دانشکده فنی و مهندسی

گروه مهندسی کامپیوتر

دستورکار آزمایشگاه ریزپردازنده

مؤلف: دکتر فهیمه یزدان‌پناه، دکتر محمد علائی

اعضای هیأت علمی گروه مهندسی کامپیوتر

مقدمه

میکروکنترلر یک تراشه الکترونیکی قابل برنامه‌ریزی است که استفاده از آن باعث افزایش سرعت و کارآیی مدار در مقابل کاهش حجم و هزینه مدار می‌گردد. با ساخت میکروکنترلرها تحول شگرفی در ساخت تجهیزات الکترونیکی نظیر لوازم خانگی، صنعتی، پزشکی، تجاری وآموزشی به وجود آمده است که بدون آن تصور تجهیزات و وسایل پیشرفته امروزی غیرممکن است. یکی از میکروکنترلرهای پرکاربرد و معروف AVR محصول شرکت Atmel می‌باشد. اولین و بهترین گزینه برای ورود به دنیای الکترونیک دیجیتال، یاد گرفتن میکروکنترلرهای هشت بیتی AVR است چرا که اصولی‌ترین و پایه‌ای‌ترین مباحث الکترونیک دیجیتال در آموزش این میکروکنترلرها وجود دارد که به عنوان پیش‌نیازی برای یادگیری سطوح بالاتر و میکروکنترلرهای قوی‌تر همانند ARM و آرایه‌های منطقی برنامه‌پذیر FPGA است.

یک سیستم میکروکامپیوتر حداقل شامل میکروپرسسور یا واحد پردازش مرکزی، حافظه موقت (RAM)، حافظه دائمی (ROM) و قطعات ورودی/خروجی می‌باشد که بوسیله گذرگاه مشترک به هم ارتباط دارند و همه مجموعه روی یک برد اصلی به نام مادربرد قرار می‌گیرند. طرز کار یک میکروکامپیوتر به این صورت است که ابتدا برنامه‌ایی که در حافظه دائمی قرار داده‌شده است، اجرا می‌شود تا سیستم بوت‌شده و در حالت آماده به کار قرار گیرد. سپس بر اساس برنامه کاربر یا برنامه سیستم عامل، دستورات در واحد پردازش مرکزی اجرا می‌شوند و برحسب نوع برنامه، داده‌های مناسب در صورت نیاز از ورودی یا از حافظه گرفته می‌شوند و بعد از پردازش به خروجی ارسال می‌شود.

هنگامی که قطعات سازنده یک میکروکامپیوتر در یک تراشه و در کنار هم قرار گیرند، یک میکروکنترلر به وجود می‌آید. در واقع میکروکنترلر یک تراشه شامل یک واحد پردازش مرکزی، به همراه مقدار مشخصی از حافظه‌های RAM و ROM، پورت‌های ورودی/خروجی و همچنین واحدهای جانبی دیگری نظیر تایمر و رابط سریال و واسط های مختلف می‌باشد.

معرفی میکروکنترلرهای AVR

میکروکنترلرها انواع مختلفی دارند و بسته به نوع کاری که موردنظر است، یکی از خانواده‌های میکروکنترلرها که برای انجام آن مناسب‌تر است، انتخاب می‌شود. مثلا اگر سرعت پردازش بسیار بالا بدون هنگ کردن و قابلیت تغییر کل برنامه در یک کاربرد خاص در یک پروژه نیاز باشد بهتر است با FPGA طراحی کرد. اگر در یک پروژه سرعت نسبتا بالا و قابلیت پشتیبانی از انواع ارتباطات جانبی مانند پورت USB، ارتباطات سریال مورد نیاز باشد (مانند استفاده در تلفن همراه، تبلت‌ها، پروژه‌های پردازش سیگنال، تلویزیون‌ها و.) .. بهتر است از میکروکنترلرهای ARM استفاده کرد. اگر در یک پروژه سرعت بالا موردنظر نباشد و فقط درست و بدون نقص انجام شدن کار موردنظر باشد (مانند پروژه‌های صنعتی) از میکروکنترلرهای PIC استفاده می‌شود که در محیط‌های پرنویز مانند کارخانه‌ها بیشتر از آنها استفاده می‌شود. و در نهایت اگر در کاربردهایی معمولی و متوسط با قابلیت‌های متوسط مانند پروژه‌های دانشگاهی موردنظر باشد از میکروکنترلرهای AVR بیشتر استفاده می‌گردد. بنابراین یاد گرفتن میکروکنترلرهای AVR در مرحله اول ضروری است چرا که از نظر معماری و کاربردها ساده‌تر بوده و مباحث اصلی و پایه‌ای در این مرحله وجود دارد.

انواع میکروکنترلرهای AVR

این میکروکنترلرها دارای چهار سری می‌باشند که هر سری کاربردها و ویژگی‌های خاص خود را دارد.

ATtiny	میکروکنترلرهای کوچک، کم‌مصرف و پرقدرت برای کاربردهای خاص می‌باشند که دارای حافظه Flash بین ۰/۵ تا ۱۶ کیلوبایت و بسته‌بندی بین ۶ تا ۳۲ پایه هستند.
ATMega	این سری دارای امکانات وسیع و دستورالعمل‌های قوی می‌باشد که دارای حافظه Flash بین ۴ تا ۵۱۲ کیلوبایت و بسته‌بندی بین ۸۲ تا 100 پایه هستند.
XMega	جدیدترین، پرسرعت‌ترین و قوی‌ترین نوع هستند که امکانات بیشتری نیز دارند که دارای حافظه Flash بین ۱۶ تا ۳۸۶ کیلوبایت و بسته‌بندی ۴۴، ۶۴ و ۱۰۰ پایه هستند.
AT90s	نوع توسعه‌یافته میکروکنترلر ۸۰۵۱ هستند که امکانات کمتری داشته و کمتر کاربرد دارند چرا که تقریبا منسوخ شده‌اند.

معماری و ساختار میکروکنترلرهای AVR

به طور کلی یک میکروکنترلر AVR از نظر ساختار داخلی دارای واحدهای زیر می‌باشد:

- واحد پردازش مرکزی (CPU)
- واحد کنترل کلاک ورودی
- واحد مقایسه‌کننده آنالوگ
- واحد حافظه برنامه Flash
- واحد کنترل وقفه
- واحد تایمر محافظ
- واحد حافظه داده EEPROM
- واحد تایمر و شمارنده
- واحد ارتباطات سریال SPI، TWI و USART
- واحد حافظه داده SRAM
- واحد مبدل آنالوگ به دیجیتال
- واحد برنامه‌ریزی و عیب‌یابی JTAG
- واحد ورودی و خروجی I/O

واحد پردازش مرکزی(CPU) که بر مبنای معماری RISC ساخته‌شده است، تمام فعالیت‌های میکروکنترلر را مدیریت کرده و تمام عملیات لازم بر روی داده‌ها را انجام می‌دهد. همچنین وظیفه ارتباط با حافظه‌ها و کنترل تجهیزات جانبی را بر عهده دارد. درون هسته AVR به تعداد ۳۲ ثبات همه منظوره، واحد محاسبه و منطق، واحد رمزگشایی دستور ID، ثبات دستورالعمل IR، ثبات شمارنده برنامه PC، ثبات وضعیت SREG و اشاره‌گر پشته SP قرار دارند.

واحد محاسبه و منطق ALU[1] در میکروکنترلر AVR به صورت مستقیم با تمام ۳۲ ثبات همه منظوره ارتباط دارد. این واحد وظیفه انجام کلیه اعمال محاسباتی و منطقی را با استفاده از ثبات‌های همه منظوره و در یک دوره تناوب از کلاک بر عهده دارد. به طور کلی عملکرد واحد محاسبه و منطق را می‌توان به سه قسمت اصلی ریاضیاتی، منطقی و توابع بیتی تقسیم‌بندی کرد در برخی از واحد محاسبه و منطق های توسعه‌یافته در معماری میکروکنترلرهای AVR از یک ضرب‌کننده با قابلیت ضرب اعداد بدون علامت و علامتدار و نیز اعداد اعشاری استفاده‌شده است.

ثبات‌ها نوعی از حافظه‌های موقت هستند که از فلیپ‌فلاپ‌ها ساخته می‌شوند و می‌توانند ۸ بیتی، ۱۶ بیتی، ۳۲ بیتی یا بیشتر باشند. از ثبات‌ها به صورت گسترده در تمام ساختار و واحدهای میکروکنترلرها استفاده می‌شود. میکروکنترلرهای AVR هشت بیتی هستند. مهمترین مسئله که در هنگام برنامه‌نویسی میکروکنترلرها با آن مواجه هستیم نحوه صحیح مقداردهی ثبات‌های آن میکروکنترلر می‌باشد

ثبات‌های واحد پردازش مرکزی

میکروکنترلرهای AVR دارای ۳۲ ثبات همه منظوره[2] (عمومی) هستند این ثبات‌ها قسمتی از حافظه SRAM میکروکنترلر می‌باشند. تعداد این ثبات‌ها ۳۲ عدد بوده و از R0 تا R31 شماره گذاری می‌شوند. هر ثبات دارای ۸ بیت است که به طور مستقیم با واحد محاسبه و منطق در ارتباط است. ثبات‌های R26 تا R31 به منظور آدرس‌دهی غیرمستقیم، در فضای حافظه داده و برنامه استفاده می‌شوند که به آن‌ها ثبات‌های اشاره‌گر می‌گویند و به ترتیب به صورت XL، XH، YL، YH، ZL و ZH نامگذاری می‌شوند. یک امکان برای دو ثبات جدا از هم فراهم‌شده است که بتوان توسط یک دستورالعمل خاص در یک سیکل کلاک به آن دسترسی داشت. نتیجه این معماری کارآیی بیشتر کدها تا ده برابر سریع‌تر از میکروکنترلرهای با معماری CISC است.

ثبات دستور[3] IR این ثبات که در هسته پردازنده قرار دارد کد دستورالعملی را که از حافظه برنامه FLASH خوانده‌شده و باید اجرا شود را در خود جای می‌دهد.

واحد رمزگشایی دستور[4] این واحد تشخیص می‌دهد کد واقع در IR مربوط به کدام دستورالعمل است و سیگنال‌های کنترلی لازم برای اجرای آن را صادر می‌نماید.

[1] Arithmetic Logic Unit
[2] General Purpose Registers
[3] Instruction Register
[4] Instruction Detector

ثبات شمارنده برنامه[1] PCاین ثبات در واقع شمارنده آدرس دستورالعمل‌های برنامه کاربر است و در هر مرحله به آدرس خانه بعدی حافظه فلش که باید اجرا شود اشاره می‌کند.

ثبات وضعیت[2] این ثبات ۸ بیتی واقع در هسته اصلی میکروکنترلر بوده و بیت‌های آن تحت تأثیر برخی عملیات واحد پردازش مرکزی فعال می‌شوند. این بیت‌ها به ترتیب از بیت صفر تا بیت ۷ به صورت زیر هستند: پرچم کری :C ، پرچم صفر :Z ، پرچم منفی :N ، پرچم سرریز :V ، پرچم علامت :S، پرچم نیم کری :H ، بیت فعال‌ساز وقفه سراسری :I و T.

ثبات اشاره‌گر پشته [3]SPپشته قسمتی از فضای حافظه داده SRAM است که جهت ذخیره اطلاعات و معمولا در اجرای دستور فراخوانی) (CALL و یا اجرای برنامه وقفه نیاز می‌باشد. اشاره‌گر پشته یا SP دارای ۱۶ بیت است و در ابتدا وقتی سیستم روشن می‌شود واحد پردازش مرکزی از محل استقرار حافظه پشته اطلاعی ندارد (پیش فرض SP=0). بنابراین باید آدرسی از حافظه SRAM که مربوط به فضای پشته است را به اطلاع سیستم رساند. با اجرای دستور PUSH R0 ، محتوای ثبات R0 با محتوای خانه‌ای از فضای پشته که SP به آن اشاره می‌کند جایگزین‌شده و بعد از آن SP یک واحد کاهش می‌یابد.

نکته: ثبات‌های درون هسته مرکزی، ثبات‌های سیستمی هستند که هر یک وظیفه مشخصی دارند و مقدار آنها دائما تغییر می‌کند. ثبات‌های فوق الذکر فقط برای شناخت بهتر AVR و درک عملکرد این واحد معرفی شدند و در عمل یک برنامه‌نویس نیازی به استفاده از آن ندارد.

معماری حافظه در AVR

حافظه داده SRAMاین حافظه جهت ذخیره‌سازی موقت اطلاعات در اختیار کاربر قرار می‌گیرد و با قطع تغذیه تراشه اطلاعات آن از بین می‌رود. این حافظه به طور مستقیم در اختیار واحد پردازش مرکزی نمی‌باشد، برای دستیابی به آن از یکی از ۳۲ ثبات عمومی به عنوان واسط استفاده می‌شود. یعنی جهت انجام هر عملیات داده نمی‌تواند مستقیما از SRAM به واحد محاسبه و منطق منتقل شود، بلکه باید ابتدا به یکی از ۳۲ ثبات عمومی برود و از آنجا به واحد محاسبه و منطق منتقل شود.

حافظه داده EEPROMاین حافظه که توسط ثبات‌های عمومی ۳۲ گانه با واحد پردازش مرکزی در ارتباط است، دائمی بوده و با قطع برق از بین نمی‌رود. از این حافظه می‌توان در مواقعی که مقدار یک داده در برنامه مهم بوده و نباید با قطع برق یا ریست شدن از بین برود استفاده کرد.

حافظه برنامه FLASH یک حافظه دائمی پرسرعت است که به دو قسمت تقسیم‌شده است: بخشی برای بوت شدن سیستم و بخشی برای برنامه کاربر مورد استفاده قرار می‌گیرد. فیوزبیت‌ها[4] و بیت‌های قفل[5] نیز در قسمت بوت لودر حافظه فلش وجود دارند. در قسمت دیگر این حافظه، برنامه کاربر وجود دارد که در هنگام کار میکروکنترلر خط به خط خوانده‌شده و اجرا می‌شود.

[1] Program Counter
[2] Status & Control Register
[3] Stack Pointer
[4] Fuse Bits
[5] Lock Bits

معرفی دیگر واحدهای میکروکنترلرهای AVR

اساسا ارتباط میکروکنترلر با محیط پیرامون به دو شکل موازی و سریال صورت می‌گیرد. واحد ورودی/خروجی به صورت موازی و واحد ارتباط سریال به صورت سریال با محیط پیرامون میکروکنترلر در ارتباط است.

واحد ورودی/خروجی (I/O)این واحد وظیفه ارتباط میکروکنترلر با محیط پیرامون را برقرار می‌کند. این واحد که با آن پورت نیز گفته می‌شود، دارای چند ثبات و بافر است و در نهایت به صورت پایه (Pin) از میکروکنترلر خارج می‌شود. در میکروکنترلرهای AVR با توجه به نوع و سری میکروکنترلر تعداد آنها بین یک تا 10 پورت متفاوت می‌باشد. پورت‌ها در AVR به صورت PORTA، PORTB، PORTC نامگذاری می‌شود. هر پورت ۸ بیت دارد که به صورت PORTX.0 تا PORTX.7 نامگذاری می‌شود.

واحد ارتباطات سریال برای تبادل داده با محیط خارجی میکروکنترلر به صورت سریال می‌باشد. در ارتباطات سریال پروتکل ارتباطی و سرعت ارسال پارامترهای مهمی هستند. پروتکل‌های ارتباطی سریال که توسط میکروکنترلرهای AVR پشتیبانی می‌شود عبارتند از:

پروتکل SPI	دارای سرعت بالا می‌باشد. از طریق این پورت میکروکنترلر را می‌توان پروگرام کرد
پروتکل USART	سرعت متوسط دارد. برای مسافت‌های طولانی و ماژول‌های ارتباطی مناسب است.
پروتکل TWI	پروتکل دو سیمه بیشتر برای ارتباط با عنصرهای جانبی سرعت پایین است.

واحد کنترل کلاک ورودی این واحد وظیفه تأمین کلاک میکروکنترلر را بر عهده دارد. منابع کلاک میکروکنترلر به دو دسته منابع کلاک خارجی و کلاک داخلی تقسیم می‌شود. همانطور که می‌دانید پالس کلاک یک پالس منظم و دارای فرکانس ثابت است که تمامی واحدهای میکروکنترلر با لبه‌های آن پالس کار می‌کنند. برای تولید پالس کلاک به دو چیز احتیاج می‌باشد یکی کریستال و دیگری اسیلاتور. خوشبختانه در این واحد هم اسیلاتور وجود دارد و هم کریستال اما بر حسب نیاز به داشتن فرکانس‌های مختلف احتیاج به اسیلاتور خارجی و یا کریستال خارجی نیز داریم. منابع کلاک در میکروکنترلرهای AVR به صورت زیر است:

منبع کلاک خارجی	از اتصال منبع کلاک با فرکانس مشخص به پایه X1 میکروکنترلر استفاده می‌شود. در این حالت پایه X2 آزاد است.
کریستال خارجی	از اتصال یک کریستال با فرکانس مشخص و دو عدد خازن بین پایه‌های X1 و X2 استفاده می‌شود. جنس بلور این کریستال‌ها معمولا کریستال کوارتز است. در این حالت از اسیلاتور داخل میکروکنترلر استفاده می‌شود.
اسیلاتور RC داخلی	اسیلاتور داخلی یک خازن و یک مقاومت است که با فرکانس مشخصی نوسان می‌کند. در این حالت هر دو پایه X1 و X2 آزاد است.

واحد تایمرها و شمارنده‌ها[1] در این واحد از یک سخت‌افزار خاص چندین استفاده متفاوت می‌شود. این سخت‌افزار خاص شامل چند ثبات و چند شمارنده هستند که کارآیی‌های متفاوتی دارند. کاربرد این واحد به عنوان تایمر، شمارنده، RTC و PWM می‌باشد. ثبات اصلی مورد استفاده در این واحد ثبات تایمر نام دارد. اگر یک پالس ورودی منظم (مانند تقسیمی از پالس کلاک میکروکنترلر) به این واحد اعمال کنیم، ثبات تایمر شروع به زیاد شدن کرده و در زمان مشخصی پر و سرریز می‌شود. بنابراین می‌توان با تنظیم مناسب پالس ورودی به این واحد زمان‌های مختلف را ایجاد کرد. اگر پالس ورودی تایمر را طوری تنظیم کنیم که ثبات تایمر هر یک ثانیه سرریز شود، در این صورت RTC یا زمان‌سنج حقیقی ساخته می‌شود. در زمان استفاده از RTC به کریستال که به کریستال ساعت معروف است نیاز داریم. اگر ثبات تایمر را قبل از سرریز شدن ریست کنیم، در این صورت می‌توان با اعمال پالس خروجی از میکروکنترلر یک PWM یا مدولاسیون پهنای عرض پالس را روی

[1] Timers & Counters

یکی از پایه‌های میکروکنترلر داشت. اما اگر پالس ورودی به این واحد نامنظم باشد (مانند پالسی که از بیرون به پایه میکروکنترلر اعمال می‌شود) در این صورت یک شمارنده تعداد پالسهای ورودی روی ثبات تایمر ساخته می‌شود.

واحد تایمر محافظ[1] این واحد متشکل از یک ثبات تایمر و یک اسیلاتور است که با فعال‌سازی آن ثبات تایمر شروع به افزایش می‌کند تا اینکه سرریز شود. با سرریز شدن ثبات تایمر محافظ میکروکنترلر ریست می‌شود. کاربرد این واحد جهت جلوگیری از هنگ کردن میکروکنترلر است. در پروژه‌های دارای اهمیت میکروکنترلر نباید هنگ کند؛ زیرا ممکن است خطرات و عواقب بدی داشته باشد. این واحد وظیفه دارد در صورت هنگ نمودن میکروکنترلر بلافاصله آن را ریست کند تا در کسری از ثانیه میکروکنترلر دوباره شروع به کار نماید.

واحد کنترل وقفه[2]. علت بوجود آمدن واحد کنترل وقفه این است که باعث می‌شود پردازنده کمتر درگیر شود. در حالتی وقفه وجود نداشته باشد، پردازنده مجبور است طی فواصل زمانی مشخصی چندین بار به واحد موردنظر سرکشی کرده و بررسی کند که داده‌ی خواسته‌شده از آن واحد آماده است یا خیر که اغلب آماده نبوده و وقت پردازنده تلف می‌شود. برای مثال فرض کنید که یک صفحه کلید به ورودی میکروکنترلر متصل است ؛ در حالت سرکشی[3] پردازنده باید یکی یکی کلیدهای صفحه کلید را در فواصل زمانی مشخص سرکشی کند تا در صورت فشرده شدن یک کلید پردازش موردنظر را انجام دهد، اما در حالتی که واحد کنترل وقفه فعال باشد، پردازنده آزاد است و تا زمانی که کلیدی زده نشده است پردازنده به صفحه کلید کاری ندارد. سپس در صورتی که کلیدی زده شود واحد کنترل وقفه یک سیگنال وقفه به پردازنده مبنی بر اینکه ورودی آمده است ارسال می‌کند تا پردازنده برای یک لحظه کار خود را رها کرده و به صفحه کلید سرویس می‌دهد، کلیدی که زده‌شده را شناسایی کرده و پردازش موردنظر را روی آن انجام می‌دهد.

واحد ارتباطیJTAG[4] یک پروتکل ارتباطی بر روی دستگاه‌ها می‌باشد که این توانایی را ایجاد می‌کند که بتوان برنامه نوشته‌شده موجود در میکروکنترلر را خط به خط اجرا نمود تا در صورت وجود اشکال در برنامه‌نویسی آن را برطرف و عیب‌یابی[5] کرد. همچنین می‌توان توسط JTAG حافظه Flash، EEPROM، فیوزبیت‌ها و بیت‌های قفل را برنامه‌ریزی یا پروگرام[6] کرد.

واحد مبدل آنالوگ به دیجیتال (ADC) کمیت‌های آنالوگ برای پردازش توسط میکروکنترلر را متناسب با ولتاژ ورودی تبدیل به کد دیجیتال می‌کند. مسائلی که در هنگام کار با این واحد درگیر آن هستیم یکی سرعت نمونه‌برداری و دیگری ولتاژ مرجع VREF است. ولتاژ مرجع در این واحد به عنوان مرجعی برای سنجش ولتاژ آنالوگ ورودی به کار می‌رود، به صورتی که بازه‌ی مجاز ولتاژ ورودی بین صفر تا VREF است. همچنین سرعت نمونه‌برداری مسئله‌ی مهمی است که در بروزرسانی سریعتر اطلاعات نقش دارد.

واحد مقایسه‌کننده آنالوگ یکی دیگر از امکانات موجود در میکروکنترلرهای AVR می‌باشد که با استفاده از آن می‌توان دو موج آنالوگ را با هم مقایسه کرد. عملکرد این قسمت مشابه عملکرد آپ‌امپ[7] در مد مقایسه‌کننده می‌باشد، با این تفاوت که در صورتی که ولتاژ پایه مثبت از پایه منفی بیشتر باشد، خروجی مقایسه‌کننده یک می‌شود.

انواع زبان‌های برنامه‌نویسی و کامپایلرهای AVR

هر کدام از زبان‌های برنامه‌نویسی نظیر اسمبلی، C، C++، بیسیک و پاسکال یک کامپایلر مخصوص به خود برای برنامه‌نویسی و کامپایل میکروکنترلرها دارند. خروجی کامپایلر از طریق سخت‌افزاری به نام پروگرامر روی میکروکنترلر برنامه‌ریزی می‌شود. زبان برنامه‌نویسی C به عنوان یکی از بهترین و منعطف‌ترین زبان‌های برنامه‌نویسی در این زمینه است که می‌توان برای برنامه‌ریزی هر نوع میکروکنترلری به کار برد. برای میکروکنترلرهای AVR از کامپایلرهای معروف می‌توان به کدویژن، AVR Studio و Bascom AVR اشاره کرد.

---

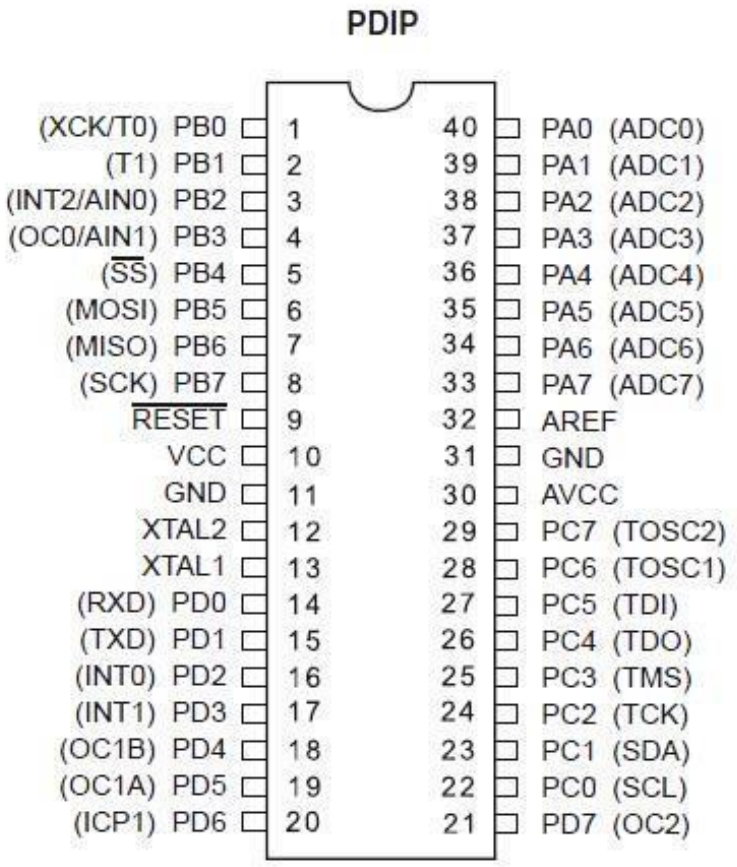
[1] Whatchdog
[2] Interrupt
[3] Polling
[4] Joint Test Access Group
[5] Debug
[6] Program
[7] OpAmp

خصوصیات، ویژگی‌ها و عملکرد ATMega32

میکروکنترلر ATMega32 بر اساس معماری RISC بهبودیافته می‌باشد. با اجرای دستورات قدرتمند در یک سیکل کلاک این میکروکنترلر سرعتی معادل 1MIPS[1] در هر MHz را فراهم می‌کند. یعنی زمانی که فرکانس کاری میکروکنترلر یک مگاهرتز است، میکروکنترلر می‌تواند یک میلیون دستورالعمل در هر ثانیه انجام دهد. خصوصیات و ویژگی‌های این میکروکنترلر به شرح زیر است:

- کارآیی بالا، توان مصرفی کم، یک میکروکنترلر ۸ بیتی از خانواده AVR با تکنولوژی CMOS.
- دارای ضرب‌کننده جداگانه داخلی که در دو کلاک سیکل عمل ضرب را انجام می‌دهد.
- دارای حافظه غیرفرار برای برنامه و داده: ۳۲ کیلوبایت حافظه فلش داخلی قابل برنامه‌ریزی؛ ۱۰۲۴ بایت حافظه EEPROM؛ ۲ کیلوبایت حافظه SRAM داخلی؛ دارای قفل برنامه برای حفاظت نرم‌افزاری از حافظه.
- قابلیت ارتباط به صورت JTAG تحت استاندارد (IEEE std. 1149.1) جهت عیب‌یابی و برنامه‌ریزی حافظه فلش، EEPROM، فیوزبیت‌ها و بیت‌های قفل.
- خصوصیات ویژه میکروکنترلر: ریست خودکار میکروکنترلر در هنگام روشن شدن، شناسایی ولتاژ تغذیه ورودی قابل برنامه‌ریزی، دارای اسیلاتور RC داخلی کالیبره شده، منابع وقفه داخلی و خارجی، دارای ۶ حالت خواب[2] برای کاهش توان مصرفی.
- ولتاژهای کاری ۷ - تا ۵٫۵ ولت در ATMega32L و ۵ - تا ۵٫۵ ولت در ATMega32.
- فرکانس‌های کاری تا ۸ MHz برای ATMega32L و تا 16MHz برای ATMega32.

تشریح عملکرد پایه‌ها در ATMEGA16/32



[1] Millions Instruction Per Second
[2] Sleep Mode

بسته‌بندی PDIP این میکروکنترلر ۴۰ پایه دارد. دارای چهار پورت ورودی/خروجی PA تا PD ۸ بیتی است.

شماره پایه	نام پایه	عملکرد پایه
۱و۲	T0 , T1	مربوط به ورودی کلاک خارجی تایمر صفر و تایمر یک می‌باشد.
۳و۴	AIN0, AIN1	به ترتیب پایه مثبت و منفی واحد مقایسه‌کننده آنالوگ می‌باشد.
۵-۸	MISO، MOSI، SCK، SS	این پایه‌ها مربوط به ارتباط spi واحد ارتباط سریال می‌باشد.
۹	RESET	پایه ریست میکروکنترلر است که تا زمانی که به منطق صفر وصل شود میکروکنترلر در حالت ریست می‌ماند.
۱۰	VCC	پایه مثبت تغذیه دیجیتال
۱۱	GND	پایه تغذیه منفی یا زمین دیجیتال
۱۲-۱۳	XT1, XT2	ورودی‌های واحد کلاک میکروکنترلر کنترلر است.
۱۴-۱۵	TXD, RXD	به ترتیب گیرنده و فرستنده داده ارتباط USART واحد ارتباط سریال می‌باشد.
۱۶-۱۷ و ۳	INT0, INT1, INT2	به ترتیب مربوط به وقفه‌های خارجی صفر، یک و دو است.
۴،۱۸،۱۹،۲۱	OC0, OC1A, OC1B, OC2	مربوط به خروجی‌های کانال PWM واحد تایمر/شمارنده است.
۲۲-۲۳	SDA ،SCL	مربوط به پروتکل دوسیمه واحد ارتباط سریال می‌باشد.
۲۴-۲۷	TDI ،TDO ،TMS ،TCK	مربوط به ارتباط JTAG می‌باشد.
۲۸-۲۹	TOSC1- TOSC2	مربوط به واحد RTC می‌باشد. در زمان استفاده از RTC به این دو پایه کریستال وصل می‌شود.
۳۰	AVCC	ولتاژ تغذیه آنالوگ واحد ADC است.
۳۱	GND	زمین آنالوگ واحد ADC است.
۳۲	AREF	پایه مربوط به ولتاژ مرجع قسمت ADC می‌باشد.
۳۳-۴۰	ADC0- ADC7	کانال‌های ورودی آنالوگ مربوط به واحد ADC می‌باشد.

نکته: بعضی از پایه‌ها دو یا سه عملکرد دارند که همزمان نمی‌توان از چند عملکرد استفاده کرد و در آن واحد تنها یک کاربرد از آنها استفاده می‌گردد. مثلا اگر از پورت A به عنوان ورودی/خروجی موازی استفاده شود دیگر نمی‌توان از قابلیت واحد ADC میکروکنترلر استفاده نمود.

معماری و ساختار داخلی میکروکنترلر ATMega32

واحد پردازش مرکزی دارای ۱۳۱ دستورالعمل می‌باشد. ثبات شمارنده برنامه PC در این میکروکنترلر ۱۴ بیت می‌باشد.

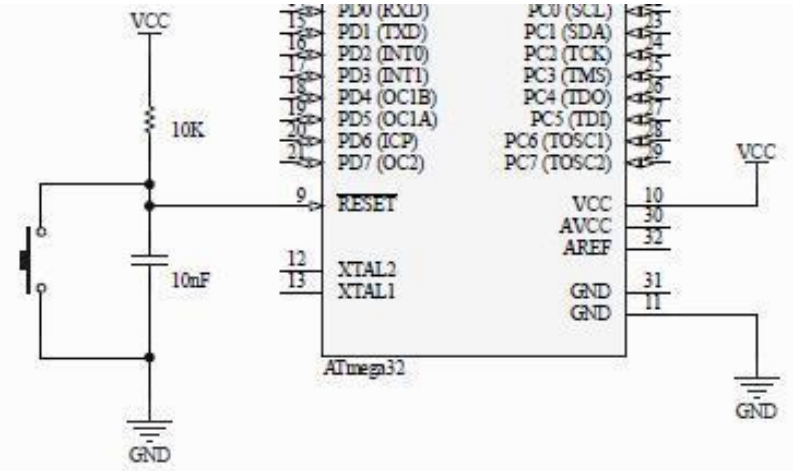
واحد حافظه برنامه Flash ۳۲ کیلوبایت برای ذخیره برنامه می‌باشد (0000H – 3FFFH) . برنامه نوشته‌شده به زبان C بعد از کامپایل شدن درون این واحد پروگرام می‌شود تا توسط واحد پردازش مرکزی خط به خط خوانده و اجرا شود. از آن جایی که همه دستورالعمل‌ها در میکروکنترلرهای AVR تعداد ۱۶ یا ۳۲ بیت دارند، حافظه فلش در این میکروکنترلر به صورت 16Kx16 سازماندهی‌شده است. از این رو، ثبات PC دارای ۱۴ بیت عرض می‌باشد تا بتواند هر 16K خانه حافظه برنامه را آدرس‌دهی نماید. از حافظه برنامه Flash در صورتی که حجم برنامه کم بوده باشد و فضای خالی موجود باشد، می‌توان برای ذخیره دائمی اطلاعات به عنوان حافظه داده نیز استفاده نمود.

واحد حافظه داده SRAM ۲ کیلوبایت دارد. به طوری که ۳۲ خانه اول در آن مربوط به ثبات‌های عمومی که در حقیقت جزو واحد پردازش مرکزی هستند، اختصاص دارد. ۶۴ خانه بعدی حافظه SRAM مربوط به نگهداری و ذخیره کلیه ثبات‌ها می‌باشد. ثبات‌های تمامی واحدها (واحد ورودی/خروجی، واحدهای سریال، تایمرها و شمارنده‌ها و غیره) در این قسمت قرار دارد. بقیه خانه‌های حافظه همان واحد SRAM اصلی می‌باشد که جهت نگهداری اطلاعات و متغیرهای برنامه که یا دائما تغییر می‌کنند می‌باشد. فقط با قطع تغذیه محتویات SRAM از بین می‌رود و در صورتی که میکروکنترلر را Reset دستی یا خارجی کنیم می‌توان محتوای حافظه SRAM را حفظ کرد.

واحد حافظه داده EEPROM یک کیلوبایت است و جزء حافظه‌های ماندگار می‌باشد که در صورت قطع تغذیه میکروکنترلر پاک نمی‌گردد و میکروکنترلر در هر زمان می‌تواند اطلاعاتی را در این حافظه بنویسد و یا اطلاعاتی را از آن بخواند، خواندن و نوشتن در حافظه EEPROM زمان تأخیر طولانی‌تری از حافظه‌های SRAM و Flash دارد.

حداقل سخت‌افزار راه‌اندازی میکروکنترلر ATMega32

برای راه‌اندازی این میکروکنترلر کافی است پایه‌های تغذیه دیجیتال ۱۰ و ۱۱ را به منبع تغذیه با ولتاژ مناسب وصل کرده و همچنین بهتر است پایه RESET را توسط یک مقاومت مناسب ۱۰ کیلواهم به مثبت تغذیه وصل نماییم. برای اینکه بتوانیم به صورت دستی میکروکنترلر را هر زمان که خواستیم ریست کنیم، احتیاج به یک سوئیچ[1] داریم. مدار کامل‌شده به صورت زیر است. علت وجود خازن در مدار زیر جلوگیری از نوسانات ولتاژ در لحظه اتصال کلید و همچنین برای آرام آرام زیاد شدن ولتاژ در هنگام شروع به کار مجدد میکروکنترلر است. مقدار خازن و مقاومت میزان زمان افزایش ولتاژ تا Vcc در هنگام وصل منبع تغذیه را تعیین می‌کند که بهتر است 10k و 10n انتخاب شود. پایه‌های ۳۰، ۳۱ و ۳۲ به ترتیب ولتاژ تغذیه آنالوگ، زمین آنالوگ و ولتاژ مرجع برای واحد ADC می‌باشند و تنها در هنگام استفاده از واحد ADC به تغذیه متصل می‌شود.



[1] Push Button

ثبات‌های واحدهای میکروکنترلر ATMega32

هر یک از واحدهای ATMega32 به جز واحد پردازش مرکزی، ثبات‌هایی دارد که تنظیمات واحد موردنظر را مشخص می‌کند. برنامه‌نویس باید ثبات‌های مربوط به هر واحد را شناخته و آنها را بسته به پروژه موردنظر به درستی مقداردهی نماید.

ثبات‌های واحد ورودی/خروجی در AVR

تمامی میکروکنترلرهای AVR دارای واحد ورودی/خروجی I/O می‌باشند. در ATMega32 چهار پورت ورودی/خروجی وجود دارد که برای هر یک پین‌هایی خروجی درنظر گرفته‌شده است. هر یک از این چهار پورت ورودی/خروجی دارای سه ثبات به شرح زیر است. در نتیجه مجموعا ۱۲ ثبات ۸ بیتی برای واحد ورودی/خروجی وجود دارد که همگی درون SRAM می‌باشند.

DDRX: یک ثبات ۸ بیتی برای تعیین جهت ورودی یا خروجی بودن هر یک از پایه‌ها است. هر بیت از این ثبات که یک باشد، جهت پایه متناظر با آن به عنوان خروجی و هر بیت که صفر باشد، پایه متناظر با آن ورودی می‌شود. در حالت پیش‌فرض این ثبات صفر است.

PORTX: در صورت انتخاب شدن ثبات DDRX به عنوان خروجی از این ثبات برای یک یا صفر کردن منطق پایه خروجی استفاده می‌شود. هر بیت از این ثبات که یک باشد پایه متناظر با آن، منطق یک را خارج می‌کند و هر بیت متناظر که صفر باشد، منطق صفر در خروجی ظاهر می‌گردد. درحالت پیش‌فرض همه بیت‌های این ثبات صفر هستند.

PINX: در صورت انتخاب شدن ثبات DDRX به عنوان ورودی از این ثبات برای خواندن منطق پایه‌های پورت استفاده می‌شود. هر منطقی که یکی از پایه‌های پورت داشته باشد، بیت موردنظر در ثبات PINX همان منطق را به خود می‌گیرد. در حالت پیش‌فرض همه بیت‌های این ثبات صفر هستند

نکته : به جای X در بالا نام پورت موردنظر قرار می‌گیرد.

نکته : نام تمامی ثبات‌ها از جمله ثبات‌های فوق با حروف **بزرگ** نوشته می‌شوند. بقیه حروف در برنامه همگی کوچک نوشته می‌شوند.

نکته : از 0b برای مقداردهی به ثبات‌ها در مبنای باینری و از 0x برای مقداردهی به ثبات‌ها در مبنای ۱۶ (هگزا) به کار می‌رود.

نکته : از ثبات PIN تنها در حالتی که پایه ورودی باشد (مانند وقتی که کلیدی به عنوان ورودی به پایه وصل باشد)، استفاده می‌شود.

مثال عملی: برنامه‌ای بنویسید که LED موجود روی PA.0 را چهار بار در ثانیه به صورت چشمک‌زن روشن و خاموش کند. سپس آن را در نرم‌افزار پروتئوس شبیه‌سازی کرده و پس از اطمینان از عملکرد صحیح برنامه توسط نرم‌افزار کدویژن روی میکروکنترلر پیاده‌سازی نمایید.

#include <mega32.h>
#include <delay.h>
void main(void)
{
DDRA.0=1;
while(1){
PORTA.0=1;
delay_ms(250);
PORTA.0=0;
delay_ms(250);
}}
	توضیح برنامه: در خط اول ابتدا کتابخانه مربوط به ATMega32 را اضافه می‌کنیم. در خط دوم کتابخانه مربوط به تابع تأخیر زمانی (delay.h) را اضافه می‌کنیم. تنها زمانی که این کتابخانه به برنامه اضافه شود می‌توان از تابع delay_ms برای تأخیر زمانی در برنامه استفاده کرد. در خط پنجم، پورت) PA.0 بیت صفرم پورت (A به عنوان خروجی تنظیم می‌شود. در خط هفتم، برنامه منطق یک را به پورت PA.0 که خروجی‌شده بود، تخصصیص می‌دهد. در نتیجه LED در این حالت روشن می‌شود. در خط هشتم، برنامه هیچ کاری انجام نداده و۲۵۰ میلی‌ثانیه منتظر می‌ماند سپس به خط بعدی می‌رود. در نتیجه این خط LED به مدت۲۵۰میلی‌ثانیه روشن می‌ماند. در خط نهم، برنامه منطق صفر را به پورت PA.0 که خروجی‌شده بود، تخصصیص می‌دهد. در نتیجه LED در این حالت خاموش می‌شود در خط دهم، برنامه هیچ کاری انجام نداده و۲۵۰میلی‌ثانیه منتظر می‌ماند سپس به خط بعدی می‌رود. در نتیجه این خط LED به مدت۲۵۰میلی‌ثانیه خاموش می‌ماند. چون برنامه در حلقه نامتناهی نوشته‌شده است، با اجرا شدن برنامه LED موجود روی PA.0 دائما روشن و خاموش می‌شود و چون زمان تأخیر۲۵۰میلی‌ثانیه است این کار ۴ بار در ثانیه انجام می‌شود.

برنامه‌نویسی، شبیه‌سازی و پروگرام کردن میکروکنترلر

مهمترین بخش میکروکنترلر، ثبات‌ها هستند. ثبات‌ها کنترل و تنظیمات تمام بخش‌های میکروکنترلر را بر عهده دارند و باید به خوبی با نحوه عملکرد آنها آشنا شد. در ATMega32 به تعداد ۶۴ ثبات برای کنترل مجموعه سیستم میکروکنترلر وجود دارد که در قسمت SRAM حافظه قرار دارد. با تنظیم و مقداردهی به این ثبات‌ها در برنامه می‌توان برنامه‌های مختلفی را راه‌اندازی و اجرا نمود. بعد از ثبات‌ها، فیوزبیت‌ها نیز بخش مهمی از میکروکنترلر می‌باشند که همانند ثبات‌ها کنترل و تنظیمات بخش‌های دیگری از میکروکنترلر نظیر فرکانس کلاک میکروکنترلر بر عهده دارد.

معرفی کلی نرم‌افزارهای پروتئوس و کدویژن

پروتئوس (Proteus) نرم‌افزاری برای شبیه‌سازی مدارات الکترونیک، بخصوص مدارات مبتنی بر میکروکنترلر می‌باشد. اصلی‌ترین کار این نرم‌افزار شبیه‌سازی است اما قابلیت استفاده برای کشیدن بردهای PCB را نیز فراهم کرده است و برای این کار هم محیطی مجزا از شبیه‌سازی در نظر گرفته است. کتابخانه‌های بسیاری از قطعات الکترونیک جهت طراحی و شبیه‌سازی مدارات الکترونیکی در این نرم‌افزار موجود است. کاربردهای این نرم‌افزار عبارتند از:

- شبیه‌سازی مدارات آنالوگ و دیجیتال.
- تست و آنالیز مدارات با استفاده از ابزار اندازه‌گیری مجازی مانند اسیلوسکوپ، مولتی متر، فانکشن ژنراتور.
- شبیه‌سازی تقریبا اکثر مدارات با میکروکنترلر کنترلرهای PIC، AVR و برخی از میکروکنترلرهای ARM.
- امکان ایجاد و ویرایش قطعات الکترونیکی.
- طراحی بردهای PCB یک تا ۱۶ لایه.

کدویژن (Code vision) یک نرم‌افزار کامپایلر زبان C است که برای برنامه‌نویسی، برنامه‌ریزی (پروگرام) و عیب‌یابی (واسط JTAG) کلیه میکروکنترلرهای AVR می‌باشد. این نرم‌افزار که دارای محیط برنامه‌نویسی توسعه‌یافته نیز می‌باشد، بیشتر به علت تولید کدهای اتوماتیک توسط ساختار کدویزارد است. این قابلیت دسترسی راحت به تنظیمات ثبات‌های میکروکنترلرهای AVR را فراهم می‌کند

پس از نصب نرم‌افزار پروتئوس، آن را اجرا کنید. زمانی که نرم‌افزار باز شد، بر روی آیکون آبی رنگ ISIS موجود در نوار ابزار بالایی کلیک کنید تا Schematic capture برای رسم مدار باز شود. برای انتخاب عنصرهای مداری باید روی آیکون P کلیک کنید تا صفحه جدیدی بنام Pick Devices باز شود. در این صفحه هر عنصری را نیاز داشته باشید تایپ کرده و سپس روی نام قطعه دوبار کلیک کنید تا قابل استفاده گردد. برای این مثال یک ATMega32 و یک LED نیاز داریم. نام آنها را تایپ کرده و از قسمت سمت چپ روی نام آنها دوبار کلیک می‌کنیم. در نهایت پنجره Pick Devices با کلیک بر روی Ok بسته می‌شود.

با بازگشت به صفحه اصلی مشاهده می‌کنید که زیر آیکون P قطعاتی که انتخاب شده، آورده شده است. با کلیک بر روی آنها میکروکنترلر و LED را درون کادر آبی رنگ صفحه اصلی در محل مناسب خود قرار داده و مدار را تکمیل می‌کنیم. به یک زمین (Ground) نیز احتیاج داریم که آن را با کلیک بر روی آیکون Terminals Mode موجود در نوار ابزار سمت چپ و سپس کلیک بر روی GROUND انتخاب کرده و در جای مناسب خود قرار می‌دهیم. در نهایت، نوبت به سیم‌کشی مدار می‌رسد. زمانی که نشانگر ماوس را در محل سیم‌کشی روی پایه‌های LED یا میکروکنترلر می‌برید، مداد سبز رنگی ظاهر می‌شود، در همین حال کلیک کنید و سیم‌کشی مدار را تکمیل نمایید. بعد از تکمیل مدار آن را با نام مناسب در یک پوشه جدید ذخیره نمایید. برای اینکه برنامه را بتوان بر روی تراشه ریخته و سپس اجرا کرد می‌بایست ابتدا میکروکنترلر را با یک فایل اجرایی که خروجی کدویژن است، برنامه‌ریزی کرد و سپس طرح را شبیه‌سازی نمود.

لازم به ذکر است، در هنگام شبیه‌سازی و کشیدن قطعات مورد نیاز آزمایش در صفحه‌ی اصلی آنچه مهم است، نظم و در هم گره نخوردن سیم-هایی است که قطعات را به هم متصل می‌کند. پس در هنگام اتصال قطعات از به هم گره نخوردن سیم‌های اتصالی چندین قطعه به هم مطمئن شوید تا شبیه‌سازی دچار خطا نشود.

پس از نصب و اجرای نرم‌افزار کدویژن برای شروع پروژه ی جدید باید مراحل زیر با دقت و به ترتیب طی شود:

۱. ابتدا ممکن است آخرین پروژه‌ای که کار کرده اید باز باشد آن را از مسیر File و سپس Close All ببندید.

۲. از منوی File گزینه New را انتخاب کرده و سپس در پنجره باز شده، Source را انتخاب و Ok کنید.

۳. فایل untitled.c باز می‌شود. در این مرحله می‌بایست از منوی File و سپس Save as آن را با نام مناسب در همان پوشه‌ای که فایل پروتئوس قرار دارد، ذخیره کنید.

۴. از منوی File گزینه New را دوباره انتخاب کرده اما این بار در پنجره باز شده، Project را انتخاب و Ok کنید.

۵. پنجره‌ای مبنی بر اینکه آیا می‌خواهید از کدویزارد استفاده کنید یا خیر؟ باز می‌شود آن را No کنید.

۶. پروژه را در همان مسیر قبلی با نام مناسب ذخیره کنید.

۷. پنجره Project Configure باز می‌شود که در این مرحله می‌بایست فایل ذخیره‌شده با پسوند. C در مرحله ۳ را با زدن دکمه Add به پروژه اضافه نمود

در این مرحله به سربرگ C Compiler رفته و در قسمت Chip نوع تراشه را تعیین می‌کنیم و در قسمت Clock فرکانس کلاک را روی فرکانس کاری میکروکنترلر تنظیم می‌کنیم (برای ATMega32 کلاک در حالت پیش‌فرض روی 1MHZ باید قرار گیرد .) در نهایت پنجره Project Configure Ok را انتخاب کنید. به این ترتیب، پروژه جدید ساخته‌شده است.

حالا باید برنامه دلخواه به زبان C را در این مرحله نوشت. بعد از نوشتن برنامه در محیط کدویژن می‌بایست از منوی Project گزینه Build را انتخاب کنید تا برنامه کامپایل و ساخته شود. در صورت بروز خطا باید ابتدا آنها را برطرف کرده تا برنامه ساخته شود. ساخت برنامه بدین معنی است که فایلی با پسوند Hex ساخته می‌شود که این فایل کد ماشین دستورات اسمبلی میکروکنترلر است و می‌توان آن را روی میکروکنترلر پروگرام کرد یا برای شبیه‌سازی در پروتئوس به کار برد. برنامه‌ای که نوشته می‌شود باید طوری نوشته شود که وقتی روی تراشه پروگرام شد دائما اجرا شود و هیچگاه متوقف نشود. راه حل این مسئله قرار دادن کدهای برنامه درون یک حلقه نامتناهی است. این عمل باعث می‌شود تا میکروکنترلر هیچگاه متوقف نشود و بطور مداوم عملکرد مورد انتظار را اجرا کند.

در نرم‌افزار پروتئوس، روی میکروکنترلر دوبار کلیک کرده تا پنجره Edit Component باز شود. در این پنجره در قسمت Program File روی آیکون پوشه (Browse) کلیک کنید تا پنجره انتخاب فایل باز شود. حالا می‌بایست به مسیر برنامه‌ای که در کدویژن نوشتید بروید و در آنجا داخل پوشه Exe فایل خروجی کدویژن با پسوند Hex را انتخاب کنید.

بعد از انتخاب فایلhex ، پنجره Edit Component را Ok می‌کنیم. با این کار برنامه نوشته‌شده به زبان C در نرم‌افزار کدویژن به داخل تراشه میکروکنترلر در نرم‌افزار پروتئوس ریخته می‌شود. حال برای شبیه‌سازی مدار روی دکمه Play پایین صفحه کلیک کرده تا شبیه‌سازی آغاز و مدار Run شود.

پروگرام کردن میکروکنترلر توسط نرم‌افزار کدویژن

بعد از وصل کردن و روشن کردن پروگرامر خود به پورت آن و نصب فایل درایور پروگرامر مراحل زیر برای پروگرام کردن را در نرم‌افزار کدویژن اجرا کنید:

۱. ابتدا از منوی Setting، گزینه Programmer را انتخاب می‌کنیم.
۲. از لیست موجود می‌بایست نوع پروگرامر را انتخاب کنید.
۳. در قسمت Communication Port می‌بایست آن پورتی که پروگرامر به آن وصل است را تنظیم کرد. اگر پورتی که پروگرامر به آن وصل است را نمی دانید کافیست به قسمت Device Manager در Control Panel مراجعه کرده و سپس در قسمت Port&LPT نام پروگرامر و پورت آن مشخص است.
۴. میکروکنترلر را روی پروگرامر در جای مناسب و در جهت صحیح قرار می‌دهیم.
۵. از منوی Tools، گزینه Chip Programmer را انتخاب می‌کنیم.
۶. تیک گزینه Program Fuse Bit را برمی‌داریم.
۷. با زدن دکمه Program All، پروگرام آغاز می‌شود.

در پیغامی که هنگام پروگرام ظاهر می‌شود از ما می‌خواهد تا فایلی برای ریختن داده درون EEPROM معرفی کنید، چون ما فایلی نداریم پس گزینه Cancel را انتخاب می‌کنیم.

نکته مهم: دقت کنید که حتما تیک گزینه Program Fuse Bit زده نشده باشد تا فیوزبیت‌های میکروکنترلر در حالت default بدون تغییر بماند.

نکته: در نرم‌افزار پروتئوس نیازی به قرار دادن مقاومت پول آپ برای پایه ریست درون نرم‌افزار نیست، اما در عمل و به هنگام پیاده‌سازی بهتر است پایه ریست پول آپ شود و برای ایجاد قابلیت ریست کردن دستی، یک کلید و خازن هم قرار می‌دهیم.

کدویزارد

هر یک از واحدهای ATMega32 به جز واحد پردازش مرکزی، ثبات‌هایی دارد که تنظیمات واحد موردنظر را مشخص می‌کند. برنامه‌نویس باید ثبات‌های مربوط به هر واحد را شناخته و آنها را بسته به پروژه موردنظر به درستی مقداردهی نماید. برای اینکه این مقداردهی راحت تر، سریعتر و با دقت بیشتر صورت گیرد از ابزار کدویزارد استفاده می‌شود. در حقیقت کدویزارد ابزار کاملی برای برنامه‌نویسی کلیه میکروکنترلرهای AVR است که به همراه نرم افزار کدویژن ارائه‌شده و بخشی از این نرم‌افزار می‌باشد. این ابزار که به جادوگر کد معروف است به شما این امکان را می‌دهد که با انتخاب کردن مدل میکروکنترلر، فرکانس کاری و تنظیم و پیکربندی کلیه واحدهای میکروکنترلر از میان گزینه‌های آماده موجود در ابزار، کدهای مربوط به تنظیمات اولیه ثبات‌ها به زبان C را به طور اتوماتیک تولید کند.

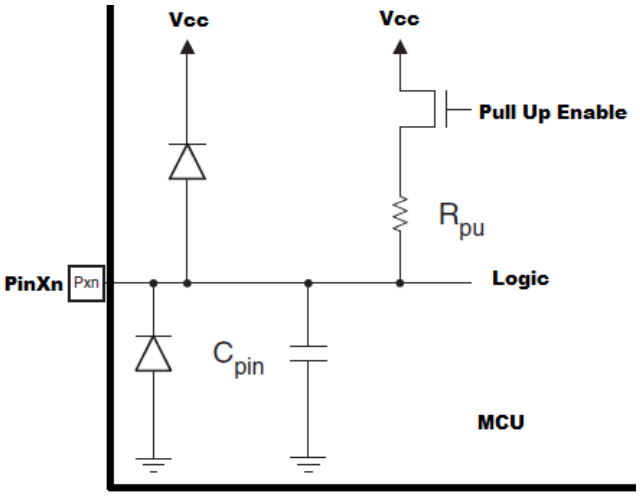
پورت‌های ورودی/خروجی یکی از مهم‌ترین واحدهای هر میکروکنترلر می‌باشد که به منظور استفاده عمومی و کاربردهای همه منظوره طراحی‌شده است. به طور کلی نکات زیر در مورد واحد I/O در میکروکنترلرهای AVRوجود دارد:

۱- هر یک از پایه‌های میکروکنترلر ممکن است چندین عملکرد از جمله پورت ورودی/خروجی باشد که به صورت همزمان نمی توان از آنها استفاده نمود. بنابراین در صورت استفاده از واحد I/O در یک پایه، فقط به عنوان پورت ورودی یا خروجی همه منظوره استفاده می‌شود.
۲- در صورت استفاده از یک پایه به عنوان **خروجی**، پایه موردنظر می‌تواند دو حالت (منطق یک یا منطق صفر) داشته باشد. در صورتی که آن پایه منطق یک داشته باشد ولتاژ آن پایه حداکثر Vcc می‌گردد و جریانی از داخل میکروکنترلر به بیرون کشیده می‌شود که به آن جریان Source گویند. زمانی که پایه منطق صفر داشته باشد، ولتاژ آن پایه حداقل صفر ولت می‌گرد و جریانی از بیرون به آن وارد می‌شود که به آن جریان sink گویند.

۳- حداکثر جریان Source و Sink در میکروکنترلر ATMega32 برای وقتی که Vcc=5v و دما ۲۵ درجه باشد، به مقدار60 میلی‌آمپر می‌رسد. با افزایش جریان Source، ولتاژ منطق یک پایه شروع با کاهش از مقدار Vcc می‌کند. هر چه جریان‌دهی پایه بیشتر باشد افت ولتاژ پایه از مقدار Vcc نیز بیشتر می‌شود. با افزایش جریان‌دهی با کاهش ولتاژ از ۵ ولت تا ۳ ولت مواجه هستیم. برای جریان Source هم به همین صورت اگر جریانی که به پایه تراشه وارد می‌شود افزایش یابد ولتاژ آن از حالت ایده آل (صفر ولت) افزایش پیدا کرده به طوری که برای Vcc=5v و دمای ۲۵ درجه به ۲ ولت نیز می‌رسد.

۴- در صورت استفاده از یک پایه به عنوان **ورودی**، پایه موردنظر می‌تواند یکی از سه حالت زیر را داشته باشد:

- حالت اتصال پایه به منطق ولتاژی صفر یا یک از بیرون.
- حالت دارای مقاومت Pull Up داخلی: در این حالت یک مقاومت Rpu از داخل میکروکنترلر تنها به صورت پول آپ می‌تواند وصل شود. شکل زیر این مقاومت را به همراه دیگر اجزای داخلی هر پایه نشان می‌دهد. هر پایه واحد I/O دارای دیودهای هرزگرد، خازن و مقاومت پول آپ می‌باشد.



- ✓ مقدار مقاومت پول آپ به طور تقریبی برابر ۵ کیلو اهم است.
- ✓ پایه‌هایی که به هیچ جایی متصل نیستند، به صورت بدون اتصال، مدار باز یا امپدانس بالا (High Z) هستند. در حالت پیش فرض کلیه پایه‌ها ورودی و بدون اتصال هستند.
- ✓ Tri-State به معنای سه حالته می‌باشد. یعنی پایه در سه حالت بدون اتصال، اتصال به صفر و اتصال به یک می‌تواند قرار گیرد.

ثبات‌های واحد I/O

در ATMega32 چهار پورت ورودی/خروجی وجود دارد که برای هر یک پین‌هایی خروجی درنظر گرفته‌شده است. هر یک از این ۴ پورت ورودی/خروجی دارای سه ثبات DDR، PIN و PORT هستند که در نتیجه مجموعا 12 ثبات ۸ بیتی برای واحد ورودی/خروجی وجود دارد که همگی در حقیقت درون SRAM می‌باشند که با مقداردهی آنها در برنامه این واحد کنترل می‌شود. ثبات DDR برای تعیین جهت ورودی یا خروجی بودن، ثبات PORT برای تعیین مقاومت PullUp در حالت ورودی بودن پایه و تعیین منطق صفر یا یک در حالت خروجی بودن پایه کاربرد دارد و ثبات PIN نیز برای خواندن منطق اعمال‌شده از بیرون در حالت ورودی می‌باشد. جدول زیر حالت‌های مختلف پایه‌ها را بسته به بیت n ام ثبات DDR و PORT و نیز بیت PUD در ثبات SFIOR نشان می‌دهد. لازم به توضیح است که ثبات SFIOR یک ثبات ۸ بیتی برای تنظیمات خاص پایه‌ها می‌باشد که بیت دوم آن PUD یا PullUp Disable مربوط به غیرفعال کردن کلیه مقاومت‌های PullUp می‌باشد

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

نحوه فعالسازی مقاومت پول آپ

همانطور که در شکل فوق نیز مشاهده می‌کنید، در حالتی که ثبات DDRX.n صفر باشد یعنی پایه ورودی بوده و در صورتی که ثبات PORTX.n را یک کنیم، مقاومت پول آپ داخلی فعالسازی می‌شود. بنابراین ثبات PORTX.n در حالت خروجی بودن پایه، منطق صفر یا یک پایه را مشخص می‌کند و در حالت ورودی بودن پایه فعال بودن یا نبودن مقاومت پول آپ را مشخص می‌کند. بنابراین برای فعال شدن مقاومت پول آپ داخلی باید سه شرط زیر برقرار باشد:

۱. بیت PUD یا Pull Up Disable غیرفعال یا صفر باشد که همیشه برقرار است مگر زمانی که میکروکنترلر ریست باشد یا هنگ کرده باشد.
۲. ثبات DDRX.n مربوطه صفر باشد یعنی پایه ورودی باشد. پس در حالت خروجی مقاومت پول آپ نمی‌تواند فعال شود.
۳. ثبات PORTX.n مربوطه یک باشد. در صورت صفر بودن نیز مقاومت پول آپ غیرفعال است.

نحوه کار با ابزار کدویزارد

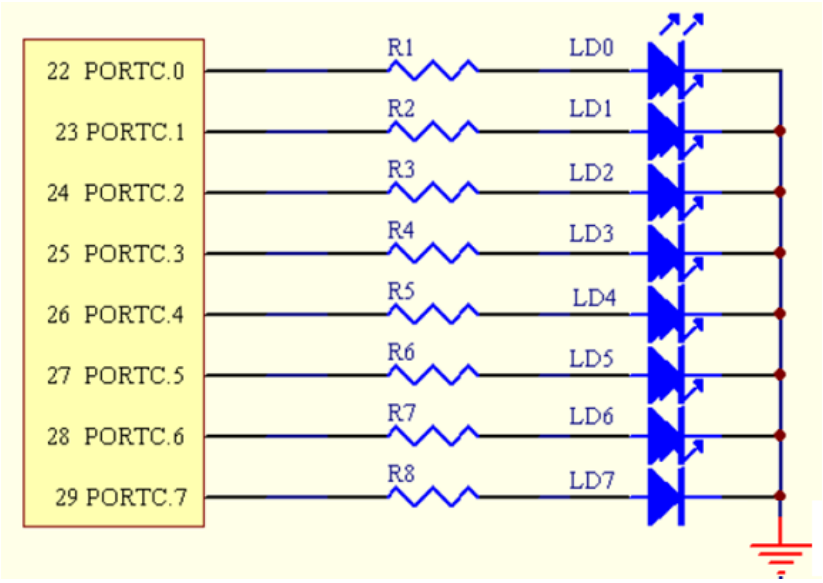
۱- ابتدا به صفحه اصلی نرم‌افزار کدویژن رفته و اگر پروژه‌ای باز است از منوی File و گزینه Close All آن را می‌بندیم.
۲- برای راه‌اندازی کدویزارد در صفحه اصلی نرم‌افزار کدویژن از منوی toolsگزینه CodeWizard AVR را انتخاب کنید. در پنجره بازشده بایستی نوع تراشه را از نظر سری ساخت مشخص کنید.
۳- در این قسمت اولین کاری که باید انجام دهید تنظیم نوع تراشه و فرکانس کاری میکروکنترلر است. فرکانس کاری میکروکنترلر باید همان فرکانسی انتخاب شود که در قسمت Fuse Bits (فیوزبیت‌ها) تنظیم‌شده است. برای میکروکنترلر ATMega32 فرکانس پیش فرض آن 1MHzمی‌باشد.

در مرحله بعد وارد سربرگ Port می‌شویم. در این مرحله باید طبق سخت‌افزار طراحی‌شده پورت‌ها را تنظیم کرد. همانطور که مشاهده می‌شود، پورت‌ها را می‌توان ورودی in یا خروجی out نمود و در صورت ورودی بودن با تغییر T یعنی Tri-State به P می‌توان مقاومت Pull-up داخلی میکروکنترلر را نیز فعال کرد و در صورت خروجی بودن نیز می‌توان با تغییر صفر و یک مقدار اولیه منطق پورت خروجی را تعیین نمود. سربرگ‌های دیگر هر کدام مختص واحدهای دیگر میکروکنترلر است که در صورت لزوم در این مرحله باید تنظیم شود. بعد از پایان تنظیمات، از منوی File گزینه Generate, save and Exit را کلیک کنید. اکنون سه بار باید فایل‌های مربوط به پروژه را با نام ترجیحا یکسان ذخیره کنید. در پنجره اول اسمی برای فایل با پسوند C. وارد می‌کنیم و ذخیره می‌کنیم و همین طور در پنجره‌های بعدی برای project و کدویزارد نام وارد کرده و ذخیره کنید. مشاهده می‌کنید که کد ویزارد بخش اولیه برنامه شما را ایجاد کرده است. اکنون شما می‌توانید شروع به ادامه برنامه‌نویسی با تغییر یا اضافه کردن کدهای ساخته‌شده نمایید.

آزمایش اول: اتصال LED به میکروکنترلر: انتقال داده ورودی به خروجی

الف) مدار شکل زیر را در پروتئوس شبیه‌سازی کنید و میکروکنترلر را با استفاده از کد زیر برنامه‌ریزی کنید. عملکرد مدار را توضیح دهید.

```
1. //Chip type : ATmega16 / ATmega32
2. //*****************************/
3. #include <mega16.h>
4. #include <delay.h>
5. unsigned char i,Num1,Num2,Num3;
6. void main(){
7. DDRD=0xff;
8. PORTB=0xff;
9. DDRB=0x00;
10. while(1){
11. Num1=PINB;
12. Num2=11-PINB;
13. Num3=3;
14. PORTD=0b00000100;
15. for(i=0; i<=Num1; i++){
16. delay_ms(100);
17. }
18. PORTD=0b00000010;
19. for(i=0; i<=Num3; i++){
20. delay_ms(100);
21. }
22. PORTD=0b00000001;
23. for(i=0; i<=Num2; i++)
{24. delay_ms(100);}
 } }
```



ب) مداری طراحی، شبیه‌سازی و پیاده‌سازی کنید که عدد هشت بیتی ورودی پورت A روی هشت عدد LED روی نمایش دهد برای وارد کردن هشت بیت ورودی داده از دیپ‌سوئیچ استفاده کنید.

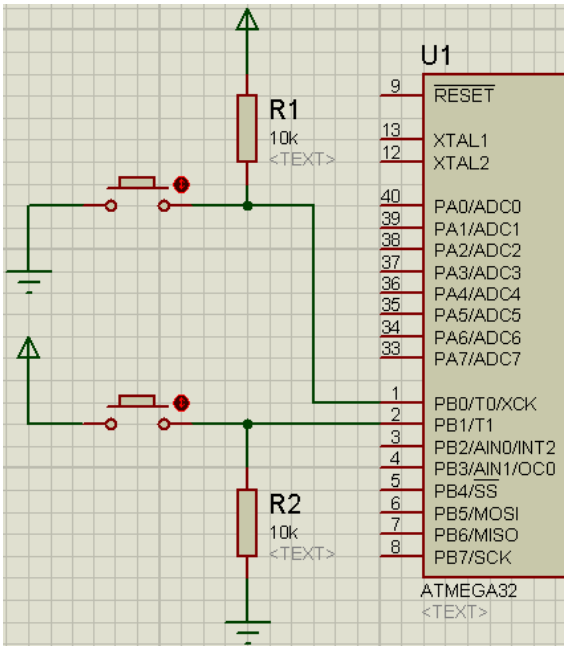
آزمایش دوم: شمارنده و شیفت‌رجیستر: اتصال کلید و LED به میکروکنترلر

الف) مداری طراحی و شبیه‌سازی کنید که یک کلید روی پورت PA0 و هشت عدد LED روی پورت B وجود داشته باشد و با هر بار زدن کلید، LEDهای موجود به صورت یک شیفت رجیستر عمل نماید.

ب) مداری طراحی، شبیه‌سازی و پیاده‌سازی کنید که یک کلید روی پورت PA0 و هشت عدد LED روی پورت B وجود داشته باشد و با هر بار زدن کلید، LEDهای موجود به صورت یک شمارنده بالا شمار عمل نماید.

توضیحات: توسط کلید می‌توان منطق صفر یا یک را به میکروکنترلر وارد کرد و بر اساس آن پردازش را انجام داد. برای خواندن منطق کلید از ثبات PIN استفاده می‌شود. نحوه استفاده از ثبات PIN به این صورت است که اگر کلید زده شد آنگاه کار مورد نظر انجام شود. اتصال کلید به دو صورت Pull-up و PullDown انجام می‌شود که در شکل زیر مشاهده می‌کنید. در کلید pull up در حالتی که کلید زده نشده منطق یک و در حالت فشردن کلید منطق صفر وارد میکروکنترلر می‌شود. معمولا از مقاومت 10k برای این کار استفاده می‌شود.

```
if(PINB.0==0) {...} //Pull-up
if(PINB.1==1) {...} //PulDown
```



اما مشکلی که در کلید وجود دارد، بوجود آمدن لرزش یا bounce در هنگام قطع و وصل کلید است. در زمانی که کاربر کلید را فشار می‌دهد تقریبا 20 میلی‌ثانیه طول می‌کشد تا منطق کلید از صفر به یک (یا بالعکس) ثابت شود تا قبل این به علت وجود جرقه کلید منطق ثابتی ندارد.

در واقع، این مشکل زمانی برای ما مسئله‌ساز می شود که زمانی که کاربر کلید را یکبار فشار می‌دهد و انتظار دارد تا میکروکنترلر متوجه یکبار فشار دادن آن شود اما به علت قرار گرفتن if در درون حلقه نامتناهی while چندین بار شرط PINB.0==0 برقرارشده و کار مورد نظر چندین بار انجام می‌شود. راه حل این مشکل ساده است و آن هم قرار دادن مقداری delay در حلقه if است.

برای حل این مشکل بسته به نوع برنامه یکی از سه روش زیر قابل استفاده است:

توضیح: به محض فشار دادن کلید توسط کاربر شرط if برقرارشده و دستورات مورد نظر اجرا می‌شود سپس به علت ایجاد تاخیر زیاد توسط تابع) delay_ms در اینجا 200 میلی‌ثانیه) با این کار احتمال اینکه زمانی که برنامه در حلقه while بهif می‌رسد و شرط برقرار باشد، کاهش می‌یابد. مزیت این کلید این است که در صورتی که کاربر کلید را فشار داده و نگه دارد تقریبا در هر 200 میلی‌ثانیه یکبار کار مورد نظر صورت می گیرد. عیب این روش نیز این است که هنوز احتمال دارد که زمانی که یکبار کلید زده شود دوبار کار مورد نظر انجام شود.	**if(PINB.0==0) {** **دستورات مربوط به بعد از زدن کلید** **delay_ms(200);** **}**
توضیح: به محض فشار دادن کلید توسط کاربر شرط if برقرارشده و برنامه به مدت 20 میلی‌ثانیه صبر می کند تا منطق کلید ثابت شود و از منطقه bounce عبور کند سپس توسط حلقه while با همان شرط برقراری کلید در این مرحله برنامه تا زمانی که کلید توسط کاربر فشرده‌شده است در حلقه گیر می‌کند و هیچ کاری انجام نمی‌دهد. به محض اینکه کاربر دست خود را بر می‌دارد، شرط برقرار نبوده و خط بعدی یعنی دستورات مربوطه اجرا می‌شود. مزیت این روش این است که در هر بار فشردن کلید برنامه تنها یکبار اجرا می شود. معایب این روش این است که تا زمانی که کاربر کلید را نگه داشته اتفاقی نمی‌افتد و به محض رها کردن کلید کار مورد نظر انجام می‌شود.	**if(PINB.0==0) {** **delay_ms(20);** **while(PINB.0==0);** **دستورات مربوط به بعد از زدن کلید** **}**
توضیح: این کلید به صورت start/stop عمل می‌کند یعنی بار اولی که کاربر کلید را فشار می‌دهد دستورات مربوط به بعد از زدن کلید دائما اجرا می‌شود تا زمانی که کاربر دست خود را از روی کلید رها کرده و دوباره کلید را فشار دهد، در این صورت دستورات دیگر اجرا نمی‌شود. دو متغیر از نوع bit با نام‌های flag و start با مقدار اولیه صفر برای این کلید باید تعریف شود. زمانی که کاربر برای اولین بار کلید را فشار می‌دهد شرط if برقرارشده و flag=1 وstart=1 می‌شود. در این صورت شرط if دوم برقرار بوده و دستورات مربوطه با هر بار چرخش برنامه درون حلقه نامتناهی while یکبار اجرا می‌شود. زمانی که کاربر دست خود را از روی کلید بر می‌دارد و منطق یک وارد میکروکنترلر می‌شودflag=0 شده و برنامه دوباره آماده این می‌شود که کاربر برای بار دوم کلید را فشار دهد. زمانی که کاربر بار دوم کلید را می‌فشارد start=0شده و دستورات مربوطه اجرا نخواهد شد سپس با برداشته شدن دست کاربر از روی کلید، همه چیز به حالت اول بر می‌گردد. این کلید طوری نوشته‌شده است که bounce در آن کمترین تأثیر مخرب ممکن را دارد.	**if((PINB.0==0) && (flag==0))** **{** **flag=1; start=!start; }** **else if (PINB.0 == 1) flag=0;** **if(start){** **دستورات مربوط به بعد از زدن کلید** **}**

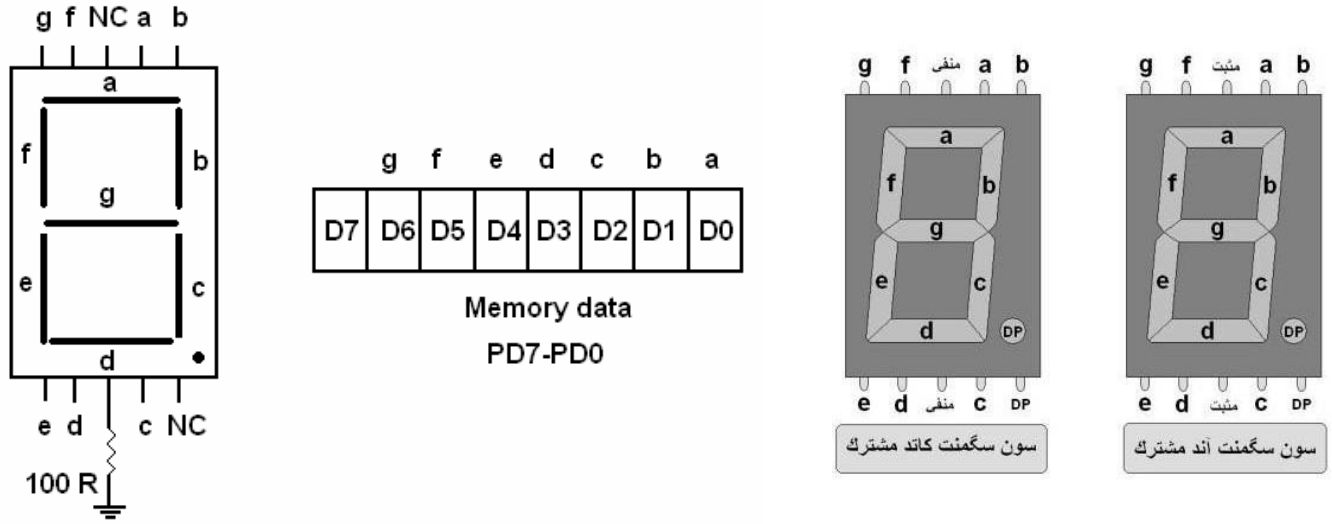
آزمایش سوم: اتصال سون‌سگمنت به میکرو: شمارنده و نمایشگر اعداد یک رقمی و چهار رقمی

الف) یک سون‌سگمنت را به میکروکنترلر متصل کنید. مداری طراحی، شبیه‌سازی و پیاده‌سازی کنید که شمارنده ارقام صفر تا ۹ به صورت چرخشی باشد.

ب) مدار فوق را طوری تغییر دهید که شمارش با زدن کلید ورودی انجام شود.

ج) دو سون‌سگمنت را به میکروکنترلر متصل کنید. مداری طراحی و شبیه‌سازی کنید که شمارنده اعداد دورقمی به صورت چرخشی باشد.

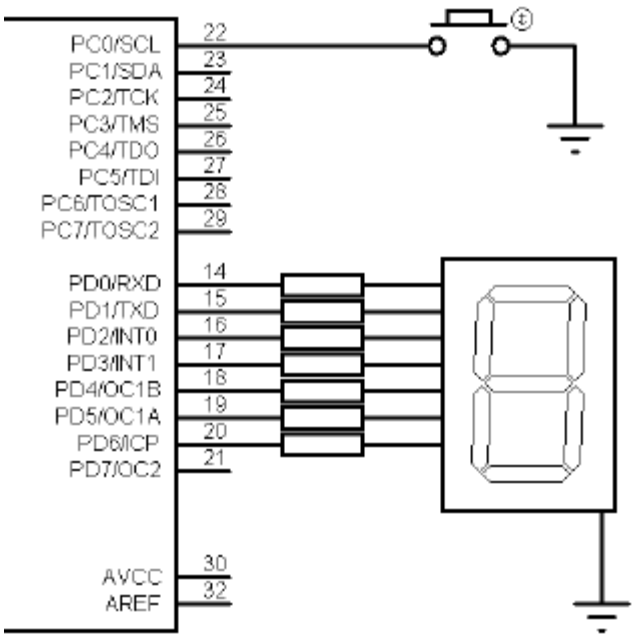
یکی از نمایشگرهای پرکاربرد سون‌سگمنت است که می‌توان توسط آن اعداد و برخی حروف‌ها را نشان داد.



روش‌های متفاوت و مختلفی برای راه‌اندازی سون‌سگمنت توسط AVR وجود دارد. ساده‌ترین روش اتصال سون‌سگمنت به میکروکنترلر استفاده از مداری به شکل زیر است. همانطور که مشاهده می‌شود در این روش از یک پورت به طور کامل استفاده می‌شود. توجه کنید نرم‌افزار پروتئوس بیت هشتم که به digit سون‌سگمنت وصل می‌شود را ندارد. برای نشان دادن اعداد روی سون‌سگمنت کافی است پایه مربوطه را پس از خروجی کردن یک کنیم. در این روش برای راحتی هنگام کار با سون‌سگمنت آرایه زیر را که شامل کد سون‌سگمنت اعداد صفر تا ۹ می‌باشد، تعریف می‌کنیم.

unsigned char seg[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

PORTA=seg[i];

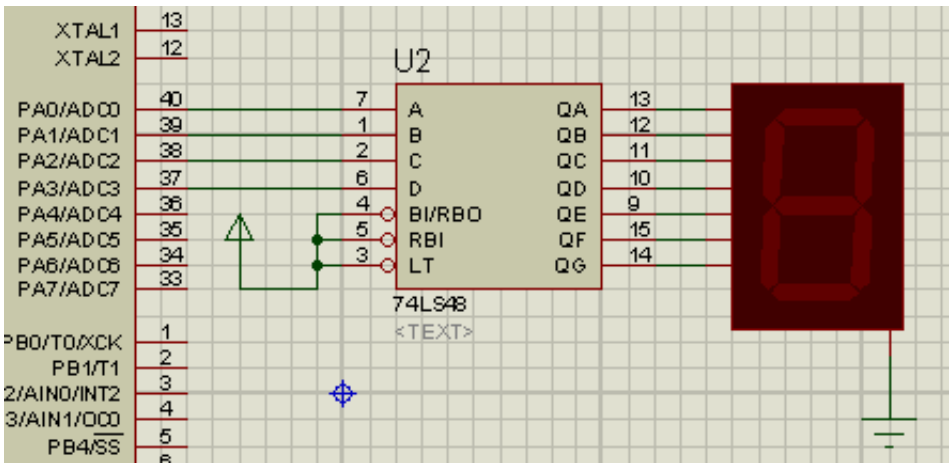


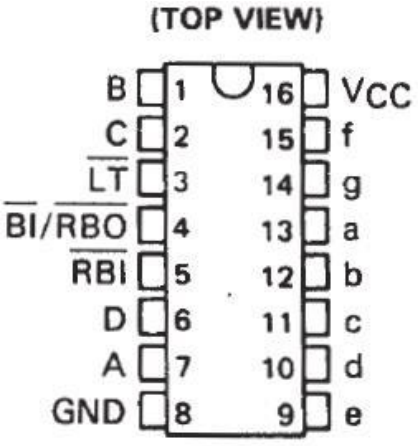
Hexadecimal	seg[6:0]						
	seg[6]	seg[5]	seg[4]	seg[3]	seg[2]	seg[1]	seg[0]
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
b	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
d	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

1 = LED off
0 = LED on

در روش قبل پایه‌های زیادی از میکروکنترلر صرف نمایش تنها یک سون‌سگمنت می‌شود. بجای استفاده از مدار فوق می‌توان از تراشه ۷۴۴۸ استفاده کرد که در این صورت تنها چهار بیت از پورت میکروکنترلر اشغال می‌شود و همچنین به علت تبدیل اتوماتیک کدهای سون‌سگمنت دیگر نیازی به استفاده از آرایه تعریف‌شده در قبل نیست. کار تراشه ۷۴۴۸خواندن یک عدد باینری چهار بیتی بوسیله پایه‌های ورودی‌اش و نمایش آن روی سون‌سگمنت می باشد. این تراشه ۱۶ پایه دارد و نحوه شماره‌بندی پایه‌های آن بصورت زیر می‌باشد.

نکته: در استفاده از تراشه ۷۴۴۸، حتما می‌بایست از سون‌سگمنت کاتد مشترک استفاده کنید. برای استفاده از سون‌سگمنت آند مشترک تراشه دیگری به نام ۷۴۴۷ وجود دارد.

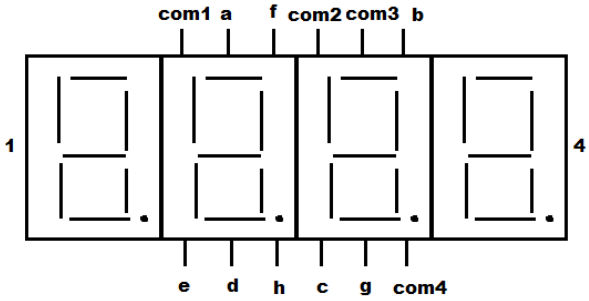




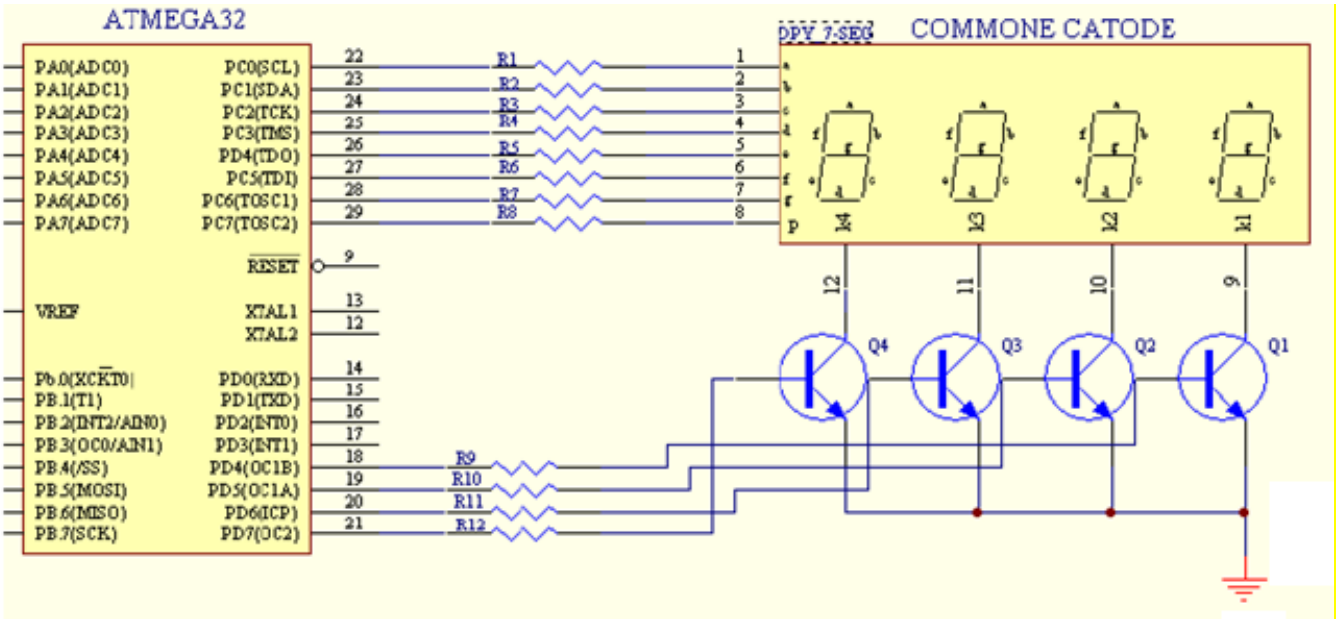
نحوه کار تراشه: ورودی این تراشه شامل چهار پایه ۱، ۲، ۶ و ۷ (که با حروف A ، B، C و D نشان داده می‌شود) می‌باشد. هر کدام از این پایه می‌توانند صفر یا یک باشند. پس عدد ورودی ما به تراشه یک عدد باینری می‌باشد که فقط چهار رقم دارد. از طرفی خروجی این تراشه کدهای مخصوص سون‌سگمنت است که پس از نمایش روی سون‌سگمنت و در مبنای ۱۰ نمایش می‌یابد. البته خروجی این تراشه تنها اعداد صفر تا ۹ می‌تواند باشد.

سون‌سگمنت‌های مالتی‌پلکس‌شده

در برخی طراحی‌ها ممکن است به بیش از یک سون‌سگمنت نیاز باشد. در این طراحی‌ها از سون‌سگمنت مالتی‌پلکس‌شده استفاده می‌شود. ۸ بیت داده برای همه سگمنت‌ها با هم مشترک است. در صورت اتصال پایه‌های Com سون‌سگمنت مربوطه روشن می‌شود.



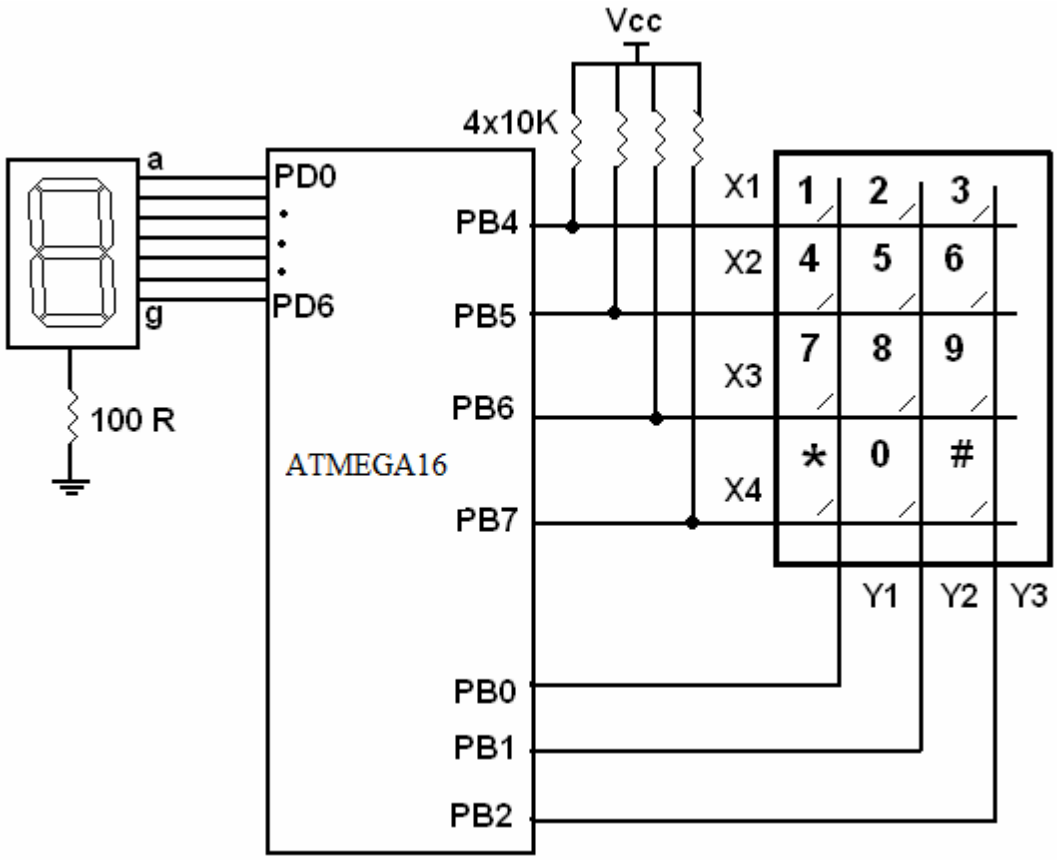
بنابراین مدار مورد نظر در این طراحی به صورت زیر است. وجود ترانزیستور npn در این طراحی بعلت تأمین مناسب جریان برای هر سگمنت است. زیرا طبق داده‌شیت هر پایه میکروکنترلر جریان بیش از 20 میلی آمپر را نمی‌تواند تأمین کند اما در حالتی که چندین سگمنت همزمان روشن است جریانی در حدود 100 میلی آمپر مورد نیاز است و بنابراین اگر ترانزیستور نباشد سگمنت‌ها به خوبی روشن دیده نخواهد شد.



آزمایش چهارم: اتصال صفحه کلید به میکروکنترلر

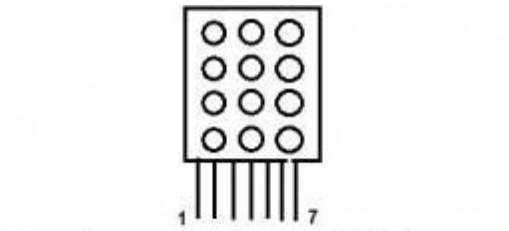
الف) مداری طراحی، شبیه‌سازی و پیاده‌سازی کنید که یک صفحه کلید ماتریسی و یک سون‌سگمنت را به میکروکنترلر متصل کنید که رقم فشرده شده صفحه کلید را روی سون‌سگمنت نشان دهد.

ب) یک صفحه کلید ۴ در ۳ و یک سون‌سگمنت مالتی‌پلکس‌شده mpx4 کاتد مشترک را به میکروکنترلر وصل نمایید. برنامه‌ای بنویسید که با فشار دادن صفحه کلید عدد ۴ رقمی زده شده را از صفحه کلید دریافت و روی سون‌سگمنت نمایش داده شود.



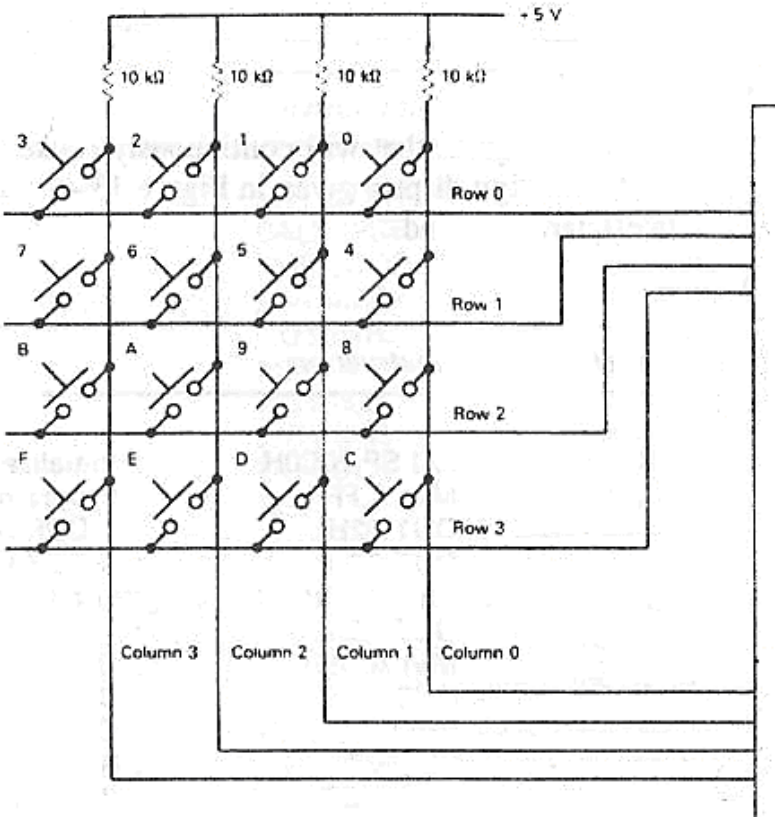
صفحه کلید یک وسیله ورودی پرکاربرد دیگر است که دارای مجموعه‌ای از کلیدها است که به صورت ماتریسی به هم بسته شده‌اند. صفحه کلیدهای پرکاربرد موجود معمولا ۴ سطر و ۳ یا ۴ ستون دارند. برای خواندن صفحه کلید ابتدا همه ستون‌ها را توسط مقاومت به تغذیه مثبت وصل کرده و از همه سطرها و ستون‌ها مستقیم به میکروکنترلر وصل می‌کنیم. سپس سطرها را به عنوان خروجی و ستون‌ها را به عنوان ورودی میکروکنترلر تنظیم می‌کنیم.

بنابراین در حالت عادی همه سطرها یک و ستون‌ها نیز چون توسط مقاومت به Vcc متصل هستند یک خوانده می‌شوند. به عنوان مثال اگر Row0 توسط میکروکنترلر صفر شود و بقیه ردیف‌ها همان یک باشند، هر کدام از دکمه‌های ردیف اول که زده شود، سیگنال صفر مستقیم توسط ستون‌ها به پایه میکروکنترلر منتقل‌شده و توسط میکروکنترلر خوانده می‌شود. بدین ترتیب می‌توان با خواندن منطق ستون‌ها به دکمه زده‌شده پی برد. برای اتصال هرگونه صفحه کلید کافی است همه ستون‌ها از یک طرف به مقاومت pull-up و از طرف دیگر به میکروکنترلر، و همه سطرها نیز مستقیم به میکروکنترلر متصل گردد.

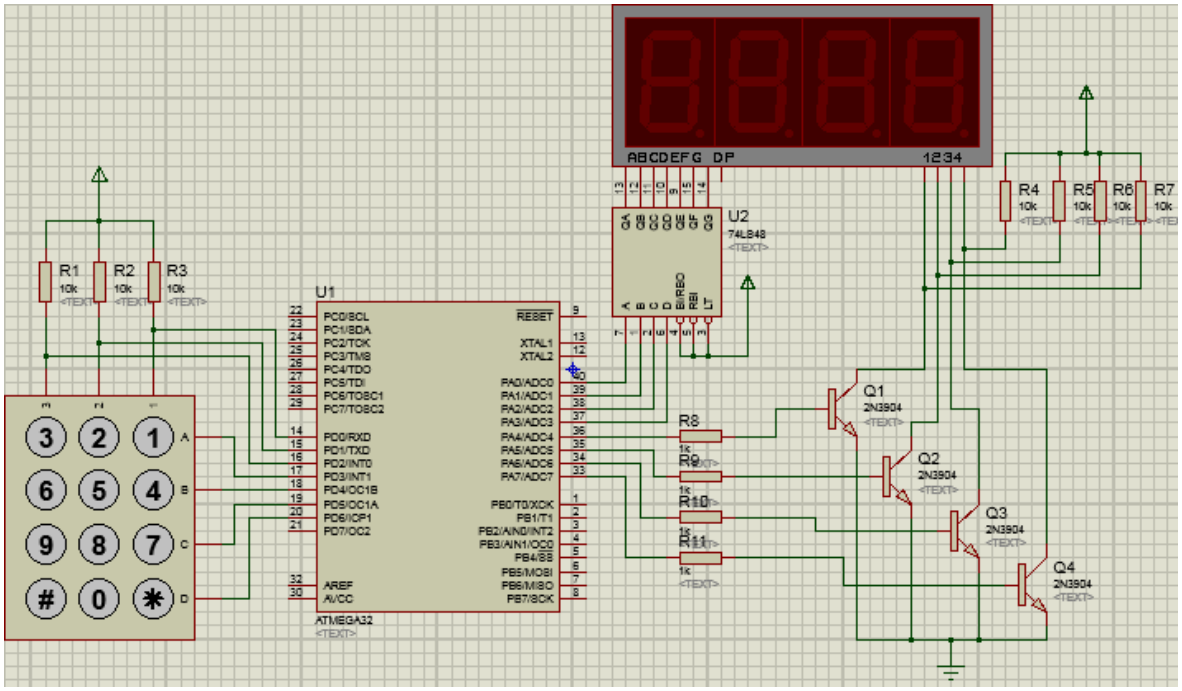


پایه های کی پد به ترتیب از چپ به راست

OUTPUT ARRANGEMENT	
OUTPUT PIN NO.	SYMBOL
1	COL 2
2	ROW 1
3	COL 1
4	ROW 4
5	COL 3
6	ROW 3
7	ROW 2



برای خواندن صفحه کلید ابتدا یک سطر را صفر و سطرهای بعدی را یک می‌کنیم. سپس کمی صبر می‌کنیم (1-2msec) و ستون‌ها را می‌خوانیم. اگر همه ستون‌ها یک باشد یعنی در آن سطر کلیدی زده نشده است، آن سطر را یک و سطر بعدی را صفر می‌کنیم و دوباره ستون‌ها را می‌خوانیم و اگر باز هم یک بود به سراغ سطرهای بعدی می‌رویم تا اینکه تمام سطرها خوانده شود و این کار را با سرعت بالا ادامه می‌دهیم. اما اگر کلیدی زده شود سطر و ستونی که کلید روی آن قرار دارد به هم وصل می‌شوند در نتیجه در هنگام خواندن سطرها متوجه صفر شدن می‌شویم.



آزمایش پنجم: اتصال LCD کاراکتری به میکروکنترلر

الف) مداری طراحی، شبیه‌سازی و پیاده‌سازی کنید که نام و نام خانوادگی شما را روی LCD نمایش دهد.

ب) مدار قسمت الف را طوری تغییر دهید که حروف روی LCD از سمت راست به چپ به صورت چرخشی حرکت کنند.

ج) با استفاده از اتصال یک LCD و یک صفحه کلید ۴×۴ به میکروکنترلر، برنامه‌ای بنویسید که با فشار دادن هر کلید کاراکتر مربوطه روی نمایشگر چاپ شود.

LCD یا صفحه نمایش کاراکتری یکی از پرکاربردترین وسایل خروجی است که می‌توان کاراکترهای قابل چاپ را روی آن نمایش داد. مشخصه اصلیLCD های کاراکتری تعداد سطر و ستون آنها است برای مثالLCD های ۲×۱۶ دارای دو سطر و ۱۶ ستون می‌باشند و در مجموع ۳۲ کاراکتر را می‌توان با آنها نشان داد.

LCDها را می‌توان با هشت خط داده یا چهار خط داده راه‌اندازی کرد، اما نرم‌افزار کدویژن تنها از چهار خط داده پشتیبانی می‌کند که در آن تنها پایه‌هایRS ، R/W، Eو D4 تا D7 به پورت دلخواه از میکروکنترلر متصل می‌شود. تنها تفاوت راه‌اندازی چهار خط داده در این است که داده‌های هشت بیتی LCD به جای یکبار، دو بار از طریق ۴ بیت ارسال می‌شوند. مزیت این راه‌اندازی نیز در این است که اتصال LCD به میکروکنترلر تنها یک پورت از میکروکنترلر را اشغال می‌کند. در نرم‌افزار پروتئوس و به هنگام شبیه‌سازی می‌توان پایه‌های ۱، ۲، ۳، ۷، ۸، ۹ و ۱۰ را بدون اتصال گذاشت، اما در عمل و پیاده‌سازی تمامی پایه‌ها باید متصل باشد.

نکته : پایه VEE برای تنظیم کنتراست نوشته‌های روی LCD می‌باشد. با وصل کردن این پایه به یک مقاومت متغیر می‌توان کیفیت نمایش را تنظیم کرد. البته در عمل می‌توان به جای استفاده از مقاومت متغیر، از اتصال پایه VEE توسط یک مقاومت 1K با زمین مدار، حداکثر کنتراست را برای LCD داشت. درنتیجه کافی است یک مقاومت 1K بین پایه 3 و یک قرار داد.

شماره پایه	نام پایه	عملکرد
1	Vss	زمین ، GND
2	Vcc	تغذیه مثبت ، 5v
3	Vee	تنظیم نور کاراکترها (کنتراست).
4	RS	اگر RS=0 باشد مقدار ورودی به عنوان یک دستور می باشد اما اگر RS=1 باشد مقدار ورودی یک داده برای چاپ شدن است.
5	$R/\overline{W}$	اگر بخواهیم در LCD بنویسیم این پایه باید صفر باشد و اگر بخواهیم از LCD مقداری را بخوانیم باید آن را یک کنیم.
6	E	پس از انجام هر عملیات ارسال یا دریافت باید پایه‌ی E را یکبار صفر و یکبار یک کنیم تا اطلاعات ثبت شوند
7 – 14	$DB_0 \rightarrow DB_7$	مسیر ورود و خروج اطلاعات LCD
15	Anod	تغذیه‌ی مثبت چراغ LCD
16	Katod	تغذیه‌ی منفی چراغ LCD

در سربرگ Alphanumeric LCD **در** کدویزارد ابتدا تیک Enable را بزنید و سپس در قسمت Character/line باید تعداد ستون‌ها را وارد نمایید و در قسمت بعدی باید نام پایه‌های متصل به LCD را مشخص کنید. با این کار در برنامه هدر فایلی با نام alcd.h و تابع راه‌انداز LCD به صورت lcd_init(16)به برنامه اضافه می‌شود.با اضافه شدن این هدر فایل از توابع زیر برای کار با LCD در برنامه دلخواه می‌توان استفاده کرد.

توابع LCD کاراکتری

عملکرد	توضیحات	نام تابع
پاک کردن تمامLCD	این تابع lcd را پاک کرده و مکان نما را در سطر و ستون صفر قرار می‌دهد.	lcd_clear();
رفتن به ستون x و سطر y ام	مکان نمای LCD را در سطر x و ستون y قرار می‌دهد و باید به جای x و y عدد سطر و ستون مورد نظر جایگذاری شود.	lcd_gotoxy(x , y);
چاپ یک کاراکتر		lcd_putchar('*char*');
چاپ یک رشته	ورودی تابع lcd_putsf یک رشته ثابت مانند "IRAN" است	lcd_putsf ("String");
چاپ یک متغیر رشته ای	ورودی تابع lcd_puts یک متغیر رشته‌ای از نوع آرایه‌ای می‌تواند باشد.	lcd_puts (string variable);
مقداردهی به یک متغیر رشته‌ای	برای استفاده از این تابع که در کتابخانهstdio.h می‌باشد.	sprintf(c,"temp=%d",i);
راه‌اندازی اولیه LCD	۱۶ تعداد ستون هاست	lcd_init(16);

فرمت متغیرهای کاراکتری ارسالی:

نوع اطلاعات ارسالی	کاراکتر
یک تک کاراکتر	c%
عدد صحیح علامت دار در مبنای ۱۰	d%
عدد صحیح علامت دار در مبنای ۱۰	i%
نمایش عدد ممیز شناور به صورت علمی	e%
نمایش عدد ممیز شناور به صورت علمی	E%
عدد اعشاری	f%
عبارت رشته ای واقع در حافظهSRAM	s%
عدد صحیح بدون علامت در مبنای ۱۰	u%
به فرم هگزا دسیمال با حروف بزرگ	X%
به فرم هگزا دسیمال با حروف کوچک	x%
عبارت رشته ای واقع در حافظهFLASH	P%
نمایش علامت ٪	%%