آزمایشگاه
میکروکنترلر

حسین سعیدی

دانشکده فنی و حرفه‌ای محمودآباد

دانشکده فنی و حرفه‌ای محمودآباد

آزمایشگاه میکروکنترلر

میکرو کنترلر چیست؟

قطعه‌ای الکترونیکی است که با نوشتن برنامه‌ای خاص برای آن، قادر است عمل مشخصی را انجام دهد. این قطعه به ظاهر ساده و جمع و جور، در واقع یک سیستم کامپیوتر کوچک است که ضمن داشتن CPU در کامپیوتر، مرکزی بقیه قطعات جانبی را نیز به همراه خود دارد درحالی که cpu در کامپیوتر، به تنهائی قادر به انجام کاری نبوده و به کمک قطعات خارجی مانند RAM و CD RAOM و کارت صوتی و کارت گرافیکی و مودم و فلاپی و مادربرد و کابل‌های رابط و HARDDICK کار می‌کند و در یک بسته بزرگ جمع آوری شده است.

در واقع میکرو کنترلر کامپیوتر کوچکی است که با منطق دیجیتال کار می‌کند و ضمن داشتن یک پروسسور(CPU) یا پردازشگر مرکزی، دارای تعدادی PORT یا درگاه ورودی/خروجی اطلاعات (IN/OUT = I/O) یا حافظه موقت و ROM یا حافظه دائمی و تعدادی REGISTER یا ثبات حافظه و غیره می‌باشد که در یک بسته سخت جمع‌آوری شده است و فقط پایه‌های قسمت‌های ذکرشده (PIN) بیرون از آن در دسترس می‌باشد.

از میکروکنترلرها، معمولاً جهت اهدافی استفاده می‌شود که فقط یک کار از آن می‌خواهند. به عنوان مثال: ساخت جعبه موسیقی، کنترل دور موتور، قفل رمز، ساعت، دزدگیر، ربات، تابلو روان، و غیره می‌توان استفاده کرد در حالی‌که کامپیوتر قادر است چند کار متفاوت را ایجا انجام دهد.

در ضمن، میکروکنترلر با تمام مزایایی که دارد، در مقابل کامپیوتر عددی نبوده و از سرعت عمل بسیار پائینی برخوردار است. با این حال در اکثر پروژه‌هایی که با میکروکنترلر طراحیمی‌شوند، این عیب‌ها هیچ مشکلی را به وجود نمی‌آورد.

تاریخچه میکرو کنترلر

با پیشرفت علم و تکنولوژی در الکترونیک تراشه‌هایی به عنوان میکروپروسسورها طراحی و تولید شدند تا قبل از سال ۱۹۷۱ میلادی اگر شخص طراح سیستمی را می‌خواست طراحی مند باید سیستم مورد نظر خود را به شرکت‌های سازنده میکروپروسسور ارائه میداد تا طرحی ساخته شود و یا اینکه مجبور بود با استفاده آی سی‌های دیجیتالی، سیستم مورد نظر را طراحی کند.

از این پس شرکت‌های سازنده میکرو پروسسورها از جمله Zilog تصمیم به ساخت میکروپروسسوری نمود که بتوان آن را در اختیار کاربر قرار داد و به هر صورت ممکن که می‌خواهد سیستم مورد نظر خود را طراحی کند و به همین دلیل میکرو پروسسور Z80 را به بازار عرضه کرد و نرم‌افزار کامپایلر به زبان اسمبلی و پروگرامر آن را نیز ارائه داد. بطور کلی اگر یک شخص از میکروپروسسور ۸ بیتیZ80 برای سیستمی استفاده کند، باید المان‌های جانبی CPU را نیز علاوه بر سخت افزار سیستم مورد نظر، در کنار میکروپروسسورZ80 قرار دهد.

معماری میکروکنترلرهای AVR (RISC) :

بطورکلی دو نوع معماری برای ساخت میکروکنترلرها وجود دارد:

۱- معماریCISC (Compex Instruction Set Computer)

تاریخچه این نوع معماری به قبل از سال۱۹۸۰میلادی برمی‌گردد. اکثر میکروپروسسورها و میکروکنترلرهای قدیمی از این نوع معماری، درآنها استفاده شده است. در این معماری تعداد دستورات بیشتر و پیچیده تر است، اما برنامه‌نویسی آن به خصوص اسمبلی ساده‌تر شده است و از طرفی سرعت اجرایی دستورات پایین‌تر است.

۲- معماری RISC (Reduced Instruction Set Computer)

بعد از سال ۱۹۸۰ میلادی، معماری جدیدی طراحی شد. در این نوع معماری تعداد دستورات کاهش پیدا کردد و از طرفی سرعت اجرایی دستورات ۱۰ برابر نسبت به معماری قبلی افزایش یافت و برنامه‌نویسی به زبان اسمبلی را قدری پیچیده و سخت کرد، اما با وجود ساختار بهینه شده میکروکنترلرهای AVR با حافظه‌های ظرفیت بالا و همچنین استفاده از معماریRISC، امکان برنامه‌نویسی به زبان‌های سطح بالاتر مانند C و بیسیک فراهم گردید.

تفاوت معماریRISC با معماریCISC :

۱- تعداد و اندازه دستورات در RISC کمتر از CISC است در نتیجه سرعت RISC بیشتر است.
۲- در معماری RISC انتقال داده از رجیستر به رجیستر و یا حافظه به حافظه صورت نمی‌گیرد.
۳- تعداد رجیسترها در RISC بیشتر است.
۴- برنامه‌نویسی به زبان اسمبلی در معماری RISC پیچیده تر از CISC است.
۵- اکثردستورات در معماری RISC در یک کلاک سیکل اجرا می‌شوند.
۶- مصرف توان معماری CISC بیشتر ازRISC است.
۷- برنامه‌هایMSDOS در معماری RISC قابل اجرا نیست و همین دلیل باعث شده است تا اکثر میکروپروسسورها(نظیر 80X86 ساخت شرکت Intel) از معماری CISC استفاده کنند.

تفاوت میکروپروسسورها(Micro Controller) با میکروکنترلرها(Micro Processor):

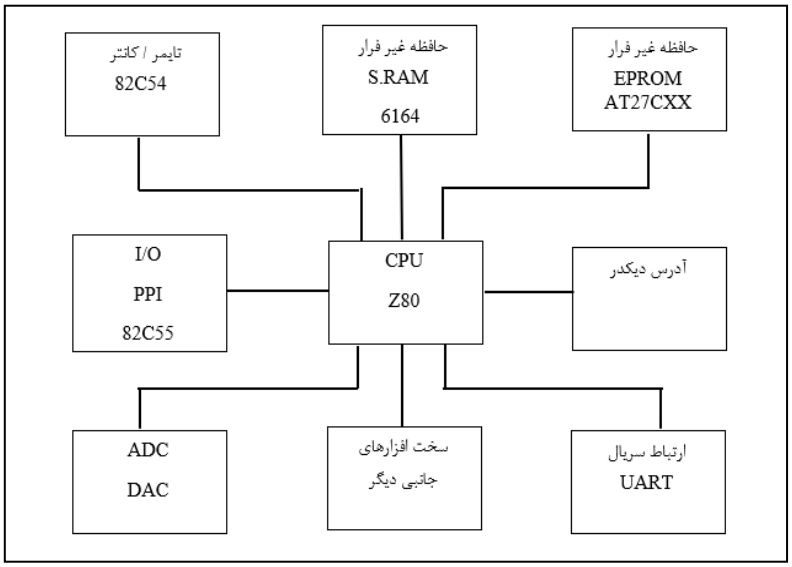
میکروپروسسورها همان طور که قبلاً ذکر کردیم فقط یک پردازنده کوچک هستند و باید المان‌های جانبی نظیر ARM,ROM تایمر یا کانتر، وقفه و سایر المانه‌های جانبی در کنار میکروپروسسور قرار داده شود تا بتوان یک سیستم چند منظوره را طراحی کرد. بطور مثال CPU یک میکرو پروسسور است و PC یک سیستم چند منظوره است که می‌تواند چندین عمل را اجرا کند، اما میکروکنترلرها دارای المان‌های جانبی محدود در یک بسته نظیر FLASH SRAM تایمر یا کانتر، وقفه و غیره هستند و فقط می‌توانند در سیستم‌های تک منظوره استفاده شوند. میکروکنترلره یک سیستم تک منظور است. میکروکنترلر به معنی کنترل کننده کوچک است و بطور مثال کنترل دمای یک دستگاه صنعتی توسط میکروکنترلر به معنی کنترل کننده کوچک است، بطور مثال کنترل دمای یک دستگاه صنعتی توسط یک میکروکنترلر یک سیستم تک منظوره است. میکروپروسسورها از معماری CISC استفاده می‌کنند اما میکروپروسسورها از معماری RISC استفاده می‌کنند اما میکروکنترلرهای پیشرفته از جمله AVR,PIC از معماری RISC استفاده می‌کنند البته میکروکنترل‌های قدیمی‌تر از جملهCISC 8051 از معماری استفاده می‌کنند و اساسی‌ترین تفاوت میکروپروسسورها انعطاف‌پذیری آنهاست که می‌توان ROM , RAM آن‌ها را افزایش داد اما در میکروکنترلرها میزان ظرفیت حافظه ثابت است.

بطور خلاصه می‌توان گفت:

۱- میکروپروسسور نیاز به حافظه‌های ROM , RAM دارد.
۲- میکروپروسسور نیاز به قطعات جانبی زیادی دارد.
۳- قابلیت اعتماد و کارایی میکروکنترلر نسبت به میکروپروسسور بیشتر است.
۴- سرعت پردازش میکروپروسسورها از میکروکنترلرها بیشتر است.

تفاوت میکروکنترلرهای AVR با دیگر میکروکنترلرها:

اینکه بگوییم چه میکروکنترلری قویتر و بهتر است بسیار سخت است و نمی‌توان در حالت کلی میکروکنترلرها را باهم مقایسه کرد. اما بطور مثال میکروکنترلرPIC به شماره PIC18F452 را می‌توان با میکروکنترلرAVR به شماره ATmega32 مقایسه کرد قابلیت‌های این دو میکروکنترلر بسیار شبیه هم است. برای اینکه شما قضاوت صحیح‌تری داشته باشید پیشنهاد می‌کنم که ویژگی‌های این دو میکروکنترلر را از برگه اطلاعاتی آنها مطالعه کنید.



همانطور که در بلوک دیاگرام شکل فوق مشاهده می‌نمائید، برای اینکه از یک میکرو پروسسور، حتی برای ساده ترین سیستم بخواهیم استفاده کنیم، باید از المانهای جانبی دیگری نیز بهره بگیریم. این عمل سبب افزایش قیمت و پیچیده شدن سخت‌افزار پروژه مورد نظر می‌گردد. از این پس شرکت‌های سازنده قطعه، به فکر تراشه‌ای بودندکه تمام امکانات جانبی میکرو پروسسور را به همراه خود CPU داشته باشد، تا شخص طراح با قیمت مناسب و سخت‌افزار کمتر بتواند سیستم مورد نظر خود را بسازد.

در سال ۱۹۸۱ میلادی شرکت Intel تراشه‌ای را به عنوان میکروکنترلر خانواده 8051 به بازار عرضه کرد. این میکروکنترلر دارای CPU ۸ بیتی، تایمر یا کانتر، وقفه، تبادل سریال، حافظه SRAM و حافظه غیر فرار(FLASH) داخلی می‌باشد. در اوایل، میکروکنترلر8051 ، از نوع حافظه PROM یعنی نوع OTP (One Time Programmable) بهره می‌برد، این نوع میکروکنترلر فقط یک بار قابل برنامه ریزی بود. بعداً که این میکرو کنترلر توسعه یافت، از حافظه‌های EPROM استفاده کردند. این نوع میکروکنترلر با شماره 87Cxx آغاز می‌شد و حسن این حافظه در این بود که می‌توانست توسط پنجره‌ی شیشه‌ای که در بالای تراشه قرار داشت، در مجاورت نور و ماورای بنفش مثل نور خورشید قرار گرفته و بعد از چند دقیقه پاک شود. شرکت سازنده میکروکنترلر 8051، تصمیم به استفاده از حافظه‌ای گرفت که بتواند با ولتاژ الکتریکی نوشته و پاک شود(Flash)، این سری با شماره 89Cxx آغاز می‌شود که امروزه نیز همچنان مورد استفاده قرار می‌گیرد. شرکت‌های سازنده دیگری، تحت لیسانس شرکت Intel از میکرو کنترلر 8051 تولید کردند، از جمله این شرکت‌ها می‌توان به شرکت Atmel اشاره کرد. این شرکت بعداً نوع توسعه یافته 8051 را با سری شمارهAT89Sxx ارائه کرد، تفاوت نوع S با نوع C در نحوه‌ی برنامه‌ریزی میکروکنترلر است، زیرا نوع S را می‌توان داخل مدار، پروگرام کرد.

برای پروگرام کردن نوع S نیازی به خارج کردن میکروکنترلر از مدار نداریم و می‌توانیم در داخل سیستم، توسط یک IC بافر، عمل پروگرام کردن را از طریق پورت LPT انجام دهیم.

سرانجام شرکت Atmel میکرو کنترلرهای جدیدی را مانند دیگر شرکت‌های سازنده قطعه از جمله شرکتMicochip که میکرو کنترلرهای PIC را تولید کرده است، شرکت Atmel نیز خانواده AVR را در سال ۱۹۹۷میلادی به بازار عرضه نموده است . در یک تالار گفتگو به زبان لاتین، برخی بر این باور بودند که واژه AVR مخفف نام سازندگان اصلی میکرو کنترلر از شرکت Atmel با نام‌های Vegard و Alf می‌باشد که حرف R نیز به معماری RISC بکار رفته اشاره دارد. برخی از افراد دیگر، فکر می‌کنند که کلمه AVR مخفف کلمه Advanced Virtual RISC می‌باشد. البته دقیقاً مشخص نیست که خانواده AVR مخفف چه کلماتی می‌باشد اما ممکن است فقط یک نماد تجاری شرکت ATMEL باشد.

انواع میکروکنترلرها

۱	میکروکنترلرهای خانواده	Z8	از شرکت	Zilog
۲	میکروکنترلرهای خانواده	8051	از شرکت	Intel
۳	میکروکنترلرهای خانواده	6811	از شرکت	Motorlla
۴	میکروکنترلرهای خانواده	PIC	از شرکت	Microchip
۵	میکروکنترلرهای خانواده	ARM	از شرکت	Atmel , …
۶	میکروکنترلرهای خانواده	AVR	از شرکت	Atmel , …

میکرو کنترلرهای AVR

میکروکنترلرهای خانواده AVR از سری میکروکنترلرهای هستند که امروزه به طور فراوان در آزمایشگاه و کارهای صنعتی و خدماتی مورد استفاده قرار می‌گیرند. میکروکنترلرAVR مخفف Advanced Virtual RISC است که خود RISC به مفهوم Reduced Instruction Set Of Computer است، به عبارت دیگر AVR به معنی کاهش دستورالعملات کامپیوتری واقعاً پیشرفته است.

انواع میکروکنترلرهای AVR

۱- میکروکنترلهای خانوادهی Attiny

این میکروکنترلرها که به AVRهای کوچک معروفند دارای پایه‌های کمتر و قابلیت کمتری نسبت به سری‌های دیگر هستند.

این میکرو کنترلرها در ابعاد کوچک 8 و 20 و 28 پایه هستند و قابلیت ای خوبی نسبتب به سری اول دارند و بیشتر در سیستم‌هایی که نیاز به پورت بالا نیست استفاده می‌شوند. یکی از اعضای 8 پایه این سری ATtiny است که دارای امکانات خوبی از جمله مبدل ADC می‌باشد.

ATtiny 10	ATtiny 12	ATtiny 15	ATtiny 25	ATtiny 28	ATtiny 85
ATtiny 11	ATtiny 13	ATtiny 22	ATtiny 26	ATtiny 45	ATtiny 2313

۲- میکروکنترلرهای خانوادهی AT90S

این سری، اعضای کلاسیک خانواده AVR را تشکیل می‌دهند و قابلیت‌های کمتری نسبت به دو سری بعدی دارند و کمتر استفاده می‌شوند.

AT90S4434	AT90S8535	AT90S4433	AT90S2323	AT90S1200
AT90S8534	AT90S4414	AT90S8515	AT90S2343	AT90S2313

۳- این میکروکنترلرها قابلیت و کارایی و پایه‌های بیشتری دارند که به میکروکنترلرهای بزرگ معروفند.

سری ATmega : این سری از میکروکنترلرهای AVR ، امکانات بیشتری نسبت به دو سری قبلی دارند و توجه مخاطبان را به خود جلب نموده‌اند. در این کتاب میکروکنترلر ATmega32 را برای آموزش انتخاب کرده‌ایم، زیرا در اکثر فروشگاه‌های الکترونیکی با قیمت مناسب عرضه شده و تمام سرفصل‌های آموزشی AVR کار کنید. میکروکنترلرهای ATmega8 و ATmega32 و ATmega128 را می‌توان به عنوان میکروکنترلرهای پرکاربرد این سری نام برد.

از آنجا که اصل سازگاری از طرف شرکت‌های سازنده قطعه رعایت می‌شود میکروکنترلرهای AVR تمام قابلیت‌های میکروکنترلر8051 را دارند و از امکانات جدیدی از جمله ACD و TWI و SPI و EEPRO و ... بر خوردار هستند. اما شرکت Atmel میکروکنترلر 8051 بکار رفته است ، بطور مثال میکروکنترلر AT89C52 دارد تا به توان به سادگی آن را در برد میکروکنترلر قدیمی جایگزین کرد.

۴- میکروکنترلرهای خانواده Xmega:

جدیدترین سری میکروکنترلر شرکت ATMELکه در سال 2008 وارد بازار شده و تمامی مشکلات میکروکنترلرهای سری‌های پیش از این را رفع کرده و با امکاناتی جدید طراحی شده است.

۵- میکروکنترلرها به صورت یک پالس کامل طراحی شده است که صنایع ماشین‌سازی و اتوماسیون صنعتی کاربرد دارد و از جمله مدارهای اضافه شده به این باکس، مدار سنسورهای دما و مدار خطایابی خط داده و مدار ارتباط با سیستم‌های CANOPEN می‌باشد.

۶- میکروکنترلرهای AVR :

این سری از میکروکنترلرها در حقیقت سری پیشرفته‌ی میکروکنترلرهای Automotive AVR است، و از تمامی امکانات این سری برخوردار هستند.

۷- میکروکنترلرهای AVR Z – LINK:

این سری از تراشه‌ها در سیستمهای Wireless کاربرد قراوانی دارند و قابلیت ارسال داده بوسیله سیگنالهای رادیویی RF و با فرکانس برابر با 2.4GHZ را دارا می‌باشند.

۸- میکروکنترلرهای LCD AVR :

این میکروها برای کار و هماهنگی با LCD طراحی شده است و می‌توانند به هر پورت یک LCD متصل شده است و با همگی در حال ارتباط باشند.

۹- میکروکنترلرLighing AVR:

کاربرد این نوع از کنترلرها در کنترل موتورها می‌باشد، چون امکانات اضافی آن در مقابل سری‌های دیگر، دارا بودن ۲ تا ۳ کانال PSC۱۲(Power Stage Controls) و کانال‌های PWM که از فرکانس 25HZ تا 150KHZ را می‌توانند در خروجی ارائه دهند.

۱۰- میکروکنترلر Smart Battrey AVR :

این میکروکنترلرها کنترل جریان پایه‌ها در زمان اتصال کوتاه و ولتاژ کاری متفاوت از دیگر تراشه‌ها که برابر 1.8V الی 9V هستند. این سری به صورت پکیج بوده و روی پکیج‌ها یک یا دو باتری موجود بوده که در زمان خاموشی هم، ساعت میکرو از کار نخواهد افتاد.

۱۱- میکروکنترلرهای USB AVR :

۱۲-این سری میکروها ارتباط و هماهنگی کامل با ارتباط پر سرعت USBمی‌باشد.

مشخصات میکروکنترلر:

همانطوریکه می‌دانید شرکت‌های سازنده قطعات الکترونیکی بر روی المان‌های تولیدی، اطلاعاتی در خصوص قطعات تولید شده درج می‌کنند. بر روی میکروکنترلر حروف یا اعدادی درج می‌شوند که هر یک بیانگر مفهوم خاصی هستند.

۱- در بلوک شماره‌ی ۱ نام میکرو و حافظه‌ی Flash آن مشخص می‌شود.

۲- در بلوک شماره‌ی ۲ ولتاژ و فرکانس کاری میکرو مشخص می‌شود که سه حالت دارد:

حرف	ولتاژکاری	محدوده فرکانسی
بدون حرف -	4 – 5.5 v	0 – 16 mhz
با پسوند L	2.7 – 5.5 v	0 – 8 mhz
با پسوند V	1.8 – 5.5 v	0 – 4 mhz

۳- در بلوک شماره‌ی ۳ معمولاً حداکثر فرکانس کاری مشخص می‌شود، برای مثال اگر عدد ۱۶ باشد حداکثر فرکانس کاری16MHZ است.

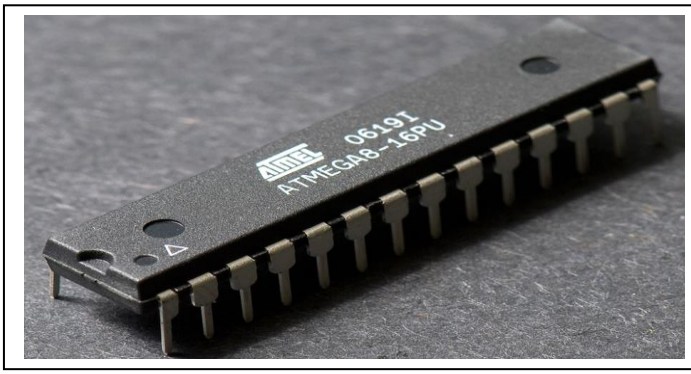
۴- در بلوک شماره‌ی ۴ شکل ظاهری یا نوع بسته‌بندی IC میکروکنترلر مشخص می‌شود که می‌تواند به صورت حروف زیر باشد.

A:	بسته بندی TQFP
J:	بسته بندی PLCC
M:	بسته بندی MLF
P:	بسته بندی PDIP
S:	بسته بندی SOIC
Y:	بسته بندی SSOP

۵- در بلوک شماره‌ی ۵ محدوده‌ی دمای کاری میکروکنترلر مشخص می‌شود که با حروف زیر مشخص می‌شود.

C: (Commercaial) تجاری (0^0 C تا 70^0 C)

I یا C: (Indutrial) صنعتی (-40^0 C تا 85^0 C)



1	2	3	4	5
ATMEGA8	—	16	P	U

مثال:

این آی‌سی میکروکنترلر ATmega8 است که دارای 8k بایت حافظه‌ی داخلی است و – به این معناست که ولتاژ کاری آن از 4v تا 5.5v است و فرکانس کاری آن از صفر تا 16MHZ است و عدد 16 بیانگر ماکزیمم فرکانس کاری16MHZاست و حرف P بیانگر شکل ظاهری آی‌سی از نوع بسته بندی PDID است و حروف U هم بیانگر محدوده‌ی دمای کاری از (-40^0 C تا 85^0 C) است.

میکروکنترلر ATmega32:

در این کتاب بطور عمده از میکروکنترلر ATmega32 را مورد بررسی قرار دهیم و می‌توان با استفاده از این آی‌سی کار با بقیه آی‌سی‌ها را تعمیم داد، هرچند پروژه‌های مختلفی از آی‌سی‌های سری دیگر را در ادامه مورد بررسی قرار می‌دهیم ولی ATmega32 دارای ویژگی‌های خاصی است، که بهتر است این آی‌سی را مورد تجزیه و تحلیل قرار دهیم.

مشخصات ATmega32:

۱- این آی‌سی دارای 40 پایه از نوع PDIP و 44 پایه از نوع TQFP است.
۲- دارای چهار پورت کامل 8 بیتی برای انتقال اطلاعات است.
۳- دارای تعداد زیادی از دستوراتی است که اکثراً در یک سیکل کار می‌کنند.
۴- از کارایی و قابلیت اعتماد بالا و عملکرد کاملاً ثابت برخوردار است.
۵- قابلیت کار تا فرکانس 16MHZ را دارد و سرعت آن 16MIPS است.(Million Instruction Per Seconds)MIPS
۶- تا فرکانس 8MHZ نیازی به کریستال خارجی ندارد.
۷- به خاطر استفاده از تکنولوژی CMOS توان مصرفی پایین ی دارد.
۸- ولتاژ کاری 4V تا5.5V دارد.
۹- دارای32کیلو بایت حافظه‌ی فلش است که 10000 بار قابلیت پاک کردن و نوشتن دارد.
۱۰-دارای 2 کیلو بایت حافظه‌ی داخلی SRAM است.
۱۱-دارای 1 کیلو بایت حافظه‌ی EEPROM با قابلیت 100000 بار پاک کردن و نوشتن است.
۱۲-دارای قفل برنامه و حفاظت از دیتا است.
۱۳-قابلیت ارتباط به صورت JTAG(Join test Access group) برای برنامه‌ریزی فلش و E2PROM و قفل فیوز بیت‌هاست. (فیوز بیت، قسمتی از حافظه‌ی فلش هستند که امکاناتی را در اختیار کاربر قرار می‌دهند که با پاک کردن میکرو از بین نمی‌روند و توسط بیت‌های قفل مربوطه، قفل می‌شوند)
۱۴-دارای سه سری تایمر هستند که دو تایمر آن به صورت تایمر-کانتر8 بیتی است و یک تایمر آن به صورت تایمر -کانتر 16 بیتی است.
۱۵-دارای یک پورت کامل مبدل آنالوگ به دیجیتال(Analog Digital Convertor) :ADC است که از پورت A استفاده می‌شود.

۱۶-دارای یک مقایسه کننده‌ی آنالوگ داخلی(Analig Refrence): AREF است.

۱۷- دارای یک کلاک ساعت واقعی(Real Time Clock): RTC با اسیلاتور مجزا است.

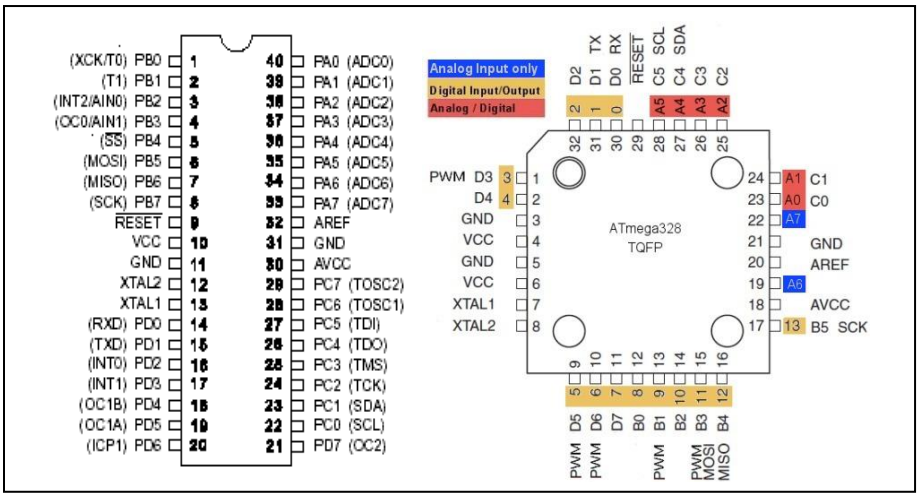
و همچنی قابلیت‌های فراوان دیگر که در حین انجام پروژه‌ها با آنها نیزآشنا خواهید شد.

میکرو کنترلرهای AVR قابلیت بیستری نسبت به میکروکنترلرهای8051 دارند و از نسل جدیدی از حافظه‌های Flash و معماری RISC که در ادامه توضیح می‌دهیم بهره می‌گیرند.

مشخصات ظاهری پایه‌های ATMEGA32:

در شکل زیر مشخصات پایه‌های ATmega32 به صورت PDIP را نشان می‌دهد هرچند که شکل این آی‌سی را به صورت TQFP نشان می‌دهد که در TQFP که به صورت 44پایه است نسبت به PDIP دارای دو تا V_{DD} وv_{SS} اضافه است ولی در بقیه پایه‌ها با هم یکسانند.

همانطور که در شکل زیر نشان داده شده است این آی‌سی دارای مشخصات زیر است.



۱- دارای چهار پورت کامل ورودی و خروجی

Port A	PA_0	PA_1	PA_2	PA_3	PA_4	PA_5	PA_6	PA_7	PA_8
Port B	PB_0	PB_1	PB_2	PB_3	PB_4	PB_5	PB_6	PB_7	PB_8
Port C	PC_0	PC_1	PC_2	PC_3	PC_4	PC_5	PC_6	PC_7	PC_8
Port D	PD_0	PD_1	PD_2	PD_3	PD_4	PD_5	PD_6	PD_7	PD_8

۲- V_{SS} که باید به منفی منبع تغذیه یا GND وصل شود. مثلاً(5V -)

۳- V_{DD} که باید به مثبت منبع تغذیه وصل شود. مثلاً(5V +)

۴- Reset (راه اندازی مجدد) که پایه‌ی Active low است در صورتی که صفر شود آی سی Reset می‌شود و نحوه‌ی وصل آن نیز به صورت زیر است.

۵- $XTAL_2$, $XTAL_1$ که پایه‌های اتصال کریستال (نوسان‌ساز) هستند و زمانی که از کریستال های خارجی استفاده می‌شود باید به این دو پایه همراه با خازن‌های خیلی کوچک عدسی (که به خازن‌های نویزگیر معروفند) وصل شود.

۶- AVCC که پایه‌ی تغذیه برای پورتA است که به VCC وصل می‌شود که اگر ADC باشد باید از طریق فیلتر پایین گذر وصل شود.

۷- AREF که ولتاژ مرجع آنالوگ برای مبدل آنالوگ به دیجیتال یا همان ADC است.

تعاریف و اصطلاحات معمول

RAM (**R**andom **A**ccess **M**emory)

به معنی حافظه خواندنی و نوشتنی می‌باشد که به راحتی می‌توان اطلاعات را روی آن قرار داد. اما به محض قطع جریان الکتریکی، تمامی اطلاعات و محتویاتش از بین می‌رود.

RAM های خارجی روی برد کامپیوتر برای همه آشنا است که در دو نوع SRAM (استاتیک) DRAM (دینامیک) موجودند و معمولاً همه سعی می‌کنند RAM یا حافظه خارجی خود را افزایش دهند تا سریع‌تر و بهتر کار کند.

ROM (**R**ead **O**nly **M**emory)

با توجه به معنی آن معلوم می‌شود که فقط حافظه خواندنی دائم می‌باشد و با قطع برق، اطلاعات ان از بین نمی‌رود. مانند حافظه دائمی ای سی میکرو که برنامه اصلی در آن ذخیره می‌شود.

PROM (**P**rogrammabl**e** **R**ea**d** **O**nly **M**emory)

این نوع حافظه فقط خواندنی است و یک بار برای همیشه توسط شرکت سازنده برنامه ریزی شده و قابل تغییر نمی‌باشد.مانند انواع آی سی های تولید صدا و ساعت و غیره.

EPROM (**Erasable P**rogrammabl**e** **R**ea**d** **O**nly **M**emory)

این نوع حافظه را می‌توان پاک کرد و دوباره مورد استفاده قرار داد، اما این کار فقط با ابزار ویژه مانند اشعه ماورا بنفش مقدور می‌باشد و لذا مصارف صنعتی مخصوص به خود را دارد.

EEPROM (**E**lectrica**l** **E**rasable **P**rogrammabl**e** **R**ea**d** **O**nly **M**emory) یا **E2PROM**

که توسط جریان الکتریکی به سادگی پاک شده و دوباره قابل برنامه‌ریزی می‌باشد و این کار را بارها و بارها می‌توان تکرار کرد که نمونه آن را در تابلوی روان می‌توان تجربه نمود.

FLASH ROM (Flash Memory Rrprogramable)

از نوع حافظه‌های جدید و جانبی هستند که ظرفیت بالائی دارند و تمامی کسانی که با IC میکروکنترلر و کامپیوتر سر و کار دارند با این نام آشنا هستند.(فلش مموری)

این حافظه ها بوسیله جریان الکتریکی قابل برنامه‌ریزی و پاک شدن می‌باشند و از این خواص ویژه آنها، ظرفیت زیاد و سرعت بالای تبادل اطلاعات و برنامه‌ریزی آنها می‌باشد.

DRAM (Dynamic RAM)

در این RAM داده‌ها برای باقی ماندن نیاز به دوره‌ی تناوب دوباره پر شدن یاRefreshing دارند.

CPU (Centeral Processor Unit)

CPU یا واحد پردازش مرکزی از قسمت‌های اصلی واحد محاسبه و منطقی واحد ثبات و واحد کنترل تشکیل شده است، که پردازش اطلاعات را به عهده دارد و توسط واحدهای ورودی و خروجی اطلاعات از ورودی به خروجی و بالعکس انتقال می‌دهد و اطلاعات گرفته شده را مورد پردازش قرار می‌دهد.

واحدI/O(Input / Output):

واحد I/O یا واحد ورودی و خروجی که اطلاعات توسط قسمت‌های ورودی و خروجی، داخل CPU می‌شوند و یا از CPU گرفته می‌شوند و به اصطلاح ارتباط با دنیای خارج هستند.

PORT SERIAL

تقریبا همان معنا و مفهوم سریال – پشت بر هم – مداوم را دارد. منظور اسریال سازی اطلاعات می‌باشد و به عبارت دیگر، اطلاعات یک بایت از حافظه را به صورت بسته پشت سر هم قرار داده و جهت انتقال از آن استفاده می‌کنند.

پورت سریال عمل بسته بندی و انتقال اطلاعات را بر عهده دارد که از معروف‌ترین و قدیمی‌ترین آنها، پورت سریال ۹ پایه می‌باشد و به ورودی COM در پشت کامپیوتر معروف است.

عملگرها

به وسیله‌ای گفته می‌شود که با گرفتن سیگنال‌های الکتریکی، عمل خاصی را انجام می‌دهند. عملگرها انواع مختلفی دارند و تقریباً می‌توان هر وسیله الکتریکی را یک عملگر نامید. حتی یک لامپ یا یک کلید را نیز می‌توان یک عملگر نامید.

سنسور(SENSOR)

به قطعاتی گفته می‌شود که شرایط مختلف محیط مانند(دما – رطوبت – نور – لرزش – حرکت و غیره ...) را به سیگنال‌های الکتریکی تبدیل می‌کند تا بتوان آن را به کمیت‌های قابل اندازه‌گیری تبدیل کرده و آن را برای فعال شدن یک عملگر بکار برد.

مثلاً قطعه‌ای به نام LM35 که یک سنسور معروف دماست، در واقع به ازای افزایش یا کاهش دما، ولتاژهای مشخص و معینی را تولید می‌کند که توسط مبدل‌های الکترونیکی به کمیت‌های قابل استفاده برای سنجش دما بکار گرفته می‌شوند.

مبدل‌های الکترونیکی ممکن است به دو صورت آنالوگ یا دیجیتال عمل کنند.

آنالوگ(ANALOGE)

هر سیستمی که مقدار آن به صورت پیوسته عمل کند، سیستم آنالوک نام دارد. مثلاً نور یک لامپ را توسط ولتاژهای مختلف پیوسته، کم و زیاد کرد.تا قبل از پیشرفت‌های حیرت انگیز الکترونیک در دهه‌های اخیر، اکثر سیستم های الکترونیکی بصورت آنالوگ عمل می‌کردند که با پیدایش سیستم‌های دیجیتال به تدریج سیستم‌های قدیمی آنالوگ منسوخ شده و به سرعت به سیستم‌های دیجیتال تغییر شکل داده‌اند.

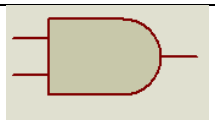
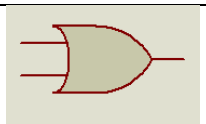
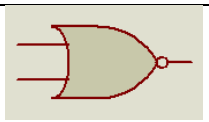
منطق دیجیتال

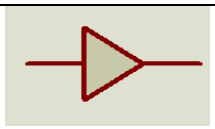
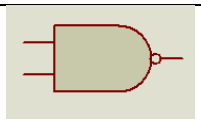
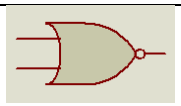
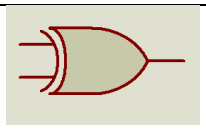
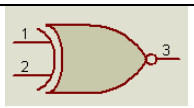
منطق دیجیتال منطقی است که بر اساس 0 و 1 و یا در واقع خاموش و روشن شدن و یا قطع و وصل کار می‌کند . بعبارت دیگر، دیجیتال به معنای جدا یا گسسته می‌باشد. در این سیستم همه چیز بر اساس 0 و 1 پایه برنامه‌ریزی شده است و منطق همه کارها بر اساس توابع ریاضی و ترکیبی از 0 و 1 ها می‌باشد.

برای ورود به این دنیای حیرت انگیز، به معلومات ریاضی قوی و مطالعه گسترده و عمیق احتیاج است که به صورت خیلی خیلی خلاصه به بیان مطالبی در این حوزه خواهیم پرداخت.

گیت‌های منطقی

همانطوری که قبلاً گفتیم ارقام باینری سطح ولتاژ را با 0 و1 معیین می‌کنند. در بحث لاجیک با ولتاژ 5v کار داریم. معمولاً 0 تا 0.7 ولتا را سطح پایین یعنی 0 و 3 تا 5 ولت را سطح بالا یعنی 1 در نظر می‌گیریم. گیت‌های منطقی مدارات ساده‌ای هستند که دو یا چند ورودی و یک خروجی دارند.

نام گیت	سمبل	تابع جبری	جدول درستی
And		X= A. B or X= A B	A B X 0 0 0 0 1 0 1 0 0 1 1 1
OR		X = A + B	A B X 0 0 0 0 1 1 1 0 1 1 1 1
Not		X = A′	A X 0 1 1 0

Buufer		$X = A$	A X 0 0 1 1
nand		$X = (AB)'$	A B X 0 0 1 0 1 1 1 0 1 1 1 0
Nor		$X = (A + B)'$	A B X 0 0 1 0 1 0 1 0 0 1 1 0
Xor Exclusive OR		$X = A \oplus B$ or $X = A'B + AB'$	A B X 0 0 0 0 1 1 1 0 1 1 1 0
Xnor Exclusive NOR Or Equivalence		$X = (A \oplus M)'$ or $X = A'B' + AB$	A B X 0 0 1 0 1 0 1 0 0 1 1 1

A/D تبدیل کننده ولتاژهای آنالوک به دجیتال

برای بیان منظور قطعات و ابزارهای مختلفی وجود دارند که می‌توانند سیستم‌های آنالوگ را به دیجیتال تبدیل کنند.

D/A تبدیل کننده ولتاژهای دیجیتال به آنالوگ

برعکس عمل بالا را انجام می‌دهد، یعنی ورودی دیجیتالی را دریافت کرده و خروجی آنالوگ را با توجه به ولتاژهای مرجع تحویل می‌دهد. باید توجه داشته باشید خروجی‌های A/D معمولاً جریان است که با یک مقاومت مناسب می‌توان آن را به ولتاژ تبدیل کرد.

رجیستر یا ثبات Register

به قسمتی از IC میکروکنترلر گفته می‌شود که مانند یک بایت از RAM عمل می‌کند و می‌توان اطلاعات را روی آن قرار داد و استفاده نمود. رجیسترها متنوع بوده و اسامی خاصی دارندکه در برنامه نویسی از آن‌ها استفاده می‌شود مانند ثبات‌های A و B و B0 و R0 و غیره که به تدریج با آنها آشنا خواهید شد.

ADRESS

قطعاً بدون داشتن یک آدرس مناسب و درست از محل اقامت شخصی، نمی‌توان به سادگی به آن محل دسترسی پیدا کرد. با توجه با پایه‌های ثبات (رجیستر) IC میکرو، لازم است نشانی خانه‌های حافظه‌ای را که اطلاعات در آن ثبت شده یا ثبت خواهد شد، معلوم گردد که به این مرحله آدرس دهی گویند.

کوچکترین جزء حافظه ،Bit است که زیاد مورد استفاده قرار نمی‌گیرد. بعضی از پردازشگرها قادر به آدرس دهی بیتی هستند و برخی دیگر این قابلیت را ندارند. اما همه آنها امکان آدرس دهی Bity (۸بیتی) را دارند.

DATA (داده – اطلاعات)

این کلمه به معنی داده یا اطلاعات می‌باشد که وارد حافظه IC می‌کنیم و یا از آن دریافت می‌کنیم تا کار مشخصی را انجام دهد.

Bit (بیت) : کوچکترین جزء حافظه بیت نام دارد که می‌تواند حالت 0 و 1 را داشته باشد.

Byte(بایت) : مجموعه‌ای از ۸ بیت می‌باشد که شامل ۸ رقم از اعداد در مبنای ۲ بوده و از 0 تا 255 وضعیت، یعنی (2^8) را می‌تواند داشته باشد.

Kbyte(کیلو بایت) : یعنی (2^{10}) که می‌شود ۱۰۲۴ را یک کیلو بایت گویند.

Mbyte(مگا بایت) : یعنی (2^{20}) که می‌شود ۱۰۴۸۵۷۶ را یک مگا بایت گویند.

Gbyte(گیگا بایت) : یعنی (2^{30}) که عدد بدست آمده را یک گیگا بایت گویند.

زبان برنامه نویسی BASIC

امروزه صدها زبان کامپیوتری مورد استفاده قرار می‌گیرند که به طور کلی آنها را به سه دسته تقسیم می‌کنند.

۱- زبان سطح پایین (L-L–L)(Low-Level-languages): این زبان‌ها به زبان ماشین یا سخت‌افزارنزدیک هستند.
۲- زبان سطح متوسط (M-L-L) (Medium-Level-languages): این زبانا مابین زبان ماشین و زبان محاوره‌ای هستند.
۳- زبان سطح متوسط (H-L-L) (High-Level-languages): این زبان‌ها به زبان محاوره‌ای نزدیک هستند که از زبان‌های معروف سطح بالا(H-L-L) می‌توان به زبان C و Basic اشاره کرد.

جالب است بدانید Basic مخفف کلمات (Beginners All-purpose Symbolic Instruction Code) می‌باشد. این زبان برنامه‌نویسی به خاطر سادگی ساختاری زیاد مورد استفاده قرار می‌گیرد. این زبان برنامه‌نویسی در سال 1964 میلادی توسط جان کمنی(john Kemney) و توماس کورتس(Thomas Kurts) در دانشگاه Dartmouth بوجود آمد که اشکال مختلف آن در برنامه‌نویسی مثلBasic A , Visual Basic , QBasic و ... مورد استفاده قرار می‌گیرد. بطور خلاصه می‌توان گفت جهت نوشتن برنامه به زبان Basic از محیط نرم افزاری یا همان کامپایلر Bascome avr استفاده کرد.

در این کتاب به آموزش زبان برنامه نویسی Basic می‌پردازیم.

دستورات توابع منطقی:

معادل AND دو عبارت یا همان "و" است	AND
معادل OR دو عبارت یا همان "یا" است	OR
معادل XOR دو عبارت یا همان "OR" انحصاری است	XOR
معادل NOT دو عبارت یا همان "نه" است	NOT

دستورات توابع ریاضی:

علامت جمع	+	تابع لگاریتمی در هر مبنایی	Log
علامت تفریق	-	علامت کوچک‌تر – مساوی – بزرگ‌تر	< = >
علامت ضرب	*	علامت نامساوی یا مخالف هم	< >
علامت تقسیم	/	علامت توان	power یا ^
علامت اعشار	.	قدر مطلق	Abs
تابع لگاریتمی در مبنای ۱۰	Log10	تابع نمایی	Exp
یک عدد تصادفی بین صفر تا محدوده‌ی معین را مشخص می‌کند RND			
یک عدد رند یا گرد شده را مشخص می‌کند Round			

دستورات توابع مثلثاتی:

سینوس یک تابع	**sin**	**Arc سینوس یک تابع**	**Asin**
کسینوس یک تابع	**cos**	**Arc کسینوس یک تابع**	**Acos**
تانژانت یک تابع	**Tan**	**Arc تانژانت یک تابع**	**Atan**
هایپربولیک سینوس یک تابع	**Sinh**	**درجه را به رادیان تبدیل می‌کند**	**DEG2RAND**
هایپربولیک کسینوس یک تابع	**Cosh**	**رادیان را به درجه تبدیل می‌کند**	**RAND2DEG**

هایپربولیک تانژانت یک تابع	Tanh		

دستور برای تغییر نام متغیر(ALIAS)

مثال : اگر بخواهیم به جای PORTC.0 از LED استفاده کنیم به صورت زیر بیان می‌کنیم: LED ALIAS PORTC.0

دستور افزایش و کاهش به مقدار یک واحد (INSR , DECR)

مثال:

INCR X یک واحد به مقدار X اضافه شود

DECR X یک واحد از مقدارX کم شود

دستور جابجایی محتوای دو متغیر (SWAP)

مثال:

A= 546 , B=645

SWAPA,B A= 645 , B=546

دستور نمایش حروف به صورت کوچک و بزرگ (Lcase , Ucase)

مثال:

X= "DFPM"

D= Lcase(x) "dfpm"

Z=ucase(x) "DFPM"

دستور جدا کردن مقادیر اعشاری از یکدیگر(INT , Frac)

مثال:

X=789.9874

A=INT(x) این دستور مقدار صحیح متغیر ۷۹۸ را از x جدا می‌کند

A=Frac (x) این دستور مقدار اعشاری متغیر ۹۸۷۴ را از x جدا می‌کند

دستور جدا کردن مقادیر پر ارزش و کم ارزش از یک متغیر (low , high)

این دو دستور مقادیر پر ارزش و کم ارزش بایت یک متغیر را جدا می‌کند.

X=&H123F

Y=low(x) y=&H3f

Z=Hrgh(x) z=&H12

دستور جدا کردن قسمتی از یک عبارت از سمت چپ یا راست (Left , Right)

X=FANIDFPM

A=LEFT(X=4) نام FANI از سمت چپ جدا می‌شود

B=Right (X=5) نام DFPM از سمت چپ راست می‌شود

دستور مشخص کردن طول یک عبارت LEN:

X= "DFPM "

Y=len(s) ‘4

دستورحذف فضای خالی LTRIM:

X= "D F P M "

Y=LTRIM(X) ‘ DFPM

دستور حذف و جایگزینی یک عبارتMID :

X= "DFPM "

Y= "AVR"

Mid(y,2,1)=x

دومین کاراکتر از متغیر y برداشته و به جای آن از متغیر x یک کاراکتر یعنی A قرار می‌دهد.

دستور انتقال بیت‌ها به چپ یا راست Rotate :

Y=3569

Rotate y, left,3 ‘y=6

Rotate y, Right,4 ‘y=3

دستور ایجاد فضای خالی Space :

Y=Space(3) **دستور گرد کردن کاراکترهای عددی**تا فضای خالی۳

دستور Fusing اعداد مورد نظر را رند می‌کند.

Y=236.63538

X=Fusing(y, " ###.# ") ‘X=236.635

دستور برگرداندن مقدار دلخواهی از جدول مورد نظر Lookup :

متغیر = lookup (نام جدول مورد نظر ، اندیس مقدار دلخواه)

Y=lookup(4,dfpm) ‘ Y=5 مقدار مورد نظر از صفر می‌باشد

F=lookup(0,dfpm) ‘Y=1

DFPM:

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

دستور برگرداندن عبارت دلخواه از جدول مورد نظر lookupstr :

متغیر = lookupstr (نام جدول مورد نظر ، اندیس مقدار دلخواه)

Y=lookupstr(5,dfpm) "God"

DFPM:

Data "in " , "the " , "name " , " of " , "a " , " God "

دستورات شرطی و پرش :

در هنگام برنامه نویسی بهتر است برای برنامه مورد نظر، نام (اسم) در نظر گرفت که برچسب (label) معروف است که با علامت : مشخص می‌شوند.

مثال:

Dfpm:

B:

و بهتر است نام برنامه اصلی با :main مشخص شود.

دستور پرش به طول بی‌نهایت Goto یا jmp :

…

Dfpm :

…

…

…

Goto Dfpm

End

برنامه به برچسب Dfpm پرش می‌کند و این عمل به طور مداوم ادامه پیدا می‌کند و یا می‌توان به صورت زیر نوشت.

…

Dfpm:

…

…

…

Jmp dfpm

End

دستور تکرار به صورت بی نهایت یا محدود DO … LOOP :

…

DO

A=A=1

LOOP

END

در برنامه مورد نظر به مقدار Aیکی اضافه می‌شود و این عمل همانند دستور قبلی تا نهایت تکرار می‌گردد و اگر بخواهیم شرط بگذاریم از عبارت Until (تا وقتی که) به صورت زیر استفاده می‌کنیم.

DO

A=A=1

Loop Until A=1000

End

در برنامه‌ی مورد نظر به مقدار A یک اضافه می‌شود و این عمل تا وقتی که A=1000 شود، تکرار می‌گردد.

دستور شرطی پرش gosub :

دستور شرطی gosub همانند دستور goto است و معمولاً برای یک بار تکرار استفاده می‌شود که باید در انتهای برچسب مورد نظر دستورRETURN نگاشته می‌شود.

دستور شرطی IF **(IF عبارت مورد نظر دستور then)**

endif

IF A=0 then

A= A+1

Else IF A=1then

Set portb.0

Else

Set portb.7

EndIF

دستور شمارش یا کانتر For – Next

(for متغیر = start to end step = مقدار گام حرکت)

Next متغیر

For A= 1 to 100 step 2

Next A

مقدار شمارش A به صورت 1,3,5,... با گام حرکت 2 است.

For B=1 to 10

Next B

مقدار شمارش B به صورت 1,2,3,… با گام حرکت یک است، در صورتی که مقدار STEP نوشته نشود گام حرکت یک در نظر گرفته می‌شود، همچنین گام حرکت می‌تواند منفی باشد.

For D = 50 to 1 step -5

Next d

مقدار شمارش از 50 تا 1 به صورت نزولی و گام حرکت 5- یعنی 40,45,50,… است.

نکته : بعد از دستور next می‌توانیم نام متغیر را بیان نکنیم.

دستور تا وقتیکه While – Wend

این دستور تا وقایکه شرط مورد نظر برآورده نشود انجام می‌شود.

…

B=10

Whie B <=100

INCR B

Wend

تا وقتیکه B <=100 است به مقدار B یکی افزوده می‌شود.

دستور Case :

با این دستور می‌توان حالت‌های مختلفی از اجرای دستورات را اجرا کرد.

متغیر مورد نظر select case

Case 1:

دستورات 1

Case 2:

دستورات 2

Case 3:

دستورات 3

…

…

Case else :

دستورات دیگر

End select

اگر متغیر مورد نظر با مقدار ۱ برابر باشد دستورات 1 اجرا می‌شود و اجرای برنامه از End select ادامه پیدا می‌کند و اگر متغیر مورد نظر با مقدار ۲ برابر باشد دستورات 2 اجرا می‌شود و اجرای برنامه از End select ادامه پیدا می‌کند و ... در غیر اینصورت دستورات دیگر اجرا می‌شوند و اجرای برنامه از End select ادامه پیدا می‌کند.

Select Case x

Case 1:

Set PortB.1

Case 2:

Reset PortB.1

Case else:

Wait 3

End Select

دستور خارج شدن از یک حلقه یا ساختار Exit:

Exite For یا Exite Do یا Exite While یا ...

با استفاده از دستور بالا می‌توان از یکی از ساختارFor یا Do یا While یا ... خارج شد.

دستور پرش با توجه به مقدار متغیر مورد نظر به برچسب‌های مورد نظر On Value

On متغیر gosub یا دستور goto دستور نام برنامه

X=0

On X goto Ali , Mahmodabad

X=1

On X goto man , hossein

Ali:

…

…

Man:

…

…

Mahmodabad:

…

Hasan:

…

…

دستور معرفی زیربرنامه DECLARE SUB

DECLARE SUB نام زیر برنامه ([By Val] یا [By REF] یا Var as Type)

اگر به صورت By Val باشد یک کپی از داده فرستاده می‌شود و در محتوای آن تغییری داده نمی‌شود و اگر به صورت By REF باشد در زمان ارسال داده، آدرس داده به زیربرنامه ارسال می‌شود و در محتوای آن تغییر داده می‌شود که در صورت استفاده نکردن از هیچ‌کدام آنها، به صورت پیش فرض ByREF در نظر گرفته می‌شود، در آخر زیربرنامه موردنظر باید END SUB تایپ گردد.

Declare SUB Ali

…

…

SUB Ali

…

…

END SUB

دستور معرفی تابع DECLARE Function:

DECLARE SUB نام تابع ([By Val] یا [By REF] یا Var as Type)

این دستور همانند دستور قبلی برای معرفی تابع مورد نظر است.

دستور فراخوانی زیربرنامه یا توابع : Call :

از این دستور برای فراخوانی توابع و زیر برنامه که با دستور Declaer مشخص شده‌اند استفاده می‌شود.

```
DIM  B As Word
DECLARE SUB Man
B=5
Man:
…
…
IF B<=10 Call  Man
…
…
```

متغیر محلی Local :

از این دستور برای تعریف متغیر در زیر برنامه‌ها و توابع استفاده می‌شود که تعریف آن همانند متغیرهای قبلی است، فقط از متغیر بیتی نمی‌توان در این‌جا استفاده کرد.

```
Sub AC
    Local X As Byte
    …
    …
….
```

دستورات تاخیری:

دستور توقف بر حسب ثانیه Wait

```
Wait  3      ۳ثانیه توقف
```

دستور توقف برحسب میلی‌ثانیه wait ms ((از 1 تا 65535))

```
Wait  500      ۵۰۰ میلی ثانیه توقف
```

دستور توقف برحسب میلی‌ثانیه waitus ((از 1 تا 255))

Waitus 60 ۶۰ میکرو ثانیه توقف

دستور توقف برحسب یک میلی‌ثانیه‌ای Dalay :

Delay ۱میلی‌ثانیه توقف

دستورات تعریف کردن میکرو، کریستال و پیکربندی‌ها:

تعریف میکرو $REGFILE

در خط اول هر برنامه معمولاً میکرو مورد نظر تعریف می‌شود که با توجه به خانواده‌های میکرو Attinyy , AT90s , ATmega نام میکرو به صورت‌های زیر تعریف می‌شود.

$Regfile = نام آی‌سی میکرو $Regfile

الف) نحوه‌ی تعریف آی سی‌های خانواده‌ی Attiny :

Attiny خانواده‌ی (Attiny 11/12/22/26/2313)

$Regfile=" attiny 12.dat "

$Regfile=" attiny 2313.dat "

ب) نحوه‌ی تعریف آی سی‌های خانواده‌ی AT90s :

$Regfile = " 1200 def.dat "

$Regfile = " 8535 def.dat "

ب) نحوه‌ی تعریف آی سی‌های خانواده‌ی ATmega :

$Regfile = " m32 def.dat "

$Regfile = " m8535 def.dat "

تعریف کریستال $Crystal :

$Crystal= عدد مربوطه

این دستور برای تعریف کردن کریستال(داخلی و خارجی) برحسب هرتز استفاده می‌شود.

$Crystal=1000000 1MHz

$Crystal=8000000 8MHz

$Crystal=16000000 16MHz

دستور تقسیم فرکانس Clock Division :

از این دستور می‌توان فرکانس مورد نظر را از اعداد ۲ تا ۱۲۸ تقسیم کرد.

Clock Division=4

دستور پایان برنامه END :

پایان هر برنامه باید دستور END نگاشته شود.

دستور توضیحات اضافه برای اطلاعات بیشتر REM یا ' :

علامت ' را در بیشتر دستورات قبلی دیده‌ایم اگر عبارت یا دستوری بعد از آن (') باشد توضیحات اضافه است که می‌توان از)' در شروع عبارت و (' در پایان عبارت استفاده کرد.

ادامه دستورات با –(Underline)

اگردستورات از یک خط بیشتر شد برای ادامه آن از علامت (-) استفاده می‌شود که از این علامت در زبان فارسی خودمان هم استفاده می‌کنیم.

آدرس شروع برنامه از حافظه Flash :

$ROM Start

برای اینکه بخواهیم شروع برنامه از آدرس دلخواه Flash memory در اختیارمان باشد از این دستور استفاده می‌شود و در صورتی که از این دستور استفاده نشود بصورت پیش‌فرض آدرس $H0000 انتخاب می‌شود.

$ROM Start = $H0000

دستورپیکربندی پورت‌ها Config port

از این پورت‌ها به عنوان ورودی یا خروجی استفاده کرد که برای این کار باید نوع ورودی و یا خروجی را مشخص کنیم.

Config port نام پورت = Input یا 0 (ورودی) , Output یا 1 (خروجی)

Config pin نام پورت ، شماره‌ی پورت = Input یا 0 (ورودی) , Output یا 1 (خروجی)

Cinfig port A=input	تمام port A به عنوان ورودی تعریف شده‌اند
Cinfig port C=Output	تمام port C به عنوان خروجی تعریف شده‌اند
Cinfig port A= 0	تمام port A به عنوان ورودی تعریف شده‌اند
Cinfig port D= 255	تمام port D به عنوان خروجی تعریف شده‌اند
Cinfig pin A.0= 1	اولین port A به عنوان خروجی تعریف شده است

هشتمین port D به عنوان ورودی تعریف شده است	Cinfig pin D.8= 0

در صورتکی بخواهیم مقاومت‌های pull-up یک port را فعال کنیم و از آن port به عنوان ورودی استفاده کنیم باید از دو دستور زیر استفاده شود.

DDR نام پورت = 0 : port نام پورت = 255 یا 1

DDR(Data Direction Register رجیستر جهت داده)

DDRB=0 ; port B =255

از تمام portB به عنوان ورودی استفاده می‌شود و مقاومت‌های pull-up آن فعال هستند.

DDRC.0=0 ; port C.0 =1

از تمام portC به عنوان ورودی استفاده می‌شود و مقاومت‌های pull-up آن فعال هستند.

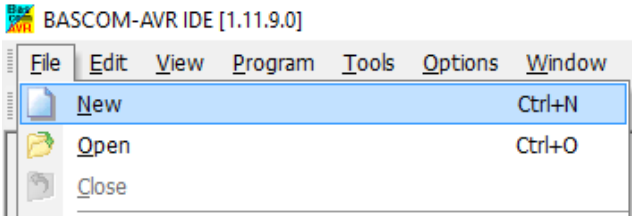
محیط برنامه نویسی BASCOM

معرفی منوهای محیط BASCOM

منوی FILE

ایجاد فایل جدید (FILE NEW)

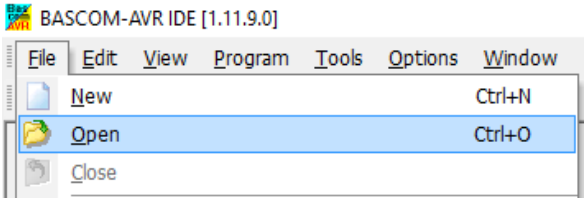
با انتخاب این گزینه یک پنجره جدید که شما قادر به نوشتن برنامه در آن هستید ایجاد می شود .



باز کردن فایل (OPEN FILE)

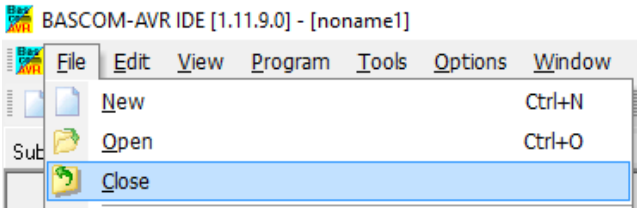
با انتخاب این گزینه شما قادر به فراخوانی فایلی که در حافظه موجود است می‌باشید .

BASCOM فایل‌ها را بصورت استاندارد ASCII ذخیره می‌کند بنابراین شما می‌توانید از ویرایشگری مثل NOTEPAD برای نوشتن برنامه استفاده کنید و سپس آنرا به محیط انتقال دهید.



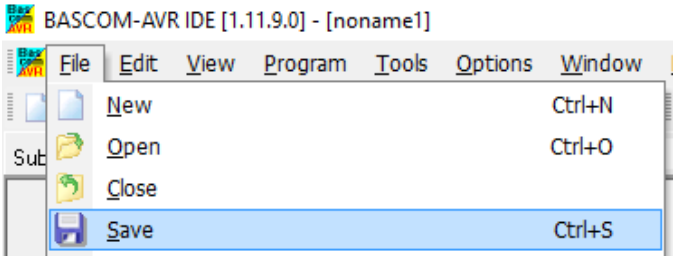
بستن فایل (CLOSE FILE)

این گزینه پنجره برنامه فعال را می بندد . اگر در فایل تغییری ایجاد کرده‌اید ابتدا باید قبل از بستن آن را ذخیره نمایید .



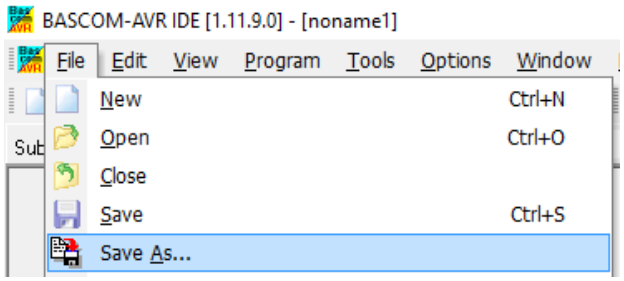
ذخیره فایل (FILE SAVE)

با این گزینه شما قادر به ذخیره فایل بصورت ASCII در کامپیوتر خواهید بود .



ذخیره کردن بعنوان (FILE SAVE AS)

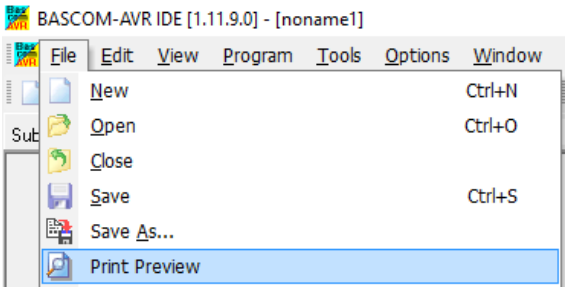
با این گزینه قادر خواهید بود فایل موجود را با نام دیگر ذخیره کنید.



ادامه منوی FILE...

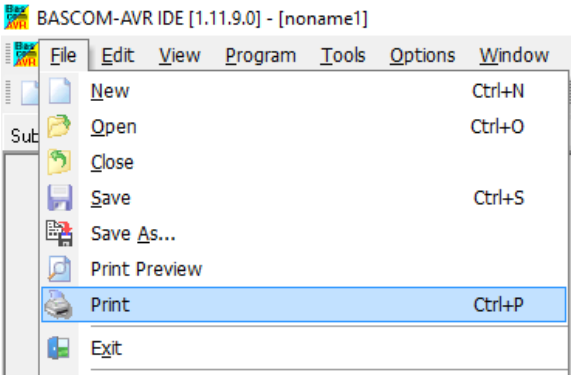
نمایش پرینت فایل (FILE PRINT PREVIEW)

این گزینه نشان می‌دهد که فایل متنی موجود برنامه در هنگام پرینت به چه صورت خواهد بود .



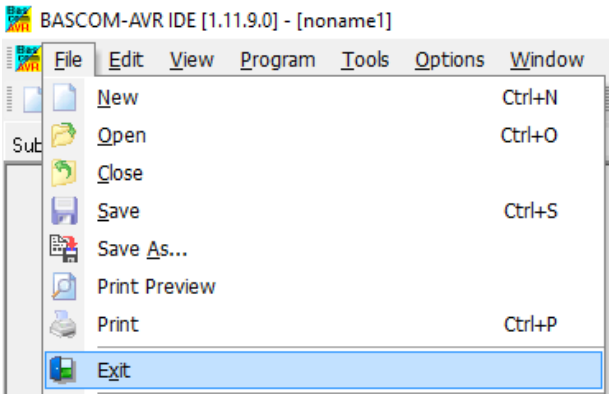
پرینت فا یل (FILE PRINT)

با این گزینه شما می توانید فایل موجود در برنامه را پرینت نمایید .



بستن فایل (Exit)

با این گزینه شما قادر خواهید بود از محیط BASCOM خارج شوید ولی در صورتی که شما در برنامه‌تان تغییری داده‌اید و آن را ذخیره نکرده‌اید , پیش از خروج هشدار می‌دهد.



منوی EDIT

EDIT UNDO

با این گزینه شما می توانید دستکاری اخیرتان در برنامه را از بین ببرید .

EDIT REDO

با این گزینه شما می توانید دستکاری اخیرتان را که از بین برده بودید دوباره برگردانید .

EDIT CUT

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

EDIT COPY

با این گزینه شما می توانیدمتن انتخاب شده را کپی کرده و به محل جدیدی انتقال دهید .

EDIT PAST

با این گزینه شما می توانید متنی را که قبلا COPY یا CUT کرده بودید در محل مورد نظر بچسبانید .

منوی EDIT ...

EDIT FIND

با این گزینه شما می توانید متنی را در برنامه‌تان جستجو کنید .

EDIT FIND NEXT

با این گزینه شما می توانید متن مورد جستجو را دوباره جستجو نمایید .

EDIT REPLACE

با این گزینه شما می توانید متنی را جایگزین متن موجود در برنامه نمایید یعنی در قسمت TEXT TO FIND متن مورد جستجو که باید توسط متن دیگری جایگزین شود را تایپ کنید و در قسمت REPLACE WITH متنی را که باید جایگزین شود تایپ می‌کنیم .

EDIT GOTO

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

منوی EDIT ...

EDIT TOGGLE BOOKMARK

با این گزینه شما می‌توانید در جاهای خاصی از برنامه که مورد نظر شماست نشانه گذاری کنید و به آنها توسط دستور EDIT GOTO BOOKMARK دسترسی پیدا کنید .

EDIT GOTO BOOKMARK

با این گزینه شما می توانید به نشانه هایی که قبلا گذاشته اید .

EDIT IDENT BLOCK

با این گزینه شما می‌توانید متن انتخاب شده را به اندازه یک TAB به سمت راست منتقل کنید .

EDIT UNIDENT BLOCK

با این گزینه شما می‌توانید متن انتخاب شده را به اندازه یک TAB به سمت چپ منتقل کنید .

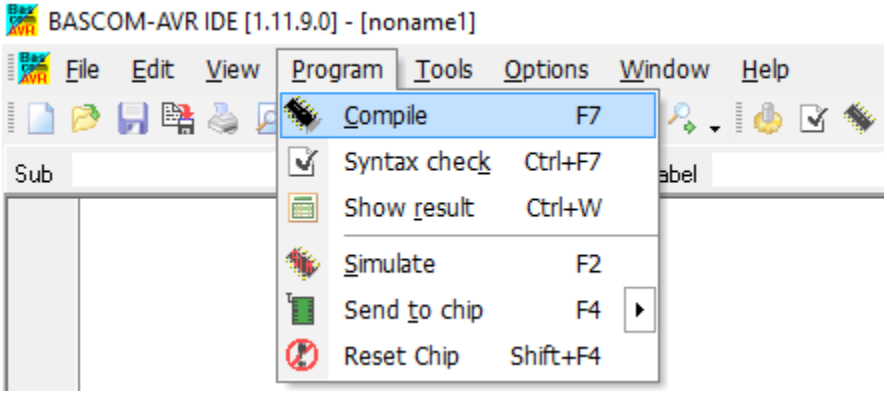
منوی PROGRAM

PROGRAM COMPILE

با این گزینه (یا کلید F7) شما قادر به ترجمه برنامه به زبان ماشین (COMPILE) خواهید بود برنامه شما با انتخاب این گزینه پیش از COMPILE ذخیره خواهد شد و فایل‌های زیر به انتخاب شما در OPTION COPILER SETTING ایجاد خواهند شد :

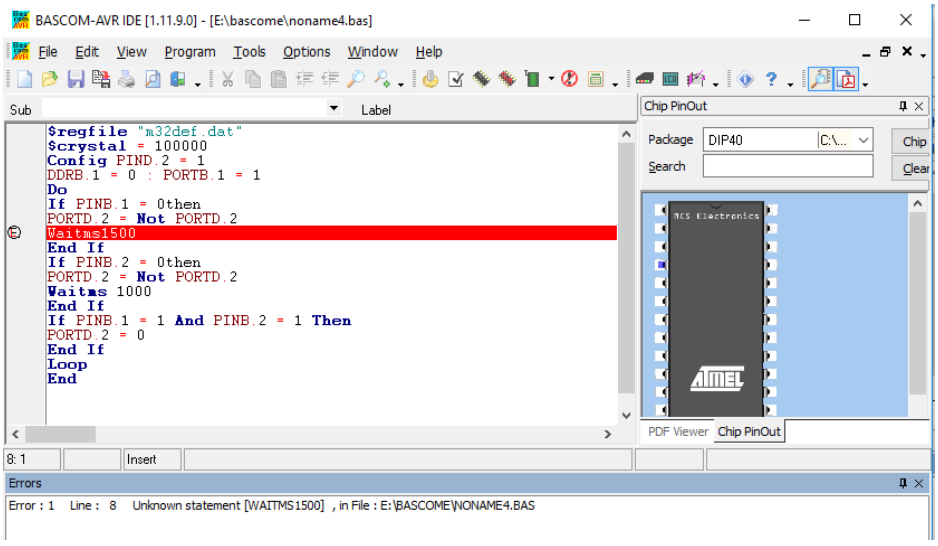
- XX.BIN فایل باینری که می تواند در میکروکنترلر PROGRAM شود .

- XX.DBG فایل DEBUG که برای نرم افزار شبیه ساز BASCOM مورد نیاز است .
- XX.OBJ فایل OBJECT که برای نرم افزار AVR STUDIO مورد نیاز است .
- XX.RPT فایل گزارشی
- XX.HEX فایل هگزادسیمال اینتل که برای بعضی از انواع PROGRAMMER ها مورد نیاز است .
- XX.ERR فایل خطا که فقط در هنگام بروز خطا ایجاد می شود.
- XX.EPP داده های که باید در EPROM برنامه ریزی شود در این فایل نگهداری میگردند .



منوی PROGRAM ...

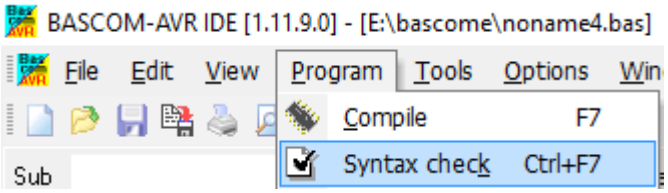
اگر خطایی در برنامه موجود باشد شما پیغام خطا را در یک کادر محاوره‌ای دریافت خواهید کرد و COMPILE متوقف می‌شود. با کلیک بر روی هر کدام از آنها به خطی که خطا در آن رخ داده پرش خواهید کرد .



PROGRAM SYNTAX CHECK

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

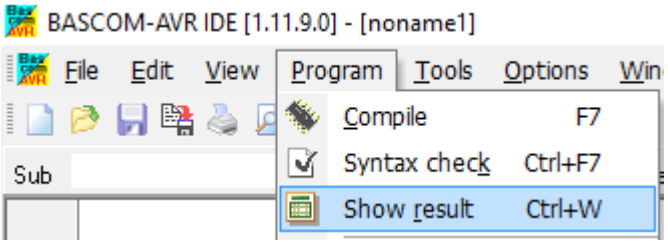
.



PROGRAM SHOW RESULT

از این گزینه برای دیدن نتیجه COMPILE می‌توان استفاده کرد .

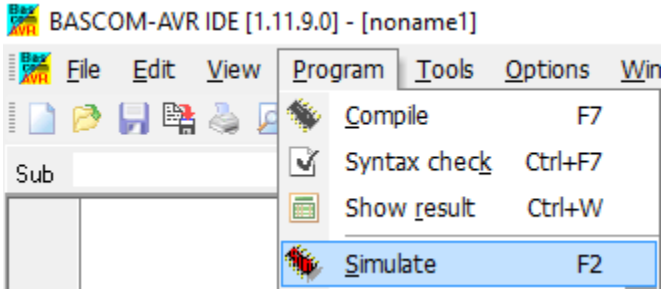
گزینه OPTION COMPILE OUTPUT را برای تعیین اینکه کدام فایل‌ها باید ایجاد شوند را ببینید . فایل‌هایی که محتوای آنها قابل مشاهده اند REPORT ERROR می‌باشند .



منوی PROGRAM ...

PROGRAM SIMULATOR

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



SEND TO CHIP

توسط این گزینه یا کلید F4 پنجره محیط برنامه ریزی ظاهر خواهد شد. شما می توانید توسط این گزینه میکرو مورد نظر خود را PROGRAM کنید .

منویTOOLS

TERMINAL EMULATOR

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

LCD DESIGNER

توسط این گزینه می توانید کاراکترهای دلخواه خود را طراحی نمایید و بر روی LCD نمایش دهید.

منوی TOOLS ...

GRAPHIC CONVERTOR

با کلیک بر روی این منو پنجره محیط GRAPHIC CONVERTOR برای تبدیل تصویر با پسوند BMP.* به تصویری با پسوند BGF.* که قابل نمایش بر روی GRAPHIC LCDاست ظاهر می‌شود .

فایل دلخواه خود را با پسوند BMP.* توسط دکمه LOAD وارد کرده و سپس با دکمه SAVE آنرا در کنار برنامه خود با پسوند BGF.* (BASCOM GRAPHIC FILE) ذخیره کنید .فایل تبدیل شده بصورت سیاه و سفید دوباره نمایش داده می‌شود و با کلیک بر روی دکمه OK می توان از محیط خارج شد . فایل ذخیره شده با فراخوانی در برنامه قابل نمایش بر روی LCD گرافیکی است . انتخاب نوع LCD توسط قسمتLCD TYPE انجام می گیرد . فونت نوشتاری نیز می تواند 8*6 یا 8*8 پیکسل باشد .

منوی OPTION

OPTION COMPILER

با این منو شما می‌توانید گزینه‌های مختلف کامپایلر را طبق زیر اصلاح نمایید :

OPTION COMPILER CHIP

انتخاب میکرو برای برنامه ریزی توسط این گزینه انجام می‌شود . در صورتی که از دستور $REGFILE در برنامه استفاده کرده‌اید به انتخاب میکرو توسط این گزینه نیازی نیست .

OPTION COMPILER OUTPUT

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

OPTION COMPILER 12C,SPI,1WIRE

توسط این گزینه می توان پایه های مربوط به ارتباطات 12C SPI و 1 WIREرا تعیین کرد .

منوی OPTION ...

OPTION COMPILER COMMUNICATION

نرخ انتقال (BOUD RATE) ارتباط سزیال توسط این گزینه تعیین می‌شود که می توان یک نرخ جدید نیز تایپ کرد. گزینه FREQUENCY انتخاب فرکانس کریستال استفاده شده است که می تواند فرکانس اختیاری نیز باشد .

OPTION COMPILER LCD

این گزینه دارای قابلیت های زیر می باشد :

در قسمت LCD TYPE نوع LCD را مشخص می کنیم .گزینه BUS MODE مشخص می‌کند LCD بصورت ۸ بیتی یا ۴ بیتی کار می‌کند .توسط گزینه DATA MODE تعیین می‌کنیم LCD بصورت PIN کار کند یا BUS و گزینه LCD ADDRESS مشخص کننده آدرس LCD در مد BUS است .

در صورت پیکره بندی هر یک از امکانات فوق در برنامه نیازی به تنظیم کردن آنها در این منو نیست .

OPTION PROGRAMMER

در این منو شما می توانید PROGRAMMER مورد نظر خود را انتخاب نمایید .

معرفی محیط شبیه سازی (SIMULATOR)

نوار ابزار در این محیط

RUN

با فشردن این دکمه شبیه سازی آغاز می شود .

PAUSE

باعث توقف موقت شبیه سازی می شود و با فشردن دکمه RUN شبیه سازی ادامه پیدا می‌کند .

STOP

باعث توقف کامل شبیه سازی برنامه جاری می‌شود .

STEP INTO CODE

با استفاده از این دکمه می‌توان برنامه را خط به خط اجرا نمود و هنگام فراخوانی توابع به داخل آنها رفته و مراحل اجرای آنها را بررسی کرد . این کار را با فشردن کلید F8 نیز می‌توانید انجام دهید بعد از هر بار اجرای این دستور شبیه سازی به حالت PAUSE می‌رود .

STEP OVER

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

RUN TO

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

شبیه سازی سخت افزاری THE HARDWARE SIMULATOR

با کلیک بر روی این گزینه پنجره ای ظاهر می‌شود . که قسمت بالایی یک LCD مجازی می‌باشد که برای نشان دادن داده‌های فرستاده شده به LCD استفاده می شود . نوار LED های قرمز رنگ پایین خروجی پورت‌ها را نشان می‌دهد . با کلیک بر روی هر یک از LED های سبز رنگ که بعنوان ورودی هستند وضعیت آن معکوس می‌شود و روشن شدن LED بمنزله یک کردن پایه

پورت است .یک صفحه کلید نیز تعبیه شده است که با دستور ( )GETKBD در برنامه قابل خواندن می‌باشد . در ضمن مقدار آنالوگ نیز هم برای مقایسه کننده آنالوگ و هم برای کانال های مختلف ADC قابل اعمال است.

REGISTERS

این دکمه پنجره ثبات‌ها را با مقادیر قبلی نمایش می‌دهد . مقدارهای نشان داده شده در این پنجره هگزادسیمال می باشد که برای تغییر هر کدام از آنها روی خانه مربوطه کلیک کرده و مقدار جدید را وارد کنید .

I/O REGISTERS

برای نمایش ثبات‌های I\O استفاده می شود . که مانند R قابل مقدار دهی است .

VARIABLES

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

برای تماشای یک متغیر آرایه ای می توانید نام متغیر همراه با اندیس آنرا تایپ کنید و برای حذف هر سطر می توانید دکمه CTRL+DEL را فشار دهید .

WATCH

این گزینه برای وارد کردن وضعیتی که قرار است در خلال شبیه سازی ارزیابی شود مورد استفاده قرار می‌گیرد و هنگامی که وضعیت مورد نظر صحیح شد شبیه سازی در حالت PAUSE قرار خواهد گرفت. حالت مورد نظر را در مکان مورد نظر تایپ نموده و دکمه ADD-BUTTON را فشار دهید . هنگامی‌که دکمه MODIFY-BUTTON فشار داده شود، وضعیت مورد نظر را مورد بازنگری قرار می‌دهد و می‌توان ارزش آنرا تغییر داد. برای حذف هر وضعیت شما باید آنرا انتخاب کرده و دکمه REMOVE را فشار دهید .

LOCAL

متغیرهای محلی موجود در SUB یا FUNCTION را نشان میدهد . البته نمیتوان متغیری را به آن اضافه نمود .

UP

وضعیت ثبات وضعیت (STATUS REG) را نشان میدهد . FLAG ها را می‌توان توسط کلیک بر روی CHECK BOX ها تغییر وضعیت داد .

INTERRUPTS

این گزینه منابع وقفه را نشان می‌دهد . هنگامیکه هیچ ISR برنامه نویسی نشده باشد , همه دکمه ها غیر فعال خواهند بود و اگر ISR نوشته شود، دکمه مربوط به آن فعال می‌شود و با کلیک بر روی هر کدام از دکمه ها , وقفه مربوطه اجرا می‌شود . در ضمن می‌توان روی یک پایه خاص پالس نیز ایجاد نمود .

برنامه نویسی در محیط Bascom AVR

ابتدا از منوی ابزار گزینه File⟶ new را انتخاب کنید تا صفحه‌ای برای نوشتن برنامه آماده شود. اولین گام برای نوشتن برنامه این است که نوع میکرو را مشخص کنید از این رو می‌توانید از جدول زیر با توجه به محدوده ولتاژ و محدوده فرکانسی مورد نظر، نوع میکرو را مشخص کنید.

پس از مشخص کردن نوع میکرو کافی است در سطر اول دستور $regfile " m32def.dat " را بنویسید.

با این کار به سیستم می‌فهانیم که قرار است از میکرو atmega 32 استفاده کنیم.پسوندdef.dat نیز همان بانک اطلاعاتی است که از قبل در کتابخانه نرم افزار وجود دارد و در آن انواع میکروها تعریف شده است.

در خط دوم نیاز است که فرکانس کاری مشخص شود و با دستور$crystal=1000000 مشخص می‌شود.

در سطر سوم باید وضعیت ورودی‌ها و خروجی‌های سیستم را با دستور Config مشخص کنیم.

Config input ورودی سیستم

Config output خروجی سیستم

پس از مشخص شدن ورودی وخروجی به سراغ نوشتن برنامه می‌رویم. در بالا به برخی از دستورات برنامه‌نویسی در محیط bascome avr به زبان بیسیک توضیح داده شده است.

اما برای درک بهتر دستورات در محل بکارگیری هرکدام از دستورات در برنامه بطور مجزا و مفصل شرح داده خواهد شد.

بعنوان مثال:

می‌خواهیم برنامه ای بنویسیم که یک LED به مدت 500 میلی ثانیه روشن و خاموش شود.

همانطور که گفته شد با توجه به محدوده فرکانسی و محدوده ولتاژ نوع میکرو را مشخص کرده و دستورات زیر را می‌نویسیم.

مشخص کردن نوع میکرو	$regfil"m32def.dat"
مشخص کردن فرکانس کاری	$crystal=1000000
مشخص کردن خروجی سیستم	Config portb=output

با نوشتن دستور فوق portb بعنوان خروجی در نظر گرفته می‌شود.

همانطور که میدانیم میکروATMEGA32 از ۴ پورت به نام های A , B , C , D تشکیل شده است که هر پورت شامل ۸ پین میباشد.

با دستور فوق هر ۸ پایه port b بعنوان خروجی در نظر گرفته می‌شود. اما اگر شماره pin را مشخص کردیم آنگاه فقط یک پایه بعنوان خروجی در نظر گرفته می‌شود.

مثلاً config portb.3 که در این صورت پایه شماره ۴ مربوط به port b که در واقع همان portb.3 است بعنوان خروجی فعال می‌شود.

حالا می‌رویم سراغ اصل برنامه:

می‌دانیم در الکترونیک و دیجیتال Set , On, +5v, Vcc, 1 کنایه از روشن شدن یا همان یک شدن است و بلعکس Reset, 0 , Off, -5v, Gnd کنایه از خاموش شدن یا صفر شدن است.

Set	On	+5v	Vcc	1
Reset	Off	-5v	Gnd	0

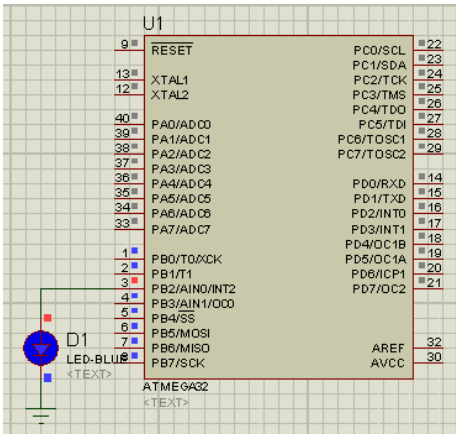
اگر منظور از سوال، تکرار خاموش و روشن شدن باشه باید از دستور LOOP .....DO استفاده کرد که به معنی تکرار حلقه بی‌نهایت است. یعنی هر آنچه در این حلقه قرار دارد بصورت بی‌نهایت تکرا می‌شود.

بنابراین می‌توانیم اینگونه بنویسیم:

```
$regfil"m32def.dat"          مشخص کردن نوع میکرو
$crystal=1000000             مشخص کردن فرکانس کاری
Config portb.2=output        مشخص کردن port خروجی
Do                           شروع کن
Set portb.2                  پایه سوم پورت b را روشن کن
Waitms 500                   به مدت ۵۰۰ میلی ثانیه صبر کن
Reset portb.2                پایه سوم پورت bرا خاموش کن
Waitms 500                   به مدت ۵۰۰ میلی ثانیه صبر کن
Loop                         تکرار حلقه
End                          پایان برنامه
```

خوب همانطور که در برنامه بالا دیده شده یک led به portb.2 وصل و به مدت 500ms روشن و 500ms خاموش می‌شود.همین برنامه را بصورت زیر هم می‌توان نوشت:

```
$regfile "m32def.dat"
$crystal = 100000
Config Portb.2 = Output
Do
Portb.2 = 1
Waitms 500
Portb.2 = 0
```



```
Waitms 500
Loop
End
```

در واقع فقط بجای Set , Reset از اعداد صفر و یک استفاده کردیم که همان مفهوم روشن و خاموش است.

حالا اگه قرار باشه روی همون portخروجی بجای یک عدد led تعداد 8 عدد led را روشن و خاموش کنیم کافی است در سطر سوم از برنامه port مورد نظر را بصورت کامل بعنوان خروجی تعریف کنیم. یعنی

```
Config Portb = Output
```
در اختیار گرفتن portb بصورت کامل بعنوان خروجی

سپس کافی است مثل برنامه قبلی عمل کنیم، با این تفاوت که فقط یک پایه را در نظر نگیریم و همه‌ی پایه ها را در نظر بگیریم:

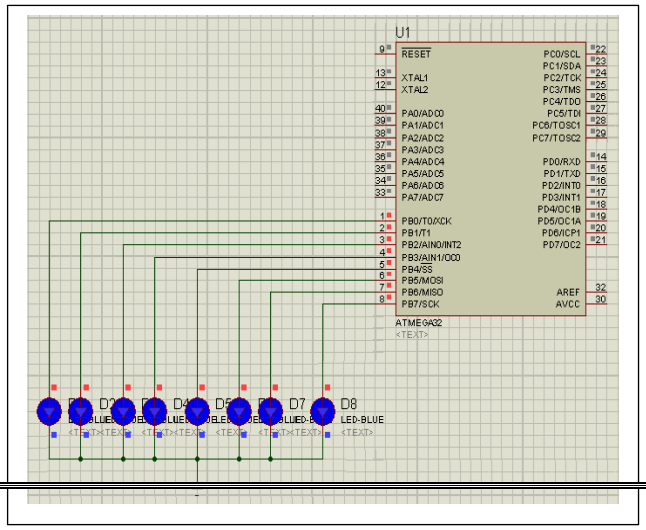
```
$regfile "m32def.dat"
$crystal = 100000
Config Portb = Output
Do
Set Portb
Waitms 500
Reset Portb
Waitms 500
Loop
End
```

اولویت اول درنوشتن برنامه این است که با استفاده از دستورات برنامه نویسی بتوانیم برنامه‌ای بنویسیم که حداقل تعداد سطر را دارا باشد تا سرعت پردازش داده افزایش یابد و همچنین در صورت وجود error احتمالی به راحتی بتوان آن را رفع کرد.

به همین منظور در برنامه بالا اگر قرار باشد همان 8 عدد led روشن و خاموش شود کافی است از دستور toggle استفاده شود. یعنی هر مقداری که دارید، معکوس می‌شود. اگر مقدار اولیه ما صفر بود تبدیل به یک و اگر یک بود تبدیل به صفر می‌شود.

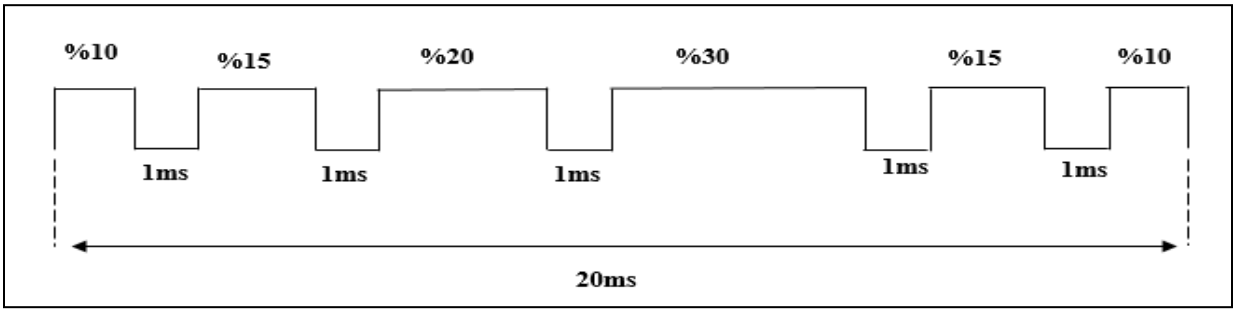
$1 \longrightarrow 0$ و $0 \longrightarrow 1$

```
$regfile "m32def.dat"
$crystal = 100000
Config Portb = Output
Do
Toggle Portb        هر مقداری که داریم معکوس می‌کند
Waitms 500
```



```
Loop
End
```

اکنون می‌خواهیم برنامه ای بنویسیم که شکل موج خروجی زیر را نشان دهد؟

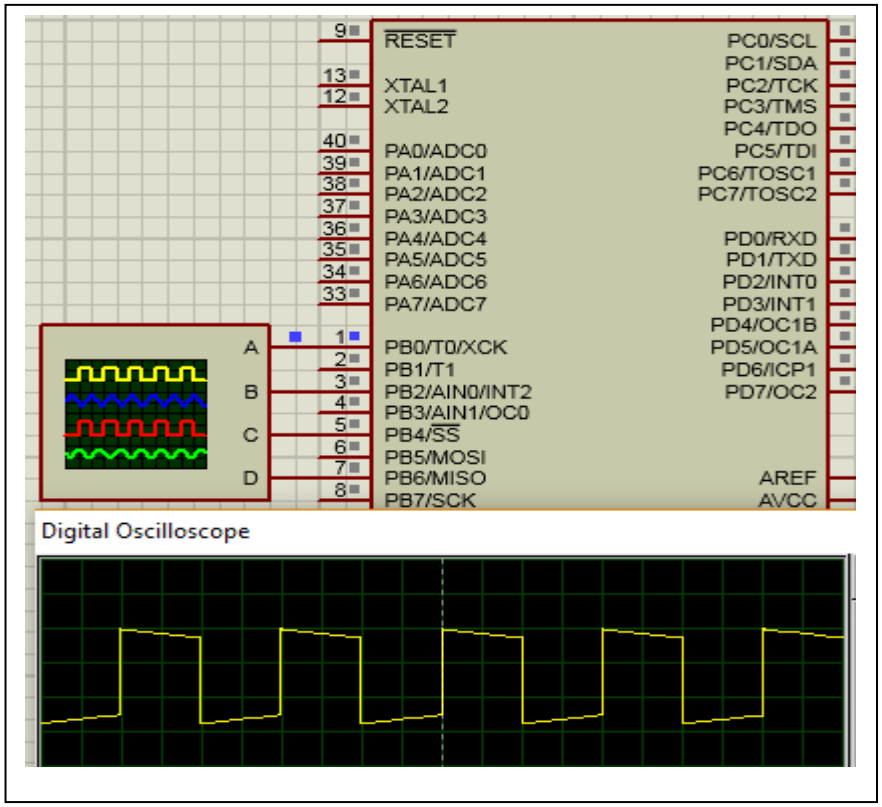


با کمی دقت می‌توانیم دریابیم که شکل موج فوق دارای لبه های بالایی یعنی Set یا یک هستند و همچنین لبه های پایینی یعنی reset یا صفر هستند. اکنون با استفاده از تجربه برنامه قبلی می‌توانید برنامه ای بنویسید که شکل موج بالایی را نشان دهد.

همانطوریکه در شکل موج مشخص است لبه‌های پایین رونده همگی به مدت 1ms می‌باشند و در واقع مدت زمان خاموش بودن یا همان صفر بودن آنها 1ms می‌باشد. اما در این شکل موج ، لبه‌های بالا رونده هستند که باید مدت زمان روشن بودنشان را مشخص کنیم. کاملا واضح است مجموع طول مدت زمان شکل موج 20ms است و در نتیجه با توجه به درصدهای ذکر شده در لبه‌های بالارونده می‌توان دریافت که باید مقدار درصد را در کل مدت زمان شکل موج(20ms) ضرب کنیم و نتیجه بدست آمده همان مدت زمان روشن بودن هر لبه خواهد بود. بعنوان مثال 10% مدت زمان 20ms می‌شود 2ms و یا 30% مدت زمان 20ms می‌شود 6ms و به همین ترتیب برای 15% و 20% ، 3ms و 4ms خواهد بود. با نوجه به توضیحات بالا به راحتی می‌توانیم کد برنامه را بنویسیم.

```
$regfile "m32def.dat"
$crystal = 1000000
```

پس از نوشتن کد برنامه کافی‌ست در برنامه شبیه سازی بجای led از اسیلوسکوپ استفاده کنیم در این حالت شکل موج پالسی بالا را مشاهده خواهیم کرد.

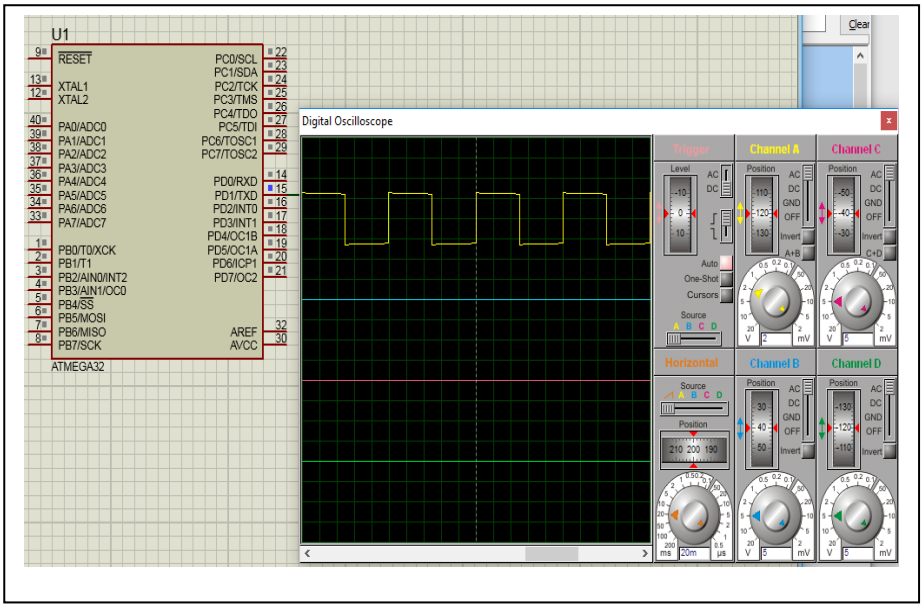



```
Config Portb.0 = Output
Do
Portb.0 = 1
Waitms 2
Portb.1 = 0
Waitms 1
Portb.0 = 1
Waitms 3
Portb.1 = 0
Waitms 200
Portb.0 = 1
Waitms 4
Portb.1 = 0
Waitms 1
Portb.0 = 1
Waitms 6
Portb.1 = 0
Waitms 1
Portb.0 = 1
Waitms 3
Portb.1 = 0
Waitms 1
Portb.0 = 1
Waitms 2
Portb.1 = 0
Waitms 1
Loop
End
```

برنامه‌ای بنویسید که روی یکی از پایه‌های میکرو یک پالس به مدت 100ms تولید شود.

```
$regfile = "m32def.dat"
```

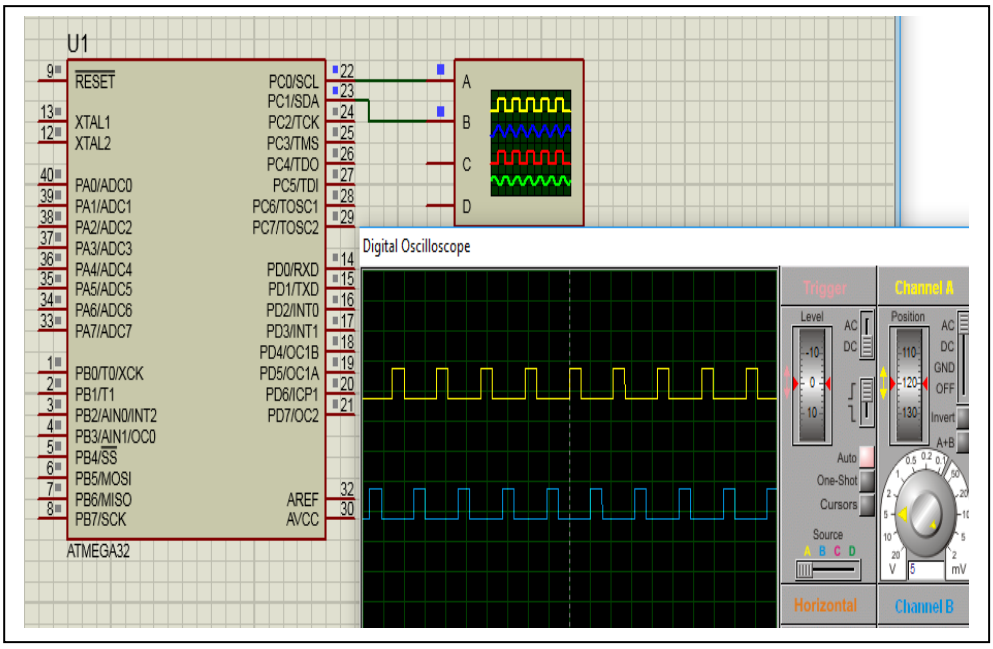
```
$crystal = 1000000
Config Portd.1 = Output
Config Timer1 = Timer , Prescale = 1
Timer1 = 0
Start Timer1
Do
If Timer1 > 50000 Then
Stop Timer1
Timer1 = 0
Start Timer1
Portd.1 = Not Portd.1
End If
Loop
End
```



برنامه‌ای بنویسید که دو پالس مربعی همزمان با فرکانس‌های 2khz و 8khz رابر روی پایه‌های pc.0 و pc.1 ایجاد کند؟

```
$regfile "m32def.dat"
```

```
$crystal = 1000000
Config Portc.0 = Output
Config Portc.1 = Output
Do
Portc.0 = 1
Waitms 0.5
Toggle Portc.0
Waitms 0.01
Portc.1 = 1
Waitms 0.125
Toggle Portc.1
Waitms 0.01
Loop
End
```



بعد از آشنائی با نرم افزار BASCOM-AVR و همچنین نرم افزار شبیه سازی PROTEUS و نوشتن کدهای دستوری در خصوص روشن و خاموش شدن LED حتماً قادر خواهید بود برنامه‌های کاربردی بیشتری را مورد استفاده قرار دهید.

دیود نوری به زبان ساده

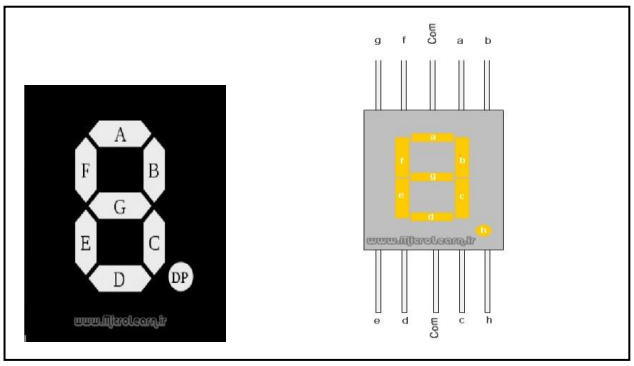
LED یا دیود نوری یک قطعه‌ی ساده‌ی الکترونیکی است که هنگامی که جریان الکتریکی از آن می‌گذرد از خودش نور ساطع می‌کند. دیود نوری نیز مانند همه‌ی دیودهای دیگر دارای دو سر آند (مثبت) و کاتد (منفی) است. به طوری که فقط جریانی را هدایت می‌کند که طرف مثبت ولتاژ به سمت آند و جهت منفی ولتاژ به سمت کاتد متصل باشد. یعنی جریان از آند به سمت کاتد جاری می‌شود. LED هایی که برای کاربردهای عمومی مورد استفاده قرار می‌گیرند برای روشن شدن نیاز به جریانی در حدود ۷۵ میلی آمپر و یا کمتر دارند تا روشن شوند، به همین دلیل برای محدود کردن جریان از مقاومت هم استفاده می‌شود. در میکروکنترلر AVRجریانی که هر Pin پشتیبانی می‌کند در حدود ۲۰ میلی آمپر است. پس می‌توان با حذف مقاومت یک LED را به طور مستقیم توسط میکروکنترلر روشن کرد. در واقع اگر مقاومتی اضافه شود شدت نور LED کم می شود. در شکل زیر یک LED به طور صحیح راه اندازی شده است.

سون سگمنت چیست؟

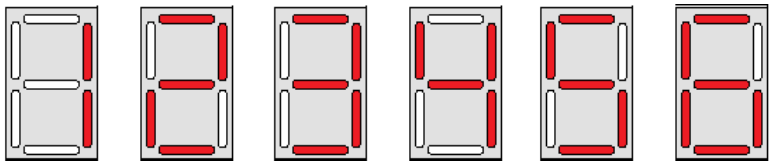
امروزه سون سگمنت جزء رایج‌ترین نمایشگرهای موجود در بازار است چرا که ارزان است، برنامه‌ریزی آن آسان بوده و از لحاظ اندازه هم قطعه‌ای کوچک می‌باشد. سون سگمنت‌ها در ساعت‌های دیجیتال، شمارنده‌ها و دیگر وسایل الکترونیکی جهت نمایش اطلاعات عددی مورد استفاده قرار می‌گیرند. همانطور که از اسم آن پیداست سون‌سگمنت دارای هفت قطعه دیود نوری می‌باشد که به فرمی مستطیلی مانند چیده شده‌اند. هر LED یک سگمنت یا قطعه نامیده می‌شود چون با روشن شدن هر قطعه، آن قطعه در تشکیل شکل عدد مورد نمایش شرکت می‌کند. یک قطعه‌ی دیود نوری هشتم هم وجود دارد که نقطه اعشاری را معین می کند.

چیدمان سون سگمنت

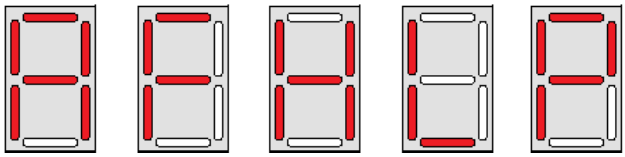
همانطور که در شکل زیر نمایش داده شده است هفت LED موجود در سون سگمنت با حروفA ، B، C، D، E، Fو G برچسب گذاری می‌شوند. البته بعضی از سون سگمنت‌ها دارای یک LED دیگر نیز برای نمایش ممیز اعشاری (dp) هستند که در شکل با حرف H نمایش داده شده است.



سون سگمنت ها برای نمایش ارقام عددی مورد استفاده قرار می‌گیرند. شکل زیر نشان می‌دهد که برای نمایش اعداد ۱ و ۲ و ۳ و ۴ و ۵ و ۶ کدامLED ها باید روشن شوند. رقم های دیگر (۷ و ۸ و ۹ و ۰) نیز با روشن کردنLED های مناسب قابل نمایش می باشند:



یک سون سگمنت همچنین برای نمایش بعضی از حروف انگلیسی نیز قابل استفاده است. شکل زیر نشان می‌دهد که برای نمایش حروف A ، F , H , P, T , L کدامLED ها باید روشن شوند:



انواع سون سگمنت

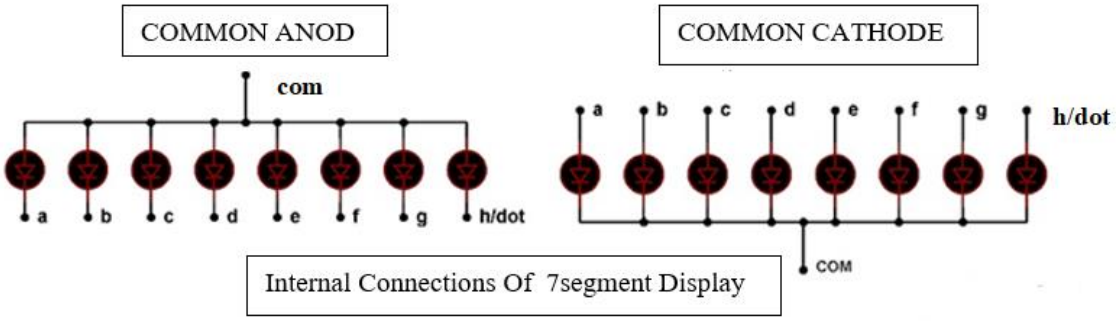
در بازار دو نوع سون سگمنت وجود دارد:

۱-کاتد مشترک :

در سون سگمنت کاتد مشترک سر منفی یا کاتد تمامی LED ها از داخل به پایه ی مشترک متصل شده اند که باید به زمین وصل شود. به منظور روشن کردن LED مورد نظر نیاز است تا ولتاژ +۵ ولت به قطب مثبت و یا آند آن LED اعمال شود، و در صورت نیاز یک مقاومت محدودکننده‌ی جریان نیز به قطب مثبت متصل گردد.

۲ - آند مشترک :

در سون سگمنت آند مشترک سرهای مثبت یا آند همه‌یLED ها از داخل به پایه‌ی مشترک سون سگمنت متصل شده‌اند که باید به منبع ولتاژ +۵ ولت وصل گردد. جهت روشن کردن یک LED باید پین مربوط به کاتد آن را به زمین متصل نمود.



سون سگمنت‌ها در اندازه های مختلف ۰,۲۸ ، ۰,۳، ۰,۳۲، ۰,۳۶، ۰,۳۹، ۰,۴، ۰,۵، ۰,۵۶، ۰,۶، ۰,۸، ۱,۰، ۱,۲، ۱,۵، ۱,۸، ۲,۰، ۲,۳، ۳,۰، ۴,۰، ۵,۰ و ۷,۰ اینچی در بازار موجود هستند. همچنین سون سگمنت‌ها در رنگ‌های مختلف قرمز، سبز، زرد، نارنجی، آبی و سفید عرضه می گردند.

چگونگی تشخیص آند مشترک یا کاتد مشترک بودن 7 segment

جهت مشخص کردن آند یا کاتد مشترک بودن سون سگمنت ابتدا باید سر مشترک سون سگمنت را پیدا کرد که معمولا پایه وسط (بالا و یا پائین) سون سگمنت می‌باشد، سون سگمنت ها از ۸ دیود نوری تشکیل شده اند که یک سر دیودها به هم وصل شده‌اند و به عنوان سر مشترک استفاده می شوند و در واقع از داخل به هم متصلند، ابتدا باید یک سر مولتی متر را روی پایه وسط قرار دهید و سر دیگر مولتی متر را روی پایه‌های دیگر جابجا کنید، اگر به تمام پایه‌ها راه داد (دیود ها را روشن نمود) این پایه، پایه مثبت است و همه دیود ها سالم اند. ولی اگر در این میان به یکی راه نداد این دیود سوخته است .

نکته:

در سون سگمنت آند مشترک تمام سرهای مثبت دیودها به همدیگر وصل شده اند و در کاتد مشترک تمام سرهای منفی. حال برای تشخیص نوع سون سگمنت (آند یا کاتد مشترک بودن) ابتدا باید مولتی متر را در حالت تست دیود قرار داد، سپس اگر کابل مثبت (VΩhz) مولتی‌متر را به پایه‌ی وسط اتصال دهیم و به ازای اتصال کابل منفی (COM) مولتی‌متر هریک از سگمنت‌ها روشن شوند ، در این حالت می‌گوئیم سون سگمنت کاتد مشترک است، و برعکس اگر کابل منفی (COM) مولتی متر را به پایه‌ی وسط اتصال دهیم و به ازای اتصال کابل مثبت (VΩhz) هر یک از سگمنت‌ها روشن شود در این حالت می‌گوئیم سون سگمنت آند مشترک است.

با اینکار ضمن تشخیص آند و کاتد مشترک بودن سون سگمنت، می‌توان از سالم بودن هر یک از سگمنت‌ها اطلاع پیدا کرد و همچنین می‌توان مشخص نمود که هر پایه مربوط به کدام حرف انگلیسی نام گذاری شده است.

راه اندازی سون سگمنت با میکروکنترلر :AVR

راه اندازی مستقیم :

در راه اندازی مستقیم ۸ پایه‌ی سون سگمنت را به هشت پایه‌ی خروجی از میکروکنترلر متصل می‌کنیم. همچنین باید بدانیم که مثلا برای نمایش عدد ۵ کدامLED ها باید روشن شوند و سپس پایه‌های مربوط به آنLED ها را یک کنیم. به همین دلیل نیاز به یک جدول درستی داریم.

جدول درستی برای نمایش ارقام 0 تا 9 و حروف قابل نمایش:

برای نمایش اعداد صفر تا ۹ بر روی یک سون سگمنت کافی است تا هفت پین مربوط به اعداد و یک پین نقطه‌ی اعشاری را برنامه نویسی کنیم. برای این کار از جدول زیر استفاده می شود.

جدول کد های hex در وارقام کاراکترها برای Sevensegment آند مشترک(C.A) و کاتد مشترک(C.K)

	کاتد مشترک										آند مشترک									
عدد	کد	p	g	f	e	d	c	b	a	عدد	کد	p	g	f		e	d	c	b	a
0	3F	0	1	1	1	1	1	1	1	0	C0	1	1	0		0	0	0	0	0
1	06	0	0	0	0	0	1	1	0	1	F9	1	1	1		1	1	0	0	1
2	5B	0	1	0	1	1	0	1	1	2	A4	1	0	1		0	0	1	0	0
3	4F	0	1	0	0	1	1	1	1	3	B0	1	0	1		1	0	0	0	0
4	66	0	1	1	0	0	1	1	0	4	99	1	0	0		1	1	0	0	1
5	6D	0	1	1	0	1	1	0	1	5	92	1	0	0		1	0	0	1	0
6	7D	0	1	1	1	1	1	0	1	6	82	1	0	0		0	0	0	1	0
7	07	0	0	0	0	0	1	1	1	7	F8	1	1	1		1	1	0	0	0
8	7F	0	1	1	1	1	1	1	1	8	80	1	0	0		0	0	0	0	0
9	6F	0	1	1	0	1	1	1	1	9	90	1	0	0		1	0	0	0	0

هیچکدام مزیتی بر دیگری ندارد فقط بستگی به نوع طراحی مدار شما دارد که می‌خواهید پایه وسط را منفی بدهید و بقیه را مثبت بدهید یا برعکس، اما بطور معمول کاتد مشترک پراستفاده تر باشد اما لزوماً مزیت محسوب نمی‌شود.

نحوه‌ی اتصال به میکروکنترلر:

بسته به این که جدول درستی چگونه تنظیم شده باشد نحوه‌ی اتصال به میکروکنترلر هم متفاوت خواهد بود. اگر کدهای هگز جدول درستی برای ترتیب A B C D E F G H پایه‌های سون‌سگمنت نوشته شده باشند، پایه‌ی A به PIN.7 پورت میکرو و به همین ترتیب پایه‌ی H به PIN.0 پورت میکرو متصل می‌شود. ولی اگر جدول درستی برای ترتیب H G F E D C B A نوشته شده باشد باید PIN.0 پورت به پایه‌ی A متصل گردد و به همین ترتیب PIN.7 به پایه ی H وصل شود.

۲ (راه اندازی با استفاده از آی سی‌های راه انداز:

استفاده از آی‌سی‌های راه‌انداز مبدل BCD به سون سگمنت کار را ساده‌تر می‌کند؛ تعداد خطوط برنامه نویسی را کاهش می دهد؛ تعداد پایه‌های مورد نیاز در میکرو را کاهش می‌دهد و نیاز به کد کردن به صورت دستی را رفع می نماید.

کد BCD چیست؟

در نمایش اعداد دهدهی برای هر رقم یک کد بایری ۴ بیتی در نظر می‌گیریم که به صورت جدول زیر است:

DECIMAL	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

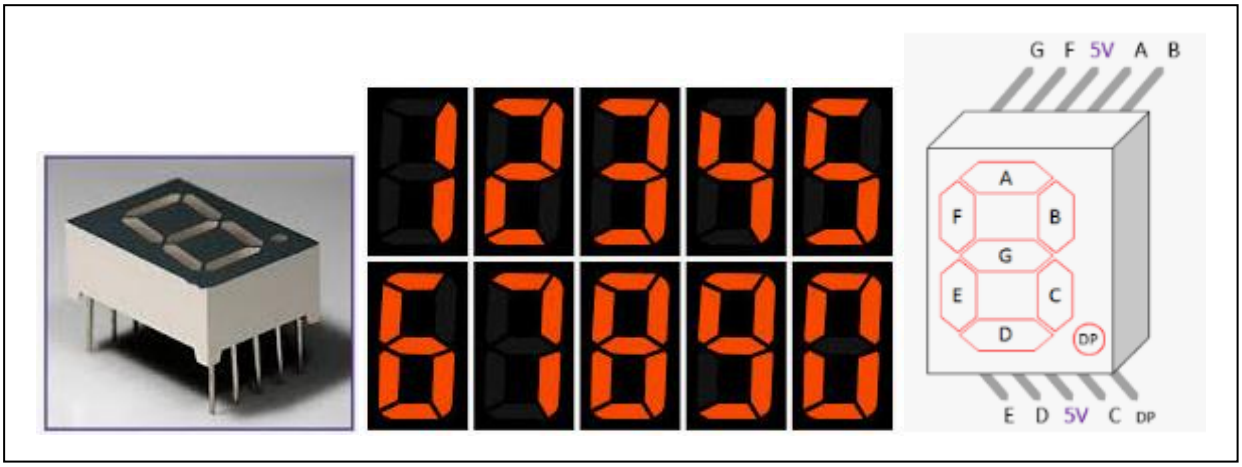
مزیت bcd آن است که همان عدد مبنای ده خودمان است که با یک و صفر بیان شده است (از یک طرف مبنای ده است از یک طرف باینری) مثلاً عدد ۵۹۶۴ به bcd می شود: ۰۱۰۱-۱۰۰۱-۰۱۱۰-۰۱۰۰ در نتیجه در ورودی کلیدها و یا خروجی صفحه نمایش مدارات میکروپروسسوری و جایی که می‌خواهیم با کاربر انسانی تعامل داشته باشیم از BCD استفاده می کنیم.

مبدل‌های BCD جهت استفاده آی‌سی‌های 7447 (برای سون سگمنت آند مشترک) و 7448 (برای سون سگمنت کاتد مشترک) می باشند، که عدد 0 تا 9 را به صورت کد BCD دریافت نموده و پس از کد کردن آن‌ها هفت بیت مورد نظر را به سون سگمنت می دهند تا عدد دریافت شده را نمایش دهد.

مالتی پلکس کردن سون سگمنت ها:

گاهی نیاز داریم که چند سون سگمنت را توسط یک پورت مقدار دهی کنیم. به عبارت دیگر با یک پورت یک عدد چند رقمی را با سون سگمنت هایی نمایش دهیم. به این کار مالتی پلکس کردن سون سگمنت ها گفته می شود.

در ادامه برنامه‌هایی که از آی‌سی‌های 7447 و 7448 و همچنین سون‌سگمنت مالتی پلکس استفاده می‌شود را توضیح خواهیم داد.



پس از آشنایی با سون سگمنت و مشخص کردن آند مشترک و کاتد مشترک بودن آنها می‌توانیم شروع به نوشتن برنامه جهت نمایش اعداد و یا حروف روی این قطعه کنیم.

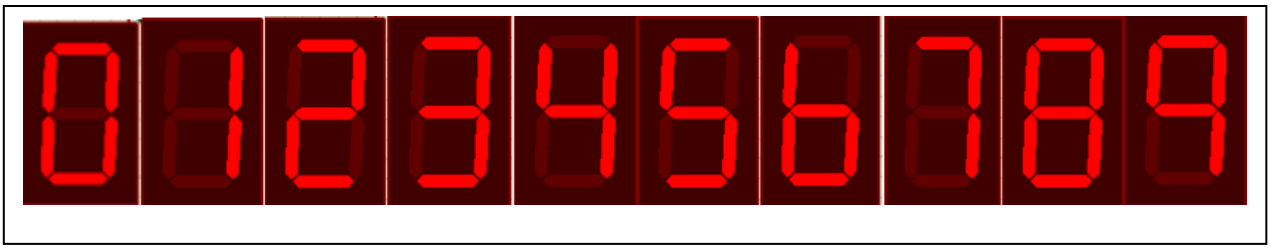
همانطور که قبلاً بیان شد ابتدا نوع میکرو و فرکانس کاری و همچنین پورت خروجی را تعیین می‌کنیم، سپس در مرحله بعدی چون قراره روی سون سگمنت اعداد بین صفر تا نه(9 - 0) را ببینیم و در هر بار این اعداد تغییر خواهند کرد باید یک متغیر تعیین کنیم.

با استفاده از دستور FOR - NEXT می‌توانیم حلقه‌ای بنویسیم تا اعداد (9 – 0) را بشمارد.

همچنین با استفاده از دستور LOOKUP می‌توان مقدار متغیر مشخص شده را از جدولی که آدرس دهی می‌شود فراخوانی کرد.

با توجه به توضیحات بالا اکنون می‌توانیم برنامه‌ای بنویسیم که اعداد(9 – 0) را بصورت صعودی بر روی سون سگمنت آند مشترک بشمارد.

```
$regfile "m32def.dat"
$crystal = 1000000
```




```
Config Portd = Output
Dim A As Byte
Do
For A = 0 To 9
Portd = Lookup(a , Man)
Waitms 200
Next A
Loop
End
Man:
Data &HC0 , &HF9 , &HA4 , &HB0 , &H99 , &H92 , &H83 , &HF8 , &H80 , &H98
```

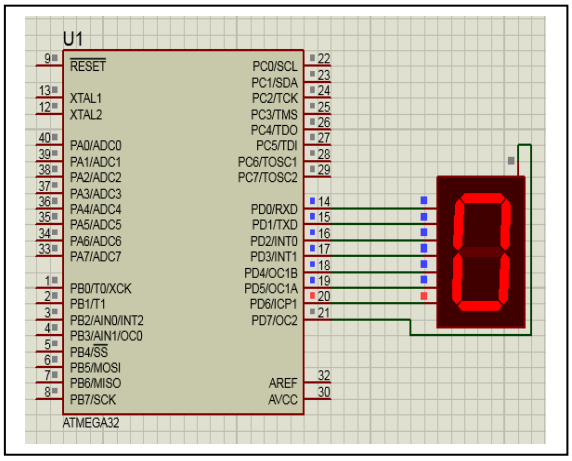
تعیین متغیر از جنس byte تا بازه عددی بین 0 تا 255 را شامل شود

شمارش اعداد در بازه 0 تا 9 با استفاده از حلقه for - next

فراخوانی مقدار متغیر(A) از جدول MAN با استفاده از دستور LOOKUP

برگشت به حلقه FOR و باز خوانی مقدار متغیر از جدول MAN

معادل هگز اعداد 0 تا9



نوشتن برنامه به جهت استفاده از سون سگمنت کاتد مشترک، کافیست معادل هگز اعداد در کاتد مشترک را بجای هگز اعداد در آند مشترک وارد کنیم و در شبیه سازی نیز بجای سون سگمنت آند مشترک، سون سگمنت کاتد مشترک را انتخاب کنیم تا اعداد 0 تا 9 را به صورت صعودی بشمارد.

حال اگر قرار باشه برنامه‌ای بنویسید تا اعداد 0 تا 9 را بصورت نزولی بشمارد، می‌توانید پس از نوشتن حلقه FOR یک STEPC -1 وارد کنید و For A = 0 To 9 را بصورت معکوس یعنی از For A = 9 To 0 انجام دهید.

یا می‌توانید در جدول تعریف شده MAN معادل هگز اعداد را بصورت برعکس از 9 تا 0 وارد کنید.

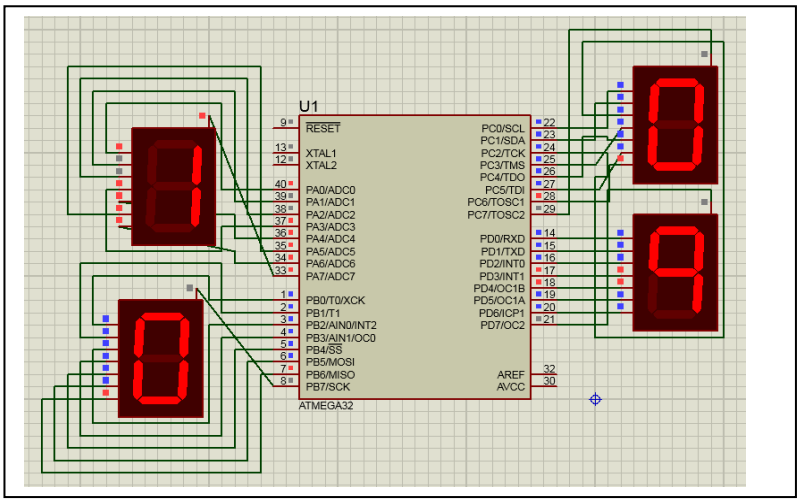
اکنون پس از آشنایی با سون سگمنت و نوشتن برنامه‌ای که بتواند اعداد را بصورت صعودی و یا نزولی بشمارد می‌تواند برنامه‌ای بنویسید که بتواند بر روی 4 عدد سون سگمنت اعداد 0 تا 9999 را ببینید.

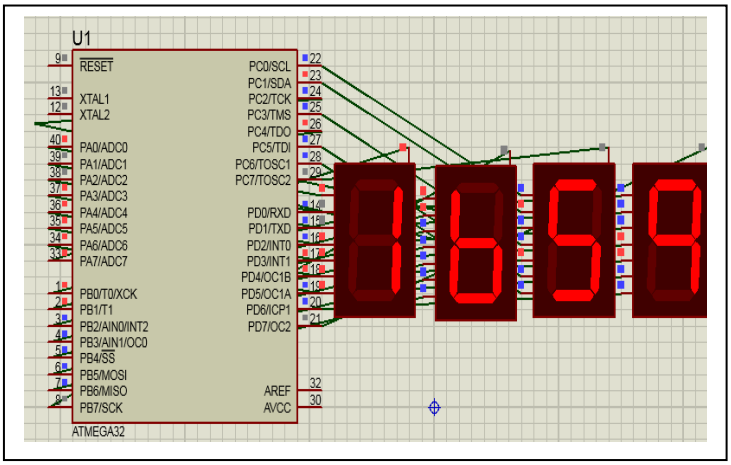
کافی‌ست همان برنامه بالا را 4 بار تکرار کنید، البته با نوشتن این برنامه هر 4 پورت میکروکنترلر مورد استفاده‌ی شما درگیر و مشغول خواهد شد، زیرا هر سون سگمنت به یک پورت اختصاص داده خواهد شد.

ابتدا 4 پورت خروجی تعریف می‌کنیم، در مرحله‌ی بعدی 4 متغیر جهت یکان، دهگان، صدگان و هزارگان از جنس BYTE تعیین می‌کنیم. سپس با استفاده از حلقه FOR - NEXT بازه‌ی عددی هر یک از سون سگمنت‌ها جهت نمایش اعداد را مشخص می‌کنیم، تا با استفاده از دستور LOOKUP مقدار متغیر را از جدول تعریف شده فراخوانی کند.

```
$regfile "m32def.dat"
$crystal = 1000000
Config Porta = Output
Config Portb = Output
Config Portc = Output
Config Portd = Output
Dim A As Byte
Dim B As Byte
Dim C As Byte
```

```
Dim D As Byte
Do
For A = 0 To 9
Porta = Lookup(a , Man)
Waitms 100

For B = 0 To 9
Portb = Lookup(b , Man)
Waitms 100

For C = 0 To 9
Portc = Lookup(c , Man)
Waitms 100

For D = 0 To 9
Portd = Lookup(d , Man)
Waitms 100
Next A
Next B
Next C
Next D
Loop
End
Man:
Data &HC0 , &HF9 , &HA4 , &HB0 , &H99 , &H92 , &H83 , &HF8 , &H80 , &H98
```

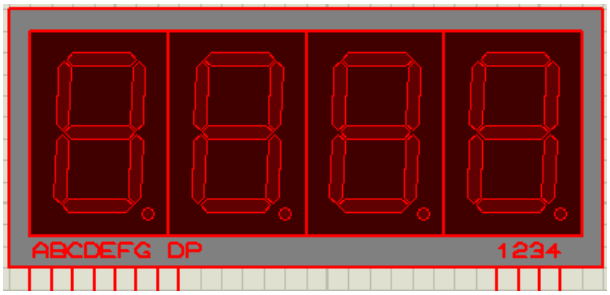




بدیهی است در برنامه فوق شما قادر خواهید بود که فقط شمارش اعداد بین 0 تا 9 را بر روی هر سون سگمنت مشاهده کنید و عمل شمارش یا Counter برروی هر سون سگمنت در محدودی 0 تا 9 انجام خواهد گرفت و در واقع اعداد 0 تا 9999 فقط رویت می‌شود و عمل شمارش یا Counter بر روی آن انجام نمی‌شود.

در برنامه بالا مشاهده شد، با استفاده از ۴ عدد سون سگمنت اعداد بین0 تا9999 رویت شد.

حتما دریافتید که برای انجام این برنامه نیاز به تعریف ۴ پورت خروجی از میکروکنترلر خواهد بود و هر ۴ پورت میکروکنترلر اشغال خواهد شد و نیز تعداد سیم‌هایی که از میکرو به سون سگمنت‌ها اتصال داده شده‌اند بسیار زیاد است و در محیط فیزیکی بر روی برد واقعی انجام این نوع سیم‌کشی تا حدود زیادی برای کاربر سخت خواهد بود، اگر چه در محیط شبیه سازی نیز تاحدودی این مشکل وجوددارد. برای رفع این مشکل از سون سگمنت مالتی پلکس استفاده می‌کنیم.

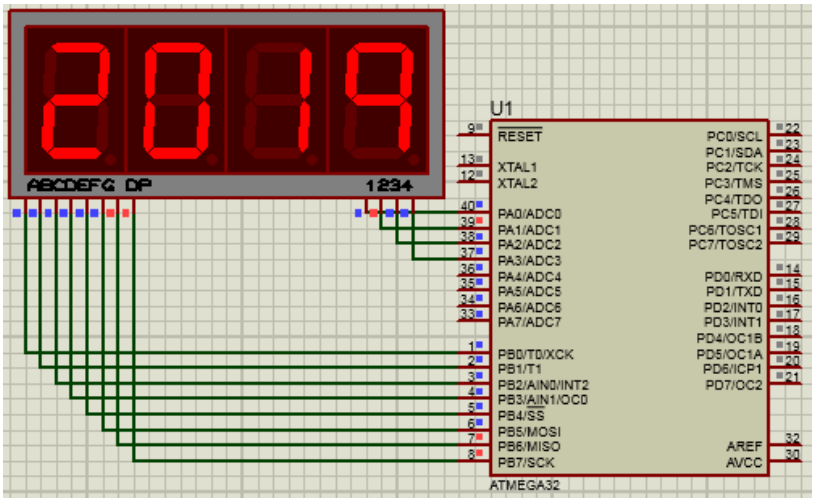


با توجه به شکل بالا یک سون سگمنت مالتی پلکس ۴تایی از ۴ سون سگمنت تشکیل شده است که در یک پک عرضه شده است.پایه های A,B,C,D,E,F,G,DP همان پایه‌های مشترک بین ۴ سون سگمنت می‌باشند.

جهت نمایش اعداد یا حروف بر روی این نوع سون سگمنت‌ها در برنامه نویسی، کافی‌ست ابتدا پایه مربوط به همان سون سگمنت را فعال کرده، سپس معادل هگز عدد را وارد کنیم.

```
$regfile "m32def.dat"
$crystal = 1000000
Config Porta = Output
Config Portb = Output
Do
Porta = &B00000001          فعال کردن سون سگمنت اول
Portb = &HA4                نمایش عدد 2 در portb
Waitms 1
                            فعال کردن سون سگمنت دوم
```

```
Porta = &B00000010
Portb = &HC0                    نمایش عدد 0 در portb
Waitms 1
Porta = &B00000100              فعال کردن سون سگمنت سوم
Portb = &HF9                    نمایش عدد 1 در portb
Waitms 1
Porta = &B00001000              فعال کردن سون سگمنت چهارم
Portb = &H98                    نمایش عدد 2 در portb
Waitms 1
Loop
End
```

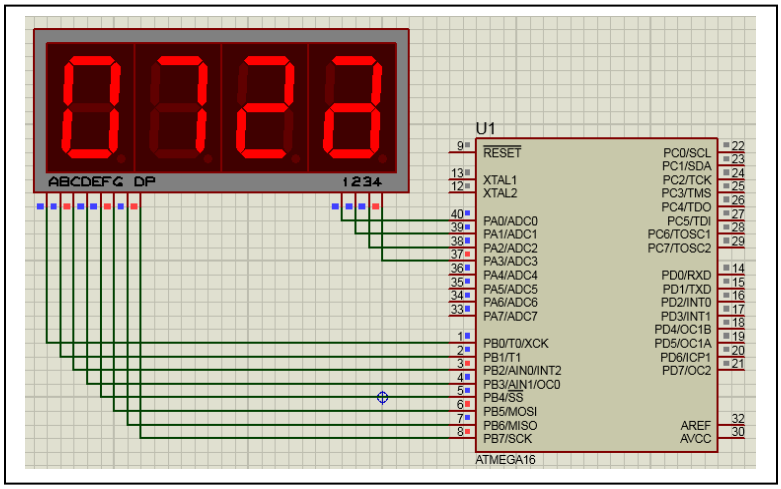


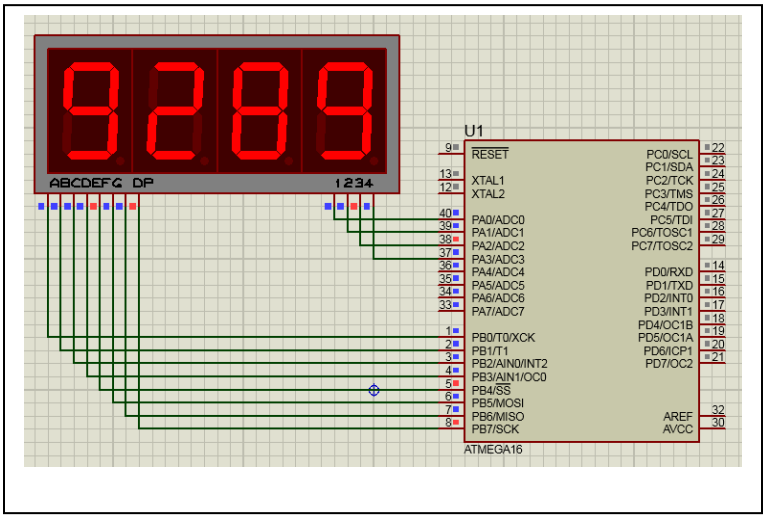
حال اگر قرار باشد روی همین 7seg مالتی پلکس اعداد شمارش 0 تا 9999 صورت پذیرد با مروری به برنامه گذشته به راحتی می‌توانیم برنامه مرود نظر را بنویسیم.

ابتدا 4 متغییر تعیین کنیم، سپس همانند برنامه قبلی باید اول پایه‌ی مربوط به همان 7seg را فعال کرده و در مرحله بعد دستور نمایش عدد خوانده شده از جدول معرفی شده را اعمال می‌کنیم.

```
$regfile "m16def.dat"
```

```
$crystal = 1000000
Config Porta = Output
Config Portb = Output
Dim A As Byte
Dim B As Byte
Dim C As Byte
Dim D As Byte
Do
For A = 0 To 9
For B = 0 To 9
For C = 0 To 9
For D = 0 To 9
Porta = &B00000001
Portb = Lookup(a , Dfpm)
Waitms 7
Porta = &B00000010
Portb = Lookup(b , Dfpm)
Waitms 7
Porta = &B00000100
Portb = Lookup(c , Dfpm)
Waitms 7
Porta = &B00001000
Portb = Lookup(d , Dfpm)
Waitms 7
Next A
Next B
Next C
Next D
Loop
End
```





Dfpm:

```
Data &HC0 , &HF9 , &HA4 , &HB0 , &H99 , &H92 , &H82 , &HF8 , &H80 , &H90
```

اگه قرار باشه یه برنامه‌ای بنویسیم که عمل cunter یا همان شمارش اعداد را در بازه 9999-0 را با استفاده از دو کلید انجام بده باید برنامه زیر را بنویسیم.

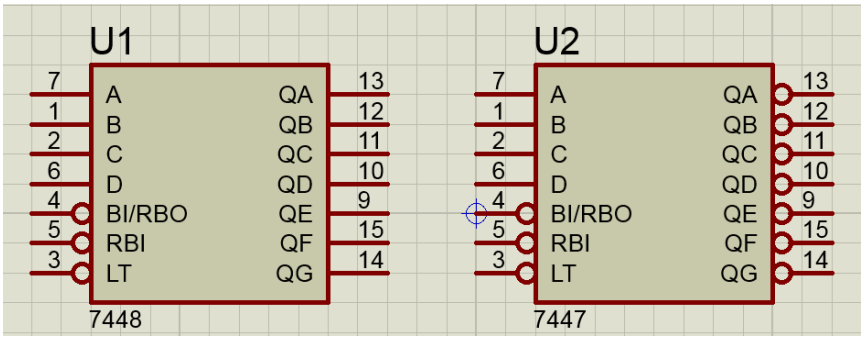
```
$regfile = "m8def.dat"
$crystal = 4000000
Config Portd = Output
Config Portc = Output
Config Pinb.0 = Input
Config Pinb.2 = Input
Dataport Alias Portd
Comm Alias Portc
Dim A As Byte , B As Byte , C As Word , D As Word , _
  , N As Byte , M As Byte , V As Byte , T As Byte , Hh As Byte
Declare Sub Q
Declare Sub W
Declare Sub E
Declare Sub K
Declare Sub X
Do
Call E
Call Q
Call K
Loop
Sub K
If Pinb.0 = 1 Then
Incr N
Call X
End If
```

```
If Pinb.2 = 1 Then
N = 0
M = 0
T = 0
V = 0
Call X
End If
End Sub
Sub E
If N > 9 Then
N = 0
Incr M
If M > 9 Then
M = 0
Incr V
If V > 9 Then
V = 0
Incr T
If T > 9 Then
T = 0
End Sub
End If
End If
End If
End If
Sub Q
Comm = &B0111 : A = N : Call W
Comm = &B1011 : A = M : Call W
Comm = &B1101 : A = V : Call W
Comm = &B1110 : A = T : Call W
```

```
End Sub
Sub W
Dataport = Lookup(a , Sa)
Waitms 4
Dataport = &H00
End Sub
Sub X
  For Hh = 1 To 2
  Call E
    Call Q
  Next
End Sub
Sa:
Data &B00111111 , &B00000110 , &B01011011 , &B01001111
Data &B01100110 , &B01101101 , &B01111101 , &B00000111
Data &B01111111 , &B01101111
```

بعد از نوشتن برنامه‌هایی در مورد سون سگمنت حالا می‌خواهیم برنامه‌ای بنویسیم که بدون نیاز به معادل هگز اعداد و نوشتن عدد مورد نظر بتوانیم روی سون سگمنت اعداد را مشاهده کنیم.

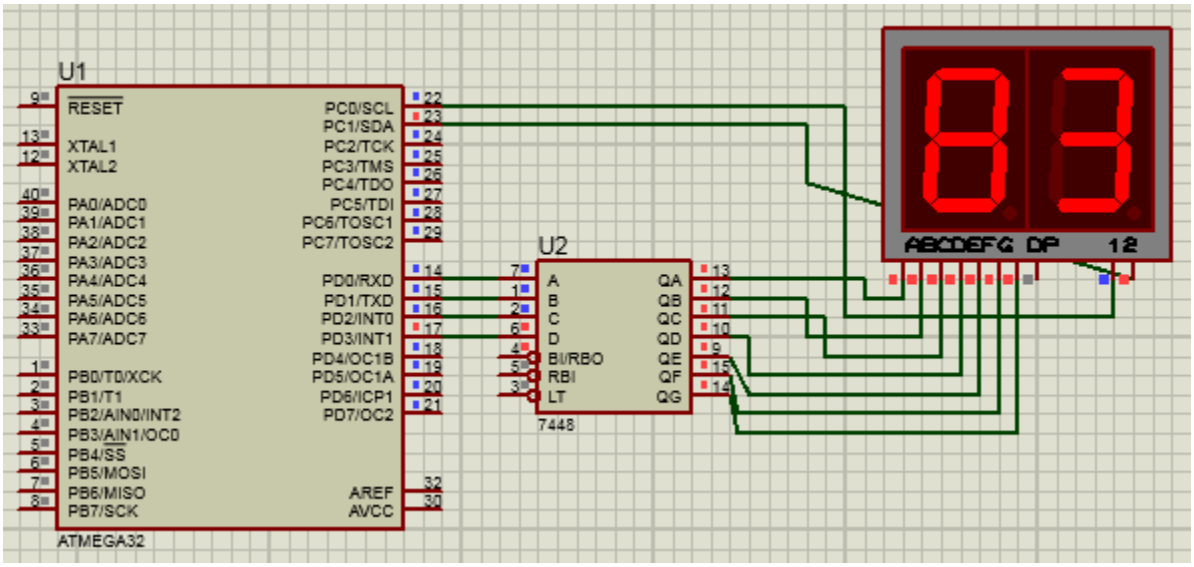
برای اینکار باید از آی سی های 7448 , 7447 استفاده کنیم. آی سی 7448 جهت استفاده سون سگمنت‌های کاتد مشترک و آی سی 7447 جهت استفاده سون سگمنت‌های آند مشترک مورد استفاده قرار می‌گیرد.



پایه‌هایQA , QB ,QC , QD, QE , QF, QG را به سون سگمنت و پایه‌هایA , B , C , D به میکرو کنترلر متصل می‌گردند.

```
$regfile = "m32def.dat"
```

```
$crystal = 1000000
Config Portc = Output
Config Portd = Output
Do
Portc = &B00000010          فعال کردن سون سگمنت اول
Portd = 8                   نمایش عدد 8 بر روی portd
Waitms 1
Portc = &B00000001          فعال کردن سون سگمنت دوم
Portd = 3                   نمایش عدد 3 بر روی portd
Waitms 1
Loop
```



پس از آشنائی با میکرو و نحوه‌ی برنامه نویسی و شبیه سازی بوسیله میکرو کنترلرAVR نوبت به معرفی LCD می‌رسد تا بتوانید بر روی LCD ، کاراکتر و یا المان‌های گرافیکی را نمایش دهید.

LCDها به دو نوع اصلی کاراکتری و گرافیکی تقسیم می‌شوند. همانطوریکه از نام هردو پیداست بر روی LCD کاراکتری می‌توانیم فقط کاراکترها را نمایش دهیم که شامل حروف، اعداد، علامت‌ها و هرچیزی که تحت عنوان کارکتر معرفی می‌شوند.

ابتدا پایه‌هایLCD کاراکتری را معرفی می‌کنیم.

شماره پایه	عملکرد پایه	شماره پایه	عملکرد پایه
1	GND	9	D2 یا DB2
2	VCC	10	D3 یا DB3
3	CON	11	D4 یا DB4
4	RS	12	D5 یا DB5
5	WR	13	D6 یا DB6
6	E	14	D7 یا DB7
7	D0 یا DB0	15	ANOD
8	D1 یا DB1	16	KATOD

GND : پایه اتصال به زمین

VCC : پایه اتصال تغذیه +5V

CON : تنظیم شفافیت نور صفحه

RS : رجیستر

WR : پایه نوشتاری

E : توانساز – Enabel فعال کننده

از پایه D0 تا D7 جهت انتقال دیتا استفاده می‌شود.

بعد از شناختن پایه های LCD حالا وقتشه که به میکرو بفهمانیم خروجی ما دیگه LED یا 7SEG نیست، بلکه LCD است. برای اینکار باید طبق روال قبل از دستورCONFIG که همان دستور مشخص کننده ورودی و خروجی است استفاده کنیم.

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 ,
Rs = Portc.2
```

همانطوریکه پیداست با نوشتن دستور بالا LCD و تک تک پایه‌های LCD به یکی از PORT های خروجی میکرو نسبت داده شده است. نوشتن دستور بالا که در واقع همان مشخص کردن PORT خروجی است، پیکربندی LCD کاراکتری نام دارد. در اینجا با نوشتن LCDPIN مشخص می‌کنیم که LCD ما از جنس کاراکتری است، سپس PIN ها را بر اساس بیت‌های پر ارزش و یا کم ارزش به خروجی نسبت می‌دهیم.

بیت‌های پر ارزش				بیت‌های کم ارزش			
D7	D6	D5	D4	D3	D2	D1	D0

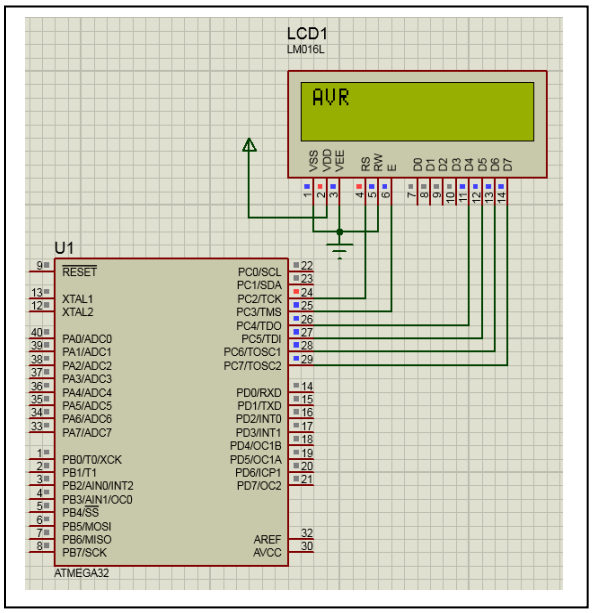
البته اگه قرار باشه از بیت‌های کم‌ارزش استفاده بشه باید همه‌ی پین‌های D0 تا D7 در پیکربندیLCD تعریف شوند، در غیر اینصورت تعریف D4 تاD7 کافیست.(مثل کد بالا)

در ادامه هریک از پایه‌ها به یک pin نسبت داده می‌شوند(Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7) و همچنین پایه E به Portc.3 و RS نیز به portc.2 نسبت داده می‌شود.

بعد از پیکربندی lcd کاراکتری کافیست با نوشتن دستور " lcd " بین " " هر کاراکتری که مورد نظر است را نوشته و در نهایت بر روی همان port که به عنوان خروجی معرفی شده بود بر روی lcd نمایش داده خواهد شد.

پیکر بندی LCD کاراکتری

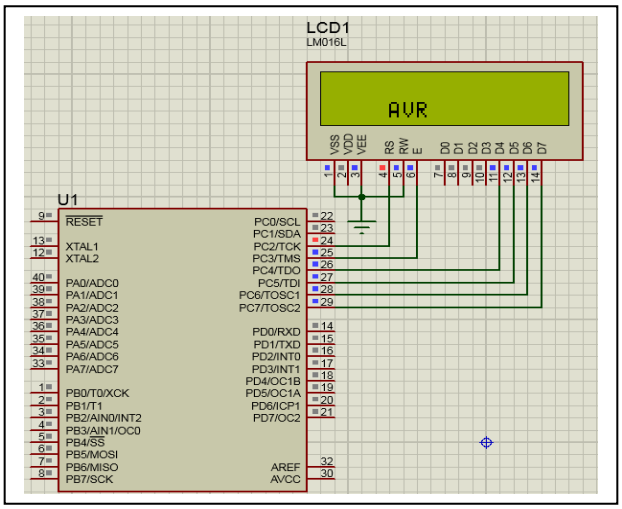
```
$regfile = "m32def.dat"
$crystal = 1000000
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
Cursor Off          خاموش بودن مکان نما
Do
Lcd "AVR "          نمایش کلمه AVR
Waitms 200
Cls                 پاک کردن مقدار قبلی
Loop
End
```



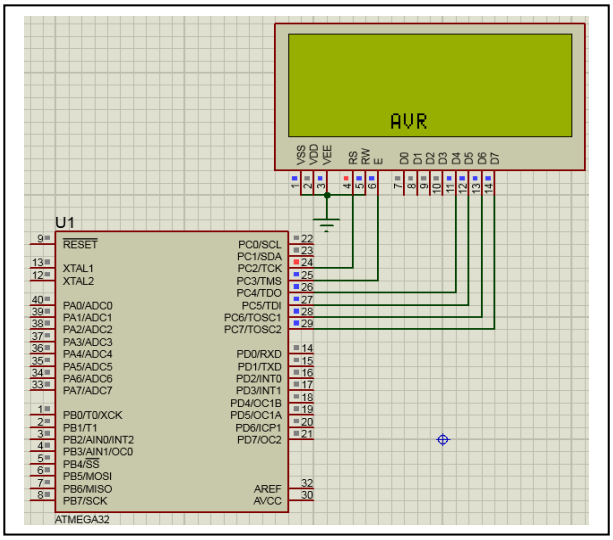
همانطوریکه مشاهده شد پس از پیکربندیLCD کاراکتری و نوشتن کلمه AVR در بین " " بصورت پیش فرض در سطر اول و ستون اول نمایش داده می‌شود. با دستور Cursor Off نیز می‌توان مکان نما را خاموش نمود. دستور CLS نیز کلمه یا کاراکتر قبلی نمایش داده شده در LCD را پاک می‌کند تا در اجرای دستور حلقه LOOP DO مقدار یا همان کلمه یا کاراکتر جدید نمایش داده شود در غیر اینصورت بدلیل تکرار برنامه، کلمه AVR دائماً نمایش داده می‌شود.

در پیکربندی lcd کاراکتری اگر اندازه lcd مشخص نشود باز هم بصورت پیش فرض اندازه lcd ، 16*2 در نظر گرفته می‌شود. جهت استفاده از lcd های دیگر باید با استفاده از دستور Config Lcd = 20 * 4 اندازه lcd را مشخص نمود.

جهت نمایش کاراکتر در سطر و یا ستون های دلخواه باید از دستور Locate استفاده نمود Locate 2, 5 که مشخصاً سطر دوم و ستون پنجم محل شروع نمایش کاراکتر خواهد. در غیر اینصورت بصورت پیش فرض سطر اول و ستون اول شروع نمایش کاراکتر خواهد بود.



به همین ترتیب جهت نمایش کاراکتر مورد نظر در ستون‌های سوم و چهارم در lcd های 4*20 می‌توان با استفاده ازدستور locate سطر و ستون را مشخص نمود. Locate 4, 5



جهت مشخص نمودن سطر از دستورات زیر نیز می‌توان استفاده کرد. Home f , Home t , Home l , Home u و همچنین از دستورات theti line (خط سوم) forth line (خط چهارم) lower lin (خط پایین) uper line (خط بالا) می‌توان استفاده کرد.

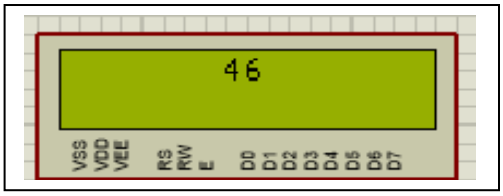
اکنون می‌خواهیم برنامه‌ای بنویسیم که بر روی lcd کاراکتری عدد 45 را نمایش داده پس از گذشت ms200 یک واحد به آن اضافه کند یعنی 46 و همینطور پس از گذشت ms200 دوباره به مقدار قبلی یعنی 45 برگردد.

جهت نوشتن این برنامه از دستور INCR , DECR استفاده می‌کنیم. این دستورات به ترتیب یک واحد افزایش (INCR) و یک واحد کاهش(DECR) در مقدارعددی تعریف شده خواهد داشت.

```
$regfile = "m32def.dat"
$crystal = 1000000
```

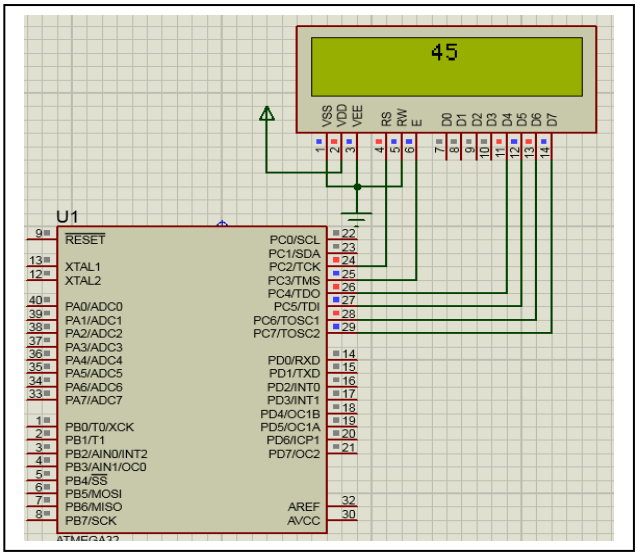
```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
Cursor Off
Dim M As Byte
Do
M = 45
Locate 1 , 8
Lcd M
Waitms 500
Cls
Incr M
Locate 1 , 8
Lcd M
Waitms 500
Cls
Decr M
Locate 1 , 8
Lcd M
Waitms 500
Cls
Loop
End
```

در این برنامه ابتدا عدد45 معرفی و سپس در LCD نمایش داده می‌شود و بعد از 500MS یک واحد به آن اضافه شده و عدد 46 به مدت 500MS نمایش داده می شود و این عمل به دلیل قرار گرفتن در حلقه LOOP DO تا بی‌نهایت تکرار می‌شود.



افزایش یک واحد به مقدار عددی متغییر M

کاهش یک واحد از مقدار عددی متغییر M

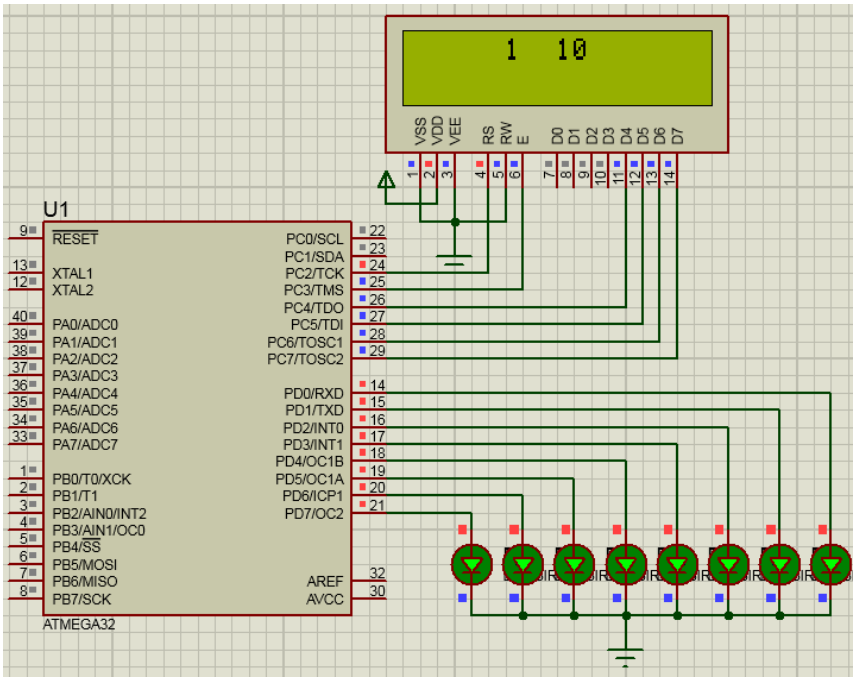


پس از نوشتن این برنامه و آشنا شدن با دستور Incr و Decr می‌توانیم برنامه‌ای بنویسیم که اعداد بین 0 تا 10 را بشمارد، سپس8 عددLED بصورت چشمک‌زن به‌مدت200ms روشن و خاموش شوند.

جهت نوشتن این برنامه کافی‌ست از دستور Loop Until استفاده کنیم. بدین صورت که ابتدا جهت شمارش اعداد متغییری را تعیین کرده و سپس با استفاده از دستور incr می‌توان عمل افزایش را انجام داد، در ادامه جهت توقف متغییر در عدد مورد نظر (10) از دستور Loop Until M = 10 استفاده می‌کنیم. بدین ترتیب متغییر M در عدد 10 توقف می‌نماید. سپس با استفاده از دستور Toggle پورت خروجی مورد نظر را روشن و خاموش می‌کنیم.

```
$regfile = "m32def.dat"
```

```
$crystal = 1000000
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
Config Portd = Output
Cursor Off
Dim M As Byte
Do
Incr M              افزایش یک واحد به مقدار عددی متغییر M
Locate 1 , 8
Lcd M               نمایش اعداد بین 0 تا 10
Waitms 200
Loop Until M = 10   توقف تکرار حلقه شمارش اعداد بین 0 تا 10
Cls
Do
Toggle Portd        خاموش و روشن( 0 و 1) شدن مقدار portd
Waitms 200
Loop
End
```



البته برنامه فوق را می‌توان به صورت دیگری هم نوشت.

در این برنامه وارد مبحث جدید و مهم به نام پرش‌های شرطی می‌شویم.

بدین معنا که عملیات برنامه، قبل از ورود به حلقه (while wend) که باید شروع به تکرار می‌کرد،(شرط ورود به حلقه) را تست می‌کند و در صورت حقیقی(True) بودن شرط برنامه را از داخل حلقه آغاز می‌کند. در غیر اینصورت، یعنی اگر شرط ذکر شده باطل و غلط باشد(Flse) باشد برنامه داخل حلقه نشده و از سطر بعدی ادامه پیدا می‌کند.

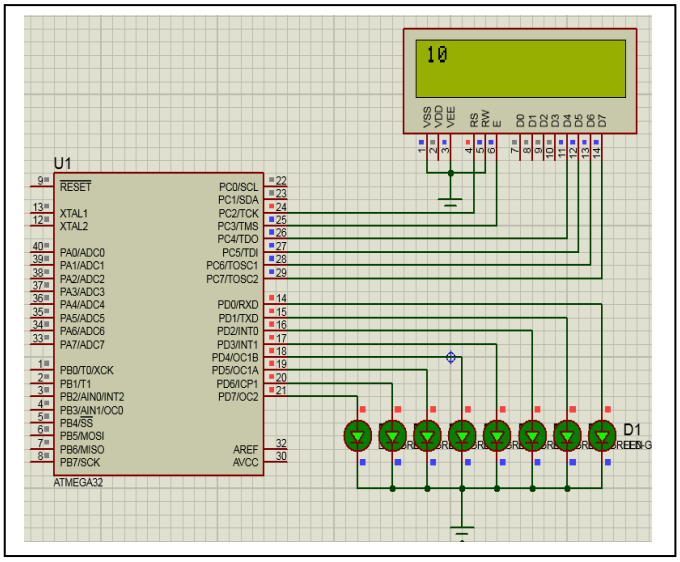
بطور کلی ممکن است در حالت‌هایی(به علت درست نبودن شرط) این دستور کاملاً اجرا نشود. در این برنامه ابتدا از مقدار 1 تا 10 را با فاصله زمانی 100ms شمرده و زمانی که به مقدار 10 رسید portd را که به led ها وصل و خاموش هستند، tToggle می‌کند. یعنی وقتی به این دستور رسید

```
While H = 10
Toggle Portd
Waitms 40
```

از روی آنها رد شده و تا زمانی ادامه می‌یابد که متغییر ما برابر 10 باشد. با توجه به اینکه شرط ورود به حلقه While H = 10 این است که مقدار A=10 گردد، و لذا وقتی Aمساوی مقدار عددی 10 شد، وارد حلقه شده و دستور حلقه را اجرا می‌کند که همان روشن شدن LED ها می‌باشد و بعد از 40MS تاخیر، بازهم دستور داخل حلقه اجرا کرده و این عمل را متناوباً تکرار خواهد کرد.

```
$regfile = "m32def.dat"
$crystal = 1000000
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
Config Portd = Output
```

```
Cursor Off
Dim H As Byte
Do
Incr H        افزایش یک واحد به مقدار عددی متغییر H
Cls
Lcd H
While H = 10  تا زمانی‌که متغییر Hبرابر عدد 10 شد
Toggle Portd  خاموش و روشن(0 و1) شدن portd
Waitms 40
Wend          پایان شرط
Waitms 100
Loop
End
```



با تسلط بر روی دستورات قبلی می‌توان برنامه‌ای نوشت که بر روی LCD کاراکتری ساعت با نمایشگر ثانیه، دقیقه و ساعت را نشان دهد.

به زبان ساده می‌توان بیان کردکه هر ساعت دارای سه متغییر عددی، ثانیه، دقیقه و ساعت می‌باشد که هرکدام از این سه متغییر خود دارای بازه‌های عددی مختلفی می‌باشند.

ثانیه : 0 تا 59

دقیقه : 0 تا 59

ساعت : 0 تا 23

واضح است که جهت نمایش اعداد مربوط به هر یک از بازه‌های ثانیه، دقیقه و ساعت باید یک متغییر تعیین و به آن نسبت داد تا بتواند عمل شمارش را انجام دهد. پس Dim C As Byte و Dim B As Byte و Dim A As Byte را تعریف می‌کنیم.

قبل از تعیین متغییر باید بدانید که هر متغیر تا چه میزان مقدار عددی متغییر می‌کند. همانطوریکه می‌دانید مقدار متغیر ثانیه 59 با تغیر می‌کند، همین‌طور مقدار متغیر دقیقه نیز 59 بار تغییر می‌کند و همچنین مقدار متغیر ساعت 23 بار تغییر خواهد داشت. پس می‌توانیم جنس متغیر را از نوع byte انتخاب کنیم چون در بازه 0 تا255 قرار دارند.

پس از معرفی متغیرها می‌توانید با استفاده از دستور (incr C) (یک واحد افزایش) یک واحد به مقدار متغیرها اضافه کنید. سپس باید از دستورif اسفاده کرده و یک حلقه شرطی ایجاد کنیم به این ترتیب که اگر مقدار متغیر ثانیه از 59 بزرگتر شد، آنگاه یک واحد به

مقدار متغیر دقیقه اضافه کن(incr B) و در ضمن مقدار متغیر ثانیه را برابر صفر قرار دهد. در ادامه اگر مقدار متغیر دقیقه نیز از 59 بزرگتر شد، آنگاه یک واحد به مقدار ساعت اضافه کن (incr A) و مقدار متغیر دقیقه را برابر صفر قرار دهد.

در مرحله‌ی بعد جهت نمایش متغیرهای (A , B , C) می‌توانید به شکل زیر عمل کنید:

```
Lcd A ; "" ; ":" ; "" ; B ; "" ; ":" ; "" ; C ; ""
```

در دستور بالا همانطور که قبلاً توضیح داده شده جهت نمایش مقدار متغیر بر روی lcd باید از دستور (نام متغیر)LCD استفاده کنیم. همچنین اگر قرار باشد بین(" ") عبارت و یا کاراکتری نوشته شود، عیناً همان عبارت یا کاراکتر نمایش داده می‌شود، لذا فضای خالی یا همان Spas نیز بعنوان یک کاراکتر محسوب شده و فضای خالی نمایش داده می.شود. همچنین ":" در lcd بصورت : نمایش داده خواهد شد.

نکته : می‌توان از دستور Locate جهت مشخص نمودن سطر وستون محل نمایش بر روی صفحه lcd استفاده کرد.

```
$regfile "m32def.dat"
$crystal = 1000000
Config Portc = Output
Config Lcdpin = Pin , Db4 = Portc.0 , Db5 = Portc.1 , Db6 = Portc.2 , Db7 = Portc.3 , E = Portc.5 , Rs = Portc.4
Dim A As Byte , B As Byte , C As Byte
A = 00
B = 00
C = 00
Cursor Off
Do
Locate 1 , 3
Incr C
Waitms 100
Lcd A ; "" ; ":" ; "" ; B ; "" ; ":" ; "" ; C ; ""
If C > 59 Then
C = 00
Incr B
Locate 1 , 3
Lcd A ; "" ; ":" ; "" ; B ; "" ; ":" ; "" ; C ; ""
If B = 59 Then
```

B = 00

C = 00

Incr A

Locate 1 , 3

Lcd A ; "" ; ":" ; "" ; B ; "" ; ":" ; "" ; C ; ""

If A = 23 Then

A = 00

B = 00

C = 00

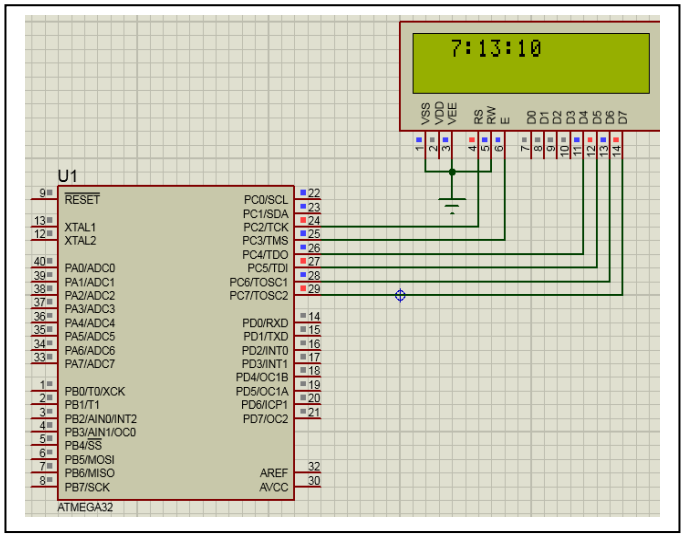
End If

End If

End If

Loop

End



جهت نمایش ساعت بر روی صفحه lcd می‌توان از حلقه دستورFor – Next نیز استفاده کرد. مانند برنامه بالا تعیین سه متغیر و معرفی آن ، سپس مشخص کردن بازه عددی هر متغیر که برای ثانیه و دقیقه 59 – 0 و برای ساعت 23 -0 است. در مرحله‌ی بعد جهت نمایش مقدار متغیرها بر روی lcd از دستور Lcd "" ; H ; ":" ; M ; ":" ; S ; ""استفاده می‌کنیم. پس از گذشت چند waitms کافی‌ست next ها را به ازای هر for برای هر متغیر بنویسیو . در ادامه جهت تکرار برنامه از حلقه do – loop استفاده کنیم.

$regfile "m32def.dat"

$crystal = 8000000

Config Lcdpin = Pin , Db4 = Portc.0 , Db5 = Portc.1 , Db6 = Portc.2 , Db7 = Portc.3 , E = Portc.5 , Rs = Portc.4

Config Lcd = 16 * 2

Dim S As Byte , M As Byte , H As Byte

Cursor Off

Cls

Do

Home

For H = 0 To 23

For M = 0 To 59

For S = 0 To 59

Cls

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

Waitms 1

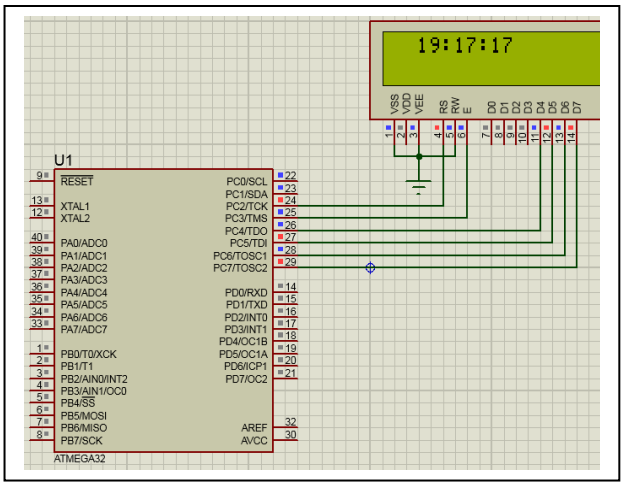
Next H

Next M

Next S

Loop

End



ساعت و تقویم شمسی

$regfile "m32def.dat"

$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , Rs = Portc.3 , E = Portc.2

Dim H As Byte

Dim M As Byte

Dim S As Byte

Dim Daey As Byte

Dim Month As Byte

Dim Years As Word

Do

Reza:

```
For Years = 1398 To 1500
For Month = 1 To 12
For Daey = 1 To 31
For H = 0 To 23
For M = 0 To 59
For S = 0 To 59

If Month > 6 And Month < 12 And Daey > 30 Then
Daey = 1 : Goto Reza
If Month = 12 And Daey > 29 Then
Daey = 1
Goto Reza:
Locate 1 , 8

Lcd H ; ":" ; M ; ":" ; S
Locate 2 , 8

Lcd Years ; " : " ; Month ; " : " ; Years
Cls
Next Years
Next Month
Next Daey
Next H
Next M
Next S
End If
End If
Loop
End
```

به همین ترتیب می‌توان برنامه تقویم را هم نوشت.

یعنی مشخص کردن سه متغیر جهت روز، ماه، سال که هر کدام بازه‌های عددی 0-30 و یا برای ۶ ماهه اول0-31 برای روز، 0-12 برای ماه و همینطور برای سال را می‌توانید مشخص کنید. در ادامه کافی‌ست از دستورات incr و یا حلقه for- next جهت شمارش استفاده کنید.

در این قسمت کتاب، یک پروژه ساده الکترونیکی را که همان تولید کننده صدای بوق می‌باشد.

برنامه ای بنویسید که برروی یکی از پایه‌های میکروکنترلر(portc.2) تولید صدای بوق ممتد را انجام دهد.

برای این‌کار با یک دستور جدید آشنا می‌شویم که شکلکلی آن به این قرار است:

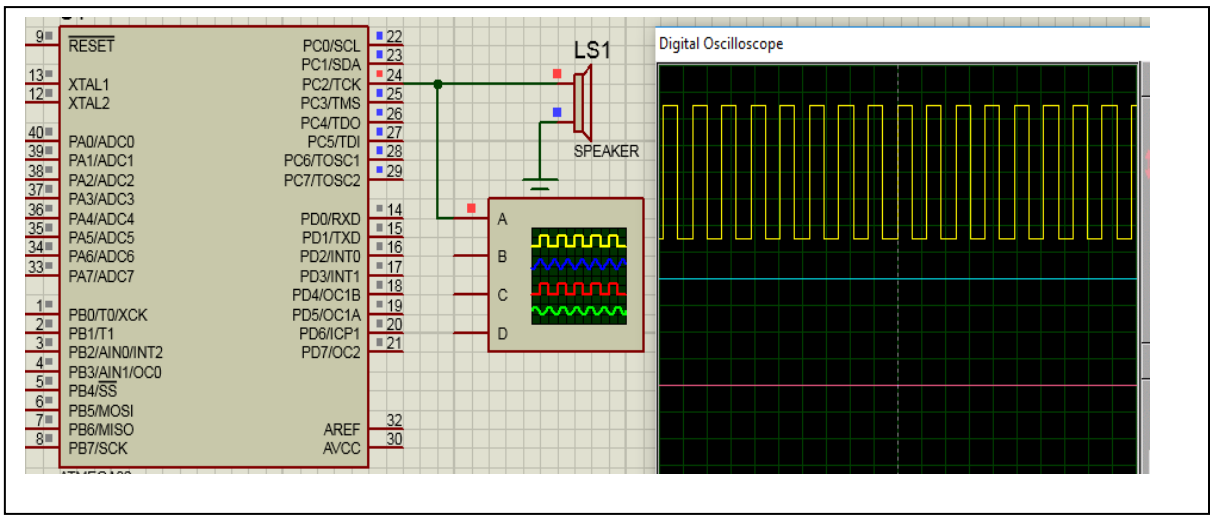
Sound pin Duration pulses

که در این دستور،Sond به همان معنای واقعی خود در لغت یعنی صدا می‌باشد، و منظور از Pin همان پایه‌ای است که در مدار مورد استفاده قرار گرفته است.

Duration که به معنای لغوی (مدت، دوام، زمان ، استمرار) می‌باشد، ثابت یا متغیری است که مشخص کننده تعداد پالس‌های ارسالی می‌باشد.

Pulses به معنای لغوی (پالس) ثابت یا متغیری است که مدت بالا و پائین بودن (دامنه) فرکانس موردنظر می‌باشد و هر دوی این متغیرها، می‌توانند اعدادی بین 1-65535 را داشته باشند.

بنابراین در دستور Sound Portc.2 , 1000 , 100 پالس‌هایی با فرکانس ۱۰۰۰ و دامنه ۱۰۰ تولید خواهد شد.



در شبیه سازی می‌توانید از اسیلوسکوپ استفاده کنید و شکل موج پالسی تولید شده را مشاهده کنید. همچنین می‌توانید فرکانس کاری را عوض کرده و تغییرات صداهای جدید تولید شده را تجربه کنید.

در این قسمت می‌خواهیم با نوع دیگر lcd یعنی همان lcd گرافیکی آشنا شویم. Lcd گرافیکی نیز در اندازه‌های مختلفی در بازار عرضه شده اند. Lcd گرافیکی نیز شبیه lcd کاراکتری نیاز به پیکر بندی دارد،اما با کمی تفاوت در نحوه‌ی پیکربندی این امر محقق می‌شود. در ادامه به نحوه پیکربندی lcd گرافیکی می‌پردازیم.

جهت پیکربندی lcd گرافیکی باید پایه‌های lcd را معرفی کنیم.

نام پایه از میکرو که باید وصل شوند	نام پایه در lcd	شماره پایه
GND	GND	1
GND	GND	2
+5V	+5V	3
-9V به وسیله پتانسیومتر	-9V	4
PORTB.2	WR	5
PORTB.3	RD	6
PORTB.0	C/E	7
PORTB.1	C/D	8
NC	NC	9
PORTB.4	RESET	10
PORTC.0	D0	11
PORTC.1	D1	12
PORTC.2	D2	13
PORTC.3	D3	14
PORTC.4	D4	15
PORTC.5	D5	16
PORTC.6	D6	17
PORTC.7	D7	18
PORTB.5	FS	19
NC	NC	20

پایه‌های ۱و۲جهت اتصال به زمین یا همان GND در نظر گرفته شده است.

پایه شماره ۳ جهت اتصال به تغذیه 5V+ بکار می‌رود.

پایه شماره ۴ جهت اتصال به تغذیه 9V- بکار می‌رود.

پایه شماره ۵ جهت کنترل پایه نوشتاری(wr=write) بکار می‌رود.

پایه شماره ۶ جهت کنترل (rd=read) بکار می‌رود.

پایه شماره ۷ مخفف (Chip Enabele) می‌باشد و شماره پایه‌ای است که برای فعال کردن چیپ موجود در داخل lcd بکار می‌رود.

پایه شماره ۸ مخفف (Code data)می‌باشد و شماره پایه‌ای است که برای کنترل پایه کد موجود در lcd بکار می‌رود.

پایه شماره ۱۰ جهت کنترل reset بکار می‌رود.

از پایه شماره ۱۱ الی ۱۸ که جهت انتقال data (D0 – D7) در نظر گرفته شده است.

پایه شماره ۱۹ مربوط به پایه‌ای است که برای کنترل (Font select) بکار می‌رود.

پایه شماره ۲۰ مربوط به پایه‌ای است که مشخص کننده تعداد ستون‌های متنی lcd است که می‌تواند ۶ یا ۸ باشد. تعداد پیکسل‌های عمودی بر این عدد تقسیم خواهد شد تا تعداد ستون‌ها بدست آید. اگر تعداد پیکسل‌های عمودی ۲۴ باشد، و عدد ۸ انتخاب شود، با تقسیم ۲۴ بر ۸ تعداد ستون‌ها ۳ فقره خواهد شد. اما اگر عدد ۶ انتخاب شود، تعداد ستون‌ها ۴ فقره خواهد شد.

در ادامه جهت پیکربندی lcd گرافیکی ابتدا نوع lcd را از نوع گرافیکی (Graphlcd) انتخاب کرده و سپس اندازه lcd را مشخص کنیم. اندازه‌ای که معمولاً پرکاربرد می‌باشد 128*128 می‌باشد.

در مرحله بعد باید یک port کامل را جهت انتقال Data (dataport) و همین‌طور یک port کامل، جهت port های کنترلی(Controlport) در نظر بگیریم. منظور از dataport همان db7 تا db0 در پیکربندی lcd کاراکتری است که بر اساس بیت‌های پرارزش و یا کم ارزش (d0- d3 کم ارزش) - (d4- d7 پر ارزش) در پیکر بندی lcd کاراکتری به‌کار گرفته می‌شوند. البته در lcd گرافیکی باید تمامی pinهای dataport مورد استفاده قرار گیرد.

همچنین در خصوص controlport نیز معرفی پایه‌های کنترلی بکاربرده شده در lcd گرافیکی می‌باشد که در بالا معرفی شده‌اند.

```
Config Graphlcd = 128 * 128 , Dataport = Portc , Controlport = Portb , Ce = 0 , Cd = 1 , Wr = 2 ,
Rd = 3 , Reset = 4 , Fs = 5 , Mode = 8
```

Mode=8 همان پایه‌ی شماره 20 می‌باشد که مشخص کننده تعداد ستون‌های متنی lcd است که می‌تواند ۶ یا ۸ باشد و در بالا معرفی شده است.

پس در پیکربندی lcd گرافیکی وقتی می‌گوئیمControlport = Portb , Ce = 0 یعنی پایه‌ی شماره 7 ، lcd گرافیکی باید به portb.0 متصل گردد و همینطور وقتی می‌گوئیم Cd=1 یعنی پایه‌ی شماره lcd،8 گرافیکی بایده به portb.1 متصل گردد و یا Wr=2 یعنی پایه شماره lcd،4 باید به portb.2 میکرو متصل گردد، Rd=3 یعنی پایه‌ی شماره lcd،5 باید به portb.3 وReset نیز به portb.4 وFs نیز به portb.5 میکرو متصل گردد. و در نهایت Mode=8 و یا Mode=6 نیز به جائی از میکرو متصل نمی‌شود و در واقع همان NC (No Conection) است.

پس از نوشتن پیکربندیlcd گرافیکی می‌توانیم مانند lcd کاراکتری، نمایش کاراکترهای مختلف بر روی lcd گرافیکی را شروع کنیم.

نکته: تمامی موارد و دستوراتی که در lcd کاراکتری مورد استفاده قرار می‌گرفت اعم از دستوراتی مانند Locate و ... در lcdگرافیکی نیز مورد استفاده قرار می‌گیرد. همچنین جهت نمایش کاراکتر نیز از دستور Lcd "avr" استفاده می‌شود.

اکنون پس از معرفی کامل و کاربرد هر کدام از پایه‌ها و نحوه‌ی نوشتن پیکربندی LCD گرافیکی می‌توانیم برنامه‌ای بنویسیم که بر روی LCD گرافیکی متنی را نمایش دهد. و در ادامه می‌خواهیم با استفاده از دستورات و مشخص کردن مختصات در صفحه‌ی LCD گرافیکی اشکال هندسی مختلفی را نمایش دهیم.

نکته: می‌دانیم هر آنچه که در تمامی صفحات مانیتوری اعم از تلویزیون، صفحه موبایل و ... جهت نمایش از پیکسل‌های بسیار ریز و زیادی تشکیل می‌شوند و از اتصال این نقاط به یکدیگر تصویر قابل مشاهد خواهد بود، به همین دلیل در برنامه نویسی نیز باید به این امر توجه شود.

با استفاده از دستور Pset X , Y , COLOR یک پیکسل به مرکزیت X و Y به ازای Color=1 روشن می‌شود که اگر Color=0 باشد، این پیکسل خاموش می‌شود. X و y می‌توانند از 0 تا 127 باشند، چون اندازه lcd ما 128*128 می‌باشد. بعبارت ساده‌تر ابتدا باید یک موقعیت را بر حسب پیکسل مشخص کرده سپس تصاویر را روی lcd نمایش داد. البته اگر این دستور را ننویسید میکرو به صورت پیش‌فرض پیکسل را فعال کرده و شما می‌توانید به راحتی تصاویر را نمایش دهید.

برنامه‌ای بنویسید که بر روی صفحهLCDگرافیکی یک خط مورب را نمایش دهد.

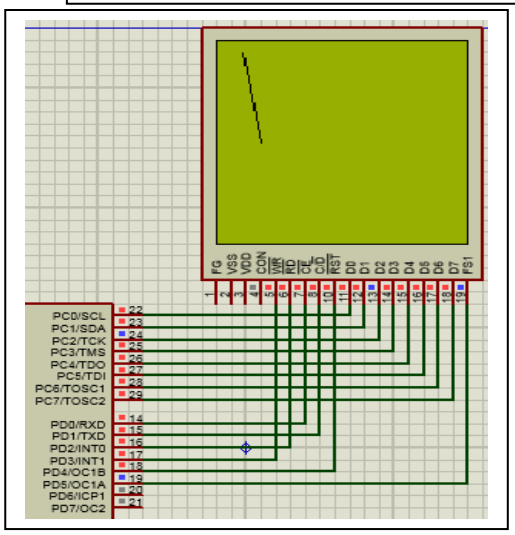
برای اینکار ابتدا باید با استفاده از دستورLine نقطه‌ی شروع و پایان خط را مشخص کنیم. صورت کلی دستور ایجاد خط به شکل زیر است.

Line (X_0 ,Y_0) - (X_1, Y_1) ,Coler

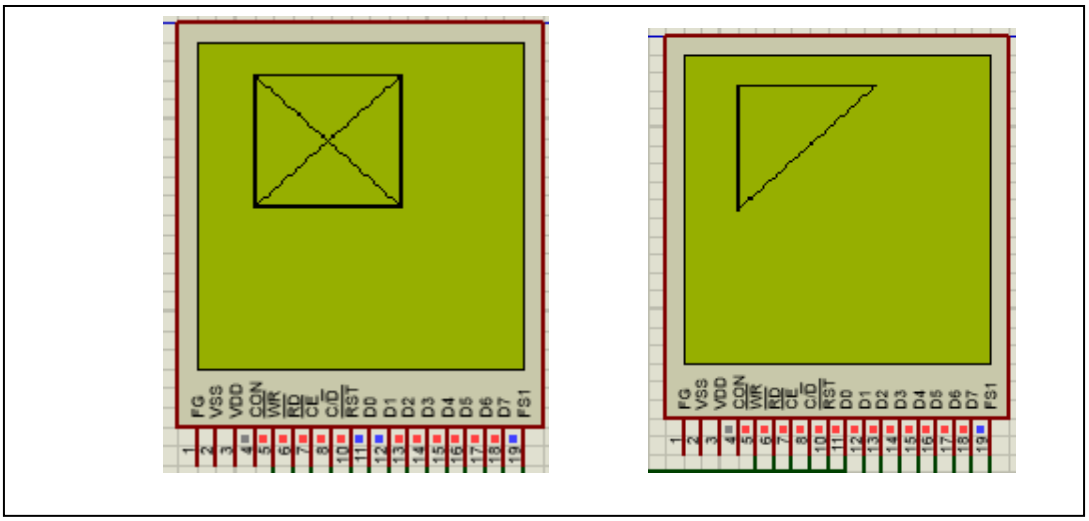
همانطوریکه پیداست جهت ایجاد یک خط حتماً باید ۴ نقطه در صفحهLCD گرافیکی مشخص گردد، دو نقطه در محورXها و همینطور دو نقطه در محورYها تعیین می‌گردد و به ازای Coler مساوی 255 خط رسم شده و به ازای Coler برابر 0 خط حذف می‌شود. حالا می‌توانیم با نوشتن دستور فوق و ارائه مقادیر عددی با توجه به اندازه lcd یک خط با مختصات دلخواه رسم کنیم.

```
$regfile = "m32def.dat"

$crystal = 1000000

Config Graphlcd = 128 * 128 , Dataport = Portc , Controlport = Portd , Ce = 0 , Cd = 1 , Rd = 2 , Wr = 3 , Reset = 4 , Fs = 5 , Mode = 8

Do

Cursor Off

Line(10 , 5) - (20 , 64) , 255

Waitms 200

Loop

End
```

رسم خط به مختصات(5,10) و (20 , 64))



همانطوریه که مشاهده کردید با مشخص کردن ابتدا و انتهای خط، با استفاده از دستور line می‌توان یک خط را رسم نمود. بدیهی است که با ارائه چندین نقطه میتوان یک مربع و یا مستعطیل و حتی مثلث و یا اشکال هندسی دیگر را که با اتصال چند خط به یکدیگر تشکیل می‌شوند را رسم نمود.




```
Line(20 , 10) -(20 , 64) , 255
Line(20 , 10) -(80 , 10) , 255
Line(80 , 64) -(80 , 10) , 255
```

```
Line(20 , 10) -(20 , 64) , 255
Line(20 , 10) -(80 , 10) , 255
Line(80 , 64) -(80 , 10) , 255
Line(80 , 64) -(20 , 10) , 255
Line(80 , 64) -(20 , 64) , 255
Line(20 , 64) -(80 , 10) , 255
```

همانطوریکه ملاحضه کردید برای برنامه نویسی این نوع پروژه‌ها ، علاوه براینکه باید رموز و ریزه‌کاری‌های برنامه نویسی آی سی های میکرو کنترلر را باید یاد بگیرید، آشنایی و تسلط بر تعدادی از اصول و فرمول‌های ریاضی و محورهای مختصات و غیره نیز لازم است.

حالا می‌خواهیم برنامه‌ای بنویسیم که بر روی lcd گرافیکی یک دایره کامل را رسم کند.

با توجه به موضوعاتی که در بالا بیان شد واضح است که برای رسم دایره نیز نیاز است که با مختصات lcd آشنا باشیم. برای رسم دایره نقطه‌ای را جهت مرکزیت دایره در نظر می‌گیریم و سپس با توجه به اندازه lcd عددی را بعنوان شعاع دایره نیز در نظر می‌گیریم. حالا کافی‌ست با استفاده از دستور Circle دایره‌ی مورد نظر را رسم کنیم.

دستور دایره به صورت کلی به شکل زیر بیان می‌شود:

CIRCLE (X_0 , Y_0) , RADIUS, COLOR

(X_0 , Y_0) موقعیت مرکز دایره را مشخص می‌کند.

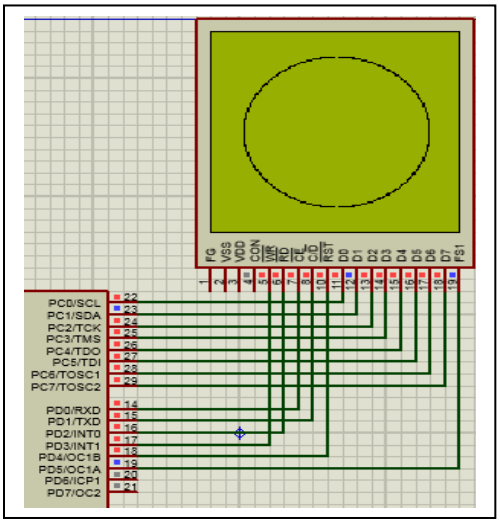
RADIUS شعاع دایره را معلوم می‌کند.

COLOR اگر 255 باشد، دایره رسم می‌شود و اگر 0 باشد پاک می‌شود.

اکنون در حالت کلی دستور مقدار عددی را قرار می‌دهیم: Circle(64 , 64) , 50 , 255

به این ترتیب دایره‌ای به مرکزیت 64 و64 و شعاع 50 خواهیم داشت. بدیهی است بدلیل اینکه اندازه LCD 128*128 می‌باشد، مرکز LCD نقطه 64 و64 خواهد بود و حداکثر شعاعی که می‌تواند در خود جای دهد 64 خواهد بود.

```
$regfile = "m32def.dat"

$crystal = 1000000

Config Graphlcd = 128 * 128 , Dataport = Portc , Controlport = Portd , Ce = 0 , Cd = 1 , Rd = 2 ,
Wr = 3 , Reset = 4 , Fs = 5 , Mode = 8

Do

Cursor Off

Circle(64 , 64) , 50 , 255

Waitms 200

Loop

End
```



برنامه‌ای بنویسید که بر روی LCD گرافیکی دوائرمتحدالمرکز را با شعاع دلخواه نمایش دهد.

برای نوشتن کد این برنامه باید به دستورات قبلی که کار کرده‌اید نیزرجوع کنید. با کمی دقت می‌توان دریافت، فرق خاصی با برنامه قبلی ندارد. دایره‌هایی که مرکز آنها یکی است، یعنی فقط شعاع آنها متغیر است .پس به این ترتیب می‌توان با تعریف یک متغیر و نوشتن حلقه for – next و قرار دادن متغیر به جای شعاع در دستور دایره، دوائر متحدالمرکز را رسم کرد.

ذکر این نکته خالی از لطف نیست که می‌توان اندازه شعاع را به دلخواه انتخاب نمود و با همان اندازه تغییر خواهد کرد.

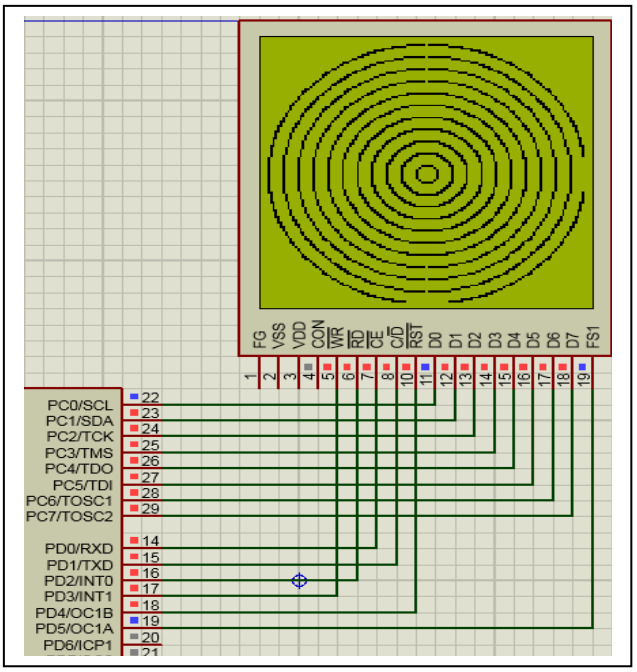
```
$regfile = "m32def.dat"

$crystal = 1000000

Config Graphlcd = 128 * 128 , Dataport = Portc , Controlport = Portd , Ce = 0 , Cd = 1 , Rd = 2 ,
Wr = 3 , Reset = 4 , Fs = 5 , Mode = 8
```

```
Dim C As Byte
Do
For C = 4 To 64 Step 4
Cursor Off
Circle(64 , 64) , C , 255
Waitms 200
Next C
Loop
End
```

حلقه for – next با فاصله 4

رسم دایره با شعاع متغیر

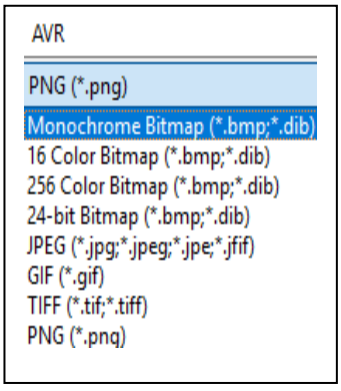
پایان حلقه for – next

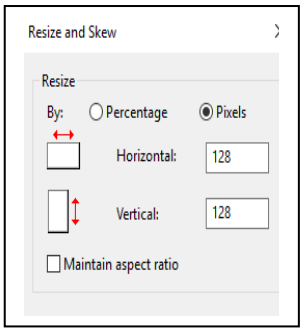


بدیهی است با جابجایی مرکز و یا شعاع می‌توانید اشکال هندسی مختلفی را نمایش دهید.

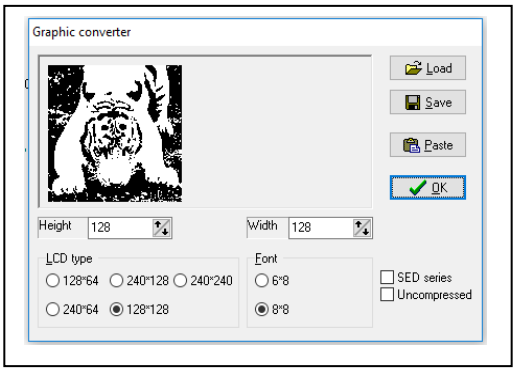
جهت نمایش دوائر متحدالمرکز به صورت جمع شدن به داخل کافی است با step منفی حلقه for شمارش را آغاز کند و یا حلقه for از عدد ۶۴ به طرف عدد ۴ جهت شمارش استفاده شود.

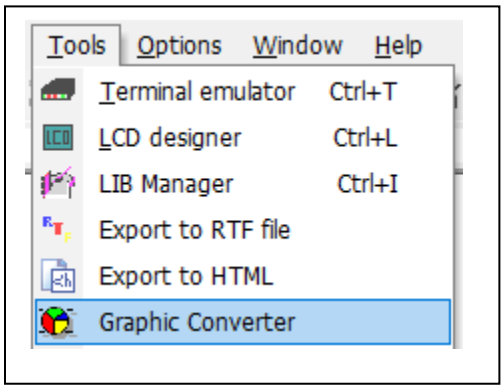
در ادامه می‌خواهیم یک تصویر واقعی را بر روی صفحه lcd گرافیکی نمایش دهیم. برای اینکار ابتدا باید عکس مورد نظر را در محیط paint ، Resaiz کرده ، در پنجره باز شده حالت Pixels را انتخاب کرده و گزینه های Horizantal وVertical را به اندازه LCD (128*128) تغییر می‌دهیم. در مرحله بعد با پسوند(*.bmp;*.dib) آن را ذخیره می‌کنیم.





با مراجعه به محیط نرم‌افزار BASCOM-AVR از نوار ابزارTools گزینه Graphic Converter را انتخاب کرده و سپس در پنجره باز شده گزینه Load را کلیک کرده و عکس مورد نظر را که در محیط Paint ذخیره کرده بودیم، فراخوانی می‌کنیم و تصویر را با پسوندbgf ذخیره می‌کنیم.





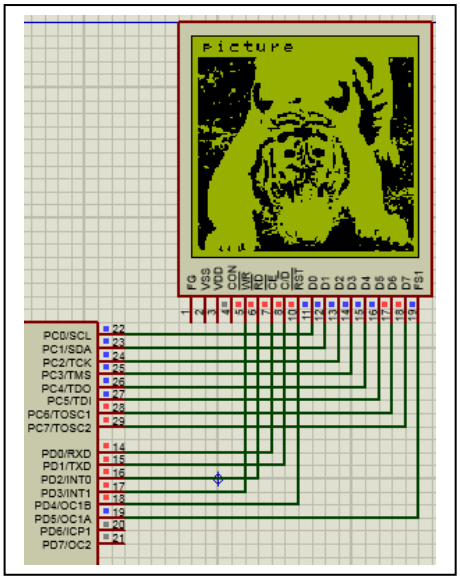
لازم به ذکر است در پنجره باز شده Graphic Converter بر اساس پیکربندی lcd که انجام داده اید باید اندازه lcd را نیز 128*128 انتخاب کنید.

```
$regfile = "m32def.dat"

$crystal = 1000000

Config Graphlcd = 128 * 128 , Dataport = Portc , Controlport = Portd , Ce = 0 , Cd = 1 , Rd = 2 ,
Wr = 3 , Reset = 4 , Fs = 5 , Mode = 8

Do

Cursor Off

Cls

Locate 1 , 1

Lcd "picture"

Waitms 500

Cls Text

Showpic 5 , 10 , Glcd

Waitms 500

Loop

End

Glcd:

$bgf "avr.bgf"
```

نمایش کلمه picture در سطر اول ستون اول

نمایش تصویر در موقعیت (5,10) ازجدول Glcd

فراخوانی تصویر ذخیره شده با پسوندbgf



باید این نکته را نیز در نظر داشت که در فرآیند نمایش تصویر روی lcd گرافیکی سه مرحله ذخیره سازی صورت می‌پذیرد. مرحله اول Resaiz در محیطpaint که با پسوند bmp.dib ، مرحله دوم Tools، Graphic Converter که باید با پسوندbgf

ذخیره می‌شوند و در نهایت، مرحله سوم که ذخیره سازی فایل bas می‌باشد که هر سه مرحله باید در یک پوشه و یا در یک درایو قرار داشته باشند تا فراخوانی به درستی صورت پذیرد و نمایش تصویر صورت پذیرد.

نکته :

در قسمت ذخیره سازی فایل با پسوند bgf هر نامی را که جهت ذخیره‌سازی در نظر گرفته‌اید، باید در برنامه قرار دهید.

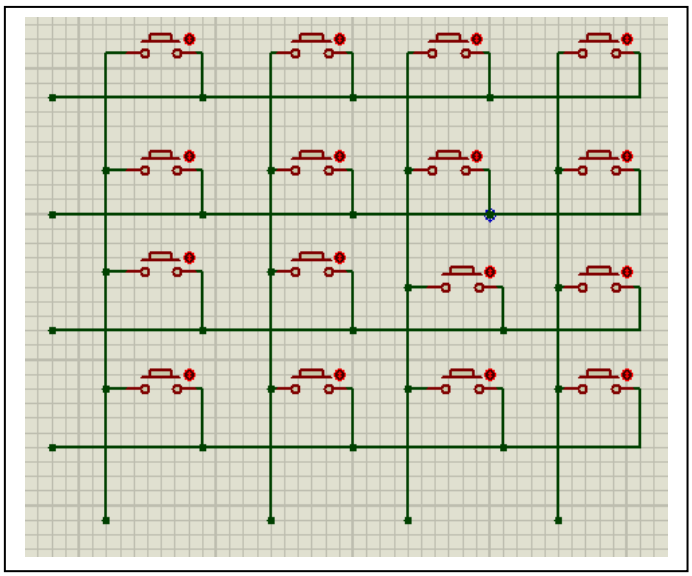
$bgf "نام فایل مورد نظر و ذخیره شده . bgf"

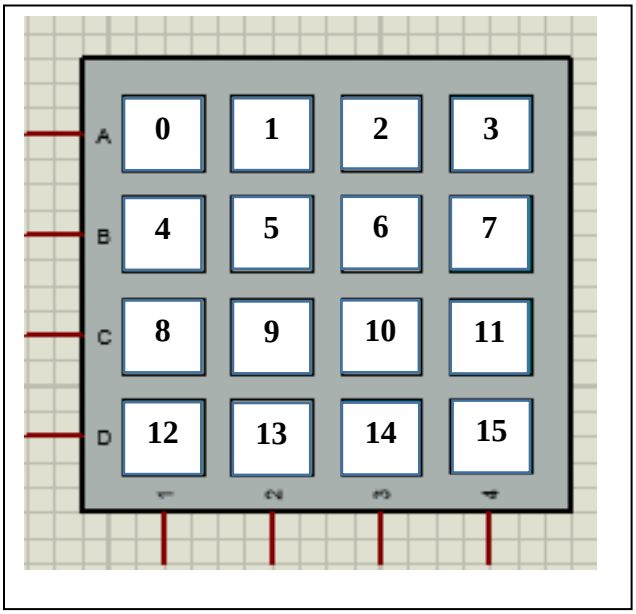
روشن کردن دو دیود نورانی توسط صفحه کلید 4*4

از اینجا به بعد، وارد مبحثی می‌شویم که مربوط به استفاده از صفحه کلیدهای موجود در بازار می‌شود که در صورت تسلط بر این برنامه‌های کوچک و ساده، بعداً خواهید توانست از صفحه کلید کامپیوتر نیز به نحو احسن استفاده کنید و این همان کار مهم و نخستینی است که برای طراحی و برنامه نویسی تابلوهای نویسنده روان لازم و ضروری است.

استفاده از صفحه کلیدها، طیف وسیعی از مدارهای الکترونیکی، از جمله مدارهای کنترل از راه دور و قفل رمز و غیره را در برمی‌گیرید. اگر قرار باشد اعداد 0 تا 15 را توسط کلیدهای فشاری معمولی به آی سی اعمال کنیم، طبیعی است که به ۱۶ عدد کلید احتیاج خواهد بود که هرکدام از آنها دارای دو پایه هستند و در نتیجه به ۳۲ رشته سیم اتصال نیاز خواهد بود که تعداد ۳۲ فقره از پایه‌های خروجی آی سی را اشغال خواهند کرد در حالیکه می‌دانیم میکرو atmega8 فقط دارای سه port خروجی و یا بعبارت دیگر فقط ۲۴ پایه قابل استفاده می‌باشد و لذا در این شرایط با مشکل روبه‌رو خواهیم شد.

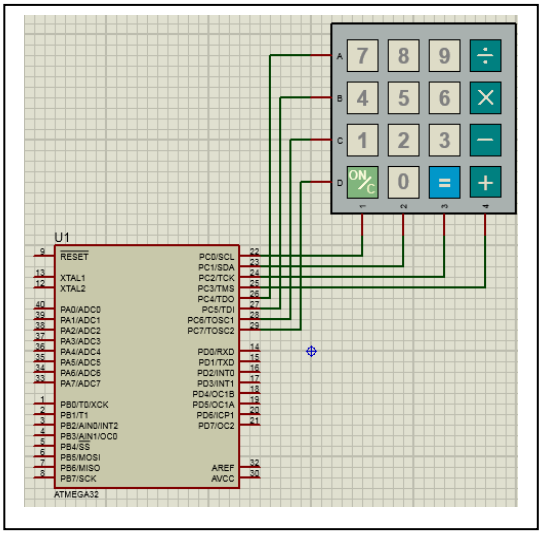
برای حل این مشکل، مانند طریقه اتصال LED های نویسنده روان و یا تقریباً سیستم 7Segment مولتی‌پلکس که قبلاً با آن آشنا شده‌اید، ۱۶ عدد کلید را طوری به هم وصل می‌کنیم که هر کدام از پایه‌های آن‌ها در ردیف‌های افقی و عمودی به هم وصل می‌کنیم که هر کدام از پایه‌های آن‌ها در ردیف‌های افقی و عمودی به همدیگر وصل می‌شوند و در نتیجه بجای ۳۲ رشته سیم، فقط به ۴+۴=۸ عدد خروجی نیاز خواهند داشت که در اینصورت، فقط یک port از آی‌سی برای استفاده از آن‌ها کفایت خواهد کرد. این نحوه اتصال را در شکل زیر مشاهده می‌کنید.





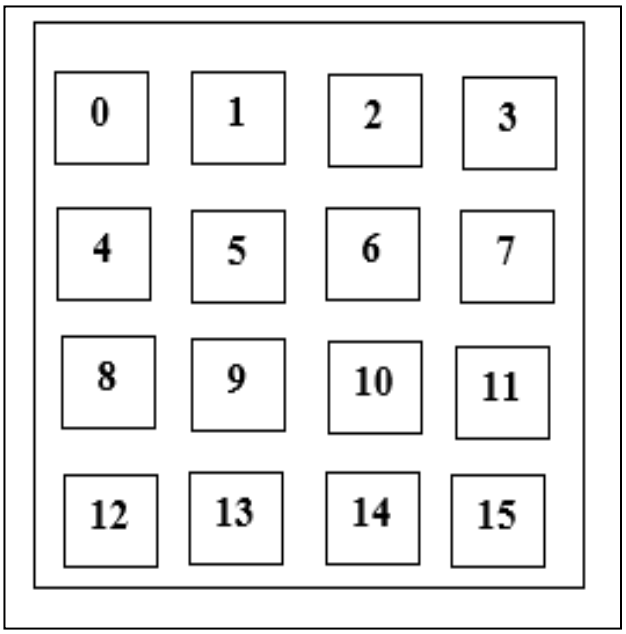
صفحه کلیدهای 4*4 موجود در بازار نیز به همین طریق طریق سیم‌بندی شده‌اند و فقط ۸ عدد پایه خروجی دارند که چهار ردیف عمودی اتصالات آن‌ها با حروف A , B , C , D و چهار ردیف افقی آنها با اعداد 1 , 2 , 3 , 4 مشخص شده‌اند. در شکل مقابل این موضوع به خوبی نشان داده شده است.

در شکل زیر نحوه اتصال صفحه کلید را به آی‌سی میکروکنترلر مشاهده می‌کنید.

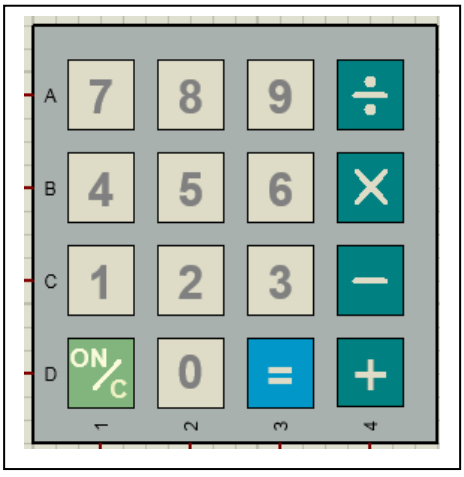


نحوه اتصال صفحه کلید به portd از میکرو طبق جدول زیر می‌باشد

portd	0	1	2	3	4	5	6	7
صفحه کلید	1	2	3	4	A	B	C	D

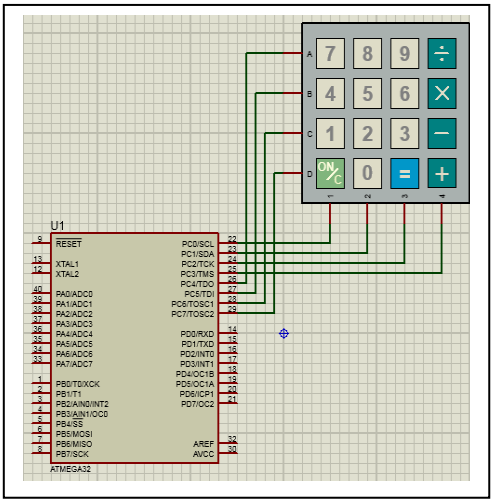


در صفحه کلید 4*4 با فشار هر کلید، یک مقدار عددی به میکرو ارسال می‌گردد که این اعداد از 0 تا15 می‌باشد و چون صفحه کلید دارای 16 کلید می‌باشد، پس باید 16 کد متفاوت به میکرو ارسال شود. اگر صفحه کلید را مطابق شکل فوق به میکرو اتصال دهیم، با فشار هر کلید، اعداد به ترتیبی که در شکل روبه‌رو مشاهده می‌کنید، ظاهر خواهند شد.

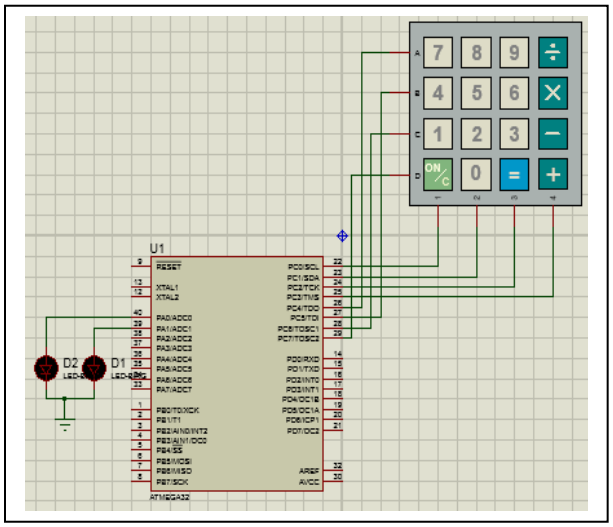


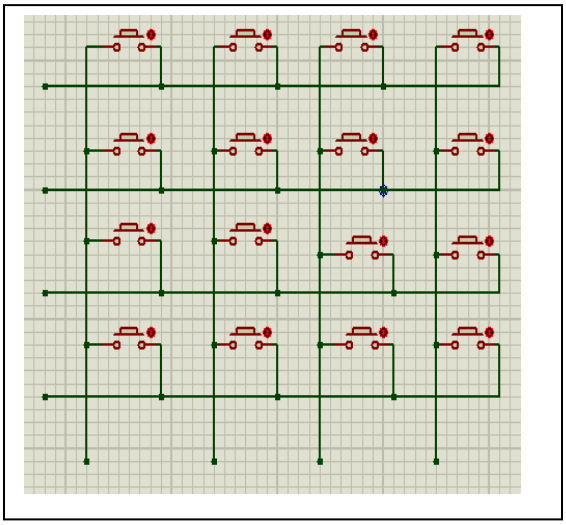
باید توجه داشته باشید که در صفحه کلیدهای موجود در بازار، اعداد اسکن شده از روی صفحه کلید بر روی آن نوشته نمی‌شود و ممکن است اعداد دیگری(بسته به موارد مصرف صنعتی آن) بر روی آ چاپ شده باشد. مانند صفحه کلید ماشین حساب که شکل آن را در روبه‌رو مشاهده می‌کنید.

اگر صفحه کلید ماشین حساب را طبق شکل مقابل به میکرو وصل کنیم و کلیدهای آن را یک به یک فشار دهیم، اعداد زیر اسکن خواهد شد.



با فشار دادن کلید مربوط به عدد 7 ، عدد 0 ظاهر خواهد شد.	با فشار دادن کلید مربوط به عدد 1 ، عدد 8 ظاهر خواهد شد.
با فشار دادن کلید مربوط به عدد 8 ، عدد 1 ظاهر خواهد شد.	با فشار دادن کلید مربوط به عدد 2 ، عدد 9 ظاهر خواهد شد.
با فشار دادن کلید مربوط به عدد 9 ، عدد 2 ظاهر خواهد شد.	با فشار دادن کلید مربوط به عدد 3 ، عدد 10 ظاهر خواهد شد.
با فشار دادن کلید مربوط به عدد ÷ ، عدد 3 ظاهر خواهد شد.	با فشار دادن کلید مربوط به عدد - ، عدد 11 ظاهر خواهد شد.
با فشار دادن کلید مربوط به عدد 4 ، عدد 4 ظاهر خواهد شد.	با فشار دادن کلید مربوط به عدد on/c ، عدد 12 ظاهر خواهد شد.
با فشار دادن کلید مربوط به عدد 5 ، عدد 5 ظاهر خواهد شد.	با فشار دادن کلید مربوط به عدد 0 ، عدد 13 ظاهر خواهد شد.
با فشار دادن کلید مربوط به عدد 6 ، عدد 6 ظاهر خواهد شد.	با فشار دادن کلید مربوط به عدد = ، عدد 14 ظاهر خواهد شد.
با فشار دادن کلید مربوط به عدد* ، عدد 7 ظاهر خواهد شد.	با فشار دادن کلید مربوط به عدد + ، عدد 15 ظاهر خواهد شد.





متن برنامه روشن کردن دو دیود نورانی توسط صفحه کلید 4*4

```
$regfile "m8def.dat"
$crystal = 8000000
Config Portb = Output
Config Kbd = Portd , Debounce = 50 , Delay = 1
Dim A As Byte
Do
A = Getkbd( )
If A < 16 Then
Select Case A
Case 0:
Set Portb.0
Case 1:
Set Portb.1
End Select
End If
Loop
End
```

در سطر اول و دوم برنامه طبق معمول، مربوط می‌شود به معرفی آی سی میکروی بکار رفنه در مدار و مقدار فرکانس کریستال خارجی در آن می‌رسیم به دستور Config Kbd = Portd , Debounce = 50 , Delay = 1 شکل کلی دستور به شرح زیر می‌باشد: Konfig Kbd = Port X , Debounce = Value [, Delay = Value]

Portx مشخص کننده پورتی است که برای اسکن صفحه کلید از آن استفاده می‌کنیم که در اینجا portd انتخاب شده است.

Debounce بصورت پیش فرض عدد ۲۰ می‌باشد که در اینجا ۵۰ انتخاب شده است و می‌تواند ماکزیمم مقدار تا ۲۵۵ را داشته باشد.

DELAY برای انتخاب تاخیری اختیار است که نویزهای محیط را کاهش می‌دهد.

دستور Dim A As Byte انتخاب یک متغیر مانند A از نوع بایت می‌باشد.

دستور A=GetKbd() از فرامین مهم برنامه‌نویسی (صفحه کلید) می‌باشد که:

این دستور صفحه کلید را خوانده و عدد متناظر با کلید فشرده شده را در متغیر انتخابی قرار می‌دهد و اگر کلید فشرده نشود، عدد ۱۶ را بر می‌گرداند.

در اینجا مقدار متناظر با کلید فشرده شده توسط دستور Getkbd() خوانده شده و در متغیر A قرار می‌دهد.

دستور if A < 16 then خواسته شده که اگر مقدار A کوچکتر از ۱۶ شد، برود دستور بعدی را اجرا کند، چون مقدار اسکن شده بیش از ۱۵ نبوده و مابین 0 تا15 می‌باشد.

دستورات Select Case A به این دستورات اصطلاحاً Select – Case گفته می‌شود و منظور از آن‌ها این است که : اگر عدد 0 از اسکن صفحه کلید اسکن شود، Case0: را انتخاب و portb.0 را که به led1 وصل است set یا در واقع 1 خواهد کرد و برعکس، اگر 1 اسکن شود، آنگاه دستور Case1: را اجرا کرده و portb.1 را که به led2 بسته شده است set یا 1 خواهد کرد.(این دستور را می‌توان برای تمامی خروجی‌ها تکرار کرد)

```
Select Case A
Case():
Set portb.0
Case 1:
Set portb.1
```

دستورات

به معنی پایان دادن به دستورات Select و اتمام دستورات شرطی و پایان برنامه می‌باشد.

توضیح بیشتر درباره دستور Select – Case

```
End Select
End If
Loop
end
```

شکل کلی دستور به صورت زیر هست:

```
Select Case Var
Case test1:
Statement 1
Case test2:
Statement 2
….....
….....
….....
Case Else:
Statement 3
End Select
```

اگر متغیر به test1 برابر باشد، دستور Statment1 اجرا می‌شود اگر Var با test2 برابر باشد، دستور Statment2 اجرا می‌شود و الی آخر ..... و در نهایت اگر var با هیچکدام از مقادیر ... و test1 , test2 برابر نباشد، دستور Statment3 اجرا می‌شود که بعد از Else Case نوشته شده است.

روشن کردن دو دیود نورانی توسط صفحه کلید 4*4 با یک برنامه جدید

```
$regfile "m16def.dat"
$crystal = 1000000
Config Portb = Output
Config Kbd = Portd , Debounce = 50 , Delay = 1
Dim A As Byte
Do
A = Getkbd()
A = Lookup(a , Dfpm)
If A > 16 Then
Select Case A
Case 4:
Set Portb.0
Case 5:
Set Portb.1
End Select
End If
```

Loop

End

Dfpm:

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

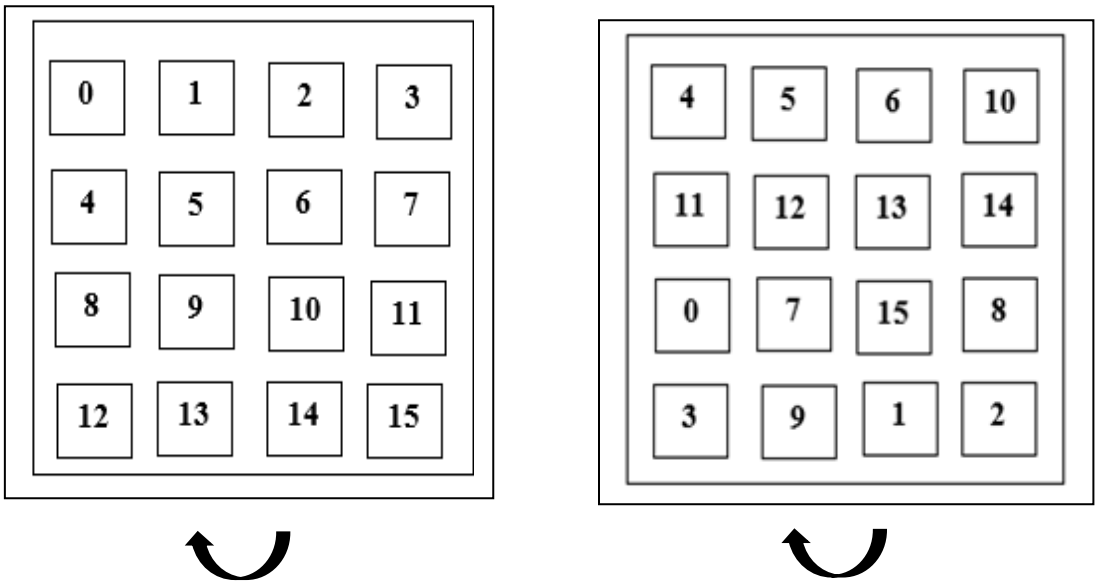
با یک نگاه کلی متوجه خواهید شد که همان برنامه قبلی است با این تفاوت که بعد از دستور A=GETKBD( ) دستور جدید A=LOOKUP(A , DFPM) که قبلاً نیز با آن آشنا شده‌اید(در قسمت7SEG)، اضافه شده است که در واقع با این دستور، برنامه ارجاع به یک زیر برنامه با نماد DFPM مشخص گردیده است.

و در زیر برنامه بعد از دستور END با دستورات :

DFPM:

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

اعدادی را که قرار است با فشار صفحه کلید از 0 تا 15 اسکن شوند ، درخواست شده است.



برای این منظور باید در برنامه از دستور LOOKUP استفاده کرده . پس در زیر برنامه Data طبق جدول زیر کپی می‌کنیم.

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

چگونه این اعداد اسکن می‌شوند؟

در دستور A = Lookup(a , Dfpm) ابتدا به زیر برنامه Dfpm پریده و داده با اندیس A را بر میگرداند که اندیس اولین داده برابر 0 و اندیس دومین داده برابر 1 الی تا اندیس 15 ..... که این اندیس را متغیر A تعیین می‌کند و از فشار دادن کلیدها مقدار اندیس A تعیین می‌کند و از فشار دادن کلیدها مقدار اندیس A معلوم می‌گردد که از 0 تا 15 می‌باشد.

مثلاً با فشار اولین کلید بالا از سمت چپ مقدار A=0 شده و در نتیجه در دستور A = Lookup(0 , Dfpm) مقدار 0 در داخل پرانتز قرار گرفته پس به زیر برنامه Dfpm پریده و با داده اندیس 0 که همان اولین داده است را برمی‌گرداند که برابر 4 می‌باشد.

4 = Lookup(0 , Dfpm)

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

که حالت‌های زیر اتفاق خواهد افتاد:

داده با اندیس 0	4 = LOOKUP(0 , DFPM)
داده با اندیس 1	5 = LOOKUP(1 , DFPM)
داده با اندیس 2	6 = LOOKUP(2 , DFPM)
داده با اندیس 3	10 = LOOKUP(3 , DFPM)
داده با اندیس 4	11 = LOOKUP(4 , DFPM)
داده با اندیس 5	12 = LOOKUP(5 , DFPM)
داده با اندیس 6	13 = LOOKUP(6 , DFPM)
داده با اندیس 7	14 = LOOKUP(7 , DFPM)
داده با اندیس 8	0 = LOOKUP(8 , DFPM)
داده با اندیس 9	7 = LOOKUP(9 , DFPM)
داده با اندیس 10	15 = LOOKUP(10 , DFPM)
داده با اندیس 11	8 = LOOKUP(11 , DFPM)
داده با اندیس 12	3 = LOOKUP(12 , DFPM)
داده با اندیس 13	9 = LOOKUP(13 , DFPM)
داده با اندیس 14	1 = LOOKUP(14 , DFPM)
داده با اندیس 15	2 = LOOKUP(15 , DFPM)

نکته مهم :

به دستور مقابل توجه کنید : A = LOOKUP(A , DFPM)

در دستور بالا کلمه A دو بار به کار برده شده است، که A داخل پرانتز با A بیرون پرانتز از مظر مقدار با هم فرق دارند و برابر نیستند. A داخل پرانتز، اندیس داده برگشتی از جدول می‌باشد و A بیرون پرانتز داده برگشتی از جدول می‌باشد.

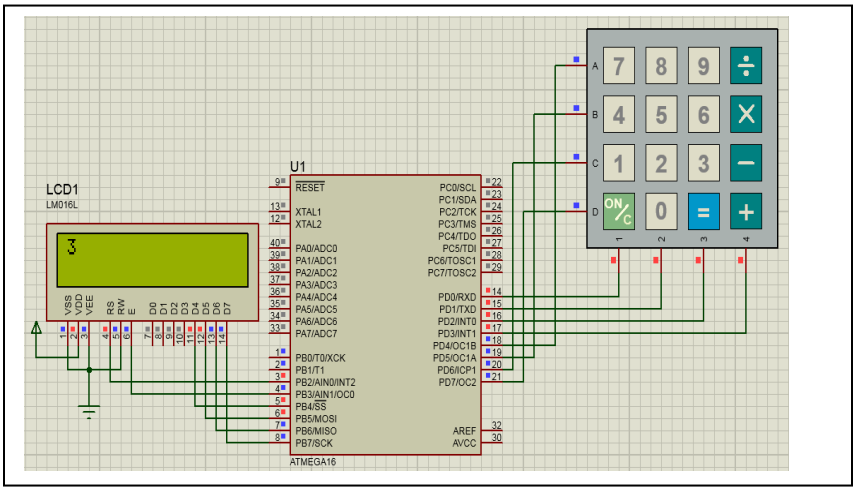
استفاده از LCD کاراکتری برای نمایش اعداد اسکن شده توسط صفحه کلید

متن برنامه :

```
$regfile "m8def.dat"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Portb.4 , Db5 = Portb.5 , Db6 = Portb.6 , Db7 = Portb.7 , Rs = Portb.2 , E = Portb.3
Config Portb = Output
Config Kbd = Portd , Debounce = 50 , Delay = 1
Dim A As Byte
Do
A = Getkbd()
A = Lookup(a , Dfpm)
If A < 16 Then
Select Case A
Case 0 To 15
Cls
Lcd A
End Select
End If
Loop
End
Dfpm:
Data 4 , 5 , 6 , 10 , 11 , 12 , 13 , 14 , 0 , 7 , 15 , 8 , 3 , 9 , 1 , 2
```

شرح و تفسیر برنامه :

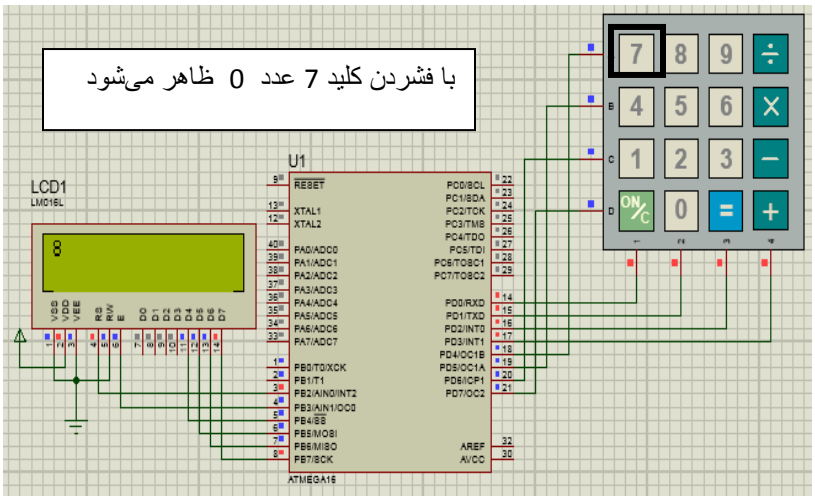
در این پروژه، به جای LED از LCD کاراکتری برای مشاهده تغییرات بوجود آمده از اسکن صفحه کلید استفاده شده است و هیچ چیز تازه‌ای در برنامه وجود ندارد تا توضیح داده شود. چرا که این برنامه، تلفیقی از دو برنامه صفحه کلید و lcd کاراکتری می‌باشد که قبلاً بارها آنرا تجره کرده اید و شرح و تفسیر کامل این برنامه را به عهده خودتان قرار می‌دهیم تا اندوخته و یافته های خود را تا این جای کتاب، محک بزنید و خود را بیازمائید.



نمایش اعداد اسکن شده از صفحه کلید بر روی lcd کاراکتری

```
$regfile "m16def.dat"
$crystal = 1000000
Config Lcdpin = Pin , Db4 = Portb.4 , Db5 = Portb.5 , Db6 = Portb.6 , Db7 = Portb.7 , Rs = Portb.2 , E = Portb.3
Config Portb = Output
Config Kbd = Portd , Debounce = 50 , Delay = 1
Dim A As Byte
Do
A = Getkbd()
If A < 16 Then
Select Case A
Case 0 To 15
Cls
Cursor Off
Lcd A
End Select
End If
Loop
End
```

همانطور که ملاحضه می‌کنید، این برنامه نسبت به برنامه‌های قبلی ساده است و برای متغییر A درخواست شده است که با فشار کلیدهای صفحه کلید بطور مرتب از 7تا + اعداد 0تا 15 را نمایش دهد. یعنی به همان ترتیبی که قبلاً اشاره کرده بودیم که اعداد اسکن شده توسط دستور جدید: Cace 0 to 15 انجام می‌گیرید.



کلید تبدیل راه پله با استفاده از دستور IF

اکنون طبق آموخته‌های قبلی میخواهیم برنامه‌ای بنویسیم که عمل کلید تبدیل راه پله را انجام دهد.

فرض کنید شما در یک ساختمان دو طبقه حضور دارید و قرار است که از پله‌ها بالا بروید، در این حالت شما کلید را روشن کرده و به طبقه بالا می‌روید و پس از رسیدن به طبقه بالا با کلید دیگری که در همان طبقه بالاست باید چراغ خاموش شود. اما همانطور که می‌دانید هر کلید دو وضعیت خاموش (صفر) و یا روشن (یک) می‌باشد، پس بطور ذاتی برگشتن به طبقه پایین اصلاً منطقی نیست و کاری بیهوده است.

همانطوریکه گفته شده هر کلید دارای دو وضعیت صفر و یا یک است. در اینجا دو کلید داریم و در نتیجه چهار وضعت خواهیم داشت. در ادامه با استفاده از آموخته‌های قبلی می‌توانیم چهار و ضعیت را به صورت زیر بنویسیم.

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

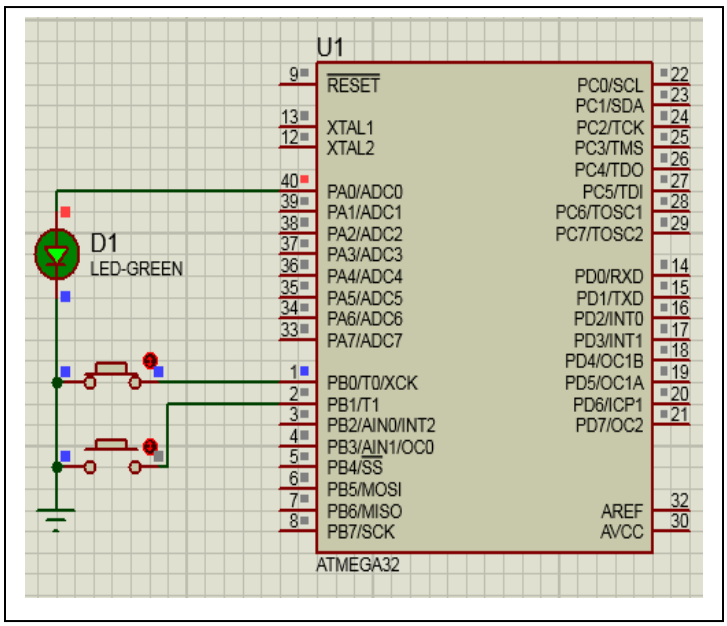
فرض کنید در جدول روبه رو ستون A کلید طبقه اول و B کلید طبقه بالایی و F لامپ یا همان خروجی باشد . در وضعیت 0 و 0 طبیعی است که نتیجه خروجی 0 خواهد بود زیرا هر دو ورودی 0 می‌باشند، در وضعیت 0 و 1 نتیجه خروجی 1 خواهد شد چون یکی از ورودی‌ها 1 است و در نتیجه حتما خروجی بایدیک شود. همینطور در وضعیت 1 و 0 نیز و 1 نتیجه خروجی 1 خواهد شد چون یکی از ورودی‌ها 1 است و در نتیجه حتما خروجی بایدیک شود. در وضعیت 1 و 1 نیز مانند وضعیت اولی (0 و0) نتیجه خروجی 0 خواهد بود زیرا هر دو ورودی شبیه به هم هستند.

با توجه به موارد یاد شده در بالا می‌توان نتیجه گرفت که هرگاه مقادیر ورودی برابر بود خروجی 0 خواهد شد و هرگاه مقادیر ورودی با یکدیگر متفاوت بود نتیجه خروجی 1 خواهد شد. پس در نهایت می‌توانیم بگوئیم که جدول بالا بیانگر گیت XOR می‌باشد.

اگر قرار باشد برنامه کلید تبدیل راه پله را با استفاده از دستور IF بنویسیم ابتدا باید دو از pinهای میکرو را تحت عنوان ورودی تعریف کرد، بعنوان مثال Pinb.1 = Input و Pinb.0 = Input که همان A و B های تعریف شده در جدول بالا هستند و همچنین یکی از portهای میکرو را بعنوان خروجی در نظر گرفت، مثلاً porta.0 که همان F یا نتیجه خروجی در جدول بالا می‌باشد.

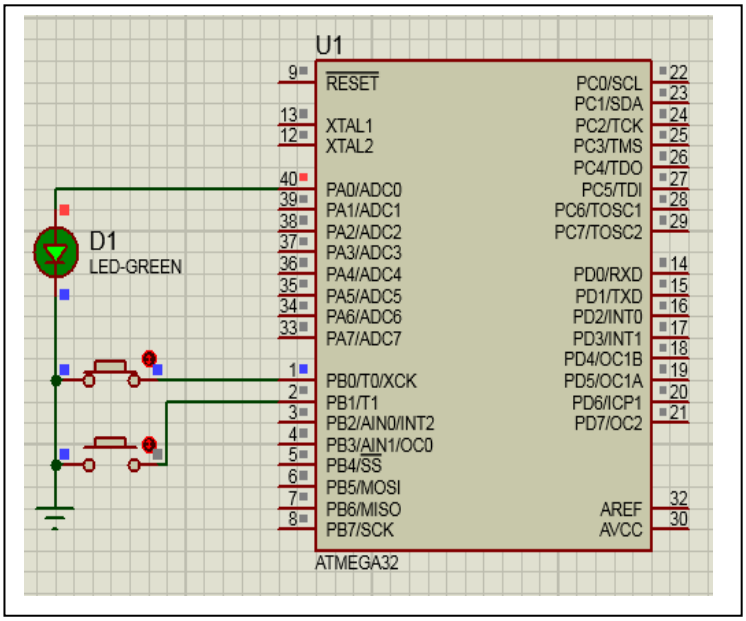
پس از معرفی ورودی و خروجی می‌توانیم بنویسیم If Pinb.0 = 0 And Pinb.1 = 0 آنگاه نتیجه خروجی Porta.0 = 0 شود. در ادامه یا می‌توانیم از دستور elseif (یعنی در غیر اینصورت) استفاده کنیم، همچنین برای سه وضعیت دیگر نیز همین عمل تکرار می‌شود.

```
$regfile "m32def.dat"
$crystal = 1000000
Config Porta.0 = Output
Config Pinb.0 = Input
Config Pinb.1 = Input
Do
If Pinb.0 = 0 And Pinb.1 = 0 Then
Porta.0 = 0
Elseif Pinb.0 = 0 And Pinb.1 = 1 Then
Porta.0 = 1
ElseIf Pinb.0 = 1 And Pinb.1 = 0 Then
Porta.0 = 1
ElseIf Pinb.0 = 1 And Pinb.1 = 1 Then
Porta.0 = 0
End If
Loop
End
```



کلید تبدیل راه پله با گیت xor

```
$regfile "m32def.dat"
$crystal = 1000000
Config Porta.0 = Output
Config Pinb.0 = Input
Config Pinb.1 = Input
Dim A As Bit
Do
A = Pinb.0 Xor Pinb.1
Porta.0 = A
Loop
End
```



چشمک زن Led

```
$regfile "m32def.dat"
$crystal = 8000000
Config Portc = Output
Config Timer1 = Timer , Prescale = 8
Timer1 = 34285
Enable Interrupts
Enable Ovf1
Enable Timer1
On Ovf1 M1
Start Timer1
Do
Loop
M1:
Timer1 = 34285
Start Timer1
Toggle Portc
```

Return

شمارش عدد و توقف باkey

```
$regfile "m32def.dat"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Portc.0 , Db5 = Portc.1 , Db6 = Portc.2 , Db7 = Portc.3 , E = Portc.5 , Rs = Portc.4
Config Lcd = 16 * 2
Config Pina.0 = Input
Dim A As Byte
A = 3
Do
Incr A
Cls
Lcd A
Waitms 200
Loop Until A = 35
End
```

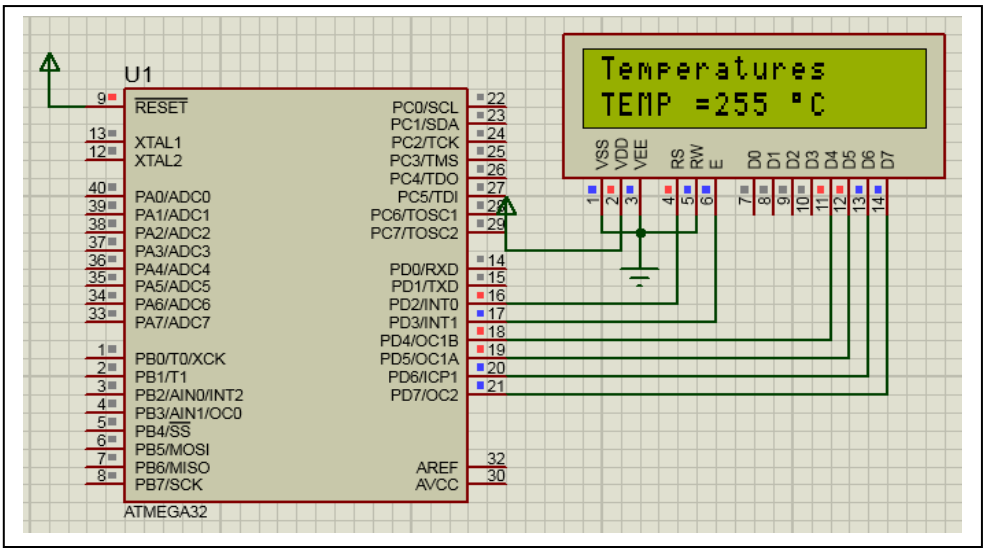
در این مرحله نیز با یک دستور جدید آشنا می‌شود. هرگاه لازم باشد شمارش تا مقدار محدودی انجام شود از دستور loopuntil استفاده می‌شود و به معنی توقف تکرار حلقه می‌باشد.

توقف تکرار حلقه در عدد مورد

برنامه‌ی سنسور دما با LM35 روی LCD را بنویسید.(با کاراکتر درجه)

```
$regfile "m32def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Adc = Single , Prescaler = Auto , Reference = Avcc
Dim A As Word
```

```
Deflcdchar 0 , 28 , 20 , 28 , 32 , 32 , 32 , 32 , 32
Start Adc
Cls
Cursor Off
Do
Locate 1 , 1
Lcd "Temperatures"
Wait 2
A = Getadc(۰)
A = A / 4
Locate 2 , 1
Lcd "TEMP =" ; A ; " " ; Chr(0) ; "C" " ; "
Waitms 200
Loop
End
```



همچنین برنامه فوق را می‌توان به طریق زیر هم نوشت.

```
$regfile "m32def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portd.3
, Rs = Portd.2
Config Adc = Single , Prescaler = Auto , Reference = Avcc
Dim Lm35 As Word
Start Adc
```

Do

Lm35 = Getadc(0)

Lm35 = Lm35 / 4

Cls

Cursor Off

Cls

Locate 1 , 1

Lcd "Temperatures"

Locate 2 , 1

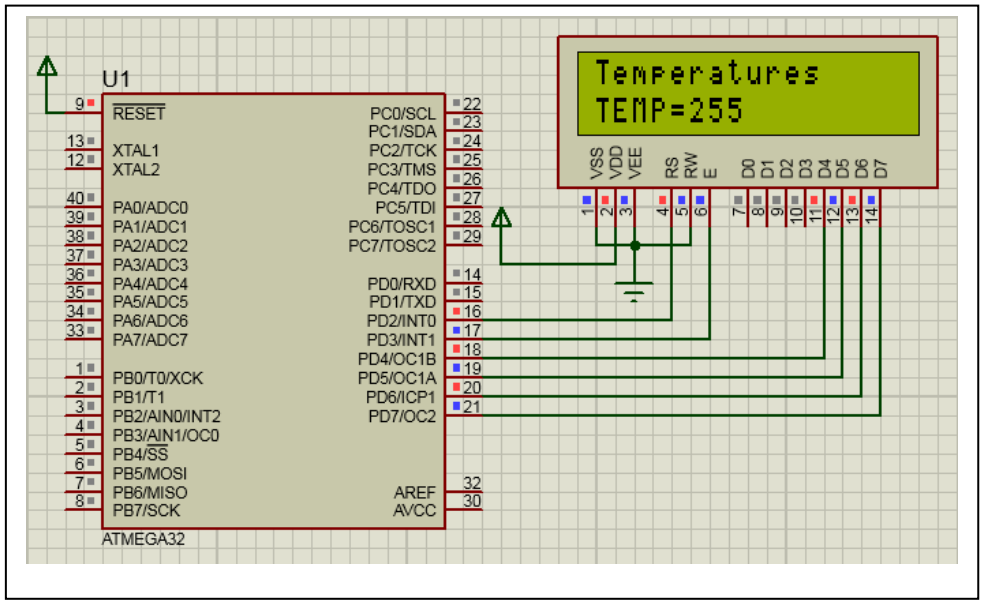
Lcd "TEMP="

Lcd Lm35

Waitms 200

Loop

End



برنامه ی PWM را بنویسید.

$regfile "m32def.dat"

$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2

Cursor Off

cls

Config Portc.6 = Input

Config Portc.7 = Input

Config Portd.4 = Output

Config Portd.5 = Output

Config Portd.7 = Output

Dim A As Word

Dim B As Word

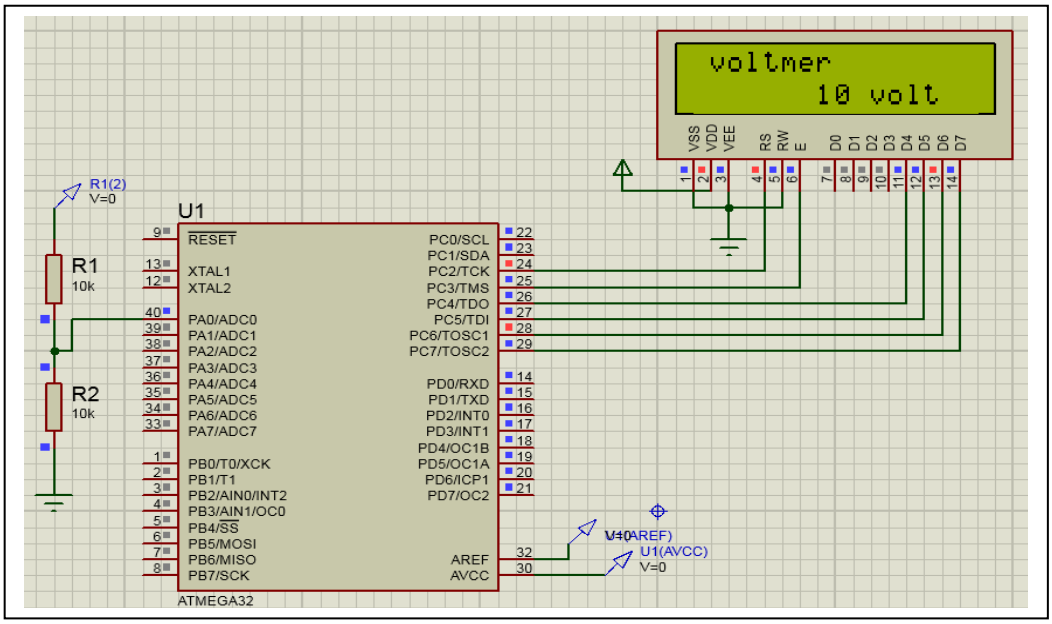
Config Timer1 = Pwm , Pwm = 8 , Compare A Pwm = Clear Up , Compare B Pwm = Clear Down , Prescale = 1

```
A = 100
Do
If Pinc.6 = 1 Then
Incr A
End If
If Pinc.7 = 1 Then
Decr A
End If
ocr2 = 130
Pwm1a = A
Pwm1b = 100
Home
Lcd A ; " "
Loop
End
```

برنامه voltmeter را بنویسید؟

```
$regfile "m32def.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 Adc = Single , Prescaler = 8
Config Portc = Output
Config Portd = Output
Dim X As Word
Cursor Off
Do
```

```
Start Adc
X = Getadc(0)
X = X / 100
Locate 1 , 1
Lcd " voltmer"
Locate 2 , 8
Lcd X ; " " ; "volt"
Waitms 200
Cls
Loop
End
```



برنامه ی کنترل دور استپ موتور را بنویسید.

```
$regfile = "m32def.dat"
$crystal = 1000000
Config Graphlcd = 128 * 128 , Dataport = Portc , Controlport = Portd , Ce = 0 , Cd = 1 , Rd = 2 , Wr = 3 , Reset = 4 , Fs = 5 , Mode = 8
Dim A As Byte , B As Byte , E As Byte , F As Byte
Cursor Off
Do
Cls Graph
Locate 2 , 78 : Lcd "step motor"
Locate 3 , 30 : Lcd "control"
Waitms 500
Cls Text
Waitms 500
Circle(20 , 80) , 10 , 128
Waitms 500
For A = 1 To 50
E = 128
For B = 1 To 4
```

```
Rotate E , Left
Portc = E
Waitms 20
Next B
Next A
For A = 1 To 50
F = 129
E = 128
For B = 1 To 4
Rotate E , Left
If A = 24 Then
F = 9
Portc = F
Waitms 50
Next B
Next A
End If
Loop
End
```