دقیقا شبیه دو برنامه قبلی ست. برای نمایش کاراکتر یا شکل دلخواه بر روی LCD کارکتری می بایست آن را خودمان طراحی کنیم ، و برای این منظور در نرم افزار Bascom AVR از منوی Tools به قسمت LCD Designer رفته و سپس کاراکتر مورد نظر را ایجاد می کنیم.



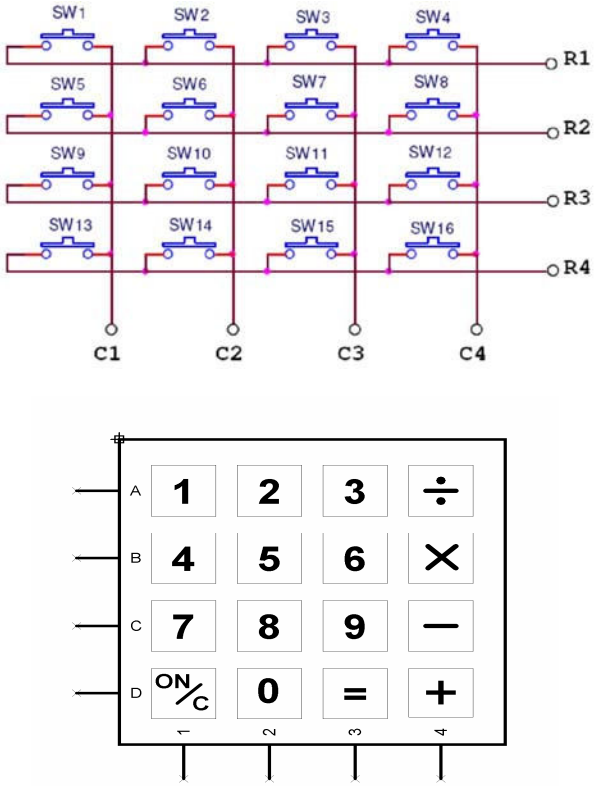
با فشردن دکمه OK نرم افزار کد هایی را با نام Deflcdchar(?) برای نمایش روی LCD به ما تحویل می دهد که می بایست در جای مناسب از آن بهره برد (مانند همین برنامه). و توجه شود که به جای علامت سوال (؟) کد Deflcdchar حتما باید عددی بین صفر تا ۷ به دلخواه قرار گیرد.



۳ – ۶ معرفی و پیکره بندی کی پد (Keypad) :

برای ارتباط و ارسال دستورات و اطلاعات به میکرو ، به دستگاه های ورودی نیاز است. کلیدهای مکانیکی (میکروسوئیچ ها) یکی از ساده ترین وسایل ورودی برای میکروکنترلر می باشند. کلیدها در دو نوع ساده (تکی) و ماتریسی (کی پد) وجود دارند.

کی پد از چند سطر و ستون تشکیل می شود. این سطر و ستون ها در حالت عادی هیچ اتصالی با هم نداشته و در صورت فشردن کلید ، سطر و ستون مربوط به آن کلید به هم متصل می گردد.



شکل ۱۰-۶ کی پد

برای پیکره بندی کی پد 4*4 داریم:

```
Config KBD = Port D , Debounce = 50 , Delay = 100
```

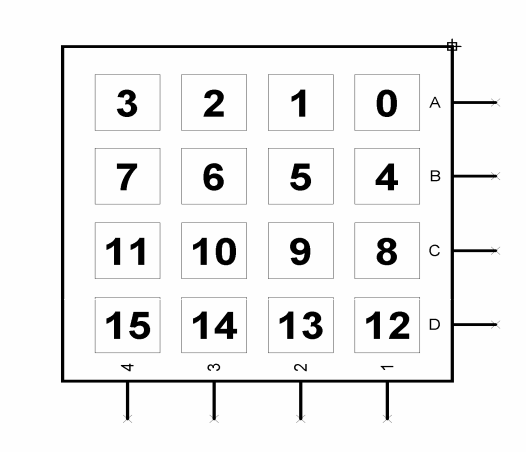
Debounce برای مدت زمان هنگام فشردن است (لرزش) و Delay مربوط به تاخیر بعد از فشردن می باشد (بر حسب میلی ثانیه).

دستور ( )Getkbd توسط این دستور ، میکروکنترلر کد مربوط به یکی از دکمه های صفحه کلیـد را خوانـده و سپس عدد متناظر با کلید فشرده شده را در متغییر مورد نظر قرار می دهد.

زمانی که کلیدی فشرده نشود ، این دستور عدد ۱۶ را بر می گرداند.

A = Getkbd( )

ترتیب عدد های ارسالی کی پد به میکرو به صورت زیر است (البته با شرط رعایت ترتیب پایه ها) :



شکل ۱۱-۶ ترتیب شماره گذاری اعداد کی پد توسط میکروکنترلر

نکته) برای مرتب کردن اعداد طبق ترتیب کی پد (یا دلخواه) ، می بایست از جدول Lookup استفاده کرد که بـه ترتیب جایگزین عدد صفر تا ۱۵ می شوند.

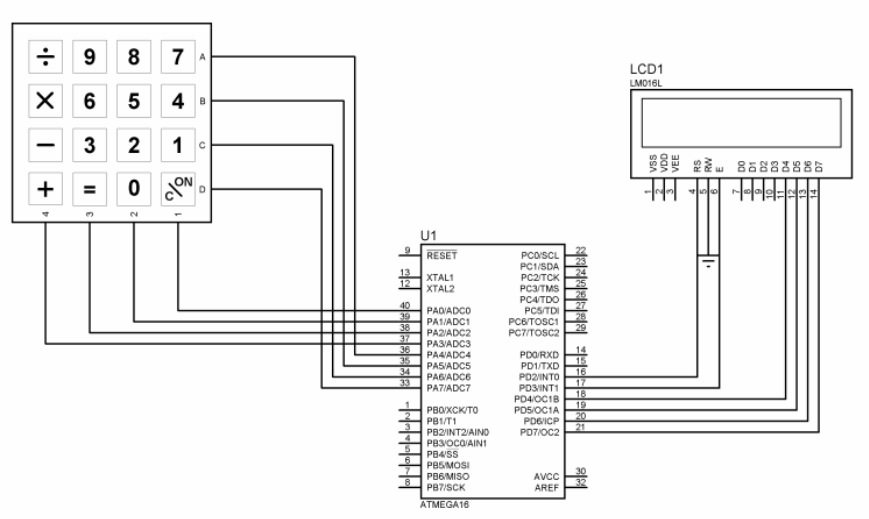
آزمایش۱۴) راه اندازی مقدماتی کی پد (صفحه کلید) :

دستورات جدید) Getkbd ( ) ، Config KBD

هدف) در این آزمایش با فشردن هر یک از کلیدهای کی پد ، عدد واقعی متناظر با آن روی صـفحه LCD نمـایش داده می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , _
Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Config Kbd = Porta , Debounce = 50 , Delay = 150
Dim A As Byte
```

```
Do
A = Getkbd( )
If A < 16 Then
Cls
Lcd A
End If
Loop
End
```



شرح برنامه) همانطور که در بالا هم اشاره شد ، در این آزمایش بافشردن هریک از کلیدها عـدد واقعـی مربـوط بـه پیکره بندی کی پد روی LCD ظاهر می شود که مطابق شکل ۱۱-۶ از 0 تا 15 است.

در ضمن توجه شود که این ترتیب اعداد با توجه به طریقه اتصال پایه ها از کی پد به میکرو در مدار بالا می باشد و اگر این اتصال را طور دیگری اتصال دهیم ، مشاهده می کنیم که جای اعداد هم عوض می شود.

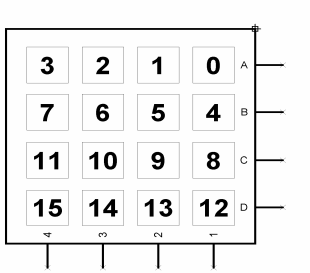
در خط ششم کی پد را در برنامه پیکره بندی کرده ایم و در خط هفتم متغییری با نام دلخواه A ، برای قرار گیـری اعداد دریافتی از کی پد به آن تعریف کرده ایم. در خط نهم برنامه اعداد دریافتی از کی پد در متغییـر A قـرار مـی گیرد. در خط دهم این شرط را برای برنامه قرار داده ایم که اگر اعداد گرفته شده از کی پد کوچکتر از عدد 16 بـود برنامه ادامه پیدا کند چون در زمانی که هیچ کلیدی را فشار نمی دهیم به طور مداوم عدد 16 و یا بزرگتـر از آن بـه میکرو ارسال می شود و اگر از این شرط استفاده نکنیم ، عملکرد برنامه دچار اختلال خواهد شد. (نوشتن این شرط در تمامی برنامه هایی که از دستور پیکره بندی KBD استفاده شده اسـت الزامیـست). در خـط دوازدهـم هـم هـر عددی که توسط فشردن کلیدهای کی پد در متغییر A قرار گرفته ، عینا روی صفحه LCD مشاهده مـی شـود. بـا توجه به قرار گرفتن دستورات مربوطه در حلقه بی نهایت Do – Loop ، برای هر بار فـشرده شـدن کـی پـد عـدد متناظر با آن روی LCD نمایش داده می شود.

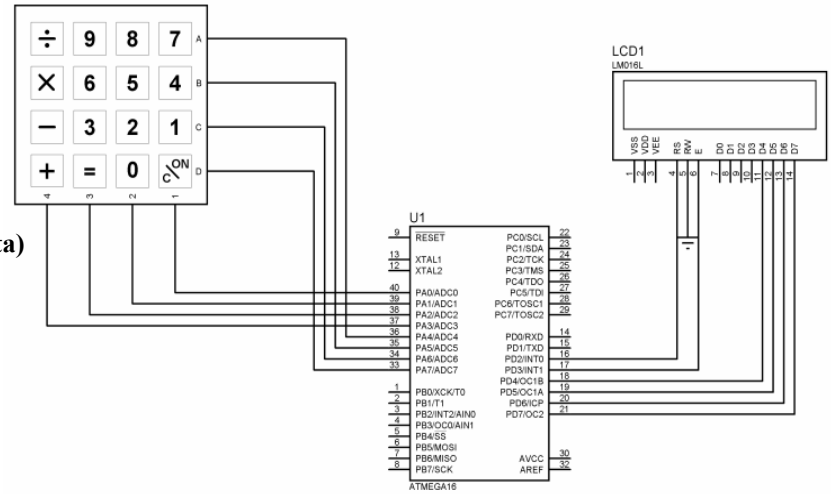
آزمایش ۱۵) راه اندازی کامل کی پد:

دستورات جدید) متغییر String ، Sub ، Declare ، Call

هدف) در این آزمایش با فشردن هر یک از کلیدهای کی پد ، عدد و علامت های نوشته شـده روی آن ، در صـفحه LCD مشاهده می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , _
Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Config Kbd = Porta , Debounce = 50 , Delay = 150
Declare Sub Key2
Dim A As Byte
Dim B As Byte
Dim C As String * 4
Key1:
A = Getkbd( )
If A = 16 Then Goto Key1
B = Lookup(a , Cdata)
If B >= 10 Then Call Key2
Lcd B
Goto Key1
Key2:
B = B / 10
Decr B
C = Lookupstr(b , Sdata)
Lcd C
If B = 3 Then
Cls
End If
Goto Key1
Return
Cdata:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
Sdata:
Data "+" , "=" , "Data "/" , "*" , "-" , "on/c
```





شرح برنامه) عملکرد این برنامه شبیه برنامه قبلی ست اما چون می خـواهیم اعـداد نوشـته شـده روی کـی پـد را جایگزین اعداد واقعی آن (همان اعداد متناظری 0 تا 16 که توسط دستور ( ) Getkbd دریافت می شـود) کنـیم برنامه کمی گسترده تر شده است. در ابتدا سه متغییر با نام های دلخواه A (برای اعداد در یافتی از کی پد) ، B (برای نمایش اعداد روی LCD) و C که از نوع رشته ایی با ظرفیت ۴ کاراکتر است (برای نمایش علائم ریاضی و ... روی LCD) تعریف می کنیم. در خط هفتم دستور Declare Sub را داریم که وجود آن الزامیست زیرا در ادامـه برنامه از دستور Call (فراخوانی) استفاده شده است. دو لیبل (زیر برنامه) با نـام هـای دلخـواه Key1 و Key2 در این برنامه وجود دارد که Key1 برای نمایش اعداد و Key2 برای نمایش علائم و ... می باشد. در خط سـیزدهم بـا توجه به شرط قرار داده شده ، فقط در صورتی برنامه ادامه می یابد که اعداد قرار گرفته در متغییر A حتما کوچکتر از عدد 16 باشند (یعنی یکی از کلیدها حتما باید فشرده شـود) و اگـر عـدد داخـل متغییـر A برابـر 16 بـود (در صورتی که هیچ کلیدی فشرده نشده باشد) ، با توجه به دستور Goto ، برنامه به لیبـل Key1 برگـشته و در یـک حلقه تکرار قرار می گیرد. در خط چهاردهم با استفاده از دستور Lookup و نیز جدولی که در لیبل **Cdata** بـرای آن تعریف کرده ایم (با توجه به قرار گیری اعداد به صورت ترتیبی) ، به محض فشرده شدن هـر یـک از کلیـدهای کی پد و متعاقب آن عدد تولید شده از 1 تا 15 ، یکی از اعداد لیبل Cdata که با جدول Lookup مربوط است به داخل متغییر B انتقال داده شده و روی LCD مشاهده می شود. مثلا با توجه مدار و شکل کی پد قـرار داده شـده در بالا و نیز کلیدی که فشرده می شود ، اگر عدد صفر در جدول Lookup قرار بگیرد عـدد 7 روی صـفحه LCD نشان داده می شود. و یا اگر عـدد 9 در جـدول Lookup قـرار بگیـرد متنـاظر بـا آن عـدد 2 روی صـفحه LCD مشاهده می شود.

برای نمایش علامت ها (+ ، - ، * ، /) همان اعمال گفته شده در مورد اعداد را داریم که در این مورد لیبـل Key2 و جدول Lookupstr و نیز لیبل Sdata (که کلیه این علائم در آن قـرار گرفتـه انـد) نقـش دارنـد. البتـه بـرای تشخیص اینکه چه وقت علامت ها روی LCD ظاهر شوند ، یک دستور شرطی در خط پانزدهم قـرار داده ایـم کـه طبق آن اگر عدد دریافت شده از کی پد بزرگتر یا مساوی عدد 10 بـود (یعنـی اعـداد 10 تـا 15) ، برنامـه لیبـل Key2 را با استفاده از دستور Call فراخوانی کند. در خط نوزدهم که مربوط به اعداد بزرگتـر یـا مـساوی 10 (یـا همان زیر برنامه مربوط به نمایش علامت ها) است آنها را بر عدد 10 تقسیم و سپس در خط بیستم یک واحد از آن کم می کنیم برای آن که با ترتیب جدول Lookupstr و لیبل Sdata هماهنگ شود. مثلا جای عدد 7 در شـکل کی پد بالا ، عدد 20 قرار می گیرد که اگر بر عدد 10 تقسیم شده و یک واحد از آن کم شود ، عدد 1 حاصـل مـی شود که علامت متناظر با آن (*) است و روی صفحه LCD مشاهده می شود.

۴ – ۶ بررسی عملکرد چند دستور کاربردی دیگر:

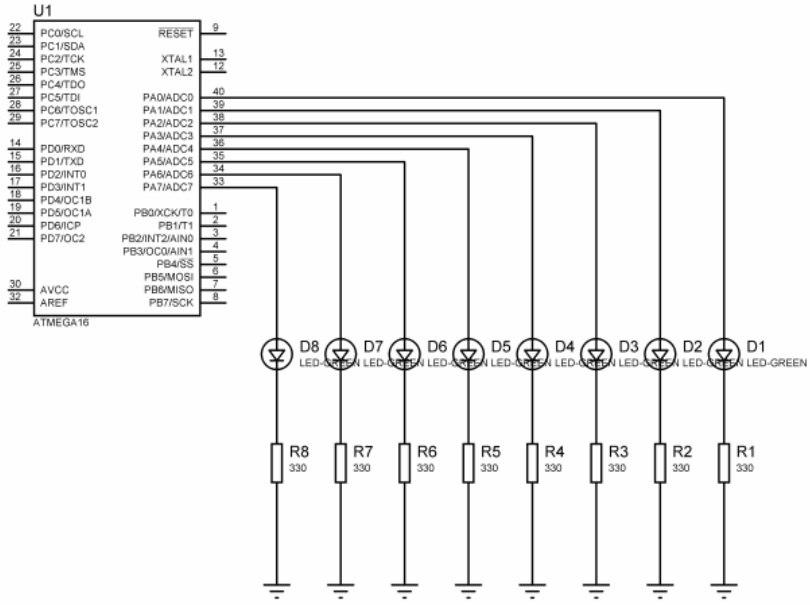
در این قسمت به بررسی چند آزمایش می پردازیم که در آن ها به یک سری از دستورات جدید نظیر ایجاد صدای بوق ، حلقه محدود کننده ، حلقه داخلی ، دستور چند شرطی و ... اشاره شده است.

آزمایش ۱۶) استفاده از دستور Rotate :

دستورات جدید) Rotate

هدف) ایجاد یک نوع چشمکزن راه رونده (چرخشی) بصورتی که در هر لحظه یکی از LED ها به ترتیب روشن و خاموش می شوند.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Output
Dim B As Byte
B = &B10000000
Do
Rotate B , Right
Porta = B
Waitms 500
Loop
End
```



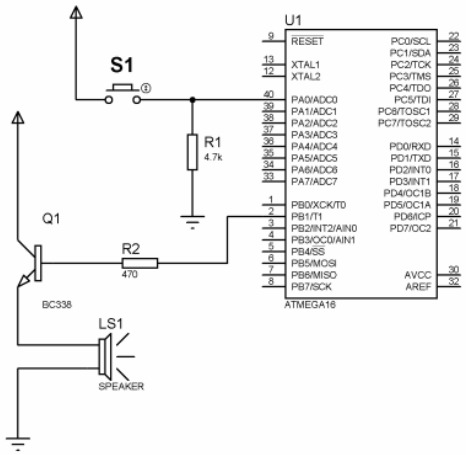
شرح برنامه) در خط پنجم یک مقدار باینری هشت بیتی را داخل متغییر B قرار داده ایم بطوری که فقط یک بیت از آن معادل عدد 1 باشد و بقیه 0 هستند. این کار باعث می شود در ابتدای کار یکی از LED ها روشن شود. سپس در خط هفتم با درج دستور Rotate باعث می شویم که سایر بیت ها به ترتیب و به سمت راست 1 شوند و نتیجه آن روی LED ها بصورت حرکت راه رونده نور مشاهده می شود.

آزمایش ۱۷) استفاده از دستور Sound :

دستورات جدید) Sound

هدف) با فشردن میکروسوئیچ ، صدای بوق از اسپیکر مدار شنیده می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta.0 = Input
Config Portb.1 = Output
Do
If Pina.0 = 1 Then
Sound Portb.1 , 100 , 150
End If
Loop
End
```



شرح برنامه) در خط ششم شرطی را برای فشردن میکروسوئیچ قرار داده ایم که در صورت فشرده شدن آن ، برنامه به خط هفتم رفته و صدای بوق با دستور Sound بر روی پین B.1 ایجاد می شود. عدد 100 مربوط به مدت پخش صدای بوق و عدد 150 مربوط به فرکانس صدای بوق می باشد. با تغییر این دو عدد به صداهای متفاوتی می توان دست یافت.

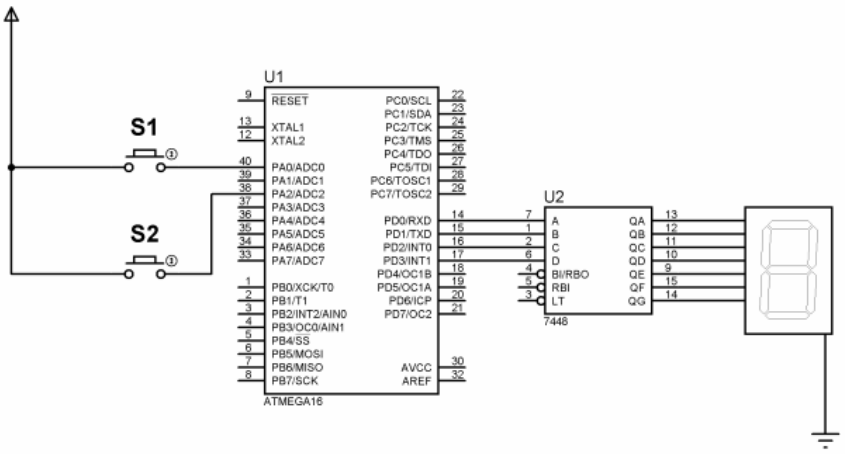
آزمایش ۱۸) استفاده از دستور Debounce :

دستورات جدید) Decr ، Incr ، Debounce

هدف) در این آزمایش با فشردن کلید S1 شمارنده بصورت صعودی می شمارد و با فشردن کلید S2 شمارنده نزولی می شمارد.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Input
Config Portd = Output
Config Debounce = 30
Dim A As Byte
Do
Debounce Pina.0 , 1 , Main1
Debounce Pina.2 , 1 , Main2
```

```
Loop
End
Main1:
Incr A
If A = 10 Then A = 0
Portd = A
Debounce Pina.2 , 1 , Main2
Wait 1
Goto Main1
Main2:
Decr A
If A = 255 Then A = 9
Portd = A
Debounce Pina.0 , 1 , Main1
Wait 1
Goto Main2
```



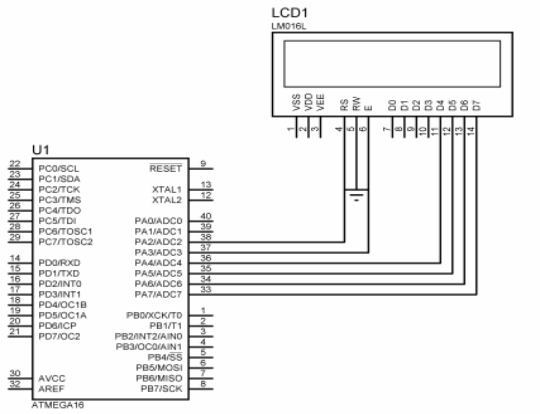
شرح برنامه) دستور Debounce را برای حساس کردن کلیدها نسبت به انجام عملی خاص تعریف می کنیم. برای بکاربردن این دستور در ابتدا می بایست آن را پیکره بندی کنیم که در خط پنجم این کار صورت گرفته است. عدد 30 نیز مقدار زمان فشرده شدن کلید و در نتیجه حساسیت آن نسبت به فشار را تعیین می کند. در خط هشتم و نهم نیز مشخص کرده ایم که اگر هر کدام از کلیدها فشرده شدند با توجه به لیبل تعریف شده برای آن ، به همان زیر برنامه پرش کرده و کار مربوطه را انجام می دهد. در این برنامه لیبل Main1 مربوط به حالت صعودی شمار و زیربرنامه Main2 مربوط به حالت نزولی شمار است. در خط سیزدهم برنامه با رسیدن به دستور Incr یک واحد به مقدار متغییر A اضافه می کند. در خط بیستم نیز برنامه با رسیدن به دستور Dcer یک واحد از مقدار متغییر A کم می کند. برای اینکه در حین انجام یکی از برنامه های لیبل ها بتوانیم به لیبل دیگری پرش کنیم (مثلا از حالت صعودی شمار به نزولی شما و یا بلعکس وارد شویم) لازم است دستور Debounce را در میان برنامه هر دو لیبل بگنجانیم.

آزمایش ۱۹) ایجاد حلقه محدود با Loop Until :

دستورات جدید) Loop Until

هدف) در این آزمایش ، یک شمارنده روی صفحه LCD ظاهر می شود که پس از شـمردن تـا عـدد ۲۵ برنامـه از حلقه Do – Loop خارج شده و شمارش متوقف می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Porta.4 , Db5 = Porta.5 , Db6 = Porta.6 , _
Db7 = Porta.7 , E = Porta.3 , Rs = Porta.2
Config Lcd = 16 * 2
Dim A As Byte
Do
Incr A
Cls
Lcd A
Waitms 500
Loop Until A = 25
End
```



شرح برنامه) در خط هشتم یک واحد به مقدار متغییر A افزوده می شود و روی LCD نشان داده می شود و ایـن افزایش تا جایی ادامه می یابد که عدد داخل متغییر A برابر با مقدار قرار داده شده در مقابل دسـتور Loop Until (خط دوازدهم) برسد که این امر باعث می شود برنامه از داخل حلقه Do – Loop که حلقه ایی بـی نهایـت بـود خارج شده و برنامه خاتمه یابد (در واقع حلقه بی نهایت را به مقدار تکرار دلخواه محدود می کنیم).

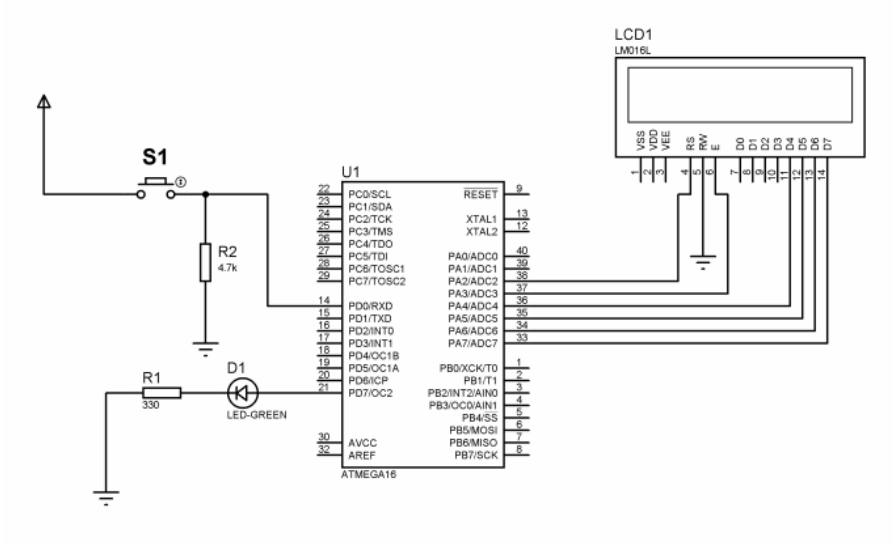
آزمایش ۲۰) ایجاد حلقه درونی بی نهایت تکرار با While_Wind :

دستورات جدید) While_Wind

هدف) در این برنامه پس از ۱۰ بار فشردن کلید S1 ، عملکرد کلید متوقف شده و LED شـروع بـه چـشمک زدن می کند و دیگر از این حالت خارج نمی شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Porta.4 , Db5 = Porta.5 , Db6 = Porta.6 , _
Db7 = Porta.7 , E = Porta.3 , Rs = Porta.2
```

```
Config Lcd = 16 * 2
Config Portd = Output
Dim A As Byte
Do
If Pind.0 = 1 Then
Incr A
End If
Cls
Lcd A
While A = 10
Toggle Portd.7
Waitms 400
Wend
Waitms 100
Loop
End
```



شرح برنامه) در خط نهم برنامه ، دستور شرطی ایی را برای کلید S1 قرار داده ایم که با هر بـار فـشرده شـدن آن یک واحد به مقدار متغییر A افزوده گردد. و درس زمانی که در خط چهاردهم با شرط حلقه While_Wind برابر شد (عدد ۱۰) ، برنامه به داخل حلقه While_Wind رفته و عمل داخل آن را که روشن و خـاموش کـردن یـک LED است را انجام می دهد و دیگر از داخل آن حلقه خارج نمی شود. یعنی برنامه بین خـط چهـاردهم و هفـدهم یک حلقه بی نهایت تکرار را انجام می دهد.

آزمایش ۲۱) کار با دستور شرطی Select_Case :

دستورات جدید) Call ، Declare Sub ، End Select ، Case Is ، Select_Case

هدف) در این آزمایش نیز مانند آزمایش ۱۸ ، با فشردن کلید S1 شمارنده بصورت صعودی می شمارد و با فـشردن کلید S2 شمارنده نزولی می شمارد.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta = Output
Config Portb = Input
Dim A As Byte
Dim B As Byte
```

Dim C As Byte

Dim D As Byte

Dim E As Byte

Declare Sub Main1

Declare Sub Main2

Declare Sub Main3

Do

Key:

C = Pinb

Select Case C

Case Is = &B00000001

D = 1

Case Is = &B00000010

D = 2

End Select

If D = 1 Then Call Main1

If D = 2 Then Call Main2

Loop

Main1:

D = E

For A = 0 To 9

B = Lookup (a , Dta)

Porta = B

Waitms 500

Call Main3

Next A

Goto Main1

Main2:

D = E

For A = 9 To 0 Step -1

B = Lookup (a , Dta)

Porta = B

Waitms 500

Call Main3

Next A

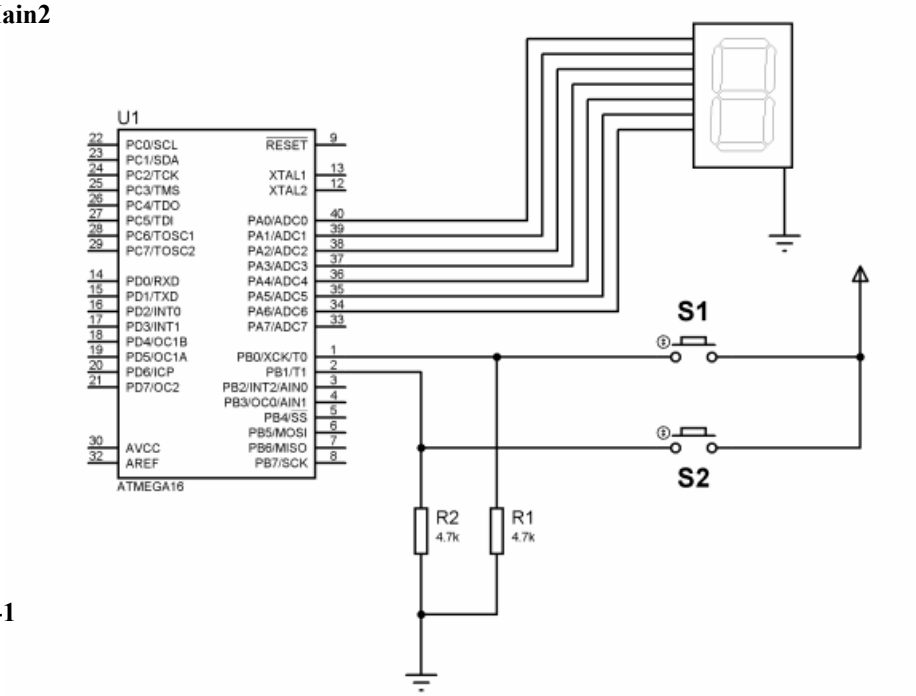
Goto Main2

Main3:

C = Pinb

Select Case C

Case Is = &B00000001



```
E = 1
Case Is = &B00000010
E = 2
End Select
If D <> E Then Goto Key
Return
Dta:
Data &H3F , &H06 , &H5B , &H4F , &H66 , &H6D , &H7D , &H07 , &H7F , &H6F
```

شرح برنامه) دستور Select_Case چند شرط را یکجا مورد بررسی قرار می دهد که در صورت رخ دادن یکی از آن شرطها ، برنامه عمل تعریف شده مربوط به آن را انجام می دهد. در این برنامه از این شرط برای انتخاب کلید فشرده شده استفاده کرده ایم که متناظر با آن لیبل مربوطه فراخوانی شده و عمل مربوط به آن که شمارنده صعودی یا نزولی است انجام می شود.

در خط دهم ، یازدهم و دوازدهم از دستور Declare Sub استفاده شده که در جلوی هر یک از آن ها نام یکی از لیبل های موجود در برنامه قرار دارد. این دستور زمانی در ابتدای هر برنامه استفاده می شود که بخواهیم در جایی از برنامه ، از دستور Call برای فراخوانی یک تابع یا لیبل (زیربرنامه) استفاده کنیم.

در خط پانزدهم همه بیت های پورت B را در داخل متغییر C قرار داده ایم تا به نوعی بتوانیم ورودی هر یک از پین ها را دریافت کنیم. در این مدار دو ورودی (کلیدها) برای این پورت قرار دارد. در خط شانزدهم با استفاده از دستور Select Case دو شرط را تعریف کرده ایم که با توجه به آنها ، اگر یکی از آن ها برقرار شود متغییر D یا عدد 1 و یا عدد 2 را می گیرد (به خط های هفدهم تا بیستم برنامه توجه شود). بعد از اتمام شرط های قرار داده شده در برنامه حتما باید دستور End Select را درج کنیم. در خط بیست و دوم و بیست و سوم هم با توجه به مقدار قرار گرفته در داخل متغییر D ، یکی از زیر برنامه های شمارنده را فراخوانی می کنیم. لیبل Main1 مربوط به شمارنده صعودی شمار است و لیبل Main2 مربوط به شمارنده نزولی شمار می باشد. البته یک لیبل دیگری به نام Main3 هم در برنامه وجود دارد که در آن همان دستورات شرطی Select با متغییر E قرار داده شده است (دقیقا مثل دستور بالایی) و وجود آن به این دلیل است که در صورتی که شرط در هنگام اجرای برنامه شمارنده ها تغییر کرد مقدار دو متغییر D و E نیز با هم نابرابر می شود (خط پنجاه و یکم) و این کار باعث می شود که برنامه از اول و با شرط جدید کار خود را آغاز نماید. مثلا هنگامی که در ابتدای برنامه کلید S1 را فشار می دهیم مشاهده می کنیم که سون سگمنت اعداد 0 تا 9 را به ترتیب می شمارد و اگر همان موقع کلید S2 را فشار دهیم می بینیم که با تغییر شرط ، این بار سون سگمنت اعداد 9 تا 0 را می شمارد.

۵ – ۶ معرفی و پیکره بندی تایمر / کانتر (Timer / Counter) :

برای طراحی پروژه های دقیق از نظر کارکرد زمانی ، حتما می بایست از مد تایمر/ کانتر میکرو کنترلر استفاده کرد. تایمر در واقع زمان سنج بوده و کلاک ساعت خود را از سیستم داخلی میکرو تامین می کند. ما با تایمر مـی تـوانیم مدت زمان وقوع یک عمل را به طور دقیق اندازه بگیریم. به عبارت دیگر با هر بار شمرده شدن توسط تـایمر در یـک دوره مشخص ، رجیستر مربوط به آن در میکرو سرریز(Over Flow) می شود و با سرریز شدن آن می توان یـک عمل خاصی را توسط میکرو کنترل کرد.

کانتر به معنای شمارنده بوده که کلاک خود را از پایه خروجی TO دریافت می کند. در واقع با هر بار رسـیدن یـک پالس به این پایه ، یک عدد بصورت صعودی یا نزولی به کانتر افزوده یا کم می شود. کانترها معمـولا بـرای شـمارش تعداد دفعات اتفاق افتادن یک عمل خاص مورد استفاده قرار می گیرند.

بررسی انواع تایمر / کانتر در میکروکنترلر AVR :

Timer / Counter 0 : هشت بیتی بوده و تنها دارای دو مد تایمر و کانتر است. این تایمر/ کانتر پس از شمردن FF (255+1) سرریز می شود (رجیستر OVF0 برای تایمرصفر ، رجیستر TCNT0 برای کانترصفر).

Timer / Counter 1 : شانزده بیتی بوده و دارای مدهای تایمر ، کانتر ، PWM ، کپچر ، مقایسه ایـی (بـا دو رجیستر OCR1a و OCR2b) می باشد. این تایمر/کانتر پـس از شـمردن FFFF (65535) سـرریز مـی شـود (رجیستر OVF1 برای تایمر۱ ، رجیستر TCNT1 برای کانتر۱).

Timer / Counter 2 : هشت بیتی بوده و دارای مدهای تایمر ، PWM و مقایسه ایی (رجیستر OCR2) می باشد. این تایمر/ کانتر پس از شمردن FF (255+1) سرریز می شود (رجیـستر OVF2 بـرای تـایمر۲ ، رجیـستر TCNT2 برای کانتر۲).

نکته) البته در بعضی از سری های دیگر میکروکنترلر AVR ، تایمر دیگـری هـم بـا نـام Timer / Counter 3 وجود دارد.

دستورات مربوط به Timer / Counter :

۱- **Start Timer)** تایمر شروع به شمردن می کند.

۲- **Stop Timer)** تایمر متوقف می شود.

۳- **Enable OVF)** وقفه سرریزی فعال می شود. (تایمر و کانتر)

۴- **Enable Interrupts)** وقفه سراسری فعال می شود.

۵- **Timer = Value)** مقدار اولیه به تایمر یا کانتر می دهد.

۶- **Var = Timer)** محتوای تایمر یا کانتر را می خواند.

پیکره بندی تایمر/ کانتر میکرو در مد Timer :

۱- مد Timer 0 :

Config Timer 0 = Timer , Prescale = 1 / 8 / 64 / 256 / 1024

۲- مد Timer 1 :

Config Timer 1 = Timer , Prescale = 1 / 8 / 64 / 256 / 1024

۳- مد Timer 2 :

Config Timer 2 = Timer , Async = ON / OFF , Prescale = 1 / 8 / 64 / 256 / 1024

توجه شود که مد تایمر ۲ بصورت آسنکرون می باشد.

فرمول محاسبه زمان برای بدست آورن مدت یک ثانیه واقعی :

$$\left[\frac{\text{بیت تایمر} * \text{Prescale}}{\text{مقدار کریستال (هرتز)}}\right] * Y = 1s$$

به جای Y می بایست یک عددی خاص قرار دهیم تا مقدار برابر ۱ ثانیه بدست بیاید.

بعنوان مثال داریم :

$$[(8 * 200) / 8000000] * 5000 = 1s$$

نکته) تایمر/ کانتر صفر و دو در صورتی که پالس کلاک خود را از طریق کریستال خارجی 32.768 KHz دریافت کند می توانند بصورت آسنکرون نیز پیکره بندی شوند. که برای بدست آوردن زمان یک ثانیه دقیق (ساعت RTC با کریستال 32.768 KHz) استفاده می شود.

نکته) برای برگشت از زیر برنامه مربوط به وقفه سرریز حتما می بایست از دستور Return استفاده شود.

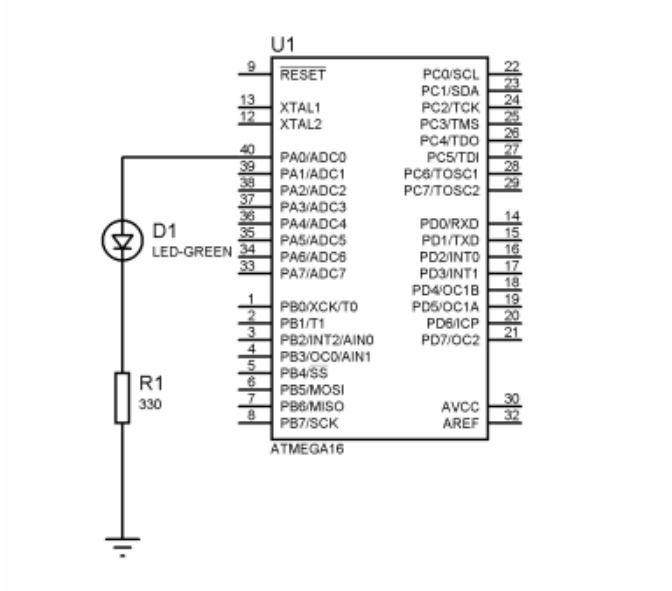
آزمایش ۲۲) کار با تایمر (ساخت مدت زمان یک ثانیه) :

دستورات جدید) Config Timer ، Enable Interrupts ، Enable Timer ، Start ، Stop

هدف) در این آزمایش با فعال کردن تایمر داخلی میکرو و ساخت مدت زمان یک ثانیه (به طور دقیق) ، دیود نورانی موجود در مدار به مدت یک ثانیه روشن و سپس به مدت یک ثانیه دیگر خاموش می ماند.

```
$Regfile = "m16def.dat"
$Crystal = 8000000
Dim A As Word
Config Pina.0 = Output
Config Timer0 = Timer , Prescale = 8
Enable Interrupts
Enable Timer0
On Timer0 Main
Timer0 = 56
Start Timer0
Porta.0 = 1
```

```
Do
Loop
End
Main:
Stop Timer0
Incr A
If A = 5000 Then
Toggle Porta.0
A = 0
End If
Timer0 = 56
Start Timer0
Return
```



شرح برنامه) در این برنامه توانسته ایم با توجه به فرمول گفته شده در بـالا ، بـا هـر بـار سـرریز (Over Flow) شدن رجیستر مربوط به تایمر (Timer0) ، مدت زمان یک ثانیه را تولید کنیم. (به مثال بالا هم توجه شود).

در خط پنجم برنامه Timer0 را در مد Timer پیکره بندی کرده ایم. به مقدار Prescale آن هم توجـه شـود. در خط ششم وقفه سراسری را فعال نموده ایم. فعال کردن وقفه سراسری در زمان پیکره بندی کلیه مـدهای Timer در برنامه الزامیست. در خط هفتم هم Timer0 را فعال می کنیم چون در غیر اینـصورت تـایمر هـیچ عملیـاتی در برنامه انجام نمی دهد. با توجه به خط هشتم ، هر موقع که مقدار داخلی رجیستر Timer0 سرریز شد ، برنامـه بـه لیبل Main پرش می کند. در خط نهم مقدار اولیه 56 را در درون رجیستر تایمر قرار می دهیم و با توجه به اینکه در این برنامه از مد Timer0 (هشت بیتی با عدد سرریزی 256) استفاده شده پس تایمر این برنامـه بـا شـمردن 200 بیت سرریز می شود (200 = 56 – 256). در خط دهم دستور شروع به کار فعالیت تایمر را می دهیم و در خط دهم خروجی پین A.0 را فعال می کنیم که متعاقب آن LED روشن می شود. همانطور کـه گفتـه شـد ، بـه محض سرریز شدن رجیستر مربوط به تایمر صفر (خط هشتم) برنامه به لیبل Main پرش کـرده و در آنجـا ابتـدا عملیات تایمر را متوقف می کنیم (خط شانزدهم) و با توجه به خط هفـدهم یـک واحـد بـه مقـدار متغیـر A مـی افزاییم. این افزایش را با توجه به شرط قرار داده شده در خط نوزدهم به مقدار 5000 می رسانیم و مهمترین هـدف ما از این کار این است که با توجه به فرمول گفته شده بتوانیم مقدار یک ثانیه دقیق را بدست آوریم. البته می توانیم اعداد دیگری هم در برنامه قرار دهیم به شرط آنکه با توجه به فرمول ، مدت زمان یک ثانیه را به ما تحویل بدهد.

$$[(8 * 200) / 8000000] * 5000 = 1s$$

در آخر نیز با دستور Toogle وضعیت پین A.0 را معکوس کرده (دیود نورانی خاموش می شود) سپس متغییـر A را خالی کرده و دوباره تایمر را مقداردهی می کنیم که نتیجه آن ایجاد حالت چشمکزن برای LED می باشد.

پیکره بندی تایمر / کانتر میکرو در مد Counter :

۹- مد Counter 0 :

Config Timer 0 = Counter , Edge = Rising / Falling

۲- مد Counter 1 :

Config Timer 1 = Counter , Edge = Rising / Falling

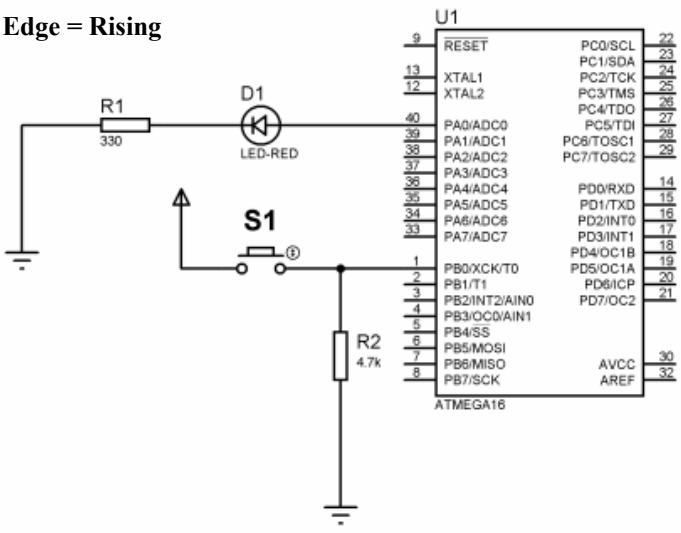
نکته) در برابر عبارت Edge ، گزینه Rising حساس به لبه بالا رونده می باشد (حساس به یک یا منبع 5 ولت) و کزینه Falling حساس به لبه پایین رونده است (حساس به صفر یا زمین منبع تغذیه).

آزمایش ۲۳) کار با کانتر :

دستورات جدید) Enable Ovf

هدف) در این آزمایش پس از پنج بار فشردن میکروسوئیچ S1 ، دیود نورانی موجود در مدار روشن می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Pina.0 = Output
Config Timer0 = Counter , Edge = Rising
Enable Interrupts
Enable Ovf0
On Ovf0 Main1
Counter0 = 251
Reset Porta.0
Do
Loop
End
Main1 :
Toggle Porta.0
```



Counter0 = 251

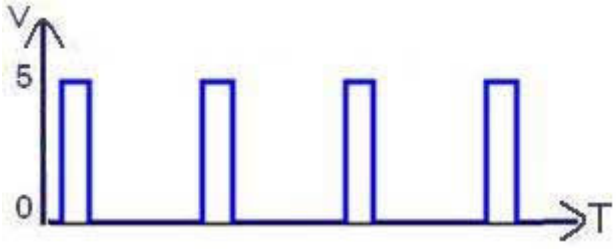
Return

شرح برنامه) در خط چهارم Timer0 را در مد Counter با لبه حساس به بالا رونده (ولتاژ 5 ولت) پیکره بندی کرده ایم. در خط ششم رجیستر کانتر را فعال می کنیم. با توجه به خط هفتم ، به محض اینکه رجیستر کانتر سرریز شود برنامه به لیبل Main1 پرش می کند. در خط هشتم ، رجیستر کانتر را مقدار دهی کرده تا باقی مانده آن برای سرریز شدن 5 بیت شود (5 = 250 – 256). عدد ۵ همان تعداد فشرده شدن کلید S1 برای روشن کردن LED است. در خط نهم خروجی پین A.0 را صفر کرده ایم تا در ابتدای کار دیود نورانی خاموش باشد. همانطور که گفته شد بعد از اینکه 5 بار کلید S1 را که به پین B.0 / T0 (پین مختص کانتر صفر) نیز متصل است فشرده شود (یا ولتاژ VCCبه آن داده شود) ، رجیستر کانتر سرریز شده برنامه به لیبل Main1 پرش کرده و در آنجا حالت پین A.0 را معکوس می کند که به ئنبال آن دیود نورانی روشن می شود و با توجه به مقدار دهی دوباره رجیستر کانتر در خط پانزدهم و برگشت برنامه به حالت اول اگر پنج بار دیگر کلید S1 را فشار دهیم LED خاموش می شود و این روند ادامه دارد.

تعریف PWM :

کلمه PWM مخفف عبارت Pulse With Modulation (مدولاسیون عرض پالس) است و هر گاه یک ولتاژ DC (مثلا ۵ ولت) را به سرعت قطع و وصل کنیم یک شکل موج مربعی ایجاد می شود که در آن زمان وصل شدن را T_{on} و زمان قطع شدن را T_{off} در نظر می گیریم. به مجموع این دو زمان T گفته می شود که دارای فرکانس خاصی است. ($T = T_{on} + T_{off}$)

در واقع PWM پالسی است با فرکانس ثابت و پهنای متغییر.



شکل ۱۲-۶ پالس PWM

یکی از کاربردهای PWM ، کنترل سرعت دور موتورهای DC می باشد. فرکانس PWM برای کنترل موتورهای DC معمولا در بازه 200 تا 300 هرتز انتخاب می شود.

نسبت زمان وصل به کل زمان (T_{on} / T) را دیوتی سایکل (Duty Cycle) یا چرخه کار مـی نامنـد و هـر چقـدر مقدار دیوتی سایکل بیشتر باشد ، مقدار سطح DC نیز افزایش خوهد یافت و بر عکس.

پیکره بندی تایمر/ کانتر میکرو در مد PWM :

۱- مد PWM 1 :

Config Timer 1 = PWM , PWM 8 / 9 / 10 , Compare A PWM = Clear Up / Clear Down / Disconnect , Compare B PWM = Clear Up / Clear Down / Disconnect , Prescale = 1 / 8 / 64 / 256 / 1024

۲- مد PWM 2 :

Config Timer 2 = PWM , PWM = ON , Compare PWM = Clear Up / Clear Down / Disconnect , Prescale = 1 / 8 / 64 / 256 / 1024

نکته) اگر در برابر عبارت Compare PWM گزینه Clear Up قـرار داشـت خروجـی پـالس PWM بـصورت معکوس در پایه خروجی مربوطه ظاهر می شود و در صورتی که گزینه Clear Down انتخاب شود خروجی پـالس به شکل غیر معکوس در پایه مربوطه ظاهر می شود. Disconnect هم برای قطع پالس در زمان مقایسه است.

نکته) در مد PWM1 دو پایه OC1B و OC1A (به ترتیب پایه های شـماره ۱۸ و ۱۹ میکروکنترلـر) بعنـوان خروجی پالس استفاده می شود. همینطور در مـد PWM2 پایـه OC2 (پایـه شـماره ۲۱ میکروکنترلـر) بعنـوان خروجی پالس مورد استفاده قرار می گیرد.

نکته) زمانی که از مد PWM استفاده می کنیم حتما می بایست پایه های مربوط بـه آن بعنـوان خروجـی پیکـره بندی شود.

نکته) مد PWM در تایمر/ کانتر یک ۸ ، ۹ و ۱۰ بیتی است ، ولی در تایمر/ کانتر ۲ فقط بصورت ۸ بیتی اسـتفاده می شود.

فرمول محاسبه مقدار فرکانس PWM :

$$F\ (Hz) = \frac{\text{کریستال (هرتز)}}{Y * \text{Prescale}}$$

به جای Y ، اعداد زیر را قرار می دهیم:

510 برای PWM هشت بیتی.

1022 برای PWM نه بیتی.

2046 برای PWM ۱۰ بیتی.

آزمایش ۲۴) کار با PWM :

دستورات جدید) OCR

هدف) در این آزمایش یک پالس مربعی (PWM) را با محتوای عدد 100 تولید کرده و در صفحه اوسیلوسـکوپ نرم افزار شبیه ساز مشاهده می کنیم.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Timer2 = Pwm , Prescale = 8 , Pwm = On , Compare Pwm = Clear Down
Config Pind.7 = Output
Enable Interrupts
Dim A As Byte
A = 100
Do
```

Ocr2 = A

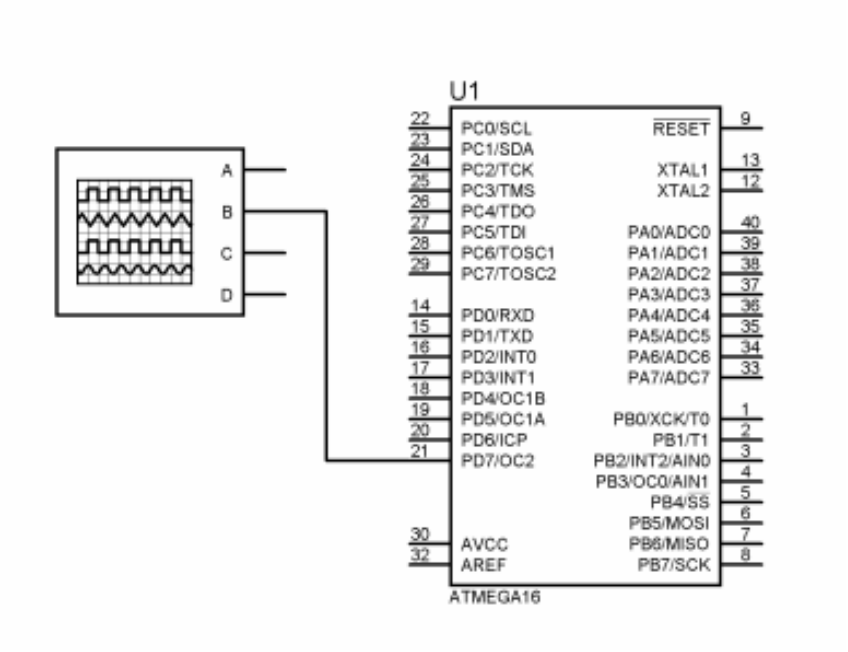
Loop

End

شرح برنامه) با توجه به مطالب گفته شده ، در این برنامه توانسته ایم یـک مـوج مربعـی را بـا فرکـانس 245 Hz تولید کنیم و روی صفحه اوسکوپ مشاهده نماییم. طبق فرمول بالا داریم:

(1000000 / (510 * 8) = 245.098)

در خط هفتم مقدار 100 را داخل متغییر A قرار داده ایم که باعث می شود دیوتی سایکل موج مربعی بر طبق آن ساخته شود (می توان این عدد را تغییر داد). و در نهایت (خط نهم) این مقدار را به خروجی اعمال می کنیم.



۶ – ۶ معرفی و پیکره بندی ADC :

در دنیای الکترونیک دو کمیت اصلی آنالوگ (پیوسته) و دیجیتال (نا پیوسته) وجـود دارد. در واقـع تمـامی سیـستم های منطقی مانند کامپیوترها ، پروسسورها و میکروکنترلرها تنها منطق دیجیتال را درک کرده و پردازش می کننـد و برای کار با سیگنال های آنالوگ (که اغلب از سنسورها به میکرو فرستاده می شوند) بایـد آن را بـا کمـک مـدارات ADC تبدیل به کدهای دیجیتال کرده و سپس به وسیله درگاه مورد نظر به واحد پردازشگر تحویل دهیم.

ADC مخفف عبارت Analog to Digital Converter می باشد و نوع ۱۰ بیتی آن در داخل میکروهای سـری Mega قرار دارد.

البته مبدل های ADC بصورت یک تراشه مستقل نیز در بازار موجود می باشند که معروف ترین آن آی سی هشت بیتی ADC0808 است.

ADC داخلی میکرو دارای دو منبع ولتاژ آنالوگ مجزا است. AVCC و AGND که AGND بایستی بـه زمـین یا ولتاژ زمین آنالوگ متصل شود و AVCC نباید بیشتر از 0.3V+ یا کمتر از 0.3V- نـسبت بـه VCC اخـتلاف داشته باشد.

ولتاژ مرجع (Voltage Reference) خارجی در صورت وجود باید بـه پایـه AREF وصـل شـود کـه ایـن ولتـاژ بایستی بین ولتاژ موجود بر روی پایه های AGND – AVCC باشد. در غیر اینصورت به VCC وصل مـی شـود. ADC مقدار آنالوگ ورودی را با تقریب متوالی به مقدار دیجیتال ۱۰ بیتی تبدیل می کنـد. کمتـرین مقـدار نـشان دهنده مقدار آنالوگ موجود در پایه AGND و بالاترین مقدار ، نشان دهنده ولتاژ پایه AREF منهای یـک LSB است.

مثلا اگر ولتاژ مرجع برابر VCC (۵ ولت) بود و با توجه به اینکه ADC میکرو ۱۰ بیتی است داریم:

5 / 1024 = 0.005

یعنی به ازای هر 5mV ولتاژ اعمالی به پایه ورودی ADC ، عدد مربوط به آن یک شماره صعود خواهد کرد.

فرمول محاسبه مقدار دیجیتال ولتاژ موجود بر روی پایه ADC :

$$ADC = \frac{1024 * V_{in}}{V_{ref}}$$

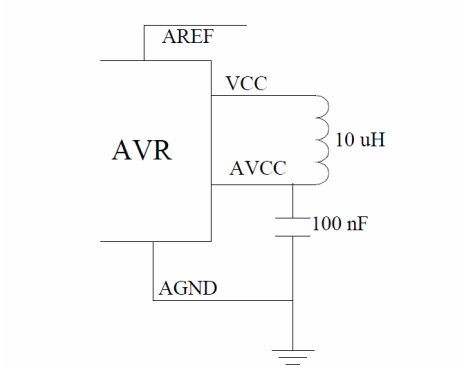
در این فرمول V_{in} ، ولتاژ موجود بر روی پایه ADC و V_{ref} ولتاژ مرجع می باشد.

برای پیکره بندی ADC داریم:

Config ADC = Single , Prescale = Auto , Reference = Optional

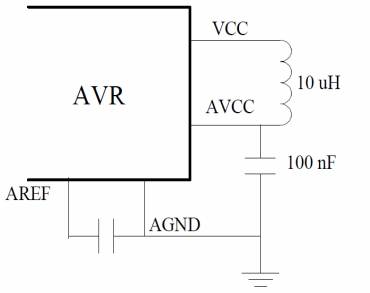
جای Optinal گزینه های زیر قرار می گیرد :

۱- OFF **:** زمانی استفاده می گردد که ولتاژ مرجع را از بیرون توسط پایه AREF بدهیم در این حالت از مدار زیر استفاده می کنیم.



شکل ۱۳-۶ کاهش نویز در زمان استفاده از ADC

۲- AVCC **:** زمانی استفاده می شود که ولتاژ پایه AVCC به عنوان مرجع در نظر گرفته شود. که در این حالت از مدار زیر استفاده می کنیم.



شکل ۱۴-۶ کاهش نویز در زمان استفاده از ADC

۳- INTERNAL **:** زمانی استفاده می شود که بخواهیم از ولتاژ مرجع داخلی آن ۲,۵۶ ولت استفاده کنیم. در این حالت نیز می توانیم از مدار بالا استفاده کنیم یا اینکه AREF را مستقیم به زمین متصل کرد. (البته AREF را می توان به VCC هم متصل کرد)

دستور ()Getadc با این دستور سیگنال آنالوگ وارد شده به کانال های صفر تا ۷ به مقدار دیجیتال تبدیل مـی شود و در متغییر B که می بایست از نوع داده Word باشد قرار می گیرد.

B = Getadc (0)

دستور Start برای شروع عملیات نمونه برداری می باشد.

دستور Stop عملیات نمونه برداری را پایان می دهد.

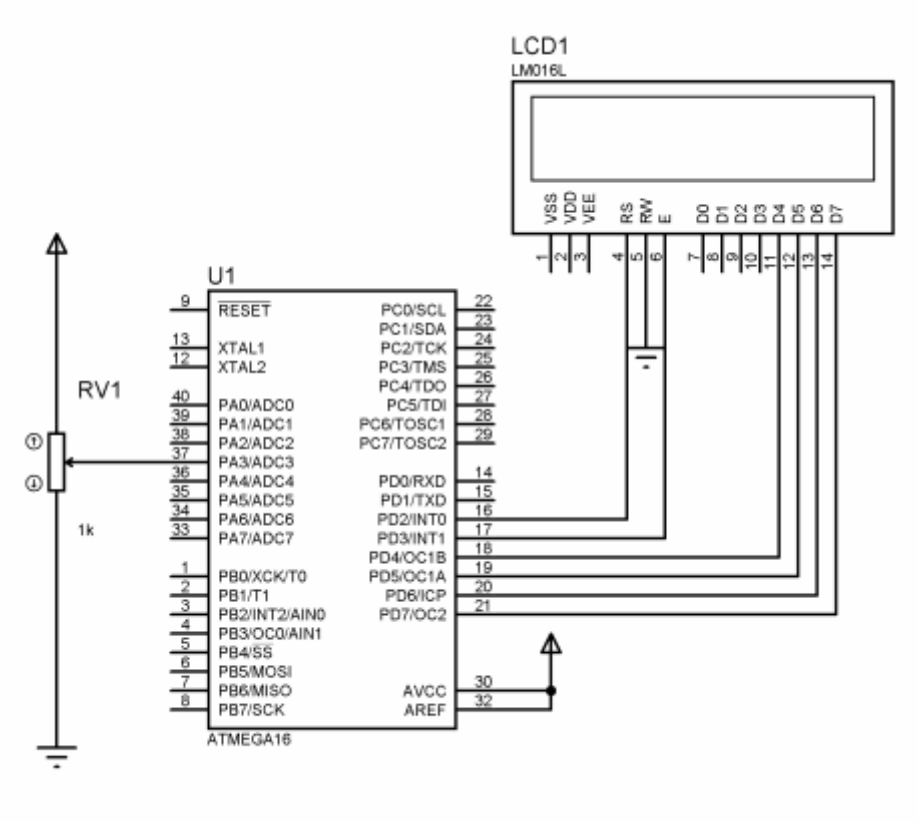
دستور Enable ADC فعال کردن وقفه ADC تا در صورت سرریز شدن به لیبل مربوطه پرش کند.

آزمایش ۲۵) راه اندازی ADC :

دستورات جدید) پیکره بندی ADC ، دستور ( ) Getadc

هدف) در این آزمایش با تغییر مقدار اهم پتانسیومتر (مقاومت متغییر) موجود در مدار ، ولتاژ آنالوگ اعمال شـده به ورودی ADC میکرو نیز تغییر کرده و سپس در میکرو به مقدار دیجیتال تبدیل می شود که می توان مقدار آن را روی LCD مشاهده کرد.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , _
Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Config Adc = Single , Prescaler = Auto , Reference = Internal
Dim B As Word
Start Adc
Do
B = Getadc(3)
Cls
Lcd B
Waitms 100
Loop
End
```



شرح برنامه) در مدار بالا با استفاده از پتانسیومتر (مقاومت متغییر) و اتصال دو پایه آن به قطب مثبت (VCC) و زمین (GND) ، یک مقسم ولتاژ درست کرده ایم که با توجه به تغییر مقدار اهم پتانسیومتر ولتاژ های مختلفی را (بین 0 تا 5 ولت) در پایه وسط آن داریم. در بسیاری از کاربردها من جمله استفاده از انواع مختلف سنسورها که با توجه به شرایط فیزیکی محیط مثل دما ، رطوبت ، دود و ... در پایه خروجی سنسور ولتاژ های مختلفی ظاهر می شود می بایست برای نشان دادن و یا کنترل مقدار آن (که معمولا از نوع آنالوگ هستند) از یکی از کانال های مبدل آنالوگ به دیجیتال (ADC) میکرو استفاده کنیم که در اینجا کانال ADC3 به کار گرفته شده است.

در خط ششم مبدل ADC را پیکره بندی کرده ایم و در ادامه آن داریم Reference = Internal که نشان می دهد از ولتاژ مرجع داخلی میکرو (همان ولتاژ 5 ولت) برای نمونه برداری استفاده کرده ایم. البته می توان از ولتاژ های مرجع دیگری هم استفاده کرد. در خط هفتم متغییری با نام دلخواه B و با ظرفیت Word تعریف کرده ایم و توجه شود که در اینجا نمی توان از ظرفیت Byte استفاده کرد چون مقدار آن برای مقادیر اعمالی از ADC کم می باشد. در خط دهم مقدار ADC را در متغییر B ریخته تا با توجه به خط دوازدهم روی LCD نشان داده شود.

در ضمن چون در این آزمایش از ولتاژ مرجع داخلی استفاده شده ، حتما می بایست پایه های AVCC و AREF به قطب مثبت منبع تغذیه 5 ولتی (VCC) وصل شوند.

۷ – ۶ معرفی و پیکره بندی وقفه ها (Interrupts) :

وقفه مکانیزمی است که در میکروکنترلرها برای پاسخگویی به برخی از اتفاقات در فرمان های خاص پیش بینی شده است. وقتی که وقفه ایی رخ می دهد ، برنامه در حال اجرا موقتا رها شده و زیر برنامه وقفه اجرا می شود و پس از پایان زیر برنامه ، برنامه اصلی از آنجایی که رها شده بود ، دوباره ادامه می یابد.

در میکروکنترلر AVR دو نوع وقفه داریم سخت افزاری و نرم افزاری داریم.

البته یک وقفه سراسری هم داریم که با فرمان Enable Interrupts فعال می شود.

برای پیکره بندی وقفه های خارجی داریم:

```
Config  INT0 = Rising / Falling / Low Level

Config  INT1 = Rising / Falling / Low Level
```

دستور Enable) برای فعال کردن وقفه های تعیین شده استفاده می کنیم.

دستور Disable) برای غیر فعال کردن وقفه های تعیین شده به کار می رود.

دستور On Interrupt) با این دستور میتوان یک زیر برنامه از زمان رخ دادن وقفه های تعیین شده را اجرا کرد.

دستور Idle) این دستورمیکرو را به حالت استراحت می برد. (صرفه جویی در مصرف توان و نیز کاهش نویز در زمان نمونه برداری ADC)

دستور Power Down) این دستور بخش CPU تراشه را در مد Power Down قرار می دهد.

آزمایش ۲۶) کار با وقفه (Interrupts) :

دستورات جدید) Enable Int0

هدف) در این آزمایش با فشردن هریک از کلیدها ، برنامه شمارنده نشان داده شده روی LCD متوقف شده و برنامه به حالت وقفه می رود تا عمل تعریف شده برای آن کلید را اجرا کند سپس پس از گذشت مدت زمانی

مشخص برنامه شمارنده از همان جایی که متوقف شده بود عملیات خود را از سر مـی گیـرد. بـا فـشردن کلیـد S1 عبارت "INTERRUPTS" به مدت ۲ ثانیه روی LCD ظاهر می شود. با فشردن کلید S2 صدای بوق از اسپیکر به مدت ۲ ثانیه پخش می شود و با زدن کلید S3 دیود نورانی به مدت ۲ ثانیه روشن می ماند و سپس خاموش می شود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Porta.4 , Db5 = Porta.5 , Db6 = Porta.6 , _
Db7 = Porta.7 , E = Porta.3 , Rs = Porta.2
Config Lcd = 16 * 2
Config Portb = Input
Config Pind.0 = Output
Config Pind.1 = Output
Dim A As Byte
Dim B As Byte
Enable Interrupts
Enable Int0
On Int0 Main2
Main1:
Cursor Off
For A = 10 To 0 Step -1
Cls
Locate 1 , 5
Lcd "HELLO Mr"
Locate 2 , 8
Lcd A
Wait 1
Next
Goto Main1
Main2:
B = Pinb
Select Case B
Case Is = &B00000001
Cls
Locate 1 , 4
Lcd "INTERRUPTS"
Case Is = &B00000010
Sound Portd.0 , 150 , 100
Case Is = &B00000100
```

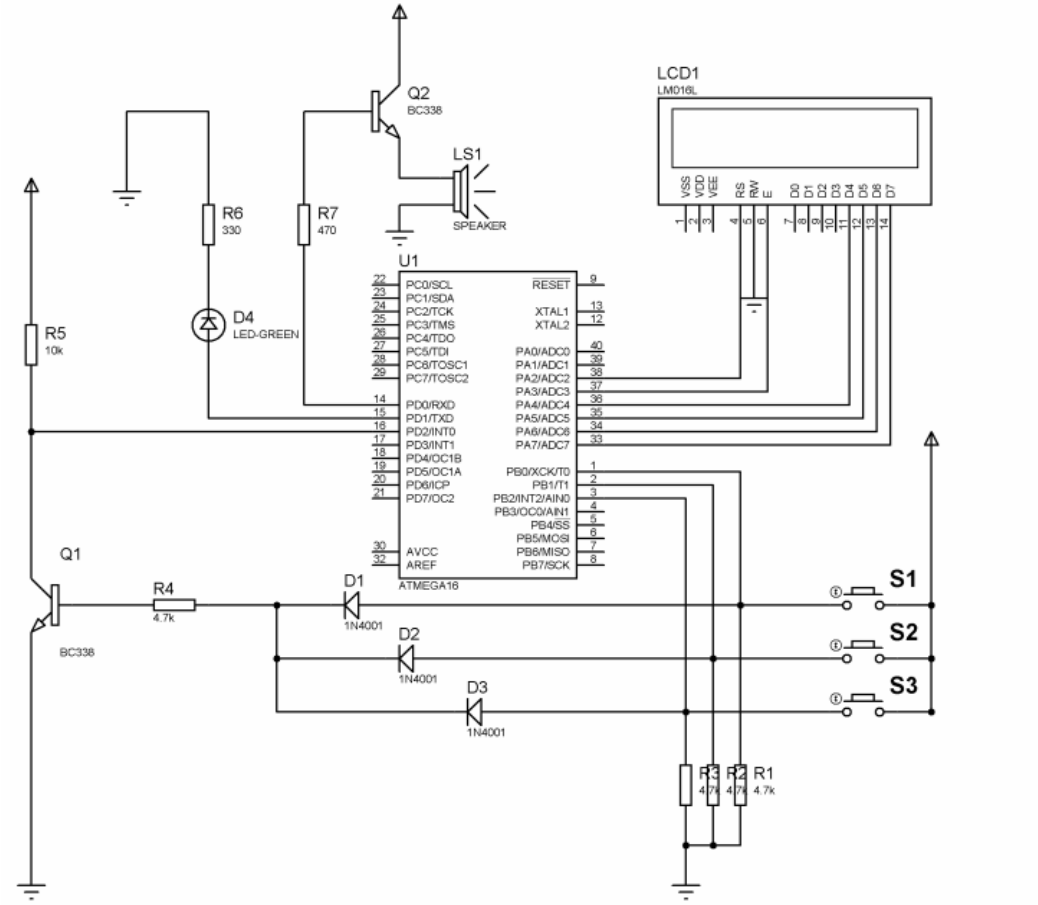
Set Portd.1

End Select

Wait 2

Reset Portd.1

Return



شرح برنامه) همانطور که در مدار بالا مشاهده می شود هر سه کلید برای ارسال دستورها و انجام کار مورد نظر به پورت B وصل شده اند و همه آنها در نهایت به پایه D.2 که همان پایه مربوط به INT0 است در ارتباطند. البته در مسیر آن از سه عدد دیود یکسوساز برای جلوگیری از تداخل دستور صادره از کلیدها و نیز یک عدد ترانزیستور برای تقویت جریان استفاده شده است.

در لیبل Main1 برنامه شمارنده را تعریف کرده ایم و در لیبل Main2 که زیربرنامه مربوط بـه وقفـه اسـت شـرط فشرده شدن برای هر یک از کلیدها و انجام عملیات مربوط به آن را مشخص کرده ایم.

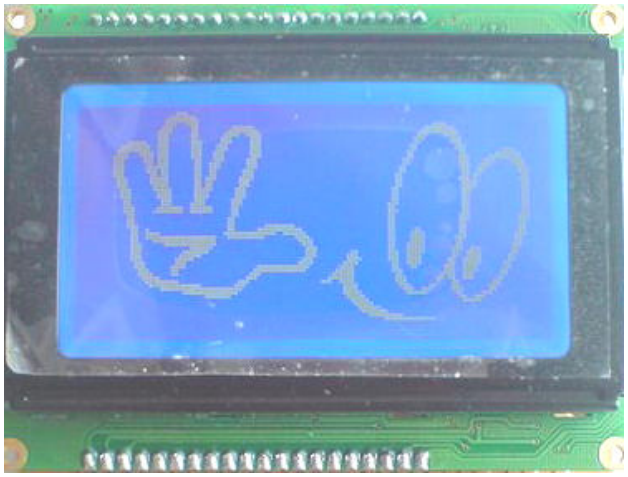
۸ – ۶ معرفی و پیکره بندی LCD های گرافیکی :

برای نمایش نمودارها و تصاویر از LCD گرافیکی استفاده می کنیم. این نوع LCD ها را می توان به دو دسته تقسیم کرد که این تقسیم بر اساس نوع تراشه مرکزی آن هاست:

۱- LCD های SED با تراشه KS108

۲- LCD های Toshiba با تراشه T6963

اگر بخواهیم این دو خانواده را با هم مقایسه کنیم ، مدل SED از نظر قیمت ارزان تر ولی از نظر سرعت پایین تر از LCD های نوع Toshiba می باشند. ولی از نظر ظاهری فرق خاصی ندارند. در ضمن انتخاب سایز فونت در LCD های T6963s با یک پایه انتخاب می شود ، ولی در SED توسط برنامه معین می گردد.



شکل ۱۵-۶ LCD گرافیکی

شرح پایه های LCD گرافیکی (T6963) :

۱- **FGND (GND) :** پایه زمین LCD بوده که باید به قطب منفی تغذیه وصل شود.

۲- **VSS (GND) :** پایه زمین LCD بوده که باید به قطب منفی تغذیه وصل شود.

۳- **VDD (VCC) :** پایه تغذیه LCD بوده و باید به قطب مثبت منبع تغذیه (۵ ولت) وصل شود.

۴- **VO:** پایه مربوط به کنتراست نمایش می باشد.(به پتانسیومتر 1K متصل می شود)

۵- **WR (Write) :** برای فعال سازی نوشتن در LCD می باشد.

۶- **RD (Read) :** برای فعال سازی خواندن از تراشه LCD است.

۷- **C/E (Chip Enable) :** برای فعال کردن چیپ موجود در LCD است.

۸- **C/D (Code / Data) :** برای انتخاب ورودی داده یا دستورالعمل می باشد.

۹- **VEE :** خروجی ولتاژ منفی. (به جایی متصل نشود)

۱۰- **RESET :** برای بازنشانی تراشه های LCD است.

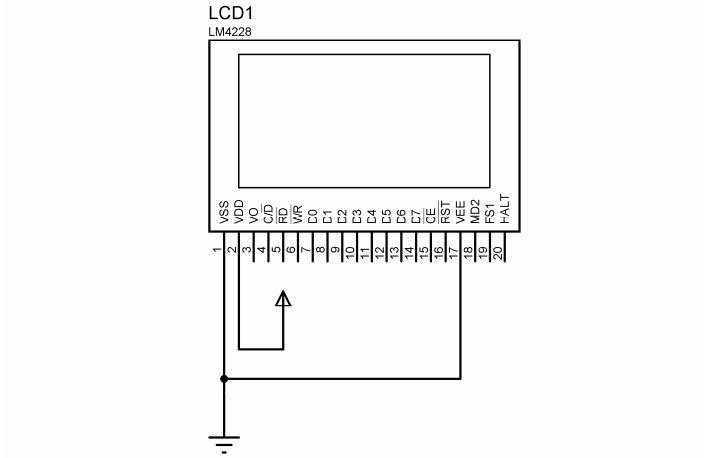
۱۱.....۱۸- پایه های **DB0** تا **DB7** برای ارتباط و تبادل داده یا فرمان از میکرو به LCD استفاده می شود.

۱۹- **FS :** برای انتخاب نوع فونت است که اگر 1 باشد فونت 8*6 و اگر 0 باشد فونت 8*8 را انتخاب می کند.

۲۰- **NC :** به جایی متصل نمی شود.

A : تغذیه مثبت برای روشن کردن LED داخلی. (دید بهتر در تاریکی)

K : تغذیه منفی برای روشن کردن LED داخلی. (دید بهتر در تاریکی)



شکل ۱۶-۶ نحوه اتصال LCD گرافیکی به منبع تغذیه

شرح پایه های LCD گرافیکی (KS108) :

۱- **CE (Chip Enable) :** برای فعال کردن چیپ اول موجود در LCD است.

۲- **CE2 (Chip Enable 2) :** برای فعال کردن چیپ دوم موجود در LCD است.

۳- **VSS (GND) :** پایه زمین LCD بوده که باید به قطب منفی تغذیه وصل شود.

۴- **VDD (VCC) :** پایه تغذیه LCD بوده و باید به قطب مثبت منبع تغذیه (۵ ولت) وصل شود.

۵- **VO :** پایه مربوط به کنتراست نمایش می باشد.(به پتانسیومتر 1K متصل می شود)

۶- **D/I (Data / Instruction)** : انتخاب کننده ثبات LCD می باشد. (پایه CD)

۷- **R/W (Read / Write)** : برای تعیین نوع عمل درخواستی از LCD است. (پایه RD)

۸- **EN (Enable)** : برای فعال سازی می باشد.

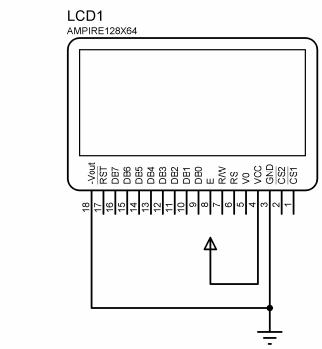
۱۶..... ۹- پایه های **DB0** تا **DB7** برای ارتباط و تبادل داده یا فرمان از میکرو به LCD استفاده می شود.

۱۷- RESET : برای بازنشانی تراشه های LCD است.

۱۸- VEE : خروجی ولتاژ منفی.

۱۹- A : تغذیه مثبت برای روشن کردن LED داخلی. (دید بهتر در تاریکی)

۲۰- K : تغذیه منفی برای روشن کردن LED داخلی. (دید بهتر در تاریکی)



شکل ۱۷-۶ نحوه اتصال LCD گرافیکی نوع SED به منبع تغذیه

توجه) ترتیب پایه ها برای همه LCD ها یکسان نیست و می بایست به کاتالوگ آن مراجعه نمود.

برای پیکره بندی LCD گرافیکی مدل T6963 داریم :

```
Config Graphlcd = 128 * 64 , Dataport = PortA , Controlport = PortB , Ce = 0 , _
Cd = 1 , Wr = 2 , Rd = 3 , Reset = 4 , Fs = 5 , Mode = 8
```

Mode مشخص کننده تعداد ستون متنی LCD است که می تواند 8 یا 6 باشد. همچنین مشخص کننده نوع فونت یعنی اگر Mode = 8 باشد ، تعداد ستون ها برابر 3 است ، و در صورت استفاده از Mode = 6 ، تعداد ستون ها برابر 4 می شود.

قبل از پیکره بندی KS108 ، نوشتن این دو دستور الزامیست;

```
$LIB  "GLCDKS108.LBX"
$INCLUDE  "FONT8*8.FONT"
```

همچنین باید به مسیر نصب نرم افزار بیسکام ، به قسمت Samples\LCDGraph رفته ، سپس فایل Font8*8.font را از آنجا کپی کرده و در فولدری که فایل های کامپایل شده برنامه در آنجا قرار دارد Paste می کنیم.

```
Config Graphlcd = 128 * 64 , Dataport = PortA , Controlport = PortB , Ce = 0 , _
Ce2 = 1 , Cd = 2 , Rd = 3 , Reset = 4 , Enable = 5
```

دستورات مربوط به LCD های گرافیکی (T6963 و KS108) :

۱- دستور LCD) برای نمایش مقدار متغییر یا عبارت (کاراکتر) در T6963 به کار می رود.

۲- دستور LCDAT) برای نمایش مقدار متغییر یا عبارت (کاراکتر) در KS108 به کار می رود.

۳- دستور CLS) این دستور تمام صفحه نمایش LCD ، هم قسمت متنی و هم قسمت گرافیکی را پاک می کند.

۴- دستور CLS Graph) فقط قسمت گرافیکی را پاک می کند.

۵- دستور CLS Text) فقط قسمت متنی را پاک می کند.

۶- 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.

۷- دستور Cursor) برای تنظیم مکان نمای LCD است.

```
Cursor  ON / OFF / BLINK / NOBLINK
```

Blink برای چشمک زدن و Noblik برای حالت بدون چشمک به کار می رود.

۸- دستور Pset) ایجاد یک نقطه بر روی LCD .

```
Pset  X , Y , Color
```

۹- دستور Line) ایجاد یک خط بر روی LCD .

Line (X0 , Y0) – (X1 , Y1) , Color

۱۰- دستور Circle) ایجاد یک دایره بر روی LCD .

Circle (X0 , Y0) , Radius , Color

۱۱- دستور Showpic) برای نمایش عکس می باشد.

Showpic X , Y , Lable

توجه) در دستورات بالا;

Color رنگ می باشد (0 یعنی بدون رنگ بوده و 255 با رنگ سیاه رسم خواهد شد).

Radius شعاع دایره می باشد.

X مکان قرارگیری افقی و **Y** مکان قرارگیری عمودی شکل یا عکس می باشد.

توجه) در زیربرنامه (Lable) مربوط به نمایش عکس ، حتما می بایست متن زیر نوشته شود که جای کلمه File ، اسم فایل عکس مربوطه قرار می گیرد.

$Bgf " File.Bgf "

آزمایش ۲۷) رسم نقطه ، خط و دایره بر روی LCD گرافیکی:

دستورات جدید) Circle ، Line ، Pset ، Config Graphlcd

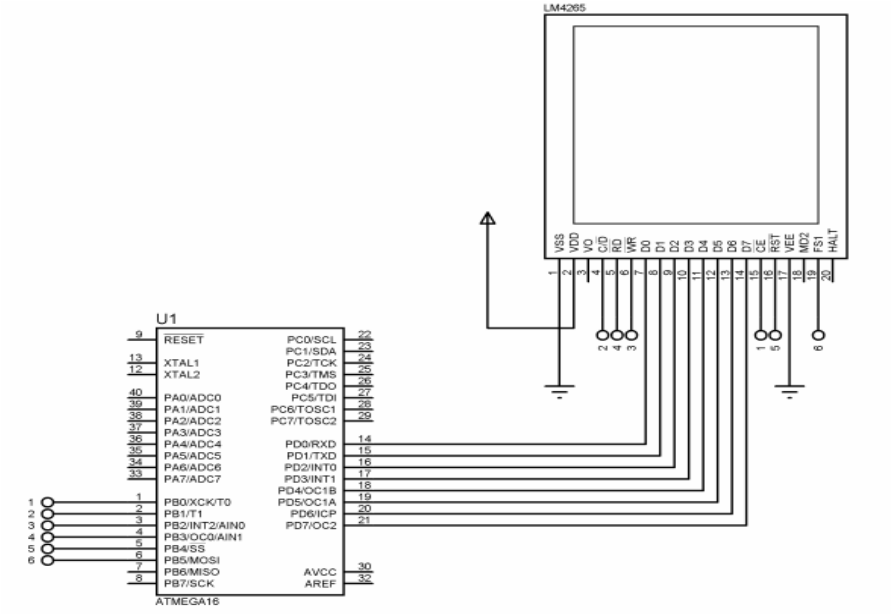
هدف) در این آزمایش می توان اشکال هندسی مورد نظر را روی LCD گرافیکی ایجاد نمود.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Graphlcd = 128 * 128 , Dataport = Portd , Controlport = Portb , _
Ce = 0 , Cd = 1 , Wr = 2 , Rd = 3 , Reset = 4 , Fs = 5 , Mode = 8
Cursor Off
Pset 64 , 100 , 1
```

Line(10 , 5) –(20 , 64) , 255

Circle(64 , 64) , 50 , 255

End



شرح برنامه) در خط سوم برنامه ، LCD گرافیکی را پیکره بندی کرده ایم. در خط ششم ، یک نقطه در مکان مشخص شده روی LCD ایجاد می کنیم. در خط هفتم ، یک خط و نیز در خط هشتم یک دایره با اندازه های دلخواه را روی LCD ایجاد کرده ایم.

آزمایش ۲۸) رسم دوایر متحدالمرکز بر روی LCD گرافیکی:

هدف) در این آزمایش دایره هایی با قطرهای متفاوت به ترتیب از کوچک به بزرگ روی صفحه LCD ظاهر می شوند.

$Regfile = "m16def.dat"

$Crystal = 1000000

Config Graphlcd = 128 * 128 , Dataport = Portd , Controlport = Portb , _

Ce = 0 , Cd = 1 , Wr = 2 , Rd = 3 , Reset = 4 , Fs = 5 , Mode = 8

Dim A As Byte

Do

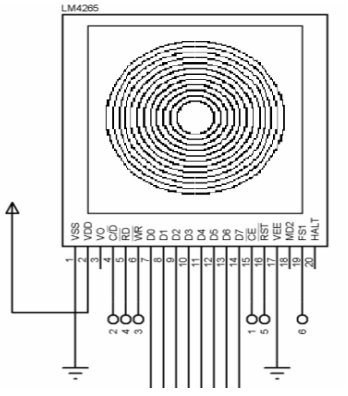
For A = 12 To 60 Step 4

Cursor Off

Circle(64 , 64) , A , 255

If A = 60 Then Cls Graph

Waitms 500



```
Next A
Loop
End
```

شرح برنامه) در این برنامه با استفاده از حلقه For – Next توانسته ایم مکان ایجاد هر دایره را که در واقع همان شعاع دایره است را به ترتیب افزایش دهیم و در متغییر A قرار دهیم ، که نتیجه آن ایجاد دایره هایی با اندازه هـای مختلف حول یک مرکز است.

آزمایش ۲۹) نمایش عکس بر روی LCD گرافیکی:

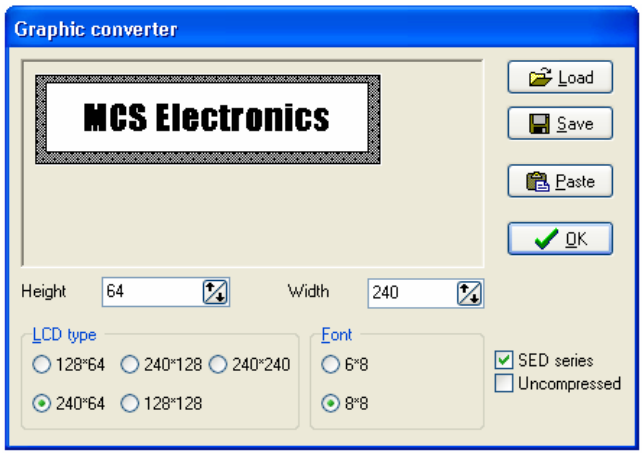
دستورات جدید) Showpic ، $bgf

هدف) در این آزمایش می توان هر عکس دلخواهی را بر روی صفحه LCD گرافیکی نمایش داد.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Graphlcd = 128 * 128 , Dataport = Portd , Controlport = Portb , _
Ce = 0 , Cd = 1 , Wr = 2 , Rd = 3 , Reset = 4 , Fs = 5 , Mode = 8
Do
Cursor Off
Cls
Showpic 0 , 0 , Ax1
Wait 10
Loop
End
Ax1:
$bgf  "Ax.bgf"
```

شرح برنامه) برای نمایش یک عکس بر روی LCD گرافیکی به ترتیب زیر عمل می کنیم:

ابتدا تصویر مربوطه را را در نرم افزارهای ویرایش عکـس (نظیـر ACDSee یـا Photoshop) تغییـر سـایز داده (متناسـب بـا نـوع LCD گرافیکـی) و سـپس آن را بـصورت فـایلی بـا پـسوند Bitmap و در حـال تـک رنـگ (Mono Chrom Bitmap) ذخیـره مـی کنـیم. در نـرم افـزار بیـسکام از منـوی Tools گزینـه Graphic Converter را انتخاب تا پنجره مربوط به آن باز شود و عکس ذخیره شده ایی که تغییر سایز دادیـم را در آنجـا بـا استفاده از گزینه Load بار گذاری می کنیم که با کلیک روی گزینه Save فایل را در فولـدر فایـل هـای کامپایـل شده ذخیره می کنیم. (این فایل با پسوند bgf. ذخیره می شود).

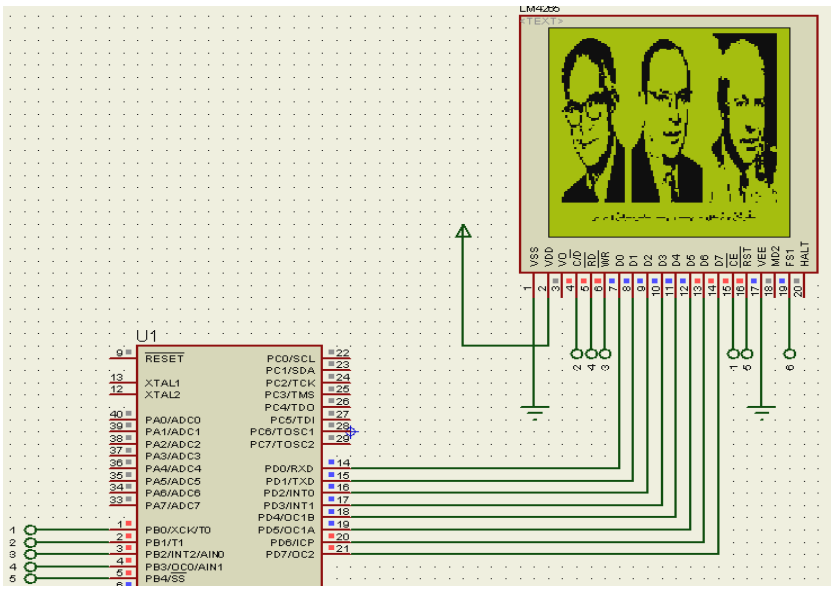


شکل ۱۸-۶ پنجره مربوط به تبدیل عکس به فرمت مورد استفاده در برنامه میکروکنترلر

به این نکته هم توجه شود که در پنجره Graphic Converter حتما باید اندازه سایز LCD درست انتخاب شود و اگر LCD از نوع KS108 بود تیک گزینه SED نیز باید زده شود.

حالا عکس مورد نظر آماده نمایش روی LCD گرافیکی می باشد. در خط هشتم برنامه بالا با نوشتن دستور Showpic 0 , 0 , Ax1 عکس مورد نظر را روی LCD نشان داده می شود. به این نکته مهم توجه شود که AX1 نام لیبل نمایش عکس است و از هر نام دلخواه دیگری نیز می توان استفاده کرد.

در خط سیزدهم نیز نوشتن دستور " $bgf "Ax.bgf اجباریست. در واقع Ax.bgf همان نام فایل عکس مورد نظر ماست که می خواهیم روی LCD نشان داده شود. برای نمایش عکس های مختلف بصورت پشت سر هم روی LCD ، می توانیم از لیبل های مختلف با نام های متفاوت استفاده کنیم.



شکل ۱۹-۶ نمایش عکس توسط LCD گرافیکی

۹ – ۶ معرفی و پیکره بندی UART :

برای برقراری ارتباط و تبادل اطلاعات بین دو یا چند میکروکنترلر و نیز ارتباط بین میکرو با کامپیوتر و سایر ماژول ها می بایست از ارتباط سریال با پروتکل های تعریف شده برای میکرو استفاده کرد.

ارتباط سریال می تواند به دو صورت سنکرون (همزمان) و آسنکرون (غیر همزمان) صورت گیرد. در ارتباط سریال سنکرون علاوه بر ارسال اطلاعات ، پالس ساعت هم برای Match کردن فرستنده و گیرنده از نظر زمانی فرستاده می شود. ولی در ارتباط سریال آسنکرون ، پالس ساعت ارسال نمی شود و به جای آن از مفهوم سرعت انتقال بیت (Baud) استفاده می شود و به عبارت دیگر سرعت انتقال در فرستنده و سرعت دریافت در گیرنده برابر خواهد بود.

Baud سرعت انتقال اطلاعات بوده و واحد آن بیت بر ثانیه (bps یا bit/s) می باشد که این مقدار باید در هر دو طرف (فرستنده و گیرنده) یکسان باشد ، در غیر این صورت در صحت داده اختلال ایجاد می شود. همچنین این در میکرو این عدد با فرکانس کریستال میکرو رابطه مستقیم دارد و باید مضربی از فرکانس کریستال انتخاب شود. به عنوان مثال در صورت کریستال 11.059MHZ انتخاب 9600 برای عدد Baud rate مناسب است.

کلمه UART مخفف عبارت Universal Asynchronous Receiver & Transfer است.

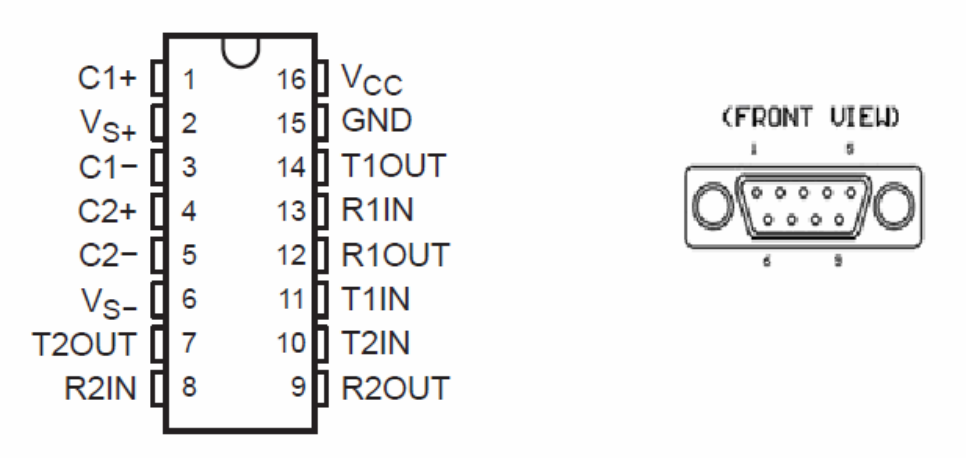
پروتکل UART یکی از راه های ارتباطی سریال میکروکنترلرها با هم و یا ارتباط میکرو با کامپیوتر می باشد. میکرونترلرهای AVR دارای یک ماژول داخلی USTART می باشند که از استاندارد Rs-232 در هر دو حالت سنکرون و آسنکرون پشتیبانی می کند. دسترسی به پورت سریال AVR از طریق سه پین TXD ، RXD ، XCX که به ترتیب پایه ارسال ، دریافت و کلاک پالس می باشد امکان پذیر است. البته توجه کنید که تنها در مد سنکرون پین XCK مورد استفاده قرار می گیرد. قالب اطلاعات در USTART میکرو AVR مانند UART کامپیوترهای شخصی است.

نکته) برای نمایش داده ارسالی و دریافتی در ارتباط سریال Rs-232 بین میکرو و کامپیوتر از محیطی به نام Terminal Emulator در نرم افزار بیسکام استفاده می شود.

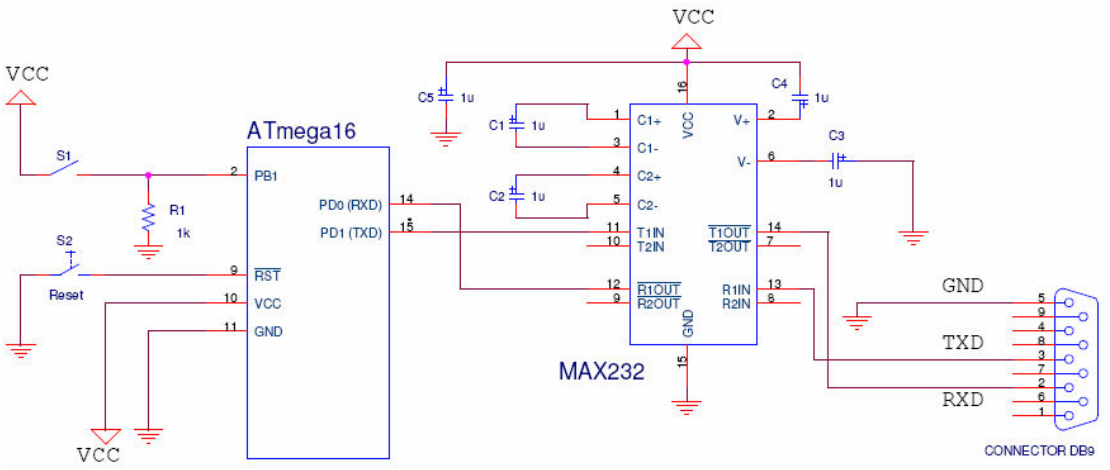
استاندارد Rs-232 یکی از پر کاربردترین استانداردها در کامپیوتر شخصی و کاربردهای صنعتی است و هر دو ارتباط سریال سنکرون و آسنکرون را پشتیبانی کرده و بصورت Full Duplex (ارتباط دو طرفه) عمل می کند. کامپیوترهای شخصی تنها ارتباط آسنکرون را پشتیبانی می کنند و از طریق تراشه پشتیبان کننده UART ، اطلاعات را از حالت موازی به سریال و برعکس تبدیل کرده و با تنظیمات زمانی ، آن را از طریق پورت سریال ارسال

یا دریافت می کند. پورت سریال Rs-232 دارای یک کانکتور 9 پین است که تنها دارای پین های ارسال (TX) و دریافت (RX) به همراه زمین (GND) بوده و مورد استفاده قرار می گیرد. در استاندارد Rs-232 منطق صفر با ولتاژ 3+ تا 12+ و منطق یک با ولتاژ 3- تا 12- نشان داده می شود.

برای اتصال میکروکنترلرها به کانکتور Rs-232 ، حتما می بایست از تراشه MAX232 یا MAX233 استفاده کرد چون منطق Rs-232 با میکروها سازگار نیست و از تراشه های MAX برای تبدیل سطوح ولتاژ استفاده می شود. در واقع کار تراشه های MAX ، تبدیل سطح ولتاژ Rs-232 به سطح ولتاژ TTL (۵ ولت) میکروها می باشد.



شکل ۲۰-۶ آی سی MAX232



شکل ۲۱-۶ مدار مربوط به اتصال Rs-232

نکته) استاندارد Rs-232 برای ارتباط در فواصل کوتاه کاربرد دارد و برای فواصل بیش از ۱۵ متر از استاندارد Rs-485 استفاده می کنیم که عملکرد و پیکره بندی آن مثل Rs-232 می باشد.

در میکروکنترلر AVR پیکره بندی UART به دو صورت سخت افزاری و نرم افزاری قابل انجام است که در حالت سخت افزاری از پایه های استاندارد RDX و TDX استفاده می شود ، ولی در حالت نرم افـزاری ایـن پایـه هـا بـه صورت مجازی توسط کاربر روی هر یک از پایه ها انتخاب می شود.

UART سخت افزاری :

در میکروکنترلرهای AVR دو واحد به نام های UART0 و UART1 داریـم کـه در حالـت پـیش فـرض واحـد UART0 فعال است ولی سایر واحدها مانند UART1 در صورت نیاز باید فعال شوند که بـه کمـک دسـتور زیـر انجام می شود :

Open Com1: "for binary as #1"

برای پیکره بندی سخت افزاری UART داریم :

Config Serialout = Buffered , Size = 20

این پیکره بندی برای ارسال داده سریال استفاده می شود.

Size مشخص کننده تعداد بایت بافر است. حافظه بافر از حافظه SRAM تامین می شود.

Config Serialout = Buffered , Size = 20

این پیکره بندی برای دریافت داده سریال استفاده می شود.

دستورات مربوط به UART سخت افزاری:

۱- دستور $Baud) این دستور سرعت انتقال اطلاعات بوده و یکی از موارد بسیار مهمی است کـه بایـد حتمـا در برنامه ای که از ارتباط سریال استفاده می شود به کار رفته شود.

$Baud = 9600

۲- دستور Print) این دستور مقدار متغییر یا جمله ای را در خروجی پورت سریال میکرو می فرستد.

Print B

۳- دستور Serout) این دستور داده ها را از طریق پورت سریال Rs-232 یا UART میکرو ارسال می کند.

Serout Var , bts , Port , Pin , Baud , Parity , Dbits , Sbits

۴- دستور Serin) این دستور داده های سریال را از طریق پورت UART می خواند.

Serin Var , bts , Port , Pin , Baud , Parity , Dbits , Sbits

Var متغییری که قرار است از طریق UART داده را ارسال یا دریافت کند

Bts تعداد بایت هایی است که ارسال می شود.

Port و **Pin** نام پورت و پایه ای که قرار است استفاده شود.

Baud مقدار Baud Rate را تعیین می کند.

Dbits تعداد بیت های داده را تعیین می کند مه می تواند 7 یا 8 انتخاب شود.

Sbits تعداد بیت های Stop است که می تواند 1 یا 2 انتخاب شود.

Parity تعداد بیت های کد Parity می باشد که در صورت انتخاب 0 به معنی بدون Parity ، انتخاب 1 به معنی Even (زوج) و انتخاب 2 به معنی ODD (فرد) می باشد.

۵- دستور Get) مقدار ورودی UART سخت افزاری یا نرم افزاری را به صورت Byte دریافت می کند و در متغییر مورد نظر قرار میگیرد.

Get #Channal , Var

#Channal شماره کانالی است که فایل باز شده را تعیین می کند و می تواند مقدار متغییر هم باشد.

۶- دستور Waitkey) این دستور تا زمانی که یک کاراکتر از بافر پورت سریال دریافت شود منتظر می ماند و پس از آمدن اولین کاراکتر آن را در متغییر Var قرار می دهد.

A = Waitkey()

۷- دستور Inkey) مقدار کد اسکی اولین کاراکتر دریافتی از پورت سریال نرم افزاری را در متغییر Var قرار می دهد.

B = Inkey()

۸- دستور Input) توسط این دستور ، زمانی کـه در محـیط Terminal Emulator هـستیم ، مـی تـوانیم از صفحه کلید کامپیوتر به عنوان ورودی استفاده کنیم. داده گرفتـه شـده از صـفحه کلیـد در متغییـر Var و متغییـر اختیاری Varn قرار می گیرد.

Input "Amir" , B , A [Noecho]

با استفاده از **Noecho** داده گرفته شده از صفحه کلید در محـیط Terminal Emulator نمـایش داده نخواهـد شد.

آزمایش ۳۰) ارسال و دریافت پیام بین دو میکرو با UART سخت افزاری:

دستورات جدید) Input ، Config Serialin ، Print ، Baud ، Config Serialout

هدف) در این آزمایش توسط دو برنامه مجزا برای هر میکروکنترلر، پیامی را از یک میکرو به میکروی دیگـر ارسـال می کنیم.

برنامه ارسال:

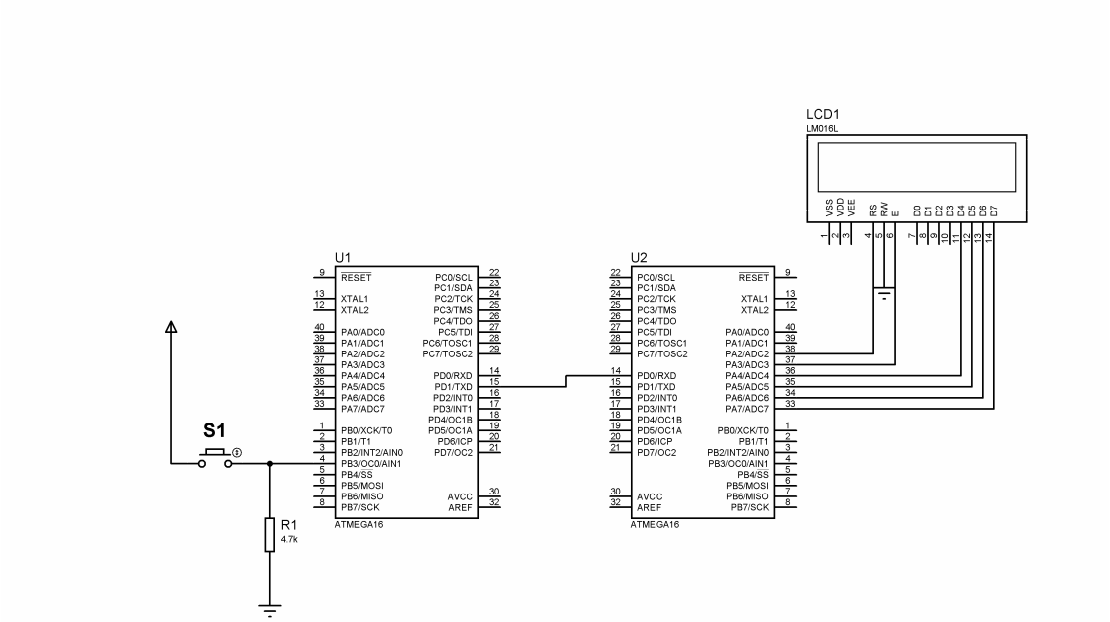
```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Portb.3 = Input
Config Portd.1 = Output
Config Serialout = Buffered , Size = 200
Enable Interrupts
Baud = 2400
Dim B As Byte
Dim S As String * 12
Main:
B = Pinb.3
If B = 0 Then Goto Main
Print "Micro"
Wait 2
S = "Controler"
Print S
Wait 2
S = "AVR"
Print S
```

Wait 2

Goto Main

End

شرح برنامه) در ابتدای این برنامه ، حتما می بایست پین D.1 را به عنوان خروجی تعریف کنیم تا ارسال صورت پذیرد. در خط پنجم برنامه میکرو را در حالت ارسال کننده **(Serialout)** پیکره بندی کرده ایم. در خط یازدهم و دوازدهم برای کلید S1 شرط تعیین کرده ایم که در صورت فشرده شدن پیامها را به ترتیب ارسال کند. در خط سیزدهم با نوشتن دستور Print ، هر عبارتی که در داخل " ..." قرار داشته باشد عینا ارسال می گردد. در خط شانزدهم و نوزدهم نیز عبارات داخل متغییر S توسط میکروکنترلر ارسال می گردند که می توانند کاراکتر ، عدد و یا هر نوع داده ایی باشند. این نکته نیز فراموش نشود که حافظه متغییر در هر دو برنامه ارسال کنده و دریافت کننده حتما باید از نوع رشته ایی (String) تعریف شود.



برنامه دریافت:

$Regfile = "m16def.dat"

$Crystal = 1000000

Config Portd.0 = Input

Config Serialin = Buffered , Size = 200

Config Lcdpin = Pin , Db4 = Porta.4 , Db5 = Porta.5 , Db6 = Porta.6 , _

Db7 = Porta.7 , E = Porta.3 , Rs = Porta.2

Config Lcd = 16 * 2

```
Enable Interrupts
Baud = 2400
Dim X As String * 12
Do
Input X
Cls
Cursor Off
Home
Lcd X
Loop
End
```

شرح برنامه) در ابتدای این برنامه ، حتما می بایست پین D.0 را به عنوان ورودی تعریف کنیم تا دریافت اطلاعات صورت پذیرد. در خط چهارم برنامه میکرو را در حالت دریافت کننده **(Serialin)** پیکره بندی کرده ایم. در خط دوازدهم با نوشتن دستور Input همه اطلاعات دریافت شده داخل متغییر X قرار میگیرد. در خط شانزدهم نیز اطلاعات دریافتی روی LCD نشان داده می شود.

UART نرم افزاری :

برای ایجاد پایه های UART نرم افزاری باید طبق دستور زیر آن را باز کنیم.

Open "device" For Mode As #Channel

Close #Channel

Device برای به کار بردن UART نرم افزاری ، برای کامپایلر باید مشخص باشد که نشان می دهد کاربر کدام پایه را با چه سرعتی و برای چه جهتی (ورودی یا خروجی) برای ارتباط سریال استفاده می کند.

برای پیکره بندی نرم افزاری UART داریم :

Open "COMpin : Speed , Data , Parity , Stopbits [Inverted]" For Mode As #Channel

Pin نام پین مورد استفاده برای کانال سریال می باشد.

Speed برای مقدار Baud می باشد.

Data تعداد بیت های ارسالی بوده و می تواند 7 یا 8 بیت داده باشد.

Parity بیت برابری یا تساوی بوده ، می تواند N برای Non ، O برای Odd (فرد) و E برای Even (زوج) باشد.

Stopbits تعداد بیت های Stop بیت دو بایت ارسالی که می تواند یک یا دو بیت باشد.

Inverted یک پارامتر اختیاری است و می تواند نوشته نشود. با نوشتن این دستور خروجی پورت سریال به صورت معکوس خواهد بود.

Mod برای تععین ارسال یا دریافت اطلاعات می باشد. اگر Output بود برای ارسال (TXD) و اگر Input بود برای دریافت (RXD) اطلاعات استفاده می شود.

Channel شماره کانال سریال باز شده می باشد و توجه داشته باشید کاربر می تواند چندین کانال سریال ورودی یا خروجی در یک میکروکنترلر ایجاد کند.

برای بستن هر کدام از کانال ها می توان از دستور Close #Channel استفاده کرد.

دستورات مربوط به UART نرم افزاری:

این دستورات دقیقا مانند دستورات UART سخت افزاری بوده ، با این تفاوت که در جلوی هر دستور می بایست شماره کانال بصورت #Channel درج شود.

$Baud برای تعیین مقدار Baud Rate در ارتباط سریال نرم افزاری است.

```
$Baud  #Channel , 9600
```

Print برای ارسال داده است.

```
Print #Channel , A
```

Input برای دریافت داده است.

```
Input #Channel , B
```

آزمایش ۳۱) ارسال و دریافت پیام بین دو میکرو با UART نرم افزاری:

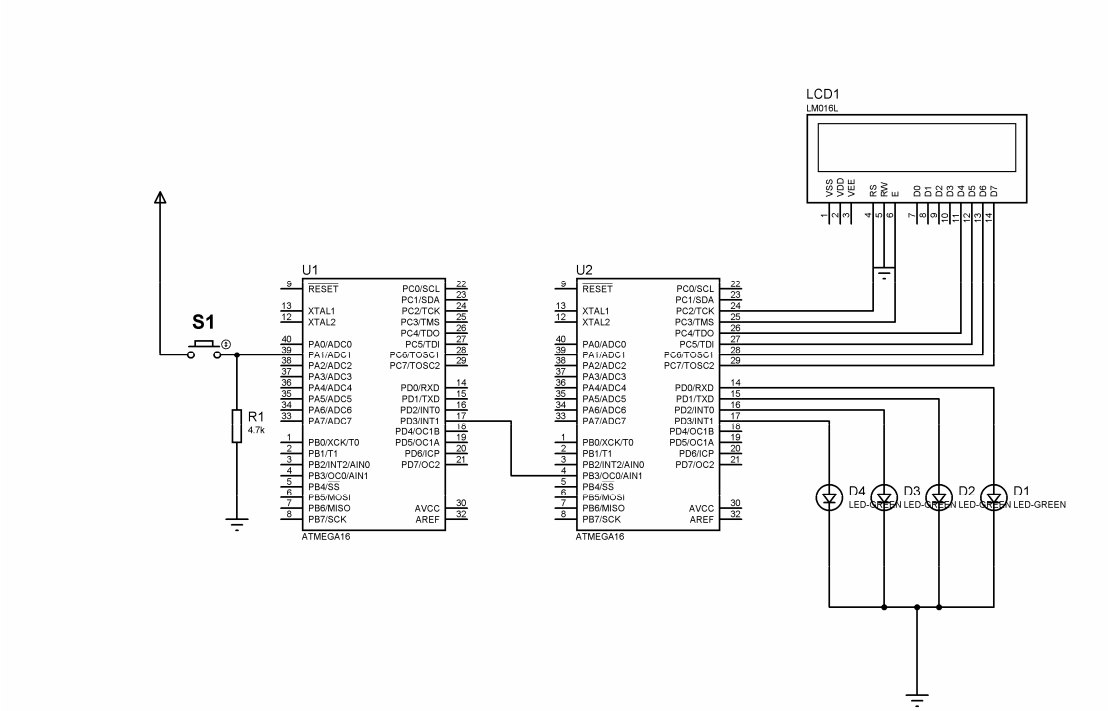
دستورات جدید) Input #1 ، Stop ، Close #1 ، Print #1 ، Open "com

هدف) در این آزمایش نیز مانند آزمایش قبلی پیامهایی از یک میکرو به میکروی دیگری ارسال می شود. البته علاوه بر پیام کدهای باینری نیز ارسال می شود که با توجه به آن LED ها در میکروی گیرنده روشن می شوند.

برنامه ارسال:

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Porta.1 = Input
Config Portd.3 = Output
Dim A As Byte
Dim B As Byte
Dim S As String * 12
Open "comd.3:2400,8,N,1,Inverted" For Output As #1
Main:
B = Pina.1
If B = 0 Then Goto Main
Print #1 , "Micro"
Wait 1
Print #1 , "Controler"
Wait 1
S = "AVR"
Print #1 , S
Wait 1
S = "Start"
Print #1 , S
For A = 0 To 15
Print #1 , A
Wait 1
Next A
S = "Stop"
Print #1 , S
Close #1
Stop
Goto Main
End
```

شرح برنامه) در خط هشتم ، پین D.3 میکرو را بصورت نرم افزاری برای ارسال اطلاعات پیکره بنـدی کـرده ایـم. برای ارسال در این حالت باید از دستور Print #1 استفاده کنیم که تقریبا شبیه حالت سخت افـزاری برنامـه قبلـی است. با دستور Close #1 نیز ارسال اطلاعات تمام و با دستور Stop تبادل قطع می شود.



برنامه دریافت:

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Portb.3 = Input
Config Portd = Output
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , _
Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
Config Lcd = 16 * 2
Dim R As String * 12
Dim H As Byte
Declare Sub Main2
Open "comb.3:2400,8,N,1,Inverted" For Input As #1
Cursor Off
Cls
Home
Lcd "Starting"
```

```
Main1:
Input #1 , R
If R = "Start" Then Call Main2
Cls
Home
Lcd R
Goto Main1
Main2:
Do
Input #1 , R
If R = "Stop" Then
Cls
Home
Lcd "Stop"
Stop
End If
H = ValI
Cls
Home
Lcd H
Portd = H
Loop
Return
End
```

شرح برنامه) در خط یازدهم این برنامه ، پین B.3 میکرو را در حالت نرم افزاری به عنوان گیرنده تعریف کرده ایم. در خط هفدهم برنامه ، میکرو شروع به دریافت اطلاعات می کند و آنها را داخل متغییر R قرار می دهد. لیبل Main2 مربوط به دریافت دیتای شمارنده است. در خط سی و دوم برنامه با دستور Val ، متغییر رشته ایی R تبدیل به متغییر عددی شده و در متغییر H قرار می گیرد. در خط سی و ششم مقدار عددی را روی پورت H قرار می دهیم که نتیجه آن روشن شدن LED با با توجه به مقدار هگزادسیمال 0 تا 15 است.

۱۰ – ۶ معرفی و پیکره بندی SPI :

پروتکل SPI مخفف Serial Peripheral Interface است که توسط شرکت Motorola برای اولین بار اختـراع شد. این شبکه جزء پر سرعت ترین شبکه های سنکرون است که می تواند بین چند AVR یا AVR و تراشه های پشتیبان این شبکه به کار رود. در ارتباط بین دو قطعه که بصورت SPI به یکدیگر متـصل شـده انـد همیـشه یکـی Master (مستر) و دیگری Slave (اسلیو) قرار می گیرد. میکروکنترلرهای AVR بـه هـر دو صـورت مـی تواننـد پیکره بندی شوند. سیستم Master مسئولیت تولید پالس ساعت برای ارتباط با Slave را دارد که باعث هم زمـان سازی بین Master و Slave می شود. یکی دیگر از مزایای مهم آن این است کـه Master و Slave بـه صـورت چرخشی به یکدیگر متصل می شوند ، یعنی اگر از Master داده ای به سمت Slave می رود به طور همزمان مـی تواند از Master به Slave هم داده فرستاد.

پایه های بکار رفته در شبکه SPI به غیر از GND شامل:

MISO (Master In Slave Out) : این خط برای انتقال داده از Slave به Master به کار می رود.

MOSI (Master Out Slave In) : این خط برای انتقال داده از Master به Slave به کار می رود.

SCK (SPI Clock) : این خط برای انتقال کلاک پالس است که توسط Master تولید می شود.

SS (Slave Salect) : این خط برای انتخاب Slave است.

پیکره بندی SPI به دو صورت سخت افزاری و نرم افزاری قابل انجام است.

برای پیکره بندی سخت افزاری SPI داریم :

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 = 1 / 0 , Spiin = Value

Interrupt : در صورت استفاده از وقفه در ارتباط SPI از گزینه On استفاده می شود.

Data Order : در صورت انتخاب LSB ، ابتدا LSB و سپس MSB داده ارسال خواهد شد و برعکس.

Master : اگر میکرویی که در حال برنامه نویسی برای آن هستیم Master باشد گزینه Yes و اگر Slave باشد گزینه No را انتخاب می کنیم.

Polarity : اگر بخواهیم زمانی که SPI در حالت بیکاری (IDEL) است پایه کلاک بالا باشد ، گزینه High انتخاب می شود. انتخاب Low باعث پایین قرار گرفتن پایه کلاک می شود.

Phase : تعداد بیت ارسالی بین دو بایت داده (Stopbit) را مشخص می کند که بهتر است صفر انتخاب شود.

Clock Rate : مشخص کننده فرکانس کلاک SPI می باشد.

Noss : زمانی که در حالت Master نمی خواهیم سیگنال SS ایجاد شود ، عدد 1 را انتخاب می کنیم.

برای پیکره بندی نرم افزاری SPI داریم :

Config Spi = Soft , Din = Pin , Dout = Pin , SS = Pin / None , Clock = Pin , Spin = Value

دستورات مربوط به SPI :

دستور Spiin) این دستور اطلاعات را از روی خط Bus شبکه SPI دریافت و در متغییر مورد نظر می ریزد.

Spiin A , Byte

دستور Spiout) این دستور اطلاعاتی که در متغییر مربوطه قرار دارد را روی خط درگاه SPI ارسال می کند.

Spiout B , Byte

دستور Spimove) توسط این دستور عمل دریافت و ارسال اطلاعات به طور همزمان انجام می شود. (Full Duplex)

A = Spimove B

دستور فوق عدد ثابت یا متغییری که در B قرار دارد را روی شبکه Bus ارسال و داده های دریافتی از شبکه را در A قرار می دهد.

آزمایش ۳۲) ارسال و دریافت پیام بین دو میکرو با پروتکل SPI :

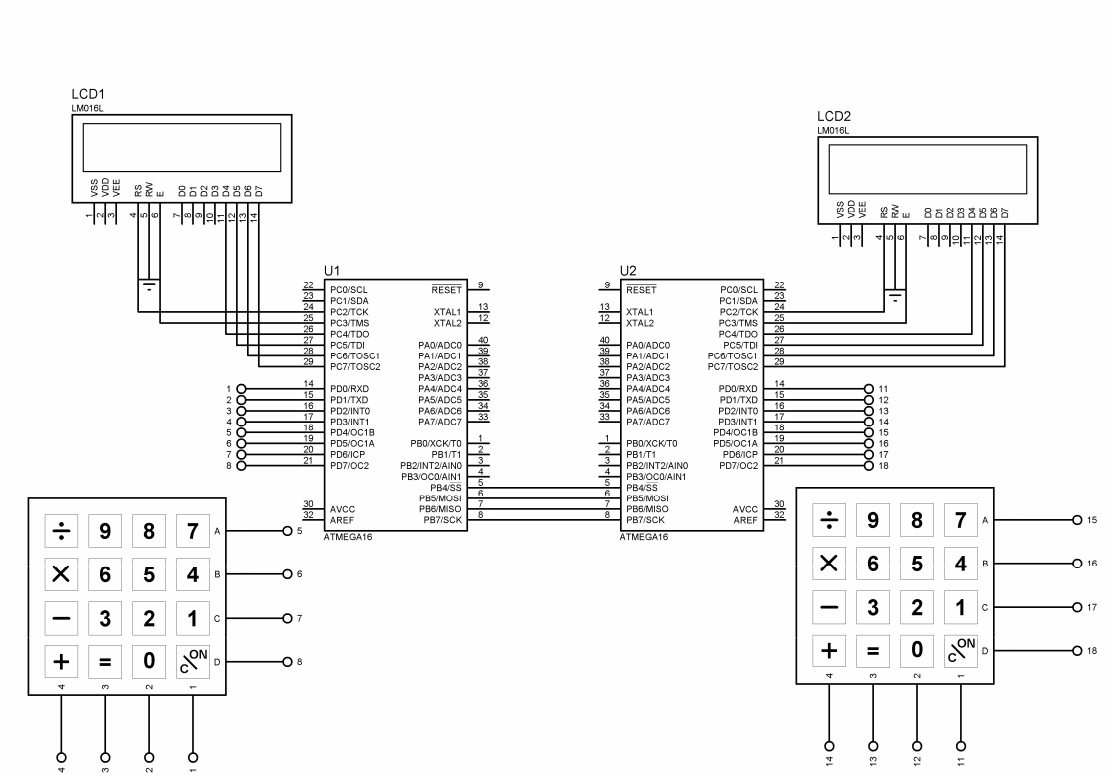
دستورات جدید) Spimove ، Spiinit ، Config Spi

هدف) در این آزمایش با استفاده از پروتکل SPI ، تبادل اطلاعات دو طرفه بین دو میکرو بصورت سنکرون انجام می شود. به عبارت دیگر با فشار دادن کلیدهای کی پد میکروکنترلر Master ، مقدار عدد آن روی صفحه LCD میکروکنترلر Slave نشان داده می شود و نیز با فشردن کلیدهای میکروکنترلر Slave ، عدد مربوط به آن روی صفحه LCD میکروکنترلر Master مشاهده می شود.

برنامه میکروکنترلر Master:

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , _
Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
Config Lcd = 16 * 2
Config Kbd = Portd
Config Spi = Hard , Interrupt = Off , Data Order = Lsb , Master = Yes , _
Polarity = High , Phase = 1 , Clockrate = 128
Dim K As Byte
Dim S1 As Byte
Dim S2 As Byte
Spiinit
Main:
K = Getkbd ( )
If K = 16 Then S2 = 100
S2 = Lookup (k , Dta)
If S2 >= 10 Then S2 = 100
S1 = Spimove (s2)
If  S1 = 100 Then Goto Main
Cls
Home
Lcd S1
Goto Main
Dta:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
```

شرح برنامه) در خط هفتم این برنامه میکروکنترلر را برای پروتکل SPI در حالت Master پیکره بندی می کنیم. در خط دوازدهم دستور Spiinit را داریم که برای ارسال درست داده ها ، معمولا قبل از نوشتن دستورات ارسـال و دریافت استفاده می شود. در خط هجدهم با درج دستور **Spimove** محتوای متغییر S2 که شامل اعداد گرفته شده از کی پد است برای ارسال در متغییر S1 قرار می گیرد و نیز دستورات دریافـت شـده از میکروکنترلـر Slave نیـز توسط همین دستور در همان متغییر S1 قرار می گیرد تا روی LCD قابل نمایش باشد.



برنامه میکروکنترلر Master:

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , _
Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
Config Lcd = 16 * 2
Config Kbd = Portd
Config Spi = Hard , Interrupt = Off , Data Order = Lsb , Master = No , _
Polarity = High , Phase = 1 , Clockrate = 128
Dim K As Byte
```

```
Dim S1 As Byte
Dim S2 As Byte
Spiinit
Main:
K = Getkbd ( )
If K = 16 Then S2 = 100
S2 = Lookup (k , Dta)
If S2 >= 10 Then S2 = 100
S1 = Spimove (s2)
If S1 = 100 Then Goto Main
Cls
Home
Lcd S1
Goto Main
Dta:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
```

شرح برنامه) این برنامه نیز دقیقا عملکردی شبیه به برنامه میکروکنترلر Master دارد با این تفاوت که در خط هفتم SPI را در حالت Slave پیکره بندی کرده ایم.

۱۱ – ۶ معرفی و راه اندازی حافظه EEPROM :

توسط حافظه EEPROM می توان وقایع مختلف را در میکروکنترلر ذخیره کرد و بدون اینکه از بین بروند می توان بارها و باره از آن بهره برد. برای دسترسی به این حافظه می توان از دستورات زیر برای خواندن و نوشتن در EEPROM استفاده نمود.

دستور WriteEeprom) برای نوشتن روی حافظه استفاده می شود.

```
WriteEeprom  Var , Address
```

محتوای متغییر Var در آدرس Address حافظه EEPROM داخلی نوشته می شود. بعد از نوشتن این دستور می بایست مقداری تاخیر در برنامه ایجاد شود تا عملیات نوشتن تکمیل شود. متغییر می تواند بسته به حافظه از نوع داده Byte یا Word باشد و همینطور آدرس می تواند یک عدد باشد.

همچنین می توان برای EEPROM یک متغییر تعریف کرد.

```
Dim Var As Eram Var Type
```

که Var Type می تواند داده های نوع Byte یا Word و ... باشد. و نیز می توانیم داده خود را در آدرسی دلخواه در EEPROM قرار بدهیم.

نکته) می توان از متغییرهای آرایه ایی هم برای EEPROM استفاده نمود.

```
Dim Var(10) as Byte
```

دستور ReadEeprom) برای خواندن از روی حافظه می باشد.

```
ReadEeprom  Var , Address
```

توسط این دستور محتوای EEPROM از آدرس دلخواه Address خوانده می شود و در متغییر Var از نوع داده Byte یا Word ذخیره می شود. آدرس می تواند یک عدد ثابت باشد.

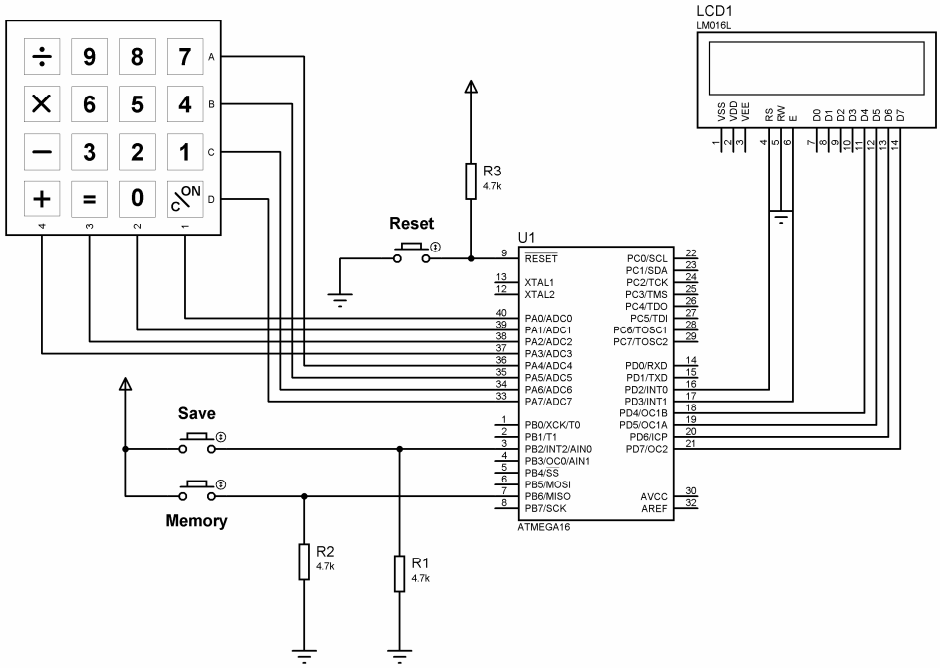
آزمایش ۳۳) ذخیره سازی دائمی یک عدد در حافظه EEPROM :

دستورات جدید) Readeeprom ، Writeeeprom

هدف) در این آزمایش می توان عـدد نـشان داده شـده روی LCD را بـا فـشردن کلیـد Save در حافظـه دائمـی میکروکنترلر ذخیره کرد که با ریست کردن و یا حتی خاموش کردن میکروکنترلر ، عدد ذخیـره شـده از بـین نمـی رود. برای مشاهده کردن عدد ذخیره شده کافیست کلید Memory را فشار دهیم.

```
$Regfile = "m16def.dat"
$Crystal = 1000000
Config Portb = Input
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , _
Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Config Kbd = Porta , Debounce = 50 , Delay = 150
Declare Sub Save
Declare Sub Memory
Dim A As Byte
Dim B As Byte
Main:
Cls
Cursor Off
Lcd "Enter Key"
Do
A = Getkbd ( )
If A <> 16 Then
B = Lookup (a , Cdata)
Cls
Lcd B
End If
If Pinb.2 = 1 Then Call Save
If Pinb.6 = 1 Then Call Memory
Loop
Save:
Cls
Lcd "Save In Memory"
Wait 1
Writeeeprom B , 0
Waitms 4
Cls
```

```
Goto Main
Memory:
Readeeprom B , 0
Cls
Lcd "Memory = " ; B
Return
Cdata:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
```



شرح برنامه) در خط دوازدهم لیبل Main مربوط به عبارت "Enter Key" می باشد که در ابتـدای راه انـدازی مدار ، بر روی صفحه LCD ظاهر می شود. به محض فشردن یکی از کلیـدهای کـی پـد برنامـه بـه حلقـه Do – Loop رفته تا عدد متناظر با آن را نشان دهد. در خط بیست و سوم و بیست و چهارم نیز دو شـرط بـرای فـشردن کلیدهای Save و Memory قرار داده شده است که بـرای انجـام عملیـات مربـوط بـه آنهـا ، لیبـل مـورد نظـر را فراخوانی کند. در خط سی ام ، دستور Writeeeprom را داریم که محتوای متغییر B را در حافظه دائمـی میکـرو (EEPROM) و در آدرس 0 ذخیره می کند. به این نکته هم توجه شود که پس از نوشتن دستور ذخیره سـازی اطلاعات حتما بایستی برای تکمیل عملیات نوشتن در حدود ۴ میلی ثانیه تاخیر ایجاد شود (خط سی و یکم).

در خط سی و پنجم نیز با نوشتن دستور Readeeprom می توان محتوای ذخیره شده داخل متغییر B را بازیـابی نمود و بر روی صفحه LCD نمایش داد.

فصل ۷

معرفی و راه اندازی انواع موتورهای الکتریکی

در میکروکنترلر AVR

موتور الکتریکی وسیله ایی است که انرژی الکتریکی را به انرژی دورانی مکانیکی تبدیل می کند و در انـواع مختلـف طراحی می شود.

الف) موتورهای AC تک فاز و سه فاز

ب) موتورهای DC ، که خود در انواع و اندازه های مختلف تولید می شود و در اینجا به بررسی و طریقه راه انـدازی برخی از آنها می پردازیم.

۱ – ۷ قسمت های تشکیل دهنده هر موتور الکتریکی :

۱- استاتور (قسمت ثابت)

۲- روتور یا شفت (قسمت متحرک)

۳- براشر یا جاروبک (الکترودهای تغذیه)

هر موتور الکتریکی از یک آهنربای الکتریکی دائم ، چند آهنربای الکتریکی برای ایجاد میدان مغناطیسی و سیم پیچ تشکیل شده است ، جریان الکتریکی از طریق دو عدد جاروبک به سیم پیچ ها اعمال شده که در آن ها یـک میـدان الکتریکی بوجود می آورد و این میدان باعث گردش روتور می شود.

۲ – ۷ دو مشخصه مهم هر موتور الکتریکی عبارتند از :

گشتاور : همان قدرت موتور می باشد.

سرعت RPM : تعداد دور موتور در دقیقه می باشد.

۳ – ۷ موقعیت های کاری موتورها :

۱- لحظه راه اندازی موتور ، که در این لحظه موتور چون سلف است جریان راه اندازی شدیدی را مـی کـشد (تـا ۳ برابر جریان نامی مدار) و باعث افت ولتاژ در مدار می شود.

۲- کنترل دور موتور

۳- جهت چرخش (چپ گرد یا راست گرد)

۴- ناحیه ترمز و ایست کامل

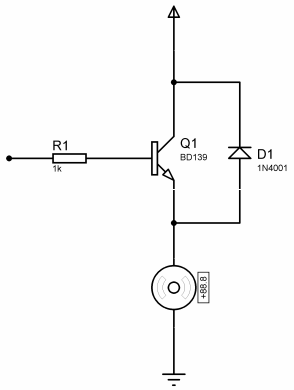
نکته) در موتورهای الکتریکی برای بدست آوردن سرعت یا قدرت مـورد نیـاز ، بـا کمـک چرخدنـده یـا گیـربکس ، تغییراتی را در موتور ایجاد می کنیم تا شرایط مورد نظر فراهم شود.

۴ – ۷ راه اندازی و کنترل موتور الکتریکی DC ساده :

این نوع موتور از یک آهنربای دائم و چند آهنربای الکتریکی تشکیل شده است ، جریان الکتریکی از طریـق دو عـدد جاروبک به سیم پیچ ها اعمال می شود و در آن ها یک میدان بوجود می آورد که این میدان باعث گردش موتور می شود. سرعت موتور DC وابسته به ولتاژ و گشتاور آن وابسته به جریان است.

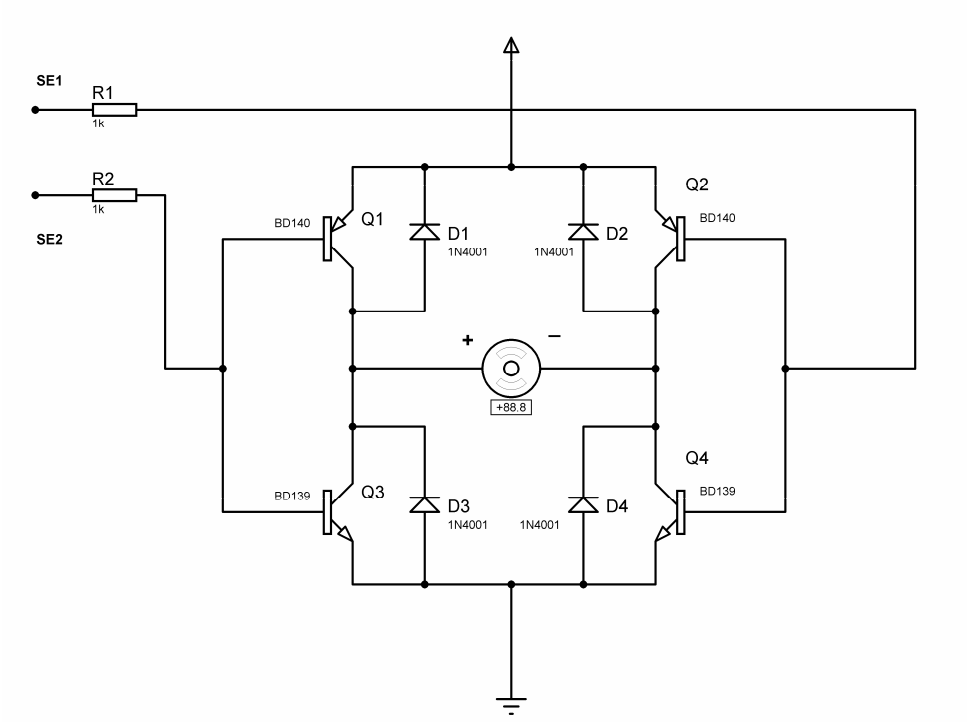
منظور از کنترل و یا راه اندازی موتور ، همان روشن کردن موتور است. این راه اندازی را به دو صـورت تـک سـیمه و دو سیمه (پل H) می توان انجام داد و همینطور برای کنترل سرعت از پالس PWM استفاده می کنیم.

۱- راه اندازی تک سیمه: در این روش یک پایه از میکروکنترلر برای راه اندازی موتور در نظر گرفته می شـود. بـه علت محدودیت جریان دهی پایه های میکروکنترلر ، در خروجی باید از ترانزیستور برای تقویت جریان استفاده کرد.



شکل ۱-۷ راه اندازی تک سیمه

۲- راه اندازی دو سیمه (پل H): در این روش علاوه بر راه اندازی موتور ، می توان جهت چرخش موتـور را نیـز تعیین کرد. برای این کار از یک پل استفاده می شود که توسط ۲ پایه کنترل می شود. در ایـن پـل از ۴ ترانزیـستور استفاده شده است که در حالت کار موتور فقط ۲ ترانزیستور به صورت ضربدری فعال هستند ، مثلا با دادن ولتاژ بـه ورودی SE1 ترانزیستورهای Q1 و Q4 روشن شده و جریان در موتور از سمت + به سمت – حرکت می کند که باعث چرخیدن موتور می شود.



شکل ۲-۷ راه اندازی دو سیمه به روش پل H

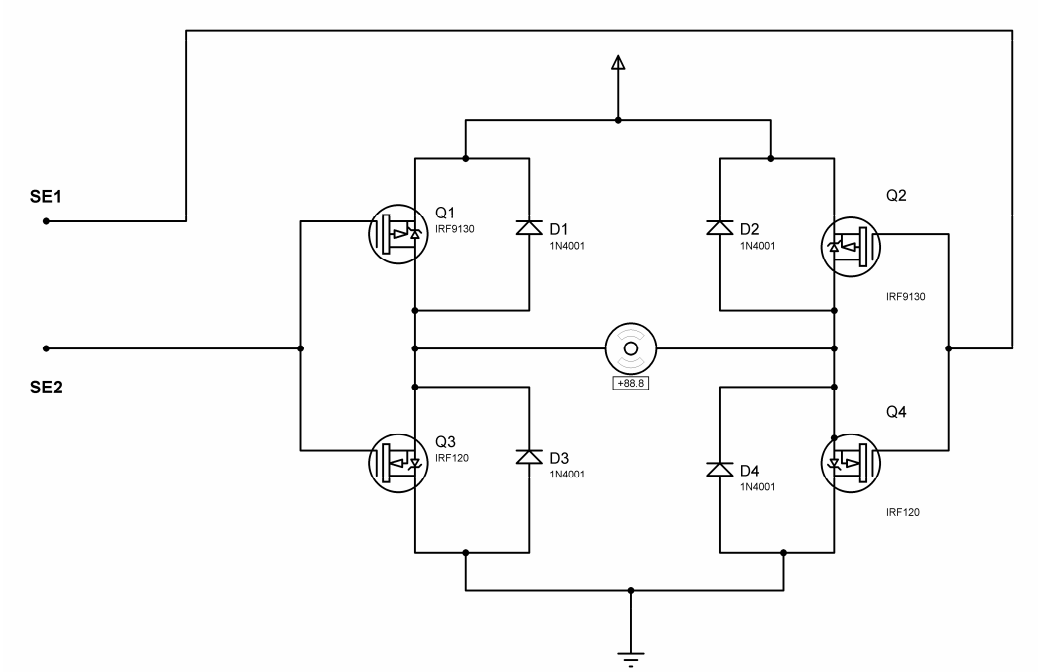
جدول وضعیت کاری پل H بصورت زیر است :

SE2	SE1	حالت	وضعیت موتور
0	0	1	خاموش
0	1	2	راستگرد
1	0	3	چپگرد
1	1	4	غیر مجاز (اتصال کوتاه)

جدول ۱-۷ وضعیت کاری موتور در روش پل H

نکته) در انتخاب ترانزیستورها دقت شود که حتما جریان قابل عبور از آنها مساوی و یـا بزرگتـر از جریـان مـصرفی موتور باشد. مثلا برای جریان های قوی بهتر اسـت از ترانزیـستورهای TIP120 (منفـی) و TIP125 (مثبـت) یـا 2N3055 که از نوع دارلینگتون هستند و دارای قدرت جریان دهی ۵ آمپر می باشند ، استفاده شود. همینطـور بـه منبع تغذیه به کار برده شده برای تامین جریان بالا نیز دقت شود که در اکثر موارد با موازی کردن یک خازن بـزرگ (معمولا بالای 1000 میکروفاراد) با منبع تغذیه ، در زمان راه اندازی موتور افت ولتاژ ناشی از آن کمتر خواهد بود.

توجه) برای راه اندازی موتور با جریان کم میکرو ، می توان از ترانزیـستورهای MOSTEF اسـتفاده کـرد ، چـون تحریک آن ها با ولتاژ است و جریان زیادی از میکرو کشیده نمی شود.

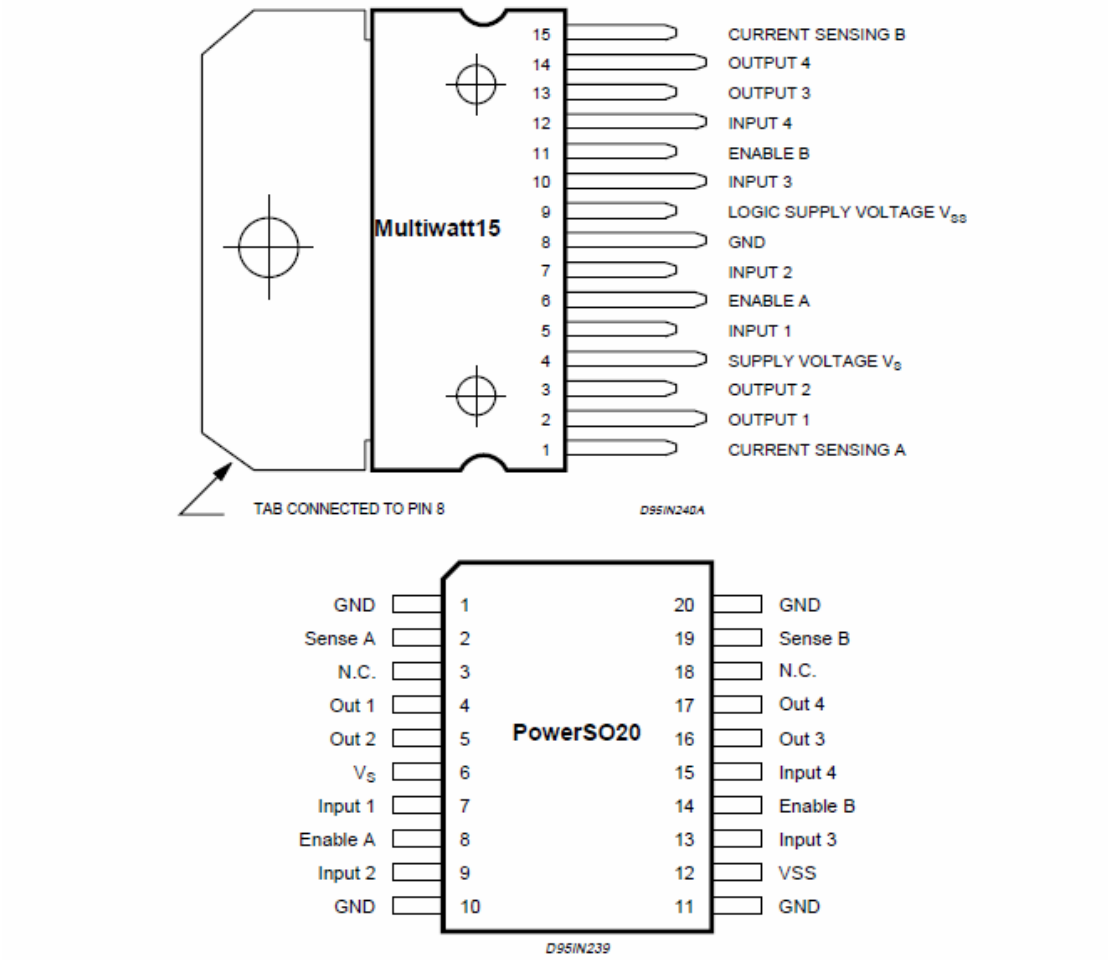


شکل ۳-۷ پل H با ترانزیستورهای Mosfet

۵ – ۷ راه اندازی و کنترل موتور DC با آی سی درایور:

روش دیگر برای راه اندازی موتورهای DC ، استفاده از آی سـی درایـور موتـور مـی باشـد. یکـی از آی سـی هـای پرکاربرد در راه اندازی موتورها ، آی سی L298 بوده کـه دارای قابلیـت هـای فراوانـی در راه انـدازی و کنتـرل دور موتورها می باشد. این آی سی ها قابلیت راه اندازی دو موتور را بطور هم زمان دارا می باشند. ولتـاژ تغذیـه ایـن آی سی بین ۷ تا ۴۷ ولت بوده و بیشترین ولتاژ و جریانی را که برای هر موتور فراهم می کند ، به ترتیب ۴۶ ولت و ۲

آمپر می باشد و دمای بین ۴۰- تا ۱۵۰ درجه سانتی گراد را تحمل می کند. ایـن آی سـی در ۲ نـوع بـسته بنـدی Multiwat15 و PowerSo20 تولید و عرضه شده است.



شکل ۴-۷ آی سی درایور موتور DC

۶ – ۷ معرفی پایه های آی سی L298 :

Logic Supply Voltage (Vss) : پایه تغذیه آی سی بوده که به قطب مثبت منبع تغذیه وصـل مـی شـود و حداقل ۷ ولت برای تغذیه این آی سی نیاز است. برای کاهش نویز بین این پایه و پایه زمین حتما یک خـازن 100 nf می بایست قرار گیرد.

GND : پایه گراند (زمین) آی سی بوده که به قطب منفی تغذیه و پایه گراند موتورها وصل می شود.

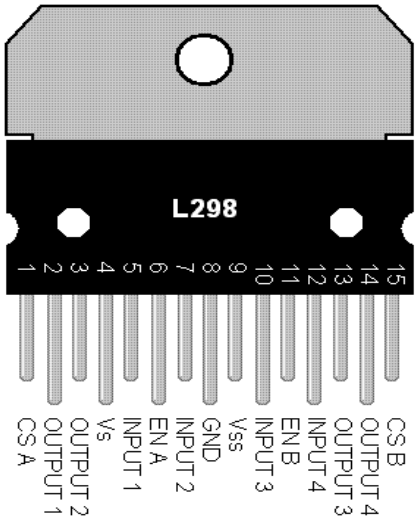
Supply Voltage (Vs) : پایه تغذیه موتورها بوده که بسته به نوع موتور می توان ولتاژ ۱,۵ تـا ۵۰ ولـت را بـه این پایه اعمال کرد.

Enable : پایه های فعال ساز خروجی موتورها می باشد و با دادن ولتاژ بین ۲,۳ تا ۷ ولت به این پایـه هـا موتـور مربوطه فعال می شود.

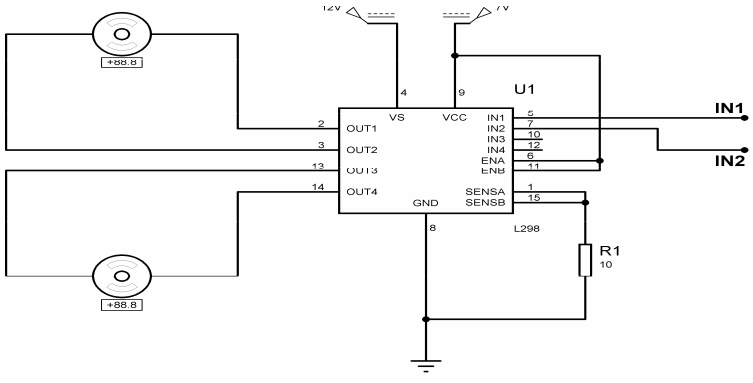
Current Sensing (Sense) : از این پایه ها به عنوان سنسور جریان استفاده می شود و بین هر یک از این دو پایه و زمین یک مقاومت متغییر قرار می گیرد که بوسیله آن جریان بار می توان کنترل کـرد. در صـورتیکه کنتـرل جریان در مدار مهم نیست ، می توان این دو پایه را به زمین وصل کرد تا با حداکثر قدرت کار کند.

Input : پایه های ورودی آی سی بوده که برای کنترل موتورها می باشد و از میکروکنترلـر یـا هـر مـدار کنترلـی دیگری فرمان می گیرد.

Output (Out) : خروجی آی سی بوده که مستقیما به موتورها وصل می شود.

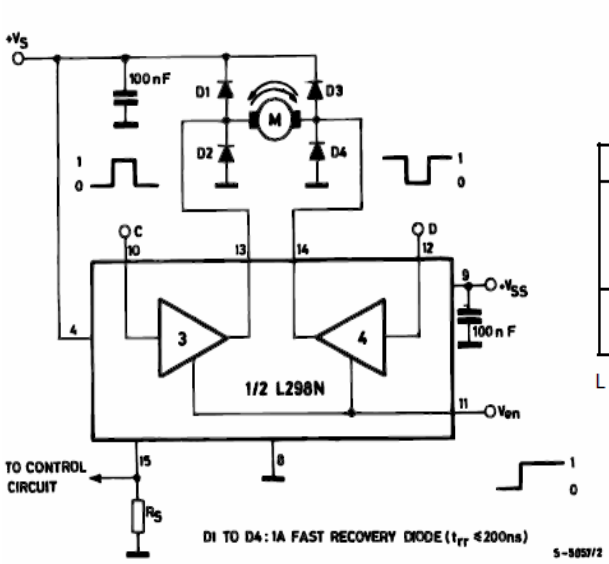


شکل ۵-۷ آی سی L298



شکل ۶-۷ نحوه اتصال موتور DC به آی سی L298

در مدار بالا به این نکته توجه شود که آی سی درایو دیود هرزگرد داخلی ندارد و بـرای خنثـی کـردن ولتـاژ القـایی معکوس موتورها باید از دیود های خارجی استفاده شود ، در غیر اینصورت درایو آسیب می بیند.



Inputs		Function
Ven = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
Ven = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

شکل ۷-۷ نحوه اتصال موتور DC به آی سی L298

۷ – ۷ جدول وضعیت کاری آی سی درایو L298 :

موتور B	وضعیت پایه2	وضعیت پایه1	... ...	موتور A	وضعیت پایه 3	وضعیت پایه 2
متوقف	0	0	... ...	متوقف	0	0
راستگرد	1	0	... ...	راستگرد	1	0
چپگرد	0	1	... ...	چپگرد	0	1
متوقف	1	1	... ...	متوقف	1	1

جدول ۲-۷ وضعیت کاری L298

نکته مهم: بین دو پایه موتورها از خازن های عدسی 104 (100 nf) استفاده کنید ، زیرا :

اولا ، این خازن مانع اثر اینرسی جریانی موتور به آی سی می شود که از سوختن آی سی جلوگیری مـی کنـد (هـر موتور ذاتا ژنراتور نیز می باشد).

ثانیا ، چون موتور بار سلفی – اهمی هست ، این خازن باعث اصلاح ضریب قدرت و در نتیجه کارکرد بهتر آن خواهد شد.

ثالثا ، نویز ایجاد شده در اثر جرقه جاروبک ها را حذف می کند.

۸ – ۷ راه اندازی و کنترل موتورهای پله ایی (Stepper Motor) :

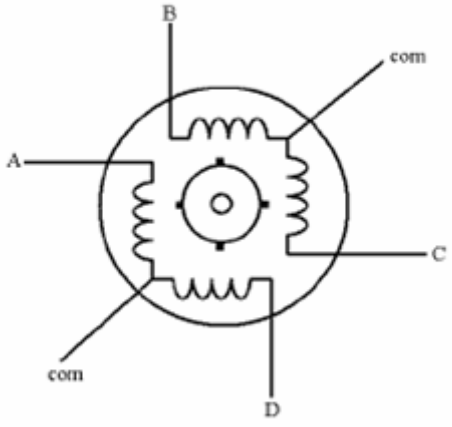
یکی دیگر از انواع موتورهای الکتریکی ، استپر موتور یا موتور پله ایی است که حرکت آن کاملا دقیق و از پیش تعیین شده می باشد و با ارسال بیت های 0 و 1 به سیم پیچ های آن می توان آن را حرکت داد. در داخل استپر موتور یک روتور درونی شامل آهنرباهای دائمی ، توسط یک دسته از آهنرباهای خارجی که به صورت الکترونیکی روشن و خاموش می شوند ، کنترل می شود. این موتور عموما دارای چهار قطب می باشد که سیم پیچ ها روی این چهار قطب قرار می گیرند و با ارسال 0 و 1 به این سیم پیچ ها در واقع میدان مغناطیسی ایجاد می شود که این میدان باعث حرکت روتور مغناطیسی موجود در موتور پله ایی می شود. یک موتور پله ایی ترکیبی از یک موتور DC و یک سلونویید است. موتورهای پله ایی ساده توسط بخشی از یک سیستم دنده ایی در حالت های موقعیتی معینی قرار می گیرند ، اما موتورهای پله ایی نسبتا کنترل شده ، می توانند بسیار آرام بچرخند. معمول ترین موتورهای پله ایی ، چهار سیم استاتور دارند که با سر وسط جفت شده اند و در مجموع تشکیل پنج سیم را می دهد که یک سیم آن مشترک بوده و مخصوص ولتاژ تغذیه است. موتورهای پله ایی کاربردهای فراوانی در روبات ها ، انواع پرینترها ، درایورهای CD ، هاردیسک و ... دارند.

انواع موتورهای پله ایی بصورت زیر است :

۱- موتورهای پله ایی رلوکتانسی متغییر

۲- موتورهای پله ایی مغناطیس دائم

۳- موتورهای پله ایی هیبرید



شکل ۸-۷ ساختمان یک موتور پله ایی ساده

نحوه عمل کرد یک موتور پله ایی ، با موتور DC تفاوت چندانی ندارد. برای راه اندازی این نوع موتور کافی ست به ترتیب به سیم پیچ ها ولتاژ داده شود. البته باید این سیم پیچ ها را به توالی 0 و 1 کرد ، در غیر این صورت موتور به صورت دلخواه نخواهد چرخید.

جهت موتور	D	C	B	A
خلاف جهت عقربه های ساعت	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1

جهت موتور	D	C	B	A
در جهت عقربه های ساعت	0	0	0	1
	0	0	1	0
	0	1	0	0
	1	0	0	0

جدول ۳-۷ جهت و زاویه حرکت استپر موتور

زاویه پله :

هر موتور زاویه حرکت مخصوص به خودش را دارد که به وسیله آن می توان تعداد پالس های که باید به موتور اعمال کنیم تا یک دور کامل بچرخد را تعیین کنیم.

پله در دور = 360 / (زاویه پله)

بعنوان مثال ، یک موتور با زاویه پله 1.8 به 200 پالس برای یک دور چرخیدن نیاز دارد.

360 / 1.8 = 200

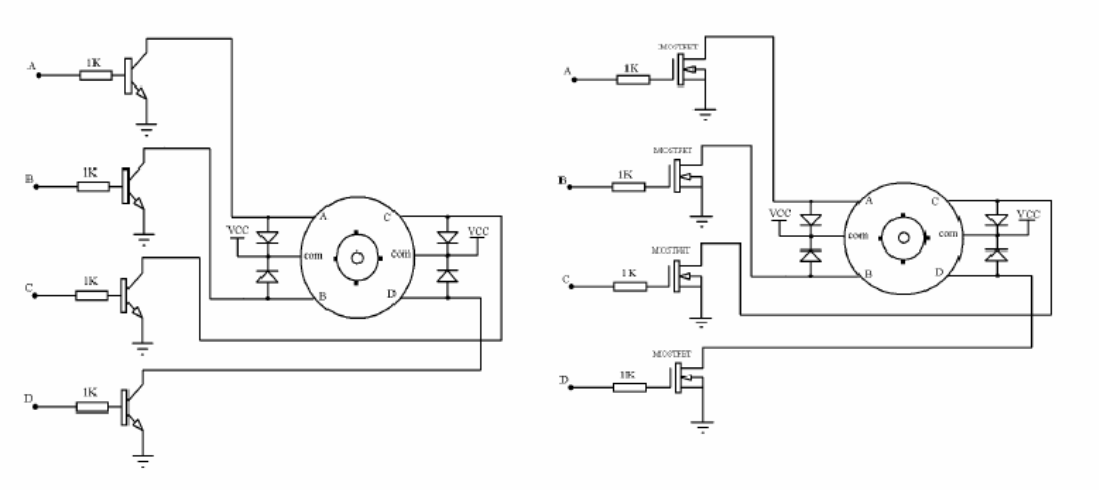
نکته) برای کنترل دقیق تر زاویه حرکت موتور ، می توان آن را بصورت نیم پله راه اندازی کرد ، که در این صورت تعداد پالس ها برای یک دور چرخش کامل دو برابر می شود.

جهت موتور	D	C	B	A
موافق جهت عقربه های ساعت	1	0	0	1
	0	0	0	1
	0	0	1	1
	0	0	1	0
	0	1	1	0
	0	1	0	0
	0	1	0	0
	0	1	1	0
	0	0	1	0
	1	0	1	0
	1	0	0	0
	1	0	0	1

جهت موتور	D	C	B	A
در خلاف جهت عقربه های ساعت	1	0	0	0
	1	0	0	1
	0	1	0	1
	0	1	0	0
	0	1	1	0
	0	0	1	0
	0	0	1	1
	0	0	0	1
	1	0	0	1
	1	0	0	0
	1	1	0	0
	1	1	0	0
	1	1	1	0
	0	0	1	0
	0	0	1	1
	0	0	0	1

جدول ۴-۷ جهت و زاویه حرکت استپر موتور به روش نیم پله

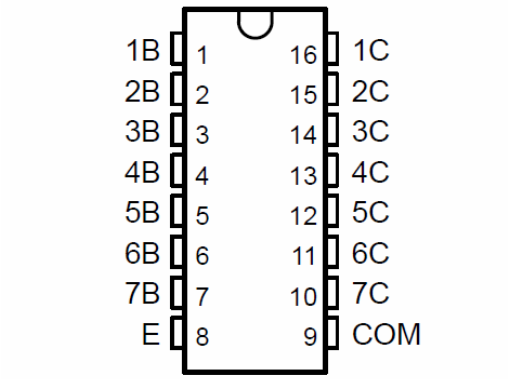
نکته) برای راه اندازی موتورهای پله ایی (با توجه به نیاز جریان کشی بالا) توسط میکروکنترلر بـه راه انـداز نیـاز است که می توان از ترانزیستورها یا آی سی درایو استفاده کرد.



شکل ۹-۷ نحوه درایو کردن استپر موتور

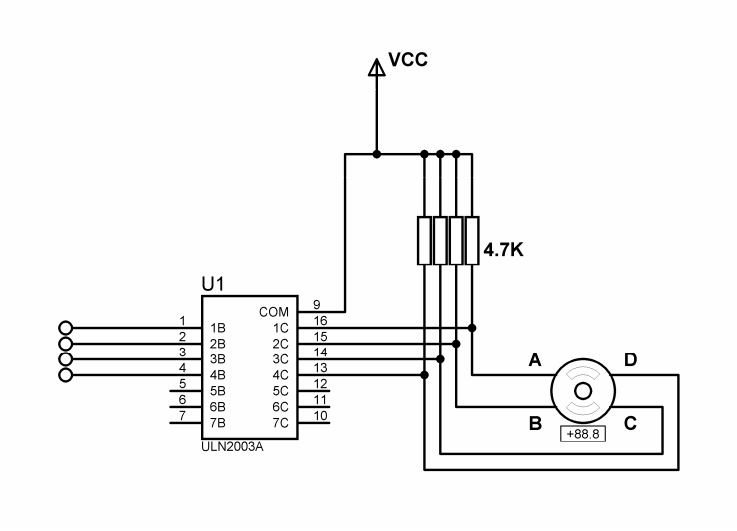
۹ – ۷ راه اندازی و کنترل استپر موتور با آی سی درایو :

یکی از معروف ترین و پرکاربرد ترین آی سی درایو برای راه اندازی استپر موتور ، آی سی ULN2003 می باشد.



شکل ۱۰-۷ آی سی ULN2003

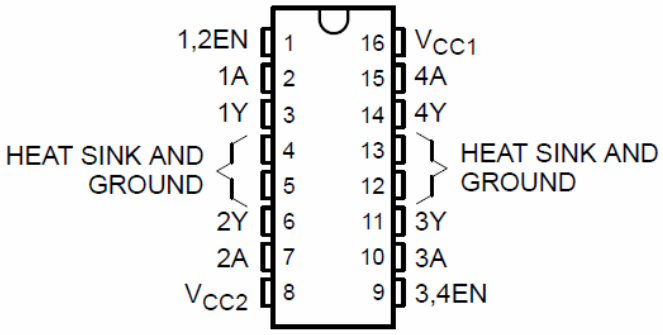
حداکثر جریان خروجی درایو ULN2003 در حدود ۵۰۰ میلی آمپر می باشد و ولتاژ خروجی آن به میـزان تغذیـه موجود بین پایه زمین آی سی و سیم مشترک موتور است و می تواند حداکثر ۵۰ ولت باشـد ، ولتـاژ ورودی آن نیـز حداکثر ۳۰ ولت است و آی سی از ورودی جریان ۲۵ میلی آمپر را به ازای بیشترین ولتاژ می کشد.



شکل ۱۱-۷ درایو کردن استپر موتور با آی سی ULN2003

۱۰ – ۷ معرفی پایه های آی سی L293D :

نوع دیگری از آی سی درایو که برای راه اندازی استپر موتورهای نوع دو قطبی (دو سیمه) استفاده می شود ، تراشه L293D است و شامل دو پل ارتباطی می باشد.



شکل ۱۲-۷ آی سی L293D

EN : پایه های فعال ساز تراشه بوده که باید آنها را به VCC وصل کنیم یا توسط میکروکنترلر بـه آن هـا فرمـان بدهیم.

A : پایه های مخصوص اعمال پالس بوده که به میکروکنترلر وصل می شوند. نحوه اعمال پالس به این پایـه مطـابق جدول زیر است :

Stepper Table	Coil A1	Coil B1	Coil A2	Coil B2
Step 1	ON	ON	OFF	OFF
Step 2	OFF	ON	ON	OFF
Step 3	OFF	OFF	ON	ON
Step 4	ON	OFF	OFF	ON

جدول ۵-۷ حالت های مختلف اعمال پالس برای حرکت استپر موتور

Y : پایه هایی که می بایست به موتورها وصل شوند (تغذیه موتورها) ، در واقع یک موتور به پایه های 3 و 6 و موتور دیگر به پایه های 11 و 14 متصل می شود.

VCC1 : پایه تغذیه خود آی سی می باشد که 5 ولت است.

VCC2 : پایه ولتاژ موتور بوده که معمولا 6 ولت است.

Ground : پایه هایی که به زمین وصل می شوند.

نکته) آی سی SN754410 ، یک نوع دیگر از درایوهای موتور است که مشخصات آن شبیه به آی سی L293D می باشد.

۱۱ – ۷ راه اندازی و کنترل سروو موتورها (Servo Motor) :

سروو موتور ، نوعی موتور پرقدرت است که می تواند حول یک زاویه خاص با دقت بالا بچرخد. سروو موتورها هم مانند موتورهای پله ایی برای تجهیزاتی که نیاز به تعیین دقیق موقعیت روتور (شفت) را داریم ، استفاده می شود. اگر بخواهیم سروو موتور را با موتور پله ایی مقایسه کنیک برتری سروو موتور به موتور پله ایی در گشتاور اولیه راه اندازی و کنترل دقیق تر موقعیت شفت به کمک مدار فیدبک کنترل به وسیله شفت انکدر است که این باعث شده ، استفاده از سروو موتور در صنایع ، پر کاربردتر از موتورهای پله ایی باشد. کاربرد سروو موتورها در بازوی ربات و باز و بسته کردن درب و دریچه و ... می باشد. سروو موتور دارای ۳ سیم می باشد که دو سیم آن برای تغذیه موتور و یکی برای کنترل موتور می باشد. سیگنال هایی که به کمک آن کنترل موقعیت می کنیم ، یک تک پالس با عرض متغییر است که عرض این پالس ۱ تا ۲ میلی ثانیه تغییر می کند.

در صنعت برای استفاده از سروو موتور نیاز به درایورهای مخصوص آن و PLC می باشد. البته سـروو موتورهـایی بـا قدرت کمتر هم وجود دارد که می توانند توسط میکروکنترلر هم کنترل شوند.

شکل ۱۳-۷ سروو موتور

۱۲ – ۷ پیکره بندی سروو موتور در میکروکنترلر AVR :

```
Config Servos = X , Servo1 = PortB.0 , Servo2 = PortB.1 , Reload = RL
```

X : تعداد موتورهای سروو است که می خواهیم توسط AVR کنترل کنیم و اضافه کردن هر موتـور یـک بایـت از فضای SRAM را اشغال می کند.

RL : به ازای بازه زمانی متغییر RL ، ISR داخلی اجرا می شود. مثلا اگر RL=10 انتخاب گـردد ، ISR داخلـی هر ۱۰ میکروثانیه تکرار می شود.

نکته: برای راه اندازی سایر موتورها در میکروکنترلر AVR به جز سروو موتور نیازی به پیکره بندی آن ها نداریم و فقط با Set یا Reset کردن پین های میکروکنترلر می توان به موتورها فرمان روشن یا خاموش را بدهیم.

فصل ۸

معرفی و راه اندازی انواع سنسورها

در میکروکنترلر AVR

۸ – ۱ تعریف سنسور :

سنسور یا حسگر قطعه ایی ست که کمیت های فیزیکی و شیمیایی مانند دما ، نور ، رطوبت ، فشار ، گـاز و... را بـه کمیت های الکتریکی پیوسته (آنالوگ) یا ناپیوسته (دیجیتال) ، تبدیل مـی کنـد. سنـسورها در واقـع کمیـت هـای فیزیکی و شیمیایی را اندازه گیری کرده و آنها را به سیگنال های الکتریکی تبدیل می کند. نوع عملکـرد سنـسورها باعث شده ، بخش جدا نشدنی سیستم های کنترل اتوماتیک و روباتیک واقع شوند.

برخی از انواع پرکاربرد سنسورها که در پروژه های میکروکنترلر به وفور استفاده می شوند :

۱- سنسور نور

۲- سنسور صدا

۳- سنسور رنگ

۴- سنسور فاصله

۵- حرکت و لرزش

۶- سنسور دما

۷- سنسور دود

۸- سنسور گاز

۹- سنسور مادون قرمز(Infrared Receiver)

۱۰- سنسور مغناطیسی

۱۱- سنسور وزن و فشار

۱۲- سنسور مافوق صوت (آلتراسونیک)

و...

۲ – ۸ کاربرد سنسورها:

۱- **شمارش تولید:** سنسورهای القائی، خازنی و نوری

۲ - **کنترل حرکت پارچه و ... :** سنسور نوری و خازنی

۳ - **کنترل سطح مخازن:** سنسور نوری و خازنی و خازنی کنترل سطح

۴ - **تشخیص پارگی ورق:** سنسور نوری

۵ - **کنترل انحراف پارچه:** سنسور نوری و خازنی

۶ - **کنترل تردد:** سنسور نوری

۷ - **اندازه گیری سرعت:** سنسور القائی و خازنی

۸ - **اندازه گیری فاصله قطعه:** سنسور القائی آنالوگ

نکته) برای پیکره بندی سنسورها در میکروکنترلر AVR ، نیاز به وارد کردن دستور خاصی نداریم ، بلکه خروجی سنسور را که معمولا منطق 0 یا 1 است را به یکی از پین های ورودی میکرو داده و آن را توسط برنامه نویسی کنترل می کنیم و یا اگر خروجی سنسور سیگنال های آنالوگ بود ، با استفاده از ADC داخلی میکرو ، آن را کنترل می کنیم.

نکته) برخی از سنسورها به دلیل پیچیده بودن مدارهای جانبی آنها بصورت ماژول (بوردهای کوچک الکترونیکی) در بازار ارائه می شوند.

در ادامه ، دو نوع از پرکاربردترین سنسورها را بررسی می کنیم.

۳ – ۸ سنسورهای گیرنده و فرستنده مادون قرمز(IR) :

اساس کار سنسورهای مادون قرمز بویژه در روباتهای مسیریاب ، ارسال امواج مادون قرمز و دریافت بازتاب آنها از روی یک سطح می باشد. در این کاربرد دیودهای فرستنده و گیرنده مادون قرمز در یک راستا کنار همدیگر به گونه ایی قرار گرفته اند که اگر مقابل آن ها اشیاء سفید یا سیاه قرار گیرند ، امواج منتشر شده توسط فرستنده بر روی

گیرنده بازتاب می شود و مقدار نور مادون قرمز دریافت شده و شدت آن اندازه گیری می شود که با توجه بـه آن در خروجی سنسور گیرنده سیگنالی متناظر با امواج بازتابی رسیده از اشیاء ظاهر می شود و به ورودی میکروکنترلر می رود.

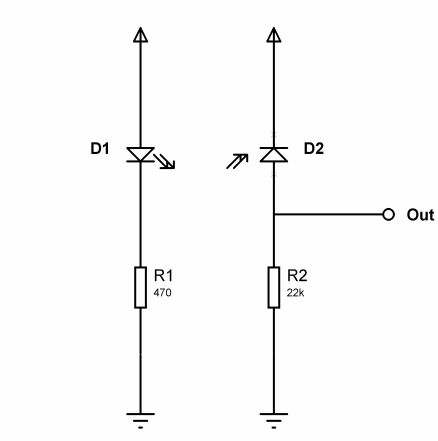
البته میزان بازتاب نور بستگی به رنگ سطح و مانع نیز دارد ، رنگ سیاه کمترین باز تابش و رنگ سفید بیشترین باز تابش را دارد ، رنگ سبز هم حد وسط را دارد.

بیشترین کاربرد این سنسور جهت ارسال سیگنال کدها از کنترل به تلویزیون بصورت دید مستقیم دو دیود فرستنده و گیرنده در مقابل هم است که منجر به تعویض کانال و ... در تلویزیون می شود و نیز در روبات های مسیر یـاب بـه همان روشی که در بالا اشاره شد مورد استفاده قرار می گیرند. نور دیود فرستنده با چشم قابل دیدن نیست ولـی بـا دوربین موبایل می توان آن را مشاهده کرد.

این سنسور، هم بصورت دو دیود جدا ازهم فرستنده (بدون رنگ) و گیرنده (سیاه رنگ) وهم بصورت یک پکیج که هر دو را در کنار هم قرار داده اند ، در بازار وجود دارد.

شکل ۱-۸ LED های مادون قرمز

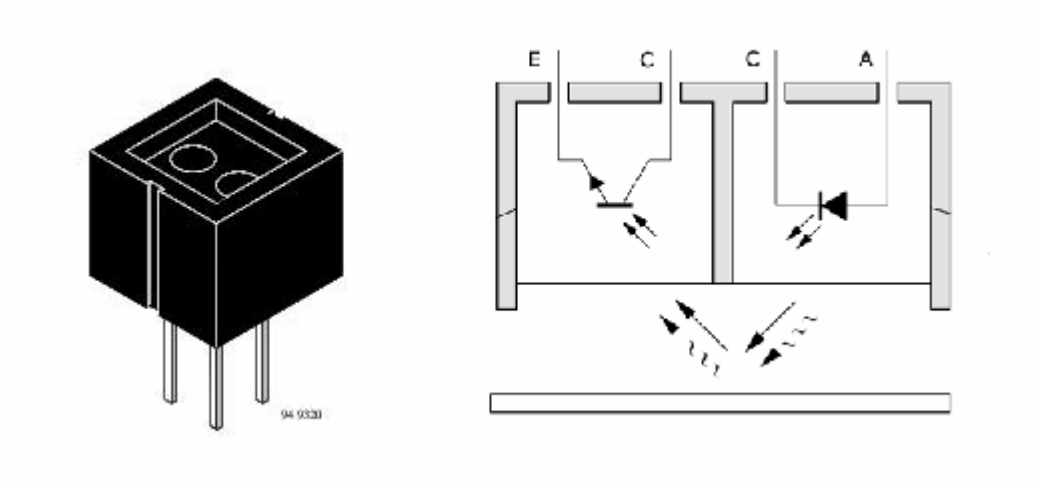
بایاس دیودهای (مادون قرمز) گیرنده و فرستنده بصورت زیر است :



شکل ۲-۸ مدار بایاس LED های مادون قرمز

نکته) بایاس دیود فرستنده مستقیم و بایاس دیود گیرنده معکوس می باشد.

معروف ترین سنسور IR (مادون قرمز) که بصورت پکیج به بازار عرضه شده ، سنسور CNY70 می باشد و به شکل زیر است :

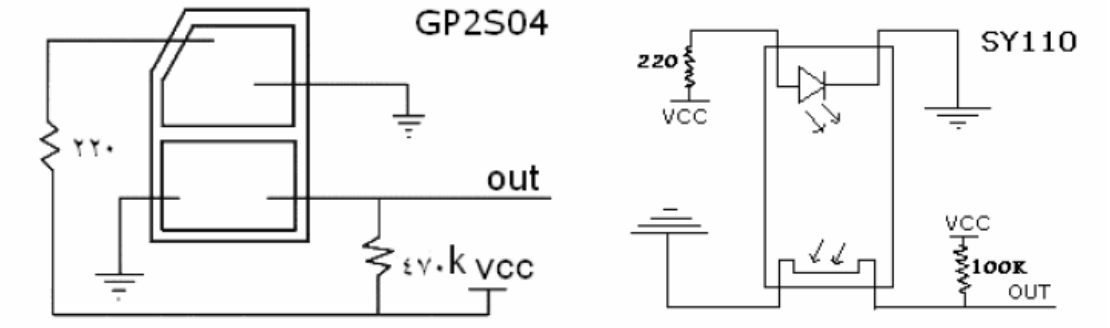


شکل ۳-۸ پکیج IR به شماره CNY70

بایاس این نوع سنسور نیز مانند حالت قبل می باشد. یعنی پایه های C به زمین وصل شده ، پایه A بوسیله یک مقاومت ۳۳۰ یا ۴۷۰ اهم به قطب مثبت و پایه E نیز توسط یک مقاومت 10K به قطب مثبت تغذیه اتصال می یابد. در ضمن از محل بین اتصال پایه E و مقاومت 10K خروجی سنسور گیرنده می باشد.

نکته) پکیج های IR دیگری نیز در بازار وجود دارند که از نظر عملکرد و طریقه بایاس ، دقیقا شبیه به سنسور CNY70 می باشند ، مانند :

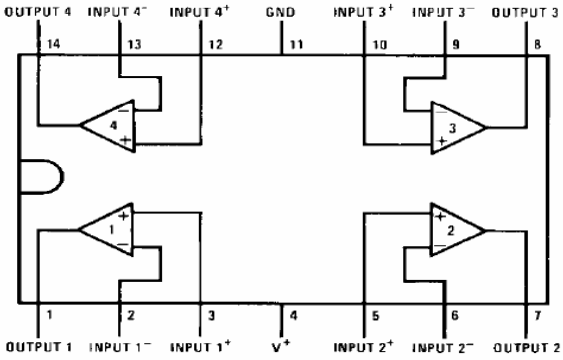
ON2179 ، GP2S04 ، SY110 و ...



شکل ۴-۸ پکیج های مختلف IR

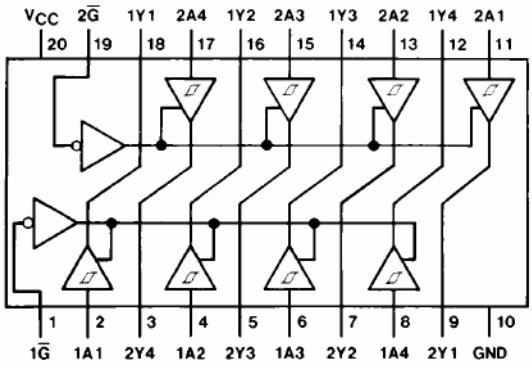
نکته) خروجی سنسور گیرنده مادون قرمز را می توانیم مستقیما به میکروکنترلر وصل کنیم ، ولی برای اینکه بتوانیم بطور دقیق سیگنال دریافتی از سنسور مادون قرمز را به منطق 0 و 1 تبدیل کنیم ، می بایست از یک آی سی بافر یا OP AMP استفاده کنیم.

یکی از آی سی های پرکاربرد برای این کار ، آی سی LM324 بوده که در داخل آن ۴ عدد آپ امپ دارد و ورودی آنالوگ را با یک ولتاژ مرجع مقایسه می کند و به صفر و یک برای میکرو تبدیل می کند.



شکل ۵-۸ آی سی LM324

و همینطور یک آی سی بافر مناسب برای تقویت سیگنال های دریافتی از IR ، آی سی 74244 است.



شکل ۶-۸ آی سی 74244

۴ – ۸ سنسور اندازه گیری دما :

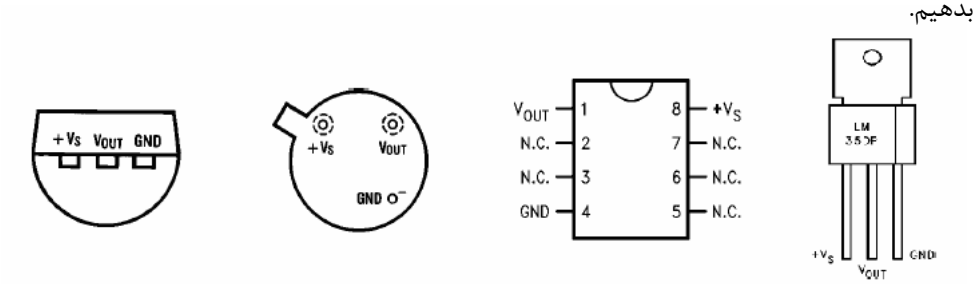
یکی از معروف ترین نوع سنسورهای دما ، سنسور LM35 بوده که دارای سه پایه می باشد. ولتاژ خروجی این سنسور به صورت خطی و متناسب با دمای محیط تغییر می کند ، به طوری که به ازای هر درجه سانتی گراد ، ولتاژ

خروجی آن ۱۰ میلی ولت افزایش می یابد. مثلا در دمای ۳۵ درجه سانتی گراد ، ولتاژ خروجی سنسور به صورت زیر است :

35 * 10mV = 0.35 V

رنج دمای قابل اندازه گیری توسط این سنسور از 55- تا 150+ درجه سانتیگراد می باشد.

برای اندازه گیری دما توسط میکروکنترلر می بایست خروجی سنسور LM35 را به یکی از کانال های ADC میکرو بدهیم.



شکل ۷-۸ انواع پکیج های مختلف سنسور LM35

۵ – ۸ معرفی برخی دیگر از سنسورهای پرکاربرد :

Smt160 سنسور دما با خروجی دیجیتال

Pir – dz035 سنسور تشخیص انسان

LHI648 سنسور حرارتی حساس به بدن

LHI 944 سنسورتشخیص حرکت (انسان و حیوان)

MBF200 سنسور اثر انگشت

RFM-100 سنسور کارت مغناطیسی RFID

LM3915 سنسور صدا

MPXM2202 سنسور فشار

Cmps03 سنسور قطب نما

ADXL – 200 سنسور شتاب ، شیب ، سرعت و فاصله

GR500 سنسور وزن

Tsl230 سنسور تشخیص رنگ

Gp2s04 سنسور تشخیص سیاه و سفید

Tsl2550t سنسور تجزیه نور

Gs209 سنسور تشخیص فلزات

HAS 400-S سنسور اندازه گیری جریان

ACS712 سنسور اندازه گیری جریانAC , DC

MQ-2 سنسور تشخیص دود

MQ-4 سنسور تشخیص گاز متان

MQ-5 سنسور تشخیص گاز

MQ-9 سنسور تشخیص گاز مونوکسید کربن

EGO سنسور اکسیژن

Tgs4161 سنسور تشخیص دی اکسید کربن

Ss1118 سنسور اکسیژن

Ke-25 سنسور اکسیژن

Mlx90614 سنسور دمای راه دور (مادون قرمز)

O2A سنسور رطوبت و دما در یک پک با خروجی دیجیتال

Rhu-207 سنسور رطوبت با خروجی مقاومتی

S2H سنسور رطوبت مقاومتی

HS1101 سنسور رطوبت با خروجی خازنی

3610 سنسور رطوبت با خروجی ولتاژ dc

SHT11 سنسور رطوبت با خروجی دیجیتال

SHT75 سنسور رطوبت با خروجی دیجیتال

فصل ۹

بررسی سایر دستورات در

نرم افزار بیسکام

۱- پیکره بندی کیبورد کامپیوتر:

Config Keyboard = Pin X , Y , Data = Pin X , Y , Keydata = Table

۲- پیکره بندی ماوس کامپیوتر:

Config Ps2emu = Int , Data = Data , Clock = Clock

۳- پیکره بندی مقایسه کننده آنالوگ :

Config ACI = On / Off , Compare = On / Off , Trigger = Toggle / Rising / Falling

۴- پیکره بندی تایمر Wathdog :

Config Watchdog = Time

۵- پیکره بندی ارتباط I2C (ارتباط با دو سیم) :

Config I2c Delay = X

۶- پیکره بندی ارتباط 1Wire (ارتباط با یک سیم) :

Config 1Wire = Pin

۷- پیکره بندی ساعت RTC :

Config Clock = Soft [Gosub = Sectic]

$Date متغییر داخلی ساعت بوده که تاریخ را در خود نگه می دارد.

$Time متغییر داخلی ساعت بوده که زمان را بر حسب ساعت ، دقیقه و ثانیه در خود نگه می دارد.

۸- پیکره بندی نمایش تاریخ :

```
Config  Date = DMY , Separator = Char
```

۹- پیکره بندی RC (محاسبه ثابت زمانی مقاومت و خازن) :

```
Var = Getrc ( Pin , Number )
```

مقدار Var حتما می بایست از جنس Word باشد.

فصل ۱۰

پروژه های کاربردی

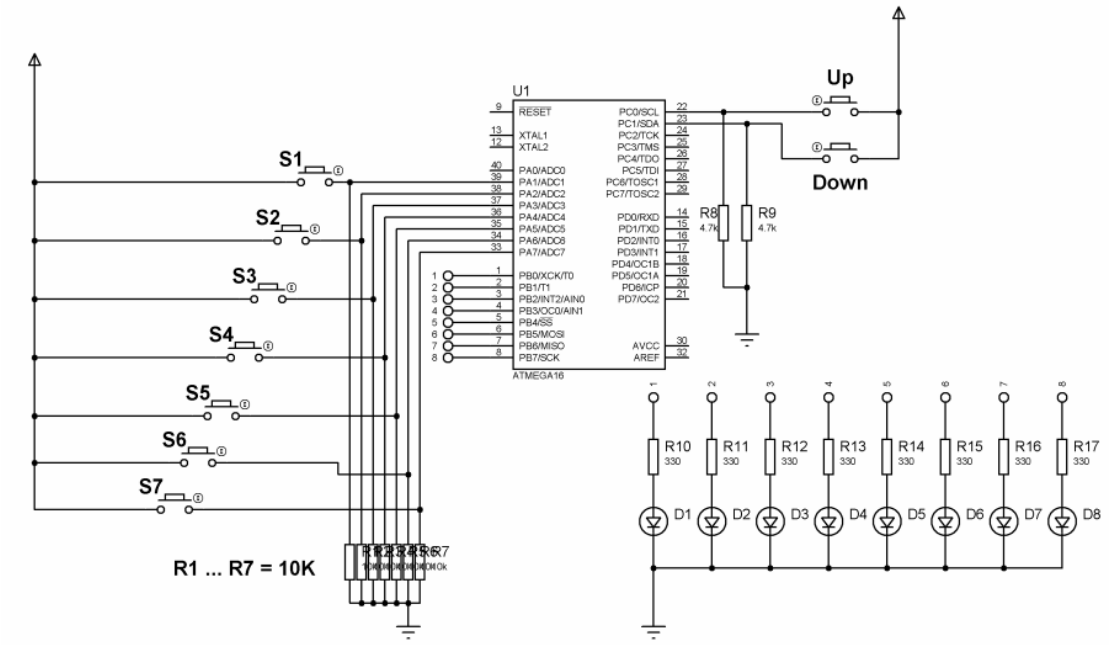
در این بخش به بررسی چند پروژه کاربردی و جالب میکروکنترلر AVR ، با توجه به دستورات و نکاتی که در فصل های قبل به آن اشاره شد می پردازیم.

در هر پروژه ابتدا توضیحات مختصری در مورد عملکرد آن داده شده و سپس در ادامه به بررسی الگوریتم و نقشه مربوط به آن می پردازیم. این نکته لازم به ذکر است که به دلیل طولانی بودن برنامه هر یک از این پروژه ها ، از آوردن آن ها در این قسمت خودداری کرده ایم و برای دسترسی به آن می بایست به فایل های ضمیمه این مجموعه در فولدر Projects ، قسمت ۲ ، مراجعه شود.

۱ - فلاشر

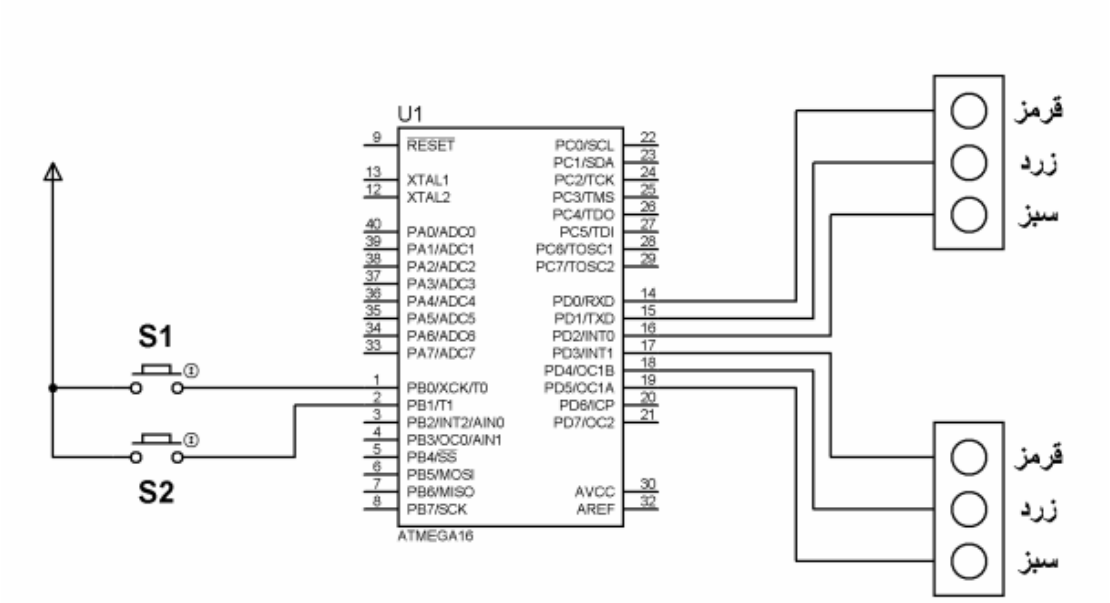
این پروژه یک فلاشر هفت برنامه ایی با قابلیت تنظیم سرعت می باشد. کلیدهای S1 تا S7 برای اجرای برنامه های مختلف می باشد که با فشردن هر یک از آنها فلاشر یکی از آنها را اجرا می کند . با فشردن کلید Up سرعت اجرای برنامه بیشتر و با فشردن کلید Down سرعت اجرا کمتر می شود.

در این پروژه از ۹ زیر برنامه استفاده شده که زیر برنامه Main0 مربوط به ایجاد حال انتظار برای فشردن هر یک از کلیدهای S1 تا S7 برای اجرای هر یک از برنامه ها می باشد و زیربرنامه Main9 نیز برای این است که اگر در حین اجرای یکی از برنامه ها کلید دیگری فشرده شود برنامه هم با توجه به آن تغییر کند. مابقی زیربرنامه ها به ترتیب مربوط به برنامه هر یک از کلیدهای S1 تا S7 می باشند.



۲ - چراغ راهنما

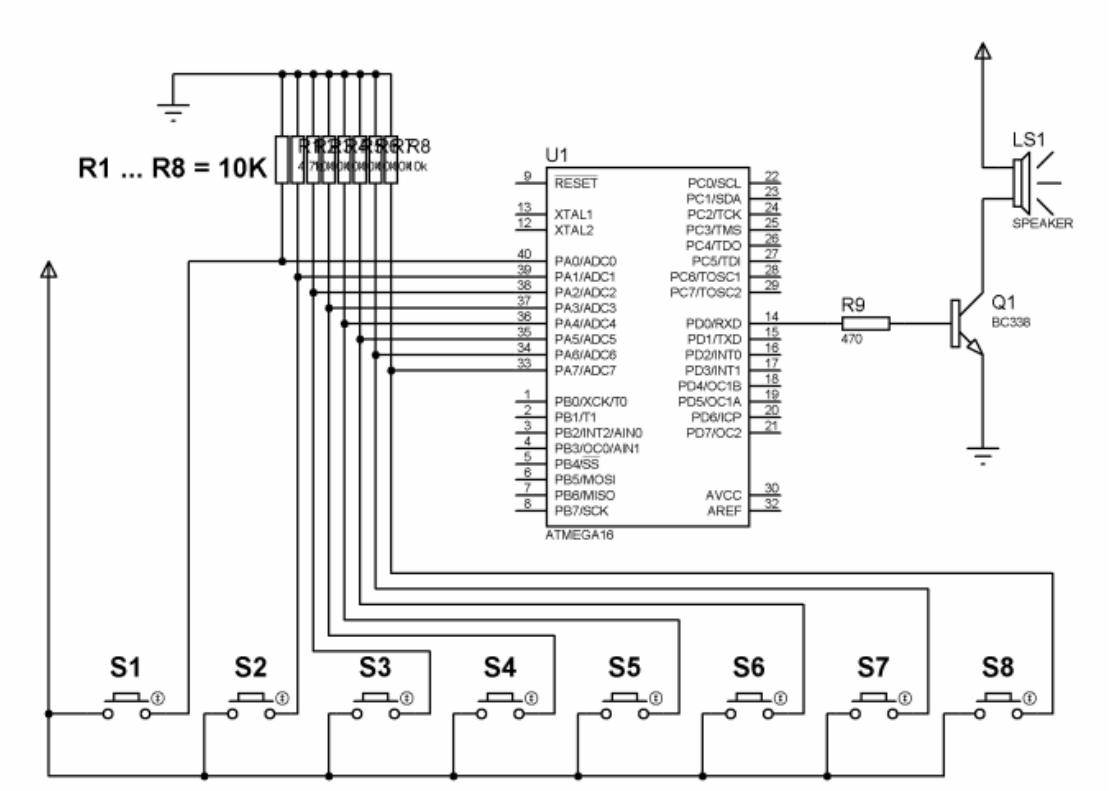
برای این پروژه دو حالت مختلف تعریف شده که حالت اول با فشردن کلید S1 اجـرا مـی شـود و آن اجـرای حالـت چشمکزن برای چراغ راهنما است و معمولا در ساعات کم ترافیک از آن استفاده می شود. بـا فـشردن کلیـد S2 بـه حالت اصلی عملکرد چراغ راهنما وارد می شویم یعنی در ابتدا چراغ یک خیابان سبز بـوده و چـراغ خیابـان دیگـری قرمز می باشد این حالت برای ۸ ثانیه باقی می ماند و سپس چراغی که سـبز بـود زرد مـی شـود ولـی چـراغ قرمـز همچنان در همان حالت باقی می ماند. پس از ۳ ثانیه چراغ زرد قرمز شده و چراغی که قرمز مانده بود سبز می شود و این عمل به طور مرتب تکرار می شود. در برنامه این پـروژه زیربرنامـه Main1 مربـوط بـه کلیـد S1 و زیربرنامـه Main2 مربوط به کلید S2 می باشد.



۳ - ارگ و موسیقی

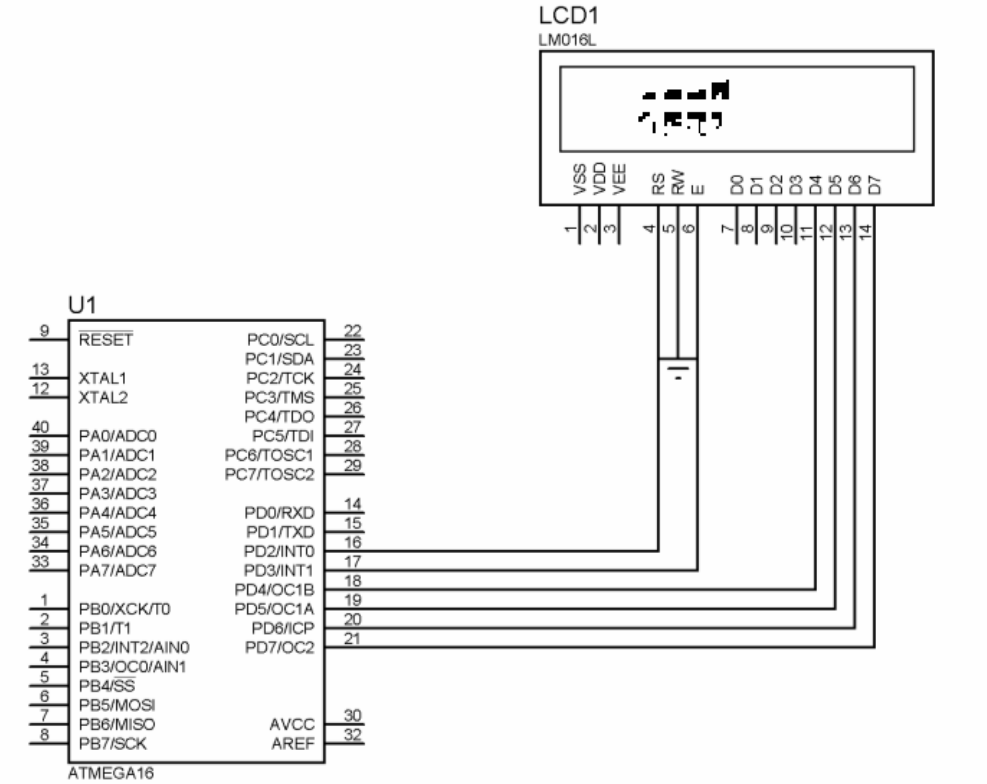
در این پروژه با فشردن هر یک از کلیدهای S1 تا S7 صدای نت های متلفی از اسپیکر شنیده می شود که در برنامه آن با توجه به فشرده شدن هر یک از کلیدها فرکانس خاصی برای آن تعریف شده است.

البته دو برنامه دیگر در فولدر همین پروژه به نام های noname2 و noname3 قرار دارد که اولی مربوط به پخش یک موسیقی و دومی مربوط به پخش صدای آزیر پلیس است و این کار با تغییر سریع و متوالی فرکانس ها صورت گرفته است. آنها را نیز می توان با همین مدار اجرا نمود.



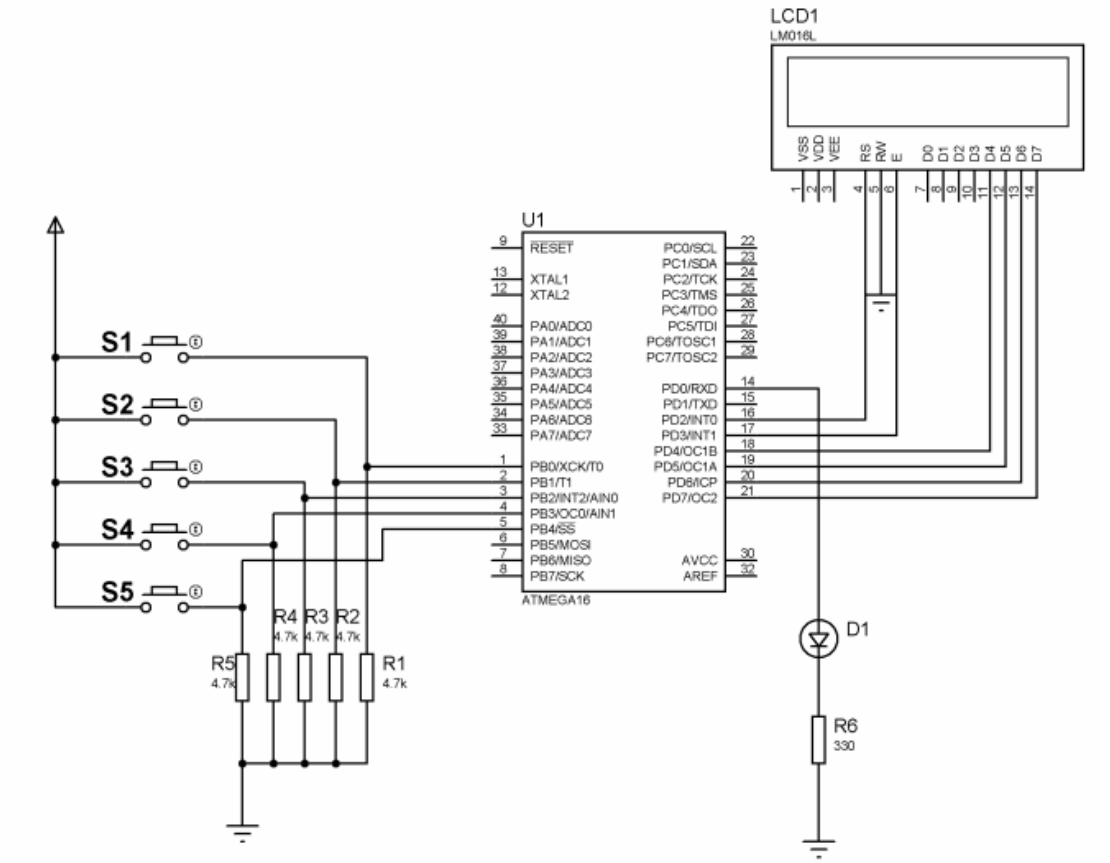
۴ - اجرای انیمیشن روی LCD

در این پروژه با استفاده از ابزار LCD Designer نرم افزار بیسکام طرح یک اسب بر روی LCD کاراکتری پیاده سازی شده است که در حال دویدن است. البته برای ایجاد حالت های مختلف اسب از سه زیربرنامه (Animat1 و Animat2 و Animat3) استفاده شده و نیز برای حرکت اسب بصورت افقی بر روی LCD از حلقه For – Next استفاده کرده ایم.



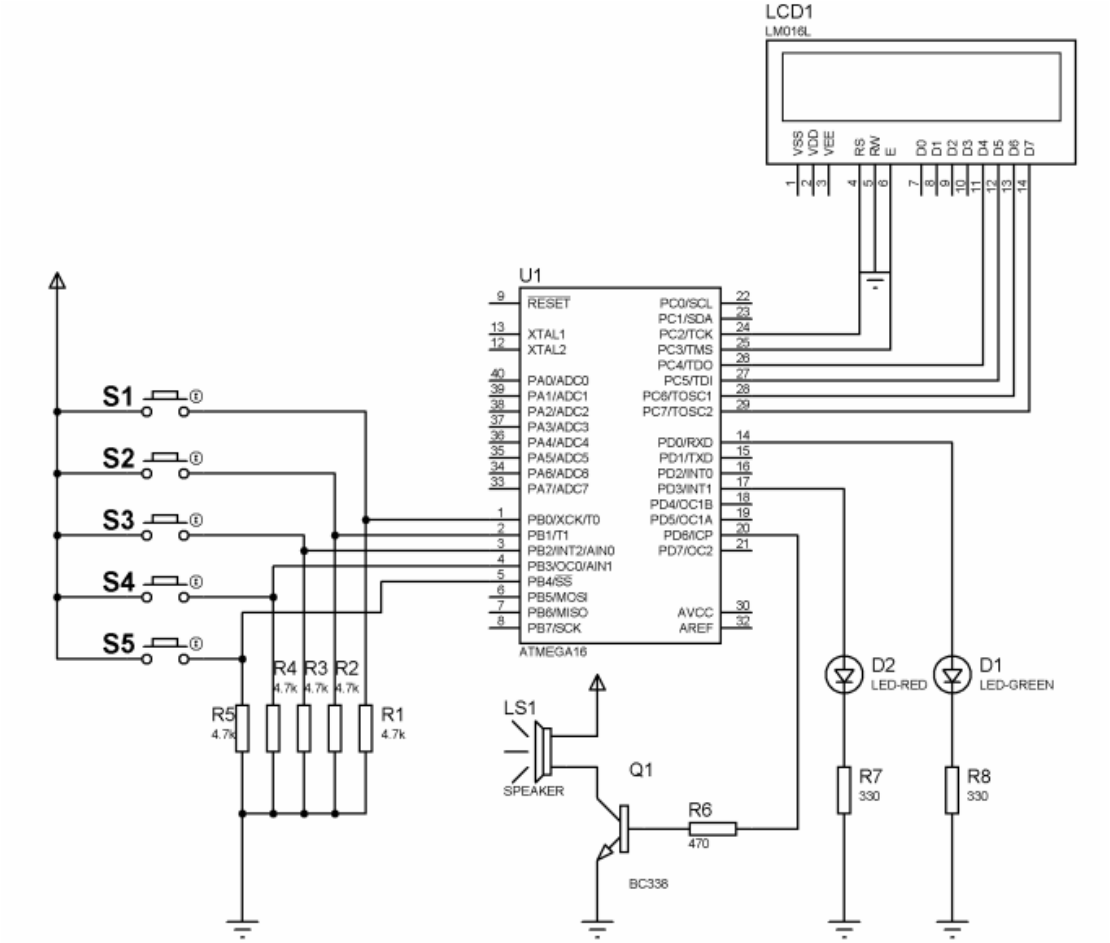
۵ - ساعت دیجیتال

این پروژه یک ساعت دیجیتالی بوده که به محض روشن شدن از ما مقدار دقیقه و ساعت را می خواهد و می بایست با کلیدهای S1 و S2 که مربوط دقیقه هستند و نیز کلیدهای S3 و S4 که مربوط به ساعت هستند تنظیم شود ، سپس با فشردن کلید S5 تنظیمات ثبت شده و ساعت فعالیت خود را شروع می کند. در برنامه این پروژه سه متغییر به نام های S (برای ثانیه) ، M (برای دقیقه) ، و H (برای ساعت) تعریف شده که می بایست در هر ثانیه یک واحد به متغییر S افزوده شود و زمانی که به عدد ۶۰ رسید متغییر S صفر شده و یک واحد به متغییر M افزوده گردد و همین عمل برای متغییر M هم تعریف شده یعنی زمانی که به عدد ۶۰ رسید مقدار متغییر M صفر شده و یک واحد به مقدار متغییر H اضافه می شود. متغییر H نیز باید تا عدد ۲۴ یعنی ساعت ۲۴ پر شود و پس از آن صفر گردد. البته لازم به ذکر است که این ساعت زیاد دقیق نیست برای همین امر در برنامه نویسی به جای درج یک ثانیه در مقابل دستور Wait مقدار ۷۵۵ میلی ثانیه را قرار داده ایم تا کمی به زمان واقعی نزدیکتر شویم.



۶ - تایمر معکوس

تایمر معکوس معمولا زمانی استفاده می شوند که نیاز باشد وسیله خاصی را در مدت زمان معینی روشن کرد و از این لحاظ عملکردی شبیه به ساعت دارند با این تفاوت که عکس آن عمل می کند یعنی در هر ثانیه یک واحد از متغییرها کم می کند تا به صفر برسد. با پایان یافتن شمارش تایمر ، دیود نورانی شماره ۲ (قرمز)روشن می شود که نشانگر روشن کردن یک وسیله خاص است البته در ۱۰ ثانیه باقی مانده به صفر شدن تایمر آلارم نیز از اسپیکر پخش می شود. تنظیمات این پروژه نیز دقیقا شبیه تنظیمات ساعت است. این تایمر قابلیت تنظیم بین یک دقیقه تا ۲۵۶ ساعت را دارد.



۷ - دماسنج

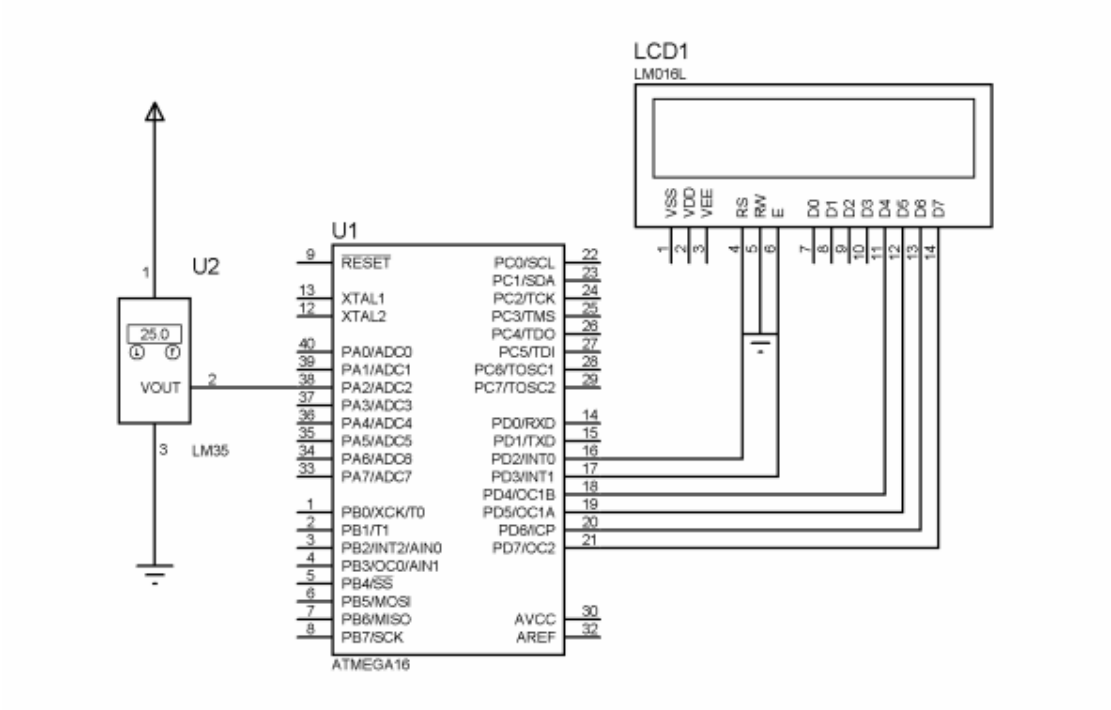
با این پروژه می توان مقدار دمای محیط را (از صفر تا ۹۹ درجه سانتیگراد) ، اندازه گیری نمود. سنسور LM35 با توجه به دمای محیط ولتاژی خاص را روی پایه خروجی خود ایجاد می کند که به ورودی ADC میکروکنترلر داده و پس از تبدیل شدن به عدد دیجیتال (توسط برنامه نوشته شده) روی صفحه LCD قابل نمایش است. در واقع به ازای افزایش یک درجه سانتیگراد ، ولتاژ خروجی LM35 ، ۱۰ میلی ولت افزایش پیدا می کند. از طرفی ADC میکروکنترلر ۱۰ بیتی است (یعنی ۱۰۲۴) و با توجه به ولتاژ رفرانس ۵ ولتی تعریف شده برای میکرو داریم:

$5 / 1024 = 0.005$ V

پس با اعمال 0.25 ولت به ورودی ADC عدد دیجیتال به صورت زیر می شود :

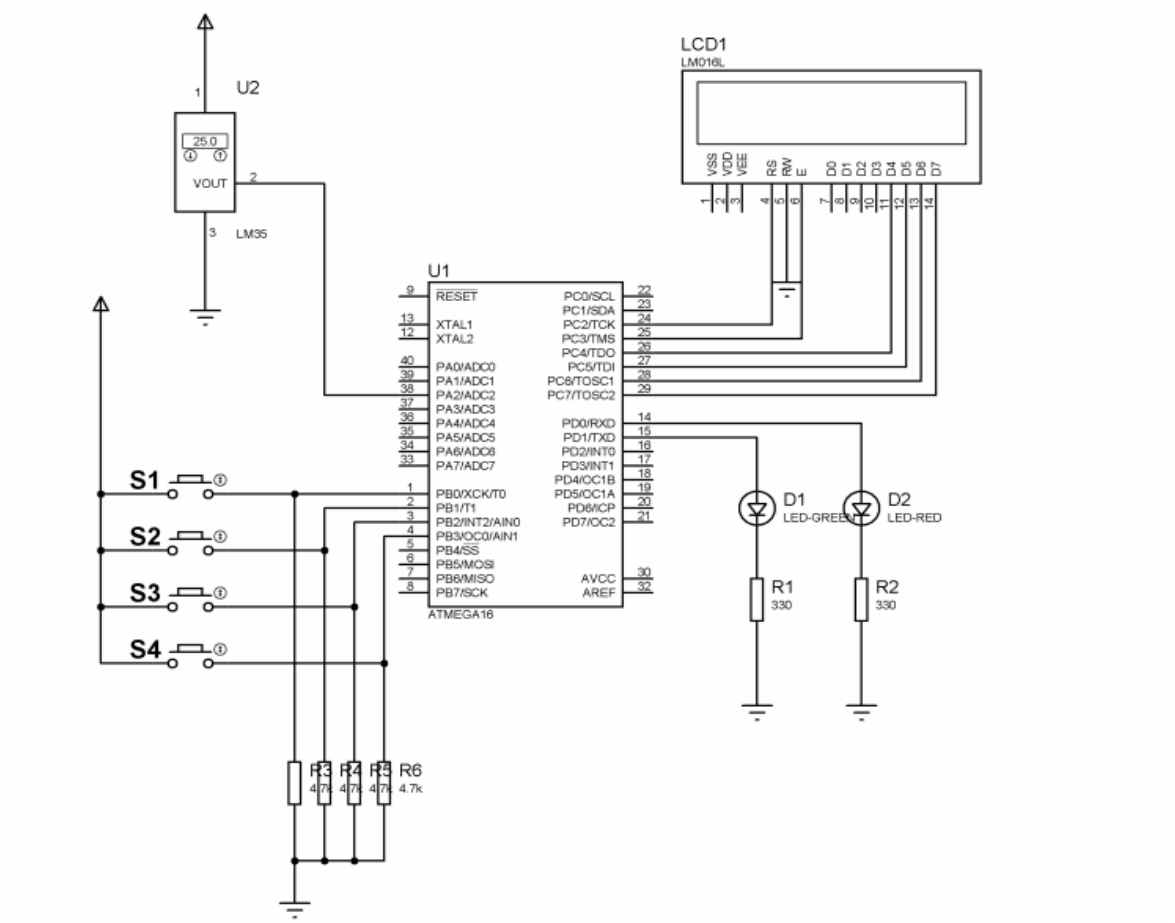
$0.25 / 0.005 = 50$

در نهایت برای بدست آوردن مقدار واقعی دما می بایست عدد را بر ۲ (و یا ۴) تقسیم کنیم.



۸ - ترموستات

این پروژه در جایی استفاده می شود که بخواهیم در یک دمای خاص وسیله ایی را روشن و خاموش کنیم یا به اصطلاح آن را کنترل نماییم. در اینجا نیز می توانیم با تعیین دو دمای مینیمم و ماکزیمم رنجی خاصی را تعریف کرده تا در صورت افزایش دما از مقدار تعریف شده LED قرمز روشن شود و اگر دما از یک حد خاصی کمتر شد LED سبز رنگ روشن شود. حد بالا و پایین دمای پیش فرض به ترتیب ۳۰ و ۱۰ درجه سانتیگراد است که می توان با کلیدهای موجود آن را تغییر داد. عملکرد این پروژه بسیار شبیه به عملکرد دماسنج می باشد.



۹ - اهم متر دیجیتال

در این پروژه قادر خواهیم بود مقدار مقاومت R2 را بصورت تقریبی روی LCD مشاهده کنیم. کلیـدهای S1 و S2 برای تغییر رنج اهمتر از اهم به کیلواهم و بلعکس می باشند. در این مدار از تقسیم مقاومتی برای ایـن کـار اسـتفاده شده که اگر به جای مقاومت R2 هر مقدار مقاومتی را قرار دهیم مقدار آن توسط میکروکنترلر اندازه گیری شـده و روی LCD نشان داده می شود.

برای محاسبه مقدار R2 کافیست ابتدا جریان عبوری از مقاومت R1 را به روش زیر بدست آورد :

$$I = (5 - VR2) = R1$$

در ضمن چون ADC میکرو ۱۰ بیتی است و با توجه به ولتاژ مرجع تعریف شده ۵ ولت برای میکرو داریم :

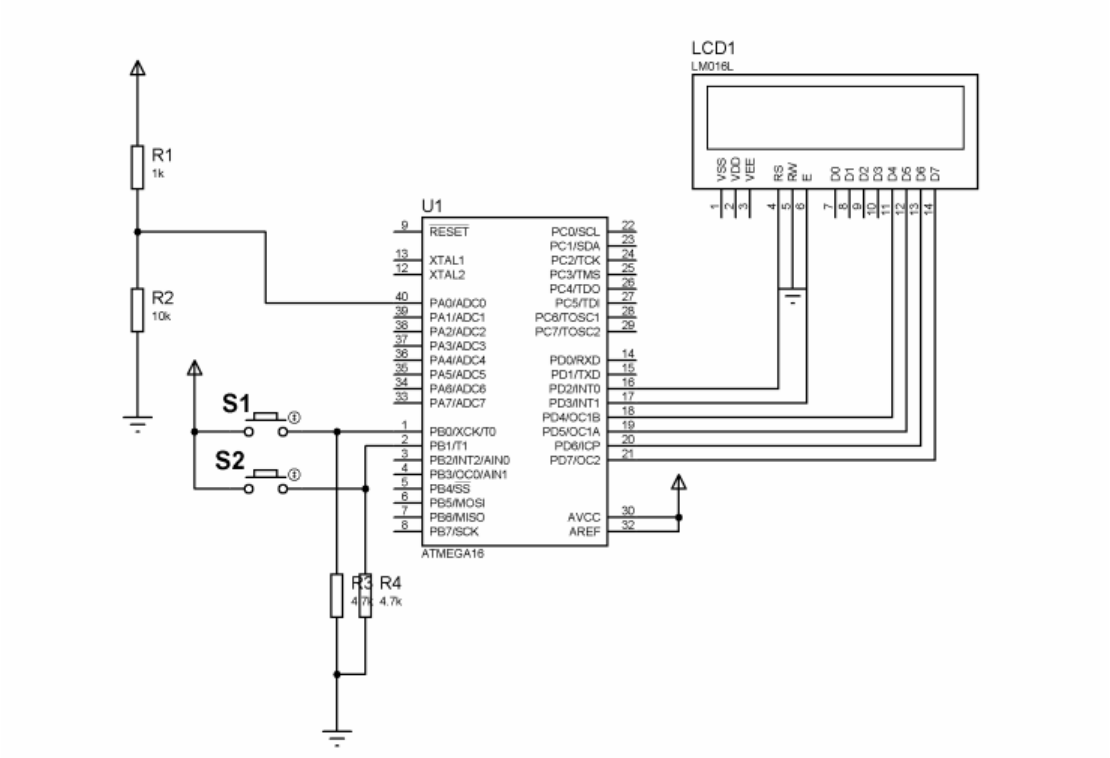
$$5 / 1024 = 0.005\ V$$

بنابراین مقدار ولتاژ اعمالی برای یک شماره صعود عدد دیجیتال شده ، 0.005 V می شود.

در نهایت برای بدست آوردن مقدار ولتاژ VR2 باید ADC را در عدد 0.005 V ضرب کنیم.

حلا با توجه به مقادیر بدست آمده مقدار اهم مقاومت R2 از رابطه زیر بدست می آید :

$$R2 = VR2 / I$$



۱۰ - فرکانس متر

این پروژه قابلیت اندازه گیری پالسهایی مربعی با فرکانس های مختلف را دارد که حتما می بایست به پین B1 (پایه شماره ۲ میکروکنترلر) اعمال شود. کلید S1 برای تغییر رنج فرکانسی و کلید S2 برای انجام عملیات خواندن و نشان دادن فرکانس اعمالی به میکرو می باشد. برای این کار در ابتدا باید توسط تایمر شماره ۲ مدت زمان یک ثانیه را بسازیم تا در این مدت بتوانیم تعداد پالس های رسیده به میکرو را که معادل همان فرکانس است اندازه بگیریم. پس داریم :

F = 8 MHz / 8 = 1 MHz

زمان صعود یک شماره T = 1 / F T = 1 / 1 MHz = 1 us

زمان سرریز تایمر = 200 * 1us = 0.0002 s

در نهایت برای بدست آوردن یک ثانیه واقعی ، تایمر ۲ می بایست ۵۰۰۰ بار سرریز شود :

1s / 0.0002 = 5000

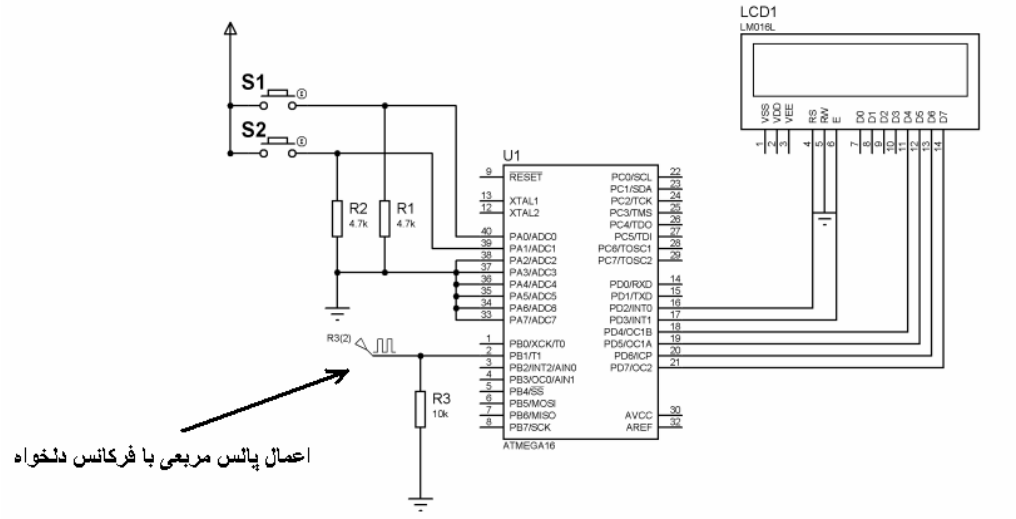
5000 * 0.0002 = 1s

حالا باید بررسی کنیم در مدت زمان یک ثانیه چه تعداد پالس به ورودی کانتر وارد می شود که به همان تعداد باید به متغییر A افزوده گردد.

تعداد پالس شمارش شده از فرمول زیر بدست می آید :

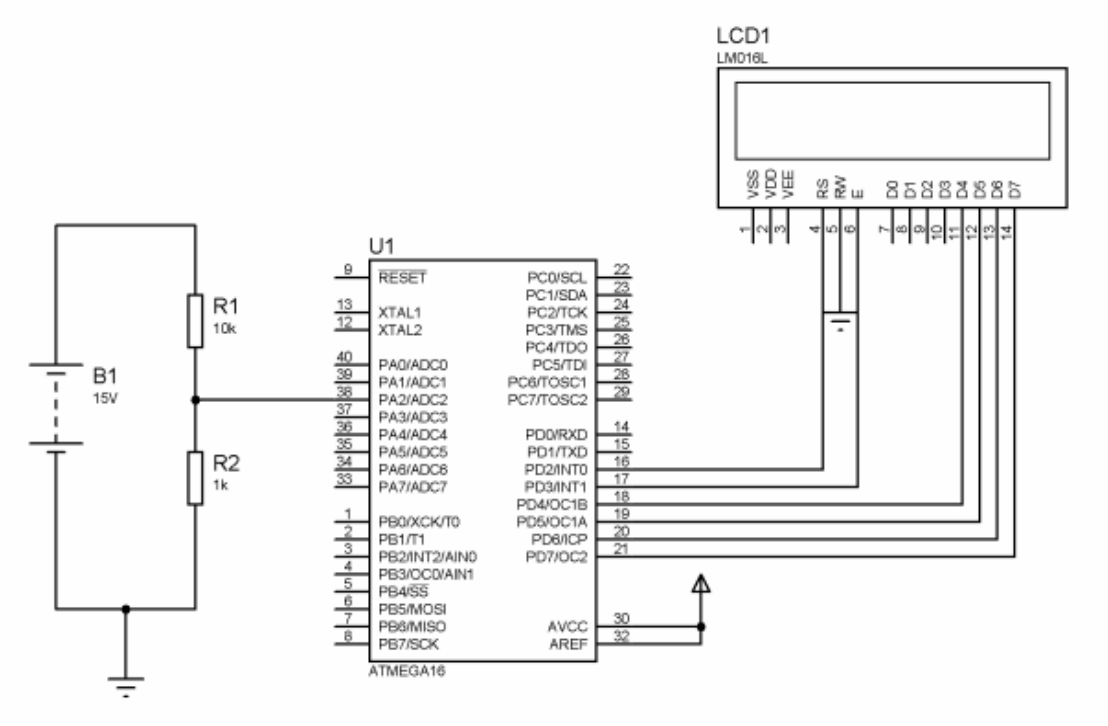
F = 65536 * A

F = F + Counter1



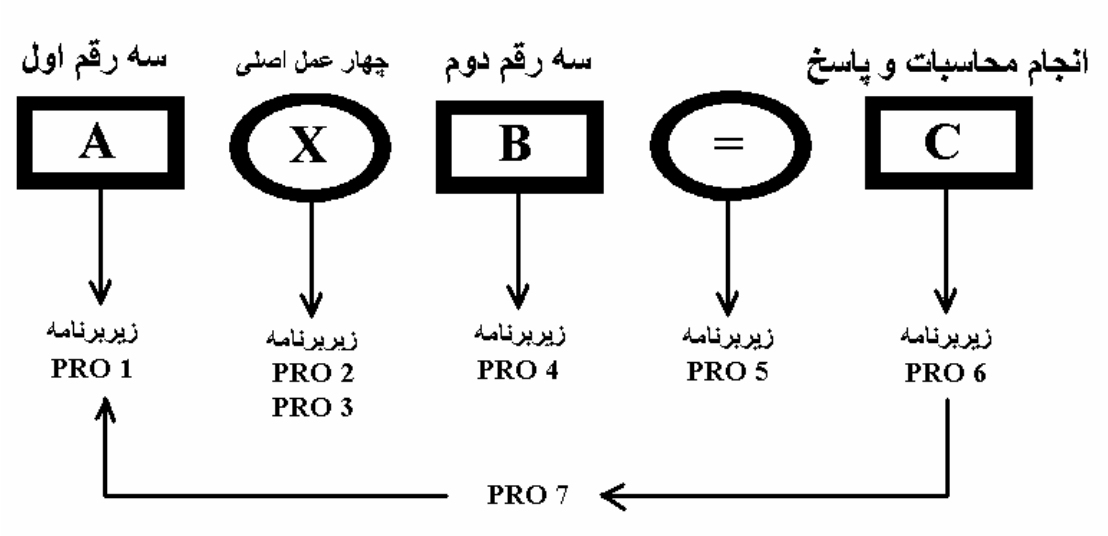
۱۱ - ولت متر دیجیتال

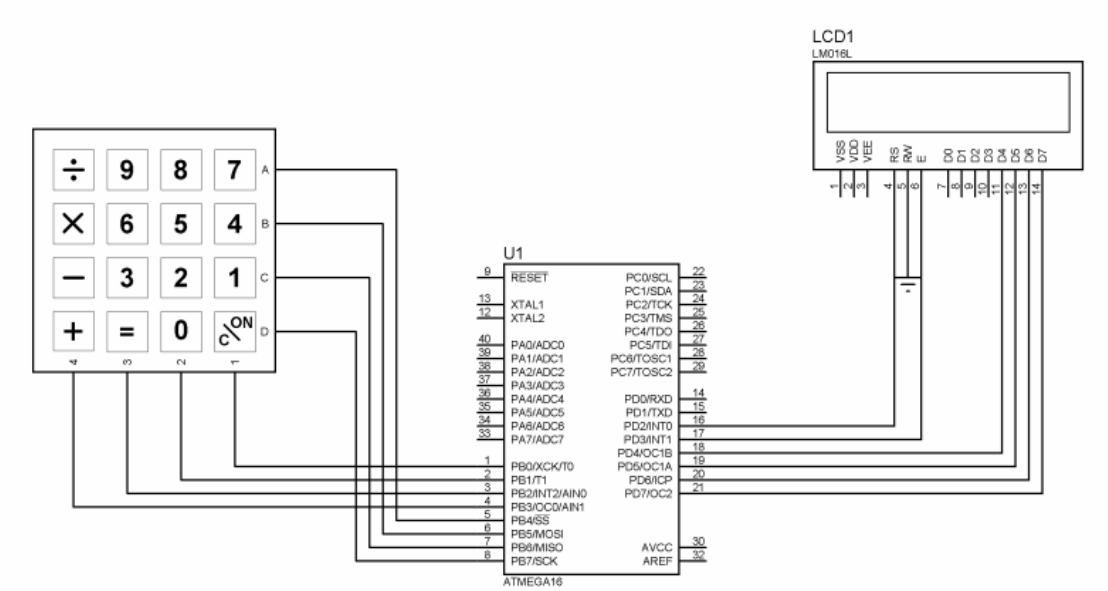
در این پروژه با استفاده از تقسیم مقاومتی و اعمال آن به ADC میکرو می توان ولتاژ DC بین ۰ تا ۵۰ ولت را اندازه گیری نمود. در برنامه این پروژه توجه شود برای کالیبره کردن مقدار ورودی گرفته شده از کانال ADC ، آن را در عدد 0.05377 ضرب نموده ایم و نیز برای حذف اعداد اعشاری بیش از دو رقم ، از دستور Fusing استفاده می کنیم.



۱۲ - ماشین حساب

این پروژه قابلیت انجام ۴ عملی اصلی جمع ، تفریق ، ضرب و تقسیم بر روی اعداد سه رقمی را دارد. در برنامه این پروژه از زیربرنامه هایی من جمله Pro1 تا Pro7 استفاده شده که هر یک از انها برای انجام یکی از عملیات اصلی محاسباتی می باشد که دیاگرام آن بصورت زیر است:

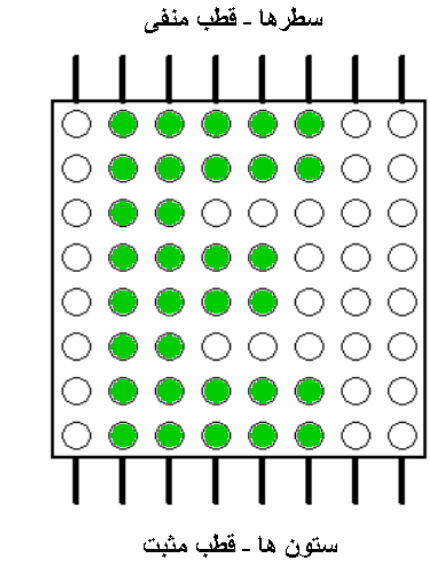




۱۳ - تابلو روان

در این پروژه می توان نوشته یا هر شکل گرافیکی دلخواهی را روی تعدادی از LED ها که بصورت ماتریسی کنار هم قرار گرفته اند ، نمایش داد. برای این کار کافی ست LED ها را بصورت منظم و قانون مند در مدت زمان بسیار کوتاهی (کسری از ثانیه – حدود یک صدم ثانیه) روشن و خاموش کرد. این کار باید توسط میکروکنترلر صورت پذیرد که به اصطلاح اسکن کردن هم گفته می شود.

به LED هایی که بصورت ماتریسی کنار هم قرار می گیرند دات ماتریس هم گفته می شود. همانطور که در شکل زیر هم مشاهده می شود دات ماتریس ها با توجه به تعداد LED های موجود دارای تعدادی پایه هستند که یکسری از آنها پایه های مربوط به ستون ها و یکسری دیگر مربوط به سطرها هستند که باید آند و یا کاتد بودن آنها را با یک عدد باتری تشخیص دهیم تا برنامه را بتوانیم بر همان اساس تنظیم کنیم. با روشن کردن LED های مختلف می توان هر حرف یا شکلی را روی دات ماتریس نمایش داد. در اینجا دات ماتریس حرف " E " را نمایش می دهد.



به اسکن دات ماتریس توسط میکروکنترلر جاروب (Sweep) گفته می شود که به دو روش ستونی و سطری انجام می گیرد.

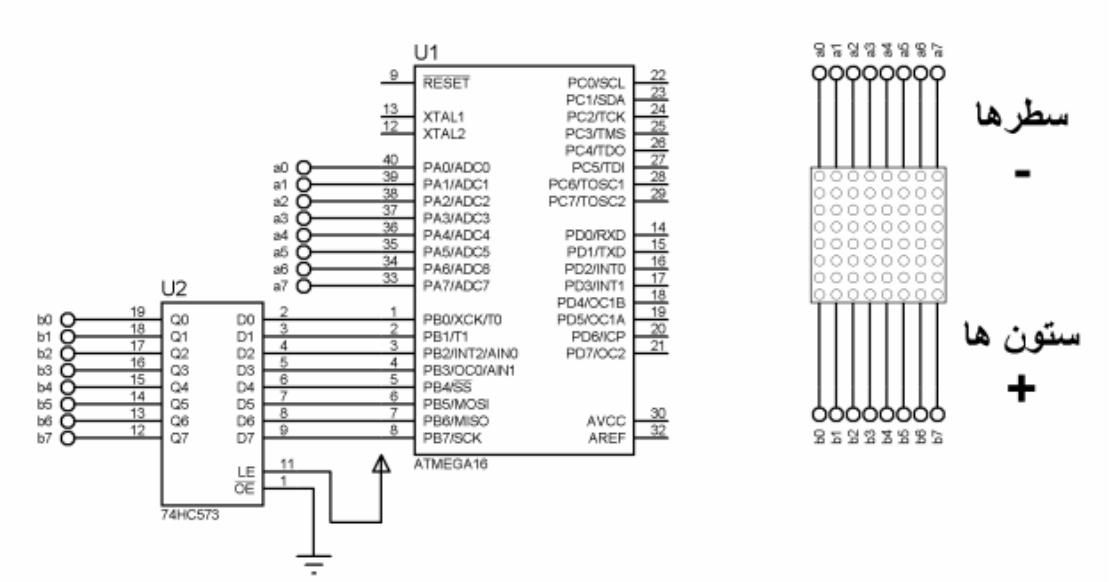
نکته) در ساخت تابلوی روان توجه شود چون تعداد LED ها زیاد است باید از تعدادی ترانزیستور یا آی سی بافر برای تقویت جریان استفاده شود. در اینجا از آی سی 74573 استفاده شده است.

تابلو روان با جاروب سطری

در این پروژه با توجه به نوع برنامه ویسی و اسکن دات ماتریس ، سطرها به ترتیب و یکی پس از دیگری در کسری از زمان روشن و سپس خاموش می شود و همزمان با این کار دیتا (اطلاعات باینری مربوط به طرحی یا نوشته ایی که قرار است روی LED ها نمایش داده شود) به ستون ها وارد می شود که اجرای همزمان آن باعث نمایش تصویر مورد نظر خواهد شد. در اینجه چون سطرهای دات ماتریس مورد استفاده کاتد یا منفی است پس برای روشن کردن هر سطر باید آن را معادل صفر باینری در برنامه قرار بدهیم و نیز چون ستون ها آند یا قطب مثبت است پس در دیتای ارسالی باید برای نمایش هر طرح که به تبع آن یک یا چند LED روشن می شود عدد یک باینری را معادل آن قرار دهیم.

در این پروژه حرف " E " از سمت پایین دات ماتریس به طرف بالا حرکت می کند و این کار بطور مداوم تکرار می شود. در شکل زیر به دیتای این برنامه و عددهای یک باینری که حرف " E " را تشکیل می دهند دقت کنید!

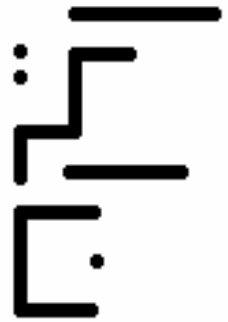
```
Data &B01111110
Data &B01111110
Data &B00000110
Data &B00111110
Data &B00111110
Data &B00000110
Data &B01111110
Data &B01111110
```

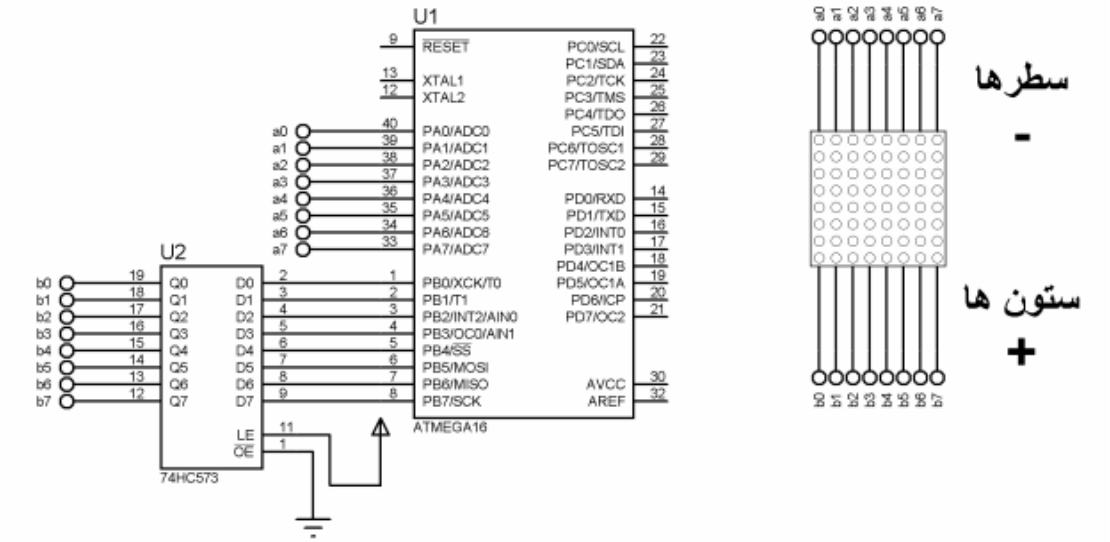


تابلو روان با جاروب ستونی

در این پروژه کلمه " ایران " از سمت چپ دات ماتریس وارد شده و از سمت راست خارج می شود. عملکرد این پروژه بسیار شبیه پروژه قبلی ست با این تفاوت که این کار با جاروب ستونی صورت می پذیرد. یعنی ستون ها باید به ترتیب روشن و خاموش شوند و سپس دیتا وارد سطرها شود. نکته قابل توجه در این پروژه این است که ترتیب قرار گیری اعداد باینری معکوس حالت قبل (روش جاروب سطری) است. یعنی برای روشن کردن ستون ها عدد یک باینری را اعمال می کنیم و برای فرستادن دیتا از عد صفر باینری (برای ایجاد یک کلمه یا شکل) استفاده می کنیم. در شکل زیر به دیتای این برنامه و عددهای صفر باینری که کلمه ایران را تشکیل می دهند دقت شود!

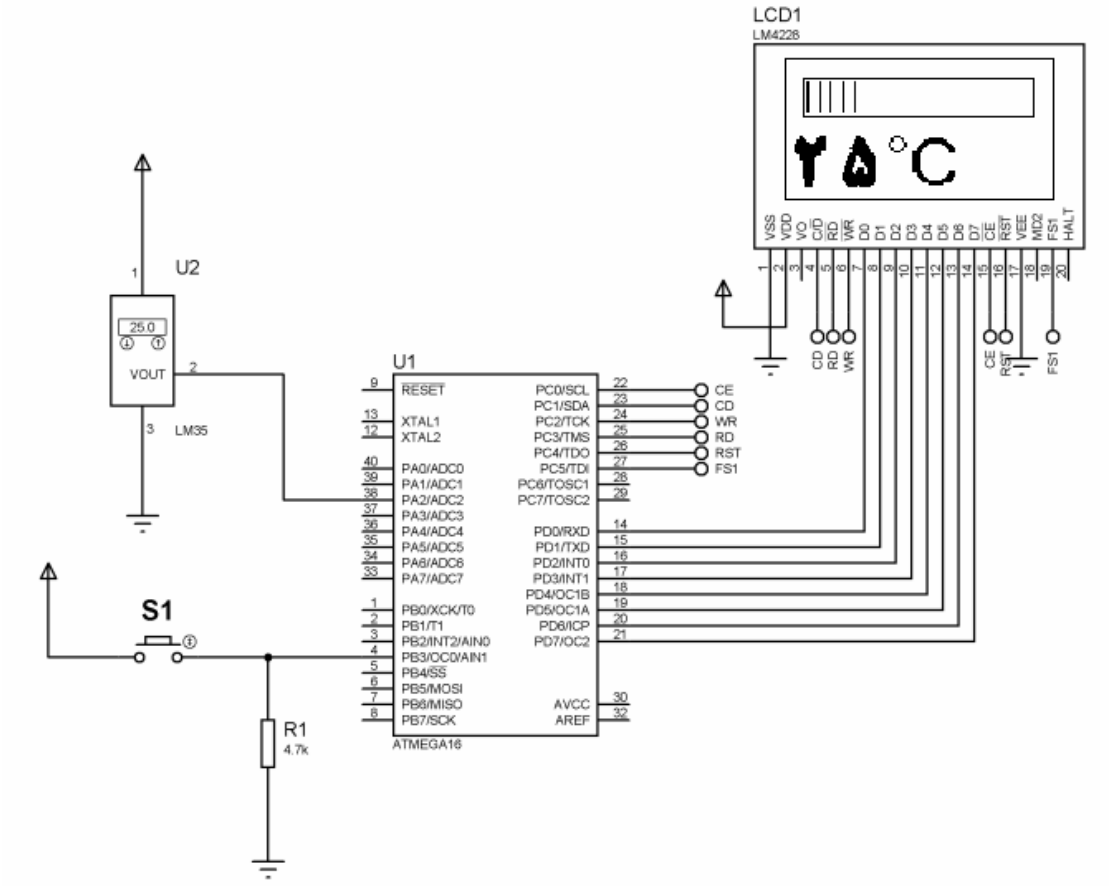
```
Data &B11000001
Data &B11111111
Data &B01001111
Data &B01011111
Data &B11011111
Data &B00011111
Data &B01111111
Data &B01000001
Data &B11111111
Data &B00001111
Data &B01111111
Data &B01110111
Data &B01111111
Data &B00001111
```





۱۴ - دماسنج گرافیکی

در این پروژه ، مقدار دمای محیط به صورت اعداد فارسی همراه با نمودار روی LCD گرافیکی قابل نمایش است. در ابتدای کار به محض روشن کردن مدار کلمه " دماسنج " روی صفحه LCD ظاهر می شود که با فشردن کلید S1 دمای محیط روی LCD نشان داده خواهد شد. در برنامه این پروژه دو زیربرنامه مهم وجود دارد که زیربرنامه Main1 مربوط به تبدیل ADC و محاسبه مقدار دمای گرفته شده از سنسور LM35 می باشد. زیربرنامه Main2 نیز وظیفه تبدیل عدد دما به کاراکترهای فارسی را برای نمایش (تشخیص یک رقمی و یا دو رقمی بودن عدد دما) و ایجاد نمودار بر روی LCD گرافیکی را دارد. البته توجه شود که کلمه " دماسنج " ، اعداد فارسی و کادر نمودار را باید در ابتدا بصورت فایل گرافیکی با پسوند bmp. در نرم افزار Paint (نقاشی ویندوز) و هم اندازه با سایز LCD گرافیکی ، ایجاد کنیم.

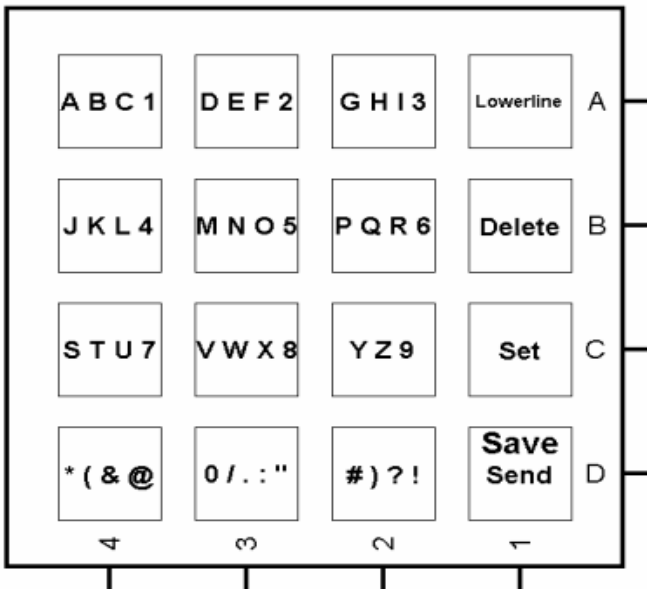


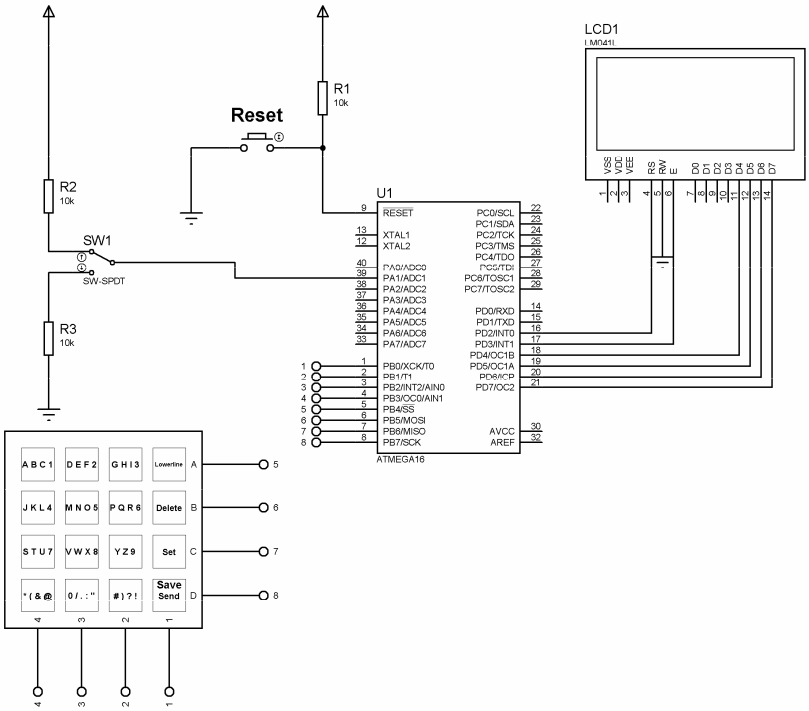
۱۵ - ایجاد پیامک و ذخیره سازی آن

در این پروژه با تایپ حروف درج شده بر روی کی پد می توانیم پیام مورد نظر را روی صفحه LCD مشاهده و نیز در داخل حافظه EEPROM میکروکنترلر ذخیره کنیم که با قطع برق هم پاک نمی شود.

برای تایپ کردن پیام مورد نظر در ابتدا می بایست کلید SW1 را در وضعیت شماره 1 قرار دهیم و سپس شروع به انجام این کار نماییم. در هنگام تایپ توجه شود پس از درج هر کلمه برای تثبیت شدن آن حتما می بایست دکمه Set روی کی پد فشرده شود. دکمه Lowerline برای پرش به سطر بعدی است و دکمه Delete برای حذف کردن کاراکترها می باشد. برای ذخیره کردن پیام مورد نظر هم باید کلید Save را فشار داد. پس از ذخیره سازی برای مشاهده پیام ذخیره شده در ابتدا کلید SW1 را در وضعیت شماره 2 قرار داده و سپس با کلید Reset میکرو را دوباره راه اندازی می کنیم و مشاهده می کنیم که پیام ذخیره شده روی صفحه LCD ظاهر می شود. حتی اگر برنامه شبیه سازی را ببندیم و از نو آن را اجرا کنیم باز هم پیام ذخیره شده روی LCD مشاهده می شود.

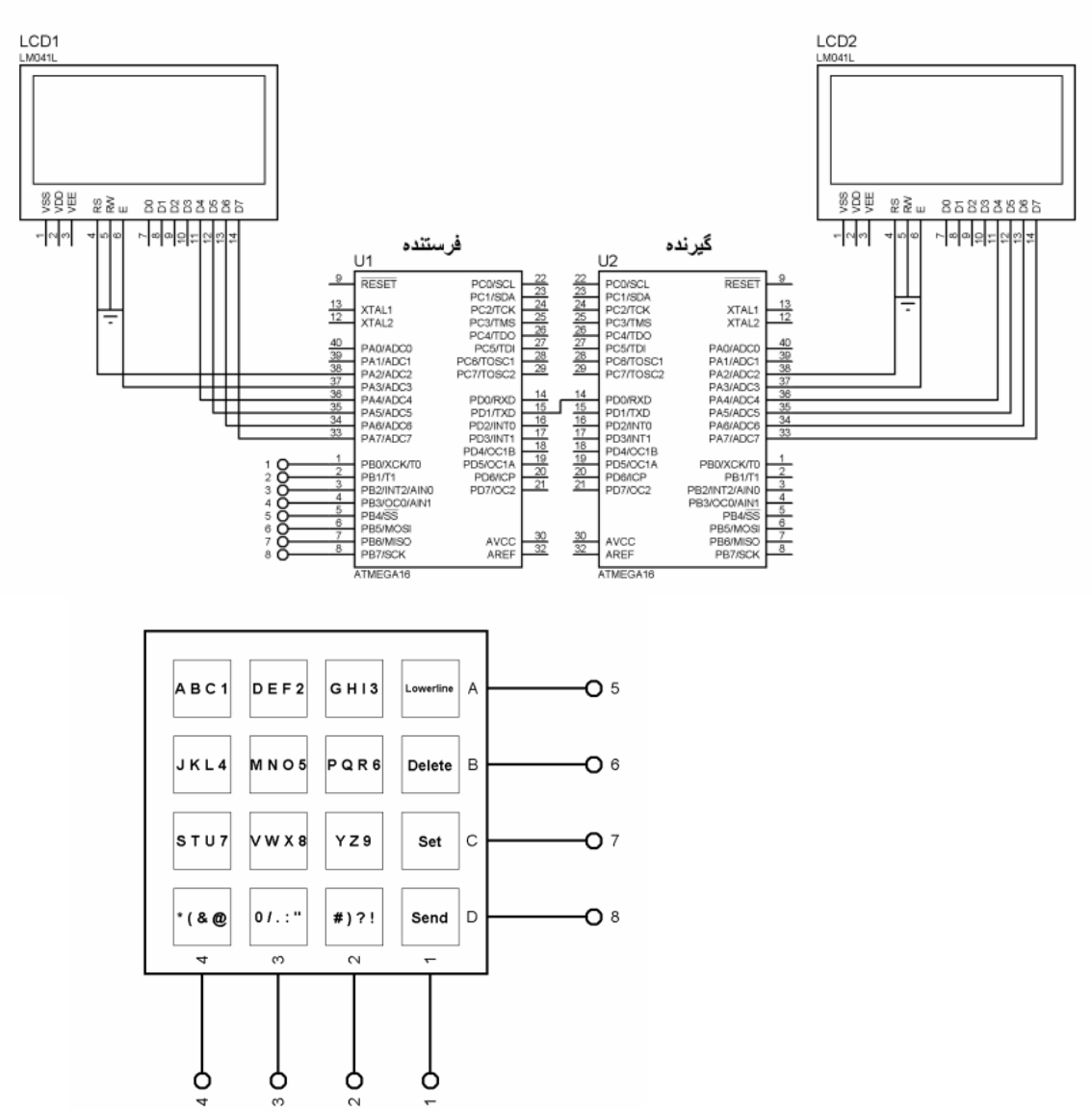
در برنامه این پروژه از تعداد زیادی متغییر برای ایجاد کاراکترها و زیربرنامه های مختلف برای اختصاص هر کلمه و دستور خاص به یکی از کلیدهای کی پد استفاده شده است. در ضمن توجه شود که با تغییراتی در کی پد اصلی نرم افزار پروتئوس (که فقط دارای ارقام می باشد) ایجاد کرده ایم و آن را برای این پروژه بصورت سفارشی تغییر داده ایم.




LCD1
VSS
VDE
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
R1
10k
Reset
R2
10k
SW1
SW-SPDT
R3
10k
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/XCK/T0
PB1/T1
PB2/INT2/AIN0
PB3/OC0/AIN1
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP
PD7/OC2
AVCC
AREF
ATMEGA16
A B C 1
D E F 2
G H I 3
Lowerline
J K L 4
M N O 5
P Q R 6
Delete
S T U 7
V W X 8
Y Z 9
Set
* (& @
0 / . : "
) ? !
Save
Send

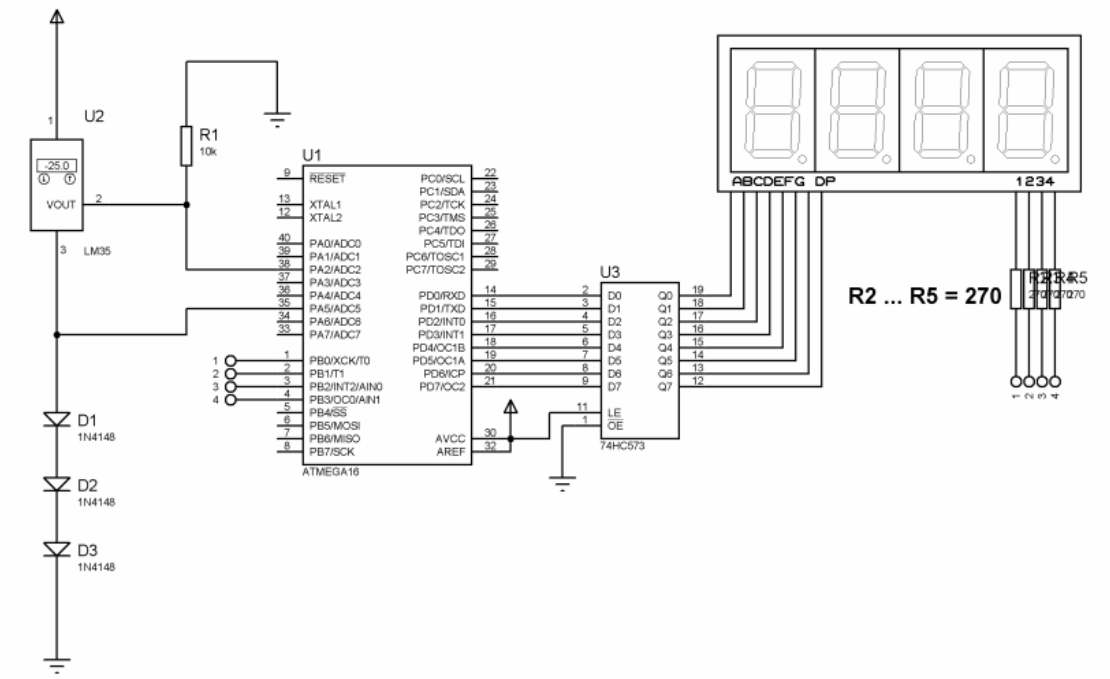
۱۶ - ارسال پیامک از یک میکروکنترلر به میکروکنترلر دیگر

عملکرد این پروژه نیز مانند پروژه قبلی است با این تفاوت که با فشردن کلید Send پیام مورد نظر از میکروکنترلر فرستنده به میکروکنترلر گیرنده منتقل شده و در صفحه LCD آن قابل مشاهده است. این انتقال توسط ارتباط سریال UART صورت می گیرد. در ضمن برای این پروژه دو برنامه جداگانه نوشته شده که noname1 برای میکروکنترلر فرستنده و noname2 برای میکروکنترلر گیرنده می باشد. نکته مورد توجه در برنامه های فرستنده و گیرنده ارسال دیتای همزمانی است که باعث می شود ارتباط بین میکروکنترلر فرستنده و گیرنده درست و همزمان برقرار شود تا میکروکنترلر گیرنده ابتدا و انتهای یک پیام را تشخیص بدهد. این دیتا با علامت " # # " در برنامه نویسی مشخص شده است.



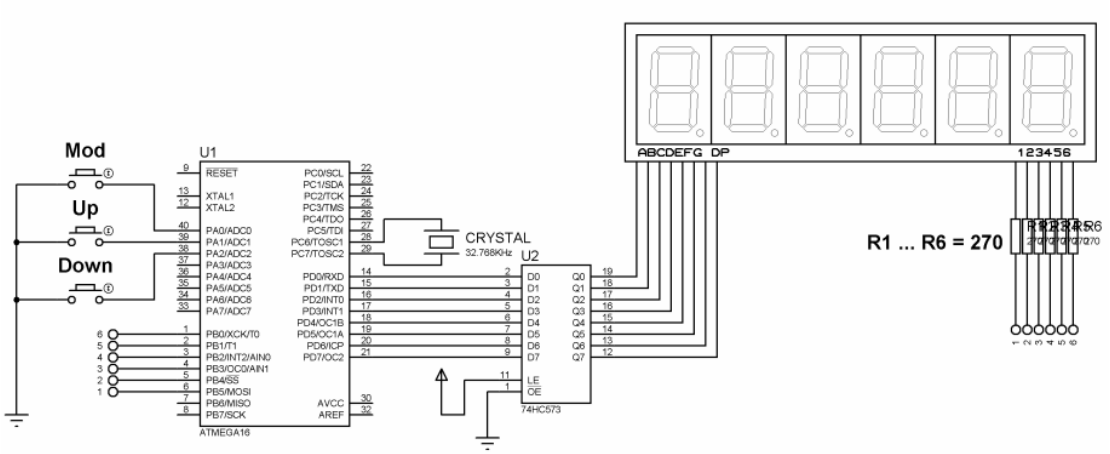
۱۷ - دماسنج با نمایش دمای منفی

در این پروژه می توان دمای ۳۶ - تا ۹۹ + درجه سانتی گراد را اندازه گیری نمود. عملکرد این پروژه تقریبا شبیه پروژه دماسنج قبلی می باشد با این تفاوت که اعداد را روی 7Seg نمایش می دهد و نیز قابلیت اندازه گیری دمای کمتر از صفر درجه را نیز دارد. برای اندازه گیری دمای منفی می بایست دو کانال ADC را همزمان فعال کرده ، اولی را به خروجی سنسور LM35 و دومی را به پایه شماره ۳ این سنسور که همان تغذیه منفی است وصل می کنیم. تفاضل این دو پایه باعث می شود که بتوانیم دمای کمتر از صفر را نیز اندازه بگیریم. البته برای بالا بردن دقت اندازه گیری از سه عدد دیود یکسوساز هم در مسیر تغذیه منفی سنسور دما استفاده کرده ایم. برای نمایش عدد دما روی 7Seg ها از روش اسکن ماتریسی استفاده شده که زیر برنامه Main2 مربوط به همین روش است. برای تقویت جریان مصرفی 7Seg ها از آی سی بافر 74HC573 استفاده شده است.



۱۸ - ساعت RTC

از مزیت های مهم این نوع ساعت دقیق بودن آن است که با استفاده از کریستال خارجی 32.768 KHz این کار صورت می گیرد. در این پروژه با پیکره بندی دستور ساعت RTC و مقداردهی اولیه آن ، ساعت 12:00:00 را روی 7Seg نشان می دهد و برای تنظیم آن می توان از کلیدهای Up و Down استفاده نمود. کلید Mod نیز برای تغییر حالت بین تنظیم ساعت و دقیقه می باشد. دستورات به کار رفته در حلقه Do – Loop مربوط به تنظیمات ساعت می باشد. زیربرنامه های Main و Dat مربوط به نمایش ساعت روی 7Seg ها می باشد و زیربرنامه Button مربوط به ایجاد یک تاخیر زمانی در هنگام فشردن کلیدها می باشد.



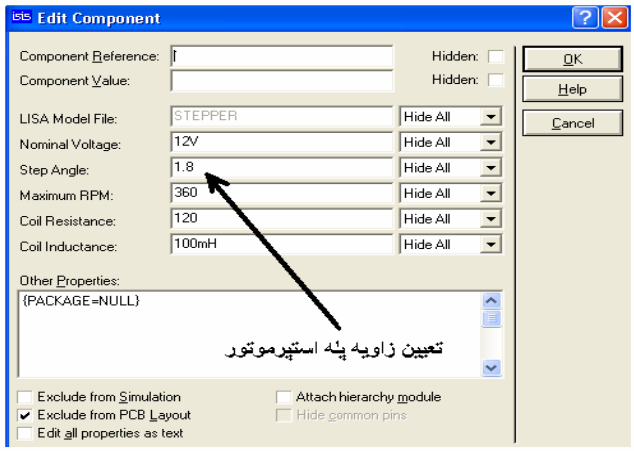
۱۹ - راه اندازی استپر موتور

در این پروژه با استفاده از کدهایی که توسط میکرو به استپر موتور می دهیم باعث می شود که موتور در ابتـدا یـک دور کامل درجهت ساعتگرد بچرخد و سپس بصورت پادساعتگرد به حالت اول خود بر می گردد. تعداد پـالس هـایی که باید برای یک دور کامل به موتور بدهیم از تقسیم عدد ۳۶۰ بر مقدار درجه موتر بدست می آید که در این پروژه عد ۲۰۰ بدست می آید :

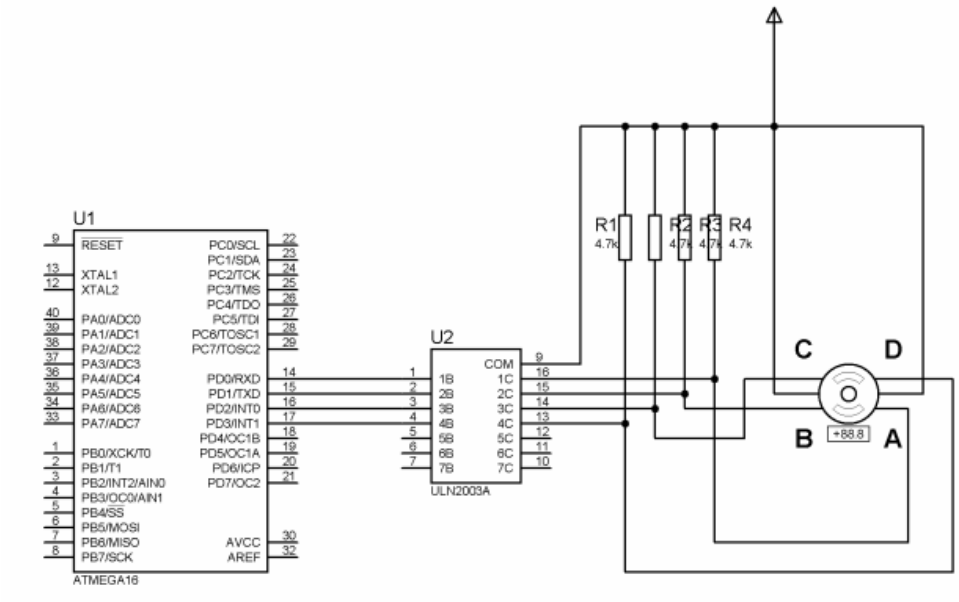
$$360 / 1.8 = 200$$

پس به هر یک از سیم های استپر موتور می بایست 50 پالس اعمال شود.

برای تنظیم زاویه پله موتور در نرم افزار پروتئوس باید روی استپر موتور دابل کلیک کرده تا پنجـره زیـر بـاز شـود و سپس در قسمت مشخص شده درجه موتور را وارد کنیم.

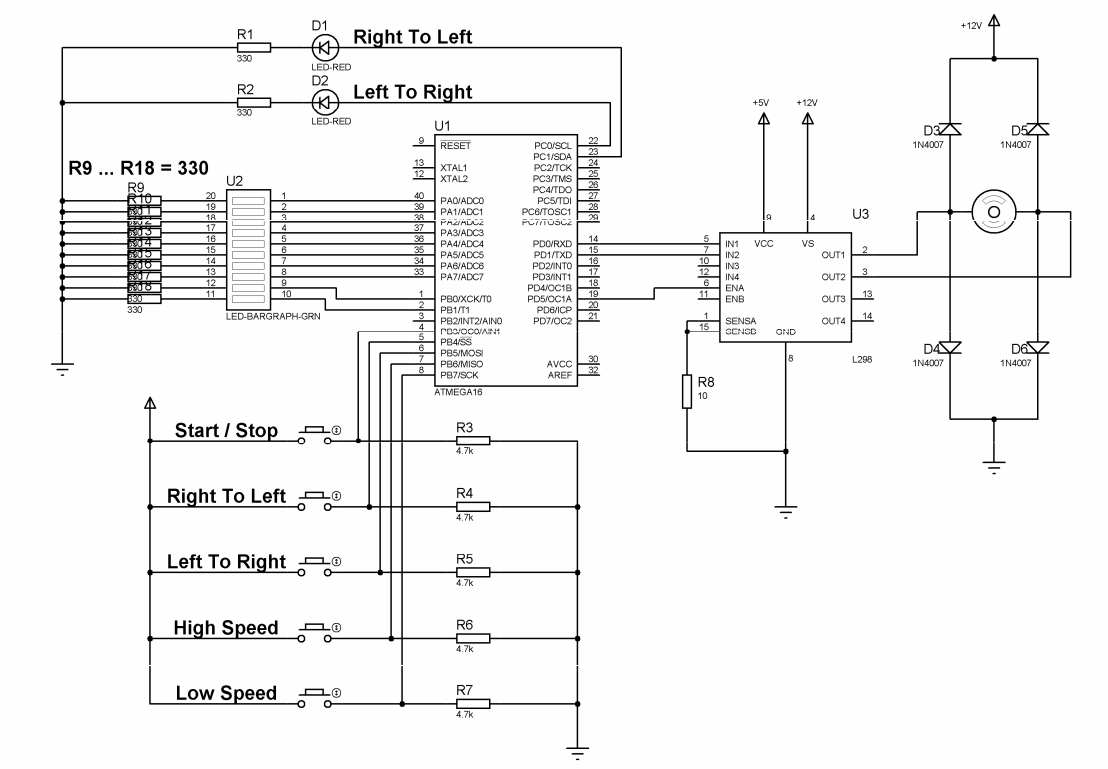


برای تقویت جریان مصرفی موتور از آی سی معروف Uln2003 استفاده شده است.



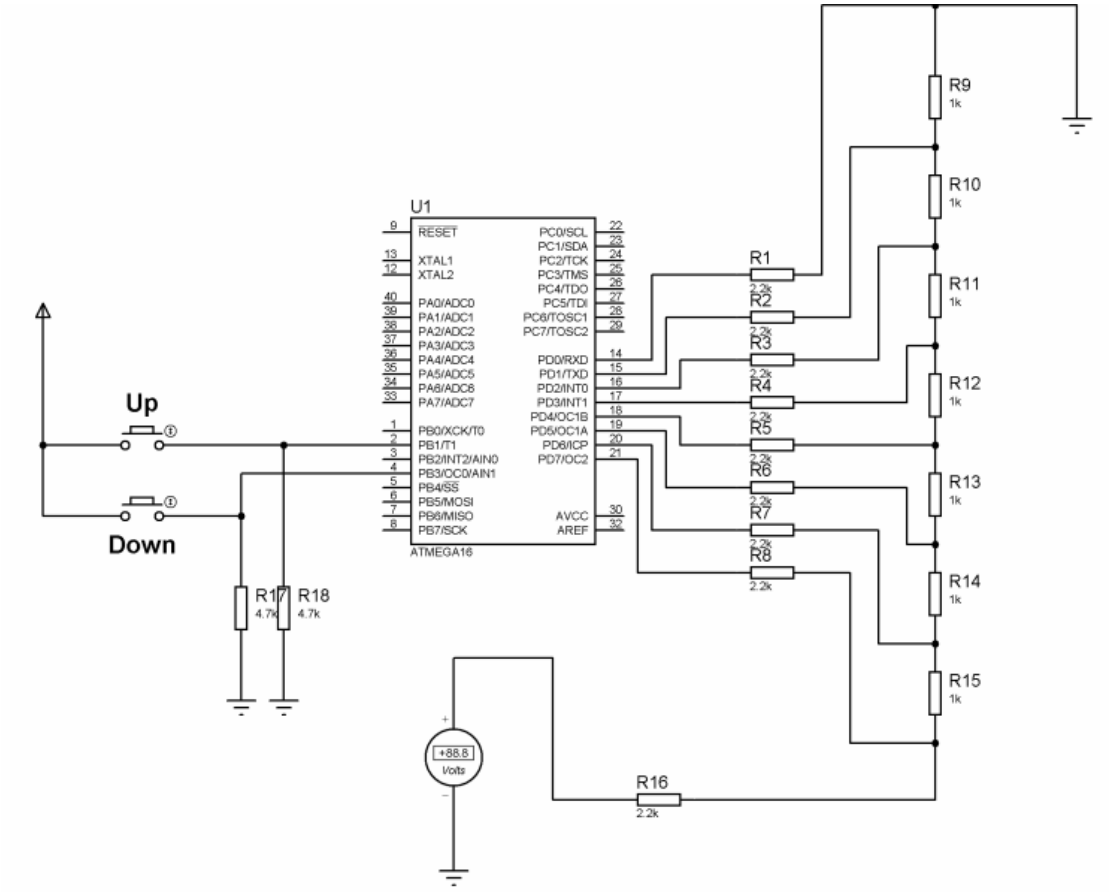
۲۰ - کنترل دور موتور با PWM

در این پروژه با استفاده از پالس های PWM می توان سرعت حرکت یک موتور DC را کم و یـا زیـاد نمـود و نیـز جهت آن را می توان عوض کرد. برای این کار ۵ عدد کلید در نظر گرفته شده که عملکرد مربوط به هر کدام در کنار آن نوشته شده است و با توجه به نوع عملکرد LED مربوط به آن نیز روشن می شود. برای تقویت جریـان مـصرفی موتور از آی سی L298 استفاده شده و همینطور برای حفاظت این آی سی در برابر جریان های برگشتی موتـور ، از ۴ عدد دیود یکسوساز استفاده کرده ایم. در متن برنامه نیز برای فشردن هر یک از کلیـدها حالـت هـای مختلفـی را تعریف کرده ایم که به آی سی L298 دستور می دهد موتور در چه جهتی بچرخد و نیز چـه مقـدار از پـالس هـای PWM را به موتور اعمال نماید. زیر برنامه Leds(n As Word) مربوط به نمایش LED های ستونی مربوط بـه مقدار سرعت است.



۲۱ - ولوم دیجیتال

در این پروژه با استفاده از دو کلید Up و Down مقدار سطح ولتاژ اعمال شده به دو سر مقاومت R16 تغییر می کند که محدوده آن بین ۰ تا ۵ ولت است و در واقع می توان به جای مقاومت از هر وسیله ایی که نیاز به تنظیم دارد استفاده کرد که در واقع این مدار به جای ولوم آن عمل می کند. در برنامه این پروژه با توجه به فشرده شدن کلیدهای Up و Down اعداد بین ۰ تا ۲۵۵ در خروجی پورت D میکروکنترلر قرار می گیرد که بصورت باینری است و برای تبدیل آن به مقدار آنالوگ می بایست از مبدل دیجیتال به آنالوگ یا DAC استفاده شود که در این مدار از قرار گرفتن تعداد مقاومت به صورت نردبانی (Lader) این کار صورت می گیرد و نتیجه آن روی ولتمتر قابل نمایش است. در ضمن مقادیر اعمال شده در این پروژه ، در حافظه دائمی میکروکنترلر ذخیره می شود تا پس از خاموش کردن مدار و سپس استفاده مجدد آن در همان مقدار باقی بماند.



۲۲ - دیمر دیجیتال

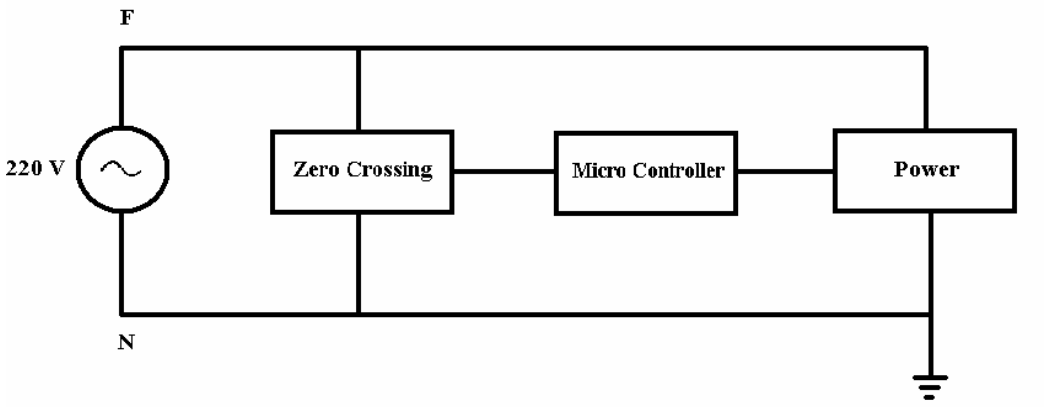
دیمر وسیله ای ست برای کنترل توان مصرفی وسایل برقی از قبیل لامپ ها (فقط لامپ های رشته ایی) ، هیترها ، موتورهای AC و ... که می توان آن را به دلخواه تغییر داد. مثلا در این پروژه می توانیم با فشردن کلیدهای Up و Down ، نور لامپ را کم و یا زیاد کنیم.

این پروژه از سه قسمت زیر تشکیل شده است:

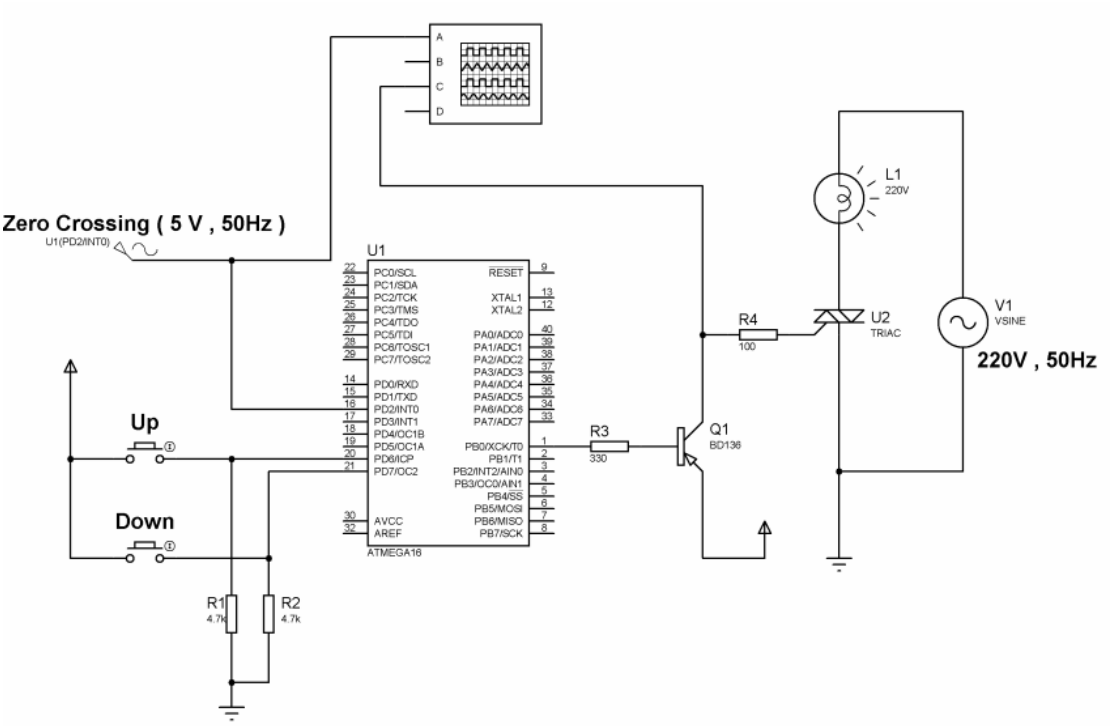
۱ - مدار آشکار ساز عبور از صفر (Zero Crossing)

۲ - مدار کنترل کننده شامل میکروکنترلر AVR

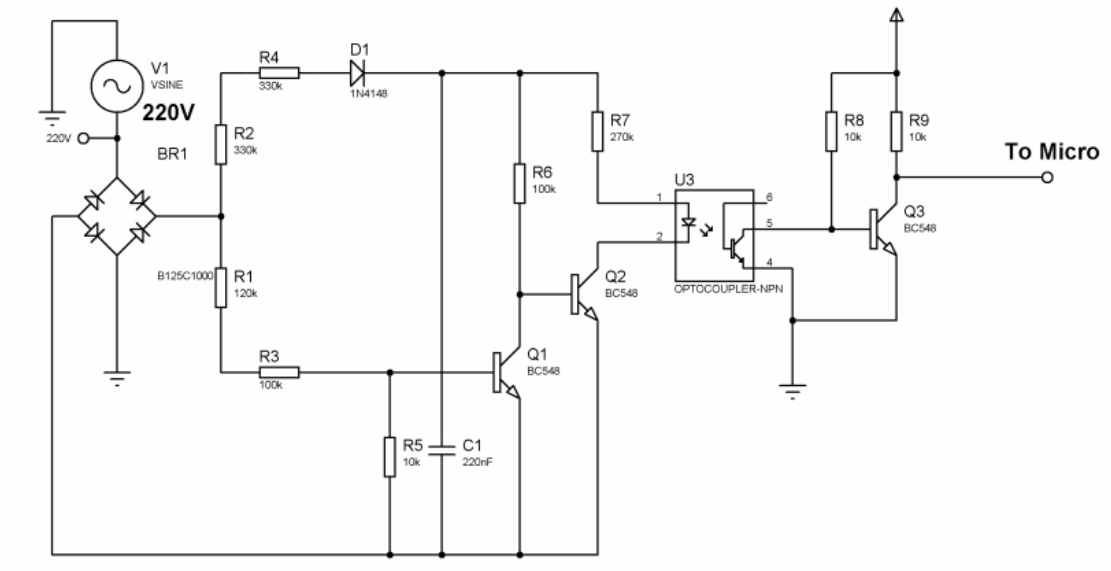
۳ - مدار قدرت شامل ترایاک



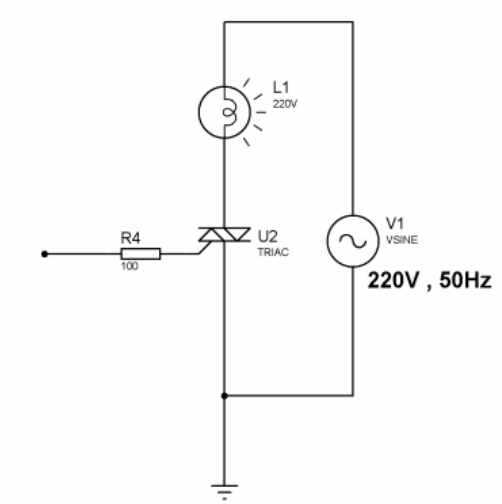
در این پروژه در ابتدا با استفاده از مدار Zero Crossing آغاز سیکل جدید موج سینوسی برق را تشخیص می دهیم ، یعنی هر بار که سطح ولتاژ صفر شد یک پالس تولید شده و سپس آن را به پایه ورودی وقفه خارجی میکروکنترلر انتقال می دهیم. میکروکنترلر هم با رسیدن این پالس ، پس از گذشت مدت زمانی بسیار کوتاه و مشخص یک پالس مربعی در پایه B.0 تولید می کند که در واقع همان زاویه آتش است. زاویه آتش را در ابتدا توسط یک عدد ترانزیستور تقویت شده و سپس به پایه گیت ترایاک اعمال می گردد. با تغییر پالس های زاویه آتش می توان ترایاک و در نتیجه نور لامپ متصل به آن را کنترل کرد. در برنامه این پروژه زیر برنامه Pow مربوط به اعمال پالس زاویه آتش با توجه به وقفه رسیده توسط مدار عبور از صفر می باشد. مقادیر وارد شده توسط کلیدهای Upو Down در حافظه EEPROM میکرو ذخیره می شوند تا با همان مقادیر باقی بمانند.



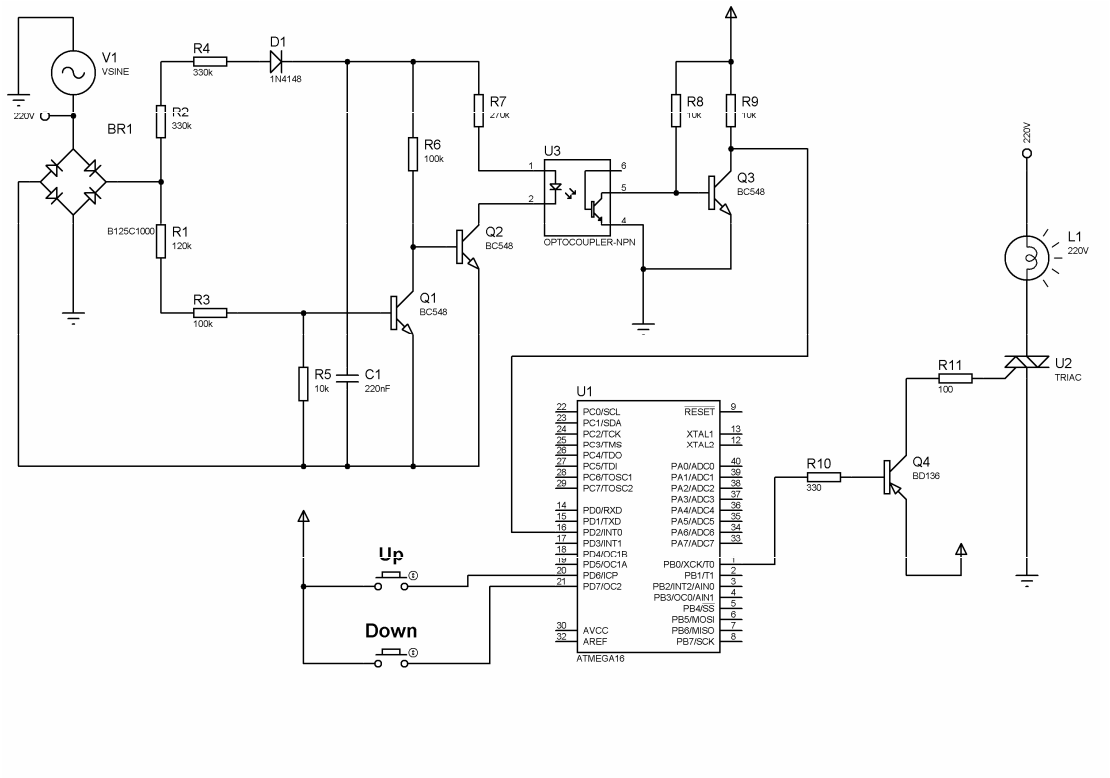
در شبیه سازی این پروژه به جای استفاده از مدار عبور از صفر ، ولتاژ AC را مستقیما به پایه D.2 اعمال کرده ایـم ، ولی برای ساختن مدار واقعی حتما می بایست از مدار زیر استفاده کنیم:



مدار قدرت این پروژه هم بصورت زیر است :



مدار تکمیل شده پروژه دیمر دیجیتال :

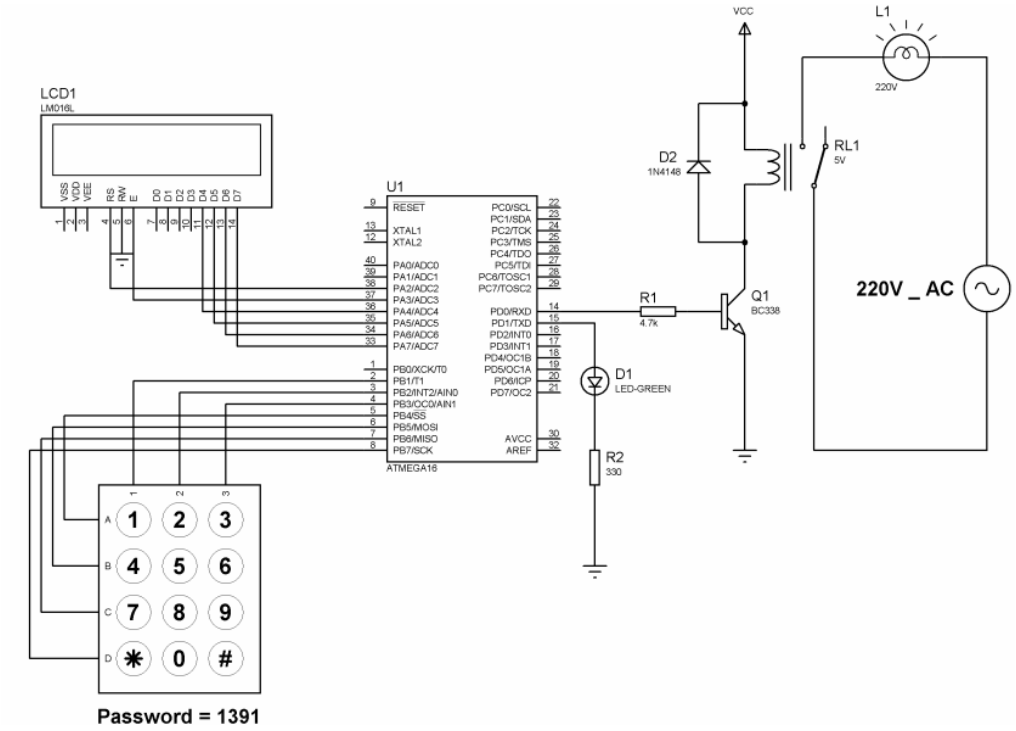


۲۳ - قفل رمز دیجیتال

در این پروژه با وارد کردن عدد 1391 و سپس فشردن کلید "#" کی پد ، پایه های D.0 و D.1 میکروکنترولر فعال می شوند که به تبع آن رله و لامپ ۲۲۰ ولتی و نیز LED موجود در مدار ، روشن می شوند. در صورتی که رمز را بطور اشتباه وارد کنیم کلمه "Error" روی صفحه LCD ظاهر می شود.

بعد از اینکه با وارد کردن رمز صحیح ، سیستم فعال شد ، با فشار دادن کلید شماره ۱ لامپ و LED خاموش می شوند و برای فعال سازی آن ها می بایست دوباره رمز صحیح را وارد کنیم. برای تغییر رمز می بایست کلید شماره ۲ را انتخاب کنیم و در ابتدا رمز اولیه سیستم را وارد کرده ، سپس رمز دلخواه خود را وارد می کنیم که می تواند از یک تا ۸ رقم باشد. رمز جدید برای همیشه در حافظه دائمی میکروکنترلر ذخیره می شود.

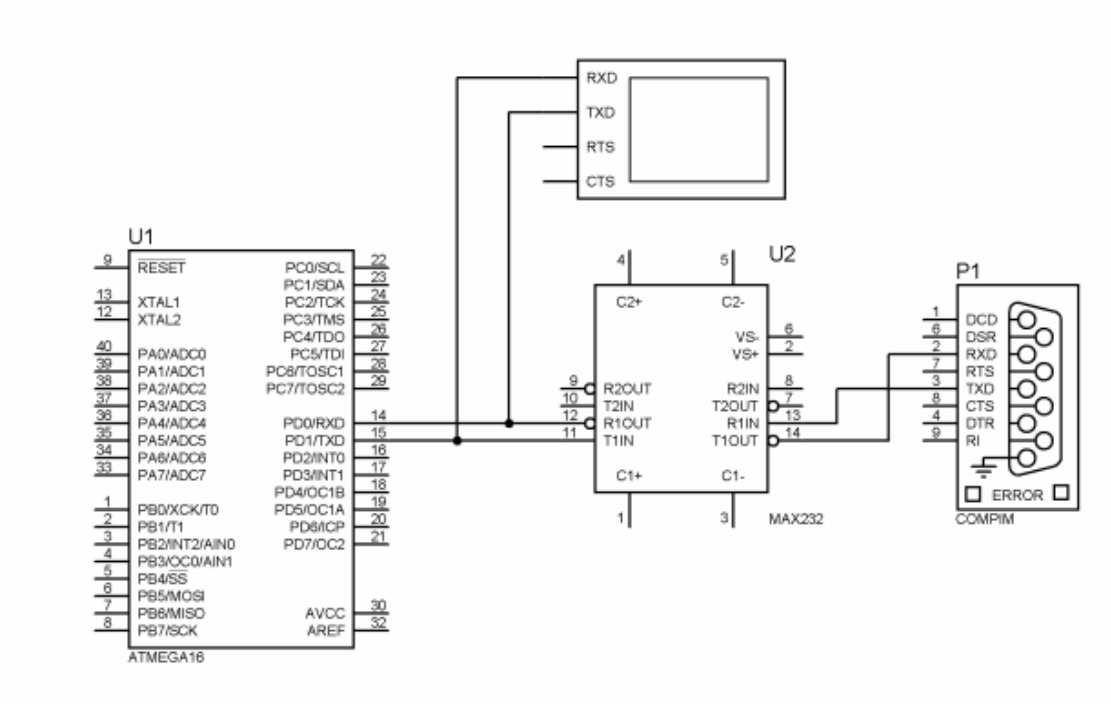
در برنامه این پروژه متغییر M مربوط به رمز سیستم می باشد. زیر برنامه Pass_01 مربوط به شروع فعالیت سیستم برای کنترل پورتها و آمادگی برای دریافت رمز می باشد. زیربرنامه Pass_03 برای انتخاب حالت خروج (Exit) و تغییر رمز است. زیر برنامه Pass_04 برای اعمال تغییر رمز می باشد و همینطور زیربرنامه Pass_05 برای وارد نمودن رمز جدید توسط کاربر می باشد. زیر رنامه های Pass_06 و Pass_07 برای دریافت و تبدیل اعداد گرفته شده از کی پد می باشند. در زیربرنامه Code که مروط به جدول Lookup کی پد است ، از کدهای اسکی استفاده شده که در نهایت از حالت رشته ایی به اعداد ریاضی تبدیل می شوند.



۲۴ - ارتباط میکروکنترلر با کامپیوتر

در این پروژه با استفاده از ارتباط سریال UART بین میکرو و کامپیوتر اطلاعاتی را ارسال و دریافت می کنیم. برای انجام این کار در حالت واقعی می بایست از محیط Terminal Emulator نرم افزار بیسکام بهره بگیریم.

پس از برقراری ارتباط در ابتدا پیغام Enter AVR روی صفحه مانیتور کامپیوتر از طرف میکروکنترلر ظاهر می شود که می بایست کلمه AVR را از طریق صفحه کلید کامپیوتر وارد کنیم. در صورتی که ان کلمه را بصورت صحیح وارد نکنیم کلمه Error ظاهر می شود. بعد از وارد نمودن کلمه AVR و ارسال آن از طریق کامپیوتر به میکرو ، کلمه WELLCOME روی صفحه مانتیتور ظاهر می شود و سپس با نمایش عبارت What Is Your Name? از ما می خواهد که اسم خود را وارد کنیم. و در آخر کلمه Hello در کنار اسم وارد شده نمایش داده می شود.



۲۵ - ربات مسیریاب

این پروژه یک ربات مسیریاب با قابلیت تعقیب خط مشکی بر روی صفحه سفید و نیز تعقیب خط سفید بر روی صفحه مشکی می باشد که می تواند زوایه های تیز و ۹۰ درجه را براحتی تشخیص داده و بدون انحراف از آن ها عبور می کند. برای شروع به کار این ربات کافیست در ابتدا کلید S1 فشرده شود و سپس ربات را در حالت یا مکان مورد نظر قرار بدهیم. برای غیر فعال کردن ربات نیز از همان کلید S1 استفاده می کنیم. با تغییر حالت هر یک از هفت سنسور قرار داده شده (از 0 به 1 و بلعکس) می توان جهت حرکت چرخ ها را عوض کرد که همان تغییر مسیر ربات خواهد بود.

این ربات (و همینطور اکثر رباتهای دیگر) از قسمت های زیر تشکیل می شوند :

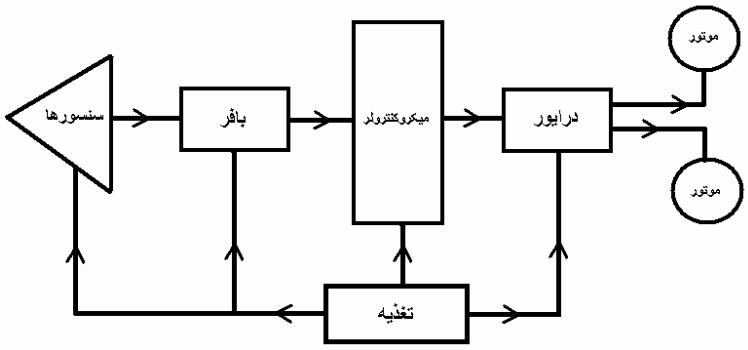
۱- **سنسورها :** برای دریافت اطلاعات از محیط

۲- **بافر :** برای تقویت سیگنال سنسورها و نیز تبدیل آنها به مقدار منطقی (0 و 1)

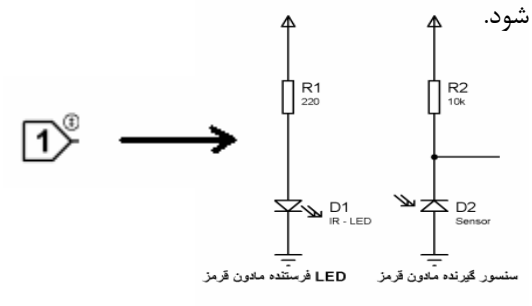
۳- **میکروکنترلر :** برای پردازش اطلاعات و کنترل سیستم (کلیه ورودی ها و خروجی ها)

۴- **درایور :** تقویت جریان خروجی میکروکنترلر برای راه اندازی موتورها

۵- **موتورها :** حرکت کردن ربات



برای سنسورهای این پروژه از LED های فرستنده و گیرنده مادون قرمز (IR) استفاده شده که قابلیت تشخیص سطح سیاه و سفید را دارند. در واقع به جای هر یک از هفت المان لاجیک در نرم افزار شبیه ساز ، یک جفت از گیرنده و فرستنده مادون قرمز به صورت زیر جایگزین می شود.

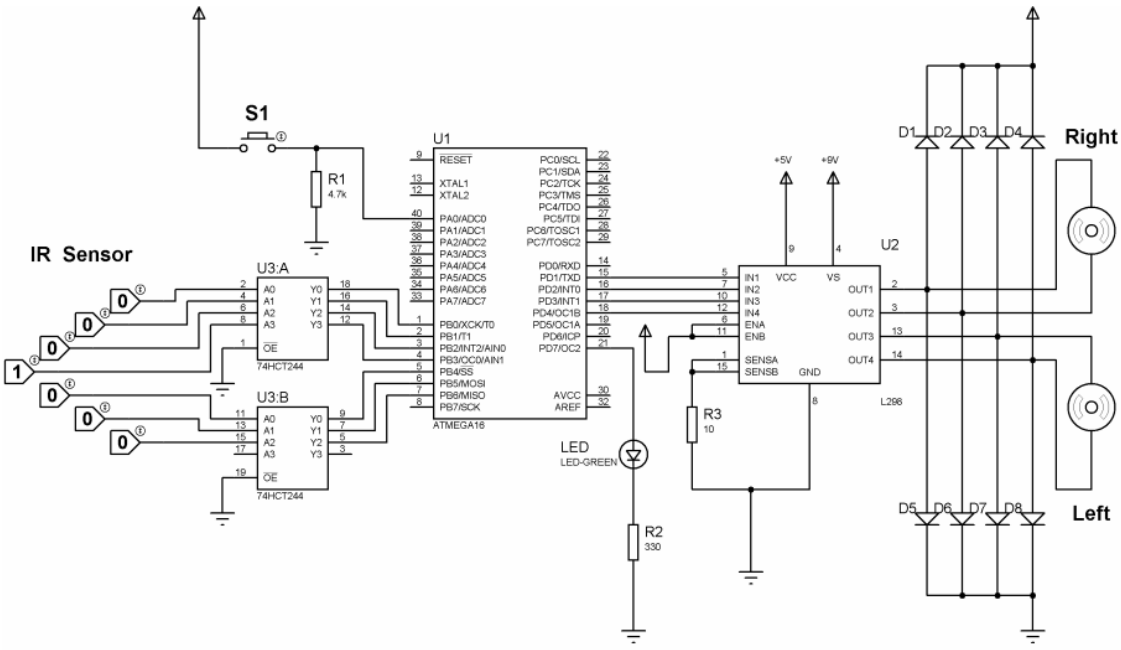


توجه شود برای سنسورها می بایست یک بورد جداگانه در نظر گرفتـه کـه در مجمـوع ۱۴ عـدد LED فرسـتنده و گیرنده مادون قرمز روی آن نصب می شود (با آرایش مثلثی).

برای بافر از آی سی 74244 استفاده شده که اطلاعات دریافتی از سنسورها را تقویت و به سـطح دیجیتـالی 0 و 1 تبدیل می کند تا برای میکروکنترلر قابل درک باشد. میکروکنترلر نیز با توجه به دریافت ورودی ها و طبق برنامه ای که به آن داده ایم ، پورت خروجی را برای به حرکت درآوردن به موقع موتورها کنترل می کند. چون جریـان پـورت خروجی میکروکنترلر برای راه اندازی موترهای الکتریکی کـافی نیـست ، مـی بایـست از درایـور مخـصوص موتورهـا استفاده کنیم که در این پروژه از آی سی L298 استفاده شده است. برای حذف جریان برگشتی موتور نیز از ۸ عدد دیود یکسوساز استفاده کرده ایم.

برای تهیه قسمت های مکانیکی این ربات توجه شود که از طراحی خوبی برخـوردار باشـد تـا در حرکـت بـا مـشکل مواجه نشود. موتورها حتما از نوع گیربکس دار تهیه شود و چرخ های آن با قطـر ۵ سـانتیمتر باشـد. بـرای شاسـی (بدنه) از پلکسی گلاس (طلق) استفاده شود چون هم سبک است و هم براحتی می توان آن را سوراخ کاری نمود.

در برنامه نویسی این پروژه ابتدا برای فشرده شدن کلید S1 دو شرط فعال بودن و یا عـدم شـروع بـه کـار آی سـی میکروکنترلر تعریف شده و نیز شرط هایی برای حالت های مختلف سنسورها که مشخص کننده مسیر حرکـت مـی باشند و همینطور ۶ زیربرنامه که با توجه به برقراری شرط ها ، برنامه به یکی از آنها پرش کرده و خروجی را با توجه به آن فعال می کند که نتیجه آن حرکت ربات بر طبق مسیر تعیین شده خواهد بود.



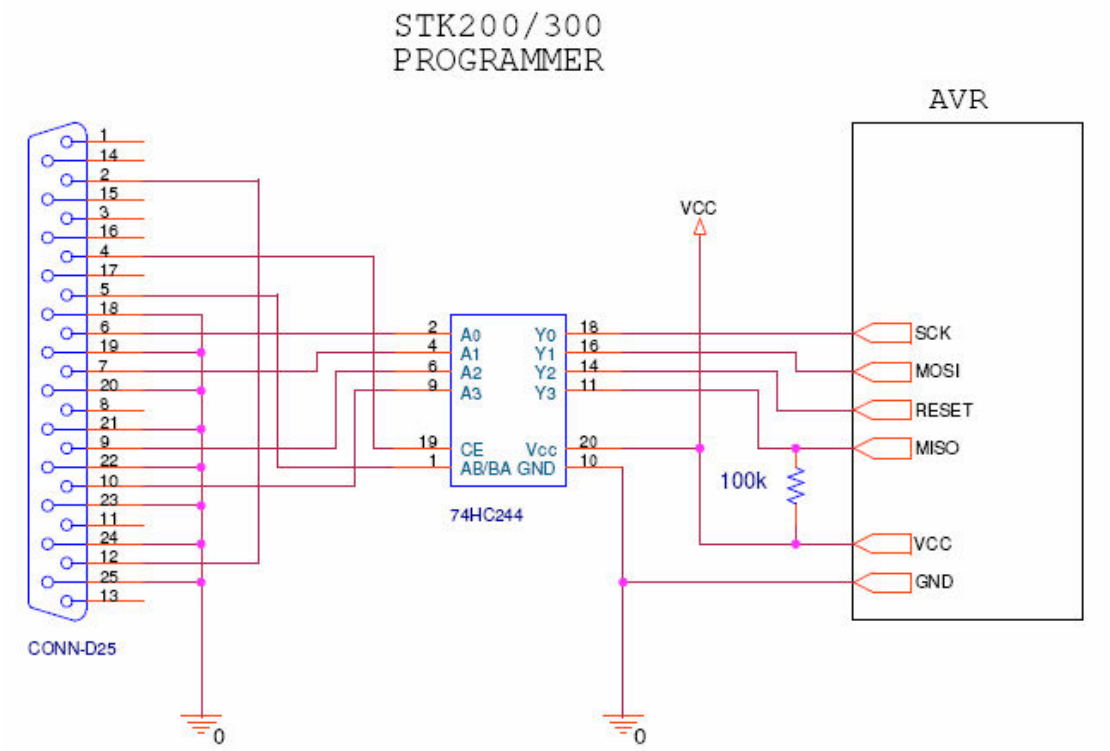
ضمیمه ۱

ساخت پروگرامر

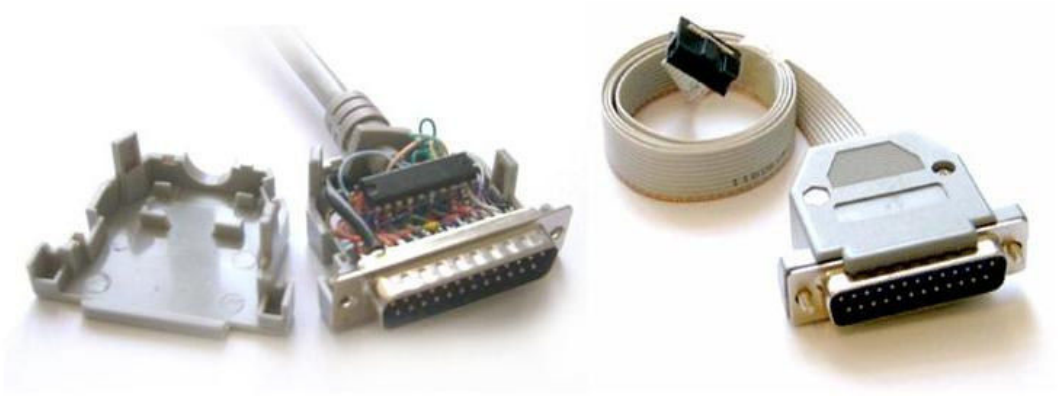
برای این که برنامه نوشته شده را از کامپیوتر به آی سی میکرونترلر انتقال دهیم ، می بایست از پروگرامر استفاده کنیم. پروگرامرها نیز با توجه به پورت هایی که پشتیبانی می کنند دارای انواع و استانداردهای مختلفی هستند.

یکی از ساده ترین نوع پروگرامر که می تواند به راحتی در منزل ساخته شود ، پروگرامر ISP (In System Programming) است که به استاندارد STK 200/300 نیز معروف است و در واقع در این مدار پورت پرینتر (LPT) کامپیوتر را توسط یک عدد آی سی بافر (به شماره 74HC244) به پایه های میکروکنترلر AVR وصل می کنیم. که بصورت مدار زیر است :

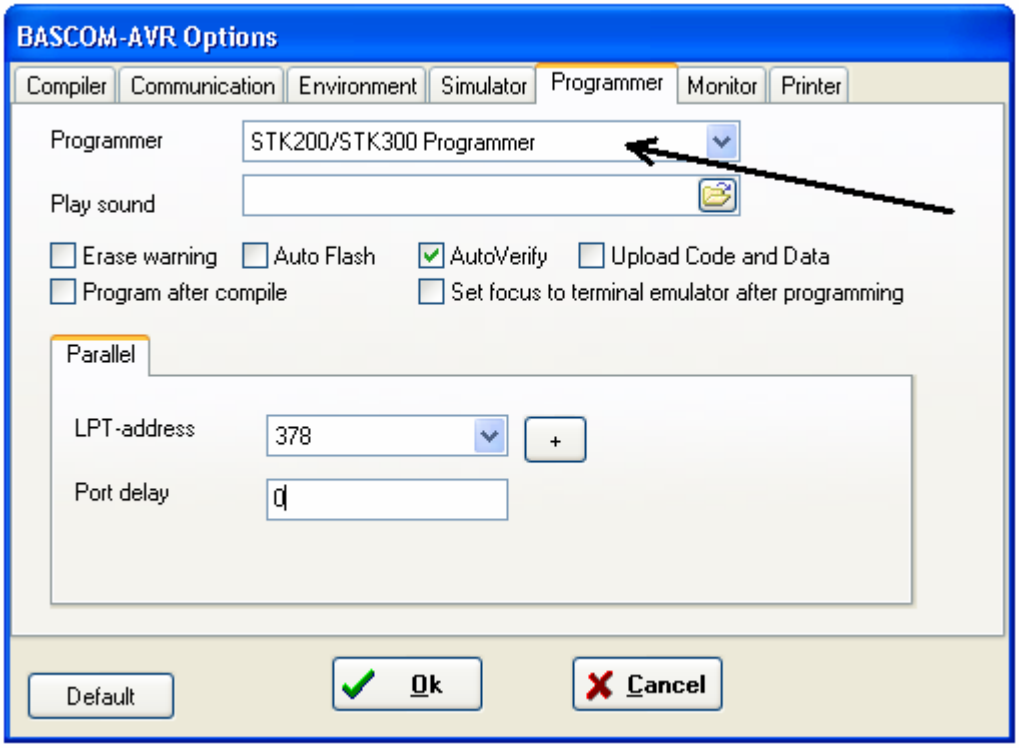
البته به این نکته توجه شود که کامپیوتر شما حتما می بایست دارای پورت پرینتر موازی باشد و در غیر اینصورت نمی توان از این مدار استفاده کرد و حتما می بایست یک پروگرامر سازگار با پورت USB از بازار تهیه نمود. در لب تاپ ها پورت پرینتر موازی وجود ندارد.



برای ساخت مدار بالا می توان از کابل مخصوص پرینتر یا کابل ۶ رشته ایی استفاده شود. مانند شکل زیر :



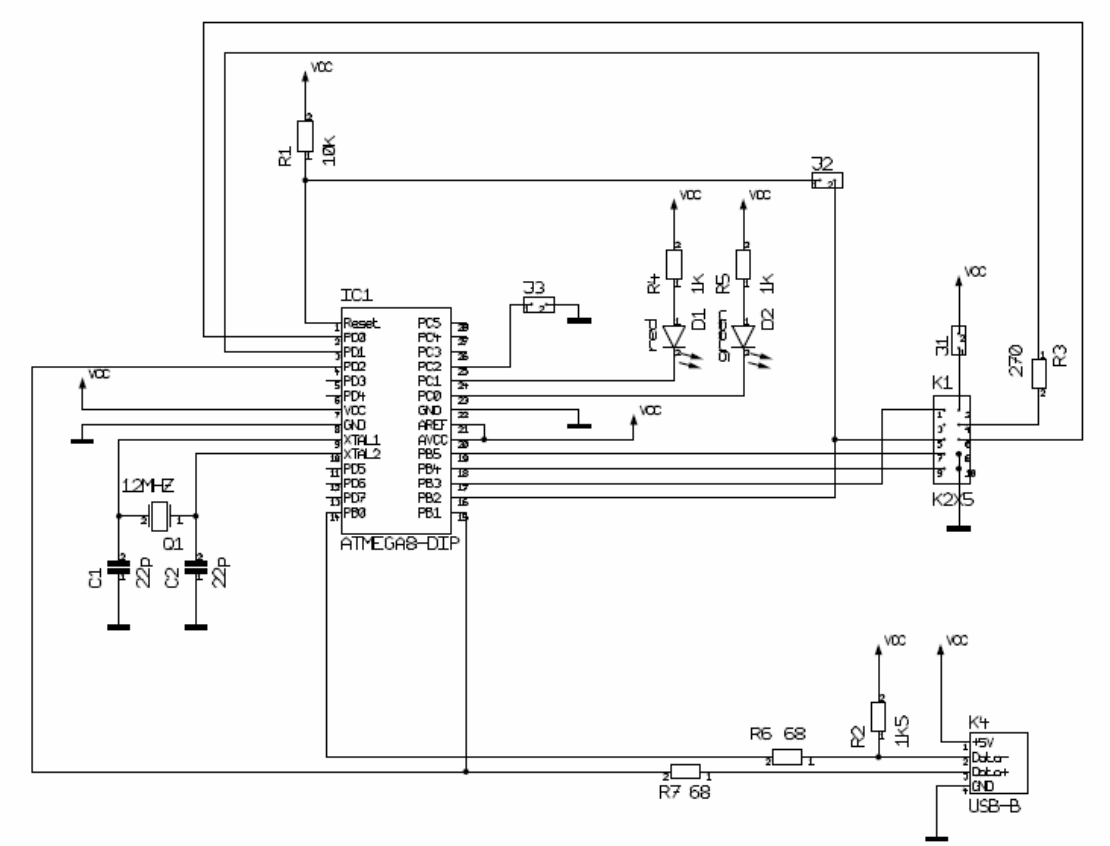
برای استفاده از این نوع پروگرامر در نرم افزار بیسکام به منوی Options رفته و گزینه Programmer را انتخـاب می کنیم و طبق شکل زیر داریم:



البته نوع دیگری از پروگرامرها در بازار وجود دارند که از پورت USB برای برقراری ارتباط با کامپیوتر استفاده کرده و با استاندارد STK 500 شناخته می شوند.

در ادامه نقشه یک نوع از پروگرام هایی که با پورت USB کار مـی کننـد آورده شـده و بایـد توجـه داشـت کـه در طراحی این نوع پروگرامر از میکروکنترلر AVR سری Mega8 استفاده می شود که خود آن بایـد در ابتـدا برنامـه ریزی شود. البته لازم به توضیح است که برنامه هگز میکروکنترلر این پروگرامر و نیز طریقه نصب و شناسایی آن بـه کامپیوتر، در فایل های ضمیمه این مجموعه قرار دارد. ساخت این پروگرامر به افراد مبتدی توصیه نمی شود.

نقشه پروگرامر USB :



نکته) به مبتدیان پیشنهاد می شود برای جلوگیری از سردرگمی و آسیب رساندن به میکروکنترلر ، از پروگرامرهای آماده موجود در بازار استفاده کنند.

ضمیمه ۲

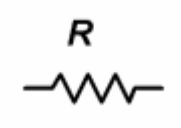
الکترونیک کاربردی

در این قسمت با برخی از فطعات و ابزارهای پرکاربرد در الکترونیک و نحوه عملکرد آنها آشنا می شوید.

۱- مقاومت (Resistor) :

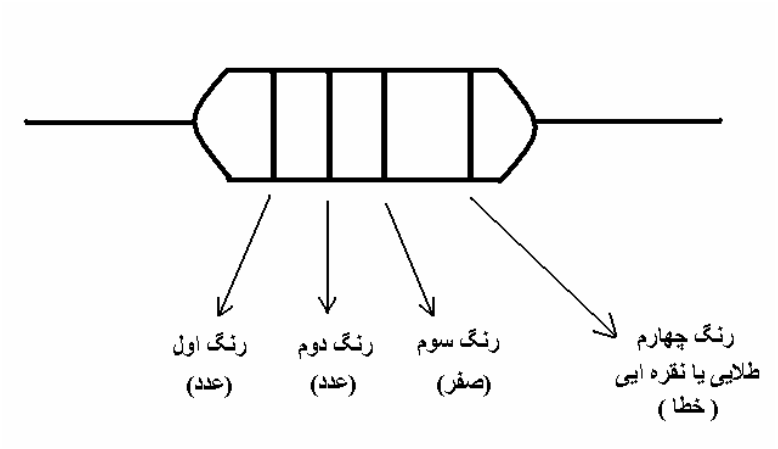
قطعه ایی الکتریکی بوده که در برابر عبور جریان مخالفت می کند. در واقع از ایـن قطعـه بـرای کـم کـردن ولتـاژ و جریان مورد نیاز قطعات الکترونیکی استفاده می شود زیرا قطعات الکترونیکی نمی توانند ولتاژها و جریان های زیـاد را تحمل کنند.

واحد مقاومت اهم (Ω) بوده و نماد آن در نقشه های الکترونیکی بصورت زیر است:



مقاومت ها در اندازه ها و شکل های مختلفی ساخته می شوند که برخی از آنها را در شکل زیر می بینید.

مقدار اهم هر مقاومت از طریق رنگ های موجود روی آن بدست می آید. در واقع هر رنگ نشان دهنده یک عدد می باشد که از کنار هم قرار گرفتن رنگ ها مقدار مقاومت محاسبه می شود. از طریق جدول زیر براحتی می توان مقدار اهم هر مقاومت را بدست آورد:



رنگ	کد
مشکی	0
قهوه ایی	1
قرمز	2
نارنجی	3
زرد	4
سبز	5
آبی	6
بنفش	7
خاکستری	8
سفید	9

به جای رنگ اول مقدار عدد آن را قرار می دهیم ، به جای رنگ دوم نیز مقدار عدد آن را قرار مـی دهـیم ، در رنـگ سوم به جای هر عددی که خواستیم قرار بدهیم به تعداد آن صفر می گذاریم.

بعنوان مثال اگر رنگ های مقاومتی به اینصورت بود : (سبز ، آبی ، قرمز ، طلایی) ، مقدار آن از روش زیـر بدسـت می آید:

رنگ اول = سبز = ۵

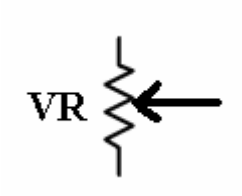
رنگ دوم = آبی = ۶

رنگ سوم = قرمز = ۲ (چون رنگ سوم است پس به جای آن دو تا صفر قرارمی دهیم)

رنگ چهارم = طلایی

رنگ چهارم نیز مشخص کننده مقدار خطای این مقاومت است که نشان می دهد مقدار ایـن مقاومـت بطـور دقیـق 5.6 KΩ نیست بلکه به مقدار ۵ درصد ممکن است کمتر یا بیشتر باشد.

مقاومت هایی که تا اینجا معرفی کردیم از نوع مقاومت های ثابت بودند یعنی مقدار آنها را نمی توان تغییر داد. ولی نوع دیگری از مقاومت ها وجود دارد که مقدار آن متغییر و قابل تنظیم است و به آن پتانسیومتر یا ولوم گفته می شود. در این نوع مقاومت ها می توان با چرخاندن پیچ آن مقدار اهم مورد نظر را بدست آورد. علامت مقاومت متغییر در نقشه های الکترونیکی بصورت زیر است:



همبنطور نوع دیگری از مقاومت های متغییر هستند که مقدار اهم آنها را نمی توان بصورت دستی تغییر داد بلکه به نوع ساختار آنها و با توجه به عوامل محیطی از قبیل:

دما (PTC و NTC)

نور (LDR)

میدان مغناطیسی (MDR)

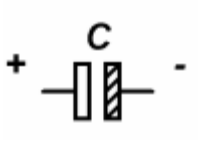
ولتاژ (VDR)

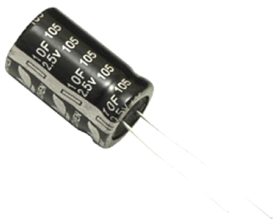
تغییر می کند.

۲- خازن (Capacitor) :

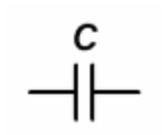
قطعه ای الکتریکی ست که می تواند ولتاژ را در خود بصورت میدان الکترواستاتیکی ذخیره کند. واحد ظرفیت خازن فاراد (F) می باشد ولی چون در مدارهای الکترونیکی از خازن هایی با ظرفیت بسیار کوچک استفاده می شود واحد آنها در حد میکروفاراد ، نانوفاراد و پیکوفاراد است. خازن در دو نوع غیر الکترولیت (شیمیایی) و غیر الکترولیت (سرامیک یا عدسی) وجود دارد.

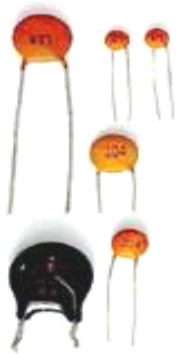
خازن ها در نوع الکترولیت معمولا به شکل استوانه ایی ساخته می شوند دارای دو قطب مثبت و منفی هستند که باید حتما رعایت شوند (پایه کوتاهتر همیشه منفی است و نیز روی بدنه خازن علامت منفی وجود دارد). واحد آنها میکروفاراد (uF) می باشد. نماد فنی آن نیز به شکل زیر است:





خازن های نوع غیر الکترولیت دارای قطب مثبت و منفی نبوده و واحد ظرفیت آنها در حد نانوفاراد (nF) و پیکوفاراد (pF) است. نماد فنی آن نیز به صورت زیر می باشد:





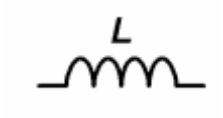
کاربرد خازن ها در انواع فیلترهای حذف نویز ، کوپلاژ سیگنال ، بای پس سیگنال ، صافی ولتاژ و ... می باشد.

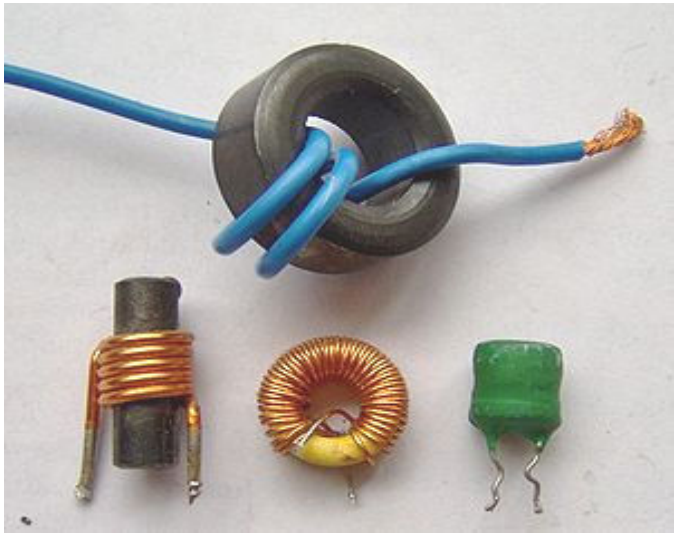
بر روی بدنه خازنهای الکترولیتی مقدار ظرفیت و ولتاژ آنها نوشته شده ولی مقدار ظرفیت خازن های غیرالکترولیت را بصورت یک عد سه رقمی می نویسند که به جای رقم سوم می بایست به تعداد آن صفر گذاشته شود. مثلا اگر روی بدنه خازنی عدد 104 نوشته شده بود ، مقدار آن بصورت زیر می شود:

$104 = 100000$ pF = 100 nF

۳- سلف (Inductance) :

قطعه ایی الکتریکی که جریان الکتریکی را بصورت میدان الکترومغناطیسی در خود ذخیره می کند. در واقـع هرگـاه چند دور سیم (روکش دار) را دور استوانه ایی بپیچیم سلف درست می شود. واحد سلف هانری (H) بوده و نمـاد آن در نقشه های الکترونیکی بصورت زیر است:





کاربرد سلف در وسایلی چون موتورهای الکتریکی، بلندگوها، ترانسفورماتورها و نیز در مدارهایی کـه نیـاز بـه تولیـد ولتاژ بالایی داریم می باشد. ونیز در مدارهای مخابراتی بعنوان نوسان ساز و مدارهای فیلتر برای حذف نویز بـه وفـور کاربرد دارند.

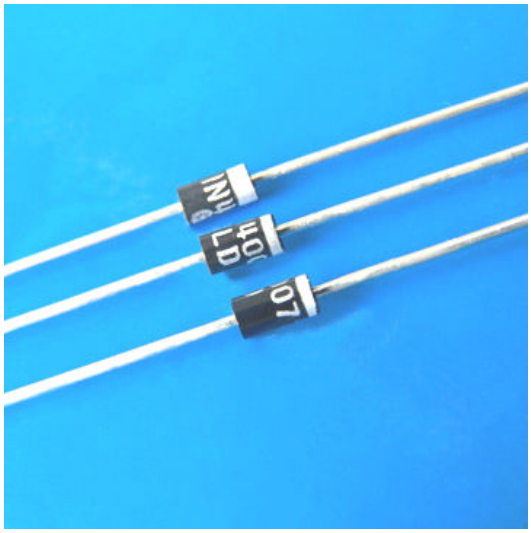
سلف های اندوکتانس پایین را گاهی به شکل ظاهری مقاومـت و خـازن هـم مـی سـازند کـه بایـد توجـه کنـیم در شناسایی آنها اشتباه نکنیم.

در شکل زیر نماد ترانسفورماتور و سلف هسته دار در نقشه های الکترونیکی را مشاهده می کنید:

۴- دیود (Diode) :

قطعه ایی الکترونیکی ست که جریان را فقط از یک طرف عبور می دهد. دارای دو پایه به نام های آند (A یا قطب مثبت) و کاتد (K یا قطب منفی) می باشد که جریان همیشه از طرف آند وارد و از طرف کاتد خارج می شود.

نماد فنی دیود به شکل زیر است:



در شکل بالا اگر به بدنه دیودها توجه کنید در یک سمت هر دیود یک نوار خاکستری رنگ وجود دارد که نشان دهنده پایه کاتد می باشد.

دیودها از مواد نیمه هادی مثل سیلیسیوم و ژرمانیوم ساخته می شوند. در ۲ سر هر دیود زمانی که جریان از آن عبور می کند ولتاژی در حدود بین 0.2V الی 0.7V افت می کند.

برای تست دیود از وسیله ایی به نام مولتی متر دیجیتال استفاده می کنیم. به اینصورت ابتدا سلکتور مولتی متر را روی حالت تست دیود (تست بوق) قرار داده سپس سیم های مولتی متر را به ترتیب به پایه های دیود متصل می کنیم. اگر سیم قرمز مولتی متر را به پایه آند و سیم مشکی رنگ را به پایه کاتد دیود متصل کنیم می بایست عددی بین ۲۰۰ الی ۷۰۰ روی صفحه مولتی متر ظاهر شود و از طرف دیگر نباید این مقدار را نشان دهد که این امر نشان دهنده سالم بودن دیود است. در غیر اینصورت و یا در صورتی که مولتی متر بوق بزند نشان می دهد که دیود سوخته است.

دیودها در انواع و اندازه های مختلفی که در مدارهای مختلف کاربردهای گوناگونی را دارند ساخته می شوند.

مثلا دیودهای نورانی (LED) یا دیودهای مادون قرمز (IR) که قبلا به آنها اشاره شده است.

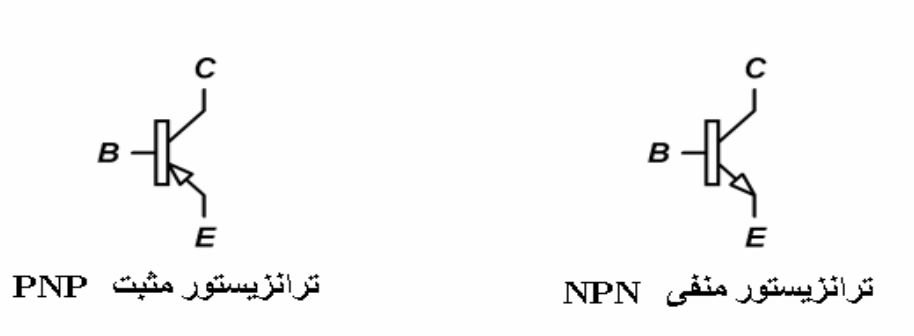
۵- ترانزیستور (Transistor) :

قطعه ایی الکترونیکی که دارای دو کاربرد مهم در مدارهای الکترونیکی می باشد:

الف) کاربرد بعنوان تقویت کنند جریان و ولتاژ

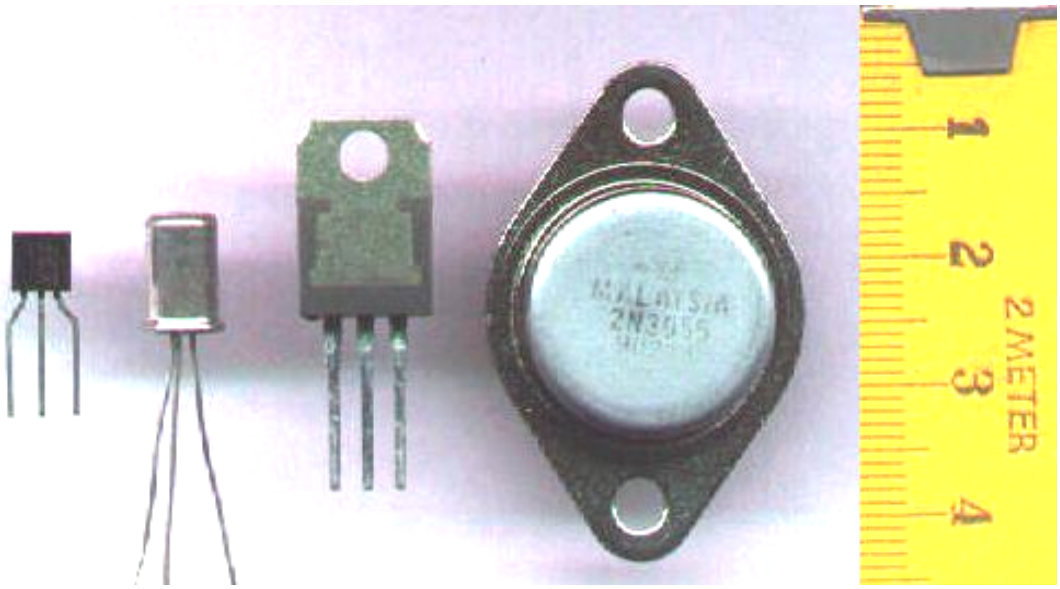
ب) کاربرد بعنوان کلیدزنی پرسرعت (سوئیچینگ)

ترانزیستورها در دو نوع مثبت (PNP) و منفی (NPN) ساخته می شوند و نماد فنی آنها بصورت زیر است:



همانطور که در شکل بالا مشاهده می شود ، هر ترانزیستور دارای سه پایه به نام های B (بیس) ، C (کلکتـور) و E (امیتر) می باشد. که باید به ترتیب قرار گیری آنها در مدارهای الکترونیکی دقت شود. در مـدارهای الکترونیکـی معمولا پایه بیس معمولا بعنوان ورودی ، پایه کلکتور بعنوان خروجی و پایه امیتر به یکی از قطب های باطری وصـل می شود. (در نوع NPN پایه امیتر به قطب منفی باطری و در نوع PNP پایه امیتر به قطب مثبت بـاطری متـصل می شود)

در شکل زیر انواع مختلفی از ترانزیستورها مشاهده می شود:

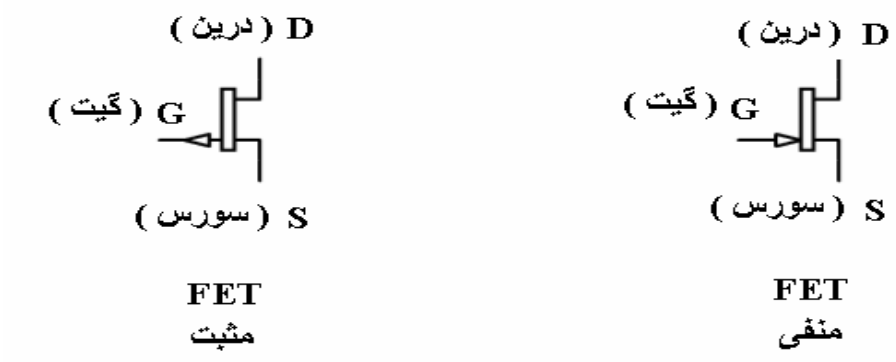


برای تست و شناسایی ترتیب پایه های یک ترانزیستور همانند دیودها از وسیله ایی به نام مولتی متر دیجیتال استفاده می شود. به اینصورت ابتدا سلکتور مولتی متر را روی حالت تست دیود (تست بوق) قرار داده سپس سیم های مولتی متر را به ترتیب به پایه های ترانزیستور متصل می کنیم ، فقط در یک حالت مشاهده می کنیم روی صفحه مولتی متر عددی در حدود بین ۵۰۰ الی ۷۰۰ مشاهده می شود یعنی یک پایه ترانزیستور نسبت به دو پایه دیگر این حالت را نشان می دهد (به اصطلاح یک پایه به دو پایه دیگر راه می دهد) که همان پایه ، بیس ترانزیستور می باشد. حالا اگر سیم قرمز به این پایه وصل بود ، ترانزیستور از نوع منفی (NPN) می باشد و اگر سیم مشکی بود ترانزیستور از نوع مثبت (PNP) می باشد. دو پایه دیگر امیتر و کلکتور می باشند. برای تشخیص این دو پایه کافیست به عدد نشان داده شده روی صفحه مولتی متر توجه کنید ، عدد کمتر نشان دهنده پایه کلکتور و عدد بیشتر نشان دهنده پایه امیتر می باشد.

ترانزیستورها هم مثل دیودها از ماده ایی به نام سیلیسیوم و ژرمانیوم ساخته می شوند و باید دقت کنیم ولتاژهای زیاد باعث سوختن این قطعه حساس می شود.

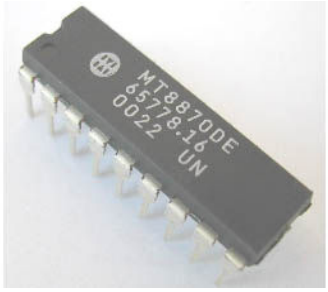
هر ترانزیستور یک شماره مختص به خود را دارد که برای کاربرد خاصی تعریف شده است و در واقع نمی توان آنها را به جای هم در مدارهای الکترونیکی به کار گرفت. بعنوان مثال ترانزیستوری با شماره C945 از نوع دوقطبی منفی بوده و دارای کاربردهای عمومی در حد توان پایین است.

ترانزیستورهایی که در بالا اشاره شد از نوع دوقطبی (BJT) بود ولی انواع دیگری از ترانزیستورها وجود دارند که اثر میدان (FET) یا (MOSFET) نامیده می شوند و بیشتر در مدارهای دیجیتالی کاربرد دارند. ترتیب نام گذاری پایه های این نوع ترانزیستورها با نوع قبلی متفاوت است.



۶ – آی سی (IC) :

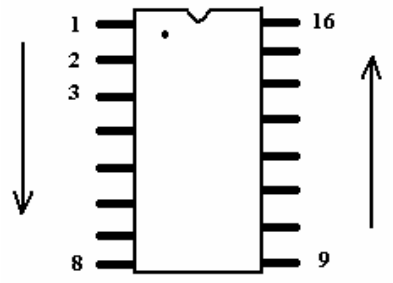
قطعه ایی الکترونیکی بوده و در واقع یک مدار الکترونیکی کامل شامل تعداد زیادی ترانزیستور ، دیود ، خازن ، مقاومت و ... را در خود جای داده است. در واقع برای کوچک کردن مدارهای الکترونیکی و برای صرفه جویی در وقت و هزینه از این قطعه استفاده می شود. کلمه آی سی مخفف عبارت Integrated Circuit می باشد که به معنی مدار مجتمع است. نماد فنی یک آی سی بر روی نقشه های الکترونیکی معمولا بشکل یک مربع یا مثلث است که پایه های آن با شماره مشخص شده اند.



آی سی ها کاربرد فراوانی در ساخت مدارهای الکترونیکی دارند و در واقع برای هر کاربرد یک آی سی مختص به آن در انواع و اندازه های مختلفی طراحی و ساخته می شود. تعداد پایه های هر آی سی ممکن است سه الی ده ها و صد ها پایه باشد که هر کدام آنها مختص یک کار است. برای هر آی سی شماره مختص به آن در نظر گرفته شده که مشخص کننده نوع و کاربرد آن می باشد. برای همین نمی توان آی سی ها را مثل دیودها و ترانزیستورها با مولتی متر تست کرد و پایه های آن را تشخیص داد. رای شناخت یک آی سی و نحوه عملکرد و نیز شناسایی پایه های آن می بایست به دیتاشیت (صفحه راهنما) هر آی سی که توسط شرکت سازنده آن ارائه می شود مراجعه نمود. (در اینترنت می توان به دیتاشیت کلیه آی سی ها دسترسی داشته باشیم)

همه آی سی ها به دو دسته آنالوگ و دیجیتال تقسیم بندی می شوند که در مدارهای مربوط به خودشان کاربرد دارند. معروفترین نوع آی سی های دیجیتال دو سری TTL و CMOS می باشند که به ترتیب با شماره های 74xx و 40xx شناسایی می شوند. معروف ترین سری آی سی های آنالوگ نیز Opamp ها هستند که بعنوان تقویت کننده کاربرد دارند.

ترتیب شماره گذاری پایه های همه آی سی ها بصورت زیر است (به شکاف U شکل و نقطه روی آی سی توجه کنید) :

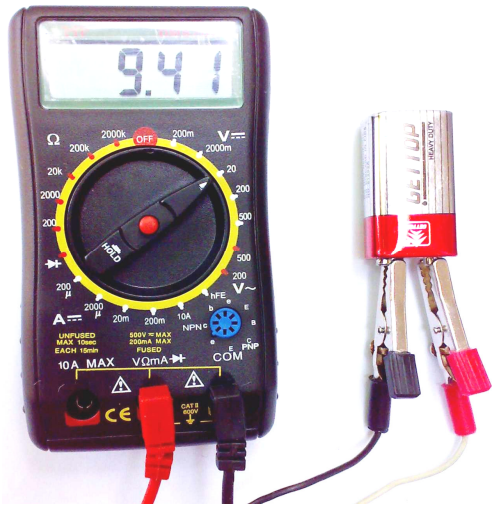


آشنایی با مولتی متر:

یکی از دستگاه های اندازه گیری بوده که می تواند مقدار اهم ، ولت و آمپر را اندازه گیری نماید. البته علاوه بـر ایـن موارد با مولتی متر می توانیم تست اتصال (تست بوق) ، تست قطعات نیمه های چون دیود و ترانزیستور ، تـست و اندازه گیری مقدار ظرفیت خازن و ... را هم انجام دهیم.

مولتی مترها بر دو نوع آنالوگ (عقربه ایی) و دیجیتال موجود می باشند که در شـکل زیـر آن هـا را مـشاهده مـی کنید:

مولتی متر آنالوگ



مولتی متر دیجیتال

برای استفاده از مولتی متر در ابتدا سیم های آن را می بایست به محل مشخص شده روی آن متصل نمود (سیم مشکی رنگ همیشه به محل Com وصل می شود) و سپس آن را روشن می کنیم و برای اندازه گیری و یا تست قطعات می بایست کلید سلکتور روی مولتی متر را در محل مشخص شده آن قرار دهیم و حتما توجه کنیم که در انتخاب کلید سلکتور اشتباه نکنیم چون ممکن است به مولتی متر آسیب وارد شود.

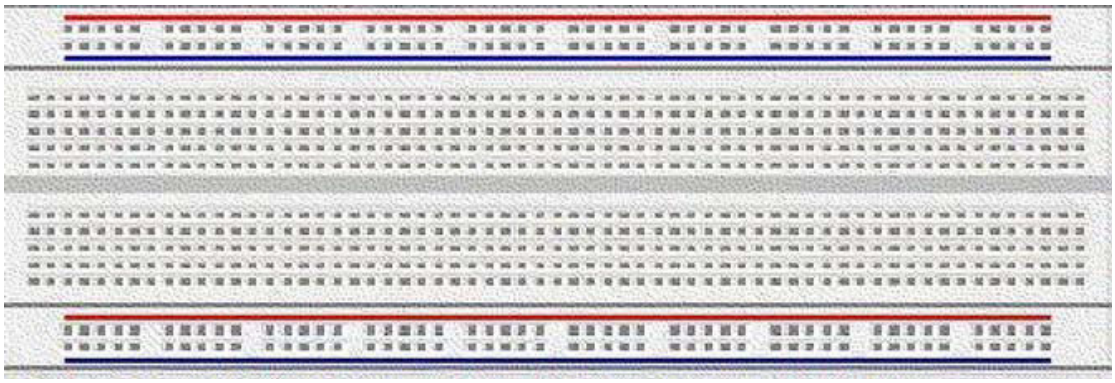
نوع دیگری از مولتیمترهای دیجیتال وجود دارند که بیشتر در میزهای آزمایشگاهی الکترونیک مورد استفاده قرار می گیرند:

مولتی متر رومیزی

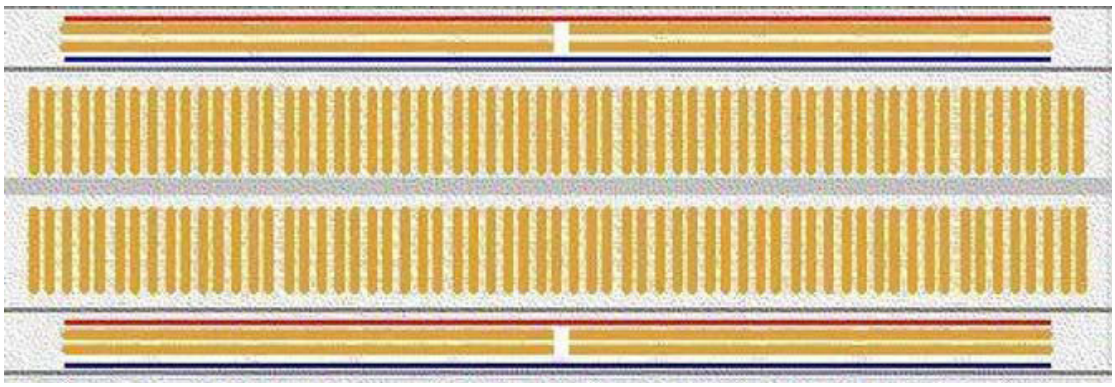
آشنایی با بردبرد :

برای این که بتوانیم مدارهای الکترونیکی را بدون نیاز به تهیه فیبر مدارچاپی و لحیم کاری ، مونتاژ کنیم از وسیله ایی به نام بردبرد استفاده می کنیم. از مزیت های بردبرد می توان به نصب سریع قطعات روی آن و نیز جدا کردن آن ها برای مونتاژ یک مدار دیگر بدون آسیب رسیدن به قطعات اشاره کرد که در زمان و هزینه باعث صرفه جویی قابل ملاحظه ایی می شود.

نمای یک بردبرد به شکل زیر است:

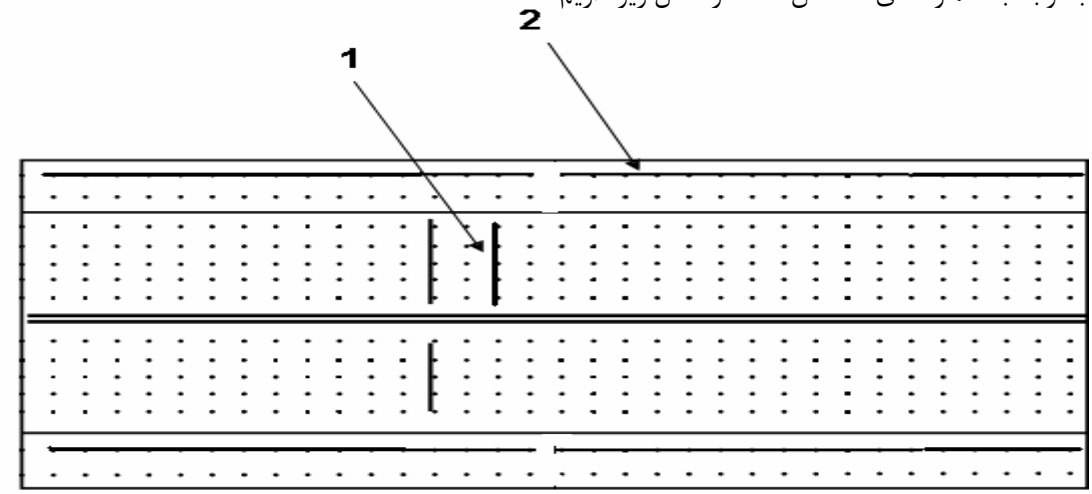


اتصالات داخلی یک بردبرد به شکل زیر است:



به نحوه اتصال داخلی پین های بردبرد توجه شود تا در هنگام مونتاژ مدارها دچار مشکلاتی از قبیل اتصالی یا عدم اتصال بین قطعات نشوید.

با توجه به شماره های مشخص شده در شکل زیر داریم:

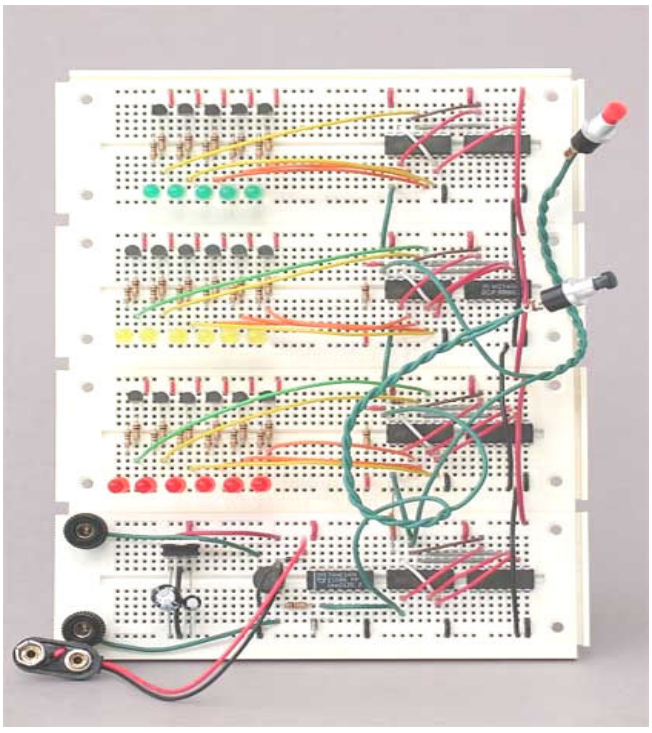


۱- این پین ها بصورت شیارهای عمودی به هم متصل هستند. ولی به شیار کناری هیچ ارتباطی ندارند.

۲- این پین ها بصورت شیارهای افقی به هم متصل هستند.(البته تا محل مشخص شده)

نکته : بدلیل وجود خاصیت خازنی بین پین های برد بورد در فرکانس های بالا ، مدارهای مخابراتی نظیر فرستنده و گیرنده را نباید روی این بردها مونتاژ کنیم.

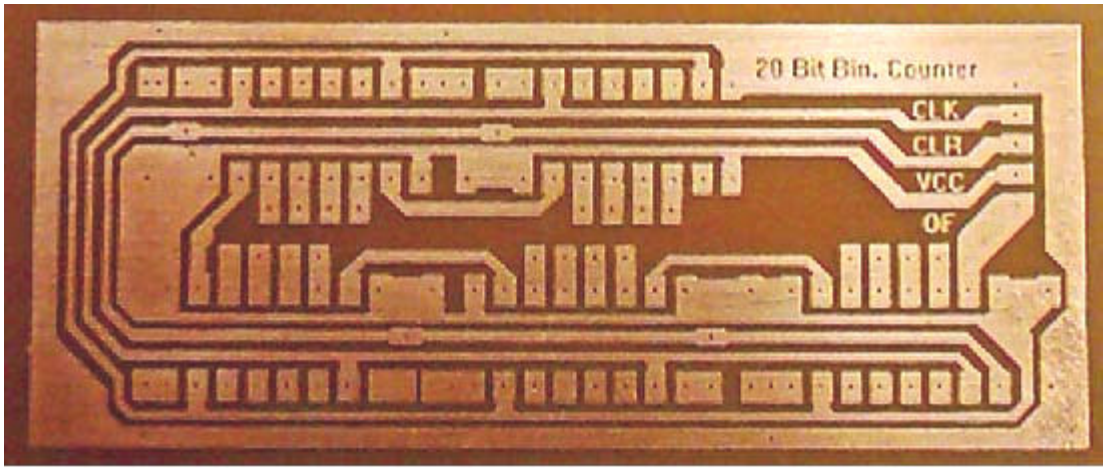
نمونه یک مدار الکترونیکی مونتاژ شده بر روی چند بردبرد:



آشنایی با فیبر مدارچاپی :

برای ساخت و مونتاژ یک مدار الکترونیکی با اتصالات دائمی از فیبرهای مدار چاپی که اصطلاحا PCB نامیده می شود استفاده می کنیم. در این روش کلیه قطعات و کانکتورها و سیم های ارتباطی می بایست توسط لحیم کاری به فیبر مدارچاپی متصل شوند. و دیگر نمی توان آن را مثل بردبرد براحتی باز نمود.

فیبرهای مدار چاپی از نوعی کاغذ فشرده ساخته می شوند که یک طرف آن لایه مس وجود دارد و در واقع برای اتصال بین قطعات از این لایه مسی استفاده می شود که همان نقش سیم های ارتباطی بین قطعات را دارد.

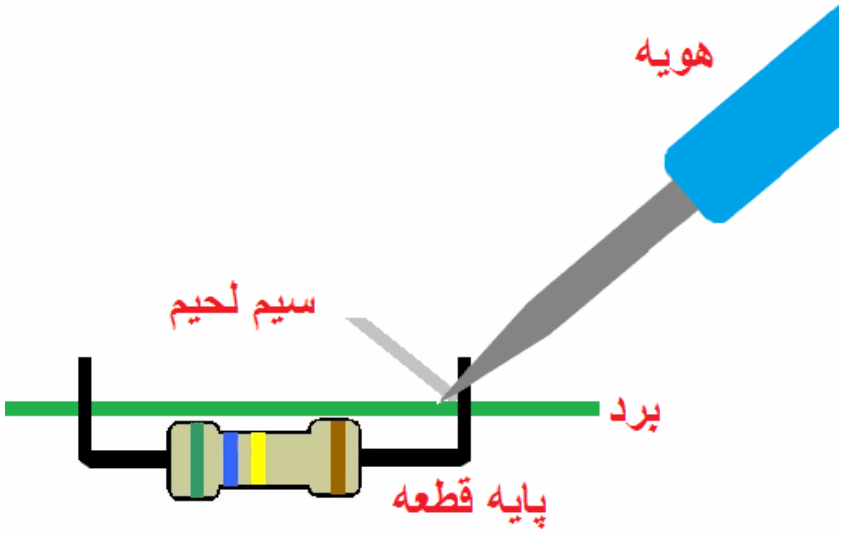


برای تهیه PCB به روش دستی می بایست در ابتدا طرح مدار مورد نظر را روی فیبر مدار چاپی که یک طرف آن کاملا مس قرار دارد ایجاد کنیم. برای این کار از ماژیک ضد آب و یا لتراست استفاده می کنیم. سپس با استفاده از اسید مخصوص مدارچاپی که حلال مس می باشد محلولی از آب جوش و اسید به نسبت ۲ به یک در ظرفی غیر فلزی درست کرده ، فیبر را در داخل ظرفی پلاستیکی می اندازیم. پس از گذشت مدت زمانی در حدود نیم ساعت مس موجود بر روی فیبر در محلول حل شده و فقط قسمتی که توسط ماژیک یا لتراست طراحی شده بود باقی می ماند. حال فیبر را با آب شسته و سپس طرف مسی آنرا با سمباده ایی نرم تمیز می کنیم سپس با استفاده از مته سه نظام محل قرارگیری پایه قطعات را سوراخ کاری کرده و نسبت به لحیم کاری آن اقدام می کنیم.

برای لحیم کاری قطعات الکترونیکی توجه شود تا از هویه های وات پایین (حدود ۴۰ وات) استفاده کنیم. نوک هویه باید تیز و تمیز باشد تا در هنگام لحیم کاری باعث ایجاد مشکل نشود.

در شکل زیر تصویر یک هویه مناسب مخصوص لحیم کاری مدارات الکترونیک را مشاهده می کنید:

برای لحیم کاری قطعات الکترونیکی بر روی PCB ابتدا می بایست سطح مسی فیبر را با یک سمباده نرم کاملا تمیز کنیم سپس قطعات را در جای مناسب قرار داده و بصورت شکل زیر لحیم کاری را انجام می دهیم:



همانطور که در شکل بالا مشاهده می شود سیم لحیم و نوک هویه را همزمان به پایه قطعه نزدیک کرده تا سیم لحیم ذوب شود و قلع مذاب اطراف پایه را فرا بگیرد. سپس هویه و سیم لحیم را از اطراف پایه قطعه دور کرده کمی صبر می کنیم تا قلع خنک شود و در این هنگام پایه قطعه به فیبر متصل شده و نباید از جای خود تکان بخورد.

در هنگام لحیم کاری توجه شود مدت اتصال هویه به پایه قطعات بیشتر از ۵ ثانیه نشود زیرا باعث معیوب شدن قطعه می گردد. سیم لحیم هم از جنس مرغوب انتخاب شود. اضافه پایه قطعات را با استفاده از سیم چین کوتاه می کنیم.

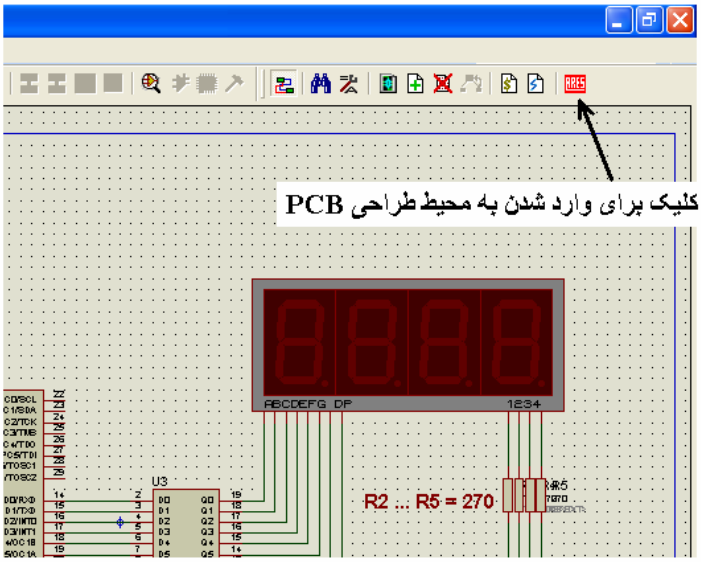
ضمیمه ۳

تهیه PCB با استفاده از نرم افزار پروتئوس

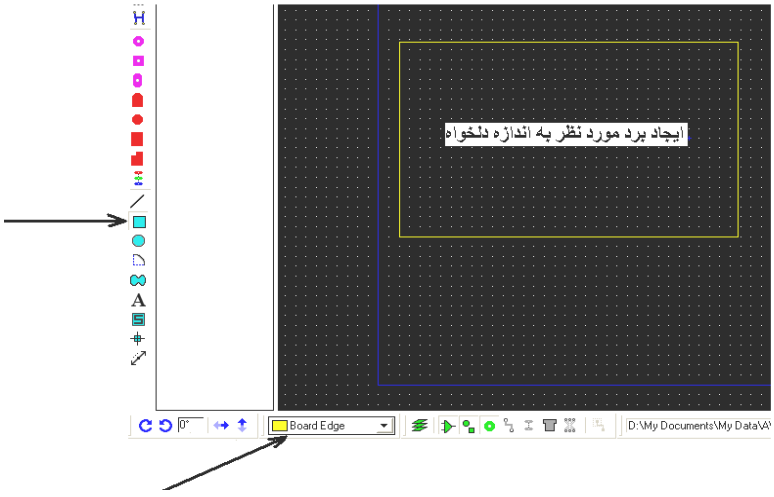
و روش پرینت – اتو

یکی دیگر از روش های تهیه PCB با استفاده از نرم افزارهایی چون پروتل و پروتئوس می باشد که در اینجا نرم افزار پروتئوس را مورد بررسی قرار می دهیم.

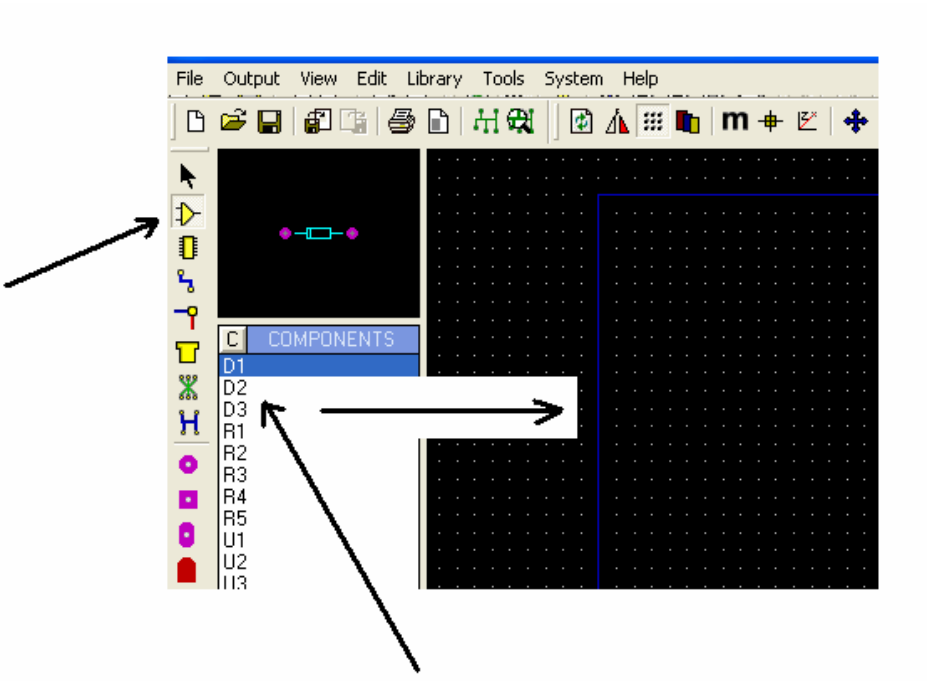
بعد از اینکه مدار مورد نظر خود را در محیط نرم افزار پروتئوس شبیه سازی کردیم و نتیجه مطلوب را گرفتیم روی آیکون ARES کلیک می کنیم تا به محیط طراحی PCB این نرم افزار وارد شویم.



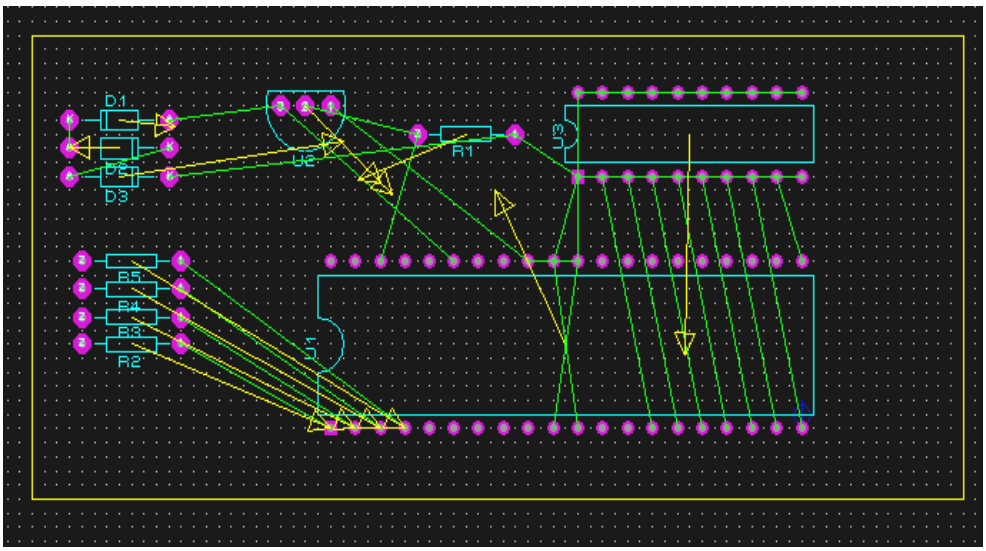
اول با استفاده از ابزار های سمت چپ اندازه ، ابعاد برد مورد نظر را تعیین می کنیم. به شکل زیر توجه کنید:



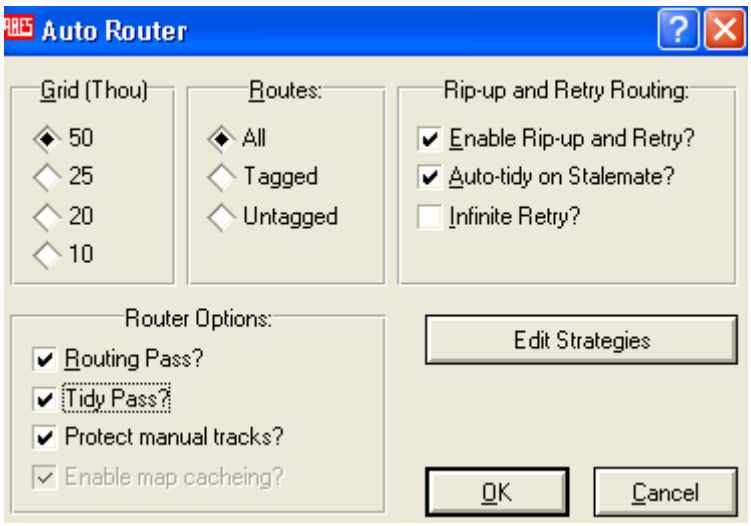
سپس قطعات را به ترتیب انتخاب کرده و آنها را در جای مناسب و دلخواه خود قرار می دهیم.



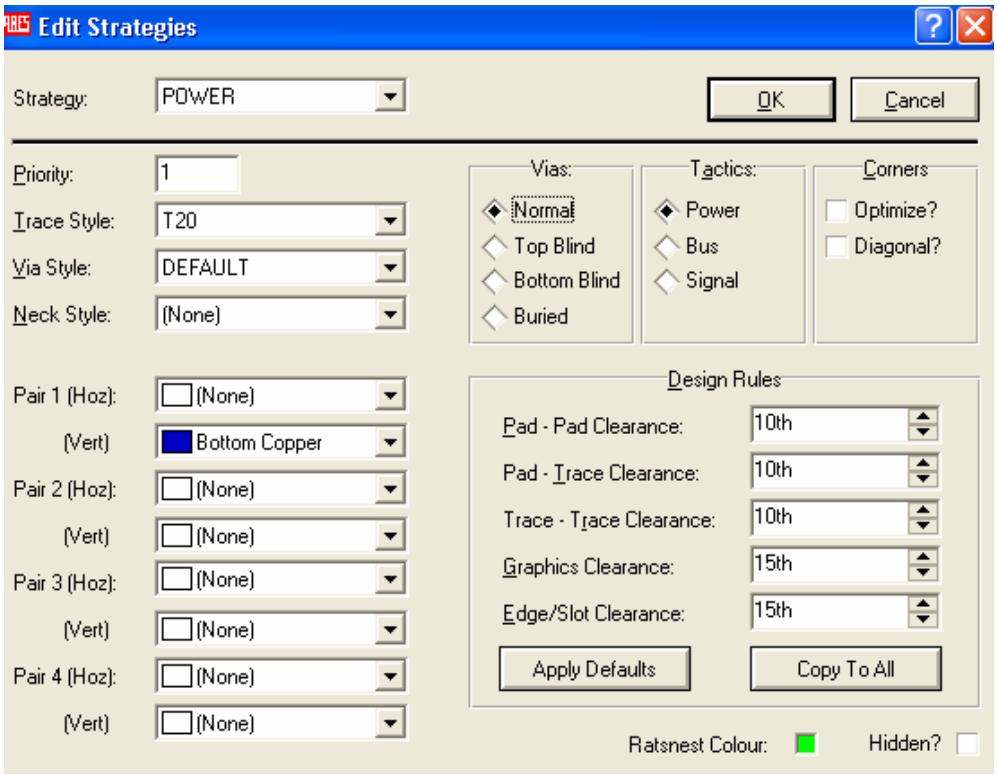
البته به روش اتوماتیک هم می توان قطعات را در جای مناسب خودشان قرار دهیم. برای این کار بـه منـوی Tools رفته و گزینه Auto Placer را انتخاب می کنیم. قطعات مانند شکل زیر در جای خود قرار می گیرند که می توانیم جای آنها را با توجه به سلیقه خود تغییر بدهیم.



مرحله بعد کشیدن خطوط بین قطعات است که برای این کار از منـوی Tools گزینـه Auto Router را انتخـاب می کنیم. پنجره زیر باز می شود که تنظیمات را مانند آن قرار دهید:



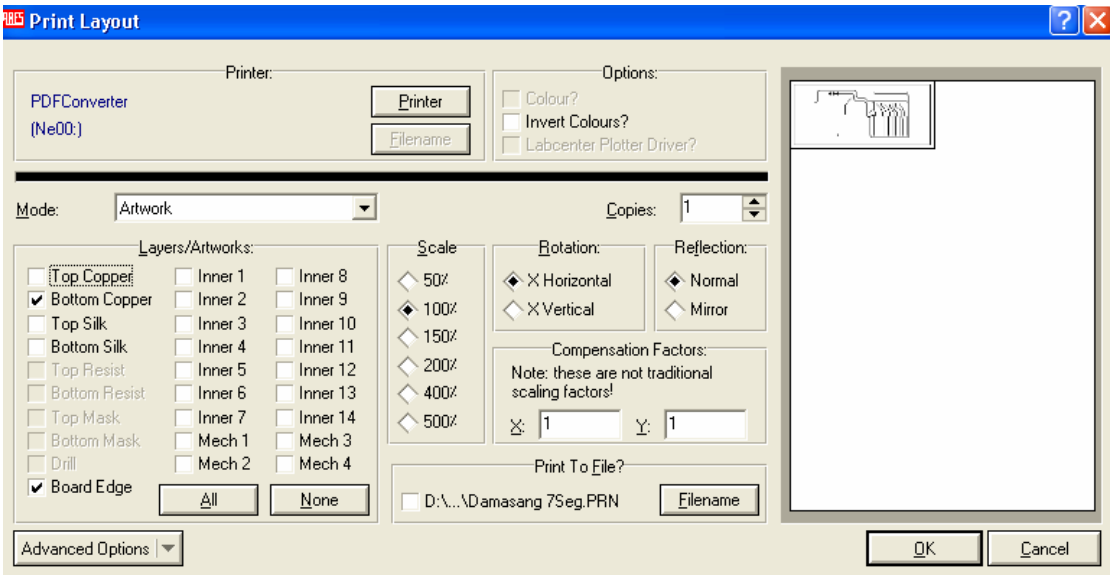
سپس برای تنظیمات دقیق تر روی دکمه Edit Strstgies کلیک کرده تا پنجره زیر باز شود و تنظیمـات را ماننـد آن قرار می دهیم:



حالا روی گزینه OK در هر دو پنجره باز شده کلیک کرده مشاهده می کنیم سیم کشی بین قطعات انجام می شود.

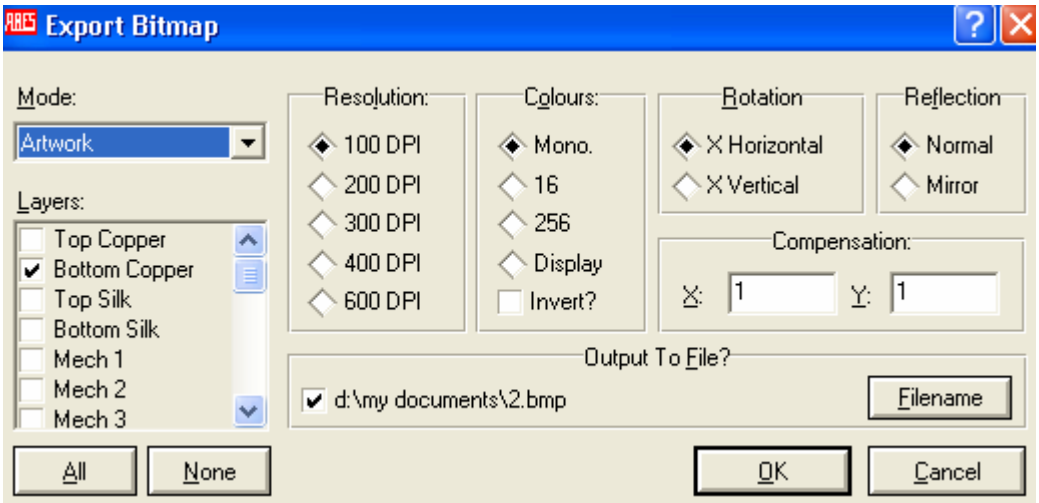
البته به روش دستی هم می توان بین قطعات سیم کشی انجام داد.

برای تهیه فایل پرینت به منوی Output رفته و گزینه Print را انتخاب می کنیم. پنجره زیر باز می شود ، تنظیمات را مانند آن قرار می دهیم:



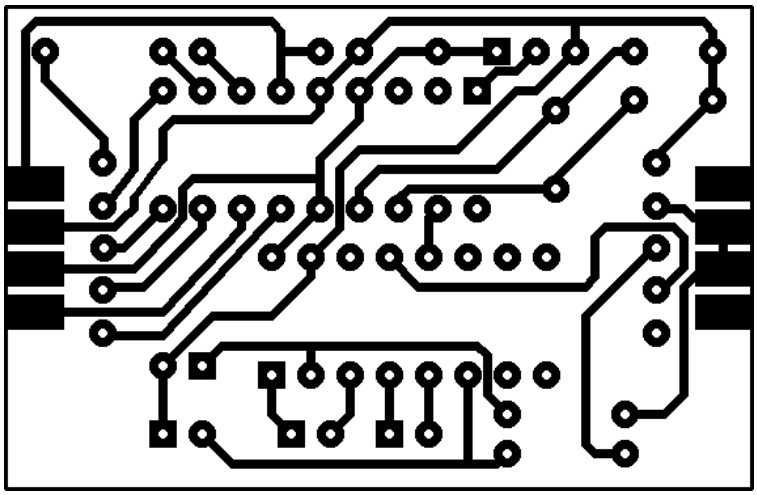
با کلیک کردن دکمه OK طرح PCB مورد نظر توسط پرینتر روی کاغذ چاپ می شود.

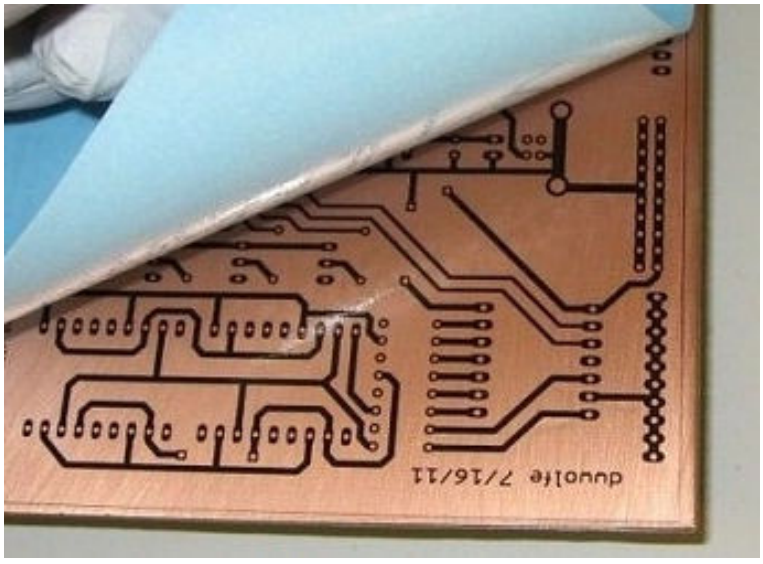
اگر دستگاه پرینتر در دسترس نبود می توان فایل عکس PCB را با فرمت bmp. تهیه کرد و در فرصتی مناسب آن را در جای دیگری بدون نیاز به نرم افزار پروتئوس چاپ نمود. برای این کار از منوی Output گزینه Export Bitmap را انتخاب می کنیم و تنظیمات پنجره باز شده را مانند شکل زیر قرار می دهیم:



توجه شود در قسمت Filename می توان محل ذخیره شدن فایل را مشخص نمود.

حالا برای اینکه طرح مورد نظر را روی فیبر مدار چاپی پیاده سازی کنیم در ابتدا لازم است که PCB مـورد نظـر را حتما توسط پرینتر لیزری بر روی کاغذ گلاسه چاپ کرده و سـپس آن را روی سـطح مـسی فیبـر قـرار مـی دهـیم بطوری که طرح مدار چاپی دقیقا روی مس قرار بگیرد. سپس اتو داغ را روی آن قرار داده تا طرح کـاملا روی سـطح مسی فیبر چاپ شود. حالا فیبر را در داخل ظرف آب گرم قرار داده تا کاغذ به کلی از فیبر جدا شود. در مرحله آخـر هم می بایست فیبر را در داخل محلول اسید مخصوص مدارچاپی و آب جوش قرار دهیم تا طرح مورد نظـر بـر روی فیبر باقی بماند. اکنون پس از شستشوی فیبر نوبت به سوراخ کاری و سپس لحیم کاری قطعات فرا می رسد.





ضمیمه۴

نحوه درایو و راه اندازی ادوات خارجی

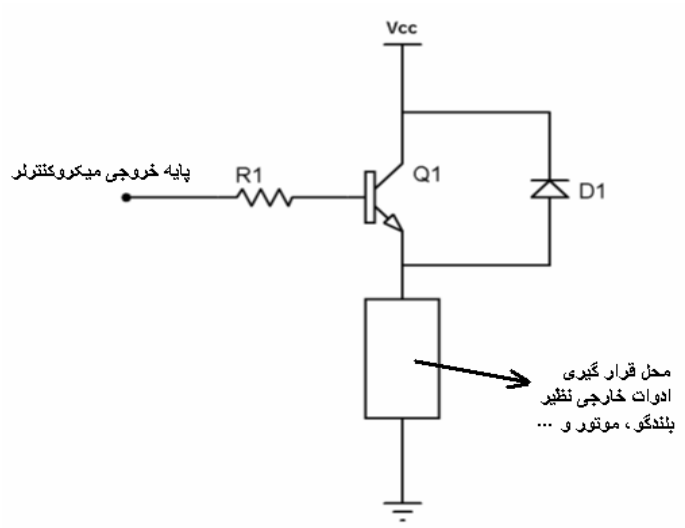
توسط میکروکنترلر

همانطور که قبلا هم اشاره شده به دلیل جریان بسیار کمی که در پایه های خروجی میکروکنترلرها داریم ، عملا نمی توانیم اد.واتی خارجی پر مصرف نظیر بلندگو ، موتورها ، دات ماتریکس ها و ... را توسط میکرو راه اندازی کنیم ، در واقع جریان هر پایه میکروکنترل فقط می تواند یک LED را روشن کند. بنابراین برای راه اندازی سایر قطعات پر مصرف توسط میکروکنترلرها می بایست از مدارهایی که اصطلاحا درایور نامیده می شوند و بین میکروکنترلر و قطعه ایی که قرار است توسط میکرو کنترل شود قرار می گیرد تا بتواند جریان مصرفی آن را تامین نماید.

در ادامه به بررسی چند روش مختلف درایو کردن ادوات خارجی توسط میکرو کنترلر می پردازیم.

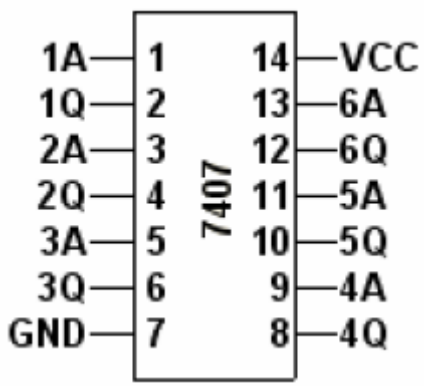
۱- استفاده از ترانزیستور :

یکی از ساده ترین راه ها برای درایو کردن ادوات خارجی ، استفاده از ترانزیستور می باشد که هم به عنوان بافر (تقویت جریان) و هم به عنوان سوئیچینگ (کلید زنی پر سرعت) می توان از آن استفاده نمود. و معمولا به شکل زیر در مدار قرار می گیرد:



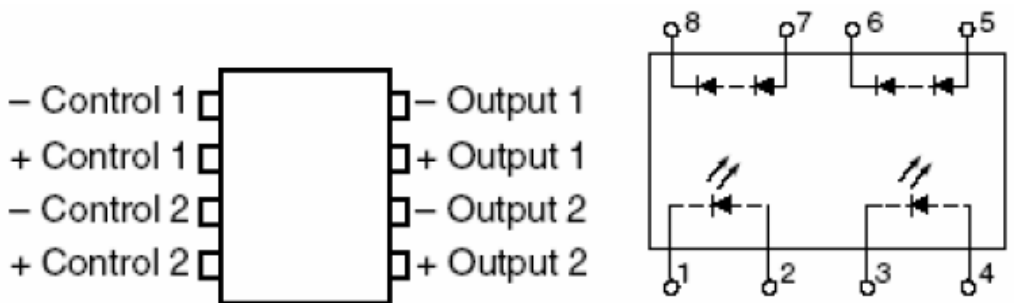
۲- استفاده از آی سی بافر :

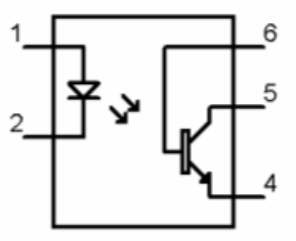
یکی دیگر از روش های درایو کردن ، استفاده از آی سی های بافر می باشد که در جریان های پایین مـورد اسـتفاده قرار می گیرند. انواع مختلفی از آی سی های بافر در نوع TTL و CMOS وجود دارند که یک نمونه از آن در شکل زیر مشاهده می شود:



۳- استفاده از اُپتوکوپلر (Opto Coupler) :

این قطعات که به جداکننده نوری معروفند به شکل آی سی در بازار موجودند و برای ایزوله کـردن دو قـسمت یـک مدار به کار می رود. جدا کننده های نوری به این صورت عمل می کنند که در داخل آنها معمولا از دیود حساس بـه نور ، ترانزیستور حساس به نور و یا SCR های نوری استفاده شـده و در مقابـل آن دیـود نـورانی قـرار دارد کـه بـا تحریک دیود نورانی توسط پایه های ورودی این قطعه توسط میکروکنترلر ، خروجی آن مثل یک کلیـد بـاز و بـسته عمل می کند.

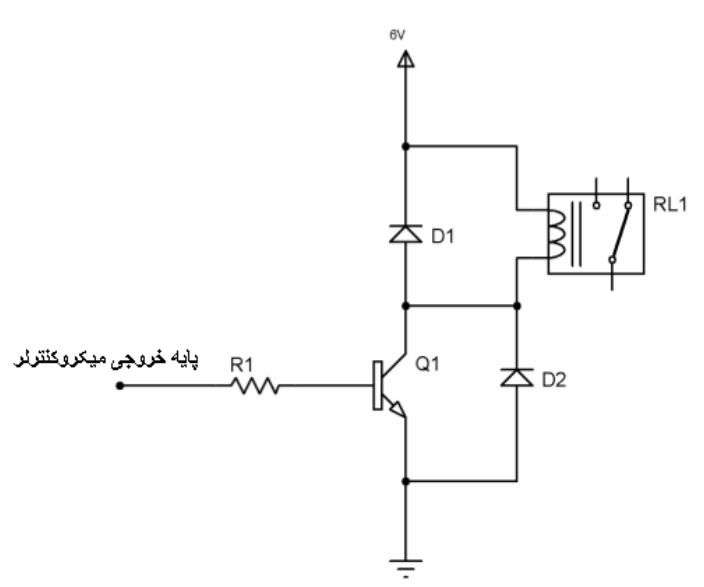




۴- استفاده از رله مغناطیسی :

زمانی که بخواهیم با استفاده از میکروکنترلر وسیله ایی با جریان و ولتاژ مصرفی بـالا مثـل لامـپ ۲۲۰ ولـت و ... را کنترل کنیم لازم است از رله های مغناطیسی به عنوان درایور استفاده کنیم. مکانیسم یک رله به این صـورت اسـت که داخل آن یک سیم پیچ با هسته داشته که با تحریک میدان مغناطیسی ایجاد شده و سبب بسته شدن تیغه های درون آن و در نتیجه باعث باز و بسته شدن کلیدها می گردد. رله ها معمولا ۵ یا ۶ پایه دارند که دو تای آن مربوط به تحریک رله و مابقی مربوط به کلیدها می باشند. ولتاژ تحریک سیم پیچ رله هـا معمـولا ۳ ، ۵ ، ۶ ، ۹ ، ۱۲ و ۲۴ ولت است که روی بدنه آنها نوشته شده است.

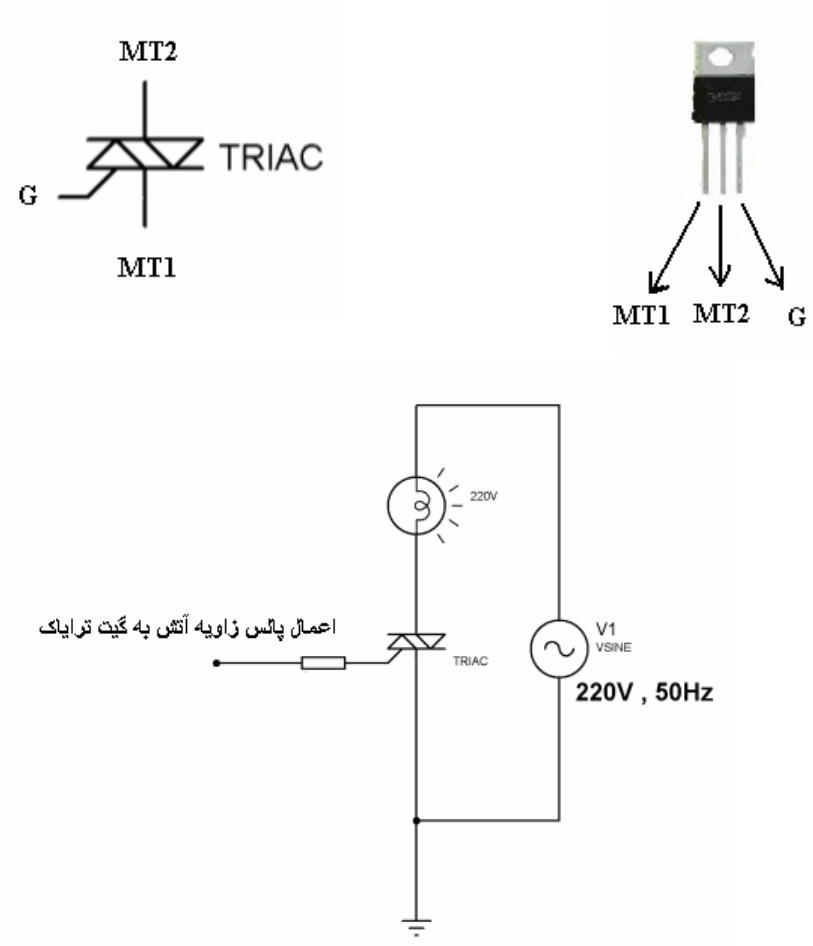
در راه اندازی رله توجه شود که حتما برای تحریک آن باید از ترانزیستور یا آی سی هـای بـافر بـرای تـامین جریـان مورد نیاز آن استفاده شود. مانند مدار شکل زیر:



۵- استفاده از ترایاک :

یکی از معایب رله ها کم بودن سرعت آن ها در مدارهایی که نیاز کلید زنی نسبتا زیاد دارد می باشد. بعنـوان مثـال در مداری مثل فلاشر و یا دیمر نمی توان از رله استفاده نمود. بنـابراین بهتـرین قطعـه مـورد اسـتفاده در ایـن نـوع مدارات ، ترایاک می باشد. ترایاک قطعه ایی ست دارای سه پایه به نام هـای G (گیـت) ، MT1 و MT2 کـه در واقع مانند یک کلید عمل می کند. به این صورت که با دادن یک پالس با پایه گیت که اصطلاحا زاویه آتـش نامیـده می شود بین پایه MT1 و MT2 اتصال برقرار شده و در این هنگام ترایاک مانند یک کلید بسته عمل می کند. بـا قطع شدن پالس نیز اتصال بین پایه های MT1 و MT2 قطع شده و در این هنگام ترایاک مانند یک کلید باز عمل می کند. یکی از معروفترین شماره های ترایاک برای استفاده های عمومی ، BT139 می باشد.

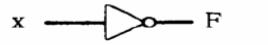
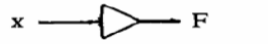
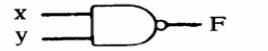
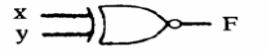
نماد فنی ترایاک در نقشه های الکترونیکی بصورت زیر است:



برای راه اندازی گیت ترایاک توسط میکروکنترلر می بایست مانند راه اندازی رله ، از یک عدد ترانزیستور استفاده شود تا بتواند جریان لازم را فراهم نماید.

ضمیمه ۵

گیت های منطقی

نام	نماد ترسیمی	تابع جبری	جدول درستی
AND		F = xy	x y | F ۰ ۰ | ۰ ۰ ۱ | ۰ ۱ ۰ | ۰ ۱ ۱ | ۱
OR		F = x + y	x y | F ۰ ۰ | ۰ ۰ ۱ | ۱ ۱ ۰ | ۱ ۱ ۱ | ۱
NOT		F = x′	x | F ۰ | ۱ ۱ | ۰
Buffer		F = x	x | F ۰ | ۰ ۱ | ۱
NAND		F = (xy)′	x y | F ۰ ۰ | ۱ ۰ ۱ | ۱ ۱ ۰ | ۱ ۱ ۱ | ۰
NOR		F = (x + y)′	x y | F ۰ ۰ | ۱ ۰ ۱ | ۰ ۱ ۰ | ۰ ۱ ۱ | ۰
Exclusive-OR (XOR)		F = x ⊕ y = xy′ + x′y	x y | F ۰ ۰ | ۰ ۰ ۱ | ۱ ۱ ۰ | ۱ ۱ ۱ | ۰
Exclusive-NOR (XNOR)		F = x ⊙ y = xy + x′y′	x y | F ۰ ۰ | ۱ ۰ ۱ | ۰ ۱ ۰ | ۰ ۱ ۱ | ۱

ضمیمه ۶

حداقل وسایل و قطعات مورد نیاز

برای انجام آزمایش های این مجموعه

۱- کامپیوتر یا لپ تاپ

۲- پروگرامر AVR

۳- مولتی متر دیجیتال

۴- بردبرد

۵- آی سی میکروکنترلر AVR سری Mega16 (۲ عدد)

۶- LED رنگی (۱۰ عدد)

۷- LCD کاراکتری 16*2 (یک عدد)

۸- کی پد 4*4 (یک عدد)

۹- مقاومت 4.7 K (۱۰ عدد) ، 470 اهم (۱۰ عدد) و 330 اهم (۱۰ عدد)

۱۰- خازن 100 uF (۲ عدد)

۱۱- سون سگمنت کاتد مشترک (سه عدد)

۱۲- آی سی رگولاتور 7805 (یک عدد)

۱۳- آی سی دیکدر 7448 (یک عدد)

۱۴- میکروسوئیچ (۱۰ عدد)

۱۵- بلندگو ۸ اهم (یک عدد)

۱۶- ترانزیستور منفی BC338 (۲ عدد)

۱۷- LCD گرافیکی 128*128 نوع SED (یک عدد)

۱۸- باتری کتابی ۹ ولتی به همراه سر باتری (یک عدد)

۱۹- پین هدر

۲۰- مقداری سیم خشک

فهرست منابع

۱- کاهه، علی، میکروکنترلرهای AVR، انتشارات نص

۲- بخت آور، محمود، آموزش مقدماتی میکروکنترلر AVR به زبان ساده برای جوانان، انتشارات آشینا

۳- طالبی، حسین، پروژه های میکروکنترلر AVR، انتشارات تخت سلیمان

۴- محسن زاده، مبین، مرجع کامل میکروکنترلرهای AVR، انتشارات موسسه دیباگران تهران

۵- کاظم لو، مهدی، ۱۰ پروژه با AVR، انتشارات آفرنگ

۶- مجموعه مقالات و مطالب موجود در سایت های اینترنتی